CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Multi-Level Preemption in TSN: Feasibility and Requirements Analysis

Mubarak Ojewale*
Patrick Meumeu Yomsi*
Borislav Nikolic

*CISTER Research Centre

CISTER-TR-200310

2020/05/19

Multi-Level Preemption in TSN: Feasibility and Requirements Analysis

Mubarak Ojewale*, Patrick Meumeu Yomsi*, Borislav Nikolic

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

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, borni@isep.ipp.pt

https://www.cister-labs.pt

Abstract

To overcome the limitation of strictly non-preemptive frame transmission in Ethernet networks, the IEEE802.1Qbu standard was introduced. This standard specifies a one-level frame preemption paradigm wherein, depending on their priority levels, frames are grouped into two classes: namely, the ``express frames" and the ``preemptable frames". These two classes are given with the interpretation that (1) only express frames can preempt preemptable frames; and (2) two frames belonging to the same class cannot preempt each other. While this approach partially solves the problem, some preemptable frames can still suffer long blocking periods, irrespective of their individual priority levels. Indeed, there are frames that do not fall into the express frames class, but nevertheless have firm timing requirements that can only be met if they can benefit from preempting lower priority frames. To ameliorate the condition of such frames, we propose a multi-level preemption paradigm. Specifically, we expose the limitations of the one-level preemption approach experimentally; and we present the feasibility and implementation requirements of the multi-level preemption scheme in details.

Multi-Level Preemption in TSN: Feasibility and Requirements Analysis

Mubarak Adetunji Ojewale[1], Patrick Meumeu Yomsi[1], and Borislav Nikolić[2]

[1]CISTER Research Centre, ISEP, Polytechnic Institute of Porto, Portugal
[2]Institute of Computer and Network Engineering, TU Braunschweig, Germany
Emails: [1]{mkaoe, pmy}@isep.ipp.pt, [2]bnikolic@ida.ing.tu-bs.de

***Abstract*—To overcome the limitation of strictly non-preemptive frame transmission in Ethernet networks, the IEEE 802.1Qbu standard was introduced. This standard specifies a one-level frame preemption paradigm wherein, depending on their priority levels, frames are grouped into two categories: namely, the "express frames" and the "preemptable frames". These two categories are given with the interpretation that (1) only express frames can preempt preemptable frames; and (2) two frames belonging to the same category cannot preempt each other. While this approach partially solves the problem, some preemptable frames can still suffer long blocking periods, irrespective of their individual priority levels. Indeed, there are frames that do not fall into the express frames category, but nevertheless have firm timing requirements that can only be met if they can benefit from preempting lower priority frames. To ameliorate the condition of such frames, we propose a *multi-level preemption paradigm*. Specifically, we expose the limitations of the one-level preemption approach experimentally; and we present the feasibility and implementation requirements of the multi-level preemption scheme in details.**

I. Introduction

The legacy Ethernet standards were originally designed targeting only non real-time applications and desirable features like (1) *frame preemption* [1]; (2) *global time synchronization* [2]; (3) frame replication and elimination for reliability [3]; (4) path control and reservation [4]; and finally (5) centralised network configuration [5] among other features were missing in order to make it also suitable for real-time applications. A tremendous amount of work has been achieved in this direction over the past few years and several modifications and/or amendments have been made to the standards. Ethernet has successfully been patched and augmented with all the aforementioned features [6], thus leading to a set of *updated standards*, referred to as *Time-Sensitive Networking* (TSN) [7]. Before these standards, frames transmission were thought of as atomic operations, i.e., they were transmitted over links of the network by following a fully non-preemptive policy, irrespective of the individual frame priorities [8], [9]. Consequently, the transmission of a low-priority frame could prevent and/or block that of any high-priority frame for a non-negligible period of time. To illustrate this claim, assuming an IEEE 802.1 network operating at $1Gb/s$, the size of a low-priority frame can perfectly be as large as the maximum valid Ethernet frame, i.e., 1542 bytes [10]. This means a blocking of $12.336\mu s$, which is prohibitive for real-time domains like the industrial automation, where latency requirements can be as stringent as $5\mu s$ at each node [11]. To circumvent and/or mitigate this limitation, the IEEE 802.1Qbu [1] and IEEE 802.3br [12] standards have defined a *one-level preemption* scheme, which is specified as follows. Two Medium Access Control (MAC) service interfaces referred to as the *express MAC interface* (eMAC); and the *preemptable MAC interface* (pMAC) are implemented. Frames assigned to the eMAC and pMAC service interfaces are called "*express frames*" and "*preemptable frames*", respectively. Then, only express frames can preempt preemptable frames. Finally, frames transmitted through the same service interface cannot preempt each other and are served in a first-in-first-out (FIFO) manner. Fig. 1 illustrates such a scenario at a given output node.

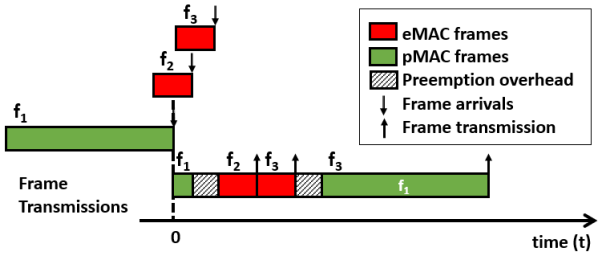


Fig. 1: Illustration of the one-level preemption scheme.

In this figure, three frames (f_1, f_2 and f_3) are considered for transmission. Frames f_2 and f_3 are eMAC frames (in red colored box), whereas f_1 is a pMAC frame (in green colored box). The downward arrows define the frame arrivals and upward arrows define the time instant at which the transmission of each frame is completed. Frame f_1 arrives first (at time 0) and starts its transmission. However, upon the arrival of f_2, f_1 is preempted and f_2 is transmitted. Now during the transmission of f_2, frame f_3 arrives, but it cannot start its transmission because f_2, which is in the same service category, is being transmitted. The transmission of f_3 starts only after the transmission of f_2 is completed. Finally, as all the eMAC frames, which arrived after the preemption of f_1 have been transmitted, f_1 can resume its transmission.

As shown in the figure, preemption favors a prompt service of all eMAC frames, but this comes at the cost of some overheads, unfortunately. The overheads stem from the fact that fragments of split frames have to form valid Ethernet

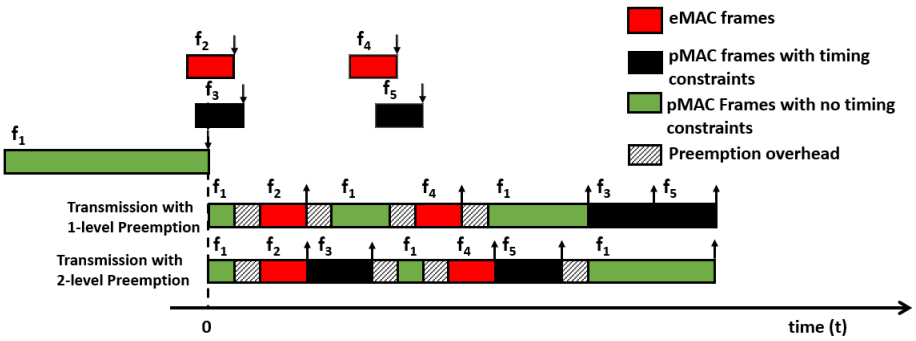


Fig. 2: Frame transmissions under 1-level preemption vs. 2-level preemption.

frames. As a matter of fact, pieces of information (e.g., the fragment count, an error correction code to detect whether all fragments have correctly been transmitted, etc.) must be added to the fragments of the preempted frame – here, f_1 – so that the network nodes can correctly transmit and receive all of them. In addition, these overheads are used to correctly reconstruct the original frame at each receiver node. Note that the "receiver node" herein includes all downstream switches and the final destination node of a frame. We will keep this convention throughout this paper.

II. Motivation for promoting a multi-level preemption paradigm

From the discussion conducted in Section I, it follows that the one-level preemption scheme greatly improves the responsiveness of eMAC frames, but exposes serious limitations and poor performance, when it comes to the transmission of pMAC frames. This may nullify the benefits the standards sought to bring about in the first place. Indeed, there are frames that cannot be classified as eMAC frames, but have firm timing requirements. These frames should not be blocked for prohibitively long time periods by lower priority frames in order not to jeopardize the schedulability of the entire system. On another front, the current specification of the one-level preemption operation in the standards does not allow these frames to leverage the basic benefits of enabling preemption, unfortunately. Fig. 2 illustrates the case. In this figure, five frames (f_1, f_2, f_3, f_4, and f_5) are considered for transmission. Frames f_2 and f_4 are eMAC frames (in red colored boxes) with stringent timing constraints; frames f_3 and f_5 are pMAC frames (in black colored box) with firm timing constraints; and finally f_1 is a pMAC frame with no timing constraint at all (in green colored box). In this settings, we assume that all frames with stringent and firm timing requirements must be transmitted before the arrival of the next one of the same type. We also assume that f_3 and f_5 have a higher priority than f_1. In this scenario, frame f_1 arrives first (at time 0) and starts its transmission, followed by frame f_2, and finally frame f_3.

▷ **In the one-level preemption paradigm**: upon the arrival of f_2, frame f_1 is preempted since f_2 is an eMAC frame. Then, upon the completion of this frame, f_1 resumes its transmission. This holds true despite the earlier arrival of f_3, which has a higher priority than f_1. This is due to the one-level of preemption scheme as pMAC frames cannot preempt each other, thus exhibiting a *priority inversion* problem! As a result, f_3 and f_5 are delayed until the completion of f_1, thereby preventing f_3 from meeting its timing requirement.

▷ **In the 2-level preemption transmission paradigm:** The aforementioned issue is circumvented. Here, frames f_3 and f_5 received better services. Frame f_3 is transmitted immediately after the completion of f_2, and f_5 is also transmitted immediately after the completion of f_4. This results in f_3 being able to meet its timing requirement, as well as a noticeable improvement in the completion time of both f_3 and f_5.

From this example, it follows that pMAC frames with firm timing requirements suffer diminished responsiveness under the 1-level preemption paradigm, but the situation improves already under a 2-level preemption paradigm. Since this type of frames are not uncommon in real-time applications, this represents a clear motivation for exploring, not only a 2-preemption, but a multi-level preemption scheme in TSN to draw the maximum benefit from this approach.

This research. In this paper, we advocate for a multi-level preemption paradigm in Ethernet in order to circumvent the limitations encountered in the one-level preemption scheme. Our contribution is twofold. First, we propose a new framework wherein the non-preemptive transmission constraints among non-express frames is relaxed. Specifically, we allow non-express preemptable frames with stringent timing requirements to preempt frames that are free from any timing requirements (as shown in Fig. 2). Then, we describe the operation dynamics of our approach and the actual implementation recommendations for its feasibility.

Paper organization. The rest of the paper is organized as follows. Section III discusses relevant related works while the system model is presented in Section IV. Section V explains the current implementation of the one-level preemption mechanism in Ethernet networks while Section VI details the feasibility and implementation requirements of the multi-level preemption scheme. Section VII presents experimental results to demonstrate the weaknesses of the existing one-level preemption scheme, which the multi-level preemption approach can efficiently resolve. Finally, Section VIII concludes the paper and presents future research directions.

III. Related Work

Most of the literature on frame preemption in Ethernet have focused on the impact of preemption on the end-to-end transmission delay. To this end, simulation is commonly used (see [9], [13]–[15] for more details). Most of the authors relying on this methodology recognize the effectiveness of frame preemption in Ethernet and have reached the conclusion that it allows system designers to drastically reduce the transmission delays of express frames, while the performance of preemptable frames does not degrade much. Kim et al. [14] showed that preemptive switched Ethernet provides better performance than the standard IEEE 802.1Q/p protocol. This holds true especially when real-time and non-real time frames are transmitted over the same network. In the same vein, Thiele and Ernst [9] observed that the end-to-end delays of express frames under preemptive Ethernet are very close to those of IEEE 802.1Qbv [2], thus suggesting the standard Ethernet with enabled preemption as a viable alternative to IEEE 802.1Qbv.

It is well established in the research community that simulation is neither an exhaustive, nor a rigorous means of evaluating the performance of a system. This is due to the fact that it does not guarantee the occurrence of the instance that would produce the worst-case scenario. Consequently, despite the vast quantitative performance results obtained through this technique, more sophisticated approaches have been developed to provide guarantees on end-to-end delays of Ethernet frames. In this direction, Thiele and Ernst [9] presented a Compositional Performance Analysis (CPA) for the one-level preemption scheme under both the standard Ethernet and TSN. Assuming a preemption-enabled network, Zhou et al. [15] presented a VHDL design layout for the transmission and reception processes as well as an FPGA-based hardware implementation of the sender and receiver nodes. By using an FPGA-based implementation, Hotta et al. [10] also provided a quantitative evaluation of the performance gains associated to frame preemption. The measurement results show that the maximum latency of express frames could be significantly reduced (from $27.57\mu s$ to $2.46\mu s$ @ 1Gb/s). Other works like [8], [13] have pointed out a few challenges that could result from preemption operations for low-priority frames, including: (i) starvation; (ii) buffer overflow (when there are more preempted frame fragments to be stored than the buffer size of switches/nodes); and (iii) possible distortion in the order of arrival of express frames.

To the best of our knowledge, no work has investigated the feasibility of multiple preemption levels in Ethernet networks, and especially for preemptable traffic. In this work, we fill this gap by investigating the feasibility of an additional MAC Merger sublayer interface to support more preemption levels. We assume 3 frame classes and 2 preemption levels to investigate the feasibility of multi-level preemption in Ethernet.

IV. System Model

We consider a network traffic consisting of a set $\mathcal{F} \stackrel{\text{def}}{=} \{f^1, f^2, \ldots, f^n\}$ of $n \geq 1$ flows partitioned into two service categories: the *express traffic* and the *preemptable traffic*.

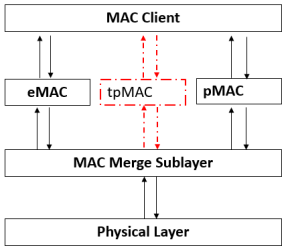


Fig. 3: The MAC merge sublayer managing service interfaces.

Each flow $f^i \stackrel{\text{def}}{=} (src^i, dst^i, C^i, D^i, T^i) \in \mathcal{F}$ consists of a potentially infinite number of frames f^i_k (with $k \geq 1$) and is characterized by a 5-tuple, where: (1) src^i is the source node; (2) dst^i is the destination node; (3) C^i is the maximum size of each frame f^i_k; (4) $D^i \leq T_i$ defines the time window within which each frame f^i_k must reach dst^i; and finally (5) T^i is the minimum timespan between any two consecutive frame transmissions of f^i. All frames generated by a flow inherit its priority and the smaller the superscript of a flow, the higher its priority. We assume that the preemptable service category is further partitioned (for proof of concept) into two sub-categories referred to as the *time-sensitive preemptable traffic* (tpflows), with firm timing requirements; and *best effort preemptable traffic* (bpflows), with no timing requirements. Also, we assume that every express flow has a higher priority than all preemptable flows and the following rules apply:

- R_1– Every express flow can preempt all preemptable flows;
- R_2– Every tpflow can preempt all bpflows;
- R_3– Flows belonging to the same service category cannot preempt each other and are transmitted in a FIFO manner.

V. Current Implementation of Preemption Mechanism in Ethernet Networks

The IEEE 802.1Qbu and 803.3br standards [1], [12] describe not only the preemption operation, but also the hardware modifications required on the switches/bridges to support preemption. Preemption occurs at the MAC Merge sublayer, which is between the physical and the MAC layers (see Fig. 3). Frames at this sublayer are referred to as *mPackets* [12]. Before each mPacket transmission, the sublayer verifies if the next switch/node supports preemption by performing a verification operation (see [12], page 42 for details). Preemption capability is enabled at the sender node only after the verification confirms that it is supported by the receiver node (i.e., the downstream node) [16]. When this is the case, additional information are added to the mPacket header, describing its preemption characteristics. Then, the sublayer has the capability to preempt any preemptable mPacket currently being transmitted and can also prevent a new one from starting its transmission [12]. In addition, it is important to preserve the Ethernet frame format when mPackets are preempted. The IEEE 802.3br Standard ensures this feature by defining mPacket formats in a preemption enabled environment.

Fig. 4 shows that an express frame (see Fig. 4b) differs from a typical MAC frame (see Fig. 4a) by only one octet, referred

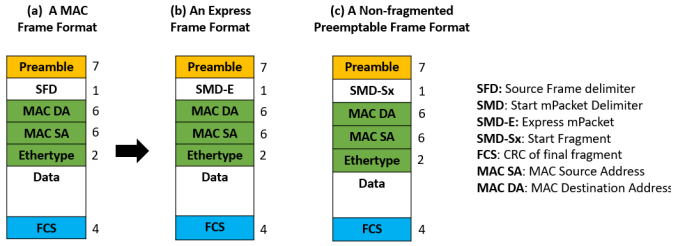


Fig. 4: Frame format as in IEEE 802.3 Standards (the numbers are in bytes and represent the size of each field).

to as "Start Frame Delimiter" (SFD). This octet is replaced by the "*Start mPacket Delimiter-Express*" (SMD-E). In practice the SFD and SMD-E have the same value, though. On the other hand, a preemptable frame that has not been preempted (see Fig. 4c) also differs from a typical MAC frame by only a single octet, here the SFD is replaced by the "*Start mPacket Delimiter Start Fragment*" (SMD-Sx).

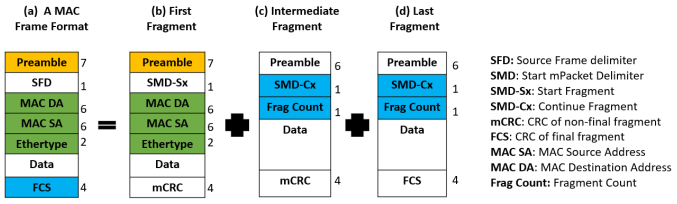


Fig. 5: Frame fragment format as in IEEE 802.3 Standards (the numbers are in bytes and represent the size of each field).

Now, assuming a frame that has been preempted, it is worth noticing that the first fragment remains almost the same as the original frame. There are only two differences: (1) the *size*, which is smaller (because the original frame has been divided into fragments); and (2) the *error checking code* (FCS), which is replaced by a newly generated mPacket error checking code (mCRC) (see Fig. 5b). The header of all the other fragment(s) consist of three components: (i) a preamble; (ii) a "Start mPacket Delimiter for Continuation fragment" (SMD-Cx) and (iii) a "Frag_count", which is used to monitor the correct order of fragment arrivals and detect missing fragments (see Fig. 5c). All the fragments are appended with the same mCRC except the last fragment which ends with the FCS of the original preempted frame (see Fig. 5d). At the receiver node, a Medium Independent Interface (xMII) inspects the SMD for each frame upon arrival and the value of the SMD indicates whether the frame is an express or a preemptable frame [17]. Express frames (i.e., frames containing SMD-E) are processed by an Express Filter, whereas preemptable frames are processed by a "Receive processing" construct. This specific Receive processing is responsible for guaranteeing that all the fragments of a preempted frame have been received completely and in correct order. To this end, it uses both the "mCRC" and the "frag_count" values. The frame transmission of any preempted frame is guaranteed to complete because each fragment has an mCRC computed based on the segment its original frame contained. This means that the reception of a sequence of fragments of a preempted frame is completed as soon as the last four octets of the mPacket does not match the mCRC [12].

The current frame preemption specification does not allow any form of *padding* to be added to any fragment. That is, the data portion of any fragment cannot be augmented to meet the minimum Ethernet frame size requirement, which is 84 bytes. To enforce this, the standards command any preemptable frame not to be preempted until the following two requirements are fulfilled: (1) the size of the fragment that is currently being transmitted is at least 84 bytes; and (2) the remaining fragment, which results from the occurrence of a preemption, also meets the minimum frame size requirement. With these two requirements in mind, the longest non-preemptable Ethernet frame fragment is 143 bytes long (see [9], Lemma 1 for a detailed proof). This implies that any express frame can be blocked for at most $1.144\mu s$ and $11.44\mu s$, assuming a $1Gb$ and $100Mb$ speed Ethernet, respectively. We recall that without preemption, the blocking was $12\mu s$, assuming a $1Gb$ speed Ethernet. Roughly speaking, this means a reduction of 90.5%, when preemption is enabled. On another front, the total overhead induced by the occurrence of each preemption is 12 bytes (i.e., 6-byte preamble, 1-byte SMD-Cx, 1-byte FCnt; and finally 4-byte mCRC). The *Inter Frame Gap* (IFG) between two consecutive transmission has to be accounted for before the next frame/fragment is transmitted. According to the standards, the size of each IFG is equal to the amount it takes to transmit 12 bytes of data. This brings the total overhead associated to the occurrence of each preemption to 24 bytes (i.e., $0.19\mu s$, assuming $1Gb/s$ speed).

VI. On the Feasibility and Requirements of the Multi-Level Preemption Paradigm

To achieve multi-level preemption in TSN, it is important that switch nodes are capable of identifying more than two service categories of frames in first place. That is, each switch node where preemption is enabled should distinguish frames belonging to a different category than the traditional eMAC and pMAC categories. To this end, we must enrich the definition set of the SMD octet, which is used to determine whether a frame is preemptable or not at the MAC merge sublayer. By doing so, it will be possible to further partition the set of preemptable frames. In their current specifications, the standards define eleven different SMD values [12]. In addition to allow us distinguish between eMAC and pMAC frames, these values also describe the verification frames (i.e., the frames sent to determine if the next node supports preemption). On another front, the current specification of the preemption operation does not allow frames belonging to the same category to preempt each other [12]. For the sake of interoperability, it is important to keep this convention, as discussed later in this paper. To this end, an additional MAC Merge sublayer interface is needed to support each additional preemption level so that frames of each service category are assigned to a unique MAC Merge sublayer interface.

The "*Transmit Processing*" function is responsible for the frame transmission at the MAC Merge sublayer interface. The current verification procedure in this function can only check

if preemption is supported by the receiver node. It does this by sending a "request frame" to the receiver node to inquire if it supports preemption(s). The receiver node, in return, sends a response to confirm that it supports preemption(s) or otherwise. In practice, this information is interpreted by the sending node based on the SMD value of the response frame. The transmission of any preemptable frame starts only after this verification process. From this discussion, both the "*Transmit Processing*" and the "*Receive Processing*" functions would require modifications before multi-level preemption can be enabled. For the transmission, we would need the SMD values to also inform the sending node of the preemption level the receiver node supports.

A. Modifications in the transmission mechanism of frames

Fig. 6 illustrates a *Transmit Processing* state diagram to support an additional level of preemption, which extends the basic 1-level preemption scheme (see [12], page 50). In this figure, all newly proposed transitions and/or states are indicated with red dotted lines. All labels, functions and variables are as defined in the standard (see [12], pages 45–48). The transmission process is triggered when a frame reaches the INIT_TX_PROC state. Below we provide the reader with a description of the state diagram operation.

▷ **On the transmission of express frames**. When an express frame reaches the "Transmit Processing" function, i.e., at the IDLE_TX_PROC, it is identified as such and then it transits to the EXPRESS_TX state, which is responsible for transmitting it in a non preemptive manner. Upon completion, the function transits to the E_TX_COMPLETE state, where it either returns to the idle state (IDLE_TX_PROC) or resumes the transmission of a pending preempted frame.

▷ **On the transmission of preemptable frames**. In contrast to the express frame transmission process, when a preemptable frame reaches the IDLE_TX_PROC state, the Transmit Processing function first checks whether the receiver node has preemption capabilities enabled and transits back to the IDLE_TX_PROC state (see connector "C"). If the answer is positive, the function transits to the START_PREEAMBLE state, which triggers the transmission in a preemptable manner. Note that the transmission can be interrupted only if an express frame arrives and the preemptable transmission has reached a feasible preemption point. In case a preemption occurs, an MCRC value is computed for the preemptable frame fragment (TX_MCRC) and the function transits to a waiting state (RESUME_WAIT). All the pending express frames are then transmitted (see connector "B") and the transmission of the preemptable resumes only afterwards. Upon completion or if preemption did not occur during frame transmission, then the function transits back to state IDLE_TX_PROC. For a more detailed information about each state, the reader is referred to the IEEE 802.3br Standard (see [12], pages 48–52).

▷ **On the proposed modifications for enabling an additional preemption level**. An additional preemption level does not require any alteration in the transmission process of express frames since we are concerned with the transmission of preemptable frames with firm timing requirements. For such frames, new states have to be added. The changes are illustrated with red dotted lines in Figure 6. In this figure, we define two preemptable MAC Merge Sublayer interfaces, referred to as p_1 and p_2, and enforce the following rules.

- Any p_1 frame can preempt any p_2 frame, but the reverse is not true.
- Upon preemption, any p_2 frame can resume its transmission only if all pending express and p_1 frames/fragments have completed their transmission.

With this new preemption level specification, in addition to checking whether the receiver node has preemption capabilities, we must check its preemption level (0, 1 or 2). When a preemptable frame reaches the IDLE_TX_PROC state, then the function transits (i) to the START_PREEAMBLE state if it is a p_1 frame; or (ii) to the newly defined START_PREEAMBLE_2 state if it is a p_2-frame. In this context, each p_1 frame is transmitted in a similar manner as a traditional pMAC frame, whereas each p_2 frame is transmitted such that it can be preempted by both express and p_1 frames. When a preemption occurs, the p_2 frame resumes and completes its transmission only after the completion of all pending express and p_1 frames/fragments.

B. Modifications in the reception mechanism of frames

In a similar manner, Fig. 7, which is an extension of the "*Receive Processing*" state diagram (see [12], page 51), illustrates the newly proposed transitions and/or states in red dotted lines. Again, all labels, functions and variables are as defined in the standard (see [12], page 45–48). The reception process is triggered when a frame reaches CHECK_FOR_START.

▷ **On the reception of express frames**. When an express frame reaches the CHECK_FOR_START state, it is identified as such and then it transits to the EXPRESS state, which is responsible for receiving it in a non preemptive manner. Upon completion, the function transits to the IDLE_RX_PROC state.

▷ **On the reception of preemptable frames**. In contrast to express frames, when a preemptable frame reaches CHECK_FOR_START, the modified "Receive Processing" function transits to the REPLACE_SFD where the SFD of the frame is replaced by a newly computed SMD value. Afterwards, the function transits to P_RECEIVE_DATA, which triggers the reception in a preemptable manner. Note that the reception can be interrupted only if an express frame arrives and the preemptable transmission has reached a feasible preemption point. In case a preemption occurs, the reception is suspended and the function transits to P_WAIT_FOR_DV_FALSE, where it receives the preempting express frame(s). Upon complete reception of these frames, the function transits to P_WAIT_FOR_RESUME, which resumes the reception of the preempted frame. Upon completion or if preemption did not occur during the frame reception, the function transits back to IDLE_RX_PROC.

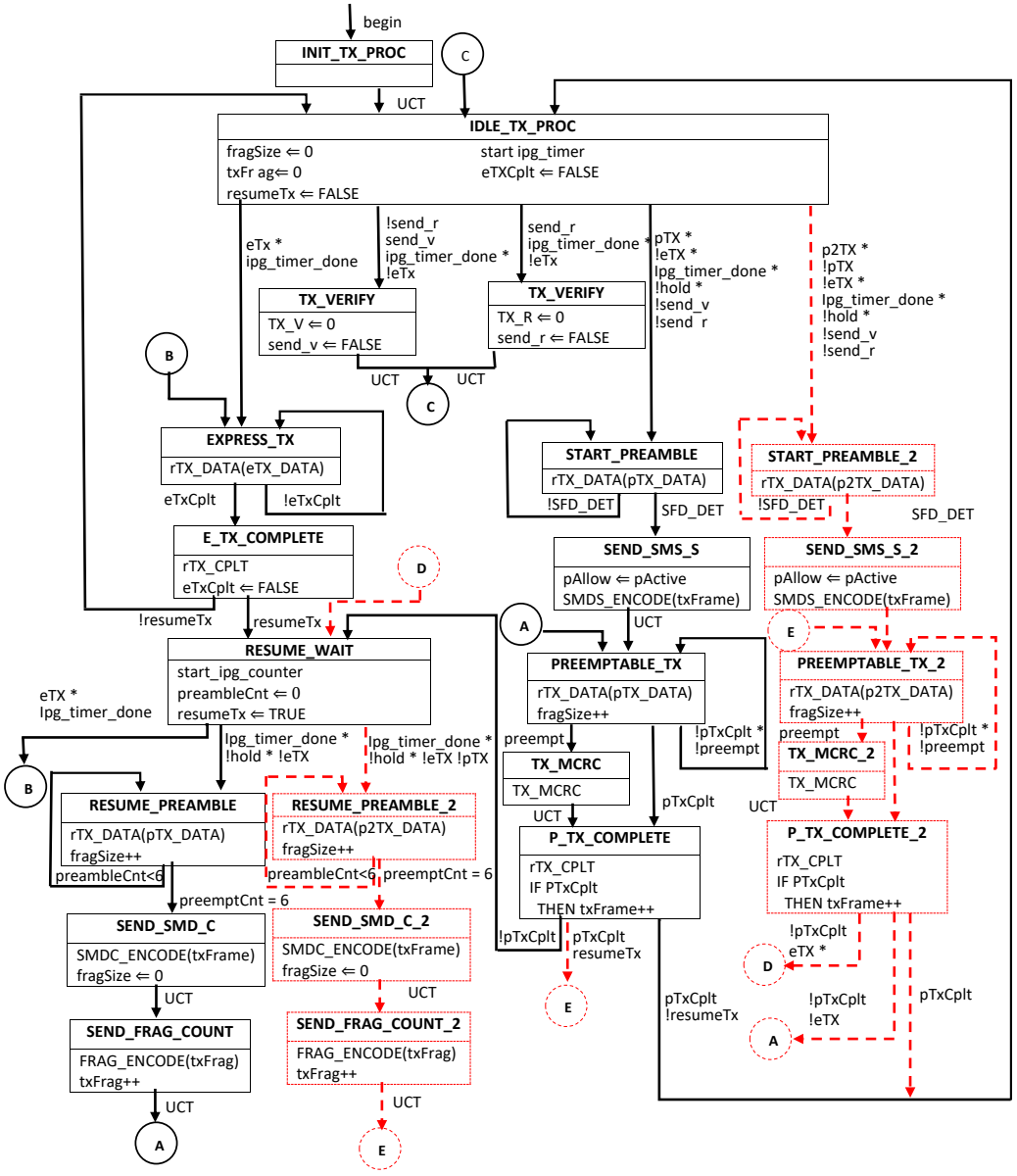


Fig. 6: Modified "Transmit Processing" state diagram for two-level preemption. Added features are represented in red color.

▷ **On the proposed modifications for enabling an additional preemption level**. Similar to the transmission process, an additional preemption level does not require any alteration in the reception of express frames. For the reception of preemptable frames with firm timing requirements, new states have to be added. The changes are illustrated with red dotted lines in Fig. 7. In this figure, we also define two preemptable MAC Merge Sublayer interfaces referred to as p_1 and p_2 and enforce the following rules.

- Any p_1 frame can preempt any p_2 frame, but the reverse is not true.
- Upon preemption, any p_2 frame can resume its reception only if the reception of all pending express and p_1 frames/fragments has completed.

With this new preemption level specification, the receiver node must confirm that it has preemption capabilities, and its supported level preemption (0, 1, or 2). When a preemptable frame reaches CHECK_FOR_START, then the function transits to: (i) P_RECEIVE_DATA if it is a p_1 frame; or (ii) the newly defined P2_RECEIVE_DATA state if it is a p_2 frame. In this context, each p_1 frame is received as a traditional pMAC frame, whereas each p_2 frame is received such that it can be preempted by both express and p_1 frames. When a preemption occurs, the reception of the p_2 frame resumes only if all pending express and p_1 frames/fragments have completed.

At this stage, we have described the modifications required to enable multi-level preemption. In addition, there are other key operational factors (*interoperability* and *frame buffering*)

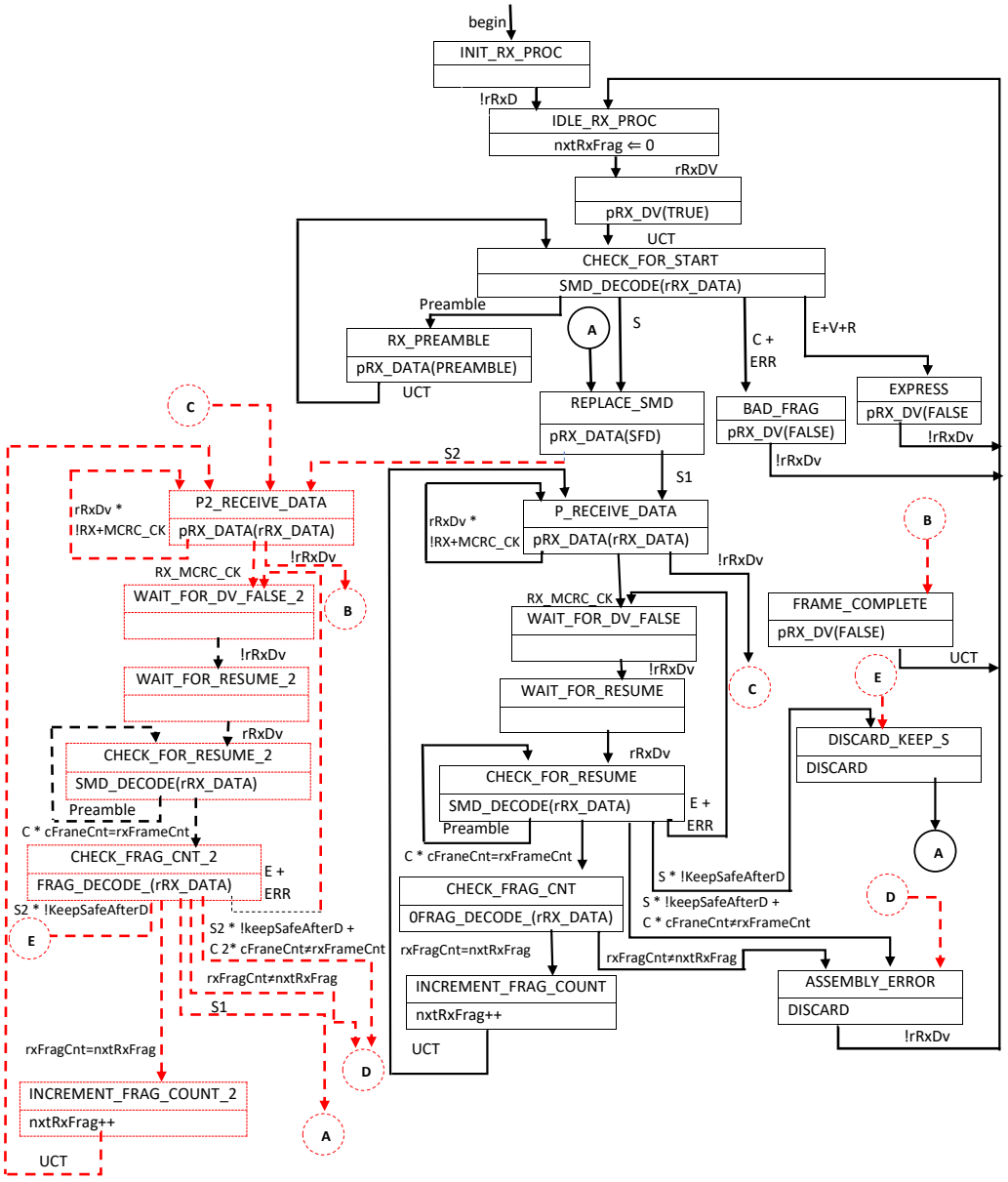


Fig. 7: Modified "Receive Processing" state diagram for two-level preemption. Added features are represented in red color.

that need to be revisited before a full roll-out of multi-level preemption enabled nodes. Below, we address these concerns.

C. Interoperability

In practice, nodes with more than one preemption level capability will coexist with others with at most one. This coexistence can be guaranteed by the verification process mentioned in the previous subsection. After a node sends a verification request to check whether the receiver node supports preemption, the reply should not only indicate this information, but also the preemption level supported. Similar to the 1-level preemption scheme, multi-level preemption is enabled only if the receiver node supports it. If k-level preemption (with $k \in \mathbb{N}$) is supported, then the sending node will transmit the frames in a k-level preemptive manner. We note that in this work, we assume a 2-level preemption paradigm for simplicity sake.

D. Frame Buffering

In the multi-level preemption scheme, careful attention must be paid to how the input buffers are used in the reception of frames. We recall that the reception of a sequence of fragments belonging to a preempted frame is completed as soon as the last four octets of the fragment does not match the mCRC. Once this mismatch occurs, the receiver node assembles the content of its buffer as a single frame. Under a multi-level preemption scenario, this mismatch may occur due the reception of a preemptable frame belonging to a

	f^1	f^2	f^3
Source	ES1	ES2	ES3
Destination	ES4	ES5	ES6
Payload (bytes)	300	700	1496
Deadlines	120μs	250μs	-
Period	500us	700us	1ms
Service category	express	tpflow	bpflow

TABLE I: Network flow properties

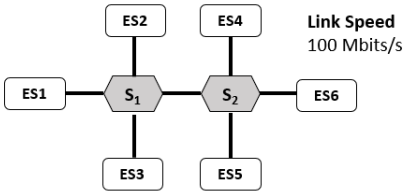


Fig. 8: Network topology

higher preemption service category. In this case, there will be two sets of preemptable frame fragments at the input port of the receiver node. We recommend that frames of different preemption categories should be received in different input buffers. This would guarantee the integrity of each frame.

VII. Experimental results

This section reports on the experiments performed to demonstrate the limitation of the one-level preemption scheme and the need for a multi-level preemption approach. The experiments were performed by using NeSTiNg [18], a simulation model for TSN using the OMNeT++ framework [19].

▷ **Setup**. We consider a network topology consisting of six end-stations and two full-duplex preemption enabled TSN switches S_1 and S_2 as shown in Fig. 8. For proof of concept purposes, we assume only three flows (f^1, f^2 and f^3). The parameters of the flows are specified in Table I, together with their source and destination nodes. Two batches of experiments are conducted: (a) Only flow f^1 is assigned to eMAC; and (b) Flows f^1 and f^2 are assigned to eMAC. A simulation time of $10^6\mu$s is used for each simulation run. This simulation time was large enough to showcase our approach.

▷ **Results and discussion**. About Scenario (a), f^1 was able to meet its deadline in all simulation instances, with a worst-case end-to-end delay of 109μs and an average end-to-end delay of 91.16μs (see Fig. 9a, the blue plot). This trend is explained by the fact that f^1 had exclusive access to the eMAC interface. In this scenario, flow f^2 could not always meet its deadline, unfortunately. This is due to the fact that it shares the pMAC interface with f^3, whose transmission induced a blocking time up to 123.36μs. The observed worst-case end-to-end delay of flow f^2 was 365μs, with an average delay of 253.116μs (see Fig. 9a, the red plot).

The deadline miss observed for f^2 in Scenario (a) can be resolved by moving this flow to the eMAC interface as opted for in Scenario (b) (see Fig. 9b, the red plot). In this new scenario, f^2 now records a worst-case end-to-end delay of 197.4μs (with an average of 187.79μs), but f^1 is now unable to meet its deadline in some of the simulation instances (see Fig. 9b, the blue plot). The observed worst case end-to-end delay for f^1 is now 128μs, which is well above its deadline. As a matter of fact, moving f^2 to the eMAC interface provides it with a prompt service, but this changes the picture for flow f^1, unfortunately. This is due to the blocking time induced by the transmission of f^2. Thus, it is not possible to meet the deadlines of both f^1 and f^2 simultaneously, while assuming the traditional 1-level preemption scheme.

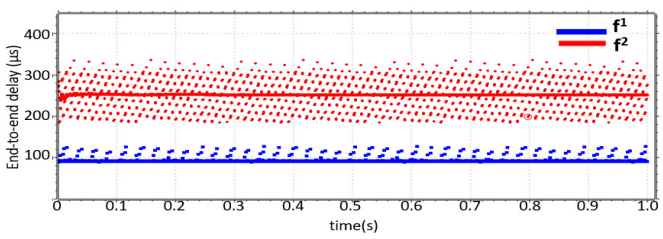
Assuming Scenario (a) and a 2-level preemption scheme, all instances of f^1 would be able to meet its deadline (the worst-case end-to-end delay of this flow would remain 109μs); while all instances of f^2 would also be protected from a long blocking by flow f^3. The worst-case end-to-end delay of this flow would be 203.5μs, thus meeting its deadline and promoting multi-level preemption schemes.

VIII. Conclusion and future directions

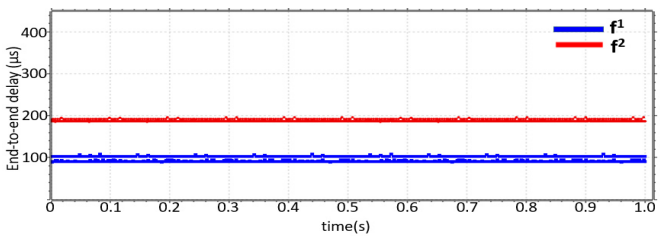
In this paper, we highlighted the limitations of the 1-level preemption scheme for TSN. Then, we examined the preemption mechanism in details to determine the feasibility of a multi-level preemption scheme. We detailed the required modifications to reach this goal and provided implementation recommendations to guarantee frame integrity and interoperability. Finally, in the experimental section, we identify a simple scenario where the proposed approach would be beneficial. Moving forth, we plan to implement this approach in OMNeT++ and perform a formal performance analysis.

Acknowledgment

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020).



(a) Only flow f^1 is assigned to eMAC.



(b) Flows f^1 and f^2 are assigned to eMAC.

Fig. 9: Evaluation of the end-to-end delay for flows f^1 and f^2.

REFERENCES

[1] IEEE, "IEEE approved draft standard for local and metropolitan area networks-media access control (MAC) bridges and virtual bridged local area networks amendment: Frame preemption." *P802.1Qbu/D3.1*, pp. 1–50, 2015.

[2] ——, "IEEE standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *Std 802.1Qbv-2015*, pp. 1–57, 2016.

[3] ——, "IEEE standard for local and metropolitan area networks–frame replication and elimination for reliability," *Std 802.1CB-2017*, pp. 1–102, 2017.

[4] ——, "IEEE standard for local and metropolitan area networks—bridges and bridged networks - amendment 24: Path control and reservation," *Std 802.1Qca-2015*, pp. 1–120, 2016.

[5] ——, "IEEE draft standard for local and metropolitan area networks–media access control (mac) bridges and virtual bridged local area networks amendment: Stream reservation protocol (SRP) enhancements and performance improvements," *P802.1Qcc/D2.2*, pp. 1–214, 2018.

[6] 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 Com. Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.

[7] IEEE, "Time-Sensitive Networking Task Group." [Online]. Available: http://www.IEEE 802.org/1/pages/tsn.html

[8] C. Simon, M. Máté, M. Maliosz, and N. Bella, "Ethernet with time sensitive networking tools for industrial networks," *Infocommunications Journal*, vol. 9, no. 2, pp. 6–14, 2017.

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

[10] 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.

[11] Y. Kim, "Very low latency packet delivery requirements and problem statements," in *IEEE 802.1 AVB Task Group Int. Meeting. USA*, 2011.

[12] IEEE, "IEEE 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, 2016.

[13] W. K. Jia, G. H. Liu, and Y. C. Chen, "Performance evaluation of IEEE 802.1qbu: Experimental and simulation results," in *38th IEEE Conf. on Local Computer Networks*, 2013, pp. 659–662.

[14] J. Kim, B. Y. Lee, and J. Park, "Preemptive switched ethernet for real-time process control system," in *11th IEEE Int. Conf. on Industrial Informatics*, 2013, pp. 171–176.

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

[16] L. L. Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[17] IEEE, "IEEE standard for local and metropolitan area networks–bridges and bridged networks," *Std 802.1Q-2014*, pp. 1–1832, 2014.

[18] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "NeSTiNg: Simulating IEEE time-sensitive networking (TSN) in OMNeT++," in *Int. Conf. on Networked Systems*, 2019.

[19] OMNET++, "Discrete Event Simulator." [Online]. Available: https://omnetpp.org/