CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Journal Paper

Worst-Case Traversal Time Analysis of TSN with Multi-level Preemption

Mubarak Ojewale*
Patrick Meumeu Yomsi*
Borislav Nicolic

*CISTER Research Centre

CISTER-TR-210208

2021

Worst-Case Traversal Time Analysis of TSN with Multi-level Preemption

Mubarak Ojewale*, Patrick Meumeu Yomsi*, Borislav Nicolic

*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, borislav.nikolic@metropolitan.ac.rs

https://www.cister-labs.pt

Abstract

To achieve low latency transmission of time-sensitive flows in Ethernet networks, the IEEE introduced the IEEE 802.1Qbu Standard, which specifies a 1-level preemption scheme for IEEE 802.1 networks. The specification allows the suspension of a preemptable frame prior to its completion for the speedy transmission of an express frame; but any other preemptable frame cannot be transmitted before the completion of the already preempted frame. While this approach improves the performance of express frames, the 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. The work focused on the feasibility and implementation requirement of such an approach, but a formal analysis of the worst-case performance guarantees under the proposed scheme was not provided. In this paper, we fill this gap by presenting the aforementioned analysis of the TSN IEEE 802.1Qbu networks under the multi-level preemption assumption. Evaluation is performed with a realistic automotive use-case and the results showcase an improvement of up to 53:07% for preemptable frames with firm timing requirements.

Journal of Systems Architecture 116 (2021) 102079

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Worst-case traversal time analysis of TSN with multi-level preemption

Mubarak Adetunji Ojewale [a,*], Patrick Meumeu Yomsi [a], Borislav Nikolić [b]

[a] CISTER Research Centre, ISEP, Polytechnic Institute of Porto, Portugal
[b] Faculty of Information Technology, Metropolitan University, Serbia

ARTICLE INFO

Keywords:
Frame preemption
Ethernet
Time-sensitive networking
Real-time networks

ABSTRACT

To achieve low latency transmission of time-sensitive flows in Ethernet networks, the IEEE introduced the IEEE 802.1Qbu Standard, which specifies a 1-level preemption scheme for IEEE 802.1 networks. The specification allows the suspension of a *preemptable frame* prior to its completion for the speedy transmission of an *express frame*; but any other preemptable frame cannot be transmitted before the completion of the already preempted frame. While this approach improves the performance of express frames, the 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. The work focused on the feasibility and implementation requirement of such an approach, but a formal analysis of the worst-case performance guarantees under the proposed scheme was not provided. In this paper, we fill this gap by presenting the aforementioned analysis of the TSN IEEE 802.1Qbu networks under the multi-level preemption assumption. Evaluation is performed with a realistic automotive use-case and the results showcase an improvement up to 53.07% for preemptable frames with firm timing requirements.

1. Introduction

One of the major requirements of distributed real-time systems is "*real-time communication*", i.e., the ability of data to move on the underlying network in a reliable and predictable manner. To date, an entire body of promising solutions have been proposed in literature, each targeting a set of domain-specific requirements. Few examples of well-established solutions include the *Controller Area Network* (CAN) protocol [1], the *Local Interconnect Network* (LIN) protocol [2], the *FlexRay* protocol [3], and TTEthernet [4] that mainly focus on the automotive domain; the *Avionics Full Duplex Switched Ethernet* (AFDX) [5] that targets the avionic domain; and the so called field buses such as SERCUS III [6], EtherCAT [7], and PROFINET [8] that address the industrial domain. Most of these solutions now struggle to keep up with the growing bandwidth and performance demands of emerging applications in their respective domains [9]. On another front, with the convergence of Operation Technology (OT) and Information Technology (IT) [10,11], yet another challenge has settled on the table, justifying the need for new communication technologies that would facilitate the handling of heterogeneous traffic (real-time, non real-time, long and short frames) on the same network infrastructure. In this regard, among all the candidate solutions, Ethernet [12] bursts forth as the leading and most promising replacement for all previous technologies. This is due to its ability to scale up to the increasingly stringent timing requirements and distance, as well as its high bandwidth capacity [13]. Unfortunately, the legacy Ethernet standards were originally designed targeting only non real-time applications and desirable features for real-time applications like (1) frame preemption [14]; (2) global time synchronization [15]; (3) frame replication and elimination [16]; (4) path control and reservation [17]; and finally (5) network configuration [18] among other features were missing. A tremendous amount of work has been achieved in this direction over the past few decades and several modifications and/or amendments have been made to the standards. Ethernet has successfully been enhanced with all the aforementioned features [19], thus leading to a set of *updated standards*, referred to as *Time-Sensitive Networking* (TSN) [20].

▷ **IEEE 802.1Qbu.** The IEEE 802.1Qbu [21] (now merged with the IEEE 802.1Q-2018 Standard [22]) TSN sub-standard introduces a 1-level frame preemption feature into the IEEE 802.1 networks. This allows to temporarily suspend the transmission of a frame prior to its completion for the speedy transmission of a higher priority frame. This standard is closely connected to the IEEE 802.3br [14] Standard, which allows urgent time-critical data frames to split non-critical frames in transit over a physical link into smaller fragments [23]. Specifically,

* Corresponding author.
E-mail addresses: mkaoe@isep.ipp.pt (M.A. Ojewale), pmy@isep.ipp.pt (P.M. Yomsi), borislav.nikolic@metropolitan.ac.rs (B. Nikolić).

https://doi.org/10.1016/j.sysarc.2021.102079
Received 4 August 2020; Received in revised form 10 February 2021; Accepted 26 February 2021
Available online 3 March 2021

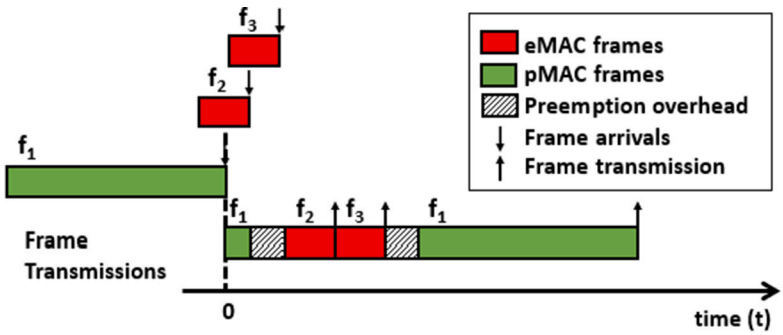


Fig. 1. Illustration of the one-level preemption scheme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

depending on their priority levels and timing requirements, frames are mapped to two Media Access Control (MAC) Sublayer service interfaces namely, (1) the "*Express MAC*" (eMAC) and (2) the "*Preemptable MAC*" (pMAC). Frames mapped to the eMAC and pMAC interfaces are referred to as the "express frames" and "preemptable frames", respectively. These two classes are managed as follows: (1) only express frames can preempt preemptable frames; and (2) two frames belonging to the same class cannot preempt each other. Fig. 1 illustrates such a scenario at a given output node.

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 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. The total overhead associated to the occurrence of each preemption is equivalent to the time taken to transmit 24 bytes of data (i.e., 0.19 µs and 1.9 µs, assuming a $1Gb$ and $100Mb$ speed Ethernet, respectively). Experimental studies show that this approach improves the performance of express frames [9,13]. Specifically, Thiele and Ernst [9] show that the performance of Standard Ethernet with frame preemption is comparable to that of IEEE 802.1Qbv [15], thus making it an interesting alternative.

▷ **IEEE 802.1Qbv.** The IEEE 802.1Qbv standard defines a time-triggered gate control mechanism for TSN flows, using a Time-Aware Shaper (TAS). Here, each queue at a switch output port has a guaranteed transmission slot in a cyclic and pre-computed dispatch schedule called the Gate Control List (GCL). A frame on a queue can be transmitted only when the gate of the queue is opened at the time period according to the GCL configuration. Note that each queue contains a unique class of frames, i.e., frames belonging to different traffic classes cannot share the same queue. The standard also specifies a so-called "guard-band" that ensured that the transmission of time-triggered flows is protected from any interference by other flows. The guard band is a time-period between the gate-close operation non-time-triggered queues and the commencement of transmission of time-triggered flows. The length of this guard band is equal to the time taken to transmit the largest possible non-time-sensitive frame on the network. While the guard band ensures that time-triggered traffic are protected from interference, it results in poor utilization of network bandwidth.

▷ **Comparing IEEE 802.1Qbu and IEEE 802.1Qbv from a qualitative standpoint:** Thiele and Ernst [9] noted that a IEEE 802.1Qbu network is less complex to configure as only two basic steps are required: (1) the assignment of priorities to flows; and (2) the assignment of flows to preemption interfaces [24]. Even though this process is non-trivial [24], it is bearable in comparison to the IEEE 802.1Qbv configuration. Here, TAS requires finding a suitable schedule for each flow at each switch output port and this problem has been shown to be NP-Hard [25], unfortunately. On another front, IEEE 802.1Qbu solves an issue of IEEE 802.1Qbv. By combining frame preemption with TAS, optimal bandwidth utilization is possible for non-scheduled traffic and this enables low-latency communication [23]. Nevertheless, the IEEE 802.1Qbv standard remains central to TSN. It defines up to 8 queues per port for forwarding traffic and frames are assigned to queues based on Quality of Service (QoS) priority. The TAS mechanism blocks all ports except one based on a predefined schedule in order to prevent delays during scheduled transmission. From an implementation standpoint, the performance of IEEE 802.1Qbu networks degrades in scenarios where the number of express frames is high [23]. This is due to the fact that frames of the same class are transmitted in non-preemptive manner. As a consequence, there are frames that cannot be classified as express, but have firm timing requirements. In order not to jeopardize the schedulability of these frames, and the schedulability of the entire system subsequently, these frames should not be blocked for prohibitively long time periods. The 1-level preemption operation in the current version of the standards does not allow preemptable frames with firm timing requirements to leverage the basic benefits of enabling preemption, unfortunately.

To circumvent the limitations of the 1-level preemption scheme, a multi-level preemption scheme has been proposed [26,27]. This setup allows for more than two preemption classes and introduces a new class of flows called *time-sensitive preemptable traffic* (tpflows), i.e., flows with firm timing requirements. In this scheme, tpflow frames can be preempted by express frames and at the same time preempt lower priority frames to satisfy their timing requirements. Fig. 2 illustrates an example scenario that compares the improvement in the responsiveness of tpflows under a 1-level and a 2-level preemption setups. In this figure, five frames (f_1, f_2, f_3, f_4, and f_5) are considered for transmission from a certain switch. All frames traverse in the same direction, so they compete for transmission from the output port. We assume three types of frames: (1) frames f_2 and f_4 are express frames (in red colored boxes) with stringent timing constraints; (2) frames f_3 and f_5 are preemptable frames (in black colored box) with firm timing constraints; and finally, (3) f_1 is a preemptable frame with no timing constraint (in green colored box). In this example, it is assumed that 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

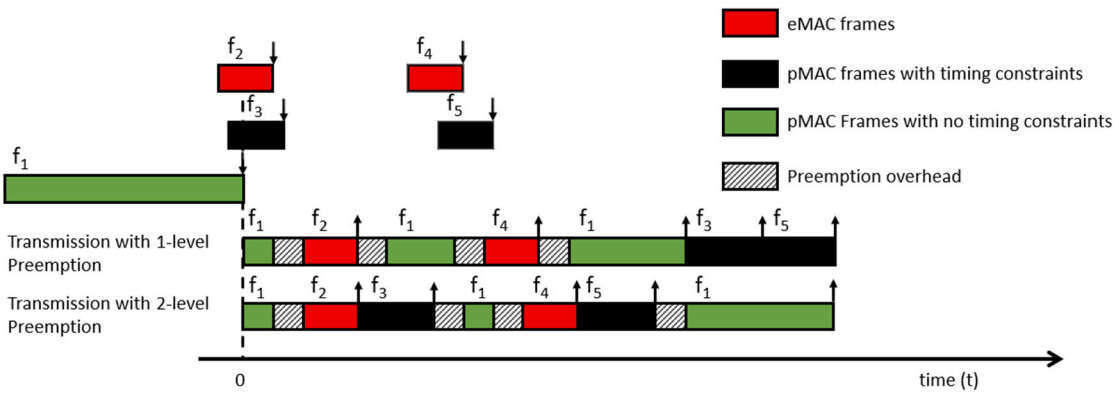


Fig. 2. Frame transmissions under 1-level preemption and 2-level preemption schemes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

higher priority than f_1. Frame f_1 arrives first (at time 0) and starts its transmission, followed by frame f_2, and finally f_3.

▷ **In the 1-level preemption paradigm:** upon the arrival of f_2, f_1 is preempted since f_2 is an express 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 situation is due to the 1-level preemption scheme as preemptable frames cannot preempt each other, thus exhibiting a *priority inversion* problem. Consequently, the transmissions of 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 paradigm:** The aforementioned issue is circumvented. Here, frames f_3 and f_5 received better services. Frame f_3 is transmitted right after the completion of f_2, and f_5 is transmitted right after the completion of f_4. This allows f_3 to meet its timing requirement, as well as reduces the completion times of both f_3 and f_5.

Ojewale et al. already provided another example (see [27], Section 7) to promote the need for multi-level preemption which will not be replicated in this paper.

Contribution. A worst-case traversal time (WCTT) analysis for TSN frames has been conducted in the literature by Thiele and Ernst [9] using Compositional Performance Analysis (CPA) framework [28]. However, the authors only considered the traditional 1-level preemption scheme as defined in the standards. Also, Ojewale et al. [26,27] propose the multi-level preemption scheme, but do not provide any formal timing guarantees for frames under such a scheme. In this paper, we build on top of these studies. We enrich the state of the art by developing a WCTT analysis of TSN frames, considering a multi-level preemption scheme. To the best of our knowledge, this is the first contribution in this direction. According to the TSN IEEE 802.1 Qbu standard, frame preemption can be implemented with or without the TSN shapers such as TAS and Credit-Based Shapers (CBS) [29]. In this work, we choose the latter scenario to focus purely on the evaluation of multi-level preemption without the added complexity of other protocol mechanisms.

Paper organization. The rest of the paper is structured as follows. The model of computation is presented in Section 2 together with key notations used throughout the paper. Section 3 provides a background on the CPA approach while Section 4 presents the proposed analysis. Section 5 contains experimental results from the evaluation of the proposed analysis, while Section 6 discusses relevant related works. Finally, Section 7 concludes the paper and presents future research directions.

2. Model of computation

Network specification. We assume an Ethernet backbone network for a real-time distributed system. We model the network as a directed graph $G \stackrel{\text{def}}{=} (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of all nodes and $\mathcal{L}$ is the set of all physical links in the network. We assume that every link is bi-directional; full-duplex; and operates at a single reference speed, say $s > 0$. The tuple G is given with the interpretation that: (1) $\mathcal{V} = \text{EP} \cup \text{SW}$, where $\text{EP} \stackrel{\text{def}}{=} \{\text{EP}_1, \text{EP}_2, \ldots\}$ represents the set of all end-points and $\text{SW} \stackrel{\text{def}}{=} \{\text{SW}_1, \text{SW}_2, \ldots\}$ the set of all switches. Each EP_k (with $k \geq 1$) has a single I/O port and can receive and send network traffic while SW consists only of forwarding nodes, each with a finite number of output ports, through which the traffic is routed. Each SW_ℓ (with $\ell \geq 1$) is enabled with 802.1Qbu capability and decides, based on its internal routing table, to which output port a received frame will be forwarded.

Traffic specification. We consider a network traffic consisting of a set $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ of $n \geq 1$ flows partitioned into two traffic classes: the *express traffic* (denoted by $\mathcal{E}$) and the *preemptable traffic* (denoted by $\mathcal{P}$), i.e., $\mathcal{F} = \mathcal{E} \cup \mathcal{P}$. In other words, $\mathcal{E} \cup \mathcal{P} = \mathcal{F}$ and $\mathcal{E} \cap \mathcal{P} = \emptyset$. Each flow $f_i \in \mathcal{F}$ consists of a potentially infinite number of frames f_i^k (with $k \geq 1$) and the smaller the subscript of a flow, the higher its priority. This means that flows with different priorities follow a total order. All frames generated by a flow inherit its priority. We assume that the preemptable traffic class is further partitioned into two disjoint sub-classes referred to as the *time-sensitive preemptable traffic* (tpflows), with firm timing requirements (deadlines); and *best effort preemptable traffic* (bpflows), with no timing requirements at all. We denote these two sub-classes as $\mathcal{TP}$ and $\mathcal{BP}$, respectively: $\mathcal{P} = \mathcal{TP} \cup \mathcal{BP}$. Frame f_i^k is the k^{th} frame of f_i and is characterized by its arrival time a_i^k and its payload p_i^k. The size p_i^k of flow f_i is the sum of its payload and transmission overheads. Throughout this paper, we assume that the maximum size of any flow is limited to 1518 bytes, i.e., the maximum Ethernet frame size as defined in the IEEE 802.3 Standard [14]. Every flow f_i is also assigned a unique preemption class $C\ell(i)$ that qualifies the set of flows with the same preemption properties as f_i. For convenience, we assume that there are m different preemption classes in the system (1 denotes the highest preemption class – comprising only flows in $\mathcal{E}$, then 2, 3, and so on till $m > 1$, the lowest one – comprising only flows in $\mathcal{BP}$). Note that $C\ell(i) \cap C\ell(j) = \emptyset$, $\forall i \neq j$ and the following rules are enforced.

- R_1– Every flow, say f_i, can preempt any other flow, say f_j, only if $C\ell(i) < C\ell(j)$, but flows in $C\ell(j)$ cannot preempt flows in $C\ell(i)$;
- R_2– Flows in the same preemption class cannot preempt each other and are transmitted with a strict priority order;
- R_3– Flows with the same priority are transmitted in a FIFO manner;
- R_4– Flows with the same priority always belong to the same preemption class but the converse is not true. In other words, flows with different priorities can be assigned to the same preemption class (see Fig. 3).

Finally, for the sake of readability, an overview of notations used throughout this paper is provided in Table 1.

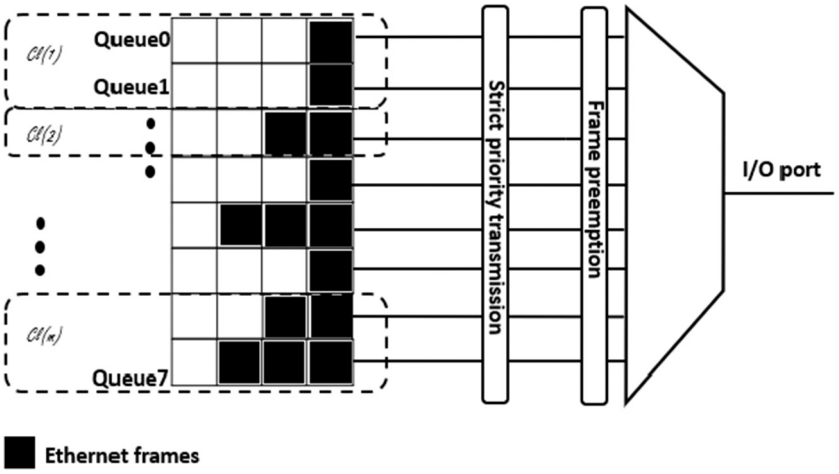


Fig. 3. System model: Preemption class vs. priority queues.

Table 1
Overview of key variables.

a_i^q	Arrival time of the qth frame (f_i^q) of flow f_i
$\mathcal{BP}$	Set of best-effort preemptable flows
C_i^+	Maximum time to transmit any frame of f_i
$C\mathscr{C}(i)$	Set of flows in the same preemption class as f_i
$\delta_i^{+/-}$(q)	Latest/earliest arrival time of f_i^q
$\mathcal{E}$	Set of eMAC flows
F_i^+	Maximum number of fragments of C_i^+
$hep(i)$	Set of flows with a higher priority than or equal to f_i
HPI_i	Higher-priority interference suffered by f_i
$hp(i)$	Set of flows with a higher priority than f_i
$lp(i)$	Set of flows with a lower priority than f_i
LPB_i	Lower-priority blocking suffered by f_i (including express frames)
mWCTT	Maximum measured end-to-end delay
$\eta_i^{+/-}(\Delta t)$	Maximum/Minimum possible arrivals of a frame of f_i within the period Δt
$p_i^{+/-}$	Maximum/Minimum payload of frames of f_i
PB_i	Blocking suffered by f_i due to the transmission of a lower-priority preemptable frame/frame fragment
$PO_i(\Delta t, q, a_i^q)$	Preemption overhead incurred by f_i^q
$Q_i(q, a_i^q)$	Queuing delay of f_i^q
rTX	Link data transmission rate
SPB_i	Same-priority blocking suffered by f_i
$sp(i)$	Set of all flows with the same priority as f_i
$\mathcal{TP}$	Set of time-sensitive preemptable flows
$\text{WCTT}(q, a_i^q)$	Worst-case traversal time of f_i^q

3. A brief background on CPA

Before we present our proposed WCTT analysis, it is paramount to provide the reader with a brief background on the CPA approach for a smooth understanding of the rest of this paper. For a complete and detailed description of the fundamental concepts, we refer the interested reader to [28]. In this framework, a component is modeled as a resource ρ, which provides some service to one or more tasks. Tasks' activations are abstracted by an event model, which defines an upper-bound (denoted by $\eta^+(\Delta t)$) and a lower-bound (denoted by $\eta^-(\Delta t)$) on the number of activations within any half-open time interval $[t, t+\Delta t)$. This framework also defines a distance function $\delta^+(q)$ (respectively, $\delta^-(q)$) which denotes an upper-bound (respectively, a lower-bound) on the maximum (respectively, minimum) time instant at which the q^{th} activation instance can occur [28,30]. Fig. 4 illustrates the event model of a task with a period and jitter of 50 time units. A jitter of 50 time units (and equal to the period) implies that two instances can occur simultaneously ($\delta^-(2) = 0$; $\eta^+(0) = 2$) or two periods apart ($\delta^+(2) = 500$; $\eta^-(500) = 0$).

The service period of each task, say τ_i, is bounded by two parameters: from above by C_i^+ and from below by C_i^-. These parameters respectively represent the longest and the shortest time it takes the resource, say ρ, to service an instance of τ_i, in the absence of any blocking or interference. The worst-case response time (WCRT) of task τ_i is investigated within a so-called level-i busy period. In summary, this is a time interval $[\sigma, \mu]$ within which only task instances belonging to $hep(i)$ (except the first job, which is generated from task $\tau_j \in lp(i)$ with the longest C_j^+) are executed throughout $[\sigma, \mu]$, but no jobs belonging to $hep(i)$ are executed in $[\sigma-\epsilon, \sigma)$ or $(\mu, \mu+\epsilon]$ for any arbitrary small $\epsilon > 0$.

Since the WCRT of τ_i may not occur at its first activation, CPA examines all the q activations of τ_i within the level-i busy period. To this end, it defines a *q-activation processing time* $B_i(q)$ which describes how long the resource ρ is busy processing q jobs of τ_i. This is computed in Eq. (1).

$$B_i(q) = Q_i^+(q) + C_i^+ \tag{1}$$

In Eq. (1), note that $B_i(q)$ assumes a non-preemptive scheduling scheme. Here, $Q_i^+(q)$ is the time interval from the start of the level-i busy period to the beginning of service of the qth instance of τ_i. It is computed as in Eq. (2).

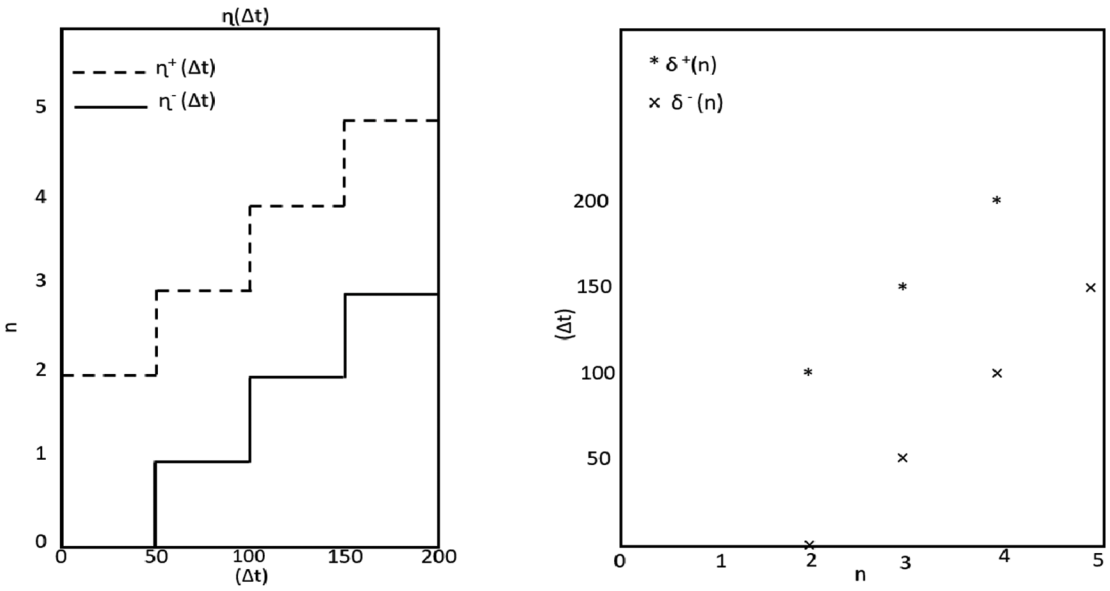
$$Q_i^+(q) = \max_{j \in lp(i)} \left\{ C_j^+ \right\} + (q-1) \cdot C_i^+ + \sum_{k \in hp(i)} C_k^+ \cdot \eta_k^+ \left(Q_i(q) + \epsilon \right) \tag{2}$$

In Eq. (2), $lp(i)$ and $hp(i)$ denote the set of tasks with a lower and higher priority than τ_i, respectively. The first term computes the maximum delay that can be experienced due to the presence of a lower priority task in the system; the second term estimates the delay experienced due to the processing of all preceding $q-1$ instances; and the third term computes all possible delays due to the service of higher priority tasks within $B_i(q)$. It is worth mentioning that Eq. (2) is reminiscent of the Worst-Case Response Time estimation for the classical single-core fixed-priority tasks and it is also solved by using a fixed point algorithm, i.e., the solution is computed in an iterative manner and the algorithm stops as soon as two consecutive values of $Q_i^+(q)$ are equal or when one value exceeds the deadline. In the latter case, the set of flows is deemed unschedulable. After obtaining $Q_i^+(q)$, we can derive the maximum number of activations q_i^+ of τ_i within $B_i(q)$ by using Eq. (3).

$$q_i^+ = \min\{q \geq 1 \mid B_i(q) < \delta_i^-(q+1)\} \tag{3}$$

This equation computes the first q_i instances of τ_i such that the completion time of instance q_i is less than the earliest arrival time of instance q_{i+1}. The response time of the q^{th} instance of τ_i is given by computing the difference between the time instant when all the q instances of τ_i have been processed and the arrival time of the instance q. Formally, this is given by Eq. (4).

$$R_i^+(q) = B_i(q) - \delta_i^-(q) \tag{4}$$



(a.) An event arrival function. (b.) An event distance function

Fig. 4. CPA event model for a task with a period and jitter of 50 time units.

Finally, the WCRT R_i^+ of τ_i is the maximum $R_i^+(q)$ over all q_i^+ activations of τ_i within the busy window and is computed through Eq. (5).

$$R_i^+ = \max_{1 \le q \le q_i^+} \left\{ R_i^+(q) \right\} \tag{5}$$

So far, we have discussed the worst-case delay at each resource node, also called the *local analysis*. Specifically, a rigorous computation of the maximum possible blocking/interference that a task can suffer is carried out locally at every resource node. This means that the potential interference between any two tasks will be effectively captured when performing the local analysis at the shared resource node. But in many cases, real-time applications involve interacting tasks sharing different resources. To evaluate the WCRT of a task in this scenario, the CPA defines an *event propagation* step in addition to the local analysis step. Here, the output of each resource along the execution path of a task serves as input for the next one and $R_i^+(q)$ at the last node is the WCRT of the task.

The CPA approach has been applied to standard Ethernet [31,32], AVB [33], and TSN [9,34]. In these works, the output port of the switches are the *resources* and flows are *tasks*. The frames are *jobs* or *instances* of the tasks and the path of a flow is modeled as a chain of dependent tasks [33]. Specifically, the resources render some *service(s)* (transmission) to some task activations (frames) upon some event occurrence (frame arrivals) within a time window (a level-i busy period). In this context, the service period of each frame is bounded by two parameters: from above by C^+ and from below by C^-. These parameters represent the longest and the shortest time to transmit the frame in absence of any interference. Obviously, these depend on the minimum and maximum possible payload $p^{-/+}$ of the flow to which the frame belongs to as well as the output link speed. Eq. (6) illustrates the relationship between these parameters.[1]

$$C^{-/+} = \frac{42 \text{ bytes} + \max\{42 \text{ bytes}, p^{-/+}\}}{rTX} \tag{6}$$

In this equation, rTX represents the transmission speed on the port link and the constant terms (here, 42) represent the protocol overhead and minimum frame payload requirement as specified in the Standards [22]. As shown in Fig. 5, the end-to-end delay that a frame experiences in a switch fabric consists of the following five components: (1) the input delay at the switch input port; (2) the processing delay; (3) the queuing delay at the switch output port; (4) the propagation delay; and finally (5) the transmission delay [35]. As pointed out by Thiele and Ernst [9], all these components are implementation dependent and usually in the order of a few clock cycles, except for the queuing delay, which is captured in Eq. (2) and whose components are illustrated in Fig. 6.

In Fig. 6, frame f_i^q arrives at time a_i^q during the transmission of a lower preemption class frame, LP (green box). Before a_i^q, frame HP (red box) of the express class had arrived as well as two other frames SP (black boxes) with the same priority and preemption class as f_i^q. The preemption operation commands that a minimum number of bytes has been transmitted and a number of bytes is left so that the preempted fragment can meet the minimum size requirement of a valid Ethernet frame. This brings the highest possible blocking that a preempting frame can incur to the size of the largest non-preemptable fragment of a preemptable frame, i.e., 143 bytes of data [9] and explains the transmission process of LP before it is preempted. Upon the preemption of LP, HP; all SP frames; and all newly arrived HP frames are transmitted, thus forcing f_i^q to be served only afterwards. Note that in the representation of the frame transmission process in Fig. 6, frames are labeled based on their contribution to the queuing delay, e.g., the segment of LP that is transmitted before f_i^q is labeled as LPB.

In addition to the queuing delay components captured in Eq. (2), the preemption overhead is yet another term that should be taken into account. This is not the case in the general CPA model, unfortunately [30]. The preemption overhead is a significant factor in the TSN frame transmission scheme. The total overhead induced by each preemption is 12 bytes (i.e., 6-byte preamble, 1-byte start frame delimiter, 1-byte frame count variable; and finally 4-byte error check variable) [26]. In addition, 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 equals the time it takes to transmit 12 bytes of data. This brings the total overhead associated to each preemption to 24 bytes [9, 26]. Thiele and Ernst [9] have also proven in that the maximum number of preemptions that a single frame can suffer is given by Eq. (7). Eq. (7) provides an upper-bound on the number of times that an

[1] The minimum size of a frame is 84 bytes w.r.t. the Standards specification.

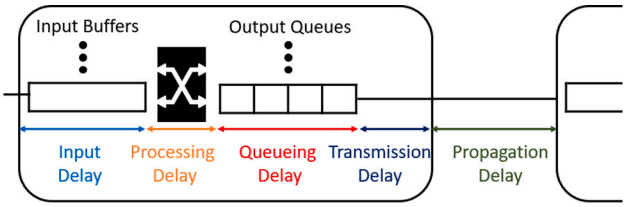


Fig. 5. End-to-end delay components.

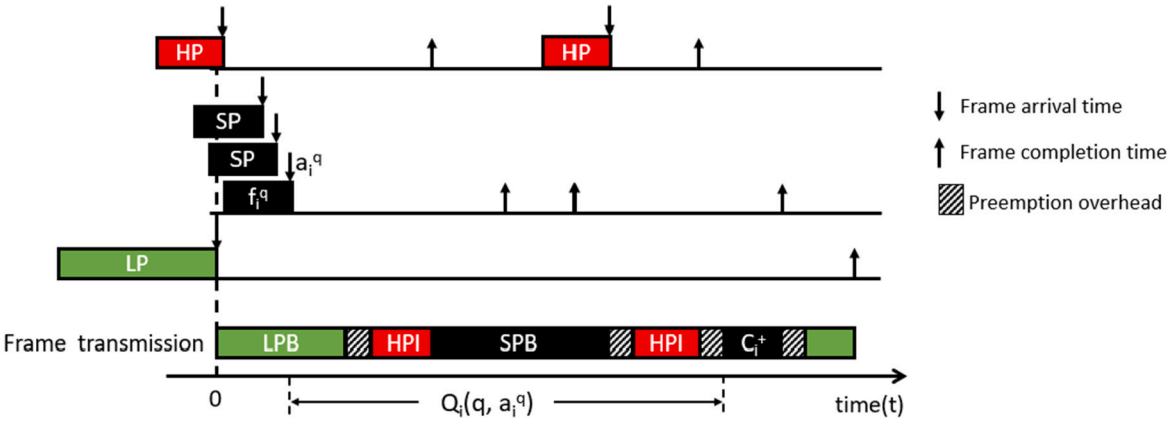


Fig. 6. Queuing delay components. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Ethernet frame can be preempted, due to the restrictions defined in the standards on the minimum valid frame size.

$$F_i^+ = \left\lfloor \frac{p_i^+ - 42\ bytes}{60\ bytes} \right\rfloor \tag{7}$$

In this equation, the constant terms (here, 42 and 60) are the minimum payload requirements for the first and subsequent fragments of a preemptable frame in the Standards [22]. For every flow f_i, its worst-case traversal time is obtained by combining Eqs. (1) and (4). We note that Eq. (4) does not take into account the preemption overhead. This computation will be refined to integrate the cost in Section 4.6. Also, note that the CPA model allows arbitrary deadlines, i.e., there is no particular correlation between the deadline and the minimal inter-arrival time of each flow.

4. Proposed approach

In the multi-level preemption scheme, each frame can belong to only one of the following three categories: (1) can preempt other frames and is non-preemptable, i.e., cannot be preempted by any other frame, because it is part of the highest priority class (e.g., express traffic); (2) can preempt other frames and can also be preempted and finally (3) cannot preempt other frames and can be preempted by any frame in categories (1) and (2). We recall the flow classes mentioned in Section 2: the class of "express flows"; the classes of "preemptable flows with firm timing requirements" (tpflows) and the class of "best effort frames" (bpflows). While categories (1) and (4) maps directly to express and bpflows, respectively, categories (2) map to tpflows. With this mapping in mind, we have everything we need to discuss the components of the end-to-end delay of a frame, say f_i^q that arrives at time a_i^q. In a nutshell, there are four components, namely: (1) the *lower-priority blocking* – i.e., the delay due to frame(s) with a lower priority than f_i; (2) the *same-priority blocking* – i.e., the delay due to frame(s) with the same priority as f_i; (3) the *higher priority interference* – i.e., the delay associated to frame(s) with a higher priority than f_i; and finally, (4) the *preemption overheads*, which is the delay due to preemption overheads.

4.1. Lower-priority blocking

In this section, we derive an upper-bound on the lower-priority blocking experienced by a frame in each flow class.

4.1.1. Lower-priority blocking for "express flows"

Every express frame can experience the maximum lower-priority blocking in two scenarios: (*i*) when it is blocked by the largest lower priority express frame since frames in the same preemption class are served in a non-preemptive manner; or (*ii*) when it is blocked by the largest non-preemptable fragment of a preemptable frame.[2] Lemma 1 provides an upper-bound on the blocking due to scenario (*ii*).

Lemma 1 ($\mathrm{PB}_i^{\mathcal{E}}$). *For any express flow $f_i \in \mathcal{E}$, the maximum blocking of any frame f_i^q (with $q \geq 1$) caused by a preemptable frame is given by Eq. (8).*

$$\mathrm{PB}_i^{\mathcal{E}} = \min\left\{ \underbrace{\max_{j \in \mathcal{P}} \left\{ C_j^+ \right\}}_{(a)}, \underbrace{\frac{143\ bytes}{rTX}}_{(b)} \right\} \tag{8}$$

Proof. On one side, if all the preemptable frames are shorter than 143 *bytes*, then the maximum blocking time to any express frame is caused by the largest of these frames. This is captured by term (*a*). On the other side, the longest non-preemptable fragment of any preemptable frame is 143 bytes length [9]. This means a maximum blocking time of $\frac{143\ bytes}{rTX}$ (where rTX is the link speed). This is captured by term (*b*). Therefore, a tight upper-bound on the blocking time caused by a preemptable frame to any express frame f_i^q (with $q \geq 1$) is given by the minimum between terms (*a*) and (*b*), and the lemma follows. □

From Lemma 1, Theorem 1 provides a tight upper-bound on the lower-priority blocking incurred by any express frame.

[2] We recall that any preemptable frame that is less than or equal to 143 bytes in size cannot be preempted.

Theorem 1 ($\text{LPB}_i^{\mathcal{E}}$). *For any express flow $f_i \in \mathcal{E}$, if $lp^{\mathcal{E}}(i)$ represents the set of express frames with a priority lower than that of f_i, then a tight upper-bound on the lower-priority blocking to any frame f_i^q (with $q \geq 1$) is given by Eq. (9).*

$$\text{LPB}_i^{\mathcal{E}} = \max\left\{\underbrace{\max_{j \in lp^{\mathcal{E}}(i)}\left\{C_j^+\right\}}_{(a)}, \underbrace{\text{PB}_i^{\mathcal{E}}}_{(b)}\right\} \tag{9}$$

Proof. Any express frame f_i^q can suffer lower-priority blocking due either to the transmission of (1) a lower-priority express frame or (2) a preemptable frame. In the first case, term (a) captures the largest blocking time since frames in the same preemption class are served in a non-preemptive manner. On the other hand, if the blocking is caused by a preemptable frame, then Lemma 1 provides an upper-bound on the lower-priority blocking, i.e., $\text{PB}_i^{\mathcal{E}}$ (term (b)). Therefore, a tight upper-bound on the blocking time suffered by any express frame f_i^q (with $q \geq 1$) is given by the maximum between terms (a) and (b), and the theorem follows. □

Note that Eqs. (8) and (9) are similar to Equations 5 and 7 in [9]. We explicitly address this behavior in the analysis for the sake of completeness.

4.1.2. Lower-priority blocking for "tpflows"

Every flow can preempt all flows in a lower preemption class by design, but the converse is not true. A tpflow is no exception to this rule. This means that each tpflow frame f_i^q can be blocked by (i) at most the largest non-preemptable fragment of any preemptable frame in a lower preemption class or (ii) a lower-priority frame of the same preemption class as f_i, since frames of the same class are served in a non-preemptive manner. Lemma 2 computes an upper-bound on the blocking time incurred by a tpflow frame due to frame(s) in lower preemption classes.

Lemma 2 ($\text{PB}_i^{\mathcal{TP}}$). *For any flow $f_i \in \mathcal{TP}$, the maximum blocking time of any frame f_i^q (with $q \geq 1$) caused by a frame of a lower preemption class is given by Eq. (10).*

$$\text{PB}_i^{\mathcal{TP}} = \min\left\{\underbrace{\max_{j \in \{lp(i) \mid C\ell(j) > C\ell(i)\}}\left\{C_j^+\right\}}_{(a)}, \underbrace{\frac{143\ bytes}{rTX}}_{(b)}\right\} \tag{10}$$

Proof. The proof of Lemma 2 is similar to that of Lemma 1. Given frame f_i^q, if all the frames in a lower preemption class are shorter than 143 *bytes*, then the maximum blocking incurred by f_i^q is caused by the largest of these frames. This is captured by term (a). Otherwise, the longest non-preemptable fragment of any preemptable frame is 143 bytes length [9] and this means a maximum blocking time of $\frac{143\ bytes}{rTX}$ (where rTX is the link speed). This is captured by term (b). Therefore, a tight upper-bound on the blocking time caused by a preemptable frame of a lower preemption class to f_i^q is given by the minimum between terms (a) and (b), and the lemma follows. □

From Lemma 2, Theorem 2 derives a tight upper-bound on the lower-priority blocking incurred by any tpflow frame.

Theorem 2 ($\text{LPB}_i^{\mathcal{TP}}$). *For any tpflow flow $f_i \in \mathcal{TP}$, the maximum lower-priority blocking of any frame f_i^q (with $q \geq 1$) is given by Eq. (11).*

$$\text{LPB}_i^{\mathcal{TP}} = \max\left\{\underbrace{\max_{j \in \{lp(i) \mid C\ell(i) = C\ell(j)\}}\left\{C_j^+\right\}}_{(a)}, \underbrace{\text{PB}_i^{\mathcal{TP}}}_{(b)}\right\} \tag{11}$$

Proof. Any tpflow frame f_i^q can suffer lower-priority blocking either due to the transmission of (1) a lower-priority frame of the same preemption class or (2) a frame of a lower preemption class. In the first case, term (a) captures the largest blocking time that can be caused by a frame of the same preemption class, since these frames are served in a non-preemptive manner by design. In the second case, if the blocking is caused by a frame of a lower preemption class, we have already shown in Lemma 2 that this delay cannot exceed $\text{PB}_i^{\mathcal{TP}}$ (i.e., term (b)). Therefore, a safe upper-bound on the blocking time suffered by any tpflow frame f_i^q (with $q \geq 1$) is given by the maximum between terms (a) and (b), and the theorem follows. □

4.1.3. Lower-priority blocking for "bpflows"

Since bpflows belong to lowest preemption class by assumption, it follows that the maximum lower-priority blocking that any bpflow frame can experience is provided by the largest lower priority frame in the same class. This is given by Eq. (12).

$$\text{LPB}_i^{\mathcal{BP}} = \max_{j \in lp_{(i)}^{\mathcal{BP}}}\left\{C_j^+\right\} \tag{12}$$

Now that we have discussed in details the computation of all lower-priority blocking terms, we can proceed with the computation of the same-priority blocking – i.e., the delay due to frame(s) with the same priority as the flow under analysis.

4.2. Same-priority blocking

In the following subsections, we compute upper-bounds on the delay experienced by a frame due to the transmission of same-priority frame(s).

4.2.1. Same-priority blocking for "express flows"

Every express frame f_i^q can be blocked by all frames of the same priority that arrive before its arrival at time, say a_i^q, within the level-i busy period. Also, all the previous $q-1$ instances of f_i must be transmitted before f_i^q is transmitted and hence, contribute to the same priority blocking term. Therefore, the maximum same-priority blocking $\text{SPB}_i^{\mathcal{E}}$ that express frame f_i^q can incur is given by Eq. (13).

$$\text{SPB}_i^{\mathcal{E}}(q, a_i^q) = \sum_{j \in sp(i)} \eta_j^+(a_i^q) \cdot C_j^+ + (q-1) \cdot C_i^+ \tag{13}$$

In Eq. (13), $sp(i)$ denotes the set of all flows with the same priority as f_i. The first term computes the maximum delay due to the transmission of all frames with the same priority that arrive before a_i^q, while the second term computes the maximum delay incurred by f_i due to the transmission of all previous $q-1$ instances.

4.2.2. Same-priority blocking for "tpflow"

Similar to express frames, every tpflow frame f_i^q can be blocked by all frames with the same priority that arrive before a_i^q within the level-i busy period as well as all the previous $q-1$ instances of f_i. In addition, because f_i^q is preemptable, it can incur additional delay even after the start of its transmission. It is guaranteed uninterrupted only during the transmission of its final non-preemptable fragment. In other words, the last fragment of f_i^q must wait for all the preceding fragments of the instance to be transmitted. That is, the size of this last fragment is the minimum valid Ethernet frame size [9]. With the above-mentioned, the maximum same-priority blocking SPB_i^{tp} that f_i^q can incur is given by Eq. (14).

$$\text{SPB}_i^{\mathcal{TP}}(q, a_i^q) = \sum_{j \in sp(i)} \eta_j^+(a_i^q) \cdot C_j^+ + (q-1) \cdot C_i^+ + \left(C_i^+ - \frac{84\ bytes}{rTX}\right) \tag{14}$$

In Eq. (14), the first term computes the maximum delay due to the transmission of all frames with the same priority as f_i that arrive before a_i^q, while the second term computes the delay incurred due to the transmission of all previous $q-1$ instances of f_i. Finally, the third term computes the maximum delay due to the transmission of all non-final fragments of f_i^q.

4.2.3. Same-priority blocking of "bpflows"

The same priority blocking term for bpflows is identical to that of tpflows in Eq. (14). This is because the behavior of frames within these classes are identical in that they are transmitted in a non-preemptive manner; and they are served in a FIFO manner. This imply that a bpflow frame f_i^q can be blocked by all frames with the same priority that arrive before its arrival time a_i^q within the level-i busy period; all previous $q-1$ instances of f_i; and all its preceding fragments (i.e., the non-end ones) if it was preempted. For this class, Eq. (14) also suffices, where $\mathcal{TP}$ in superscript is replaced with $\mathcal{BP}$ in order to reflect the corresponding frame class.

4.3. Higher-priority interference

Irrespective of its class, every frame f_i cannot commence its transmission in the presence of another frame of a higher priority [9,36]. This implies that all higher priority frames that arrive before the transmission of frame f_i^q will always impact its transmission. With the aforementioned, it follows that the higher priority interference term for the three classes of frames is given by Eq. (15).

$$\mathrm{HPI}_i(\Delta t) = \sum_{j \in hp(i)} \eta_j^+(\Delta t) \cdot C_j^+ \tag{15}$$

4.4. Preemption overheads

Each preemption operation involves some overhead upon its occurrence, which is equivalent to the time taken to transmit 24 bytes of data [9]. This overhead is always added to the transmission time of the preempted frame. As such, since express flows are transmitted in a non-preemptive manner, then they do not incur any preemption overhead. Therefore, only tpflows and bpflows incur preemption overheads as they are preemptable. From an analytical standpoint, the approach adopted to compute the preemption overhead terms are identical for these two preemption classes.

Roughly speaking, the maximum preemption overheads that a preemptable frame f_i^q can incur depends solely on the *maximum number* of preemption events that can occur between its arrival time up until the transmission of the first bit of the last non-preemptable fragment. With this in mind, the maximum number of preemptions can occur in either of the following two cases:

(1) all preemptable frames transmitted between a_i^q and the complete transmission of f_i^q are preempted for the maximum number of times beyond which cutting a frame into further fragments would violate the minimum Ethernet frame size (see Eq. (7)).
(2) all possible arrivals of higher priority frames belonging to a higher preemption class occur and each of these causes a preemption.

To assess the first case, we consider a preemptable flame f_i^q transmitted within the busy period of length, say Δt, then we calculate the maximum number of times f_i^q and all other preemptable frames that are transmitted within the window of size Δt can be preempted. We can distinguish between three different types of preemptable frames, namely: (1) preemptable frames with a lower priority than f_i; (2) preemptable frames with the same priority as f_i; and finally (3) preemptable frames with a higher priority than f_i. Below, we elaborate each of these three cases.

4.4.1. Maximum number of preemptions incurred by a lower-priority preemptable frame

This maximum number of preemptions occurs when a lower-priority preemptable frame, say f_j^k, which is obstructing the transmission of f_i^q gets preempted itself a maximum number of times. Frame f_j^k can either (i) belong to a lower preemption class than f_i^q or (ii) share the same preemption class as f_i^q.

▷ **Case** (i)**.** If f_j^k belongs to a lower-preemption class, then it is preempted at most once by f_i^q or by any other frame belonging to a higher preemption class. By design, the preempted frame will not resume its transmission before all pending frames of higher preemption class (including f_i^q) have completed their transmission.

▷ **Case** (ii)**.** If f_j^k is in the same preemption class as f_i^q, then f_j^k can be preempted several times and f_i^q can only start its transmission after the complete transmission of f_j^k. By using Eq. (7), we derive the maximum number of preemptions $N_i^{\mathcal{P},lp}$ incurred a by a lower-priority preemptable frame in the same preemption class as f_i^q in Eq. (16).

$$N_i^{\mathcal{P},lp} = \max_{\{j \mid C\ell(i) = C\ell(j) \wedge i > j\}} \left\{ F_j^+ \right\} \tag{16}$$

4.4.2. Maximum number of preemptions incurred by same-priority preemptable frames

This involves the preemptions incurred by frames with the same priority as f_i^q that arrive before a_i^q within the level-i busy period as well as those suffered by f_i^q up until its last non-preemptable fragment. By using Eq. (7), the maximum number of preemptions $N_i^{\mathcal{P},sp}(q, a_i^q)$ is computed in Eq. (17).

$$N_i^{\mathcal{P},sp}(q, a_i^q) = q \cdot F_i^+ - 1 + \sum_{j \in sp(i)} \eta_j^+(a_i^q) \cdot F_j^+ \tag{17}$$

The first term of Eq. (17) computes the maximum preemption incurred by the first (q instances of f_i apart from the last fragment, while the second term computes the maximum preemption incurred by all other frames with the same priority as f_i that arrives before f_i^q within the busy-window.

4.4.3. Maximum number of preemptions incurred by higher-priority preemptable frames

This number involves the preemptions incurred by all frames with a higher priority than f_i^q that are transmitted during Δt, i.e., the transmission period of f_i^q. It is denoted by $N_i^{\mathcal{P},hp}(\Delta t)$ and occurs when all these frames get preempted for the maximum amount times, given by Eq. (18).

$$N_i^{\mathcal{P},hp}(\Delta t) = \sum_{\{j \in hp(i) \wedge j \in \mathcal{P}\}} \eta_j^+(\Delta t) \cdot F_j^+ \tag{18}$$

Putting Eqs. (16), (17), and (18) together, the maximum number of preemptions incurred by the preemptable frames transmitted during Δt is given by Eq. (19).

$$N_i(q, a_i^q, \Delta t) = N_i^{\mathcal{P},lp} + N_i^{\mathcal{P},sp}(q, a_i^q) + N_i^{\mathcal{P},hp}(\Delta t) \tag{19}$$

With this Eq. (19), we have everything we need to derive an upper-bound on the maximum preemption overhead suffered by any preemptable frame f_i^q, as formalized by Theorem 3.

Theorem 3 ($\mathrm{PO}_i^{\mathcal{P}}(\Delta t, q, a_i^q)$)**.** *For any preemptable flow $f_i \in \mathcal{P}$, an upper-bound on the maximum preemption overhead incurred by frame f_i^q (with $q \geq 1$), arrived at time a_i^q within the busy-period Δt, is given by Eq. (20).*

$$\mathrm{PO}_i^{\mathcal{P}}(\Delta t, q, a_i^q) = \underbrace{\frac{24\ bytes}{rTX}}_{(a)} \times \min\left\{ \underbrace{\left(\sum_{j \in \{\mathcal{T} \mid C\ell(j) < C\ell(i)\}} \eta_j^+(\Delta t) \right)}_{(b)}, \underbrace{N_i(q, a_i^q, \Delta t)}_{(c)} \right\} \tag{20}$$

Proof. The maximum preemption overhead is reached in one of the following two cases: (Case 1) All the preemptable frames incur the maximum possible number of preemptions; or (Case 2) All arriving frames belonging to a higher preemption classes cause a preemption. For the first case, Eq. (19), which is captured in term (c) provide an upper-bound. For the situation described in Case 2, a computation of the maximum number of arrivals from frames in a higher preemption class is captured by term (b). Therefore, the actual number of preemptions cannot exceed the minimum of these two terms. Now, because each preemption operation generates an overhead corresponding to term (a), the theorem follows. □

4.5. Worst-case queuing delay

In Section 4, we discussed the individual components of the worst-case queuing delay. For any frame f_i^q that arrives at time a_i^q within the busy period, we obtain an upper-bound $Q_i(q, a_i^q)$ on this factor by summing up all these terms, as formally stated in Eq. (21).

$$Q_i(q, a_i^q) = \begin{cases} \text{LPB}_i^{\mathcal{E}} + \text{SPB}_i^{\mathcal{E}}(q, a_i^q) + \text{HPI}_i(Q_i(q, a_i^q)) & \text{if } f_i \in \mathcal{E}; \\ \text{LPB}_i^{\mathcal{TP}} + \text{SPB}_i^{\mathcal{TP}}(q, a_i^q) + \text{HPI}_i(Q_i(q, a_i^q)) + \text{PO}_i^{\mathcal{P}}(Q_i(q, a_i^q)) & \text{if } f_i \in \mathcal{TP}; \\ \text{LPB}_i^{\mathcal{BP}} + \text{SPB}_i^{\mathcal{BP}}(q, a_i^q) + \text{HPI}_i(Q_i(q, a_i^q)) + \text{PO}_i^{\mathcal{P}}(Q_i(q, a_i^q)) & \text{if } f_i \in \mathcal{BP} \end{cases} \tag{21}$$

Note that Eq. (21) defines a fixed-point iterative process since $Q_i(q, a_i^q)$ appears on both sides of the equation. As such, a valid solution for every frame f_i^q is obtained by starting the fixed-point algorithm with the base value C_i^+. The algorithm stops as soon as two consecutive values of $Q_i(q, a_i^q)$ are identical or the relative deadline associated to flow f_i is exceeded. In the latter case, the timing requirement of f_i is violated and there is no valid solution.

4.6. Worst-case traversal time

For every frame f_i^q arriving at time a_i^q, the discussions conducted in the previous sections command that its worst-case traversal time ($\text{WCTT}_i(q, a_i^q)$) is obtained by combining Eqs. (1) and (4), where $Q_i^+(q)$ (in Eq. (1)) and $\delta_i^-(q)$ (in Eq. (4)) are substituted by $Q_i(q, a_i^q)$ and a_i^q, respectively. Formally, $\text{WCTT}_i(q, a_i^q)$ is reported in Eq. (22).

$$\text{WCTT}_i(q, a_i^q) = \begin{cases} Q_i(q, a_i^q) + C_i^+ - a_i^q & \text{if } f_i \in \mathcal{E}; \\ Q_i(q, a_i^q) + \dfrac{84bytes}{rTX} - a_i^q & \text{if } f_i \in \mathcal{TP}; \\ Q_i(q, a_i^q) + \dfrac{84bytes}{rTX} - a_i^q & \text{if } f_i \in \mathcal{BP} \end{cases} \tag{22}$$

Finally, the worst-case traversal time for flow f_i within the level-i busy period, denoted by WCTT_i^+, is obtained by computing the maximum of all the $\text{WCTT}_i(q, a_i^q)$, where $1 \le q \le q_i^+$ and q_i^+ is the last frame of f_i in the level-i busy period. Formally, this is expressed in Eq. (23).

$$\text{WCTT}_i^+ = \max_{1 \le q \le q_i^+} \left\{ \text{WCTT}_i(q, a_i^q) \right\} \tag{23}$$

This equation concludes the computation of WCTT_i^+ for flow f_i within a switch node (i.e., the local analysis). Therefore, the overall traversal time for flow f_i (i.e., the global analysis) is obtained by summing up these values at all the individual switch nodes along its transmission path. Here, the level-i busy period is a time interval $[\sigma, \mu]$ within which only frames belonging to $hep(i)$ (except the first frame, which is generated from flow $f_j \in lp(i)$ with the longest non-preemptable fragment) are transmitted throughout $[\sigma, \mu]$, but no frame belonging to $hep(i)$ is transmitted in $[\sigma - \epsilon, \sigma)$ or $(\mu, \mu + \epsilon]$ for any arbitrary small $\epsilon > 0$.

In order to perform the analysis, the scenario leading the longest level-i busy period for each flow f_i was considered, i.e., the scenario where either (1) a frame f_j^k in $C\ell(i) < C\ell(j)$ with the longest non-preemptable fragment; or (2) a frame f_j^k in $C\ell(i) = C\ell(j)$ with the largest size and a lower priority than f_i, was released just before f_i; all the flows in $hep(i)$ release a frame at the same time as f_i and finally, all future frames are released as soon as is legally permitted to do so.

5. Evaluation

In this section we perform the evaluation of the proposed approach. Specifically, we consider two realistic use-cases from the automotive domain and evaluate the safety & tightness of the proposed analysis.

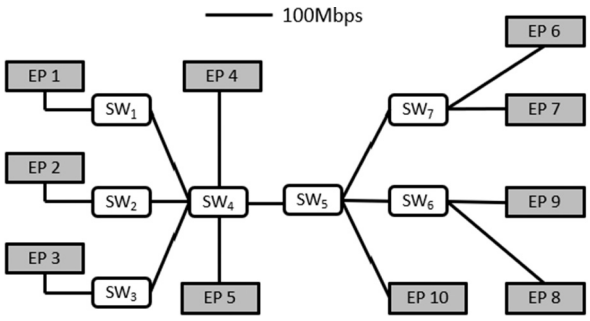


Fig. 7. Network topology.

Table 2
Flow properties.

ID	Class	Src.	Dst.	Period (µs)	Deadline (µs)	Size (bytes)
1.	Express	EP 7	EP3	5000	150	200
2.	Express	EP 6	EP 1	10 000	200	250
3.	Tpflow	EP 2	EP 9	5000	500	300
4.	Tpflow	EP 3	EP 8	5000	500	400
5.	Bpflow	EP 4	EP 10	1000	–	1300
6.	Bpflow	EP 1	EP 8	1000	–	1300
7.	Bpflow	EP 9	EP 5	1000	–	1500

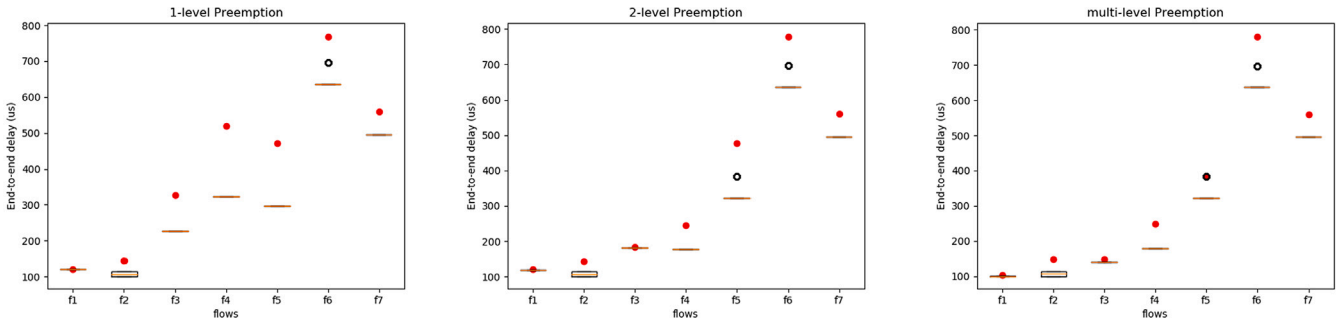
5.1. Report on use-case 1

▷ **Setup.** In this use-case, we consider a realistic network topology from the automotive domain, consisting of ten End Points EPs (a.k.a. Engine Control Units or ECUs) and seven full-duplex preemption enabled TSN switches $\text{SW}_1, \text{SW}_2, \ldots, \text{SW}_7$ displayed as illustrated in Fig. 7. We considered seven flows – $f_1, f_2, \ldots, f_7$ – where f_1 and f_2 are express; f_3 and f_4 are tpflows; and the remaining flows are bpflows. The name and make of the system are protected under an NDA. However, the specifications of the flows are provided in Table 2 and they are in the same range as those of the use-case presented by Alderisi et al. [37]. We recall that flows are ordered according to their priorities, i.e., the smaller the subscript of a flow, the higher its priority. Three batches of analyses together with the associated simulations by using NeST-iNg [38] to evaluate their tightness are conducted: (*a*) All flows are transmitted in under the 1-level preemption scheme (i.e., only express flows can preempt other flows); (*b*) all flows are transmitted under the 2-level preemption scheme (i.e., express frames can preempt all other frames; and tpflows can preempt bpflows); finally (*c*) all flows are transmitted in a fully preemptive manner (i.e., any higher priority frame can preempt any lower priority frame).

▷ **Results and discussion.** Here, we report the results obtained from the experiments. We considered the safety and tightness of the proposed analysis. Also, we reported the behavior of the network with respect to "each additional preemption level" and the "maximum frame size in each preemption class".

▷◁ **On the tightness of the proposed analysis**. Fig. 8 compares for each flow the maximum measured end-to-end delays – mWCTT – (see the yellow box plots) against the analytical WCTT bounds (see the red dots). We can notice that all mWCTT fall below the corresponding WCTT when adopting the 1-level preemption scheme (see Fig. 8(a)), the 2-level preemption scheme (see Fig. 8(b)), and finally, the fully preemptive scheme (Fig. 8(c)).

- Under the 1-level scheme, the observed gaps between the WCTT bounds and the mWCTT for express flows f_1 and f_2 are 1.6% (WCTT : 120 µs, mWCTT : 118 µs) and 16.25% (WCTT : 144 µs, mWCTT : 119.7 µs), respectively. This pessimism comes from the fact that f_1 and f_2 share the same path as f_7. The gaps



(a) 1-level preemption scheme. (b) 2-level preemption scheme. (c) fully-preemptive scheme.

Fig. 8. Observed end-to-end delay from simulation. Red dots represent WCTT bounds, black are outliers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

between WCTT and mWCTT for tpflows f_3 and f_4 are 30.92% (WCTT : 328 µs, mWCTT : 226.56 µs) and 43.05% (WCTT : 520 µs, mWCTT : 296.16 µs), respectively. This is due to the fact that tpflows share the same path as bpflows and can be blocked for long periods of time.

- Under the 2-level scheme, both the WCTT and mWCTT for express flows remain the same. However, there is a significant improvement in the performance of tpflows in comparison to the 1-level scheme. The maximum observed delay for f_3 and f_4 reduced by 21.17% (from 226.56 µs to 178.58118 µs) and 38.89% (from 296.16 µs to 180.98 µs), respectively. Also, the gaps between WCTT and mWCTT for f_3 and f_4 reduced by 2.94% (WCTT : 184 µs, mWCTT : 178.58 µs) and 16.25% (WCTT : 244 µs, mWCTT : 180.98 µs), respectively.
- Under the fully-preemptive scheme, there is further improvement in performance of express flows and tpflows. Specifically, the mWCTT values of f_1 and f_3 reduced by 14.1% (from 108 µs to 101.3 µs) and 21.09% (from 178.58 µs to 140.9 µs), respectively, as compared to the 2-level scheme. The gaps between WCTT and mWCTT for flows f_1, f_2, f_3, and f_4 under the fully-preemptive scheme are 2.7% (WCTT : 104 µs, mWCTT : 101.3 µs), 17.77% (WCTT : 148 µs, mWCTT : 119.7 µs), 4.79% (WCTT : 148 µs, mWCTT : 140.9 µs), and 26.22% (WCTT : 248 µs, mWCTT : 180.98 µs), respectively. It is worth noticing the slight degradation in the performance of f_2 and f_4 (1.67% and 1.10%, respectively). This is due to the preemption overhead incurred by additional preemption operations. From this observation, it follows that there is a trade-off to find between the number of preemption levels allowed for the flow transmissions and the additional overheads that each new preemption level brings on the table. Note that we can define only up to six intermediate preemption levels since Ethernet offers a maximum of eight priority classes. We leave this pending question of the optimal number of preemption-levels for frame transmissions for future works.

⊳⊲ **On the impact of each extra preemption level.** From Fig. 9, we observe an improvement in the responsiveness of the tpflows when an extra preemption level is added to the frame transmission scheme. Specifically, this improvement reaches 43.9% (from 328 µs to 184 µs) and 53.07% (from 520 µs to 244 µs) for f_3 and f_4, respectively. The rationale behind this trend can be cast as follows. With the introduction of the extra preemption level, f_3 and f_4 become protected from any eventual long blocking time associated to the transmission of f_6 and f_5 along their paths. Also, we note that the overhead induced by this extra preemption level is negligible: the WCTT f_1 and f_2 remains the same for both the 1-level and 2-level preemption schemes. The same condition roughly holds true for bplfows (i.e., flows f_5, f_6, and f_7). Here, we note a cumulative performance degradation of 1.78%, which is negligible for the adopted use-case (a degradation from 472 µs to 476 µs and 768 µs to 778 µs for f_5 and f_6, respectively). We also note that some mWCTT values for f_5 and f_6 (the black circles) fall outside the whiskers of the box plot. These outliers are the cases where an instance of a flow experiences an unusually high (or low) interference.

Still in Fig. 9, we observe an average improvement of 9.6% in the WCTT of tpflows when moving from a 2-level preemption scheme to a fully preemptive approach. This suggests that the performance improvement is not linear despite the benefits brought about by each extra preemption level. On the downside, each preemption level involves extra hardware implementation overheads, which may turn out not to be negligible, unfortunately. This situation gives rise to an open question: What is the optimal trade-off in terms of preemption-level scheme to adopt for the flow transmission as the performance gain brought about by enabling each extra-preemption level is diminished by the hardware implementation overheads it brings? We are aware of this, but we leave the question out for future work as it deserves a thorough investigation.

⊳⊲ **On the impact of the frame sizes in each preemption class.** From the use-case setup, the maximum byte size of tpflows is reasonably small in comparison to that of the bpflows. We vary this parameter from 400 to 1200 bytes in order to investigate its effect on the WCTT for each tpflow. The results are reported in Fig. 10.

From Fig. 10(a), we observe that the gains on the WCTT obtained from the multi-level preemption scheme over the 1-level preemption approach diminishes with an increase in the maximum frame size. Specifically, the improvement drops from 43.9% (328 µs to 184 µs) to only 7.82% (614 µs to 566 µs) for f_3 and from 53.07% (520 µs to 244 µs) to 32.3% (972 µs to 658 µs) for f_4. This significant impact of the frame size on the performance of f_3 can be explained by the increasing blocking time it suffers as the highest priority tpflow. We recall that frames in the same preemption class are transmitted in a non-preemptive manner. Consequently, careful attention must be paid to the maximum possible frame size of each tpflow during the flow-to-preemption-class mapping at design time. It is worth mentioning that the degradation is less severe when assuming a full preemptive scheme as illustrated in Fig. 10(b). Here, the performance gains over the 1-level preemption scheme dropped from 54.87% (328 µs to 148 µs) to 33.55% (614 µs to 408 µs) for flow f_3 and from 52.30% (520 µs to 248 µs) to 36.62% (972 µs to 616 µs) for flow f_4.

5.2. Report on use-case 2

To further evaluate the safety of the WCTT bounds, we consider another use-case scenario with a larger network and more flows provided by Renault and borrowed from Migge et al. [39]. The adopted network topology is replicated in Fig. 11.

⊳ **Setup.** The network comprises 5 full-duplex Ethernet switches and 14 nodes: 4 cameras (CAMs), 4 displays (DSPs), 3 control units (ECUs)

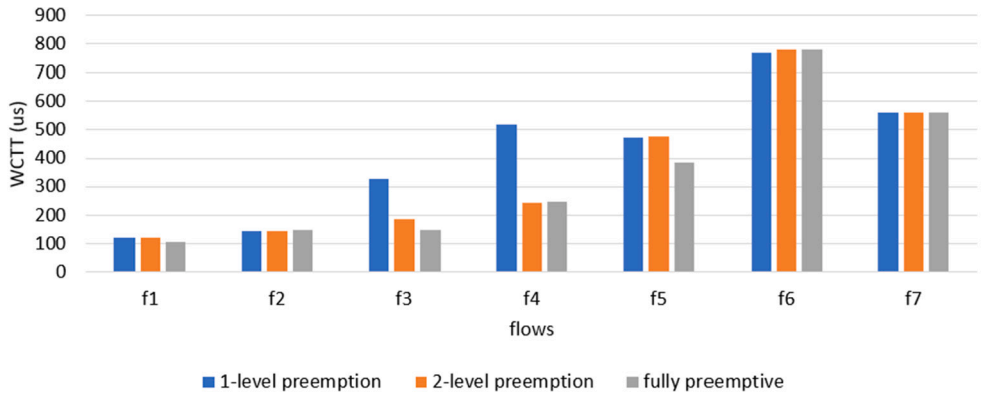
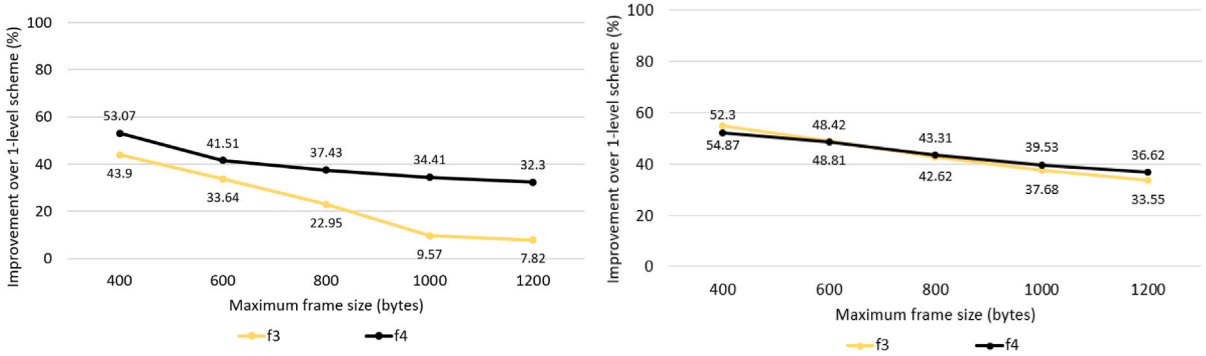


Fig. 9. WCTT for each flow under 1-level preemption, 2-level preemption and fully preemptive schemes.



(a) two-level preemption scheme.

(b) fully-preemptive scheme.

Fig. 10. Performance improvement w.r.t. maximum tpflow frame size.

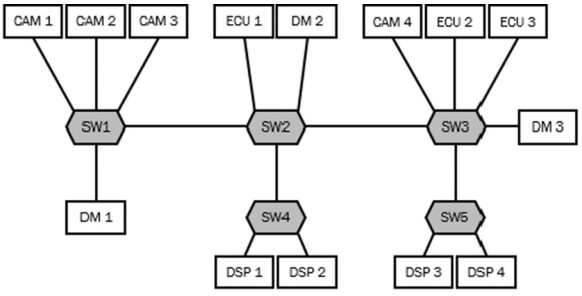


Fig. 11. Network topology.

and 3 (functional) domain masters (DMs). The data transmission rate is 100Mbit/s on all links. Assuming this setting, we consider a fully preemptive scheme, i.e., the preemption class of each flow is also its priority. The traffic specification consists of a total of 41 flows as shown in Table 3.

▷ **Results and discussion.** Fig. 12 presents the results obtained from the experiments. The numerical values are reported in Table 4. From Fig. 12, it is observed that the mWCTT of all flows are less than the computed WCTT bounds. This also demonstrates the safety of the analysis presented in this work. The mWCTT values of Audio flows – which are the highest priority flows – are lower than the WCTT values by an average of 65.92%. The mWCTT values for Command and Control Traffic – the highest priority traffic after Audio traffic – are lower than the WCTT by an average of 42.84%. And finally, for the lower priority flows (Video and Best Effort flows), the mWCTT values are lower than the WCTT by an average of 36.53%. We noticed that the gap between the mWCTTs and the WCTTs are lowest for flows with the lowest priorities (Best Effort flows) and, generally speaking, the gap between mWCTT and WCTT values increases as the priority of flows increases. This is because lower priority flows are more likely to suffer the maximum possible interferences as accounted for in the analysis. We plan to further investigate this phenomenon with the aim of reducing the pessimism in the WCTT values of higher priority flows and make the analysis as tight as possible.

Table 3
Prototype flow specification with the characteristics and performance requirements for each traffic class.

Audio streams	– 8 streams – 128 and 256 byte frames – up to sub-10 ms period and deadline – soft deadline constraints
Command and Control (C&C)	– 11 streams 256 to 1024 byte frames – up to sub-10 ms periods and deadlines – deadline constraints (hard)
Video streams	– 2 ADAS + 6 Vision streams – up to 30*1446 byte frame each 16 ms (60FPS) or each 33 ms (30FPS) – 10 ms (ADAS) or 30 ms deadline (Vision) – hard and soft deadline constraints
Best-effort: file & data transfer, diagnostics	– 11 streams including TFTP traffic pattern – up to 0.2 ms period – both throughput guarantees (up to 20Mbits per stream) and deadline constraints (soft)

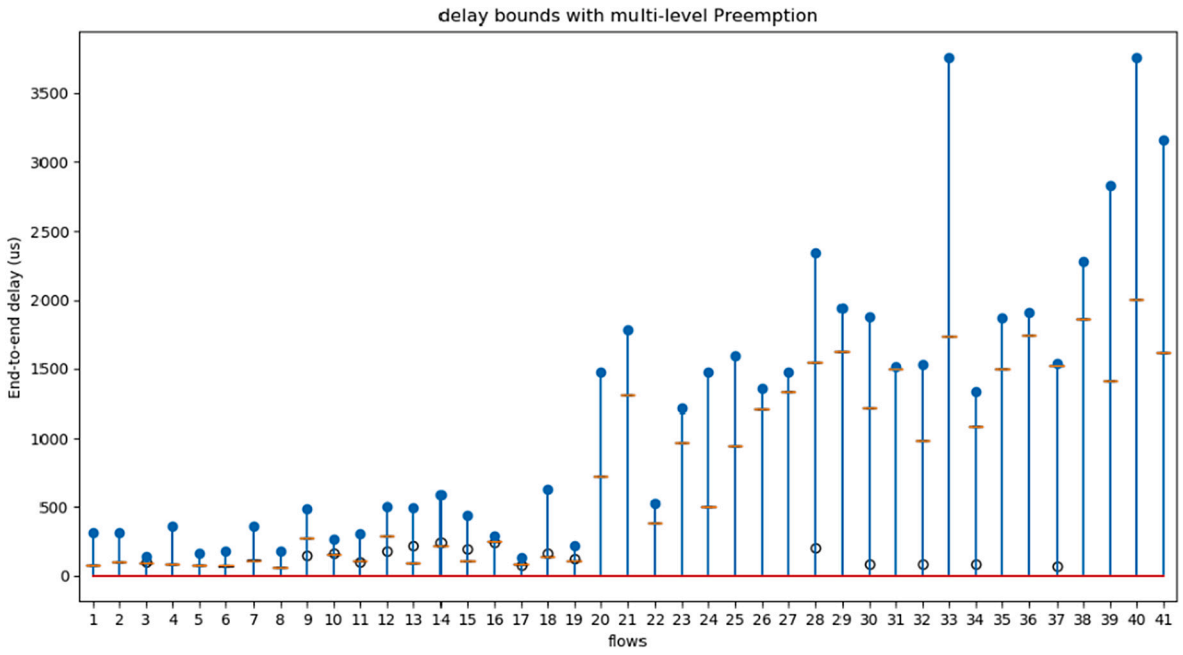


Fig. 12. Observed end-to-end delays from simulation for the Renault use-case (in yellow box plots). The blue lines represent the WCTT bounds and the circles represent outlier values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Results: Renault use-case.

mWCTT and WCTT for the Renault Automotive Use-Case								
ID	Src.	Dest.	Size (bytes)	Priority	period (µs)	Deadline (µs)	mWCTT (µs)	WCTT (µs)
1	DM2	DSP1	210	1	5000	5000	72.5	310
2	DM1	DSP1	200	1	5000	5000	95.8	310
3	DM1	DM3	189	0	5000	5000	97.5	136
4	DM2	DSP2	199	1	5000	5000	87.0	363
5	ECU1	DSP2	179	0	5000	5000	77.5	163
6	DM2	DSP2	159	0	5000	5000	75.8	177
7	DM2	DSP2	225	0	5000	5000	115.4	362
8	DM2	DSP1	138	0	5000	5000	59.5	177
9	ECU3	DSP1	240	2	10 000	10 000	304.0	490
10	ECU3	DM1	170	3	10 000	10 000	165.2	263
11	ECU3	DM3	239	3	10 000	10 000	112.6	204
12	ECU2	DSP2	200	2	10 000	10 000	322.0	499
13	ECU1	DSP2	234	2	10 000	10 000	252.0	492
14	ECU1	DSP2	214	3	10 000	10 000	247.0	589
15	ECU1	DSP1	210	2	10 000	10 000	214.0	442
16	ECU1	DM1	190	3	10 000	10 000	251.2	289
17	ECU3	DM3	210	2	10 000	10 000	90.2	133
18	ECU3	DSP1	242	3	10 000	10 000	167.5	630
19	ECU1	DM1	250	2	10 000	10 000	121.8	215
20	CAM4	DSP4	1446	5	10 000	10 000	723.2	1475
21	CAM1	DSP1	1446	5	10 000	10 000	1304.9	1783
22	CAM4	DSP4	1446	4	10 000	10 000	382.6	525
23	CAM1	DSP3	1446	4	10 000	10 000	965.8	1217
24	CAM2	DSP2	1446	4	10 000	10 000	505.9	1477
25	CAM1	DSP2	1446	4	10 000	10 000	941.1	1597
26	CAM4	DSP3	1446	5	10 000	10 000	1208.83	1355
27	CAM4	DSP4	1446	5	10 000	10 000	1329.6	1475
28	CAM1	DM3	1446	6	10 000	10 000	1529.7	2341
29	ECU2	DM1	1446	7	10 000	10 000	1621.1	1939
30	CAM1	DM2	1446	6	10 000	10 000	1500.6	1880
31	CAM2	DM1	1446	7	10 000	10 000	1499.8	1518
32	CAM2	DM2	1446	6	10 000	10 000	1220	1528
33	CAM2	DM3	1446	7	10 000	10 000	1737.5	3757
34	ECU2	DM2	1446	7	10 000	10 000	1076.2	1330
35	CAM3	DM3	1446	6	10 000	10 000	1494.9	1867
36	ECU3	DM1	1446	7	10 000	10 000	1742.4	1906
37	CAM2	DM1	1446	7	10 000	10 000	1509.7	1518
38	CAM4	DM1	1446	7	10 000	10 000	1863.7	2279
39	CAM2	DM2	1446	7	10 000	10 000	1410.7	2826
40	CAM2	DM3	1446	7	10 000	10 000	2005.6	3757
41	CAM3	DM3	1446	7	10 000	10 000	1616.2	3157

6. Related work

The Network Calculus (NC) [40]; the Trajectory Approach (TA) [41]; and the Compositional Performance Analysis (CPA) [28] have all been used as established techniques to provide timing guarantees for real-time Ethernet flows. In this section we report only the most significant contributions related to our topic discussed in this paper.

In NC, a so-called *arrival curve* and *service curve* are used to model the arrival of flows and the transmission bandwidth at a switch output port, respectively. To the best of our knowledge, only a hand-full of contributions using this approach are available in the literature on TSN related issues. Zhao et al. [42] provided a worst-case latency analysis for IEEE 802.1Qbv networks. To this end, they assumed that the Gate Control List (GCL) and the priority assignment configurations are given. They validated the performance of their approach by using both synthetic and real-world use-cases in terms of scalability and effect of GCL overlapping characteristics on individual flows. The authors, in another work, also provided a latency analysis for AVB traffic in TSN [43]. However, their analyses only target non-preemptive TSN networks and lefts the preemptive case unanswered, unfortunately.

In TA, Martin and Pinet [41] investigated the highest number of frames that share the same trajectory as it is a potential source of delay for each of these flows. The adopted approach proceeds "backwards", i.e., from the receiver node to the source node. In another context, this approach as been used by Bauer et al. in [44] for the timing analysis of AFDX with strict priority, non-preemptive flows transmitted by following a FIFO scheduling strategy. In the work, the TA has been further enhanced by exploring the basic idea according to which flows sharing a common link cannot arrive at the same time at a switch. Li et al. [45] have proven the result to be optimistic and have corrected the flaw. Nonetheless, the analysis still considers only non-preemptive frame transmission.

Cao et al. [46,47] introduced a so-called *eligibility interval* approach for the timing analysis of Ethernet Audio Video Bridges (AVB) [48] networks with Credit-Based Shapers (CBS). In this approach, the worst-case performance of a flow is examined when a flow has some pending payload to transmit and a non-negative transmission credit. This approach was proven to be tight for AVB networks and subsequently extended for the timing guarantees of AVB flows in standard TSN [49]. Thangamuthu et al. [50] proposed a worst-case performance analysis for Switched Ethernet with Burst-Limiting Shaper (BLS); Peristaltic Shaper (PS); and Time-Aware Shaper (TAS) for TSN, but concluded that only the TAS can schedule control traffic within the maximum delay imposed by the Standards. In a nutshell, PS imposes an organization on the reception and transmission of frames at each node by following well-defined time intervals. Here, every time window is divided into intervals (even and odd intervals) and all frames received within an interval are transmitted in the next interval only. BLS is an extension of CBS introduced by the AVB Standard [48], wherein a transmission budget or "credit" is assigned to each traffic class to control its transmission rate and mitigate against burst traffic. Finally TAS defines a time-triggered gate control mechanism for TSN flows. Here, each flow class has a guaranteed transmission slot in a cyclic and pre-computed dispatch schedule.

CPA has been used extensively for the timing behavior of Ethernet flows [32,36]. It uses the so-called *level-i busy period* approach, activated by the so-called critical instant, to investigate the worst-case response time of each real-time flow [30]. CPA was employed for the timing behavior of Switched Ethernet in [31] and for Ethernet AVB in [33]. The analysis for Switched Ethernet was improved by Thiele et al. [36] to tighten the worst-case response time bounds of each flows by up to 80%, then the same authors exploited the FIFO nature of Switched Ethernet transmission [32] to reduce the interference estimation in frame transmissions and achieved about 30% latency improvement over the state-of-the-art CPA at the time. Still by using CPA, Thiele et al. [51] and Thiele and Ernst [34] proposed the worst-case analysis for TSN with PS and BLS, respectively. Both contributions focused only the shapers, leaving frame preemption concerns beyond the scope of their work. On another front, Thiele and Ernst [9] used CPA to provide worst-case guarantees for both Standard Ethernet and IEEE 802.1Qbv when frame preemption in enabled. Lo Bello et al. [52] also provide a schedulability analysis for IEEE 802.1Qbv networks with preemption support. In these works, the authors address only the traditional 1-level preemption scheme, as defined in the standards. Recently, Ojewale et al. [26,27] promoted the multi-level preemption approach for IEEE 802.1Qbu networks in order to circumvent the limitations of the 1-level preemption scheme. The authors first investigated the feasibility and advantages of multi-level preemptions in time-sensitive Ethernet networks. Then, they focused on its feasibility and implementation requirements in details. Recently, Mladen et al. [53] provided an improved implementation approach for the multi-level preemption scheme. However, their works stop short of providing a formal analysis of the worst-case performance guarantees under the proposed scheme. We fill this gap in this paper by providing the worst-case traversal time analysis of TSN frames under the multi-level preemption assumption, which is the first contribution in this direction to the best of our knowledge.

7. Conclusion

In this paper, we advocated for a multi-level preemption scheme for TSN frame transmission in order to circumvent the limitations of the traditional 1-level preemption scheme as specified in the standards. We provided a formal timing guarantees for each flow under such a scheme by using a CPA based approach. Using a realistic automotive use-case, we assessed the performance improvements in terms of worst-case traversal time (WCTT) over the 1-level scheme. From our results, the multi-level preemption scheme shows an improvement up to 53.07% in the WCTT guarantee for preemptable time-sensitive frames. Then, we demonstrated the tightness of the analysis with another automotive use-case from Renault. We concluded that a careful attention must be paid to the maximum size of each of these flows during the flow-to-preemption-class assignment at design time. Also, an interesting discussion was conducted on the performance improvement brought about by each extra preemption level in the frame transmission scheme and we showed that the trend was not linear on one hand; and each extra preemption level involves additional hardware implementation overheads, which may not be negligible, unfortunately. In a near future, we seek to investigate: (1) the optimal preemption-level scheme that will prove to be the perfect trade-off between the performance that each additional preemption level brings and the overhead of enabling this level; (2) efficient flow priority assignment strategies in order to further improve the responsiveness of preemptable time-sensitive flows; and finally (3) the interoperability of multi-level preemption with other TSN mechanisms.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by the project Safe Cities - Inovação para Construir Cidades Seguras, Portugal, ref. POCI-01-0247-FEDER-041435, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020); also by National Funds, Portugal through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020).

References

[1] Bosch, CAN Specification, Vol. 50, Robert Bosch GmbH, Postfach, 1991.

[2] J.W. Specks, A. Rajnák, LIN-protocol, development tools, and software interfaces for local interconnect networks in vehicles, VDI-Berichte (2000) 227–250.

[3] T. Pop, P. Pop, P. Eles, Z. Peng, A. Andrei, Timing analysis of the FlexRay communication protocol, Real-Time Syst. 39 (1–3) (2008) 205–235.

[4] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, S. Varadarajan, TTEthernet dataflow concept, in: Eighth IEEE International Symposium on Network Computing and Applications, 2009, pp. 319–322.

[5] F. Brajou, P. Ricco, The Airbus A380-an AFDX-based flight test computer concept, in: AUTOTESTCON, 2004, pp. 460–463.

[6] E. Schemm, SERCOS to link with ethernet for its third generation, Comput. Control Eng. 15 (2) (2004) 30–33.

[7] D. Jansen, H. Buttner, Real-time Ethernet: the EtherCAT solution, Comput. Control Eng. 15 (1) (2004) 16–21.

[8] J. Feld, PROFINET-scalable factory communication for all applications, in: IEEE International Workshop on Factory Communication Systems, 2004, pp. 33–38.

[9] D. Thiele, 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] S. Kamal, S. Al Mubarak, B. Scodova, P. Naik, P. Flichy, G. Coffin, et al., IT and OT convergence-Opportunities and challenges, in: SPE Intelligent Energy Int. Conference and Exhibition, 2016, pp. 1–10.

[11] P. Pop, M.L. Raagaard, M. Gutierrez, W. Steiner, Enabling fog computing for industrial automation through time-sensitive networking (TSN), IEEE Com. Stand. Mag. 2 (2) (2018) 55–61.

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

[13] W.K. Jia, G.H. Liu, Y.C. Chen, Performance evaluation of IEEE 802.1Qbu: Experimental and simulation results, in: 38th Annual IEEE LCN, 2013, pp. 659–662.

[14] IEEE, IEEE standard for ethernet amendment 5: Specification and management parameters for interspersing express traffic, 2016, pp. 1–58, Std 802.3br-2016.

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

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

[17] IEEE, IEEE standard for local and metropolitan area networks — Bridges and bridged networks amendment 24: Path control and reservation IEEE computer society, ISBN: 9781504407724, 2015, IEEE Std 802.1Qca-2015.

[18] IEEE, 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, 2018, pp. 1–214, P802.1Qcc/D2.2.

[19] A. Nasrallah, A.S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, H. ElBakoury, Ultra-low latency (ULL) networks: The IEEE TSN and IETF detnet standards and related 5G ULL research, IEEE Commun. Surv. Tutor. 21 (2018) 88–145.

[20] IEEE, Time-sensitive networking task group. [online]. Available: http://www.IEEE802.org/1/pages/tsn.html.

[21] 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, 2015, pp. 1–50, P802.1Qbu/D3.1, September 2015.

[22] IEEE, IEEE standard for local and metropolitan area network–bridges and bridged networks, 2018, pp. 1–1993, Std 802.1Q-2018.

[23] L.L. Bello, W. Steiner, A perspective on IEEE time-sensitive networking for industrial communication and automation systems, Proc. IEEE 107 (6) (2019) 1094–1120.

[24] T. Park, S. Samii, K.G. Shin, Design optimization of frame preemption in real-time switched Ethernet, in: IEEE DATE, 2019, pp. 420–425.

[25] W. Steiner, An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks, in: 31st IEEE Real-Time Systems Symposium, 2010, pp. 375–384.

[26] M.A. Ojewale, P. Meumeu Yomsi, G. Nelissen, On multi-level preemption in ethernet, in: WiP Session, ECRTS, 2018, pp. 16–18.

[27] M.A. Ojewale, P. Meumeu Yomsi, B. Nicolić, Multi-level preemption in TSN: feasibility and requirements analysis, in: 23rd IEEE Int. Symposium on Real-Time Distributed Computing, 2020, pp. 1–9.

[28] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, R. Ernst, System level performance analysis–the SymTA/S approach, Comput. Digit. Tech. 152 (2) (2005) 148–166.

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

[30] R. Hofmann, L. Ahrendts, R. Ernst, S. Ha, J. Teich, CPA: Compositional performance analysis, 2017.

[31] J. Rox, R. Ernst, Formal timing analysis of full duplex switched based ethernet network architectures, 2010, SAE International.

[32] D. Thiele, P. Axer, R. Ernst, Improving formal timing analysis of switched ethernet by exploiting FIFO scheduling, in: DATE, 2015, p. 41.

[33] J. Diemer, D. Thiele, R. Ernst, Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching, in: 7th IEEE Int. Symposium on Industrial Embedded Systems, 2012, pp. 1–10.

[34] D. Thiele, R. Ernst, Formal worst-case timing analysis of Ethernet TSN's burst-limiting shaper, in: DATE, 2016, pp. 187–192.

[35] N. Finn, Time-sensitive and deterministic networking whitepaper, 2017, pp. 1–24.

[36] D. Thiele, P. Axer, R. Ernst, J.R. Seyler, Improving formal timing analysis of switched ethernet by exploiting traffic stream correlations, in: Int. Conf. on Hw/Sw Codesign and Sys. Syn., ACM, 2014, p. 15.

[37] G. Alderisi, G. Iannizzotto, L.L. Bello, Towards IEEE 802.1 Ethernet AVB for advanced driver assistance systems: A preliminary assessment, in: IEEE 17th ETFA, 2012, pp. 1–4.

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

[39] J. Migge, J. Villanueva, N. Navet, M. Boyer, Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks, in: Proc. Embedded Real-Time Software and Systems, ERTS 2018, 2018.

[40] F. Reimann, S. Graf, F. Streit, M. Glaß, J. Teich, Timing analysis of Ethernet AVB-based automotive E/E architectures, in: 18th IEEE ETFA, 2013, pp. 1–8.

[41] S. Martin, P. Minet, Worst case end-to-end response times of flows scheduled with FP/FIFO, in: ICNICONSMCL'06, 2006, pp. 1–7.

[42] L. Zhao, P. Pop, S.S. Craciunas, Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus, IEEE Access 6 (2018) 41803–41815.

[43] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, M. Boyer, Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus, IEEE Trans. Ind. Electron. (2020).

[44] H. Bauer, J.-L. Scharbarg, C. Fraboul, Applying trajectory approach with static priority queuing for improving the use of available AFDX resources, Real-Time Syst. 48 (1) (2012) 101–133.

[45] X. Li, O. Cros, L. George, The Trajectory approach for AFDX FIFO networks revisited and corrected, in: 20th IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications, 2014, pp. 1–10.

[46] J. Cao, P.J. Cuijpers, R.J. Bril, J.J. Lukkien, Tight worst-case response-time analysis for ethernet AVB using eligible intervals, in: IEEE WFCS, 2016, pp. 1–8.

[47] J. Cao, P.J. Cuijpers, R.J. Bril, J.J. Lukkien, Independent yet tight WCRT analysis for individual priority classes in ethernet AVB, in: 24th RTNS, ACM, 2016, pp. 55–64.

[48] IEEE, IEEE standard for local and metropolitan area networks–audio video bridging (AVB) systems, 2011, pp. 1–45, Std 802.1BA-2011.

[49] D. Maxim, Y.-Q. Song, Delay analysis of AVB traffic in time-sensitive networks (TSN), in: 25th Int. Conf. on Real-Time Networks and Systems, ACM, 2017, pp. 18–27.

[50] S. Thangamuthu, N. Concer, P.J. Cuijpers, J.J. Lukkien, Analysis of ethernet-switch traffic shapers for in-vehicle networking applications, in: DATE, EDA Consortium, 2015, pp. 55–60.

[51] D. Thiele, R. Ernst, J. Diemer, Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers, in: IEEE Vehicular Networking Conference, 2015, pp. 251–258.

[52] L.L. Bello, M. Ashjaei, G. Patti, M. Behnam, Schedulability analysis of time-sensitive networks with scheduled traffic and preemption support, Journal of Parallel and Distributed Computing 144 (2020) 153–171.

[53] M. Knezic, M. Kovacevic, Z. Ivanovic, Implementation aspects of multi-level frame preemption in TSN, in: 25th IEEE International Conference on Emerging Technologies and Factory Automation, Vol. 1, ETFA, 2020, pp. 1127–1130.

Mubarak Adetunji Ojewale is a Student Researcher at CISTER Research Centre and is pursuing a Ph.D. degree in Electrical and Computer Engineering at the University of Porto, Portugal. Mubarak holds a bachelor's degree in Computer Science from the University of Ibadan, Nigeria, with First-Class honors in 2014 and also a Masters' Degree in Computer Science from the African University of Science and Technology (AUST), Nigeria, 2016, with distinction grade. His research is focused on real-time networks and data management.

Patrick Meumeu Yomsi, is an expert in the field of real-time scheduling theory and applications. His research blends software engineering principles with a detailed domain knowledge to innovate new models and accompanying analyses which provide developers with early evaluations of safety-critical systems. Patrick received his Ph.D. in 2009 from Paris-Sud University in Orsay, France. He joined the CISTER Research Centre in 2012 and is currently a Senior Researcher. His research interests include real-time scheduling theory, real-time communication and real-time operating systems. He authored or co-authored 70+ scientific papers and his work was awarded on several occasions.

Borislav Nikolic (born 1982 in Leskovac, Serbia) received the Degree in electrical engineering from the University of Belgrade, Serbia, in 2007, and the Ph.D. Degree in computer engineering from the University of Porto, Portugal, in 2015. After that, he worked as a Postdoctoral researcher at the Institute of Computer and Network Engineering at TU Braunschweig, Germany. Currently, he is a lecturer at the Faculty of Information Technology at Metropolitan University, Belgrade, Serbia. His main research interest is in the formal analysis of on-chip and off-chip network architectures, with an emphasis on their application in the automotive domain.