Mixed-Criticality on the AFDX Network: Challenges and Potential Solutions

A. FINZI, A. MIFDAOUI, F. FRANCES, E. LOCHIN
University of Toulouse-ISAE, France

***Abstract*—In this paper, we first assess the most relevant existing solutions enabling mixed-criticality on the AFDX and select the most adequate one. Afterwards, the specification of an extended AFDX, based on the Burst-Limiting Shaper (BLS), is detailed to fulfill the main avionics requirements and challenges. Finally, the preliminary evaluation of such a proposal is conducted through simulations. Results show its ability to guarantee the highest criticality traffic constraints, while limiting its impact on the current AFDX traffic.**

***Keywords*-TSN, BLS, AFDX, mixed criticality, avionics**

I. Context and Motivations

The growing number of interconnected end-systems and the expansion of exchanged data in avionics have led to an increase in complexity of the communication architecture. To cope with this trend, a first communication solution based on a high rate backbone network, i.e., the AFDX (Avionics Full Duplex Switched Ethernet) [2], has been implemented by Airbus in the A380, to interconnect critical subsystems. Moreover, some low rate data buses, e.g., CAN [15] or ARINC 429, are still used to handle some specific avionics domains, such as the I/O process and the Flight Control Management. Although this architecture reduces the time to market, it conjointly leads to inherent heterogeneity and new challenges to guarantee the real-time requirements.

To cope with these emerging issues, with the maturity and reliability progress of the AFDX after a decade of successful use, a homogeneous avionic communication architecture based on such a technology to interconnect different avionics domains may bring significant advantages, such as easier installation and maintenance and reduced weight and costs. This homogeneous communication architecture, based on the AFDX technology, needs to support mixed-criticality applications, where safety-critical and best effort traffic co-exist. Hence, in addition to the current AFDX traffic profile, called Rate Constrained (RC) traffic, at least two extra profiles have to be handled. The first, denoted by Safety-Critical Traffic (SCT), is specified to support flows with hard real-time constraints and the highest criticality, e.g., flight control data; whereas the second is for Best-Effort (BE) flows with no delivery constraint and the lowest criticality, e.g., In-Flight Entertainment traffic.

Various solutions have been proposed in the literature to support mixed-criticality applications in embedded systems and particularly in avionics and automotive [18][26][9][8]. These solutions can be categorized according to the implemented communication paradigm, i.e., mainly event-triggered or time-triggered. This parameter is of utmost importance to quantify the reconfiguration effort needed by the alternative avionics communication architecture, in comparison to the current AFDX standard. Furthermore, it conditions the modularity level of the selected solution. The event-triggered paradigm is known as highly flexible and facilitates the system reconfiguration, but it infers at the same time an indeterminism level and needs further proofs to verify the predictability requirement. On the other hand, the time-triggered paradigm is highly predictable, but presents some limitations in terms of system reconfigurability.

Hence, our main contributions in this paper are threefold: (i) **first**, an assessment of the most relevant existing solutions enabling mixed-criticality on the AFDX to select the most relevant one; (ii) **second**, the specification of an extended AFDX, based on the Burst-Limiting Shaper (BLS) [8] defined in the Time Sensitive Networking (TSN) task group [28], favoring the main avionics requirements and challenges; (iii) **third**, the simulation of such a proposal to better understand the trends of the BLS behavior and get a first idea of its ability of guaranteeing avionics requirements, i.e., a preliminary proof of concept of such a proposal. It is worth noting that it is difficult to draw firm conclusions concerning the solution performance based on simulation, since it does not cover the worst-case behavior, a key point to prove certification requirements. Therefore, the formal analysis of our proposal will be handled as a next step to accomplish the proof of concept and it is out of scope for this paper.

In the next section, we present the common solutions for mixed-criticality on the AFDX, and their pros and cons versus avionics requirements are discussed. Afterwards, we detail in Section III the specification of our proposed solution, including software and hardware features. Finally, in Section IV we detail the preliminary performance analysis based on simulation of such a proposal and draw our first conclusions on its potential promises.

II. Mixed-Criticality Solutions vs Avionics Requirements

In this section, we first present the main avionics requirements and challenges to cope with mixed-criticality applications needs on the AFDX. Then, we discuss the existing mixed-criticality solutions pros and cons vs such requirements.

A. Avionics Requirements

The two main avionics requirements, which have been considered to select the best solution in this context, are as follows:

- **Predictability**: the impact of a system on an other is known and bounded. The communication architecture must behave in a predictable way, where the extended AFDX has to guarantee bounded latencies respecting the temporal constraints of the mixed-criticality traffic.
- **Modularity**: this requirement is related to the flexibility and exchangeability of software and hardware components. An important step towards enhancing the avionics system modularity has been fulfilled with the adoption of the IMA approach [29], i.e., common elementary components can be configured to fit different avionic applications. This feature aims to minimise the (re) configuration and readjustment effort to facilitate system maintenance and its progress over the years. For instance, the event-triggered paradigm of the AFDX is favoring such a requirement.

Moreover, we need to deal with the main challenge of enforcing the Quality of Service (QoS) features, while limiting the impact of the highest priority traffic on the current AFDX traffic and the implementation complexity. These challenges will be denoted by **Fairness**, and **Complexity** along this paper.

B. Potential Solutions for Mixed-Criticality on the AFDX

In this section, we will detail the different Ethernet-compliant real-time solutions and assess their potential ability vs the avionics requirements.The different solutions can be categorized according to the implemented communication paradigm, i.e., mainly time-triggered or event-triggered.

1) Time-triggered solutions: The main relevant solutions implementing the time-triggered paradigm on top of Switched Ethernet are Time Triggered Ethernet (TTE) [18], and two other solutions proposed by the TSN task force: the Time Aware Shaper (TAS)[26] and the Peristaltic Shaper (PS)[22].

Time Triggered Ethernet

TTE[18] is an industrial protocol developed by TTTech Computertechnik AG and is fully compliant with the Ethernet Standard. The access to the medium is done through coordinated Time Division Multiple Access (TDMA). The main features of TTE are its system-wide global time, and its fault tolerance with fault isolation and diagnosis mechanisms.

There are 3 message types: (i) the first one is Time Triggered (TT) defined by it period, offset and length, and it is configured off-line with dedicated transmission slots; (ii) the second type of traffic is Rate Constrained (RC) with specified rate and length, and is not sent at fixed points in time; (iii) the last class is Best Effort (BE) which has the lowest priority, uses the left bandwidth and has no guarantee on transmission nor reception.

TTE has a high predictability due to the implemented TDMA mechanism, which prevents over-talkative nodes from impacting the others. However, TTE relies on a complex time table to manage the traffic transmission, which reduces its modularity and increases inherently its implementation complexity. Moreover, the aim of TTE is to guarantee the best service to the TT class, even if it deteriorates the service offered to lower priorities (RC and BE). This fact limits the guaranteed fairness of TTE.

Time Aware Shaper

TAS[26] uses time-driven scheduling to manage link access between traffic classes, which makes it a good candidate for mixed-criticality traffic. For each traffic class, the frames are transmitted according to a gate schedule at each output port: it allows frames to pass when opened, and it blocks frames when closed. The different gate schedules are programmed offline, and multiple gates can be opened at the same time. Then, the selected frames are arbitrated according to their priority levels. To prevent frames transmission when the gate is closed, TAS defines guard bands. From the start of a guard band until the gate is opened, no new frames of the corresponding class are allowed to start transmission.

Unlike TTE, TAS is still under specification by the TSN Task Goup[28] but it is very close to TTE in terms of objectives and how to achieve them. Due to the gate schedule, TAS guarantees a high predictability level, but the modifications are propagated to all flows. This fact limits the TAS modularity, while inferring high implementation complexity. Additionally, when lower classes gates are opened, they are scheduled using a Static Priority, which implies a low fairness.

Peristaltic Shaper

The Peristaltic Shaper (PS) [22] uses a global time divided in odd and even phases to manage different traffic classes. If a shaped frame arrives in an odd (resp. even) phase, it can not be sent before the start of the next even (resp. odd) phase. The idle time can be used by other priorities. The Peristaltic Shaper has been proposed by the same task group as TAS. Hence, they have often been studied together and similar work has been done.

Similarly to TTE and TAS, the use of a global time in PS implies a high predictability level but a negative impact on its modularity and implementation complexity: a flow modification can impact the calculation of odd and even phases not only along its path, but also on other flows paths. However, due to the initial waiting time caused by the odd and even phases, lower priority flows may be sent more quickly than under Static Priority schedule, which makes Peristaltic Shaper an interesting solution in terms of fairness.

2) Event-triggered solutions: Among the most interesting solutions based on event-triggered paradigm, we distinguish two classes of solutions. The first class is extending the AFDX standard with well-known scheduling schemes, such as the NP-SP [6] and the Weighted-Round-Robin (WRR) [31]. The second class in this category is integrating credit-based

shapers to control generally the highest priority level, in order to limit its impact on lower priority ones and to guarantee real-time communication. This idea has been integrated in Ethernet AVB [16], and more recently in TSN with the Burst Limiting Shaper (BLS)[8].

Non-Preemptive Static Priority Scheduler

The Non-Preemptive Static Priority (NP-SP) scheduler is the simplest QoS implementation with very limited complexity. Each queue has a defined priority and the scheduler dequeues the first frame of the eligible queue (a queue with enqueued traffic) with the highest priority. This scheduler is defined in the AFDX standard [2], and due to the leaky bucket shapers in the end-systems and policers in the switches, NP-SP guarantees the predictability requirement. Like all event-triggered solutions, NP-SP allows a high modularity level, but it is a well-known as an unfair scheduler[30].

GPS-like Schedulers

The Generalized Processor Sharing (GPS) is an idealized scheduling algorithm that achieves perfect fairness: the bandwidth is shared depending on fixed weights. Many algorithms have been developed to come as close as possible to the GPS, such as the Weighted Fair Queuing (WFQ) [7] or Weighted Round Robin (WRR) [31] and Deficit Round Robin (DRR) [9]. Ordinary round-robin servicing of queues can be done in constant time. With WRR, the usual implementation consists in setting a number of frames that can be consecutively sent for each queue. The major problem, however, is the unfairness caused by possibly different packet sizes used by different flows. This flaw can be removed by using a counter to keep track of traffic transmitted as with the Deficit Round Robin (DRR). Nonetheless, these schedulers necessitate a virtual clock, which increases their implementation complexity.

In [9], an AFDX network implementing the DRR has been specified and studied. Results have shown the good performances of the proposal in terms of predictability and fairness, while increasing the implementation complexity. Moreover, like NP-SP, DRR offers a high modularity level.

Credit Based Shaper

In recent years, there has been a strong interest in the IEEE 802.1 Audio/Video Bridging (AVB) protocol, which provides end-to-end delay guarantees in Ethernet networks. AVB specifies a credit-based shaping (CBS) algorithm for real-time (RT) traffic classes A and B. Each shaped class has a credit-counter, which is replenished at a constant rate (the so-called idle slope) and consumed at the rate allowed by the port (the send slope) when data on the specific class is transferred. When the queue is empty, the credit immediately returns to 0. The different classes are scheduled using a static priority scheduler, with the CBS preventing the starvation of lower priorities and giving bandwidth guarantees, which are good properties for mixed-criticality applications.

Concerning the predictability of CBS, the different classes are isolated from each other thanks to the counter and their associated blocking effect. However, it has been shown in [4] that the impact of the blocking effect of the AVB on the latency is high, which induces a medium predictability level for this shaper. However, the worst-case latency of unshaped lower priorities is improved due to the shaping of classes A and B, which fulfills the fairness challenge. The main drawback of the CBS is that frames cannot be transmitted if the credit is below 0, no matter the state of the other queues. This fact can cause unnecessary delays if other queues are empty. This issue has been fixed by the TSN [28] task group in the Burst Limiting Shaper.

Burst Limiting Shaper

Presented in [8], the BLS is always used with a static priority scheduler, where BLS modifies the priority seen by the SP depending on a credit counter. Hence, depending on the priority value, the shaped frames can be blocked or not by other classes. However, no matter the state of the credit, if a frame is the first of the queue with the highest priority among the eligible queues, then it will be transmitted. Thus, contrary to CBS, the BLS is a non-blocking shaper, which is a large improvement of the predictability guarantees.

The priority change feature enables the BLS to reserve bandwidth for the shaped queue. This fact induces a low implementation complexity; and also improves fairness in comparison to SP, since it limits the bandwidth available to the shaped queue.

C. Discussion: selecting the most promising solution

The conclusions on the considered solutions vs the main avionics requirements and challenges are illustrated in Table I. As we can notice, there are three solutions fulfilling all the requirements: CBS, DRR and BLS.

The AVB/CBS sometimes blocks frames when the transmission link is free, which causes unnecessary delays and limits its predictability level. Hence, we discard this solution in the avionics context.

On the other hand, the DRR is a well-known scheduler, which has been extensively used and analyzed in many domains, such as in avionics [9]. However, DRR increases the implementation complexity due to its parameters, i.e., weights, tuning process. Finally, the BLS, which is the new shaper mainly studied by the automotive community [24] and also started gaining attention from the avionics community [19], guarantees the main requirements and challenges, while limiting the implementation complexity. One of the interesting feature of the BLS is actually its ability to shape one queue and leave the others to SP, unlike the DRR that shapes all the queues. Moreover, DRR reserves bandwidth for lower priority traffic; whereas the BLS lets the non-real time traffic use the remaining bandwidth left by real-time traffic.

Therefore, the BLS is considered herein as the most interesting solution to be incorporated within the AFDX standard, to enable an homogeneous avionics communication architecture for mixed-criticality applications.

Solutions	TTE	TAS	PS	BLS	AVB	NP-SP	DRR
refs.	[18]	[26]	[22]	[8]	[16]	[6]	[9]
Modularity.	L	L	L	H	H	H	H
Predictability	H	H	H	H	M	H	H
Fairness	L	L	M	H	H	L	H
Complexity	H	H	H	L	L	L	M

TABLE I
EXISTING SOLUTIONS VS AVIONICS REQUIREMENTS AND CHALLENGES

III. SPECIFICATION OF THE EXTENDED AFDX

In this part, we detail the specification of our proposal extending the AFDX standard to cope with mixed-criticality applications. First, we discuss the different options for the QoS identification on the AFDX network and we identify the best one in terms of scalability, complexity and performance. Afterwards, we describe the switch architecture including the output port scheduler and the BLS shaper. Finally, we present the most important implementation features of the proposed solution on the software and hardware levels.

A. QoS identification

In order to implement the Quality of Service (QoS), the first problem to handle is the identification of the traffic class. The AFDX already uses a system to differentiate two classes of service. Currently, there are two supported priorities, low and high, with two distinct queues in each output port, but there is only one queue used for the current AFDX traffic profile. Consequently, there is a possibility to add only one extra traffic class. In this section, we explore the different possibilities to enable the QoS identification of many priority levels.

Configuration Files

The characteristics of a Virtual Link in the AFDX network are defined in a configuration file shared by every switch in the network, called Filtering _Policing _and _Forwarding_ Configuration_Table. Its last column, denoted prioritization, defines the priority (high or low) of a VL. We propose to modify this field to add other possibilities, for example by adding new priority qualifiers, or using numbers, to define the priority. Since It is the last column, then increasing the length of the prioritization field will only change the line size but not displace other fields in the line. The drawback of this is the possible change of type of the prioritization information, and possibly a slight growth of the configuration table file. The advantage is that no modification is necessary to the current AFDX frames. However, some modifications may be required within the switch to interpret differently the configuration file.

MAC Address

A second solution consists in using a part of the constant field of the MAC address to encode the priority. This fact would slightly decrease the size of the configuration table, since the prioritization field could then be deleted. However, it requires changing the End-Systems to build the MAC address field, and the switches to guarantee the correct interpretation of the MAC address in the configuration table.

802.1Q

Another solution consists in using the 802.1Q header. In the Tag Control Information (TCI) field, the Priority Code Point (PCP) is a 3 bits field used to define the priority of a frame, which offers 8 possibilities. Unfortunately, while this solution is appealing due to the use of a well known and globally used standard, it has the same drawbacks as the MAC address solution: required changes within End-systems and switches and no real advantage compared to the current implementation.

IP Header

A fourth solution is using the Differentiated Services Code Point (DSCP), a field used in the IP header to differentiate the different classes. This solution is based on layer 3 of the OSI model, while the current switches only use layer 2 fields. It would mean accessing a higher and more complicated OSI layer. Similarly to both previous solutions, the current AFDX frames would have to be modified in order to be assigned a priority. Moreover, since the third layer is more complex, it might also be more difficult and more expensive to obtain the certification of the switches.

Discussion of potential solutions

	Config. file	MAC address	802.1Q	IP address
Complexity	++	+	+	–
Scalability	+++	+++	+	++
Performance	++	+++	+	+++

TABLE II
CLASSIFICATION SOLUTION COMPARISON

The various alternatives are compared in Table II according to three criteria. The first one is the complexity of the solution, it takes into account the modifications needed for the switch, the End-Systems, the frame structure and the frame layer accessed by the switch. The second one is the scalability of each solution, in this case the number of available classes. The last one is the performance of the solution and depends on its induced overhead.

The solution using the current configuration file is the one that does not require the modification of the switch, the End-Systems, or the frame; thus it has the lowest complexity. Moreover, it has a good scalability in terms of number of classes since any number could be added to the file. Finally, it has good performances in terms of overhead, with only one column needed in the configuration table. With the other solutions, the way a switch identifies the frame class is very different. They do not use the configuration file at all and store the class inside the frame. This means these solutions are more complex because they require more modifications than the previous one. Both the MAC and IP address use already existing field, unlike the 802.1Q which needs more

modifications of the AFDX frame and consequently more overhead. However, the MAC address and the 802.1Q are less complex than the IP Address since they are layer-2 fields. In addition, the 802.1Q is the less scalable because the number of classes is limited to 8, whereas the other can have several thousands of classes due to their field lengths.

Hence, extending the current way priorities are set in the AFDX network seems the simplest solution, since it necessitates only few modifications and does not need access to a higher OSI layer. Moreover, with this configuration file, new and old AFDX switches could be used in the same network, while having different Filtering_Policing _and _Forwarding _Configuration _Tables to handle the specific number of traffic classes.

B. Switch architecture

The AFDX standard manages the exchanged data through the Virtual Link (VL) concept. This concept provides a way to reserve a guaranteed bandwidth for each traffic flow. The VL represents a multicast communication, which originates from a single End-System and delivers packets to a fixed set of End Systems. Each VL is characterized by: (i) BAG (Bandwidth Allocation Gap), ranging in powers of 2 from 1 to 128 milliseconds, which represents the minimal inter-arrival time between two consecutive frames; (ii) MFS (Maximal Frame Size), ranging from 64 to 1518 bytes, which represents the size of the largest frame sent during each BAG. Furthermore, the AFDX supports a NP-SP scheduler based on two priority levels within switches to enable the QoS features.

In Fig.1, we illustrate the architecture of our extended AFDX switch. It consists of: (i) store and forward input ports to verify each frame correctness before sending it to the corresponding output port; (ii) a static configuration table to forward the received frames to the correct output port(s) based on their VL identifier. Hence, to manage both extra AFDX profiles, i.e., SCT and BE, within our extended AFDX switch, we need to update the configuration table to add the corresponding VL identifiers and their associated priority levels, and we need to update the QoS identification to implement at least 3 priorities; (iii) the output ports with three priority queues, multiplexed with a NP-SP scheduler, and the highest one is shaped with the BLS.

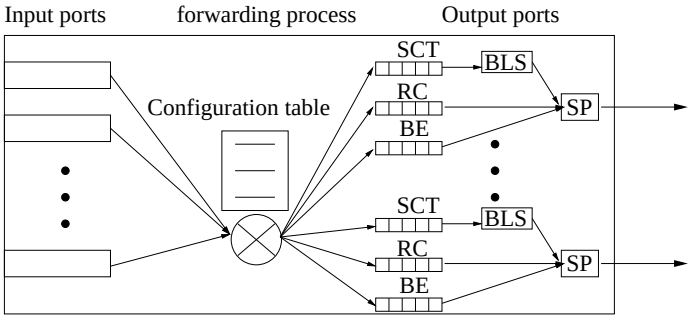

Fig. 1. An Extended AFDX switch architecture

The BLS has been characterized in [8] by an upper threshold, L_M, a lower threshold L_R, such as $0 \leqslant L_R < L_M$, and a reserved bandwidth, BW. Additionally, the priority of a queue q shaped by BLS, denoted $p[q]$, can vary between a high and a low value, denoted p_H and p_L. The low value is usually below the lowest priority of unshaped traffic. In the avionic context, to guarantee the safety isolation level between the different traffic profiles, the low value associated to the SCT is set to be lower than the RC priority level, but higher than the BE priority. Therefore, as illustrated in Fig.2, when considering one class for each traffic type, SCT queue priority oscillates between 0 (the highest) and 2, RC priority is 1 and BE has the priority 3 (the lowest). Thus, when SCT traffic is enqueued, BE traffic can never be sent no matter the state of BLS. In this case, RC is the only traffic that can be sent and this only happens when the SCT priority is 2. As a consequence, BE traffic is isolated from SCT and RC traffics.

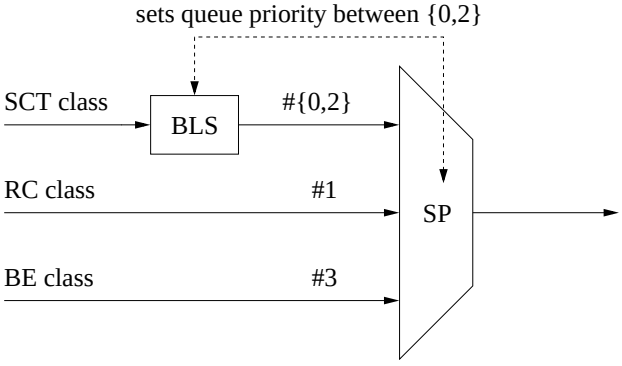

Fig. 2. Burst Limiting Shaper on top of NP-SP at the output port

The credit counter is a measure of the credit consumption, i.e. it increases when shaped traffic is sent, else it decreases, as follows:

- initially, the credit counter starts at 0 and the priority of the burst-limited flows is high (#0);
- the main mechanism of the BLS is the change of priority $p[q]$ of the shaped queue, which occurs in two contexts: 1) if $p[q]$ is high and credit reaches L_M; 2) if $p[q]$ is low and credit reaches L_R;
- when a frame is transmitted, the credit increases (is consumed) with a rate of I_{send}, else the credit decreases (is gained) with a rate of I_{idle};
- when the credit reaches L_M, it stays at this level until the end of the transmission of the current frame;
- when the credit reaches 0, it stays at this level until the end of the transmission of the current frame (if any). The credit remains at 0 until a new BLS frame is transmitted.

The different parameters of the BLS shaper are defined as follows:

- the decreasing rate is:

$$I_{idle} = BW \cdot C,$$

where C is the link speed and BW is the percentage of bandwidth reserved for BLS frames.

- the increasing rate is:

$$I_{send} = C - I_{idle}.$$

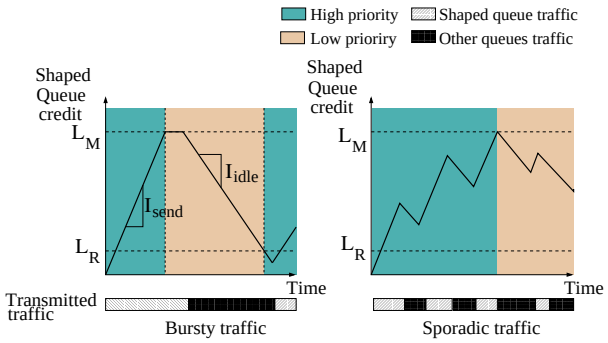

Fig. 3. BLS credit evolution

The behavior of the BLS is illustrated in Fig.3. As shown, the credit is always between 0 and L_M. It is worth noting that with the BLS, both the priority of the shaped queue and the state of all the queues, i.e., empty or not, define whether the credit is gained or lost. This aspect is depicted in Fig.3 for two arrival scenarios. The first one (left figure) shows the case of a bursty traffic, where the maximum of traffic shaped by the BLS is sent when its priority is the highest. Consequently, the other priorities send as much traffic as possible when the BLS queue priority has the low value. The second one (right figure) is for sporadic traffic, where we can see that when the shaped queue priority is highest but no frame is available, then credit is regained (the credit value decreases). Conversely, when the priority is at the low value and the other queues are empty, then the shaped frames can be transmitted and credit is consumed (the credit value increases).

C. Implementation

We describe in this part the software and hardware implementation features of the extended AFDX switch. The former is mainly related to the BLS algorithm; whereas the latter is related to the hardware overhead, in comparison to the current architecture.

From the description of the BLS, we see that the implementation at the hardware level requires a counter to track the credit and a timer to handle credit updates. These parameters, i.e., a counter and a timer, induce low extra complexity to implement a BLS on top of a NP-SP scheduler, in comparison to a regular NP-SP scheduler.

Hence, the algorithm allowing to implement the BLS corresponds to a modification of the priority scheduler, and it is presented in Algorithm 1. This algorithm is executed in two situations: 1) if a frame arrives when all queues are empty; 2) at the end of the current frame transmission, a new frame has to be elected for dequeuing.

The credits of each queue q is stored in $credits[q]$. Each shaped queue q has a dequeuing timer. Likewise, for each queue L_M, L_R, BW, p_L and p_H are stored in LMs, LRs, BWs, pLs and pHs. A queue q shaped by BLS is

Algorithm 1 BLS algorithm: dequeuing process

Require: $credits$; $timerDQs$;C LMs;LRs;BWs;pLs; pHs;

1: **for** each queue q with $pLs[q] < pHs[q]$ **do**
2: $time = getcurrentTime()$
3: $\delta_{time} = time - timerDQs[q]$
4: **if** $\delta_{time} > 0$ **then**
5: $credits[q] = \max(credits[q] - \delta_{time} \cdot BWs[q] \cdot C, 0)$
6: $timerDQs[q] = time$
7: **if** $credits[q] \leqslant LRs[q]$ and $p[q] = pLs[q]$ **then**
8: $p[q] = pHs[q]$
9: **end if**
10: **end if**
11: **end for**
12: **for** each priority level pl, highest first **do**
13: **if** length(queue(pl))>0 **then**
14: q=queue(pl)
15: **if** $pLs[q] < pHs[q]$ **then**
16: $credits[q]$=min($LMs[q]$,$credits[q]$+size(head(q)) $\cdot(1 - BWs[q]))$)
17: $timerDQs[q]$=$time$+size(head(q))/C
18: **if** $credits[q] \geqslant LMs[q]$ and $p[q]$ = $pHs[q]$ **then**
19: $p[q] = pLs$
20: **end if**
21: **end if**
22: dequeue(head(q))
23: break
24: **end if**
25: **end for**

characterized by the fact that $pLs[q] < pHs[q]$, otherwise $pLs[q] = pHs[q]$. The current priority of a queue is store in p. We suppose that several queues can be shaped and no two queues can have the same priority. All the timestamps used in the algorithm are set to the time value at the start of execution. Also, $timerDQs[q]$ represents the estimated end of the shaped frame transmission.

The credits $credit$ and the dequeuing timers $timerDQs[q]$ are initialized to zero. The initial priority is set to the high value. First, we store the current time in $time$ in line 2. Then, for each BLS queue q (line 1), in line 3, we compute δ_{time}, the difference between the current time and the time stored in $timerDQs[q]$. The duration δ_{time} represents the time elapsed since the last credit update, during which no shaped packet was sent, we call this the idle time. Then, if $\delta_{time} > 0$, then the credit is updated by removing the credit gained during the idle time that just occurred (lines 4 and 5). Next, $timerDQs[q]$ is set to the current time to keep track of the time the credit is last updated (line 6). If the credit reaches L_R, the priority changes to its high value (lines 7 and 8). Then, with the updated priorities, the priority scheduler performs as usual: each queue is checked for dequeuing (lines 12 and 13). When a BLS queue is selected, the credit expected to be consumed is added to the $credit$ variable (line 16). The time taken for the packet to be dequeued is added to the variable $timerDQs[q]$

(lines 16 and 17) so the transmission time of the packet will not be taken into account in the idle time δ_{time} (line 3). If the credit reaches L_M, the priority changes to its low value (lines 18 and 19). Finally, the packet is dequeued (line 22), and the loop is exited in line 23.

Algorithm 1 also implements the following functions:

- $getcurrentTime()$ uses a timer to return the current time;
- $queue(pl)$ returns the queue associated to the priority pl;
- $head(q)$ returns the first packet in the queue q;
- $size(f)$ returns the size of the packet f;
- $dequeue(f)$ activates the dequeuing event of packet f.

The complexity of this algorithm is the same as a priority scheduler and is $O(1)$ (the number of queues is constant).

In comparison to the current AFDX switch architecture, the main modifications required for the proposed extended AFDX switch consists in: (i) at the software level, updating the static configuration table to manage three priority levels instead of two (note that the update overhead is very limited); (ii) at the hardware level, adding an extra priority queue at the output port since the current AFDX switch already supports two priorities; and implementing the BLS for the SCT queue on top of the NP-SP scheduler, as illustrated in Fig.1.

From the global avionics communication architecture point of view, our extended AFDX necessitates the update of the End-Systems at the application layer to enable a consistent mapping between VL identifiers and the appropriate priority level. Moreover, the implementation and certification of this extended AFDX may imply extra costs, in comparison with the current one. However, this fact is counterbalanced by the major pros of such an homogeneous architecture, in terms of enhancing performance and reducing cables and weight.

IV. Evaluation

In this section, we present first the case study, and then we discuss the results of the preliminary evaluation.

A. Case study

We consider a Gigabit switch described in Fig.2, with the input traffic described in Table III. The switch is connected to 4 end-systems (sources) for each type of input traffic, and one end-system for the traffic destination (sink). The number of flows of a class k enqueued in the output port, denoted n_k^{in}, determines the load of the output port. We denote UR_k the utilisation rate of class $k \in \{SCT, RC\}$, which directly depends on n_k^{in}: $UR_k = n_k^{in} \cdot \frac{MFS_k}{BAG_k}$.

For this preliminary analysis, we consider 2 scenarios described in Table IV. The aim of scenario 1 (resp. 2) is to get a first idea of the impact of increasing the SCT (resp. RC) utilisation rate on RC and SCT latencies. In particular, we want to verify the **predictability** requirement, i.e., the deadlines are fulfilled when the load of the network is 100%; in addition to the **fairness** challenge, i.e., the impact of SCT on the RC in terms of latencies is limited. We denote UR_k the utilisation rate of class $k \in \{SCT, RC\}$, which directly depends on n_k^{in}.

Thus, in scenario 1 (resp. 2), we set RC (resp. SCT) input rates at 20%, which means generating 156 (resp. 780 flows). Then, we vary SCT (resp. RC) utilisation rate, denoted UR_{SCT} (resp. UR_{RC}) from 0 to over 70%. Moreover, L_R is set to its minimum value, L_M is set to absorb a burst of 80 frames and BW is just below its median (0.5) value. Finally, BE is used to bring the load up to 100% and we do not present its timing results as BE does not have a deadline.

We have simulated in NS2 our extended AFDX switch incorporating the BLS on top of the NP-SP scheduler and have compared it to an AFDX switch implementing a regular NP-SP with three priority levels, denoted here current AFDX. Each simulation has a duration of 5s, which represents up to 3.2 millions SCT and RC simulated frames. The results of scenarios 1 and 2 are presented in Figures 4 and 5, respectively.

Priority	Traffic type	MFS (Bytes)	BAG (ms)	deadline (ms)	jitter (ms)
0/2	SCT	64	2	2	0
1	RC	320	2	2	0
3	BE	1024	8	none	0.5

TABLE III
Avionics flow Characteristics

Scenarios	Scenario 1	Scenario 2
$(UR_{SCT}; UR_{RC})(\%)$	$([0.1..78]; 20)$	$(20; [0.5..72])$
$(n^{in}_{SCT}; n^{in}_{RC})$	$([1:160:3044]; 156)$	$(780; [1:40:564])$
$(BW; L_M; L_R)$	$(0.46; 22, 118; 0)$	$(0.46; 22, 118; 0)$

TABLE IV
Considered Test Scenarios 1 and 2

B. Numerical results

The impact of varying SCT rate

This fact is assessed through scenario 1 and the results are presented in Fig.4. We can see that the SCT latency is increased by the BLS (see Fig.4(a)), comparatively to the regular NP-SP scheduler. In fact, after an initial sharp increase, the increase of the SCT latency has the same increase rate with our Extended AFDX proposition and with current AFDX. This is due to the BLS parameters chosen: our Extended AFDX is made of two parts, a BLS and a SP, and depending on the BLS parameters and the traffic flows, one is predominant on the other.

This is confirmed by the RC latency (see Fig.4(b)): below 16%, the current and Extended AFDX curves are overlapping: the SP part is predominant. After 16% they separate, showing that BLS has now a stronger impact. While the latency with current AFDX soars, it remains constant with our our Extended AFDX. This shows the good isolation provided to RC by the BLS. In fact, while the BLS increases the SCT latency by 0.7ms, it lowers the RC latency by 4ms. As a result, the RC latency is much reduced with our Extended AFDX, while the SCT latency is only slightly increased. It is

also worth noting that with current AFDX the RC deadline is reached at 54%, while it is never reached with our Extended AFDX. Thus, the maximum utilisation rate of RC traffic is improved by 48%, from 54% to 80%, under the extended AFDX.

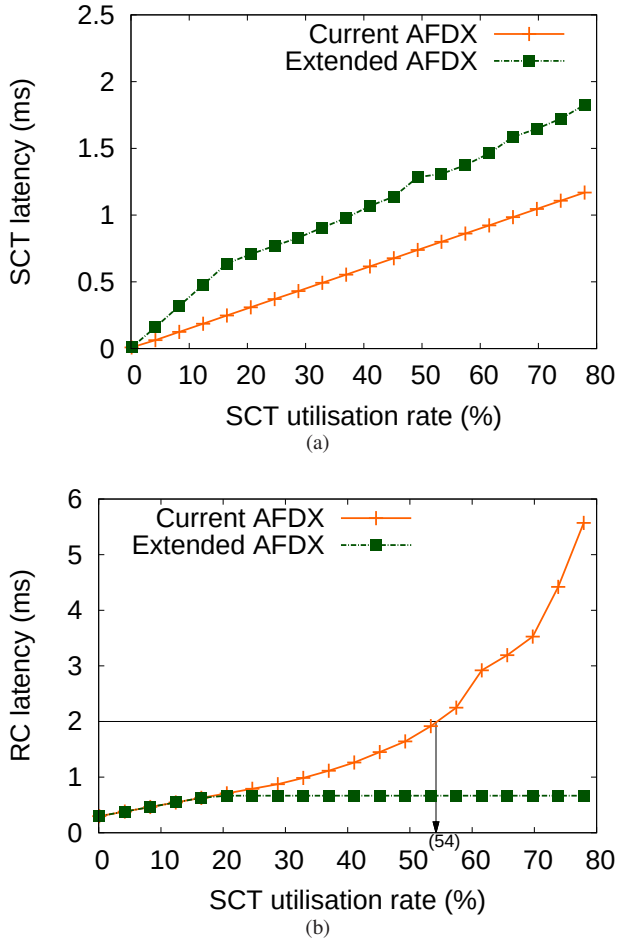

Fig. 4. Scenario 1: impact of SCT max. utilisation rate on: (a) SCT latency; (b) RC latency

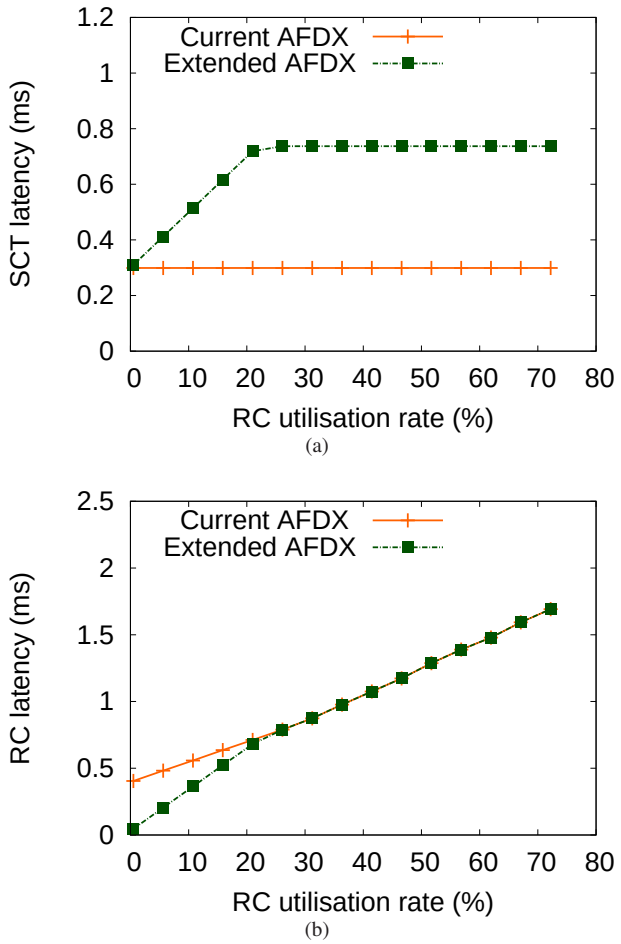

Fig. 5. Scenario 2: impact of RC max. utilisation rate on: (a) SCT latency; (b) RC latency

The impact of varying RC

This fact is assessed through scenario 2 and the results are presented in Fig.5. As we can see, the SCT latency is increased by the BLS (see Fig.5(a)), while the RC latency is either improved or identical, in comparison to the current AFDX. However, the SCT latency remains well below its deadline under the extended AFDX. Additionally, we can see that with the chosen BLS parameters, the BLS has a strong impact for low values of RC rate: in Fig.5(b)), there is a gap between the RC latency with current or Extended AFDX. This gap decreases as RC utilisation rate increases. This is due to the fact that as the RC rate increases, the impact of the BLS on RC traffic decreases until it becomes negligible and only SP rules the RC latency behavior. This shows that the RC latency can be improved by the BLS, even when the BLS parameters are not appropriately set. At the current utilisation rate of the AFDX (30% on the 100Mbps AFDX network, so 3% on a Gigabit AFDX) the gain in terms of latency with our Extended AFDX compared to the current AFDX for RC traffic is around 40%, and it is still over 17% for an utilisation rate RC at 15% of the capacity.

These results show the ability of our extended AFDX switch proposition to favor the predictability of the mixed-criticality traffic, which is one of the key avionics requirements. Moreover, our Extended AFDX switch offers good fairness property since it enables a noticeable RC latencies decrease while guaranteeing the SCT deadline.

V. Conclusions

In this paper, we have assessed the most relevant existing solutions vs the main avionics requirements, to support mixed criticality on the AFDX network. Afterwards, we have specified our extended AFDX, incorporating the BLS shaper on top of NP-SP, which we have considered as the most promising solution. Finally, we have conducted simulations to evaluate the ability of our proposal to guarantee the predictability requirement, while favoring the fairness property. Results show the noticeable enhancement of the latencies of the current

AFDX traffic (RC) in presence of the highest priority one (SCT) under our extended AFDX, with reference to the current AFDX.

As a next step, we will conduct formal analyses to compute the worst-case latencies and prove the predictability of such a promising solution to fulfill the certification needs.

REFERENCES

[1] Mohammed Abuteir and Roman Obermaisser. Simulation environment for time-triggered ethernet. In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 642–648. IEEE, 2013.

[2] Airlines Electronic Engineering Committee. Aircraft Data Network Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network, ARINC Specification 664. Aeronautical Radio, 2002.

[3] Giuliana Alderisi, Alfio Caltabiano, Giancarlo Vasta, Giancarlo Iannizzotto, Till Steinbach, and Lucia Lo Bello. Simulative assessments of ieee 802.1 ethernet avb and time-triggered ethernet for advanced driver assistance systems and in-car infotainment. In *Vehicular Networking Conference (VNC), 2012 IEEE*, pages 187–194. IEEE, 2012.

[4] Unmesh D Bordoloi, Amir Aminifar, Petru Eles, and Zebo Peng. Schedulability analysis of ethernet avb switches. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pages 1–10. IEEE, 2014.

[5] Marc Boyer, Giovanni Stea, and William Mangoua Sofack. Deficit Round Robin with network calculus. In *Performance Evaluation Methodologies and Tools (VALUETOOLS)*, 2012.

[6] Rodrigo Coelho, Gerhard Fohler, and J-L. Scharbarg. Worst-case backlog for AFDX network with n-priorities. In *RTN*, 2014.

[7] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM Computer Communication Review*, volume 19, pages 1–12. ACM, 1989.

[8] Franz-Josef Gotz. Traffic Shaper for Control Data Traffic (CDT). *IEEE 802 AVB Meeting*.

[9] Yu Hua and Xue Liu. Scheduling design and analysis for end-to-end heterogeneous flows in an avionics network. In *INFOCOM, 2011 Proceedings IEEE*, pages 2417–2425. IEEE, 2011.

[10] D. Thiele J.Diemer and R. Ernst. Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching. In *SIES*, 2012.

[11] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The time-triggered ethernet (tte) design. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 22–33. IEEE, 2005.

[12] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. *IEEE/ACM Transactions on Networking (TON)*, 2004.

[13] Hyung-Taek Lim, Daniel Herrscher, Martin Johannes Waltl, and Firas Chaari. Performance analysis of the ieee 802.1 ethernet audio/video bridging standard. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 27–36. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[14] Zhao Luxi, Pop Paul, Li Qiao, Chen Junyan, and Xiong Huagang. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Systems*, 2017.

[15] R. Bosch GmbH. CAN specification Version 2,0. Technical report, 1991.

[16] Standard. IEEE Std. 802.1Qav, IEEE Standard for Local and metropolitan area networks, Virtual Bridged Local Area Networks, Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams, 2009.

[17] Till Steinbach, Hermand Dieumo Kenfack, Franz Korf, and Thomas C Schmidt. An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 375–382. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

[18] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan. TTEthernet Dataflow Concept. In *Eighth IEEE International Symposium on Network Computing and Applications*, 2009.

[19] W. Steiner, P. Heise, and S. Schneele. Recent ieee 802 developments and their relevance for the avionics industry. In *2014 IEEE/AIAA 33rd DASC*.

[20] Wilfried Steiner. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 375–384. IEEE, 2010.

[21] Domitian TamasSelicean, Paul Pop, and Wilfried Steiner. Timing analysis of rate constrained traffic for the ttethernet communication protocol. In *Real-Time Distributed Computing (ISORC), 2015 IEEE 18th International Symposium on*, pages 119–126. IEEE, 2015.

[22] Michael Johas Teener. Back to the future:using TAS and preemption for deterministic distributed delays. *IEEE 802.1 AVB TG Meeting, San Antonio*.

[23] Sivakumar Thangamuthu, Nicola Concer, Pieter JL Cuijpers, and Johan J Lukkien. Analysis of ethernet-switch traffic shapers for in-vehicle networking applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, pages 55–60. IEEE, 2015.

[24] D. Thiele and R. Ernst. Formal worst-case timing analysis of Ethernet TSN's burst-limiting shaper. In *DATE*, 2016.

[25] Daniel Thiele, Jonas Diemer, Philip Axer, Rolf Ernst, and Jan Seyler. Improved formal worst-case timing analysis of weighted round robin scheduling for ethernet. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[26] Daniel Thiele, Rolf Ernst, and Jonas Diemer. Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers. In *VNC*. IEEE, 2015.

[27] Ken W Tindell, Alan Burns, and Andy J Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, 1994.

[28] TSN Task Group. TSN Specifications. http://www.ieee802.org/1/pages/tsn.html/.

[29] C. Watkins and R. Walter. Transitioning From Federated Avionics Architectures To integrated Modular Avionics. 26th Digital Avionics Systems Conference (DASC), 2008.

[30] Adam Wierman and Mor Harchol-Balter. Classifying scheduling policies with respect to unfairness in an m/gi/1. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 238–249. ACM, 2003.

[31] Tianran ZHOU and Xiong Huagang. Design of energy-efficient hierarchical scheduling for integrated modular avionics systems. *Chinese Journal of Aeronautics*, 2012.