Performance Enhancement of Extended AFDX via Bandwidth Reservation for TSN/BLS Shapers

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

ABSTRACT

To support mixed-criticality applications, the AFDX may integrate multiple traffic classes: Safety-Critical Traffic (SCT) with hard real-time constraints, Rate-Constrained (RC) traffic requiring bounded latencies and Best Effort (BE) traffic with no delivery constraints. These traffic classes are managed based on a Non-Preemptive Strict Priority (NP-SP) Scheduler, where the highest priority traffic (SCT) is shaped with a Burst Limiting Shaper (BLS). The latter has been defined by the Time Sensitive Networking (TSN) task group to limit the impact of high priority flows on lower priority ones. This paper proposes two bandwidth reservation methods for BLS shapers in AFDX networks. The proposed methods are evaluated on a realistic AFDX configuration. Results show their efficiency to noticeably enhance the RC delay bounds and the SCT schedulability, in comparison to an intuitive method.

ACM Reference Format:
A. FINZI, A. MIFDAOUI, F. FRANCES and E. LOCHIN University of Toulouse-ISAE, France . 2018. Performance Enhancement of Extended AFDX via Bandwidth Reservation for TSN/BLS Shapers . In *Proceedings of ACM RTN conference (RTN18).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

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.

To cope with this emerging issue, in [4], we have assessed the most relevant existing solutions enabling mixed-criticality on the AFDX vs avionics requirements. The Burst-Limiting Shaper (BLS) [5] (defined in the Time Sensitive Networking (TSN) task group [6]) on top of Non-Preemptive Strict-Priority (NP-SP) scheduler has been selected as the most promising solution favoring the main avionics requirements. In particular, the fairness of such a solution has been highlighted compared to the current AFDX implementing NP-SP. Preliminary performance evaluation of such a solution, denoted extended AFDX, has been provided based on simulations. The first results were encouraging to pursue this line through providing formal timing analysis to prove certification requirements, a key point in avionics. In particular, simulations showed the ability of the BLS to limit the impact of SCT on RC delay bounds (up to 40% delay bound decrease) and enhance schedulability up to 48%. Afterwards, in [3], we have introduced a Network Calculus-based approach to compute the delay bounds of SCT and RC classes in such an extended AFDX network, taking into account the impact of the TSN/BLS. The performance evaluation of our proposal on a realistic AFDX configuration has highlighted its efficiency, in comparison with the current AFDX (implementing only NP-SP scheduler).

In this paper, our aim is to provide optimized bandwidth reservation methods with different complexity levels for the TSN/BLS shapers, to enhance as much as possible the schedulability and the delay bounds of the different traffic classes. These methods are based on an extension of the timing analysis introduced in [3] to cover multi-hop communication. We perform a set of experiments to assess the efficiency of our introduced methods, with reference to an intuitive bandwidth reservation method. Similar works have been conducted to define bandwidth reservation methods for Credit Based Shaper (CBS) of AVB networks [1, 2]. However, to the best of our knowledge, this issue has not been handled yet in the literature for TSN/BLS shapers in avionics domain.

The rest of the paper is organized as follows. Section II introduces the system model. Section III details two optimized bandwidth reservation methods with different complexity levels for TSN/ BLS shapers. Section IV evaluates the proposed methods on a realistic avionic configuration to assess their efficiency. Finally, Section VI concludes the paper.

2 SYSTEM MODEL AND ASSUMPTIONS

In this section, we first describe the extended AFDX switch model.Then, we detail the BLS behavior and its main parameters. Finally, we present the considered schedulability condition and traffic model. The main notations are presented in Table 1.

2.1 The extended AFDX Switch

The aim of extending the AFDX switch architecture with the TSN/BLS is to handle mixed criticality data, and more specifically three AFDX traffic profiles, as illustrated in Fig.1: (i) the SCT with its priority set by the BLS and the tightest temporal deadline, e.g., Flight-control flows; (ii) the RC with the medium priority and a deadline constraint

RTN18, July 2018, Barcelona, Spain

to guarantee, e.g., current AFDX flows; (iii) the BE with the lowest priority and no time constraint, e.g., In-Flight Entertainment.

C	Link speed
MFS_f	Maximum Frame Size of flow f
BAG_f	Bandwidth Allocation Gap of flow f
J_f, Dl_f	Jitter and deadline of flow f
L_M, L_R,BW	BLS maximum and resume credit levels, BLS reserved bandwidth
I_{idle}, I_{send}	BLS idle and sending slopes
$p(j)$	Priority level of a class j with $p(j) \in \{0, 1, 2, 3\}$
UR_j^{bn}	bottleneck network utilisation rate of a class j
$EDD_{j,f}$	end-to-end delay of flow f of class j
r_f, b_f	rate and burst of flow f
R_j^{mux}	minimum guaranteed rate of class j in mux

Table 1: Notations

In Fig.1, we illustrate the architecture of the 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; (iii) the output ports with three priority queues multiplexed with a NP-SP scheduler, and the highest one is shaped with the BLS.

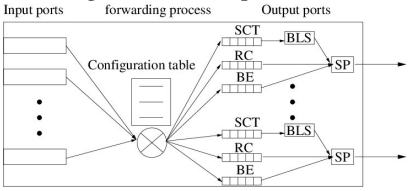

Figure 1: An extended AFDX switch architecture

2.2 BLS Shaper

The BLS belongs to the credit-based shapers class and it is generally used on top of Non-Preemptive Static Priority (NP-SP) scheduler. It has been defined in [5] 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 BLS is characterized by: (i) a decreasing rate $I_{idle} = BW \cdot C$, where C is the link speed and BW is the percentage of bandwidth reserved for BLS frames; (ii) an increasing rate $I_{send} = C - I_{idle}$. Finally, priority of a queue q shaped by BLS, denoted $p(q)$, can vary between a high and a low value (with 0 the highest), denoted p_H and p_L.

The behavior of the BLS is illustrated in Fig. 2 for two arrival scenarios. As shown, the credit is always between 0 and L_M and varies as follows:
(i) initially, the credit counter starts at 0 and the priority of the shaped queue is high;
(ii) the main feature of the BLS is the change of priority $p(k)$ of the shaped queue, which occurs in two contexts: 1) if $p(k)$ is high and credit reaches L_M; 2) if $p(k)$ is low and credit reaches L_R;
(iii) 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};
(iv) when the credit reaches L_M, it stays at this level until the end of the transmission of the current frame (if any);
(v) 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.

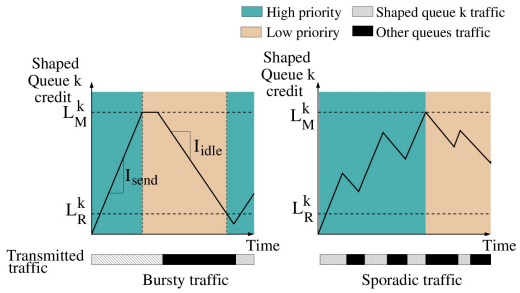

Figure 2: BLS credit evolution

2.3 Schedulability Condition and Traffic Model

First, we define a sufficient schedulability condition for SCT and RC classes, which consists in verifying that the end-to-end delay bound of each traffic flow is lower than its deadline.

The end-to-end delay expression of a flow f in the class $j \in \{SCT, RC\}$, $EED_{j,f}$, along its path $path_f$ is as follows:

$$EED_{j,f} = d_j^{es} + d_{prop} + \sum_{i \in path_f} d_{j,f}^{sw,i} \quad (1)$$

With d_j^{es} the delay within the end-system (es) to transmit the aggregate traffic of class j and d_{prop} the propagation delay along the path, which is generally negligible in an avionics network. The last delay $d_{j,f}^{sw,i}$ represents the delay within the i^{th} switch (sw) along the flow path and it consists of several parts: (i) the store and forward delay at the input port, equal to the transmission time of a maximum-sized frame; (ii) the technological latency due to the switching process, upper-bounded by $1\mu s$; (iii) the delay of the output port multiplexer (mux) due to the BLS (bls) and NP-SP (sp) scheduler. To enable the computation of upper bounds on these delays, the different parts of the network, and more particularly the BLS have been modeled in terms of service curves [3], where β_j^n is the service guaranteed for traffic class j in node $n \in \{es, sw\}$ or a component $n \in \{bls, sp\}$.

Secondly, each traffic flow f of class $j \in \{SCT, RC, BE\}$, generated by an end-system, is characterized by $\left(BAG_f, MFS_f, Dl_f, J_f\right)$ for respectively the minimum inter-arrival time, the maximum frame size integrating the protocol overhead, the deadline if any (generally equal to BAG_f unless explicitly specified and infinite for BE) and the jitter.

The arrival curve of each flow f in class j at the input of the i^{th} node $n \in \{es, sw\}$ or a component $n \in \{bls, sp\}$ along its path is modeled as a leaky-bucket curve with a burst $b_{j,f}^{n,i}$ and a rate r_f:

$$\alpha_{j,f}^{n,i}(t) = b_{j,f}^{n,i} + r_f \cdot t$$

For instance, the input arrival curve of flow f in class j at the end-system is: $\alpha_{j,f}^{es}(t) = MFS_f + \frac{MFS_f}{BAG_f} \cdot (t + J_f)$.

Therefore, the arrival curve of the aggregate traffic in class j at the input of the i^{th} node $n \in \{es, sw\}$ or a component $n \in \{bls, sp\}$

is: $\alpha_j^{n,i}(t) = \sum_{f \in j} \alpha_{j,f}^{n,i}(t)$. For instance,

$$\alpha_j^{es}(t) = b_j + r_j t \text{ with } \begin{cases} b_j = \sum_{f \in j} MFS_f + \frac{MFS_f}{BAG_f} J_f \\ r_j = \sum_{f \in j} \frac{MFS_f}{BAG_f} \end{cases}$$

3 BANDWIDTH RESERVATION METHODS

In this section, we first describe the optimization problem associated to the bandwidth reservation problem of the BLS shapers. Afterwards, we detail the two proposed methods to solve such a problem: Heuristic Deadline and Dichotomous Deadline methods.

3.1 Problem formulation

The aim is to find reserved BLS bandwidth within each multiplexer *mux* along the path of each flow $f \in RC$, minimizing the end-to-end delay bound of RC flow f, while fulfilling the following constraints:
(i) the **class rate constraint**, stating that in each output port *mux*, the input rate of an aggregate traffic of class j must be lower than the minimum guaranteed service rate, denoted R_j^{mux};
(ii) the **aggregate rate constraint**, stating that the load of an output port multiplexer is lower than the output link capacity C;
(ii) the **deadline constraint**, stating that the end-to-end delay bound of a flow f in class j ($EED_{j,f}$) must be lower than its deadline Dl_f.

If we consider testing N values for each BLS parameter (L_M^{mux}, L_R^{mux}, BW^{mux}) within m output ports for l flows, then we have a complexity of $O(l^m \cdot N^{3 \cdot m})$ for the whole network. In order to drastically reduce such a complexity, we will minimize the delay bound for the aggregate traffic in RC class within each output port, instead of conducting one minimisation per-path per-flow; thus reducing the complexity down to $O(m \cdot N^3)$.

Hence, we define a local deadline (resp. local delay bound) for class j in each output port multiplexer *mux*, denoted Dl_j^{mux} (resp. $delay_j^{mux}$), that has to be fulfilled by the aggregate traffic class $j \in \{SCT, RC\}$ in the output port *mux*:

$$Dl_j^{mux} \geqslant delay_j^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux})$$

Consider F_j^{mux} the set of flows of a class j in an output port multiplexer *mux*, the relaxed optimisation problem can be formulated as follows:

$$\begin{aligned} &\forall mux, \\ &\underset{L_M^{mux}, L_R^{mux}, BW^{mux}}{minimize} \; delay_{RC}^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) \\ &\text{s.t. } \forall mux, \forall j \in \{SCT, RC\} : \\ &(i)\; R_j^{mux} \geqslant \sum_{f \in F_j^{mux}} r_f \\ &(ii) \sum_{g \in F_{SCT}^{mux}} r_g + \sum_{f \in F_{RC}^{mux}} r_f \leqslant C \\ &(iii)\; Dl_j^{mux} \geqslant delay_j^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) \end{aligned} \tag{2}$$

Based on the guaranteed service curves for RC and SCT classes within *mux* defined in [3] and the basic Network Calculus theorem to compute the delay bound in each *mux*, i.e., the delay bound is the maximum horizontal distance between the arrival and service curves, the derived optimisation problem in (2) is a non-linear problem, with complex functions defining the delay bounds. There are many ways of solving such a problem numerically, such as brute force method, random search or heuristics. In our case, we will solve this problem based on heuristics taking advantages from conducted sensitivity analysis of our analytical model, which is not detailed in this paper due to the lack of space.

3.2 Solving the problem

Computing L_R^{mux}
The value of L_R^{mux} maximizing the minimum service rate of SCT defined in [3], while limiting the impact on RC traffic, is as follows:

$$L_R^{mux} = MFS_{RC} \cdot BW^{mux} \tag{3}$$

Computing L_M^{mux}

The delay bound of RC class within *mux* defined in [3] is as follows:

$$\begin{aligned} & delay_{RC}^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) \\ = \; & \frac{A}{(1 - BW^{mux}) \cdot C}\left(1 + \frac{BW^{mux}.(1 - BW^{mux}).MFS_{SCT}}{L_M^{mux} - L_R^{mux}}\right) \\ + \; & \frac{L_M}{(1 - BW^{mux}) \cdot C} + \frac{MFS_{SCT}}{C} - \frac{b_{RC}^{mux}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}} \end{aligned} \tag{4}$$

where $A = b_{RC}^{mux} + r_{RC}^{mux} \cdot \frac{b_{RC}^{mux}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}} + \max_{k \in \{SCT \cup RC \cup BE\}} MFS_k$ and n_{RC}^{links} the number of input links sending RC flows to *mux*.

To select the L_M^{mux} value minimising the delay function in Eq.(4), we compute the null point of the derived delay function; thus:

$$\begin{aligned} L_M^{mux} &= MFS_{RC} \cdot BW^{mux} \\ &+ \sqrt{A \cdot (1 - BW^{mux}) \cdot BW^{mux} \cdot MFS_{SCT}} \end{aligned} \tag{5}$$

Therefore, giving Eq.(3) and Eq.(5), we have reduced the number of unknown BLS parameters within *mux* to only one, i.e., BW^{mux}, to solve the optimisation problem in (2). To compute BW^{mux}, we propose Algorithm 1, which takes into account as inputs Dl_{SCT}^{mux} and Dl_{RC}^{mux}. We use a loop to compute the possible values for BW^{mux} in Line 2. Inside the loop, we compute the corresponding values of L_R^{mux}, i.e., lr, and L_M^{mux}, i.e., lm, in Lines 3 and 4. Then, we compute the SCT and RC delay bounds in Lines 5 and 6. Next, in Line 7, we verify the local deadlines conditions. If they are fulfilled, we store the delays and bw in Outputs, in Line 8. Finally, after testing all the bw in the loop, we select the BW^{mux} leading to the minimum RC delay bounds, in Line 12. If no BW^{mux} fulfills the condition, we return $+\infty$ for each delay bound. As we can notice, we need to define both local deadlines of SCT and RC within *mux* to enable Algorithm1.

Hence, we have defined two methods to compute these local deadlines: Heuristic Deadline (HD) and Dichotomous Deadline (DD) methods.

Algorithm 1 BLS Bandwidth Reservation algorithm in a multiplexer *mux* knowing the local deadlines: BLSparams()

Require: Dl^{mux}_{SCT}; Dl^{mux}_{RC}; b^{mux}_{SCT};r^{mux}_{SCT};MFS_{SCT}; b^{mux}_{RC};r^{mux}_{RC};MFS_{RC};MFS_{BE};
Ensure: BW^{mux},$delay^{mux}_{SCT}$,$delay^{mux}_{RC}$
1: Data=[Dl^{mux}_{SCT}; Dl^{mux}_{RC}; b^{mux}_{SCT};r^{mux}_{SCT};MFS_{SCT}; b^{mux}_{RC};r^{mux}_{RC};MFS_{RC};MFS_{BE}]
2: **for** bw ∈ [0.001 : 0.001 : 0.999] **do**
3: $lr = MFS_{RC} \cdot bw$
4: $l_m = L^{mux}_M$ in Eq.5
5: $d^{mux}_{SCT} = Delay^{mux}_{SCT}$(Data, bw, l_m, l_r)
6: $d^{mux}_{RC} = Delay^{mux}_{RC}$(Data, bw, l_m, l_r)
7: **if** $d^{mux}_{SCT} \leqslant Dl^{mux}_{SCT}$ and $d^{mux}_{RC} \leqslant Dl^{mux}_{RC}$ **then**
8: Outputs.add(bw,d^{mux}_{RC},d^{mux}_{SCT})
9: **end if**
10: **end for**
11: **if** notEmpty(Outputs) **then**
12: Outputs.get(IndexOfMinDRCs(Outputs))
13: **else**
14: [-,+∞,+∞] %no admissible parameters
15: **end if**

Heuristic Deadline Method

The main idea of HD method is to set the local deadlines of each class j, in any output port multiplexer mux_i along the path $path_f$ of a flow f, to take into account the port load along $path_f$. This is equivalent to the product of the sum of the multiplexer deadlines, denoted $\sum Dl^{mux_i}_j$, and the weight of class j rate going through multiplexer mux_i, relatively to the global class j rate in all the multiplexers mux in $path_f$:

$$\forall mux_i \in path_f, Dl^{mux_i}_j = \frac{\sum_{flw \in F^{mux_i}_j} r_{flw}}{\sum_{mux \in path_f} \sum_{flw \in F^{mux}_j} r_{flw}} \cdot \sum Dl^{mux_i}_j$$

Using Eq.(1), we have $\sum_{sw \in path_f} d^{sw}_{j,f} \leqslant Dl_{j,f} - d^{es}_j - d_{prop}$, with $d^{sw}_{j,f} = \frac{MFS_f}{C} + 1\mu s + delay^{mux_i}_{j,f}$:

$$\sum_{mux_i \in path_f} delay^{mux_i}_{j,f} \leqslant Dl_{j,f} - \sum_{mux_i \in path_f} (\frac{MFS_f}{C} + 1\mu s) - d^{es}_j - d_{prop} \leqslant \sum Dl^{mux_i}_j$$

$$\Rightarrow \sum Dl^{mux_i}_j = \min_{f \in F^{mux_i}_j, mux \in path_f} \left\{ Dl_{j,f} - \sum_{mux \in path_f} (\frac{MFS_f}{C} + 1\mu s) - d^{es}_j - d_{prop} \right\} \quad (6)$$

We use the minimum to reduce the complexity, even though this strengthens the constraint. This method has the benefice of being simple to use. However, it has the potential flaw of imposing a local deadline to class j in a multiplexer, without checking that the delay bounds in the other switches are able or not to reach their own deadlines. To cope with these limitations, we propose the second computation method, DD method, described in Algorithm 2.

Dichotomous Deadline Method

To compute the delay bound of class *SCT* for each output port multiplexer mux_i along the path of a flow f, we will use two deadline values: one leading to delay bounds equal or lower to the deadline, and one leading to delay bounds equal or higher than the deadline.

The first deadline is computed with the Heuristic Deadline method, and may lead to a lower SCT delay bound:

$$Dl^{under,mux_i}_{SCT} = \frac{\sum_{flw \in F^{mux_i}_{SCT}} r_{flw}}{\sum_{mux \in path_f} \sum_{flw \in F^{mux}_{SCT}} r_{flw}} \cdot \sum Dl^{mux_i}_{SCT}$$

To obtain a higher SCT delay bound, we consider the following SCT deadline:

$$Dl^{over,mux_i}_{SCT} = \sum Dl^{mux_i}_{SCT}$$

The Dichotomous Deadline (DD) method is detailed in Algorithm 2.

From Line 2 to Line 10, we initialise the dichotomous search. In Lines 3, 4 and 5, we compute the initial Deadlines, i.e., Dl^{mux}_{RC}, $Dl^{over,mux}_{SCT}$ and $Dl^{under,mux}_{SCT}$ for each mux. This leads in Lines 6 and 7 to the computation of the corresponding SCT delay bounds $delay^{over,mux}_{SCT}$ and $delay^{under,mux}_{SCT}$ using Algorithm 1. After all the mux have been considered, we compute in Lines 9 and 10 the two dichotomous variables: $\sum Dl^{under,mux}_{SCT}$ and $\sum Dl^{over,mux}_{SCT}$.

Then in Line 11, we check whether the value $\sum_{mux \in path_f} delay^{over,mux}_{SCT}$ is actually lower than $\sum Dl^{mux}_{SCT}$. If not, we return $Dl^{over,mux}_{SCT}$ since a dichotomous search is not possible. Else, we start the dichotomous search, where $\sum Dl^{mux}_{SCT}$ is bounded by $\sum_{mux \in path_f} delay^{over,mux}_{SCT}$ and $\sum_{mux \in path_f} delay^{under,mux}_{SCT}$.

In Line 14, we set the stop condition using an $\epsilon << 0$ such as: $\sum Dl^{mux}_{SCT} - \sum_{mux \in path_f} delay^{under,mux}_{SCT} \geqslant \epsilon$.

The objective is to find a solution as close as possible to the $\sum Dl^{mux}_{SCT}$ and respecting the deadline constraint.

Then, we start the next iteration in Line 15, by computing the current $\sum Dl^{cur,mux}_{SCT}$. We use it to compute the local deadlines in each mux and the resulting SCT delay bounds in Lines 17 and 18.

The final steps consist in determining whether $\sum_{mux \in path_f} delay^{cur,mux}_{SCT}$ (the sum of the current delay bounds) is lower or higher than $\sum Dl^{mux}_{SCT}$ in Line 20. Then, we redefine the values of the current loop, either $\sum Dl^{over,mux}_{SCT}$ in Line 21, or $\sum Dl^{under,mux}_{SCT}$ and $Dl^{under,mux}_{SCT}$ in Lines 23 and 24.

4 PERFORMANCE EVALUATION

In this section, we first describe our case study. Afterwards, we assess the efficiency of the introduced bandwidth reservation methods to enhance the extended AFDX performance, in comparison to an intuitive method, since there is no other existing methods.

4.1 Case study

Our case study is a representative avionics communication architecture of the A380, based on a 1-Gigabit AFDX backbone network,

Algorithm 2 Local deadline computation algorithm with dichotomous method, along the path of flow f: DichotomousDeadline()

Require: $\forall mux \in path_f, \forall g \in F_k^{mux}, k \in \{SCT, RC\}, MFS_g, r_g, b_g, \sum Dl_k^{mux}$
Ensure: $\forall mux, Dl_{SCT}^{mux}, Dl_{RC}^{mux}$
1: Data=[$\forall mux \in path_f$, Dl_{RC}^{mux}; $\forall g \in F_k^{mux}$,$k \in \{SCT, RC\}$, MFS_g, r_g, b_g]
2: **for** $\forall mux$ in $path_f$ **do**
3: Dl_{RC}^{mux}= RCHeuristicDeadline($\sum Dl_{RC}^{mux}$,data)
4: $Dl_{SCT}^{under,mux}$= SCTHeuristicDeadline($\sum Dl_{SCT}^{mux}$,data);
5: $Dl_{SCT}^{over,mux} = \sum Dl_{SCT}^{mux}$
6: $delay_{SCT,}^{under,mux}$=BLSparams($Dl_{SCT}^{under,mux}$,$data$).delay
7: $delay_{SCT}^{over,mux}$=BLSparams($Dl_{SCT}^{over,mux}$,$data$).delay
8: **end for**
9: $\sum Dl_{SCT}^{over,mux} = \sum_{mux \in path_f} Dl_{SCT}^{over,mux}$
10: $\sum Dl_{SCT}^{under,mux} = \sum_{mux \in path_f} Dl_{SCT}^{under,mux}$
11: **if** $\sum_{mux \in path_f} delay_{SCT}^{over,mux} \leqslant \sum Dl_{SCT}^{mux}$ **then**
12: return $\forall mux \in path_f, Dl_{RC}^{mux}, Dl_{SCT}^{over,mux}$
13: **else if** $\sum_{\forall mux \in path_f} delay_{SCT}^{over,mux} \geqslant \sum Dl_{SCT}^{mux} \geqslant \sum_{\forall mux \in path_f} delay_{SCT}^{under,mux}$ **then**
14: **while** $\sum Dl_{SCT}^{mux} - \sum_{mux \in path_f} delay_{SCT}^{under,mux} \geqslant \epsilon$ **do**
15: $\sum Dl_{SCT}^{cur,mux} = (\sum Dl_{SCT}^{over,mux} + \sum Dl_{SCT}^{under,mux})/2$
16: **for** $\forall mux$ in $path_f$ **do**
17: $Dl_{SCT}^{cur,mux}$= SCTHeuristicDeadline($\sum Dl_{SCT}^{cur,mux}$, data);
18: $delay_{SCT}^{cur,mux}$=BLSparams($Dl_{SCT}^{cur,mux}$,$data$).delay
19: **end for**
20: **if** $\sum_{\forall mux \in path_f} delay_{SCT}^{cur,mux} \geqslant \sum Dl_{SCT}^{mux}$ **then**
21: $\sum Dl_{SCT}^{over,mux} = \sum Dl_{SCT}^{cur,mux}$
22: **else**
23: $\sum Dl_{SCT}^{under,mux} = \sum Dl_{SCT}^{cur,mux}$
24: $\forall mux \in path_f, Dl_{SCT}^{under,mux} = Dl_{SCT}^{cur,mux}$
25: **end if**
26: **end while**
27: **end if**
28: return $\forall mux \in path_f, Dl_{RC}^{mux}, Dl_{SCT}^{under,mux}$

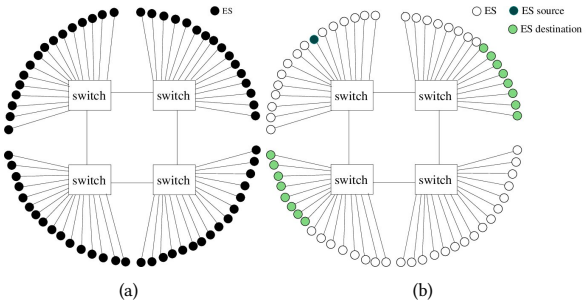

Figure 3: Representative AFDX network: (a) Architecture; (b) Traffic communication patterns

which consists of 4 switches and 64 end-systems as shown in Fig. 3 (a). Each circulating traffic flow on the backbone network is a multicast flow with 16 destinations, and crosses two successive switches before reaching its final destinations. The first switch in the path receives traffic from 16 end-systems to forward it in a multicast way to its two neighboring switches. Afterwards, the second switch in the path, which receives traffic from the two predecessor switches, forwards the traffic in its turn to the final end-systems. Each end-system receives data from 16 end-systems. Figure 3 (b) shows the traffic communication patterns between the source and the final destinations of a given flow.

In this multi-hop network, each end-system es generates n_i^{es} flows of type $i \in \{SCT, RC, BE\}$. We consider that all end-systems are identical and each generates the same number of flows n_i^{es}.

As a consequence, the utilisation rate in both the first and second switches is the bottleneck utilisation rate (i.e. maximum utilisation rate along a path) for each type of traffic $i \in \{SCT, RC, BE\}$, $UR_i^{bn} = 16 \cdot n_i^{es} \cdot \frac{MFS_i}{BAG_i} \cdot \frac{1}{C}$.

We consider the traffics SCT, RC and BE defined in Table 2 and the scenarios detailed in Table 3.

Concerning the intuitive method, we consider the following parameters: we set the reserved bandwidth to the bottleneck utilisation rate $BW = UR_{SCT}^{bn}$, and $L_R = MFS_{RC} \cdot BW$ and $L_M = 80 \cdot MFS_{SCT} \cdot (1 - BW)$, to enable the transmission of a maximum SCT burst of 80 frames within the BLS, i.e., a generated burst of 5 SCT flows per end-system.

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 2: Avionics flow Characteristics

Scenarios	$Scenario_{SCT}$	$Scenario_{RC}$
$(UR_{SCT}; UR_{RC})(\%)$	$([0.1..80]; 20)$	$(20; [0.5..80])$
$(n_{SCT}^{es}; n_{RC}^{es})$	$([1..192]; 10)$	$(49; [1..39])$

Table 3: Considered Test Scenarios

4.2 Numerical Results

In this section, we study the impact of optimized BLS bandwidth compared to the intuitive one on a multi-hop network. Hence, we compare the delay bounds of SCT and RC based on the extension of the timing analysis in [3] to the multi-hop case, under HD and DD methods in reference to the intuitive method. The results for the two scenarios are illustrated in Figures 4 and 5. It is worth noting that we only present the admissible results, i.e., when all the deadlines are fulfilled.

First, concerning the maximum bottleneck utilisation rates:
(i) in Figure 4, we note that the maximum bottleneck SCT utilisation rate is 32% with the intuitive method, 36% with HD method, and 40% with DD method;
(ii) in Figure 5, the maximum bottleneck RC utilisation rate is 28% with the intuitive method, and 40% with both HD and DD methods.

These results show an improvement of the SCT (resp. RC) schedulability up to 25% (resp. 42%) under the optimized bandwidth reservation methods, in comparison to the intuitive one.

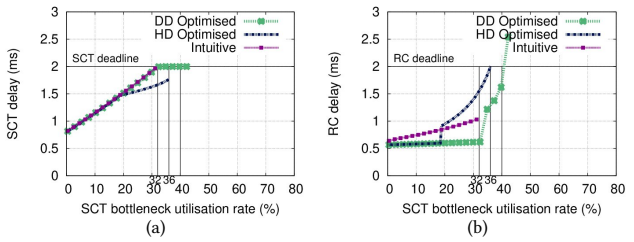

Figure 4: Intuitive vs Optimisations for $Scenario_{SCT}$: (a) SCT delay bound; (b) RC delay bound

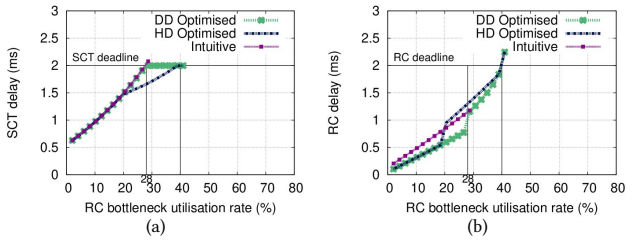

Figure 5: Intuitive vs Optimisations for $Scenario_{RC}$: (a) SCT delay bound; (b) RC delay bound

Secondly, in Figure 4(b), the RC delay bounds with HD method are lower than the delay bounds with the intuitive one until UR_{SCT} = 18%. However, for UR_{SCT} between 18% and 32%, the intuitive method is better than HD one. The same issue is visible in Figure 5(b) for UR_{RC} between 20% and 28%.

To understand the reasons of this issue, in Figure 6, we present separately the SCT delay bound in the first and in the second switch output ports, denoted $delay_{SCT}^{mux1}$ and $delay_{SCT}^{mux2}$. In Figure 6(a), we can separate the SCT delay bounds under HD method in 4 areas, with UR_{SCT}:
(i) between 0% and 20%, $delay_{SCT}^{mux1} < Dl_{SCT}^{mux1}$ and $delay_{SCT}^{mux2} < Dl_{SCT}^{mux2} \Rightarrow EED_{SCT} < Dl_{SCT}$.The multiplexer deadline is reached in neither switches;
(ii) between 20% and 45%, $delay_{SCT}^{mux1} = Dl_{SCT}^{mux1}$, $delay_{SCT}^{mux2} < Dl_{SCT}^{mux2} \Rightarrow EED_{SCT} < Dl_{SCT}$. The switch output port 1 deadline (about 1ms, see Figure 6(a)) is reached and the SCT delay bound remains at this deadline Dl_{SCT}^{mux1} until UR_{SCT} = 50% (see Figure 6(a)). However, in the second switch output port, the SCT delay remains firmly below its deadline;
(iii) between 45% and 50%, $delay_{SCT}^{mux1} = Dl_{SCT}^{mux1}$, $delay_{SCT}^{mux2} = Dl_{SCT}^{mux2} \Rightarrow EED_{SCT} = Dl_{SCT}$. The SCT end-to-end delay bound is equal to the end-to-end deadline, as the delays in both output ports are equal to their respective deadlines (see Figure 6);
(iv) between 50% and 80%, $delay_{SCT}^{mux1} > Dl_{SCT}^{mux1}$ and $delay_{SCT}^{mux2} = Dl_{SCT}^{mux2} \Rightarrow EED_{SCT} > Dl_{SCT}$. The end-to-end delay bound is higher than the end-to-end deadline as the delay in the switch output port 1 is higher than its deadline (see Figure 6).

Hence, this highlights the fact that limiting the local deadline in an output port without taking into account the state of other ones in the path decreases the performance of the RC delay bounds.

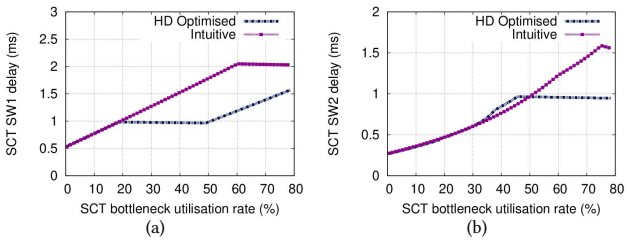

Figure 6: Intuitive vs Heuristic Deadline Optimisation for $Scenario_{SCT}$: (a) Switch 1 (b) Switch 2

Contrary to the HD method, the DD method takes into account the output ports along the flow path. In Figures 4 and 5, the RC delay bounds with DD method are better than the ones with both intuitive and HD methods. For instance at UR_{SCT} = 32%, the RC delay bound is improved by 49% with reference to the intuitive one, and by 74% compared to HD method.

We can conclude from these results that Dichotomous Deadline method leads to a great improvement over both the intuitive and Heuristic Deadline methods. The schedulability of SCT is actually increased by up to 31% and the RC delay bound is decreased by up to 75%. Nevertheless, the Dichotomous Deadline method needs much higher computation times, in comparison to Heuristic Deadline method, e.g., up to 10 ***times.***

5 CONCLUSIONS

In this paper, we have proposed two optimized bandwidth reservation methods for TSN/BLS shapers in an extended AFDX, denoted Heuristic Deadline and Dichotomous Deadline methods, to enhance the network performance, in terms of schedulability and delay bounds. The conducted performance evaluation on a realistic avionics case study highlights the benefit of using such methods, and particularly the Dichotomous Deadline method. The latter leads to a great improvement over both the intuitive and Heuristic Deadline methods, in terms of schedulability (up to 31% for SCT) and delay bounds (up to 75% for RC), but at the expense of higher computation times, e.g., up to 10 times.

As a next step, we will generalize such analyses to an extended AFDX with multiple BLS classes to offer higher configuration flexibility.

REFERENCES

[1] M. Ashjaei, G. Patti, M. Behnam, T. Nolte, G. Alderisi, and L. Bello. Schedulability analysis of Ethernet Audio Video Bridging networks with scheduled traffic support. *Real-Time Systems*, 53(4):526–577, 2017.
[2] Philip Axer, Daniel Thiele, Rolf Ernst, and Jonas Diemer. Exploiting shaper context to improve performance bounds of ethernet avb networks. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.
[3] A. Finzi, A. Mifdaoui, E. Lochin, and F. Frances. Incorporating TSN/BLS in AFDX for Mixed-Criticality Applications: Model and Timing Analysis. WFCS, 2018.
[4] A. Finzi, A. Mifdaoui, E. Lochin, and F. Frances. Mixed-Criticality on the AFDX Network: Challenges and Potential Solutions. In *ERTS2*, 2018.
[5] Franz-Josef Gotz. Traffic Shaper for Control Data Traffic (CDT). *IEEE 802 AVB Meeting*.
[6] TSN Task Group. TSN Specifications.