



**HAL**  
open science

## Efficient configuration of a QoS-aware AFDX network with Deficit Round Robin

Aakash Soni, Jean-Luc Scharbarg, Jérôme Ermont

► **To cite this version:**

Aakash Soni, Jean-Luc Scharbarg, Jérôme Ermont. Efficient configuration of a QoS-aware AFDX network with Deficit Round Robin. 18th IEEE International Conference on Industrial Informatics, Jul 2020, Warwick, United Kingdom. hal-02965556

**HAL Id: hal-02965556**

**<https://hal.science/hal-02965556>**

Submitted on 13 Oct 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Efficient configuration of a QoS-aware AFDX network with Deficit Round Robin

Aakash Soni

University of Toulouse - France

Aakash.Soni@irit.fr

Jean-Luc.Scharbarg

University of Toulouse - France

Jean-Luc.Scharbarg@enseeiht.fr

Jérôme Ermont

University of Toulouse - France

Jerome.Ermont@enseeiht.fr

**Abstract**—AFDX is the de facto communication standard in avionics domain. It is primarily used for the transmission of critical avionic flows. The mandatory certification process requires to upper bound the end-to-end transmission delay for each critical flow. Therefore, worst-case traversal time analysis has been implemented. However, it leads to a very lightly loaded network (up to 10%) as it considers very rare worst-case situations. Introducing a QoS mechanism is often a good solution to improve network utilisation since it allows differentiating critical flows based on their constraints as well as the transmission of less/non-critical flows with bounded impact on critical ones. Deficit Round Robin is such a mechanism and it is envisioned for future avionic networks. Using this mechanism, we propose to share the bandwidth between  $n - 1$  classes for critical flows and one class for non-critical ones. Therefore the contributions of this paper are (1) to propose a quantum assignment that ensures that critical flows always respect their deadlines and maximises the bandwidth for non-critical ones and (2) to propose a heuristic for the distribution of the flows among different classes.

**Index Terms**—Quantum Allocation, Deficit Round Robin, WCTT analysis, AFDX, Switched Ethernet, Quality of Service

## I. INTRODUCTION

Avionics Full Duplex switched Ethernet (AFDX) [1] has become the de facto standard for the transmission of (critical) avionic flows. It has been tailored to meet avionic constraints and a Worst-Case Traversal Time analysis (WCTT) allows the computation of an upper bound for the end-to-end delay of each flow. This analysis is based on Network Calculus (NC) for the certification of A380 and A350 backbones. It makes pessimistic assumptions that leads to a very lightly loaded network. Typically, less than 10% of the available bandwidth is used for the transmission of avionic flows on an AFDX network embedded in an aircraft. A classical solution to improve utilisation of the network is to introduce Quality-of-Service (QoS) mechanisms. First, it can decrease worst-case delay for the most constrained avionic flows. Second, less/non-critical additional flows can be transmitted on the network with bounded impact on avionic ones. Existing AFDX switches implement Fixed Priority service discipline with two priority levels. Up to now, they are rarely used. Thus avionic flows are transmitted, following a FIFO service discipline.

It has been shown [2] that distributing avionic flows between these two priority levels can significantly decrease the highest end-to-end delay that any flow can experience. These preliminary results are promising. Further studies have shown that Deficit Round Robin (DRR) allows an efficient and safe

bandwidth sharing between critical avionic flows and non-critical additional ones [3].

DRR [4] achieves fair sharing of the capacity of a server among several flow classes, thanks to quanta which are allocated to classes per round. Several papers address the analysis of DRR latency bound [5]–[7]. In particular, the analysis of a server with DRR scheduler based on NC is proposed in [5] and optimised in [7].

The work in [3] is based on the WCTT analysis of [5]. It considers a switched Ethernet network with DRR scheduler, shared by critical and non-critical flows. It considers  $n - 1$  classes for critical flows and one for non-critical ones. [3] assumes that flows have been assigned to classes. It determines a quantum assignment which guarantees that critical flows respect their deadlines (valid assignment) and maximises the bandwidth allocated to non-critical flows. A similar problem has been addressed in the context of non-critical flows with Weighted Round Robin (WRR) [8]. To the best of our knowledge, the only work that addresses this problem is [3]. It suffers from two main limitations.

- First, it does not benefit from the optimised NC WCTT analysis for DRR presented in [7]. Indeed it is based on assumptions that do not hold for this optimised analysis.
- Second, it does not propose an assignment of flows to classes. Such an assignment makes sense for avionic flows. The goal is to find the best valid assignment in terms of bandwidth for non-critical flows.

This paper integrates both aspects in the approach in [3]. First, we redesign the quantum assignment algorithm in [3] so that it can cope with the WCTT analysis in [7]. Second, we propose a heuristic for the allocation of critical flows to classes.

Section II presents the context of the study and states the problem. An improved quantum assignment is proposed in Section III. Flow distribution among classes is addressed in Section IV. The proposed solution is evaluated on a realistic case study in Section V. Section VI concludes the paper and gives some directions for future works.

## II. CONTEXT OF THE STUDY

AFDX network interconnects a set of end-systems by switches via full-duplex links. End-systems are the sources and destinations of flows statically defined as virtual links (VL). Each flow  $v_i$  is constrained by a minimum interval  $BAG_i$  between any two consecutive frames at end system level, a

maximum ( $l_i^{max}$ ) and a minimum ( $l_i^{min}$ ) frame length. The path  $\mathcal{P}_i$  followed by  $v_i$  is statically defined with a bounded switching latency  $sl$  in each switch. Unlike TSN, AFDX is an asynchronous network: it doesn't provide a global clock to avionics systems in order to mitigate certification issue.

Critical avionic flows consume a small part of the available bandwidth. Thus it is envisioned to use the free bandwidth to transmit additional less/non-critical flows which are presently transmitted on other networks. Obviously, the impact of these additional flows on the critical ones must be bounded in order to guarantee that critical flows never miss deadlines.

Fixed priority service discipline with two priority levels is implemented in existing AFDX switches and it can be used to that purpose. Critical flows are assigned the high priority and non-critical flows the low one. However, this solution presents limitations [3]. First, non critical flows have to wait as long as there are high priority ones to be served. It can lead to long delays in case of bursts of high priority frames. However, the constraint is to transmit critical flows within their deadlines. Thus critical frames could be delayed by non-critical ones, provided that it does not jeopardise the respect of their deadlines. Second, assigning all critical flows to the same priority means that they are not differentiated. It can be quite inefficient [3]: critical flows should be treated differently, depending on their deadlines.

DRR scheduler is promising to mitigate these limitations [3]. It is an approximation of fair queuing technique that allows each flow passing through a network device to get a fair share of network bandwidth at a very low implementation complexity. Therefore, an enhanced switch implementing DRR policy is under consideration, leading to a QoS aware AFDX network. In such a network, a set of  $n$  queues is associated with each output port  $h$ . Each queue stores the frames of a given traffic class.

Let us illustrate the operation of DRR scheduling on the network example in Figure 1. Two switches ( $S_1$  and  $S_2$ ) interconnect eight end-systems ( $e_1$  to  $e_8$ ). Twenty one (critical) avionic flows ( $v_1$  to  $v_{21}$ ) and one (non-critical) additional flow ( $v_{22}$ ) are transmitted on this network. Each switch output port implements a DRR scheduler. The maximum link bandwidth is  $R = 100$  Mb/s. Flow features are given in Table I. An arbitrary distribution of these flows is presented in Table II where the deadline in each class is limited to its smallest flow deadline (i.e.  $v_1$  ( $300\mu\text{sec}$ ) for  $C_1$ ,  $v_6$  ( $600\mu\text{sec}$ ) for  $C_2$  and  $v_{14}$  ( $900\mu\text{sec}$ ) for  $C_3$ ).

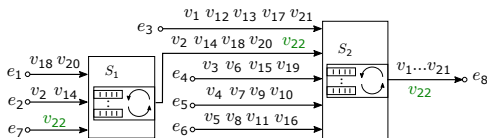


Fig. 1. AFDX Network Example

One possible scenario for DRR scheduling at  $S_2$  is shown in Figure 2 where frames of critical flows  $v_1, v_2, v_3, v_7, v_6, v_8, v_9, v_{14}, v_{16}, v_{15}, v_{17}$  arrive at the output port of  $S_2$  simultaneously and they are arbitrarily buffered in this order.

TABLE I  
VL PARAMETERS IN NETWORK EXAMPLE (FIGURE 1)

Flow	$BAG_i$	$l_i^{max}$	DL	Flow	$BAG_i$	$l_i^{max}$	DL
$v_1$	128	99	300	$v_{12}$	128	100	660
$v_2$	64	100	310	$v_{13}$	128	100	890
$v_3$	64	99	340	$v_{14}$	64	99	900
$v_4$	64	100	390	$v_{15}$	64	99	900
$v_5$	64	100	590	$v_{16}$	64	100	910
$v_6$	128	100	600	$v_{17}$	128	99	920
$v_7$	128	99	600	$v_{18}$	64	100	930
$v_8$	128	99	610	$v_{19}$	64	99	930
$v_9$	128	99	620	$v_{20}$	128	100	950
$v_{10}$	64	100	650	$v_{21}$	64	100	950
$v_{11}$	64	99	650	$v_{22}$	-	100	-

Units:  $BAG$  in  $msec$ .  $l_i^{max}$  in  $bytes$ . Deadline (DL) in  $\mu\text{sec}$ .  
For all flows  $l_i^{min} = 80$  bytes.

TABLE II  
FLOW CLASSIFICATION (ARBITRARY)

Category	Class	Flows	Frame length	Deadline
Critical	$C_1$	$v_1 - v_5$	80 - 100	300
	$C_2$	$v_6 - v_{13}$	80 - 100	600
	$C_3$	$v_{14} - v_{21}$	80 - 100	900
Non-Critical	$C_{BE}$	$v_{22}$	80 - 100	-

We assume maximum traffic in non-critical class, which means there is at least one frame from  $v_{22}$  in  $C_{BE}$  buffer at all time.

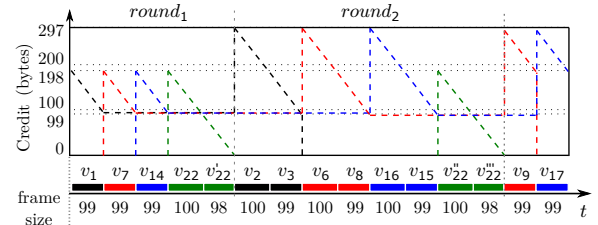


Fig. 2. DRR scheduling scenario at  $S_2$

DRR scheduler serves the classes in rounds. Each class  $C_i$  is allocated a quantum of  $Q_{C_i}$  bytes. In Figure 2, a quantum of 198 bytes is considered for each class. DRR scheduler maintains a deficit counter per class, that represents the number of bytes (a.k.a credit) that an active class is allowed to transmit each time it is selected by the scheduler. Initially, the deficit  $\Delta_{C_i}$  for all the classes is 0. Each time a class  $C_i$  is selected, its credit is set to  $\Delta_{C_i} + Q_{C_i}$ . Then,  $C_i$  can transmit frames as long as its queue is not empty and its credit is larger than the head-of-line frame in its queue. On each transmission of a frame, the credit is reduced by the frame size. If the queue becomes empty, the remaining credit is discarded, otherwise, it is preserved in  $\Delta_{C_i}$ . Then the next class is selected to be served.

In Figure 2, we assume that  $C_1$  is selected first. It gets a credit of 198 bytes (its quantum value). Frame from  $v_1$  is first in the queue and it is transmitted, since its size (99 bytes) is less than the available credit. Since the remaining credit of 99 bytes is not enough for the transmission of next frame ( $v_2$ ) in  $C_1$ , it is preserved in  $\Delta_{C_1}$  to be utilised in the next round. Therefore, the scheduler moves to the next active class  $C_2$  which also gets a credit of 198 bytes. It allows the transmission

of  $v_7$  frame but not  $v_6$  one and the deficit is 99 bytes. A similar scenario occurs in  $C_3$  that allows transmission of  $v_{14}$ . In  $C_{BE}$ , the first two frames from  $v_{22}$  are in total 198 bytes which is equal to its credit. Thus they are transmitted and the deficit is 0. In the next round,  $C_1$  gets a credit of 99 (its deficit) + 198 (its quantum) = 297 bytes.  $C_1$  consumes 199 bytes for the transmission of  $v_2$  and  $v_3$  frames. Since  $C_1$  queue is now empty its deficit is set to 0. Similarly,  $C_2$  and  $C_3$  also get 297 bytes of credit sequentially, which allows transmission of  $v_6$ ,  $v_8$ ,  $v_{16}$  and  $v_{15}$  leaving deficit of 98 bytes and so on.

In order to allow a class  $C_x$  to be served at least once in each round its quantum cannot be less than its maximum frame length  $l_{C_x}^{max}$ , so

$$Q_{C_x} \geq l_{C_x}^{max} \quad (1)$$

Therefore each class with critical traffic can be assigned the bandwidth it needs to meet the deadlines of its traffic. The remaining bandwidth (in each round) is available for non-critical flows and they do not have to wait until there is no more pending critical traffic. Several classes can be reserved for critical flows, making it possible to treat differently the flows with different deadlines.

In contrast to WRR [9] that shares bandwidth, based on number of frames (without considering their lengths), DRR allows fair sharing of bandwidth among variable length flow frames. Moreover in DRR scheduler, if a flow experiences a deficit of service due to insufficient credit in a given round then this deficit is compensated in the next round.

A WCTT analysis is mandatory in order to guarantee that critical flows always respect their deadline. Such an analysis is presented in the next paragraph.

#### A. DRR WCTT analysis

The WCTT analysis for DRR was proposed in [5] and optimised for AFDX in [7]. Both approaches are based on NC and are now summarised.

The NC is a mathematical framework, based on the (min, +) algebra, applied to guaranteed service networks in [10] to compute guaranteed backlog and delay bounds. The delay computations are based on piecewise-linear curves. In NC, the traffic and network elements are represented by arrival curves and service curves respectively.

Based on [5], in a switch deploying DRR scheduler, the worst-case delay  $D_i^h$  for a flow  $v_i$  of  $C_x$  at  $h$  is upper bounded by the maximum horizontal difference between the cumulative curve ( $\alpha_{C_x}^h$ ) of the arrival curves of  $C_x$  flows in  $h$  and the service curve  $\beta_{C_x}^h$  offered to  $C_x$ . These curves are shown in Figure 3 (left).

Since a DRR scheduler shares the network bandwidth among all the classes, each class  $C_x$  gets only a fraction  $\rho_{C_x}^h$  of the link rate  $R$  and experiences a scheduler latency  $\Theta_{C_x}^h$ . Thus  $C_x$  service is a sequence of periods where it is being served at full rate and periods where it is waiting for its turn. It leads to a staircase curve, as shown in Figure 3 (left). Such a staircase curve can lead to high mathematical complexity in delay computation. Therefore, the residual service curve for

$C_x$  at  $h$  is derived in [5] as an under-approximation of this staircase curve which is given as

$$\beta_{C_x}^h(t) = \rho_{C_x}^h \times [t - \Theta_{C_x}^h - sl]^+ \quad (2)$$

The bandwidth fraction  $\rho_{C_x}^h$  depends on the quantum assigned to  $C_x$  and the sum of quantum assigned to all the classes and is given by:

$$\rho_{C_x}^h = \frac{Q_{C_x}^h}{\sum_{1 \leq j \leq n} Q_j^h} \times R \quad (3)$$

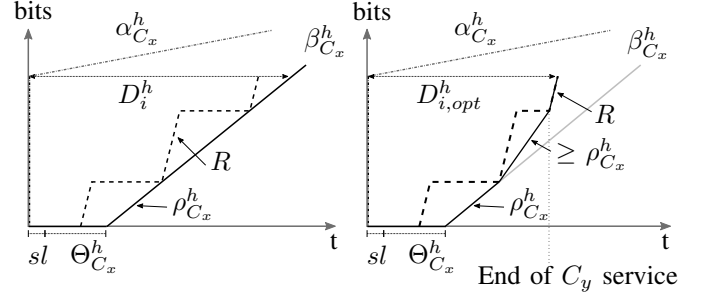


Fig. 3. Delay upper bound computation in NC

It has been shown in [7] that the computation of delay  $D_i^h$  based on  $\beta_{C_x}^h$  of [5] can be too pessimistic as it does not make any assumption on the volume of competing traffic from other classes. It assumes that the competing classes  $C_y$  ( $y \neq x$ ) remain active during the active period of  $C_x$  flows and they consume their maximum credit in each DRR service round. Hence the residual service for  $C_x$  is  $\beta_{C_x}^h$  (which depends only on  $Q_x$  and sum of quantum assigned to all the classes but not on the distribution of this sum between these classes). It might not be the case. In some cases, the traffic in  $C_y$  can be too low to remain active and to consume their full credit during the active period of  $C_x$ . So, the residual service of  $C_x$  can be much higher (illustrated in Figure 3 (right)) and it is not independent of the quantum distribution between competing classes. Thus [7] optimises the computation by quantifying the maximum amount of traffic of each competing class within the delay bound computed by [5]. Therefore the optimised delay in [7] is obtained by subtracting the time required to transmit the overestimated traffic assumed by  $\beta_{C_x}^h$  in the duration  $[0, D_i^h]$ . This optimised delay is given by

$$D_{i,opt}^h = D_i^h - \frac{\sum_{y=1, y \neq x}^{n^h} \text{over-estimated portion of } C_y \text{ traffic}}{R} \quad (4)$$

And the end-to-end delay upper bound of  $v_i$  can be computed by:

$$D_{i,opt}^{ETE} = \sum_{h \in \mathcal{P}_i} D_{i,opt}^h \quad (5)$$

where  $\mathcal{P}_i$  is the path followed by  $v_i$ .

Since [7] takes into account the volume of traffic from competing classes and not an over-approximated traffic, the obtained delay bound is tighter. For more information on  $D_{i,opt}^h$  computation, readers may refer to [7].

## B. Problem statement

As briefly explained earlier, DRR scheduling allows to differentiate critical flows with different deadlines by assigning them to different classes. Then the bandwidth assigned to these critical classes can be tailored to the minimum value which guarantees that no deadlines are missed. All the remaining bandwidth is available for non-critical flows. The idea is to maximise the quantity of additional flows that can be served before critical flows such that the delay of critical flows is increased but do not exceed the deadline.

Two points have to be addressed:

- distribute critical flows among critical classes,
- find the best possible quantum assignment for these classes, i.e. one that maximises the bandwidth available for non-critical flows while ensuring no missed deadlines for critical ones.

The second point has been addressed in [3]. However, the proposed solution is based on the basic NC approach of [5]. In the next section, we present an extension which is based on the optimised NC approach of [7]. Such an extension is not trivial, since the quantum assignment in [3] is based on assumptions that do not hold for the NC approach of [7].

We address the first point in Section IV. This flow distribution has a big impact on results, as it will be shown with the results in Table V.

## III. QUANTUM ASSIGNMENT

First we recall the existing solution proposed in [3]. Then we show how it can be extended in order to use the optimised WCTT analysis of [7]. The goal is to significantly increase the residual service of non-critical class  $C_{BE}$ .

### A. Overview of the existing quantum assignment approach

The algorithm in [3] is based on two fundamental properties:

- The worst-case delay computation by NC of [5] is independent of the quantum distribution among competing classes (Lemma 3.1 in [3]). It means that, in order to compute the end-to-end delay bound in  $C_x$ , only  $Q_{C_x}$  (a portion of  $Q$ ) and  $Q$  (the sum of quantum distributed among all the classes) is needed.
- The part of  $Q$  available to the non-critical class  $C_{BE}$  is higher for the smaller values of  $Q$  provided that each critical class is assigned the minimum quantum which guarantees that no deadline is missed, based on the WCTT analysis of [5] (Lemma 3.2 in [3]).

Based on these properties the basic idea of the proposed algorithm is to give minimum (yet sufficient) fraction of quantum sum to critical classes so that the residual quantum of the non-critical class is maximised. The basic parts of this algorithm are presented in Algorithm 1.

The process starts with an initial value  $InitVal$  for the sum of quanta  $Q$  (line 1). The minimum portion  $Q_{C_i}$  (called valid quantum) of  $Q$  required by each critical class  $C_i$  is computed. The constraint is that no  $C_i$  frame misses its deadline (line 5, function QUANTUMMIN).

The process stops as soon as the unused portion  $Q_{Resid}$  of  $Q$  is smaller than the computed  $Q_{C_i}$  value (line 6-7) or all the critical classes have been treated (line 4-8). In the latter case, the unused portion  $Q_{Resid}$  is assigned to non-critical class  $C_{BE}$  (line 9) and a solution has been obtained. The solution is improvable (line 10) if the quantum in each class is greater than the minimum possible value, i.e. its maximum frame size (equation 1). The improved  $Q$  is obtained (line 11) by dividing it by a factor representing the smallest ratio of the quantum of a class to its maximum frame size.

The outcome of this algorithm is the distribution of the smallest  $Q$  among all the classes such that every critical flow respects its deadline and the residual service of non-critical frames is maximised.

---

### Algorithm 1: Quantum assignment algorithm in [3]

---

```

input : Initial sum of quanta ( $ValInit$  : Integer)
output: Per class quantum ( $Q_{C_1}, Q_{C_2} \dots$  : Integer)
Data: Remaining quantum ( $Q_{Resid}$ )
1  $Q \leftarrow ValInit$  ;
2 while true do
3    $Q_{Resid} \leftarrow Q$  ;
4   for each critical class  $C_i$  do
5      $Q_{C_i} \leftarrow \text{QUANTUMMIN}(C_i, Q)$ ;
6     if  $Q_{C_i} > Q_{Resid}$  then
7       | terminate;
8      $Q_{Resid} \leftarrow Q_{Resid} - Q_{C_i}$ ;
9    $Q_{C_{BE}} \leftarrow Q_{Resid}$ ;
10  if ISIMPROVABLE() then
11  | IMPROVE(Q);
12  else
13  | break;

```

---

Let us illustrate the operation of this algorithm on the network example given in Figure 1 with flows divided into 3 critical classes and 1 non-critical class as shown in Table II. The maximum service rate at each output port is  $R = 100$  bits/ $\mu\text{sec}$ .

Initially, let us assume a sum of quanta  $Q = 741$  bytes. The process starts with critical class  $C_1$  whose deadline is 300  $\mu\text{sec}$ . The minimum quantum needed by  $C_1$  to meet all its deadline is determined, thanks to a binary search between the maximum possible value (741 bytes) and the maximum  $C_1$  frame size. We get  $Q_1 = 270$  bytes.

The minimum quanta for  $C_2$  and  $C_3$  are determined with a similar process. We get  $Q_{C_2} = 106$  bytes and  $Q_{C_3} = 100$  bytes. Thus, the residual quantum for non-critical class is  $Q_{C_{BE}} = 741 - (270+106+100) = 265$  bytes, which is  $\frac{265}{741} \times 100 = 35.76\%$  of  $Q$ .

Since a valid quantum distribution for each class is obtained and at least one class ( $C_3$ ) receives a quantum equal to its maximum frame length, the algorithm returns the quantum distribution  $Q_{C_1} = 270$ ,  $Q_{C_2} = 106$ ,  $Q_{C_3} = 100$  and  $Q_{C_{BE}} = 265$  as its solution.

## B. Improved quantum assignment

As mentioned earlier, [3] relies on WCTT analysis based on NC approach of [5] which is shown to be pessimistic and optimised in [7].

Due to this pessimism, the delay bounds computed in Algorithm 1 are much higher than the actual delays in the network. Delay bounds on the example in the previous paragraph are given in Table III. For the quantum distribution  $Q_{C_1} = 270$ ,  $Q_{C_2} = 106$ ,  $Q_{C_3} = 100$  and  $Q_{C_{BE}} = 265$  bytes resulting from Algorithm 1 the maximum end-to-end delays, computed by NC of [7], in critical classes are  $D_{C_1,opt}^{max} = 245.612$ ,  $D_{C_2,opt}^{max} = 390.72$  and  $D_{C_3,opt}^{max} = 483.32$   $\mu\text{sec}$ . They are significantly lower than those computed by the NC approach considered in Algorithm 1.

TABLE III  
EFFECT OF PESSIMISM ON THE RESULTS OF [3]

	Flow	$D_i$	$D_{i,opt}$	Flow	$D_i$	$D_{i,opt}$
$C_1$	$v_1$	224.42	216.49	$v_2$	<b>297.79</b>	<b>245.612</b>
	$v_3$	216.34	208.41	$v_4$	216.42	208.49
	$v_5$	216.42	208.49			
$C_2$	$v_6$	583.18	382.64	$v_7$	583.27	382.72
	$v_8$	583.27	382.72	$v_9$	583.27	382.72
	$v_{10}$	583.27	382.72	$v_{11}$	583.27	382.72
	$v_{12}$	<b>591.35</b>	<b>390.72</b>	$v_{13}$	<b>591.35</b>	<b>390.72</b>
$C_3$	$v_{14}$	<b>892.54</b>	<b>483.32</b>	$v_{15}$	613.72	382.52
	$v_{16}$	613.80	382.6	$v_{17}$	621.80	390.6
	$v_{18}$	<b>892.54</b>	<b>483.32</b>	$v_{19}$	613.72	382.52
	$v_{20}$	<b>892.54</b>	<b>483.32</b>	$v_{21}$	621.80	390.6

$D_i$  : End-to-end delays (in  $\mu\text{sec}$ ) obtained by NC from [5]

$D_{i,opt}$  End-to-end delays (in  $\mu\text{sec}$ ) obtained by optimised NC from [7]

Function *QuantumMin* in Algorithm 1 stops decreasing the quantum  $Q_{C_x}$  for a given critical class  $C_x$  when its computed delay bound (based on NC of [5]) is near (and smaller) to its deadline. Since this delay bound can be pessimistic, there remains a possibility to decrease the quantum share for  $C_x$  (eventually increasing the residual quantum share for  $C_{BE}$ ) while still respecting its deadline.

However, the optimised NC approach cannot be directly integrated in Algorithm 1. As mentioned earlier, the implementation of the function *QuantumMin* in Algorithm 1 is based on the assumption that the WCTT analysis is carried out independently for each class, i.e. in order to compute the end-to-end delay bound in a class  $C_x$ , only  $Q_{C_x}$  (a portion of  $Q$ ) and  $Q$  (the sum of quantum distributed among all the classes) is needed. This assumption does not hold for the optimised NC approach of [7]. In order to compute tight delay bounds, [7] takes into account the impact from competing classes, which is affected by the distribution of  $Q$  in these classes.

Therefore, we propose an extension to Algorithm 1 which takes the quantum distribution result from Algorithm 1 and improves it by deploying the NC approach from [7]. The main idea is to improve the quantum share of the non-critical class by decreasing the quantum of critical classes and/or increasing the quantum of non-critical class until the computed delay bound (based on NC in [7]) in critical classes is near (and smaller) to their respective deadlines.

Before giving the algorithm, the process is illustrated on the example in Figure 1 with flow classification in Table II.

1) *Process overview*: Table IV summarises the steps of the process. It starts with the quantum assignment computed by the algorithm in [3]. Using the optimised WCTT analysis of [7], the maximum end-to-end delays for critical classes are smaller than those which are considered by the algorithm in [3] (as shown in Table III). Therefore there is a margin for each critical class, which is the differences between this optimised end-to-end delay bound and the deadline. These margins are listed in Table IV. For instance, for class  $C_1$ , it is 18.129% (between 245.612  $\mu\text{s}$  and 300  $\mu\text{s}$ ). This initial quantum assignment reserves 35.76% of the bandwidth to non-critical flows.

The first step of the process consists in decreasing quanta associated with critical classes as much as possible, keeping the quantum assigned to non-critical class. It comes to find the smallest quanta for critical classes such that, for each critical class, the end-to-end delay bound computed with the optimised WCTT analysis is not greater than the deadline and the quantum is at least the maximum frame size. For the considered example, quanta for  $C_1$  and  $C_2$  are reduced to 199 and 100 bytes, while quantum for  $C_3$  is not changed. They cannot be further decreased since they correspond to the minimum frame size (for classes  $C_2$  and  $C_3$ ) or it would lead to a delay bound greater than the deadline (class  $C_1$ ). Now the non-critical class gets 39.91% of the bandwidth.

The second step consists in increasing, as much as possible, the quantum assigned to the non-critical class while keeping the quanta assigned to critical ones. This step gives a quantum of 311 bytes which leads to 43.8% of the bandwidth.

At this point, the critical class quanta cannot be decreased. However, margins for class  $C_2$  and  $C_3$  are large (around 29% and 41%). Therefore, in the third step, the quantum of class  $C_1$  (the one with no margin) is increased in order to be able to increase the quantum for non-critical class. It gives 398 bytes for  $C_1$  quantum.

Now, the bandwidth for the non-critical class is decreased to 34.21%. However, since no critical class has a margin close to 0, the second step can be applied again (as a fourth step). It gives 554 bytes for non-critical class quantum, corresponding to 48.09% of bandwidth. These steps (second to forth) gives a gain of around 5% by increasing quanta of classes  $C_1$  and  $C_{BE}$ .

A similar process can be applied again, i.e. increase the quantum of class  $C_2$  (the critical class with the smallest margin) to 125 and then the quantum of  $C_{BE}$  to 624. The new bandwidth for non-critical class is 50.04%.

Since all margins are small, there is no point to apply the same process again. However, the quanta for the critical classes can still be decreased, down to the situation where decreasing them more would lead to a delay bound greater than the deadline or a quantum smaller than the maximum frame size. At the end, the bandwidth for the non-critical flows is 54.97%.

The improvement from the existing approach in [3] is significant (from 35.76% to 54.97%). We argue that the obtained

TABLE IV  
IMPROVED QUANTUM ASSIGNMENT PROCESS

Quantum (bytes)				Margin (%)			BW
$C_1$	$C_2$	$C_3$	$C_{BE}$	$C_1$	$C_2$	$C_3$	$C_{BE}$
Quantum assignment from [3]							
270	106	100	265	18.128	34.88	46.297	35.76%
First step: decrease critical class quanta							
199	100	100	265	6.229	34.88	46.3	39.91%
Second step: increase non-critical class quantum							
199	100	100	311	0.089	29.36	40.98	43.8%
Third step: increase quantum for critical classes with no margin							
398	100	100	311	27.32	29.32	40.95	34.21%
Fourth step: increase non-critical class quantum							
398	100	100	554	7.88	0.113	12.84	48.09%
Fifth step: increase quantum for the critical class with no margin then for the non-critical class							
398	125	100	624	0.95	0.005	4.75	50.04%
Final step: decrease quanta for critical classes							
299	112	100	624	1.54	0.005	4.75	54.07%

quantum assignment should be close to optimum, assuming the WCTT analysis of [7]. Indeed, margins for critical classes are small (less than 5%) and the quantum cannot be smaller for class  $C_3$  (largest margin), since it is the maximum frame size. We recall that it has been shown in [3] that smaller values of sum of quanta  $Q$  lead to a higher bandwidth for  $C_{BE}$ .

2) *Algorithm*: Algorithm 2 implements the process described in the previous paragraph. Every WCTT computation is based on the optimised NC approach of [7].

---

**Algorithm 2:** Improved quantum assignment algorithm

---

**input** : Quantum distribution from Algorithm in [3] :

$Q_{C_1} \dots Q_{C_n}$  (Integer)

**output**: Per class quantum:  $\text{res}\{Q_{C_1} \dots Q_{C_n}\}$  (array)

```

1 while true do
2   for  $i = 1$  to  $(n - 1)$  do // Minimise  $C_i$  service
3     REDUCETOMIN( $Q_{C_i}$ );
4    $\text{res} \leftarrow \{Q_{C_1} \dots Q_{C_n}\}$ ;
5    $\text{previous}Q_{C_n} \leftarrow Q_{C_n}$ ;
6   while true do // Maximise  $C_n$  service
7     INCREASETOMAX( $Q_{C_n}$ );
8     if CHECKSTOPCONDITION( $\epsilon$ ) then
9       break;
10    else
11      INCREMENTSMALLESTDIFF();
12  if  $\text{previous}Q_{C_n} == Q_{C_n}$  then
13    break;
```

---

Three operations have to be executed one or several times.

- REDUCETOMIN( $Q_{C_i}$ ) decreases quantum assigned to critical class  $C_i$  as much as possible, while keeping the quantum of the non-critical class. The implementation is based on a binary search between the current value of  $Q_{C_i}$  and its minimum possible value, i.e. the maximum frame size of  $C_i$  traffic.
- INCREASETOMAX( $Q_{C_n}$ ) increases the quantum associated with the non-critical class as much as possible while

keeping the quanta assigned to critical classes. The value of  $Q_{C_n}$  is doubled as long as it does not lead to missed deadlines for critical classes. As soon as it is the case, a binary search is implemented between the reached value and the previous one.

- INCREMENTSMALLESTDIFFERENCE() increases the quantum assigned to the critical class with the smallest margin (so that the quantum associated with the non-critical class can be increased further). The quantum of the critical class  $C_i$  with minimum margin is doubled. If it does not lead to missed deadlines for other critical classes, the new value is kept as quantum for  $C_i$ . Otherwise, a binary search is implemented between the new value and the initial one.

The algorithm takes the quantum distribution from Algorithm 1, i.e.  $Q_{C_1}, \dots, Q_{C_n}$ , as input.

First, the quantum  $Q_{C_i}$  for each critical class  $C_i$  ( $i \neq n$ ) is reduced so that the corresponding end-to-end delay bound remains less than the delay constraint in  $C_i$  and the quantum is at least the maximum frame size (line 2–3). The resulting quantum values are stored in *res*.

Next, the maximum quantum  $Q_{C_n}$  for non-critical class  $C_n$  is computed (line 6–11). This step is divided in two parts. In the first part, the quantum  $Q_{C_n}$  is increased while the end-to-end delay bounds in all the critical classes remain less than their respective delay constraint in  $C_i$ . In the second part, the quantum of the critical class with the smallest margin is increased.

Both parts are repeated until the margin for each critical class is reduced below a value  $\epsilon$  (line 8–9). The whole process is repeated until  $Q_{C_n}$  reaches a value which cannot be further improved (line 12–13).

#### IV. FLOW DISTRIBUTION

Up to now it has been considered that critical flows have been distributed in classes and the proposed quantum assignment maximises the bandwidth available for non-critical flows with no missed deadlines for critical ones. In this section, we show that the distribution of critical flows in classes has a significant impact on achievable bandwidth for non-critical ones. Finding an optimal distribution by exhaustive search is intractable, even for configurations with limited number of flows (e.g. configuration in Figure 1 with 21 critical flows). Thus we identify flow features which have to be taken into account in order to propose a flow distribution heuristic.

Let's come back to the network configuration in Figure 1 and consider the three following flow distributions for critical flows ( $v_1$  to  $v_{21}$ ):

FD<sub>1</sub>: a single critical class  $C_1$  with flows  $v_1 \dots v_{21}$ ,

FD<sub>2</sub>: two critical classes  $C_1$  and  $C_2$  with  $v_1, v_6, v_{12}, v_{13}$  in  $C_1$  and the other critical flows in  $C_2$ ,

FD<sub>3</sub>: three critical classes  $C_1, C_2$  and  $C_3$  with  $v_1, v_2, v_3, v_8, v_9$  in  $C_1, v_4, v_5, v_7, v_{10}, v_{11}, v_{12}, v_{15}, v_{16}$  in  $C_2$  and the other critical flows in  $C_3$ .

In each flow distribution case, our improved quantum assignment algorithm is applied and the corresponding results are

shown in Table V. These results show that distributed critical

TABLE V  
FLOW DISTRIBUTION IN CRITICAL CLASSES AND THE RESULTS OF  
QUANTUM ASSIGNMENT ALGORITHM

		Flows	DL	$Q_{C_i}$	$Q_{C_{BE}}$
FD <sub>1</sub>	C <sub>1</sub>	$v_1 - v_{21}$	300	219 (67.8%)	104 (32.2%)
FD <sub>2</sub>	C <sub>1</sub>	$v_1, v_6, v_{12}, v_{13}$	300	100 (12.84%)	231 (29.65%)
	C <sub>2</sub>	$v_2-v_5, v_7-v_{11}, v_{14}-v_{21}$	310	448 (57.51%)	
FD <sub>3</sub>	C <sub>1</sub>	$v_1, v_2, v_3, v_8, v_9$	300	199 (24.12%)	376 (45.58%)
	C <sub>2</sub>	$v_4, v_5, v_7, v_{10}, v_{11}, v_{12}, v_{15}, v_{16}$	390	150 (18.18%)	
	C <sub>3</sub>	$v_6, v_{13}, v_{14}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}$	600	100 (12.12%)	
FD <sub>4</sub>	C <sub>1</sub>	$v_1 - v_4$	300	249 (23.29%)	608 (56.88%)
	C <sub>2</sub>	$v_5 - v_{12}$	590	112 (10.48%)	
	C <sub>3</sub>	$v_{13} - v_{21}$	890	100 (9.35%)	

Units: Deadline (DL) in  $\mu\text{sec}$ . Quantum distribution in *bytes*

flows in different classes is not always beneficial. Indeed,  $FD_1$  (one critical class) leads to 32.2% of bandwidth for non-critical flows and is better than  $FD_2$  (two critical classes, 29.65% of bandwidth for non-critical flows), but worse than  $FD_3$  (three critical classes, 45.58% of bandwidth for non-critical flows). Actually, two main factors impact the results:

- the difference between flow deadlines within each class: the overall deadline of a class (input of quantum assignment) is the smallest deadline among all the flows in the class. It comes to consider that all the flows in the class have to respect this smallest deadline. As soon as some flows in the class have a much higher deadline, the result is to significantly increase the required bandwidth for the class, leading to a reduction of the bandwidth available to non-critical flows. Thus, the difference between flow deadlines within a class should be as small as possible.
- the number of classes: increasing the number of classes increases the scheduler latency computed by NC approach, since this computation considers the worst case for the deficit of competing classes (quantum plus maximum frame size minus one byte). Moreover, the residual bandwidth that can be provided to any class decreases since the total bandwidth has to be shared among more classes in each DRR scheduling round.

These impacts can be illustrated on the results in Table V. Class  $C_1$  in  $FD_1$  includes flows with very different deadlines (from 300  $\mu\text{s}$  to 950  $\mu\text{s}$ ). Deadline variations are similar for  $FD_2$  (from 300  $\mu\text{s}$  to 890  $\mu\text{s}$  for  $C_1$ , from 310  $\mu\text{s}$  to 950  $\mu\text{s}$  for  $C_2$ ). Therefore  $FD_2$  is mainly impacted by the increased scheduler latency induced by the additional critical class, leading to a reduced bandwidth for non-critical flows. Conversely, the deadline variation within classes is smaller for  $FD_3$  (from 300  $\mu\text{s}$  to 620  $\mu\text{s}$  for  $C_1$ , from 390  $\mu\text{s}$  to 910  $\mu\text{s}$  for  $C_2$ , from 600  $\mu\text{s}$  to 950  $\mu\text{s}$  for  $C_3$ ) and the benefit more

than compensate for the increased scheduler latency induced by the additional critical classes. In general, the impact of deadline variation within classes is much higher than the cost of scheduler latency. This is the case for  $FD_3$  and the same trend has been observed in all conducted experiments.

Based on this observation, we propose the following heuristic to efficiently distribute these flows in different classes so that their bandwidth requirement is reduced and the residual bandwidth for non-critical class is increased. The principle is to group the flows with similar delay constraints in the same class so that they can be assigned a quantum which allows them to be served in a delay not far from their deadlines.

- Critical flows are sorted by increasing deadlines,
- We assume that the maximal number of classes is known. This is a reasonable assumption, since each output port of the QoS aware AFDX switch implementing DRR will have a fixed number of queues, thus classes.
- Based on the number  $n - 1$  of critical classes, the  $n - 2$  most significant increments in the flow deadlines are identified as the boundaries of the classes.

The process is implemented by Algorithm 3.

---

#### Algorithm 3: Flow distribution

---

**input** : Maximum number of classes :  $n$   
**output**: List of critical class boundaries :  $res$   
**Data**: Critical Flows :  $VL$  (list), Quantum distribution  $QL$  (list), Critical class boundaries :  $BList$  (list)

```

1 SORTFLOWS(VL);
2 BList ← GETSIGNIFICANTINCREMENTS(VL);
3 previousQL ← empty;
4 i ← 0;
5 while size(res) < n-2 and i < size(BList) do
6   i ← i + 1;
7   res.append(BList[i]);
8   DISTRIBUTEFLOWS(res, VL);
9   QL ← COMPUTEQUANTUMDISTRIBUTION();
10  if ISIMPROVED(QL, previousQL) then
11    | previousQL ← QL
12  else
13    | res.remove(BList[i]);
```

---

Let us illustrate this algorithm on the network example in Figure 1, assuming four classes ( $n = 4$ ). Critical flows ( $v_1$  to  $v_{21}$ ) are sorted by increasing deadlines (actually the order in Table I). The  $n - 2$  most significant increment in deadlines (i.e. potential class boundaries) are determined by decreasing values and corresponding potential class boundaries are stored in  $BList$  (line 2): in the example, between  $v_{12}$  and  $v_{13}$  ( $890 - 660 = 230 \mu\text{s}$ ), then between  $v_4$  and  $v_5$  ( $590 - 390 = 200 \mu\text{s}$ ). Then each potential boundary is considered (line 5–13). In the first iteration, the boundary between  $v_{12}$  and  $v_{13}$  is considered. It distributes flows in two classes:  $C_1$  gets  $v_1$  to  $v_{12}$  and  $C_2$  gets  $v_{13}$  to  $v_{21}$  (line 8). Quanta for the classes are computed by our improved quantum assignment algorithm (line 9). It gives



a first solution with two critical classes. In the next iteration, the same process is applied with the boundary between  $v_4$  and  $v_5$  leading to three classes ( $C_1$  gets  $v_1$  to  $v_4$ ,  $C_2$  gets  $v_5$  to  $v_{12}$  and  $C_3$  gets  $v_{13}$  to  $v_{21}$ ). It leads to a second solution with three critical classes, which, in the example, gives more bandwidth for non-critical flows than the first solution. Therefore, this second solution is kept (line 10–13). The process stops, since we assume no more than three critical classes in our example.

The resulting flow distribution  $FD_4$  is shown in Table V. It clearly outperforms previous flow distributions  $FD_1$ ,  $FD_2$  and  $FD_3$  with more than 56% of bandwidth for non-critical flows. For instance we have 32% of this bandwidth when there is a single critical class ( $FD_1$ ).

## V. EVALUATION

In this section, the two solutions proposed above are evaluated on an Airbus A380 aircraft "like" industrial network configuration. The gain introduced by Algorithm 2 is also compared to the existing solution of [3].

The given network configuration includes 96 end-systems interconnected by 8 switches forwarding 984 flows on 6276 paths (multi-cast VL). Frame lengths are between 84 and 1535 bytes and BAGs are between 2 and 128 ms.

These flows are distributed in 3 critical classes ( $C_1$ – $C_3$ ), based on the solution proposed in Section IV, as shown in Table VI. Ten additional flows are arbitrarily introduced, characterised by non-critical class  $C_{BE}$ , which share the path with each critical flow on at least one switch output port.

TABLE VI  
FLOW DISTRIBUTION AND QUANTUM ASSIGNMENT

	$C_1$	$C_2$	$C_3$	$C_{BE}$
Flows (paths)	280 (1681)	229 (1488)	475 (3107)	10 (501)
BAG	2–16	32	64–128	-
Frame lengths	84–1497	84–1371	84–1535	84–934
Deadline	13000	39000	52000	-
Quantum distribution (in bytes)				
Initial	6057	1640	1535	1012 (9.87%)
Algo 2	8130	1943	1535	5190 (30.89%)

Units: BAG in *msec*. Frame lengths in *bytes*. Deadline in  $\mu$ sec.

The quanta computed by Algorithm 2 as well as the algorithm in [3]) are shown in Table VI, where *Initial* corresponds to [3]. These quanta are then used to compute the worst-case end-to-end delays based on optimised NC approach of [7]. For comparison, the results are plotted in increasing delay order (per class) in Figure 4. It can be observed from Figure 4 that the improved quanta (by Algorithm 2) bring the delay bounds in all the critical classes closer to their respective deadlines, which allows the residual quantum share for non-critical class to increase to 30.89% as compared to the initial value of 9.87% (Table VI).

## VI. CONCLUSION

In this paper, a QoS-aware AFDX network is considered where each output port is managed by a DRR scheduler. This

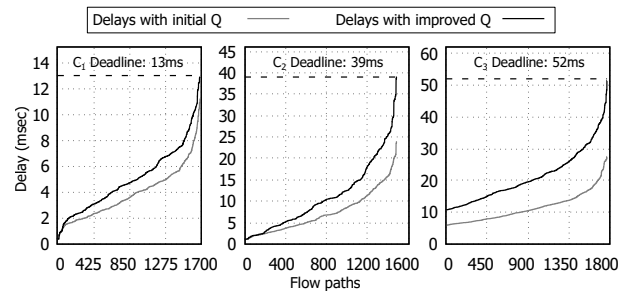


Fig. 4. Results on industrial AFDX configuration

network is shared between critical (avionic) and non-critical (additional) flows. We address the problem of distributing flows among traffic classes ( $n-1$  critical classes, 1 non-critical one) and compute the quanta which ensure the delays less than the deadlines in critical flows and maximise the bandwidth for the non-critical flow. The approach of [3] is improved in two ways. First, the quantum assignment algorithm is redesigned to exploit the optimised WCTT analysis of [7] ([3] uses the basic WCTT analysis of [5], since it makes assumptions which do not hold with the WCTT analysis of [7]). Second, a heuristic is proposed for the distribution of critical flows in classes.

The solution presented in this paper outperforms the one in [3]. On a realistic case study, the bandwidth for non-critical flows guaranteed by [3] is less than 10% and it is more than 30% with the new approach.

For future work, other policies are envisioned for AFDX, e.g. WRR [11], that also requires a configuration solution.

## REFERENCES

- [1] "Aircraft Data Network, Parts 1,2,7 Aeronotical Radio Inc." ARINC Specification 664, Tech. Rep., 2002 - 2005.
- [2] T. Hamza, J.-L. Scharbag, and C. Fraboul, "Priority assignment on an avionics switched Ethernet Network (QoS AFDX)," in *IEEE International Workshop on Factory Communication Systems (WFCS)*, 05 2014.
- [3] A. Soni, J.-L. Scharbag, and J. Ermont, "Quantum assignment for QoS-aware AFDX network with deficit round robin," in *27th International Conference on Real-Time Networks and Systems (RTNS)*, 2019, p. 70–79.
- [4] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," in *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, 1996, pp. 375–385.
- [5] M. Boyer, G. Stea, and W. M. Sofack, "Deficit Round Robin with Network Calculus," in *IEEE 6th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, 10 2012, p. 10.
- [6] S. S. Kanhere and H. Sethu, "On the latency bound of deficit round robin," in *11th International Conference on Computer Communications and Networks (ICCCN)*, 10 2002.
- [7] A. Soni, X. Li, J.-L. Scharbag, and C. Fraboul, "Optimizing Network Calculus for Switched Ethernet Network with Deficit Round Robin," in *IEEE Real-Time Systems Symposium (RTSS)*, 12 2018, pp. 300–311.
- [8] S. Noda and K. Yamaoka, "Approach to optimal wrw weight assignment method in delay-limited environment," in *13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 01 2016.
- [9] M. Katevenis, S. Sidiropoulos, and C. A. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," in *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, 10 1991, pp. 1265–1279.
- [10] J.-Y. L. Boudec and P. Thiran, *Network Calculus: a theory of deterministic queueing systems for the internet (Book)*. LNCS, 04 2012.
- [11] A. Soni, X. Li, J.-L. Scharbag, and C. Fraboul, "WCTT analysis of avionics switched Ethernet network with WRR scheduling," in *26th International Conference on Real-Time Networks and Systems (RTNS)*, 2018.