Performance analysis of the Disrupted Static Priority scheduling for AFDX

Rubén Trillo Flores, Marc Boyer

ONERA – The French Aerospace Lab
F 31055 , Toulouse, France

***Abstract*—The AFDX technology is used as backbone in several aircraft. It offers a high bandwidth (commonly 100Mb/s), and guaranteed per data flow a bound on the network traversal delay, while being a shared resource. To do so, it uses a segregation mechanism, the Virtual Link (VL), designed as a bus abstraction.**

Nevertheless, there may exist applications with such stringent latency requirements that a given AFDX topology can not ensure it. To provide low latency, the AFDX technology provides priority levels: but even in the highest priority level, the latency bound can still be too high with regard to applicative requirements. One cause of latency is the non-preemptive aspect of priority policy: when a high priority frame reaches an output port, it may have to wait the emission of a low priority frame. Then, a high priority frame might have to face a delay equal to the maximum packet emission time of the low priority flows per crossed switch.

The scheduling policy called Disrupted Static Priority (D-SP) aims to improve the latency guarantees, by interrupting low priority frames, and re-emitting them from the beginning. This policy is a trade-off between simplicity and efficiency.

A quantitative comparison between the NP-SP and the D-SP scheduling systems' performance on a realistic case-study is provided in this article.

I. Introduction

Avionics Full DupleX (AFDX) [1] is a high speed Commercial Ethernet with provisions for guaranteed Deterministic Timing and Redundancy required for Avionics applications. It is built upon protocol specifications of IEEE 802.3 and ARINC 664 P7.

Thanks to the use of Full Duplex Ethernet, the possibility of transmission collisions is eliminated. Also, the network is designed in such a way that all critical traffic is prioritized using Quality of Service (QoS) policies. Therefore, delivery, latency, and jitter are all guaranteed to be within set parameters.

On an AFDX aircraft data network we can find 3 different types of network elements, the End systems (E/S), the Switches and the Virtual Links (VL). An E/S is a device whose applications access the network components in order to send or receive data from the network. A Switch is an equipment that performs traffic policing and filtering, and forwards packets towards their destination E/S. And a VL defines a unidirectional (logical) connection from one source E/S to one or more destination E/S (Uni- or Multicast communication).

In the current AFDX standard, the scheduling policy uses a hierarchical scheme with first a Non-Preemptive Static Priority scheduling (NP-SP), and second a First In First Out (FIFO) withing each priority level. To each virtual link is associated a priority, and the scheduler separates the incoming flow in different queues according to their priority. Every queue uses the First In First Out (FIFO) organizing method. The low priority queue is served only if the high priority one is empty. When a frame of the high priority is set in its queue, if a frame of the low priority flow is being sent, it must wait until the end of this emission before being transmitted (this is a "non preemptive" scheduling).

This non-preemption can be the cause of some non negligible delays: on 100Mb/s port, a low priority frame of 1500 bytes can create a latency of 120 μs. A high priority frame crossing 4 switches can experience a delay of $480\mu s \approx 0.5ms$, which can be too high for some applications.

The scheduling policy called Disrupted Static Priority (D-SP) aims to improve the latency guarantees, by interrupting low priority frames, and re-emitting it from scratch. This policy is a trade-off between simplicity and efficiency.

The objective of this paper is to introduce and analyse this variation of the SP called Disrupted Static Priority (D-SP). This scheduling system intents to provide smaller latencies for the higher priority flows and a low complexity level.

In the section II of this paper, the D-SP scheduling system is described and compared to other similar solutions. A short introduction to network calculus is given in Section III. The section IV shows how to model the D-SP policy within the network calculus framework. An experimental analysis of the D-SP applied on a realistic AFDX case-study is carried out and it is compared to the obtained results for the NP-SP scheduling, in Section V. And finally, section VI summarizes the contribution of the paper and presents some directions for future works.

II. Static priority policy flavors

The blocking problem arises when a high priority has to be sent while a lower priority frame is being sent, as illustrated on the top of Figure 1.

In non-preemptive scheduling, the high priority frame has to wait until the end of the low priority frame (case NP-SP in Fig. 1). This non-preemption in the current AFDX policy can be the cause of some non negligible delays: on 100Mb/s port, a low priority frame of 1500 bytes can create a latency of 120 μs. A high priority frame crossing 4 switches can experience

978-1-4799-5338-7/14/$31.00 ©2014 IEEE

a delay of $480\mu s \approx 0.5ms$, which can be too high for some applications.

In CPU scheduling, static priority is often implemented in a preemptive way: when a high priority task is released, it gets the CPU, after some (relatively small) delay related to the context switch. The low priority task is then stopped and resumed, and the execution restarts from the stopping point (case P-SP in Fig. 1). Some context saving/switching/restoring costs may exist, but the task does not restart from scratch.

In network scheduling, frame interruption is not so simple: there is no common context shared by the sender and the receiver to save: solutions may exist, but it significantly increases the complexity of the MAC layer.

Here it is proposed another approach, called Disrupted Static Priority, that just cancels the low priority frame emission, and restarts it from the beginning later (case D-SP in Fig. 1).

A. Frame-preemptive Static Priority

Even if the frame preemption is not so simple as the task preemption, some solutions are under development. The problems to solve are quite common in network: how to delimit, fragment and reassemble PDU.

In the US patent [2], it is proposed to introduce a "preemption delimiter" to allow the receiver to detect such an interruption [2, Fig. 6A-6B] and to resume the preempted frame. And also to add fragment header and check code to allow reassembling by the receiver [2, Fig. 7A]. No encoding neither size of such elements are given.

The IEEE Time-sensitive networking (TSN) task group continues the work of the AVB working group. One of they proposal is the use of preemption of frames. Nevertheless, the current draft [3] is not a public document, and only some presentations are available, like [4]. An implementation have been proposed in [5] using an 8b/10b physical encoder, and some simulations have been also done.

B. Disrupted Static Priority

The Disrupted Static Priority (D-SP) scheduling policy allows to classify the input flow into several different priorities in order to cope with flows of different latency constraints. Thus, a more diligent service will be provided to the higher priority flows.

Each queue, corresponding to a certain priority, uses the First In First Out (FIFO) organizing method. That is, the packets will be scheduled in the queue in the same order as they arrive. Then, the oldest packet will be the first one to be served and so on.

High priority (HP) packets will be transmitted without interruption as long as there are packets in the HP queue. Therefore, only when there are no packets in the HP queue, the Low Priority (LP) queue will be able to transmit its packets. However, if a HP packet arrives when a LP packet is being processed, the latter will be stopped to make way for the HP packet. Then, contrary to a Preemptive scheduling, the amount of data from the LP packet that was processed until the time of interruption will be discarded. Hence, the processing of that

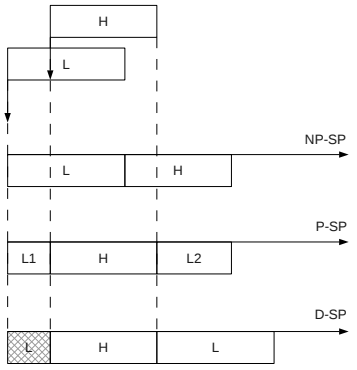

Fig. 1. Graphic example of the NP-SP, P-SP and D-SP packet schedulers

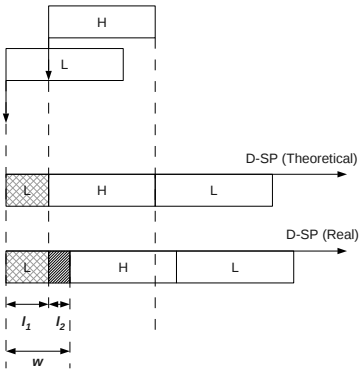

Fig. 2. Real behaviour of the D-SP packet scheduler

LP packet will restart from the beginning when there are no more packets in the HP queue.

An example of the D-SP scheduling is illustrated in figure 1.

In the previous example, we see that by using the NP-SP scheduling, the fact of having to wait for the LP packet to be fully transmitted induces a significant delay in the transmission of the HP packet. This delay is completely erased by using the D-SP scheduling (and the P-SP as well). However, the D-SP introduces a delay for the possible future incoming packets equal to the size of the LP packet fraction that was already processed and discarded afterwards.

So, by comparing with the P-SP, at the expense of decreasing the algorithm complexity, the D-SP introduces a certain delay each time there is a disruption.

Besides, note that, in reality, when the HP packet arrives to the queue, it takes some time to cease the processing of the LP packet and get the HP packet into the server. This time will be of course implementation dependant (explicit end of frame, on purpose carrier corruption, etc.). It is represented in figure 2 as l_2.

The time span l_1 corresponds to the part of the LP packet that has been processed but that is discarded. It is called the *discard*. On the other hand, l_2 is called the *transition*. And the sum of both segments is called the *waste*, denoted as w.

C. Discussion

The D-SP policy is designed as a trade-off between efficiency and simplicity, to avoid the well known blocking factor of non preemption at frame level.

There are other solutions: one is to add mechanisms to allow preemption, like in task scheduling, and another one is to avoid contention, by the use of time-triggered scheduling.

The D-SP policy can be very simply implemented: a frame must be kept in memory up to the end of transmission in case of disruption, and some mechanism to explicitly end a transmission must be added, like sending the common Ethernet delimiter (the 7-bytes preamble plus the 1 byte start of frame). It is very simple to add such basic mechanism in a MAC layer.

Nevertheless, it is not a scalable approach: each frame disruption wastes some bandwidth. The benefit for the high priority flow has a negative impact on the low priority one. The proposal is then to consider three priority levels: a very-high priority, whose frames will preempt lower priority frames, and the current high and low priority levels, without preemption.

Another solution is to use preemption, as described in II-A. This solution requires a more significant change to the hardware, and also the addition of some overhead for frame reassembling (delimiter, fragment headers, etc.). This overhead will certainly be really smaller by re-emitting the complete frame, as done in D-SP. If more than one level of preemption is allowed[1], the solution would become quite complex. There is not, up to our knowledge, a complete standard solution, so, a deeper comparison can not be done.

But one may also want to avoid the blocking problem by the use of more complex mechanism to ensure low latency, like the integration of Time Triggered (TT) flow within AFDX-like solution in the TTEthernet standard [6], [7]. In a time triggered approach, each frame has some reserved time slots (often assigned on a periodic way) on each link. If the global schedule between links is correctly done, the waiting time in the switch is null: for a frame entering a switch on a link l and forwarding to a link l', for each time slot on link l, it exists a time slot on l' such that the waiting time is reduced to the switching delay. Such Time Triggered solution ensures very low network latency.

Nevertheless, it requires a tight synchronisation between the applications producing the data and the network. Otherwise, if the data production is independent of the network schedule, a data could be computed just after a dedicated network time slot and would have to wait to the next slot. In a periodic system, it means that the delay from application to application can be larger than a period. This situation is illustrated in figure 3: a data flow must be sent on one link, and forwarded to a second one. Time slots are allocated such that the waiting time in the switch is quite null. Consider the data "A", produced just before its time slot: its application to application latency is somehow equal to the network latency, reduced to twice the emission latency (frame size divided by the bandwidth). But the data "B" is produced independently on the network schedule. Then, it has to wait to the next time slot,

[1]That is to say, a frame of high priority may preempt a low priority frame, and be itself preempted by a very high priority frame.

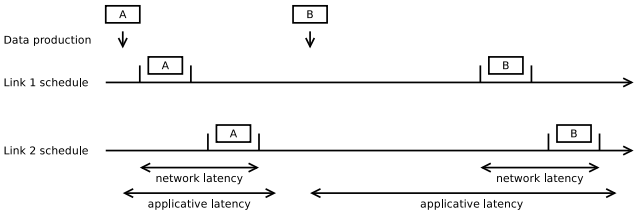

Fig. 3. TTEthernet scheduling example

and even if the network latency is quite small, the application to application latency can be larger than a period.

On the opposite, with an asynchronous solution, the network delay is bigger than in the time triggered solution, but the data is put in sending queue as soon as it is produced. If the network delay is smaller than the period, the asynchronous solution gives a lower application to application delay.

Moreover, TTEthernet also suffers from the non-preemptive aspect of Ethernet. When mixing rate-constrained (RC) traffic (AFDX-like) and TT traffic, it may happen that a RC frame is being sent when a TT slot begins. To face it, three solutions are proposed in [7]: "preemption", where the low priority frame is discarded and relayed later (as in D-SP), "timely block" where no low priority frame can be sent just before a TT slot, and "shuffling" that just shifts the TT frame just after the blocking one. The shuffling suffers from the same drawback than non-preemptive SP: on each link, a frame can encounter a blocking of a low priority frame. The two other solutions lead to bandwidth waste, like the D-SP proposal.

The D-SP solution is quiet simple, and since it "wastes" some bandwidth, it is not a scalable approach. But it could offer a reasonable trade-off to integrate some flows enquiring very stringent latencies on AFDX. Our bet is that there are not so many such flows. Currently, such flows can not use the AFDX backbone: they have to use a dedicated bus.

The trade-off is then between: using a separated bus, switching to a very different solution, like TTEthernet, or making minor modifications to AFDX. The pros and cons of each solution can be listed, but when comparing the different options for a given aircraft, several parameters must be evaluated: costs, weight, consumption, maintenance, etc.

This paper is just about the evaluation of delays, making only small update to the AFDX standard.

III. Network calculus

As [8], [9] describes, **Network Calculus** is a network analysing method which aims to compute network memory and delay bounds that characterize the worst-case scenario.

Notations: Let $\mathbb{R}$ denotes the set of real numbers, and $\mathbb{R}_{\geq 0}$ the subset of non-negative real numbers. Let be $[x]^+ = \max(0, x)$, and $\lceil x \rceil$ the ceiling function. The non-decreasing closure $f_\uparrow$ is defined as $f_\uparrow(t) = \sup_{0 \leq x \leq t} f(x)$, and $[f]^+_\uparrow = ([f]^+)_\uparrow$.

A node S is modelled by an arrival/input $A(t)$ and a departure/output $D(t)$ data flow, cumulative functions which are the number of bits seen on the flow up to time t with

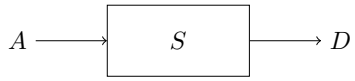

Fig. 4. Server with arrival and departure flows

the convention that $A(0) = D(0) = 0$ and that A, D are left continuous[2]. Since the output can be done only after the input, it is required that $D \leq A$.

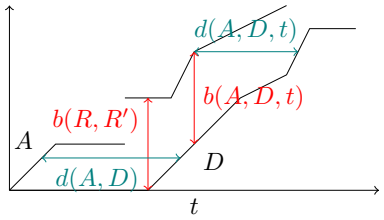

Fig. 5. Arrival and departure cumulative curves

Concretely, the network boundaries that Network Calculus computes are:

- **Backlog** - it represents the maximum number of bits that are inside the node at the time t. When considering the whole flow, the maximum on any instant must be taken into account.

$$b(A, D, t) = \{A(t) - D(t)\} \quad (1)$$
$$b(A, D) = \sup_{t \geq 0}\{A(t) - D(t)\} \quad (2)$$

- **Virtual Delay** - it represents the maximal period of time that the output data flow needs to reach the same value of the input data flow.

$$d(A, D, t) = \inf\{\tau \geq 0 | A(t) \leq D(t + \tau)\} \quad (3)$$
$$d(A, D) = \sup_{t \geq 0}\{d(A, D, t)\} \quad (4)$$

Before presenting the notions of contracts on flows and servers, a specific operator must be presented, the (min,plus) convolution.

- **Convolution/deconvolution** Let be $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}$ two functions. Their convolution and deconvolution are defined as

$$(f * g)(t) = \inf_{0 \leq s \leq t}\{f(t - s) + g(s)\}$$
$$(f \oslash g)(t) = \sup_{0 \leq s}\{f(t + s) - g(s)\}$$

Owing to the fact that the exact input/output data flows are in general unknown at design time, or can have a stochastic nature, the calculus of these bounds cannot be obtained. Therefore, Network Calculus characterizes the evolution of these flows using deterministic curves:

- **Arrival curve** $\alpha(t)$ - it characterizes the maximum number of bits that arrive at the node during a period of time Δt. A flow A has an arrival curve α iff

$$\forall t, \Delta t \geq 0 : A(t + \Delta t) - A(t) \leq \alpha(\Delta t) \quad (5)$$

[2]For a discussion on continuity in network calculus, see [10].

This relation is equivalent with $A \leq A * \alpha$.

The contract on flow can be seen either from an engineering point of view (bound on any interval) or from a mathematician point of view (convolution-based relation). For the service definition, the two points of view exist, but lead to slightly different definition.

- **Strict minimal service** - it characterizes the minimum number of bits that are served during a backlogged period of time Δt. A backlogged period is an interval where $D(t) < A(t)$, *i.e.* it exists some backlog in the server. Then, a server offers a minimal strict service of curve β iff for all input/output A, D and

$$\forall t, \Delta t \geq 0 : (\forall x \in (t, t + \Delta t], D(x) < A(x) \implies D(t + \Delta t) - D(t) \geq \beta(\Delta t)) \quad (6)$$

- **Min-plus minimal service** - it also characterizes the service, but based on the convolution. A server offers a min-plus minimal service of curve β iff for all input/output A, D

$$D \geq A * \beta \quad (7)$$

These two notions of service are of interests and are related: for example, a server offering a minimal strict service β also offers the minimal min-plus service β. A complete overview can be found in [11].

The basic results of network calculus allows, considering contracts, to compute bounds and to propagate delays. Let S be a server offering a min-plus minimal service β, a min-plus maximal service β^M and a shaping curve σ. If the input flow A has arrival curve α, then, the delay and memory can be bounded, and an arrival curve α' of the output flow D can be computed:

$$d(R, R') \leq d(\alpha, \beta) \qquad b(R, R') \leq b(\alpha, \beta)$$
$$\alpha' = \min(\sigma, (\alpha * \beta^M) \oslash \beta)$$

IV. Modelling Disrupted Static Priority

To evaluate the impact of a scheduling policy, simulation is often used. Whereas simulation is a good strategy to evaluate mean behaviour, it does not provide any guaranteed information on the worst case, especially when this worst case is a rare event [12]. In embedded critical systems, guaranteed bounds have to be provided. The network calculus is one method computing such bounds.

In this section, the D-SP policy is modelled within the network calculus theory. The general case is considered in section IV-A, for completeness, however, in the AFDX context, a simpler and tighter analysis can be done, presented in section IV-B.

A. General case

Theorem 1 (Residual services with D-SP): Let S be a server which offers a strict minimal service β, shared by 2 flows (High Priority and Low Priority), with respective arrival

curves α_H and α_L, and scheduling flows with a D-SP policy, then the corresponding residual services are:

$$\beta_H = \beta - l_2 \tag{8}$$

$$\beta_L = \beta - (l_L^{\max} + l_2) \left\lceil \frac{\alpha_H}{l_H^{\min}} \right\rceil \tag{9}$$

$$\geq \beta - \left(\frac{l_H^{\min} + l_L^{\max} + l_2}{l_H^{\min}} \right) \alpha_H \tag{10}$$

Where l_2 is the data size corresponding to the stopping time of the Low priority packet processing, i.e the *transition*; $l_H^{\min}$ is the minimum packet size of the High Priority flow and $l_L^{\max}$ is the maximum packet size of the Low Priority flow.

Sketch of Proof: By looking at figure 2 we can state that for the HP flow, the behaviour of the D-SP scheduling is equivalent to that of a non pre-emptive scheduling where l_2 would be the LP packet. Therefore, by using the expression of the residual service for a non pre-emptive scheduling [11], we already can affirm:

$$\beta_H = [\beta - l_2]^+_\uparrow$$

Note that the size of the *transition* is subtracted from the minimal service of the server and always ensuring that we take non-negative and non-decreasing expressions.

With respect to the LP flow, the result is not so direct. First, after a data flow analysis of the system, we can state that the residual service for the LP flow has the following expression:

$$\beta_L = [\beta - \alpha_H - \alpha_w]^+_\uparrow$$

It seems a logical result, as by looking at figure 2 we can see that it is like the LP flow is competing against two HP flows; the actual HP packet and the disrupted LP packet (also called *waste* packet).

The arrival curve of the HP flow is a given parameter but the arrival curve of the *waste* flow is unknown. As we know that each HP packet will generate as most one *waste* packet, we finally can get the following expression, valid for any interval time Δ:

$$\alpha_w(\Delta) \leq (l_L^{\max} + l_2) \left\lceil \frac{\alpha_H(\Delta)}{l_H^{\min}} \right\rceil$$

So, lastly by substituting in the previous expression, the residual service of the LP flow is obtained:

$$\beta_L = \beta - (l_L^{\max} + l_2) \left\lceil \frac{\alpha_H}{l_H^{\min}} \right\rceil$$

See more details of the *Proof* in the *Appendix A*.

TABLE I. CHARACTERISTICS OF THE AFDX CASE-STUDY

Entities	Number
End systems	104
Routers	8
Virtual Links	920
Latency constraints	5700

TABLE II. CHARACTERISTICS OF THE TRAFFIC FLOW

	Minimum	Average	Maximum
# VL destinations	1	6.2	83
BAG	2 ms	87.3 ms	128 ms
Maximal packet size	100 bytes	550.5 bytes	1500 bytes
# Traversed Routers	1	1.67	4
Latency Constraints	2 ms	10.87 ms	30 ms

B. AFDX case

In the D-SP policy, each high priority frame can disrupt a lower priority one. To evaluate the number of high priority frames, one can divide the amount of data by the minimal frame size, leading to the expression $\left\lceil \frac{\alpha_H}{l_H^{\min}} \right\rceil$. Nevertheless, in the AFDX context, a better evaluation of the frame number can be done.

From the point of view of the Low Priority flow, it can be assumed that the *waste* packet in the D-SP policy corresponds also to a higher priority flow because it comes always attached to the High Priority packet, preceding it. Moreover, it can be considered that the *waste* packet is part of the HP packet. Thus, in order to get the worst-case timing performance for the Low Priority flow, we simply have to increase the maximum size of the HP packet size by the maximum size of the *waste* packet.

Therefore, the current $L^{\max}$ of the VLs for the D-SP High Priority will be increased by the maximum length of the *waste* packet, which is equal to the maximum size of the LP flow plus the *transition* size. Consequently, $L^{\max} \leftarrow L^{\max} + (L_{LP}^{\max} + l_2)$.

V. EXPERIMENTS ON THE USE OF D-SP IN AVIONICS

A. Experimental setup

In order to analyse the performance of the D-SP scheduling, a realistic industrial size configuration is used. Table I summarizes the main characteristics of the AFDX network under study. This configuration is distributed in the RTaW-PEGASE tool as "AFDX_Big" sample.

Each VL has on average 6 destination end systems, that is why we have so many latency constraints. In the following table II the characteristics of the traffic flows are detailed.

This AFDX configuration has four different priority flows (this is not conform with the AFDX standard). Therefore, the network is modified with respect to the standard so that it contains the same amount of HP and LP Virtual Links. Priority is uniformly randomly distributed.

B. RTaW-PEGASE: timing analysis for AFDX networks

RTaW-PEGASE [13] computes tight upper bounds on communication delays and buffer utilization, and provides optimization algorithms that ensure correctness and efficiency. It relies on the Network Calculus formalism.

TABLE III. PRELIMINARY RESULTS

	VHP	HP	LP
# VLs	46	414	460
Total Delay D-SP (ms)	140.63	6582.43	13197.60
Total Delay NP-SP (ms)	170.21	5421.06	12060.22
Total throughput (Mb/s)	37.43	733.02	742.16

This tool will allow us to compute the delays produced in the AFDX case-study for both packet schedulers, the D-SP and the NP-SP.

C. Performance of D-SP scheduling: 10% of very high priority flows

Our assumption is that it exists a small amount of flows requiring very stringent latencies. Then, a (small) subset of "very high priority" flows will be extracted from the HP class.

To evaluate the impact of the D-SP policy, the configuration will be analyzed with two different policies: one with a "classical" static priority analysis and another one with the D-SP scheduling, where only the VHP flows can disrupt other flows, and classical SP arbitration between HP and LP flows.

Notice that the "classical" SP analysis is not standard, since it uses 3 levels of priority whereas the AFDX standard requires only 2 [1]. Nevertheless, this is a quite small improvement.

The *transition* delay, l_2, is of course implementation dependent. In this experiment, it is set to 20 bytes, which is a quite big value (the Ethernet CRC is 4 bytes large, the inter-frame GAP is 12 bytes large and the preamble+start of frame is 8 bytes large). In [4], N. Finn make the assumption "that preempting a frame adds only an extra of 20 bytes; this is the minimum practical penalty". If 20 bytes are sufficient for preemption, they are also sufficient to disruption.

We will take the configuration described in section V-A, but a 10% of the High Priority flow will become the Very High Priority (VHP) flow or the D-SP High Priority flow. Therefore, the LP flow will cover the 50% of the total flow, the HP flow the 40% and the VHP flow the 10%.

As stated in eq. 8 from theorem 1, by looking to the D-SP High Priority flow, the behaviour of the D-SP scheduling is equivalent to that of a non preemptive scheduling where l_2 would be the lower priority packet. So, first the $L^{\max}$ of both the High and Low Priority VLs will be set to l_2. Thus, we will get the delay for the D-SP High Priority flow.

Then, by looking to the lower priority flows, for the specific case of AFDX (cf. section IV-B) we can state that they are competing against the D-SP High Priority flow and also against the *waste* packet. So, in second place the current $L^{\max}$ of the D-SP High Priority VLs will be increased by the maximum length of the *waste* packet, which is equal to the maximum size of the LP or HP flow plus the *transition* length. Therefore, $L^{\max} \leftarrow L^{\max} + (L^{\max}_{(LP/HP)} + l_2)$. Thus, we will get the delay for the High and Low Priority flows.

Finally both solutions will be merged in order to get the resulting delay for the D-SP packet scheduler.

By summing up, table III shows the preliminary results for each priority class.

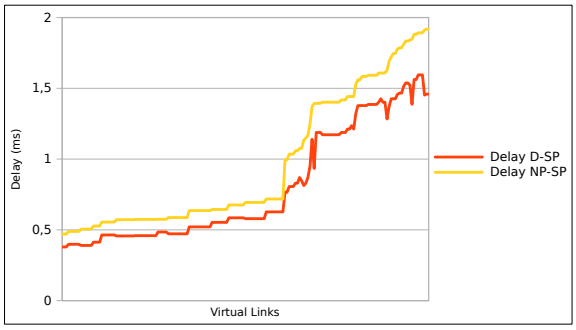

Fig. 6. Delay comparison. D-SP vs SP-NP for VHP flows

TABLE IV. GAIN PRODUCED ON THE DELAY BY D-SP OVER NP-SP FOR THE HIGHEST PRIORITY FLOW

Gain	Percentage
Minimum	11.3%
Average	17.6%
Maximum	33%

First, have a look on the impact of the D-SP vs. NP-SP for the very high priority flows (VHP).

Then, figure 6 shows the comparison of delays between the Very High Priority flow with the D-SP scheduling and the same flows with NP-SP scheduling. Each x value is a VL identifier and the two curves are the values of the delay bounds. The VLs have been sorted by NP-SP delay: this is why the NP-SP curve is more regular than the D-SP one.

As expected, the VHP flow with the D-SP scheduling presents lower delays than with the NP-SP. This obviously lies in the fact that in the D-SP scheduling a VHP packet is always processed as soon as it arrives, except if another VHP packet is being processed at that moment. And on the other hand, in the NP-SP scheduling, such packet must wait its turn if a lower priority packet is being processed.

We can see that the delays in the case of the D-SP scheduling are always below 1.6 ms!

The gain of D-SP is to avoid the blocking factor of non-preemption. This gain increases linearly with the number of crossed switches. To illustrate it, the figure 6 is redrawn, sorting VLs by the number of crossed switches, and also drawing the number of crossed switches (cf. figure 7).

It must be noticed that the worst bound for flows crossing 3 or 4 switches with the D-SP policy is lower than most of the bounds for flows crossing 2 switches with classical NP-SP policy. Moreover, the delays for flows crossing 3 or 4 switches are comparable with the ones crossing only two switches. The D-SP policy somehow allows to extend a flow path without increasing the delays.

Table IV shows the positive gain that experiment the latencies if we use the D-SP scheduling instead of the NP-SP scheduling.

Second, have a look on the impact of D-SP vs. NP-SP for the high priority flows (HP).

The figure 8 shows the comparison of delays for the High

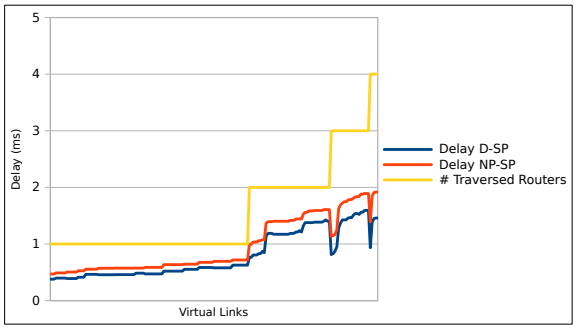

Fig. 7. Delay comparison. D-SP vs SP-NP for VHP flows augmented with path length

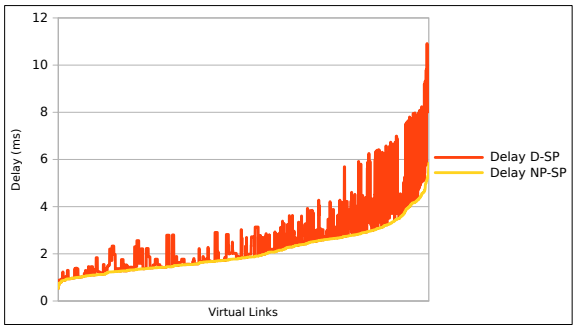

Fig. 8. Delay comparison. D-SP vs SP-NP for HP flows

Priority flow between the D-SP scheduling and the NP-SP scheduling.

In this case, the NP-SP scheduling clearly offers a better timing performance than the D-SP scheduling. Undoubtedly, in the cases where the HP packets of the D-SP have been disrupted, the delays must be way larger. We can see that by using the D-SP scheduling increases the bound on delays in a reasonable way, even if it can double for some flows.

Table V shows that the use of the D-SP scheduling instead of the NP-SP scheduling increases the bound of delay by 20% on average.

As we can see there are some flows where the loss is very important (up to 112.1%), but on average the loss is somehow equivalent to the average gain produced in the VHP flow detailed in table IV. However, this loss affects to more data flows than the gain produced in the VHP flow.

Finally, figure 9 shows the comparison of delays between the Low Priority flow of the D-SP scheduling and the NP-SP.

As expected, in this case the NP-SP also presents better delay results than the D-SP scheduling. However, the difference is even smaller this time. We can check the result of the negative losses in the following table.

TABLE V. LOSS PRODUCED ON THE DELAY BY D-SP OVER NP-SP FOR THE HP FLOW

Loss	Percentage
Minimum	0%
Average	19.4%
Maximum	112.1%

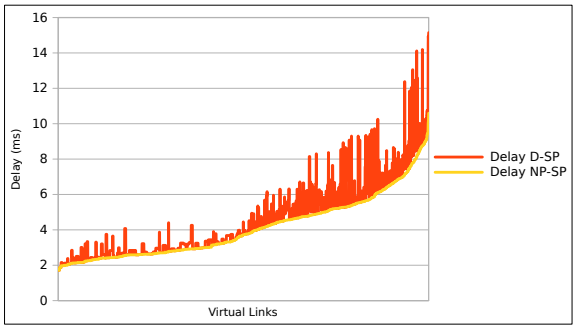

Fig. 9. Delay comparison. D-SP vs SP-NP for LP flows

TABLE VI. LOSS ON THE DELAY PRODUCED BY D-SP OVER NP-SP FOR THE HP FLOW

Loss	Percentage
Minimum	0%
Average	9.3%
Maximum	73.9%

In order to get an idea of the global impact produced by using the D-SP scheduling instead of using the NP-SP we compute the total delay for both scheduling systems.

The sum of all the delays in every VL for the D-SP turns out to be a 19920.66 ms, while for the NP-SP it is 17651.49 ms. Thus, the timing performance for the D-SP experiments a global loss of 12.9% over the NP-SP.

D. Performance of D-SP scheduling: changing very high priority flow proportion

The previous section has done a precise analysis of D-SP impact when 10% of the High Priority (HP) flow is scheduled in the D-SP Very High Priority class.

This section analyzes more coarsely this impact with different proportions of VHP class. The 5%, 10%, 15%, 25%, 50%, 75% and 100% proportions have been tested. Of course, the D-SP scheduling is designed to be used only for a small proportion of the total amount of flow: high proportions have been plotted only for completeness.

The effect of both the average gain and the average delay (printed in milliseconds) for every class priority is studied.

In figure 10 we can see that, as expected, for the VHP flow the more it is increased the proportion of D-SP HP flow over the HP flow, the timing performance gets worse, *i.e.* the average gain decreases almost linearly and the average delay increases.

Regarding the HP flow, we can see in figure 11 that again the more it is increased the proportion of VHP flow, the poorer the timing performance. As the average loss increases and so it does the delay. Obviously, by increasing the number of VHP flow, the HP has more competence against which it cannot really compete so its timing performance decreases.

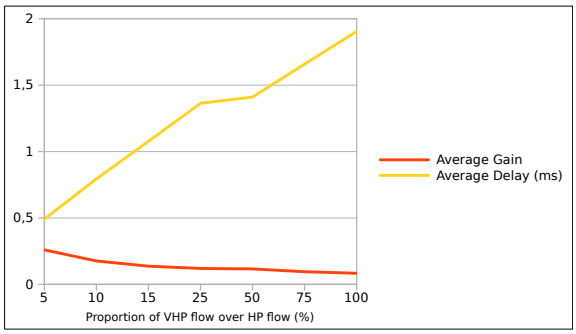

Fig. 10. Effect in the average gain and delay for the VHP flow

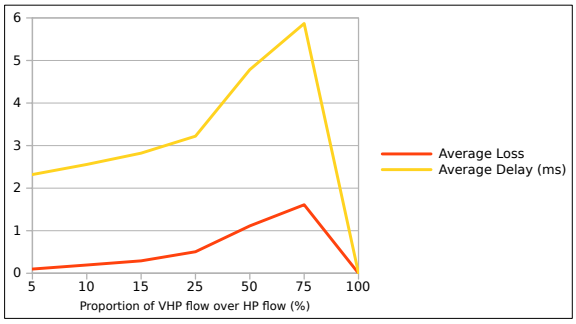

Fig. 11. Effect in the average gain and delay for the HP flow

Note that when the proportion of the VHP flow over the HP flow is 100%, actually the HP flow disappears that is why its average loss and average delay goes down to zero.

And finally, the same analysis is represented for the LP flow in figure 12. Once again, as the proportion of VHP flow increases the timing performance for the LP flow decreases.

VI. Conclusions and Future Work

The Disrupted Static Priority scheduling aims to improve the latency guarantees of any AFDX network, which currently uses the Non Pre-emptive Static Priority scheduling. Also, it is intended to keep the same low level of implementation complexity unlike the Pre-emptive Static Priority, which requires the splitting of a packet.

By means of an experimental realistic study-case using RTaW-PEGASE [13] it has been proven that the D-SP packet

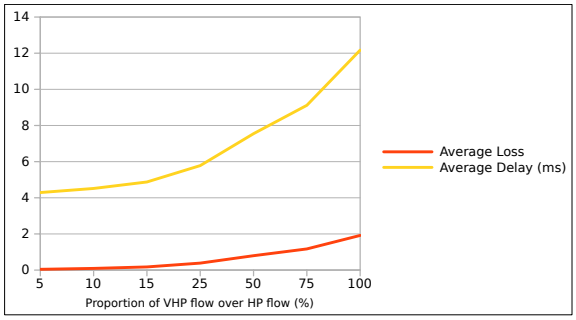

Fig. 12. Effect in the average gain and delay for the LP flow

scheduler provides way better performance in the latencies of the highest priority flow with respect to the NP-SP scheduling.

The gain is about 17% when VHP represents only 10% of the "high priority" flows, that is to say, 5% of the total number of flows. This gain is quite good, but especially, it allows to extend the path length, a hard constraint in system design when placing applications on computers.

However, this improvement also implies an increase in the delays for the lower priorities. Of course, favouring some flow, *i.e.* decreasing their latency, implies to hinder the others ones. In D-SP case, this is worth since some bandwidth is wasted.

So, finally the D-SP scheduling is a compromise between complying with lower latency constraints with some flows and keeping implementation simple. Our experiment shows that, as expected, the solution does not scale, but when considering that only a limited subset of flows (10% of the high-priority subset) requires very stringent latencies, a realistic AFDX network can offer very small latencies (from 0.5ms to 1.6ms) even for flows crossing 3 or 4 switches. Moreover, the impact on other flows is quite reasonable (20% for the high-priority flows and 9% for the low priority flows).

Adding complex mechanisms, either in hardware or software, always has a cost: in development, in power consumption, but also on efforts to ensure the correctness of the solution, for critical system requiring such certification.

This paper has presented one of the simplest solution to avoid frame blocking, and shown that it may be sufficient in some cases.

References

[1] AEEC, "Arinc 664 p7-1: Aircraft data network, part 7, avionics full-duplex switched ethernet network," Airlines Electronic Engineering Committee, Tech. Rep., september 2009.

[2] B. Matthews, H. Frazier, Y. Kim, and M. Teener, "Packet preemption for low latency," Oct. 27 2011, uS Patent App. 13/174,518. [Online]. Available: http://www.google.com/patents/US20110261814

[3] I. Computer Society, "Standard for local and metropolitan area networks-media access control (MAC) bridges and virtual bridged local area networks - amendment: Frame preemption," IEEE, Tech. Rep. 802.1Qbu, Draft 1.0, 2014.

[4] N. Finn, "Preemptive transmission advantages," http://www.ieee802.org/1/files/public/docs2012/new-avb-nfinn-preempt-advantage-0112-v02.pdf, 2012.

[5] W.-K. Jia, G.-H. Liu, and Y.-C. Chen, "Performance evaluation of ieee 802.1qbu: Experimental and simulation results," in *38th IEEE Conference on Local Computer Networks (LCN 2013)*, Oct 2013, pp. 659–662.

[6] SAE, "Time-triggered ethernet," SAE, Tech. Rep. AS6802, 2011.

[7] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan, "Ttethernet dataflow concept," in *Proc. of Eighth IEEE International Symposium on Network Computing and Applications (NCA 2009)*, july 2009, pp. 319 –322.

[8] J.-Y. Le Boudec and P. Thiran, *Network Calculus*, ser. LNCS. Springer Verlag, 2001, vol. 2050, http://lrcwww.epfl.ch/PS_files/NetCal.htm.

[9] C.-S. Chang, *Performance Guarantees in communication networks*, ser. Telecommunication Networks and Computer Systems. Springer, 2000.

[10] M. Boyer, G. Dufour, and L. Santinelli, "Continuity for network calculus." in *Proc of the 21th International Conference on Real-Time and Network Systems (RTNS 2013)*. Sophia Antipolis, France: ACM, October 16-18 2013, pp. 235–244.

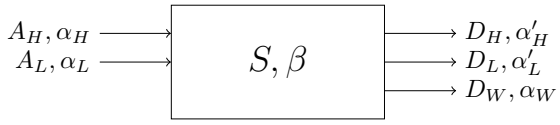

Fig. 13. Data flow through the server *S*

[11] A. Bouillard, L. Jouhet, and E. Thierry, "Comparison of different classes of service curves in network calculus," in *Proc. of the 10th International Workshop on Discrete Event Systems (WODES 2010)*, Technische Universität Berlin, August 30 - September 1 2010.

[12] H. Charara, J.-L. Scharbarg, and C. Fraboul, "Analysing end-to-end delays on an AFDX network by simulation," in *Communication Systems and Networks (IASTED-CSN), Palma de Mallorca, Spain, 28/08/2006-30/08/2006*, ser. ISBN 0-88986-606-6. http://www.ieee.org/: IEEE, 2006, pp. 171–176.

[13] M. Boyer, J. Migge, and M. Fumey, "PEGASE, a robust and efficient tool for worst case network traversal time," in *Proc. of the SAE 2011 AeroTech Congress & Exhibition*. Toulouse, France: SAE International, 2011.

APPENDIX

On this section the full proof of the theorem 1 is explained.

First, let's consider the following system represented in figure 6. We have two input flows, the High Priority flow and the Low Priority flow, passing through a server S with a service curve β. Then, three output flows depart the server, the High Priority and Low Priority flows and the *waste* flow.

Regarding the High Priority output flow, it can be directly bounded. By looking at figure 2 we can state that for the HP flow, the behaviour of the D-SP scheduling is equivalent to that of a non pre-emptive scheduling where l_2 would be the LP packet size. Therefore, by using the expression of the residual service for a non pre-emptive scheduling [11], we can affirm:

$$\beta_H = [\beta - l_2]^+_\uparrow \tag{11}$$

With respect to the LP flow, the result is not a direct application of existing results. A data flow approach will be used in order to reach the solution.

First, as illustrated in figure 13, the output of the server can be modeled as the aggregation of three kinds of flows: the high priority flow, D_H, the low priority flow D_L that is the subpart of the low priority flow correctly received, and D_w, the "wasted" part, that is the subpart of the low priority flow canceled plus the overhead of preemption (denoted w in figure 2).

The critical part of the proof is the evaluation of the wasted part. It is given as a separate lemma.

Lemma 1 (Arrival curve of waste flow): Let S be a server which offers a strict minimal service β, shared by 2 flows (High Priority and Low Priority), with respective arrival curves α_H and α_L, and scheduling flows with a D-SP policy, l_2 is the data size corresponding to the stopping time of the Low priority packet processing, i.e the *transition*; $l_H^{\min}$ is the minimum packet size of the High Priority flow and $l_L^{\max}$ is the maximum packet size of the Low Priority flow.

Then, an arrival curve of the waste flow is given by

$$\alpha_w(\Delta) = (l_L^{\max} + l_2) \left\lceil \frac{\alpha_H(\Delta)}{l_H^{\min}} + 1 \right\rceil \tag{12}$$

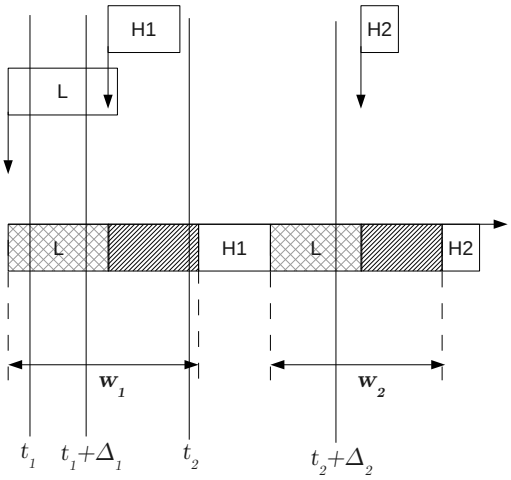

Fig. 14. First zoom on the D-SP scheduling policy

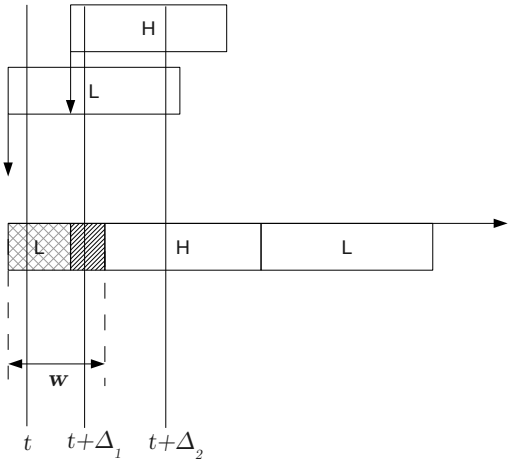

Fig. 15. Second zoom on the D-SP scheduling policy

Proof: The aim if to compute a bound on expression $D_H(t+\Delta) - D_H(t)$ for any non negative t and Δ.

Figure 14 and figure 15 show different possibilities regarding the position and duration of the interval Δ in the D-SP scheduling policy.

First, assume that there is no HP packet (*i.e.* not any bit of any HP packet) in interval $[t, t+\Delta]$, then: it can nevertheless have one "waste" packet (cf. interval $[t_1, t_1+\Delta_1]$, in figure 14).

When this occurs, we can assure that:

$$D_w(t+\Delta) - D_w(t) \leq l_w^{\max} \leq l_L^{\max} + l_2 \tag{13}$$

Second, consider that there is at least one HP packet in interval $[t, t+\Delta]$. Let n be the number of first bit of HP packet in the interval. Then, each first bit can interrupt a low priority packet and generate $l_w^{\max}$ bits of waste data flow. Moreover, at the end of the interval, it can exists a low priority packet that will be interrupted after $t+\Delta$.

Then, n is the number of starting packets from the A_H

flow in interval $[t, t+\Delta]$,

$$D_w(t+\Delta) - D_w(t) \leq (n+1) l_w^{\max} \tag{14}$$

That is because each arriving HP packet generates as most one *waste* packet, plus one more after $t+\Delta$.

Then, from the definition of the arrival curve (eq. 5), as it upper bounds the amount of data sent on any interval of width Δ, we can get a maximum number of packets transmitted during Δ:

$$nb.packets_{max}(\Delta) = \left\lceil \frac{\alpha(\Delta)}{l_{min}} \right\rceil \tag{15}$$

where l_{min} is the minimum packet size, which depends on the network characteristics.

To conclude:

$$D_w(t+\Delta) - D_w(t) \leq (l_L^{\max} + l_2) \left\lceil \frac{\alpha_H(\Delta)}{l_H^{\min}} + 1 \right\rceil \tag{16}$$

■

With this lemma, the proof of theorem 1 can be given. It is done in a quite classical way.

Proof of theorem 1:

Let be $t, \Delta \geq 0$ such that $(t, t+\Delta]$ is a backlog period. From the definition of the strict service for our particular system:

$$\begin{aligned} D_H(t+\Delta) + D_L(t+\Delta) + D_w(t+\Delta) \\ - D_H(t) - D_L(t) - D_w(t) \geq \beta(\Delta) \end{aligned} \tag{17}$$

Let be s the start of the backlog period of the high priority flow, as represented in figure 16, formally defined as

$$s = \sup \{x \leq t \mid A_H(x) = D_H(x)\} \tag{18}$$

Being both A_H and D_H left-continuous functions, then $A_H(s) = D_H(s)$. Moreover, we know that $A_H \geq D_H$, then

$$\begin{aligned} D_L(s+\Delta) - D_L(s) \geq \beta(\Delta) - A_H(s+\Delta) + A_H(s) \\ - D_w(s+\Delta) + D_w(s) \end{aligned} \tag{19}$$

By using the definition of an arrival curve (eq. 5) for the particular case of the A_H flow and for the interval Δ: $A_H(s+\Delta) - A_H(s) \leq \alpha_H(\Delta)$. And by substituting in the main equation:

$$\begin{aligned} D_L(s+\Delta) - D_L(s) \geq \beta(\Delta) - \alpha_H(\Delta) \\ - D_w(s+\Delta) + D_w(s) \end{aligned} \tag{20}$$

Now, using lemma 1, $D_w(t+\Delta) - D_w(t) \leq \alpha_w(\Delta)$

$$\begin{aligned} & D_L(s+\Delta) - D_L(s) \\ & \geq \beta(\Delta) - \alpha_{A_H}(\Delta) - \alpha_w(\Delta) \\ & \geq \beta(\Delta) - \alpha_{A_H}(\Delta) - (l_L^{\max} + l_2) \left\lceil \frac{\alpha_H(\Delta)}{l_H^{\min}} + 1 \right\rceil \end{aligned}$$

■

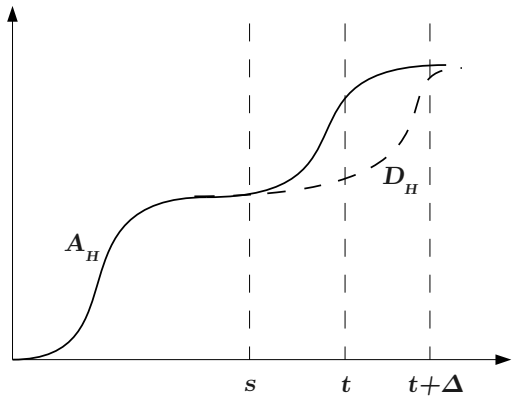

Fig. 16. Backlog period representation