Combining static priority and weighted round-robin like packet scheduling in AFDX for incremental certification and mixed-criticality support

Marc Boyer[1], Nicolas Navet[2], Marc Fumey[3], Jörn Migge[4], Lionel Havet[4]
[1] ONERA - The French Aerospace Lab, Toulouse, France
[2] University of Luxembourg, Luxembourg
[3] Thales Avionics, Toulouse, France
[4]RealTime-at-Work, Villers-lès-Nancy, France

Abstract

The Deficit Round Robin (DRR) policy can be used at the outgoing ports of communication switches to schedule distinct classes of frames, providing each class with a guaranteed share of the network bandwidth. The independence between traffic classes helps to improve the incremental design process, incremental certification and scheduling flows with mixed criticalities. DRR leads however to a less efficient use of hardware resources, this is why we also envisage the combined use of DRR and Static Priority (SP). We then provide a first quantitative assessment on a realistic case-study about the use of DRR, possibly combined with SP, in avionics networking and shed some light on its range of applicability.

1. Introduction

Avionics Full DupleX (AFDX - see[8]) is an aeronautic-specific switched Ethernet technology that supports data exchanges among avionics sub-systems with bounded latencies, without requiring the sub-systems to share a common global clock like in TDMA networks. These predictable latencies can be achieved because the workload submitted to the network by each sub-system is upper bounded and known in advance, and because the standard enforces appropriate switching mechanisms within the communication switches. In particular, the packets waiting to be sent on an output port of a communication switch are to be stored in a FIFO queue or in distinct queues depending on their priorities (high or low). The use of priorities (static priorities on a per-flow basis) actually helps meeting more stringent latencies for the most critical data flows than what would be feasible with FIFO queues alone. However, scheduling with priorities does not provide a solution to one of the current limitations of incremental certification on AFDX: defining independent sub-networks. Indeed adding a single data stream may have an impact on all the other streams that are transmitted, which requires re-computing the upper bounds on the latencies, and, possibly, involves reconfiguring the communication parameters throughout the whole system. This dependency raises issues for network engineering: one single actor is in charge of network topology and budgets (Virtual Links) definition. Next generation switches will certainly have more computing power and offer more bandwidth: we propose to take advantage of this to partition the network and simplify its design, targeting incremental certification and mixed criticality scheduling.

In this paper, we study the use of Deficit Round Robin (DRR) to schedule the transmissions of the packets on the output links of the AFDX switches. DRR is a low-complexity weighted round-robin like policy which allows the flows to share the available bandwidth according to some pre-defined percentages. Practically, DRR guarantees that a flow or a set of flows will be provided at run-time at least the share of bandwidth that has been allocated to them. If some spare bandwidth is provisioned for future use, it becomes possible to add new data flows to an existing system without any impact on the real-time performances of the pre-existing flows. Importantly, this opens the doors to identify sub-networks improving incremental certification process: independence between sub-networks and independent flow definition inside each sub-network. Among the numerous variants of Round-Robin, DRR is an appealing policy in the avionics context because it is predictable in terms of worst-case latencies [1,10] and the associated run-time overhead within the switches is contained (there are O(1) implementations [9]). Besides, DRR does not require any changes in the AFDX frame format or at the end-systems level (e.g. communication stack): only the switches needs to implement new mechanisms and offer additional configuration parameters. However, AFDX is often intended to support traffic with mixed-criticalities, specifically in terms of timing requirements, which, as it will be shown, can be best achieved by the combined use of DRR with priority scheduling, using priorities to guarantee low latencies for flows with strong timing constraints.

The first objective of the paper is to discuss how DRR can possibly be used in an avionics context, be it as the system-wide policy or in conjunction with prioritized scheduling, in order to best support both incremental certification and transmission of data flows with mixed timing criticalities. This question is topical considering the evolutions that can be foreseen in terms of technology (e.g., Gbit/s switches) and traffic characteristics (e.g., increasing amount of low-criticality entertainment-related flows). The second objective of the paper is to provide a first quantitative assessment on a realistic case-study about the use of DRR in avionics networking and identify its range of applicability. Specifically, the latency overhead with DRR (wrt to priority scheduling) is evaluated and it is shown that, even with 1Gbit/s, DDR alone is not suited for the most stringent timing constraints of that specific case-study. Then, the experiments conducted suggest that, taking advantage of the increasing bandwidth, combining static priority and DRR may provide a practical and efficient solution for next generation avionics systems with mixed-criticality traffic and the need for incremental certification.

2. The use of Deficit Round Robin in avionics networks

The main benefit of using DRR is the possibility to guarantee the service offered to the flows belonging to a certain class, independently of the rest of the traffic, simplifying incremental certification and the scheduling of flows having mixed criticalities. But, with regard to Static Priority (SP) or even FIFO scheduling, DRR usually imposes some additional latency to the flows, which is especially an issue for the most stringent timing constraints. As it will be explained, we believe that the combined used of DRR and SP might be a solution to alleviate the schedulability issue with DRR alone.

2.1 Deficit Round Robin for incremental certification and mixed criticality

Deficit Round Robin shares the bandwidth of a server, such as an AFDX link, between some "classes" of traffic. The configuration of DRR requires defining the number of classes n, and allocating to each class i a quantum Q_i. For reasons related to the algorithm itself, this quantum must not be too small compared to the size of the frames. It is generally required that the quantum Q_i allocated to a class of traffic must be greater than or equal to the maximum frame size L_i in this class ($Q_i \geq L_i$). Then, for a given class i, the worst case latency can be computed, using the following parameters:

- the global capacity of the server, β,
- the allocated quanta, $Q_1 \ldots Q_n$,
- the maximal frame size in the different classes, $L_1 \ldots L_n$,
- the traffic of the considered class.

DRR provides a way to partition the network, like IMA is a way to partition the calculators (CPU, memory, I/O...). Indeed, the network can be split into sub-networks (*i.e.* classes), devoted to some sub-system (flight control, supervision, maintenance, cabin, entertainment, etc.), enabling thus an *incremental design process process* at network level permitting to delegate VL definition to sub-systems. The global capacity is shared between sub-networks by allocating quanta $Q_1 \ldots Q_n$, to the DRR classes. If the maximal size of the frames in each class is not known in advance, the maximal AFDX frame size can be used (*i.e.*, 1518 bytes). Once the configuration is done, the worst case performance for a flow in a certain class can be computed independently of the exact composition of the other classes. It means that if the configuration of some flow changes during the design of the system, only the class the flow belongs to will be impacted.

Moreover, to compute the worst case performance of a flow in some class i, only the knowledge of the sum of the other quanta is needed, independently of their exact values. It means for instance that two classes can be merged into a new one, without impact on the other classes, provided that the quantum of the new class is the sum of the merged ones. Symmetrically, a class can be split into subclasses, as long as the sum of quanta is equal to the initial value (and that each quantum is greater than or equal to the maximal frame size in a class). Similarly if some spare bandwidth is provisioned for future use in some class or as distinct classes, it becomes possible to add new data flows to an existing system without any impact on the real-time performances of the pre-existing flows. This offers the possibility to improve *incremental certification* as explained above.

But DRR is also a means for *mixed criticality* scheduling since the partitioning of the bandwidth protects each class from each others. For example, some classes can be devoted to pure Ethernet traffic, without any VL contract (e.g., entertainment flows), while other classes may handle classical AFDX VL-based avionic flows. Nevertheless, it is clear that the independence between traffic classes comes at a price: the information about the exact composition of the traffic in the different traffic classes is not considered in the computation of the worst-case performance for a flow. This will lead to less tight upper bounds on the frame latencies, jitters and buffer sizes, and thus ultimately to less efficient use of hardware resources. We believe however that this suboptimal resource usage may be offset by the steady technological progress that makes it possible to envisage much higher data rates than the standard AFDX 100Mbit/s.

The DRR algorithm is designed in such a way that the bandwidth unused by some class is shared and used at run-time by the other classes having traffic to transmit, but this unused bandwidth cannot be taken into account in the analysis of the worst-case timing behaviour if one wants to preserve the independence between classes. Indeed, there are some works [3] in the literature that propose a global analysis of DRR, considering the characteristics of the flows in all classes, but the independence between classes is then lost, like it is the case in current AFDX analyses [6, 11].

2.2 Combining Static Priority and Deficit Round Robin to improve schedulability

As explained in the previous section, the DRR partitioning has a downside: it increases the worst-case latency. For a VL with stringent timing constraint (*i.e.*, requiring short network traversal time), this latency might not be acceptable. A solution we discuss here is the use of a hierarchical scheduling combining Static Priority and Deficit Round Robin. Having different priority levels for the frames is already possible: the current AFDX standard [8] requires two levels of priority and more could be integrated in future evolutions. Nevertheless, two levels are sufficient for our proposal. The principle is to keep all the priority levels but the lowest for the flows with the strongest timing constraints. For the flows at the lowest priority level, DRR scheduling is applied and each flow is allocated to a certain DRR class. At run-time, DRR scheduling is executed only if there are no packets from higher priority flows waiting to be transmitted. This hierarchical scheduling, Static Priority first and then Deficit Round Robin, denoted by SP/DRR, is illustrated in Figure 1.

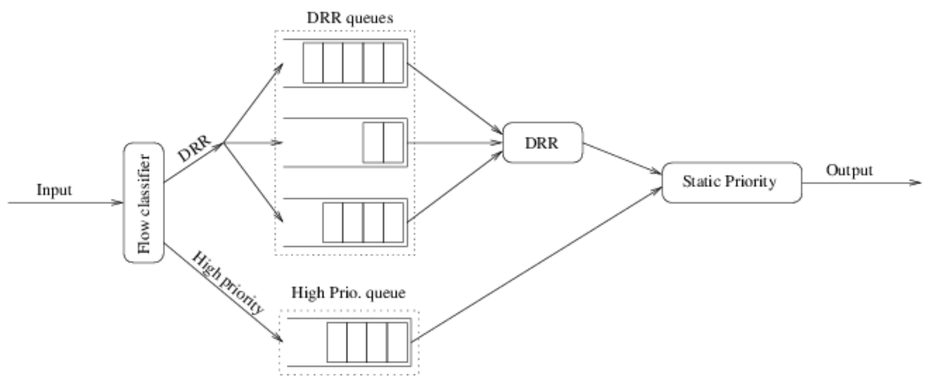

Figure 1: Hierarchical Static Priority/Deficit Round Robin scheduling, here with a single priority level for static priority flows and a single priority level for all DRR classes

Computing a bound on the network traversal time for the high priority flows is independent of the DRR scheduling, and the same analyses as in standard AFDX can be used [6,11]. But for the lowest priority flows, the DRR analyses must be adapted to take into account the higher priority flows. Building on the DRR timing analysis published in [2], the study in [1] proposes a DRR analysis extended to any hierarchical scheduling of the form *X*/DRR, and, in particular, it is able to handle SP/DRR. In SP/DRR scheduling, the performances of a given flow in a given DRR class depends on the flows at the highest priority level and DRR parameters, but still remains independent of flows in the other DRR classes.

In the design of such a SP/DRR network, two solutions can be adopted: either early design of high priority flows, or reservation-based design. By “early design of high priority flows”, we mean that the high priority flow contracts (*i.e.*, routing, frame size and BAG) must be set quite early in the design phase, and then changes must be as far as possible avoided, since any change in high priority flows contracts will have an influence on the whole network. A “reservation based” design relies on the creation of “phantom” or “spare” high priority VL. Since AFDX is being used for more than 10 years, system designer can often have a good estimate of the necessary amount of flows with strong requirements, based on the experience gathered from previous programs. Then, the latency of the DRR classes can be computed. Each time a real high priority VL is added to the system, one removes a spare high priority VL. As long as the frame size (resp. BAG) of the VL that is added is less than or equal to (resp. greater than or equal to) the one from the spare VL, the interference of this high priority flow on the low priority ones is acceptable *w.r.t.* worst-case latencies. If, at some point, there are no more spare high priority VLs available, a new analysis must be performed in order to compute the new DRR latency, based on the existing high priority VLs and some newly created spare VLs. In that case a complete incremental certification cannot be achieved, but this process will dramatically reduce the number of global analyses during the design phase. Whatever is done independence between flows cannot always be achieved in a very constrained system, where SP is needed, but DRR offers a way to reduce the dependences among the network flows.

3. Scheduling frames under Deficit Round Robin

Different policies have been defined to share the capacity of a server among several flows. GPS is an ideal policy explicitly designed to allow fair sharing. In GPS, each flow R_i receives a fractional part $c_i\,\beta$ of the global system service β (with $\Sigma c_i = 1$). As mentioned in [3]: ``a problem with GPS is that it is an idealised discipline that does not transmit packets as entities. It assumes that the server can serve multiple sessions simultaneously and that the traffic is infinitely divisible". A well-known GPS approximation is P-GPS (aka Weighted Fair Queueing [3]), that offers the same guarantees as GPS in a packet system, up to one packet size deviation. Nevertheless, P-GPS implementation is rather complex and other GPS approximations are used. The Deficit Round Robin algorithm is a practical and efficient implementation of the GPS paradigm. In DRR, each flow R_i receives a credit or quantum Q_i, and DRR ensures that each flow will get a fractional part $c_i = Q_i/(\Sigma c_i)$ of the service. DRR is among the most used GPS-like policy, since it exhibits a low complexity and can be implemented in very efficient ways, like the Aliquem implementation of DRR [9] whose algorithmic complexity is *O(1)*. However, the use of DRR involves larger latencies than under GPS and P-GPS and those latencies must be carefully analyzed. This will be discussed in §3.2.

3.1 Deficit Round Robin implementation

The DRR algorithm is presented in Figure 2. We assume that there are *n* input classes, and when a packet of the *i*-th class enters the system, it is stored into the *i*-th queue. The behaviour of the scheduler is the following: an infinite global loop scans all queues in sequence. If the *i*-th queue is non empty, it is selected, the Deficit Counter DC[i] is increased by Q_i, the quantum of the class, and as long as DC[i] is greater than or equal to the size of the head-of-line packet, the head-of-line packet is sent and the DC[i] counter is decreased accordingly. The term “deficit” comes from the fact that the DC counter records the difference between what should have been sent so far for a flow and what has actually been sent. The loop ends when the flow lacks enough credit to send the next packet or its queue gets empty. In the latter case, DC[i] is set to zero.

```
Input: Per flow quantum: $Q_1 \ldots Q_n$ (Integer)
Data: Per flow deficit: DC[1..n] (Integer)
Data: Counter: $k$ (Integer)
for i= 1 to n do
    DC[i]= 0 ;
endfor
while true do
    for i= 1 to n do
        if not empty(i) then
            DC[i]= DC[i] + $Q_i$ ;
            while (not empty(i)) and (size(head(i)) $\leq$ DC[i]) do
                DC[i]:= DC[i] – size(head(i)) ;
                Send(head(i)) ;
            endwhile
            if (empty(i)) then
                DC[i]= 0 ;
            endif
        endif
    endfor
endwhile
```

Figure 2: Deficit Round Robin algorithm

3.2 Deficit Round Robin worst-case service

The use of DRR algorithm in critical embedded systems requires a method to upper bound the network traversal time. Intuitively, DRR only offers a fractional part of the bandwidth, and there is a waiting time before a class is selected (aka latency). This intuition has been formalised in [9] and [10], and these studies provide a formula for the latency term. But these results only hold for DRR, not for SP/DRR. Cisco and Juniper Networks, both manufacturers of networking equipments, have defined their own variant of DRR designed to meet both low-latency and fairness requirements, and the latencies of these variants have been analyzed in [9]. The performance evaluation of SP/DRR in this paper does not make use of the timing analyses of these DRR variants, since they do not apply in an AFDX context, but it relies on the results from [1] that have been implemented in RTaW-Pegase (see §4.2). The timing analysis is based on the compositional Network Calculus theory [12]. If a server offering a service curve β is scheduled under DRR with n classes of respective quanta Q_i and maximal frame size L_i, then, class i receives the service β_i defined by:

$$\beta_i = \frac{Q_i}{\sum_{j=1}^{n} Q_j}\beta - \frac{Q_i\left(\sum_{j\neq i} L_j\right) + \left(\sum_{j\neq i} Q_j\right)\left(Q_i + L_j\right)}{\sum_{j=1}^{n} Q_j} \quad (1)$$

Intuitively the first term in equation (1) is the ideal share of bandwidth that class i should receive and the second term that is subtracted is the overhead induced by DRR which depends on the quantum sizes and maximum frame sizes. This overhead corresponds to the fact that, in the worst-case, a class-i packet arriving may have to wait that all the other classes have been served once before being served in its turn. The basic DRR analysis based on equation (1) is then extended in [1] to handle the case where a subset of flows is scheduled under non-preemptive static priority, providing us with an analysis for SP/DRR hierarchical scheduling.

4. Experiments on the use of DRR in avionics

4.1 Experimental setup

The use of DRR has been studied using a realistic industrial size configuration provided by Thales Avionics. Table 1 summarizes the main characteristics of the AFDX network under study.

Table 1: Structure and size of the AFDX case-study

Entities	Number
End Systems	104
Routers	8
Virtual Links	974
Latency constraints	6501

As can be seen in Table 1, each Virtual Link (VL) has on average 6 destination end systems. This explains the 6501 latency constraints, which means also that as many WCTT bounds, one for each flow, need to be evaluated. The characteristics of the traffic flows are detailed in Table 2. For the sake of simplicity, the latency and jitter that come from the sending end-systems communication stack are neglected throughout this study.

Table 2: Characteristics of the traffic

	# Virtual Link destinations	**BAG (minimum interarrival time)**	**Maximal Packet Size**	**# Traversed Routers**	**Latency Constraints**
Minimum	1	2 ms	100 bytes	1	1 ms
Average	6.6	60 ms	380 bytes	1.3	10.04 ms
Maximum	84	128 ms	1500 bytes	4	30 ms

This experimental configuration, where many flows have timing constraints as low as 1ms, is not feasible without using several priority levels and DRR alone cannot thus lead to a feasible solution. Thus, we have relaxed the deadlines of all flows not meeting their constraint under a global FIFO scheduling, by assigning to each of these flows, the smallest deadline that can be achieved under FIFO (the deadline is rounded to the immediately larger millisecond). The laxity of the most stringent flows is thus minimal and cannot be taken advantage of by DRR in the experiments that follow.

4.2 RTaW-PEGASE: timing analysis for AFDX networks

RTaW-PEGASE is a tool to compute upper bounds on communication delays and buffer utilization in AFDX and switched Ethernet networks. It relies on the Network-Calculus formalism for timing analysis, and its algorithms, coming from the literature or developed in cooperation with academics in particular within the PEGASE research project [15], have been proven correct in peer-reviewed publications (see [1,6,13] and more recently with the help of formal methods in [14] to assert the correctness of the computation). Thus, unlike most other timing analysis tools, RTaW-PEGASE is no black-box.

Another purpose of RTaW-PEGASE is to determine optimized routing and priority allocation for the flows by optimization techniques. RTaW-PEGASE, that is developed by the company RealTime-at-Work (see http://www.realtimeatwork.com/software/rtaw-pegase/), can be coupled with task scheduling tool and timing analysis software for other avionic buses such as CAN/ARINC825 or ARINC429 so as to verify complete avionics communication architecture.

Over the years, a number of traffic models and verification algorithms have been developed and integrated into the Network Calculus theory, and there are now many possibilities to choose from, each offering a specific trade-off with regard to accuracy (tightness of the bounds), computation time (e.g., linear or exponential complexity), complexity of the code and generality of the underlying models. RTaW-PEGASE has been conceived so as to enable the user to select the techniques that are best suited at each phase of the development cycle: research on Network Calculus theory, preliminary feasibility assessment, design space exploration, certification, etc. In the rest of the paper, the computation will be performed using the most precise algorithms that have been shown in [5] to give results (on average over all VLs) within 16% of the actual worst-case latencies on a set of AFDX configurations.

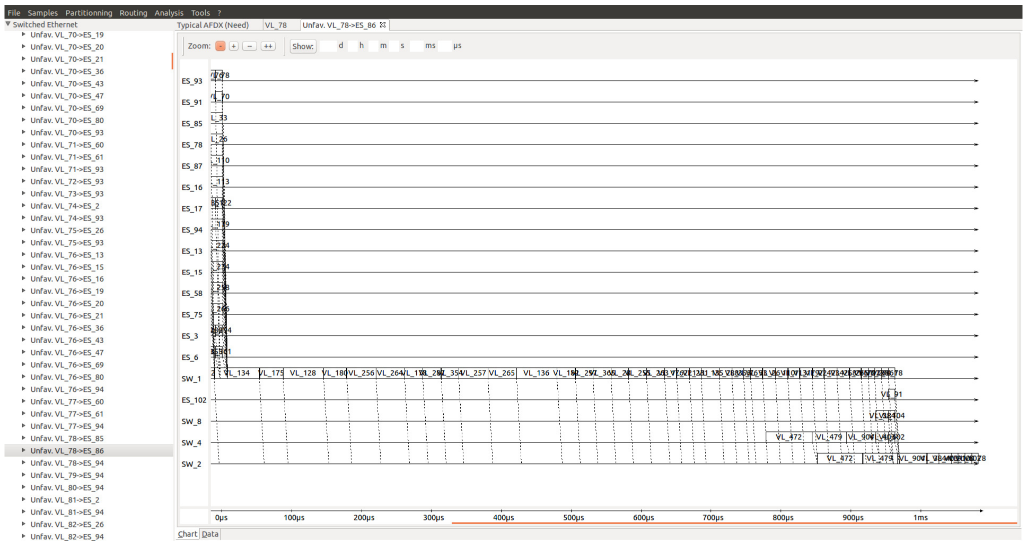

Figure 3: Screenshot of RTaW-PEGASE showing the Gantt diagram of an "unfavourable" trajectory of the system leading to degraded latencies for a certain virtual link of the reference case-study considered in the experiments. Though the situation exhibited is not necessarily the worst case, it provides usually an excellent lower bound on the latency for a flow and helps to gauge the tightness of the computed upper bounds. This feature, which relies on algorithms published in [16,17], helps to pinpoint the bottlenecks in the network.

4.3 Performance of DRR scheduling

Generating DRR configurations. On the basis of the realistic configuration described in §4.1, the VLs are then allocated to the DRR classes. The number of DRR classes is set while the allocation of the VLs to the DRR classes is done at random. If a VL is allocated to a DRR class, this will hold for all the flows of that VL throughout the whole network (*i.e.*, in all the switches). It should be pointed out that how the VLs should be allocated to the DRR classes, and how many DRR classes there should be, is out of the scope of the paper but, in practice, this choice may be driven by application-specific constraints or industrial design practices, or it can be motivated by the search of an optimum in terms of resource usage for instance. To shed some light on the issue of the DRR efficiency, we believe that a random allocation of the VLs to the different DRR classes is a reasonable choice. In particular, it cannot be assumed that the VLs are grouped together according to their criticality because this will not be necessarily the case in an incremental design process.

In the following, n_{VL} denotes the total amount of VLs of our configuration, C_k is DRR class k to whom belongs a subset of the VLs and $n_{Classes}$ is the number of DRR classes used in the experiment. Each VL is identified by an unique index assigned at random between 1 and n_{VL}, and L_k indicates the VL-index boundary between class k and $k+1$ (with L_0=0). The allocation of the VLs in the k different classes is done with the following algorithm:

$$\begin{aligned}
&n_{avg} = \frac{n_{VL}}{n_{Classes}} \\
&L_k = n_{avg} \cdot k + \mathrm{Rand}(-n_{avg}/2, n_{avg}/2) \\
&C_k = \left\{ VL_i \,\middle|\, L_{k-1} < i \le L_k \right\} \\
&\text{With } L_0 = 1 \quad \text{and} \quad L_{n_{Classes}} = n_{Classes}
\end{aligned} \tag{2}$$

This algorithm basically allocates at random between 50 and 150% of the average number of VLs per class (i.e., *navg*) to a certain DRR class. Then, then VLs are allocated at random to the classes, without exceeding the limit set for each class. The next step is to assign a round-robin quantum to each DRR class. For this, the bandwidth requirement of each DRR class is first computed:

$$\mathrm{Load}_k = \sum_{\{i \mid VL_i \in C_k\}} \frac{\mathrm{Length}_i}{\mathrm{Bag}_i} \tag{2}$$

The quantum allocated to a class is directly proportional to its bandwidth requirement, with the property that the class with the least bandwidth requirement is still allocated a quantum sufficiently large to transmit any frame. Let M denotes the maximum packet size over the system, the quantum of DRR class *k*, denoted by *Qk*, is given as follows:

$$Q_k = M \cdot \frac{\mathrm{Load}_k}{\underset{i}{\mathrm{Min}}(\mathrm{Load}_i)} \tag{3}$$

Additional bandwidth requirements under DRR. The aim of this first set of experiments is to explore the minimum network speed increase that is required to keep all the VLs under their latency constraint when several DRR classes are introduced. Using the reference configuration described in §4.1 that is feasible (*i.e.*, all the VLs are under their latency constraints) with FIFO scheduling and all links at 100Mbit/s, we generate 100 random DRR configurations according to the process described in the previous paragraph. Then, for each configuration, we search for the minimum network data rate that leads to a feasible system. During the process, like in the initial configuration, it is assumed that the speed is the same over all links, be they links from end-systems to routers or links between routers. The computation performed with RTaW-Pegase takes a few seconds for a given data rate, and the search is speed up by a dichotomic search of various granularity (first 100Mbit/s then 10 Mbit/s).

The results shown in Figure 4 show that the average additional bandwidth required by DRR increases almost linearly with the number of DRR classes. In our experiments, the average bandwidth needed under DRR (up to 8 classes) is approximately the number of DRR classes times the minimum bandwidth needed under FIFO. The curve of the minimum value over the set of experiments is significantly below the average value and less smooth. The latter phenomenon can probably be explained by the variability among the set of candidate DRR configurations that are partially generated at random. The maximum data rates needed among the set of experiments (not shown in Figure 4) exhibit the same behaviour and, ranging from 270Mbit/s to 4450MBit/s, they are much larger than the average values (from a factor 1.3 up to a factor 10).

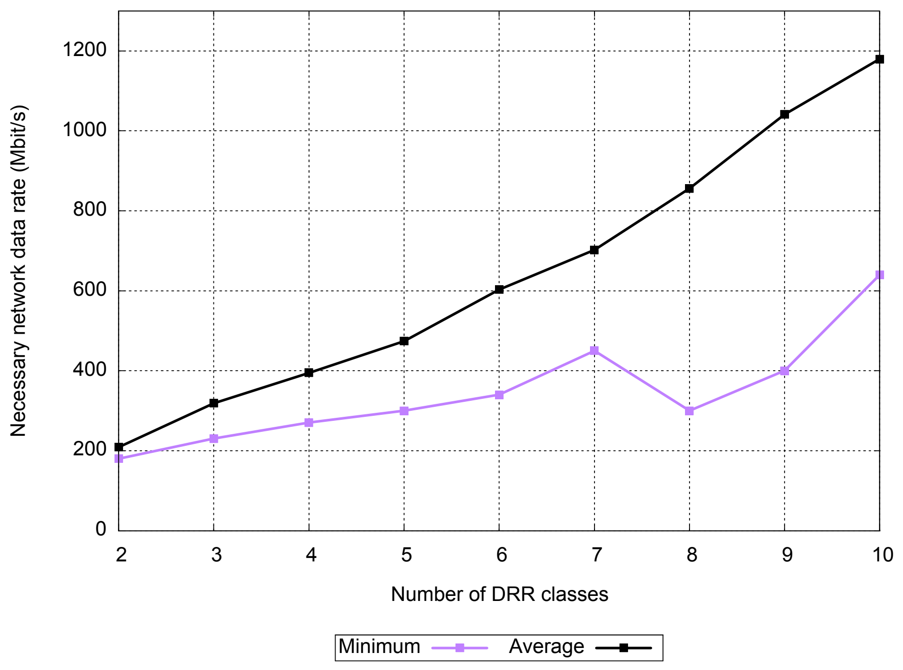

Figure 4: Network data rate required under DRR to meet the timing constraints for a number of DRR classes ranging from 2 to 10. The minimum and average values are obtained from the set of 100 DRR configurations assessed for a given number of DRR class. The reference configuration used to generate the DRR configurations is feasible under FIFO scheduling at 100Mbit/s with zero laxity for several flows.

We now consider the use of 1Gbit/s network, and if the minimum data rate needed for feasibility exceeds 1Gbit/s, the DRR configuration is deemed unfeasible. The results in Table 3 show that a 1Gbit/s network can ensure feasibility for almost all randomly generated configurations up to 6 DRR classes. The experiments shown in Figure 4 and Table 3 suggest to us that DRR can be a practical solution on 1Gbit/s networks with a traffic that is similar to the one from our reference configuration for a limited number of DRR classes.

Table 3: Percentage of DRR configurations feasible at 1Gbit/s for a number classes ranging from 2 to 10.

Number of DRR Classes	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
%	100	99	99	97	96	92	78	71	50

4.4 Performance of SP/DRR scheduling

In the first series of experiment, all VLs were assigned to DRR classes. We now explore the combined use of Static Priority and DRR (SP/DRR) with two priority levels as discussed in §2.2. For a given VL, the choice among the two policies is based on the latency constraint: if a VL has a latency constraint that is below (or equal to) a certain threshold l_c for at least one receiving end-system, it will be given the higher priority and will not be assigned to a DRR class. The VLs with their latencies above l_c are scheduled under DRR at the lower priority level. In the latter case, the DRR class to which the VL belongs is assigned at random. The experiments conducted with RTaW-Pegase share the same setup as the ones from §4.3 except for the higher priority traffic scheduled under Static Priority, with the threshold l_c set to 2ms.

The results of this experiment are presented in figure 5, along with the results obtained with DRR alone for the purpose of comparison. With SP/DRR, the same trend as with DRR alone can be observed: as the number of DRR classes grows, the bandwidth needed to obtain feasibility becomes larger. However, using priorities helps to offset the detrimental impact of a larger number of DRR classes: with 2 classes SP/DRR needs on average 14% less bandwidth than DRR while with 10 classes the gain of using SP/DRR reaches 39%.

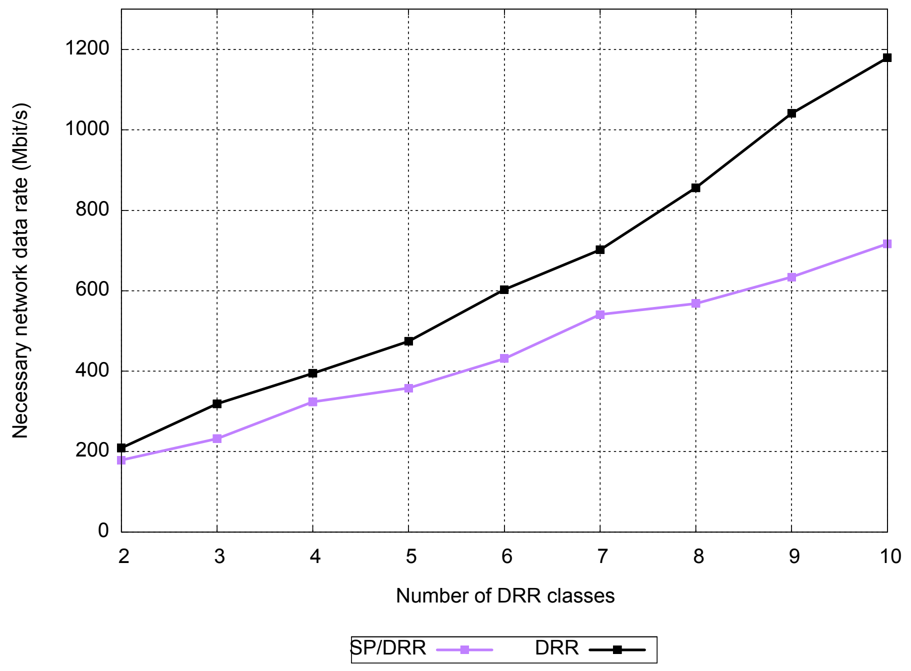

Figure 5: Average network data rate required under DRR and SP/DRR to meet the timing constraints for a number of DRR classes ranging from 2 to 10. Statistics obtained from the set of 100 DRR configurations assessed for a given number of DRR class. The VL latency threshold that distinguishes between SP and DRR scheduling is set to 2ms.

As done with DRR, we count the number of configurations not feasible at 1Gbit/s. The results in Table 4 show that a 1Gbit/s network under SP/DRR can ensure feasibility for almost all randomly generated configurations whatever the number of DRR classes.

Table 4: Percentage of DRR configurations feasible at 1Gbit/s for a number classes ranging from 2 to 10.

Number of DRR Classes	2	3	4	5	6	7	8	9	10
%	100	100	99	100	98	94	96	93	93

5. Conclusion and future work

DRR scheduling at the output ports of communication switches is a technical solution that may, by providing network partitioning, help to simplify avionics network design and their evolution, especially when incremental certification and mixed criticality scheduling are matter of concern. However, compared to static priority and FIFO scheduling, DRR leads to less efficient use of hardware resources. This paper aims to get some insight into how much more how bandwidth is needed, for a realistic system, to consider the use of DRR. Also, we discussed the possible joint use of Static Priority and DRR (SP/DRR) in order to alleviate the feasibility issues that may arise with DRR alone.

DRR, with a limited number of traffic classes, and SP/DRR, with a larger number of classes, have shown in our experimental setup to be feasible options on a 1Ggit/s network, for a traffic that is similar to the one from our reference configuration that includes VLs with strong timing constraints. These results need to be confirmed by more comprehensive experiments conducted on a wider range of configurations. However they suggest to us that, especially on Gbit/s networks, DRR and SP/DRR can be practical scheduling solutions offering a wide range of trade-offs between efficiency of resource usage on the one hand, and ease of design and evolution of the system on the other hand.

There are a number of open technical issues left, among which importantly how to best group together VLs into DRR classes and how to allocate the DRR quanta. Also, in a similar manner to what has been done in IP routers, it might be possible to develop variants of DRR better suited for the specific context of avionics systems.

References

[1] Boyer, M., Stea, G., and Mangoua Sofack, W. 2012. Deficit Round Robin with network calculus. In: *6th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'12)*.

[2] Hua, Y. and Liu, X. 2011. Scheduling Design and Analysis for End-to-End Heterogeneous Flows in an Avionics Network. In: *31rd Annual IEEE International Conference on Computer Communications (INFOCOM'11)*.

[3] Parekh, A. K., and Gallager, R. G. 1993. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM transactions on networking*, vol. 1; n° 3.

[4] ARINC. 2005. Aircraft data network Part 7: Avionic full duplex switched Ethernet (AFDX) network. ARINC Specification 664P7.

[5] Boyer, M., Navet, N., and Fumey M. 2012. Experimental assessment of timing verification techniques for AFDX, In: *Embedded Real-Time Software and Systems (ERTS 2012)*, Toulouse, France.

[6] Boyer, M., Migge, J., and Navet N. 2011. A simple and efficient class of functions to model arrival curve of packetised flows, In: *First International Workshop on Worst-case Traversal Time (WCTT), in conjunction with the 32nd IEEE Real-time Systems Symposium (RTSS)*, Vienna, Austria.

[7] Boyer, M., Migge, J., and Fumey M. 2011. PEGASE – a robust and efficient tool for worst-case network traversal time evaluation on AFDX, In: *SAE Aerotech 2011*, Toulouse, France.

[8] ARINC. 2005. Aircraft data network Part 7: Avionic full duplex switched Ethernet (AFDX) network. ARINC Specification 664P7.

[9] Lenzini, L. and Mingozzi, E. and Stea, G. 2002. Aliquem: a Novel DRR Implementation to Achieve Better Latency and Fairness at O(1) Complexity. In: *10th IEEE International Workshop on Quality of Service*.

[10] Kanhere, S. S. and Sethu, H. 2002. On the latency bound of deficit round robin. In: *11th Int. Conf. on Computer Communications and Networks (ICCCN 2002).*

[11] Grieu J. 2004. Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques. PhD Thesis. Institut National Polytechnique de Toulouse (INPT).

[12] Le Boudec, J.-Y. and Thiran, P. 2001. Network Calculus. LNCS 2050.

[13] Bouillard, A. and Thierry E. 2007. An algorithmic toolbox for network calculus, In: *Discrete Event Dynamic Systems*, vol. 17, no. 4.

[14] Mabille, E., Boyer, M., Fejoz, L and Merz, S. 2013. Certifying Network Calculus in a Proof Assistant, In: *5th European Conference for Aeronautics and Space Sciences (EUCASS)*, Munich, Germany.

[15] Boyer, M., and Navet, N., Olive, X. and Thierry, E. 2010. The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with network calculus, In: *4th Intl Symp. On Leveraging Applications of Formal Methods (ISoLA 2010)*, LNCS.

[16] Bauer, H., Scharbarg, J.-L. and Fraboul, C. 2012. Applying Trajectory approach with static priority queuing for improving the use of available AFDX resources, In: *Real-Time Systems*, 48:101–133.

[17] Bauer, H., Scharbarg, J.-L. and Fraboul, C. 2010. Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach, In: *IEEE Trans. on Industrial Informatics*, Vol. 6, n°4.