Hierarchical Traffic Shaping and Frame Packing to Reduce Bandwidth Utilization in the AFDX

H. AYED [1,2], A. MIFDAOUI[1], C. FRABOUL[2]
[1]University of Toulouse-ISAE, France
[2]University of Toulouse-ENSEEIHT/IRIT, France

Abstract*—The increasing complexity and heterogeneity of avionic networks make resource savings a challenging task to guarantee easy incremental design during the long lifetime of an aircraft. In this paper, we focus on the optimization of interconnection devices for multi-cluster avionic networks, called Remote Data Concentrators (RDC), and especially for the CAN-AFDX network. The design of this optimized RDC device consists in implementing frame packing strategies to manage upstream (sensors) flows to improve bandwidth utilization in the AFDX; and Hierarchical Traffic Shaping (HTS) algorithm to control downstream (actuators) flows to guarantee bandwidth isolation on CAN. Schedulability analysis integrating the effects of these new mechanisms is detailed and validated. Furthermore, a heuristic approach to tune the Hierarchical Traffic Shaping parameters within the RDC device is proposed to reduce as much as possible bandwidth utilization in the AFDX, while ensuring flows schedulability. The performance analysis conducted on a realistic avionic case study proves the efficiency of the optimized RDC device to reduce bandwidth utilization in the AFDX, compared to the basic device currently implemented in avionics.

***Keywords*-CAN, AFDX, RDC, Hierarchical Traffic Shaping, Frame Packing, Schedulability analysis, Optimization process**

I. Introduction

A. Context & Motivations

The complexity of avionic communication architecture is increasing inherently due to the growing number of interconnected end-systems and the expansion of exchanged data. To be effective in meeting the emerging requirements in terms of bandwidth, latency and modularity, the current avionic communication architecture, shown in Figure 1, consists of an AFDX backbone network to interconnect the critical avionic end-systems and some Input/ Output (I/O) data buses. Clusters are then interconnected via specific devices, called Remote Data Concentrators (RDCs), standardized as ARINC655 [1]. RDC devices are modular gateways distributed throughout the aircraft to handle heterogeneity between the AFDX backbone and I/O data buses.

Although RDC devices enhance avionics modularity and reduce maintenance efforts, they become in the same time a major challenge in the design process of such multi-cluster avionic architectures. Implemented RDC in new generation aircraft, such as the A400M or A350 is based on a naive frame-conversion strategy, called (1:1) strategy, where each non-AFDX frame is converted to an AFDX frame and vice-versa. This strategy is simple to implement, however it implies high network-resource use, and especially significant bandwidth

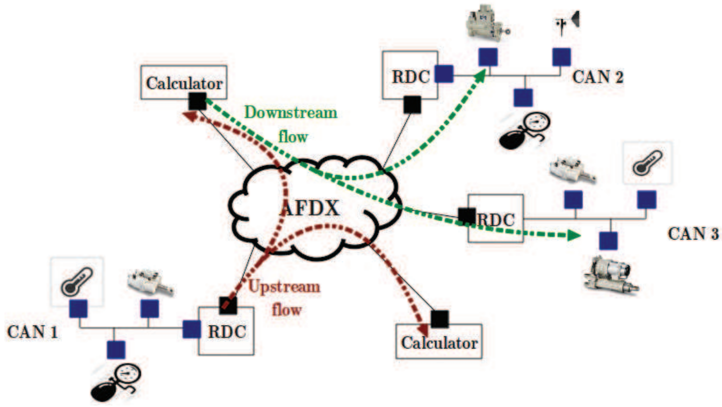

Fig. 1. *Multi-cluster avionic communication architecture*

utilization in the AFDX. This feature is important for avionic applications to guarantee easy incremental design and enhance margins for future avionic functions additions. Therefore, the design of an optimized RDC device integrating resource saving mechanisms becomes a necessity to enhance the scalability and performances of avionic applications.

In [2], as a first step the authors introduced an optimized RDC device implementing a novel dynamic frame packing strategy based on a waiting timer, called Fixed Waiting Time (FWT) strategy. Results for a representative avionic case study showed a noticeable improvement of system performance, compared to classic RDC device with a simple (1:1) strategy. Then, to obtain further enhancements in terms of bandwidth utilization in the AFDX, an accurate static frame packing strategy, called Messages-Set Partitioning (MSP) strategy, was integrated [3]. The optimized RDC device implementing the MSP strategy induces significant bandwidth utilization reduction, compared to the one implementing the FWT strategy. In this previous work, two main assumptions were considered: (i) ignoring the contentions on I/O data buses between upstream and downstream flows, i.e., incoming (and outgoing) flows to (and from) AFDX calculators from sensors (to actuators, respectively), by considering specific data buses either for sensors or actuators; (ii) RDC device can interconnect only one I/O data bus to the AFDX.

In this paper, the design of such an optimized RDC device is extended to cover the general case where: (i) I/O data buses are shared between sensors and actuators; (ii) an RDC

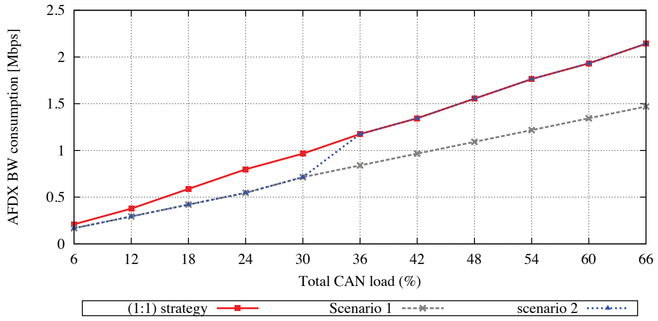

Fig. 2. *Bandwidth Utilization in the AFDX with shared I/O network*

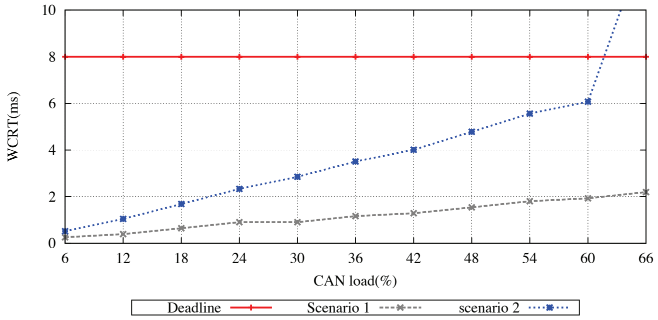

Fig. 3. *WCRT on CAN of upstream flows with shared I/O network*

device can interconnect many I/O data buses to the AFDX. Under these new assumptions, First results for a CAN-AFDX case study concerning bandwidth utilization in the AFDX are illustrated in Figure 2. These results are obtained when varying the CAN load and using MSP strategy to pack upstream frames within RDC device, with reference to (1:1) strategy. Scenarios 1 and 2 represent two configurations of priority assignment of upstream and downstream flows on CAN. The former is when upstream flows have higher priority than downstream flows, where the latter represents the inverse.

Under scenario 1, we still observe a significant reduction of bandwidth utilization in the AFDX when using the optimized RDC device implementing MSP strategy, compared to the basic one with (1:1) strategy. Hence, under this priority assignment configuration, our proposal in [3] is still efficient. However, with scenario 2, the performance of the optimized RDC device is degraded and becomes equivalent to the basic RDC device performance under high CAN load (from 36 %). To understand the reasons of this degradation, let's take a look on the Worst-Case Response Times (WCRT) on CAN of upstream flows, shown in Figure 3. As can be noticed, WCRTs increase significantly under scenario 2 because of contentions with higher priority downstream flows. However, increasing upstream flows delays on CAN is not in favor of performing frame packing within RDC device, and consequently of reducing bandwidth utilization in the AFDX.

In avionics context, the modification of application specifications can ramify maintenance efforts and incremental design process. Therefore, revising priority assignment of different flows to improve system performances can be a complicated task for designers. Our aim consists in reducing as much as possible bandwidth utilization in the AFDX induced by the RDC device, even in the worst-case scenario of priority assignment for upstream flows, i.e., upstream flows have lower priority than downstream flows. Therefore, to overcome the limitations highlighted with these first results, the key idea is favoring frame packing mechanism for upstream frames within RDC device to reduce bandwidth utilization in the AFDX. This fact consists in minimizing as much as possible WCRTs on CAN of upstream flows when having the low priority, and ensuring at the same time the temporal constraints of downstream flows.

To minimize the interference from downstream flows on the transmission of upstream flows on CAN, and consequently the WCRTs on CAN of upstream flows, we propose the usage of Hierarchical Traffic Shaping (HTS) algorithm [4] within RDC device. This algorithm will control downstream flows transmission on CAN, and consequently guarantee bandwidth isolation between upstream and downstream flows. HTS algorithm consists of a set of traffic shapers, based on the leaky bucket method [5], and connected in a hierarchical way according to a tree structure. HTS algorithm is a special case of hierarchical server-based scheduling which has been successfully implemented in various network applications [4] [6] [7]. This paper extends the use of this algorithm within avionic RDC devices to control downstream flows and consequently to reduce bandwidth utilization in the AFDX, induced by upstream flows. This HTS algorithm implements two levels of traffic shaping. The first level is based on greedy method [8] which comes for free, to control individual downstream flows, and consequently to reduce the jitter due to the AFDX network [9]; where the second level is used to shape aggregate downstream flows, scheduled according to fixed priority non-preemptive policy, to substantially reduce the number of flows introducing interference on upstream flows on CAN.

B. *Original Contributions*

The contribution of this work are:

- The design of an optimized RDC device implementing: (i) frame packing strategies to manage upstream flows to reduce bandwidth utilization in the AFDX; (ii) Hierarchical Traffic Shaping (HTS) algorithm to control downstream flows for minimizing interferences and ensuring bandwidth isolation on CAN;
- The schedulability analysis for upstream and downstream flows integrating the effects of frame packing and HTS algorithm within the RDC device;
- A heuristic approach to tune the HTS parameters to minimize as much as possible bandwidth utilization in the AFDX while ensuring flows schedulability;
- The validation of the optimized RDC device performances through a realistic avionic case study.

C. Paper Organization

In the next section, we give an overview of AFDX and CAN technologies and review the most relevant work in the domain of Hierarchical Traffic Shaping in networks. Afterwards, in Section III, we explain the main concepts of Hierarchical Traffic Shaping algorithm and its integration in the RDC device. In Section IV, we present the schedulability analysis of upstream and downstream flows in this case. Section V describes the tuning process of Hierarchical Traffic Shaping parameters. Finally, in section VI, we conduct performance analyses to evaluate the efficiency of our proposal to improve resource savings in the AFDX for a realistic avionic case study.

II. Background and Related Work

In this section, we give an overview of AFDX and CAN technologies and review the most relevant work in the domain of Hierarchical Traffic Shaping in networks.

A. Network Technologies

The AFDX [10] network is based on the Full Duplex Switched Ethernet protocol at 100Mbps. This technology manages the large amount of exchanged data through policing mechanisms added in switches and the Virtual Link (VL) concept. This concept provides a way to reserve a guaranteed bandwidth for each traffic flow. The VL represents a multicast communication which originates at a single end-system and delivers packets to a fixed set of end-systems. Each VL is characterized by: (i) BAG (Bandwidth Allocation Gap), ranging in powers of 2 from 1 to 128 milliseconds, which represents the minimum inter-arrival time between two consecutive frames; (ii) MFS (Maximal Frame Size), ranging from 64 to 1518 bytes, which represents the size of the largest frame that can be sent during each BAG.

CAN native protocol [11] is a 1 Mbps data bus that operates according to an event-triggered paradigm where messages are transmitted using a priority-based access mechanism. Collisions on the bus are resolved following a CSMA/CR protocol (Carrier Sense Multiple Access/Collision Resolution) thanks to the bit arbitration method. CAN frame includes a payload up to 8 bytes and an overhead of 6 bytes due to the different headers and bit stuffing mechanism.

B. Hierarchical Traffic Shaping in Networks

The Hierarchical Traffic Shaping (HTS) is a part of general Hierarchical Server-Based (HSB) Scheduling: each traffic shaper in the hierarchy structure is considered as a server which will bound the traffic burstiness sent within a limited time window. HSB scheduling is a common approach that has been used in many network applications to control interference between various traffic classes with different real-time requirements, i.e., Soft Real-Time (SRT) and Hard Real-Time (HRT) traffic.

Concerning industrial application and especially Real-Time Ethernet, one of the most relevant approaches based on HSB framework to guarantee a dynamic adaptation of servers was proposed in [7]. The authors presented a multi-level HSB architecture for Ethernet, implemented on commercial switches and based on FTT-SE (Flexible Time Triggered Switched Ethernet) paradigm [12]. Schedulability analysis was detailed and validated using experimentation. This approach is efficient in dynamic environment, and typically open networks. The servers parameters are assumed to verify a priori traffic temporal constraints.

In automotive applications, various approaches based on traffic shaping and HSB scheduling were proposed to improve CAN bus performances. In [9], traffic shaping algorithm based on leaky bucket method, and particularly greedy method [8], was integrated within gateways to reduce the jitter on the destination network and improve the schedulability of lower priority messages. However, this approach is considered as a limited form of HTS approach implementing only one level of traffic shapers to control individual input messages. In [6], HSB scheduler, based on earliest deadline first algorithm, was detailed to use CAN in a more flexible way compared to the native CAN. This approach improved bandwidth isolation among aperiodic traffic and was validated using simulation. Analytical approach to provide worst-case response times of messages and to guarantee messages schedulability was lacking.

In avionics application, traffic shaping is integrated in AFDX end-systems to guarantee a reserved bandwidth for each application and is standardized as Virtual Link concept. This approach guarantees bandwidth isolation between traffic flows and improve the predictability of the AFDX network. In this paper, we extend this approach by implementing HTS scheduling within RDC devices to interconnect AFDX backbone with I/O CAN buses. The main idea is to minimize the interference from downstream flows on the transmission of upstream flows on CAN, and consequently the WCRTs on CAN of upstream flows. This will favor frame packing mechanism for upstream frames within RDC device and consequently will reduce bandwidth utilization in the AFDX. Our proposal consists of two traffic shaping levels and a root server to implement native CAN scheduler. The idea of the first level of traffic shapers is similar to the one detailed in [9] where we consider greedy method, which does not increase maximum end-to-end delays as proved in [8], and reduce the jitter induced by the AFDX network. However, we extend this implementation by adding a second level of traffic shapers to substantially reduce the number of flows introducing interference on upstream flows on CAN. The schedulability analysis of upstream and downstream flows is proved and validated through a realistic avionic case study. Furthermore, unlike [7], a tuning process of the HTS parameters is proposed to minimize as much as possible bandwidth utilization in the AFDX while ensuring flows requirements.

III. Hierarchical Traffic Shaping within RDC devices

In this section, we first give an overview of the RDC architecture with the different implemented mechanisms. Then, the integration of Hierarchical Traffic Shaping (HTS) algorithm

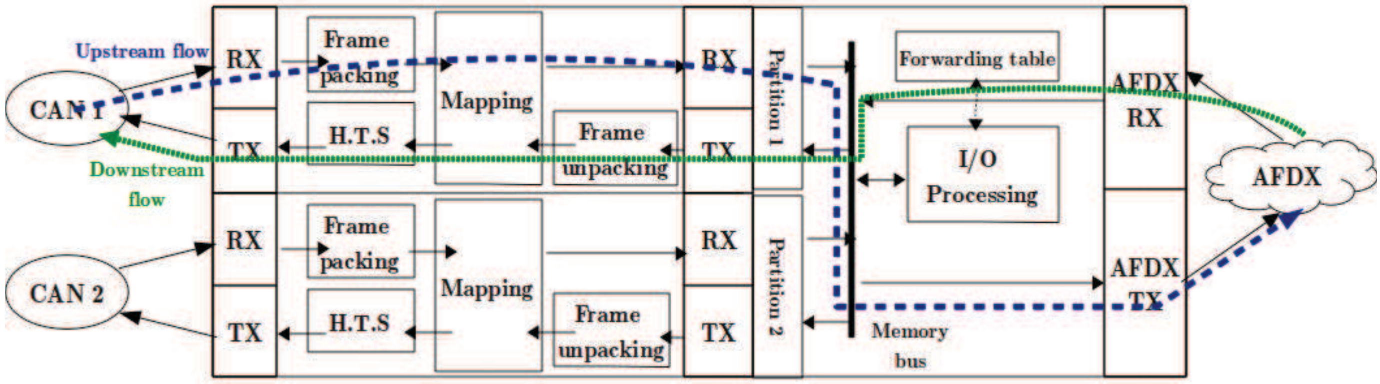

Fig. 4. *RDC Architectural Overview*

within the RDC device is detailed. Finally, the HTS modeling is described.

A. *RDC Architectural overview*

The proposed RDC device implements:

- frame packing strategies to manage upstream flows to reduce bandwidth utilization in the AFDX;
- Hierarchical Traffic Shaping (HTS) algorithm to control downstream flows for minimizing interferences and ensuring bandwidth isolation on CAN;
- support of multiple I/O networks using partitioning to provide the required safety level by ensuring that an errant I/O network running in one partition will not affect the others.

As shown in Figure 4, for each I/O network, upstream flows are first processed by the Frame Packing unit, implementing MSP strategy [3]. This latter is illustrated in Figure 5 and consists in defining off-line input frame partitioning where each sub-partition represents the associated subset of an output frame. Then, based on a static mapping table, one Virtual Link is associated to multiple CAN messages. The frame packing is synchronized with the reception of the most urgent input frame among each defined sub-partition. A timeout is implemented to avoid losing all the accumulated messages in case of non-reception of the most urgent one. This strategy will reduce the induced overhead in the AFDX, and consequently bandwidth consumption. Afterwards, the Rx Buffer of the associated partition stores the incoming frames in FIFO (First In First Out) order. The role of the I/O Central Processing Unit (CPU) is to move packets for the incoming Rx buffers to the outgoing Tx buffer of the AFDX interface.

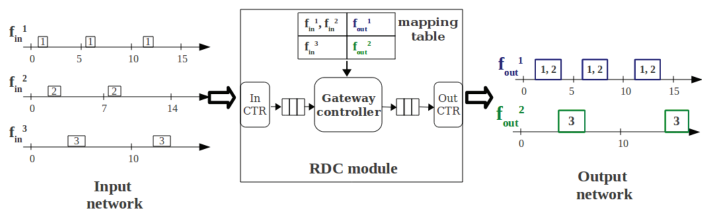

Fig. 5. *MSP Packing Strategy Process*

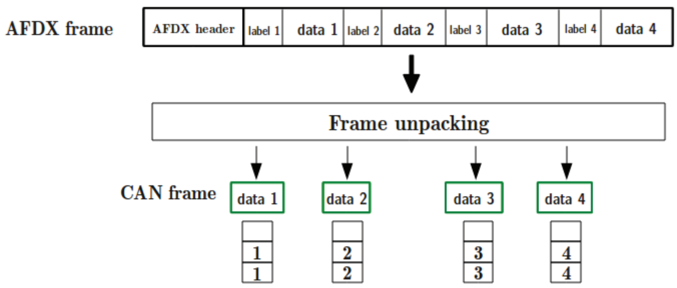

Fig. 6. *Frame unpacking process*

For downstream flows, the I/O CPU move packets from the incoming AFDX Rx buffer to the corresponding partition Tx Buffer, based on the forwarding table. Then, these received packets are processed within Unpacking Frame unit, detailled in Figure 6. This latter extracts one or many elementary data from the same AFDX frame depending on the data packing performed within the initial AFDX source [13], i.e., generated data from different AFDX applications hosted by the same end-system can be grouped within the same AFDX frame to minimize communication delay and bandwidth utilization in the AFDX. Afterwards, based on static mapping table, CAN identifiers are defined. Hierarchical Traffic Shaping unit is performed to eliminate the jitter due to the AFDX network and to minimize interference on CAN. This unit is detailed in the next section.

B. *Hierarchical Traffic Shaping Structure*

The HTS structure is based on a set of traffic shapers and servers connected in a tree structure and defined in a static manner a priori, as shown in Figure 7. This latter is organized into three levels: leaf, inner and root.

- Leaf traffic shapers are implemented to control incoming packets from the AFDX interface of the RDC device. They are based on greedy method [8] which comes for free, i.e., does not increase maximum end-to-end delays. However, they reduce efficiently the observed jitter up their reception at the RDC device [9]. This fact enhances lower priority messages schedulability on CAN. These

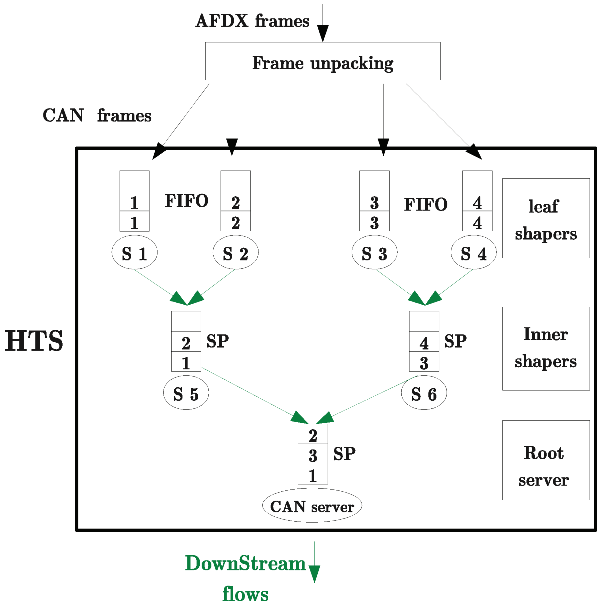

Fig. 7. *Hierarchical Traffic Shaping Structure*

shapers are based on leaky bucket method and offer to incoming packets the same guarantees than the associated AFDX Virtual Link. Hence, at the output of these traffic shapers, the minimum inter-arrival time between packets is equal to the corresponding BAG and the maximum frame size is bounded.

- Each inner traffic shaper is based on leaky bucket method to shape aggregate downstream flows of outgoing packets from leaf shapers after being classified and scheduled according to fixed priority non-preemptive policy. The aim of these shapers is to substantially reduce the number of flows introducing interference on upstream flows and to guarantee bandwidth isolation on CAN. One or many inner traffic shapers can be implemented depending on the incoming traffic rate. Indeed, shaping all the incoming flows using the same inner traffic shaper will induce small inter-arrival time between packets at the output, and consequently important interference with upstream flows. The tuning process of these inner shapers will be detailed in Section V.
- The root server implements simply fixed priority non-preemptive scheduling which represents the CAN native behavior. All the packets will be multiplexed at the root server according to their corresponding priorities.

C. Hierarchical Traffic Shaping Modeling

Consider S_{up} and S_{down} for upstream and downstream flow sets, respectively. For any stream flow $m \in S_{up} \cup S_{down}$, we associate four characteristics $\{T_m, L_m, Dl_m, P_m\}$ which represent period, maximum payload, deadline and priority used to access to the CAN bus, respectively. We consider a strict order of CAN priorities, i.e., for any two messages m_k and m_j, $P_{m_k} < P_{m_j}$ means that message m_k has higher priority than m_j.

The HTS is used to manage only downstream flows to minimize interference with upstream flows on CAN. Each leaf traffic shaper is applied for only one type of downstream flow and consequently admits the same period and authorized maximum payload than its associated flow. However, an inner traffic shaper is applied to a group of outgoing flows from leaf shapers. Then, each inner shaper sh in the set of inner shapers Sh_{inner} is characterized by $\{T_{sh}, L_{sh}, Dl_{sh}, P_{sh}\}$, where:

- T_{sh} is the period. This value is comprised between $T\min_{sh}$ and $T\max_{sh}$ depending on the characteristics of its input downstream flows set S_{sh}. To support the aggregate flow rate, $T\max_{sh}$ is at most equal to $\frac{1}{\sum_{i \in S_{sh}} \frac{1}{T_i}}$. Furthermore, to avoid overflowing the CAN bus, we consider that $T\min_{sh}$ is at least equal to 1ms, which is an arbitrary choice that integrates CAN transmission capacity and typical production periods of CAN sources. If $\frac{1}{\sum_{i \in S_{sh}} \frac{1}{T_i}} > 1\text{ms}$, than this configuration is possible; else we need to investigate other HTS configurations;
- L_{sh} is the maximum payload size where $L_{sh} = \max_{i \in S_{sh}} L_i$;
- Dl_{sh} is the deadline and is equal to the period T_{sh};
- P_{sh} is the associated priority to the inner shaper. This value depends on the considered communication way and is equal to $P\min_{sh}$ or $P\max_{sh}$. To cover the worst-case from the downstream flows point of view, this priority is considered as the lower priority among all its input downstream flows set, $P_{sh} = P\max_{sh} = \max_{i \in S_{sh}} P_i$. However, the worst-case from the upstream point of view corresponds to considering the higher priority among all its input downstream flows set, $P_{sh} = P\min_{sh} = \min_{i \in S_{sh}} P_i$.

IV. Schedulability Analysis

In this section, we present the sufficient schedulability test to perform schedulability analysis of upstream and downstream flows. Then, we detail the timing analysis of upstream and downstream flows integrating frame packing and HTS algorithm effects. Finally, we give some numerical results and first conclusions on the optimized RDC performances.

A. Sufficient Schedulability Test

For avionic embedded applications, it is essential that the communication network fulfills certification requirements, e.g., predictable behavior under hard real-time constraints and temporal deadline guarantees. The use of a frame packing process and Hierarchical Traffic Shaping within the RDC will increase communication latencies and real-time constraints have to be checked. In order to deal with the worst-case performance analysis of such networks, we consider as metric the worst-case end-to-end delay that will be compared to the temporal deadline for each frame.

The end-to-end delay of each upstream and downstream flows consists of three parts, as shown in Figure 8:

- d_{CAN}: the Worst Case Response Time (WCRT) on CAN. The classic schedulability analysis for a native CAN bus

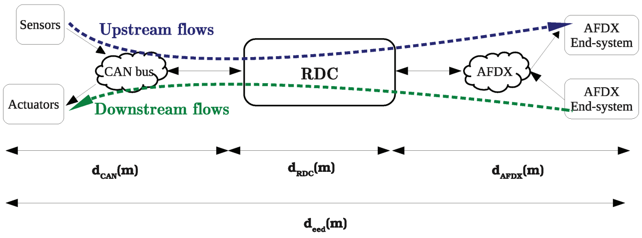

Fig. 8. *end-to-end delay for upstream and downstream flows*

[14] based on a Fixed Priority non-preemptive scheduler is considered in this paper;

- d_{RDC}: the maximal duration the message might be delayed in the RDC. This delay depends on the communication way and the crossed RDC functions;
- d_{AFDX}: the upper bound on the AFDX delay. A maximum bound on AFDX delay will be considered herein to simplify the schedulability analysis, using results in [15].

The schedulability test is as follows:

$$\forall m \in S_{up} \cup S_{down}:$$
$$d_{CAN}(m) + d_{RDC}(m) + \max_{m}(d_{AFDX}(m)) \quad \leq \quad Dl_m \tag{1}$$

where S_{up} and S_{down} are upstream and downstream flow sets, respectively.

Hence, the schedulability test becomes:

$$\forall m \in S_{up} \cup S_{down},$$
$$d_{CAN}(m) + d_{RDC}(m) \quad \leq \quad Dl_m - \max_{m}(d_{AFDX}(m)) \tag{2}$$

For any message m, Dl_m and $\max_m(d_{AFDX}(m))$ are known. Therefore, in the next sections, we will focus on the upper bounds for $d_{RDC}(m)$ and $d_{CAN}(m)$, distinguishing upstream and downstream cases to verify the schedulability test 2.

B. Timing Analysis for Upstream Flows

First, the RDC device delay imposed to upstream flows is due to frame packing unit. The upper bound of this delay was detailed in [3]. We proved that the worst-case waiting time for a CAN message m_j in the RDC device occurs when it arrives immediately after the end of reception of the most urgent message m_u in the RDC, i.e., m_u is the message with the smallest period among the subset messages packed with m_j in the same AFDX frame.

An upper bound of the waiting time in the RDC for CAN-message m_j is:

$$WT(m_j) \quad = \quad \begin{cases} 0 \quad if \quad j = u \\ T_u + d_{CAN}(m_u) \quad otherwise \end{cases} \tag{3}$$

Therefore, the upper bound of RDC delay $d_{RDC}(m_j)$ is the sum of: (i) a technological latency due to payload extraction and relaying process, called ϵ; (ii) waiting time in the RDC between the reception instant of the CAN message and the transmission instant of its associated AFDX frame $WT(m_j)$.

$$d_{RDC}(m_j) = \epsilon + WT(m_j) \tag{4}$$

Then, the upper bound on WCRT on CAN of an upstream flow is computed using classic results for Fixed Priority non-preemptive scheduler [14]. However, we consider null jitter due to leaf shapers, based on greedy method and implemented in HTS structure.

For a message $m \in S_{up}$,

$$d_{CAN}(m) \quad = \quad w(m) + C_m \tag{5}$$

where C_m is the maximum transmission time on CAN integrating the maximum payload size L_m and the transmission capacity of CAN bus, and $w(m)$ is the maximum queuing delay computed using the following expression:

$$w^{n+1}(m) = \max_{k \in lep(m)} C_k + \sum_{k \in hp(m)} \left\lceil \frac{W^n(m)}{T_k} \right\rceil * C_k \tag{6}$$

where $lep(m)$ and $hp(m)$ are the sets of messages with priorities lower or equal to m and with priorities higher than m, respectively.

We start with an initial value equal to $w^0(m) = C_m$ and continue until obtaining one of these two situations: (i) if $w^{n+1}(m) + C_m > Dl_m$, then stop and conclude that m is not schedulable; (ii) if $w^{n+1}(m) = w^n(m)$, then stop and conclude that m is schedulable.

In our case, we have to identify clearly $lep(m)$ and $hp(m)$ for each message $m \in S_{up}$ to integrate the impact of output flows of each inner shaper in the HTS structure on its worst-case response time. For each message $m \in S_{up}$,

- $hp(m) = \{k \in S_{up} \cup Sh_{inner} \ / \ P_k < P_m\}$: set of messages with priorities higher than m among upstream flows and inner shapers by considering $P\min_{sh}$ for each inner shaper $sh \in Sh_{inner}$;
- $lep(m) = \{j \in S_{up} \cup Sh_{inner} \ / \ P_j \geq P_m\}$: set of messages with priorities lower or equal than m among upstream flows and inner shapers by considering $P\min_{sh}$ for each inner shaper $sh \in Sh_{inner}$.

The schedulability test in 2 can be easily verified using Eq. 4 and 6.

C. Timing Analysis for Downstream Flows

The RDC device delay imposed to downstream flows is mainly due to the blocking time of the HTS algorithm. For each message m, first the leaf shaper, based on greedy method, comes for free and does not increase the end-to-end delay. Then, at the inner shaper sh, the worst-case blocking time is computed using the following iterative expression until convergence:

$$\begin{cases} B^0_{shaper}(m) = X_m * T_{sh} \\ B^{n+1}_{shaper}(m) = X_m * T_{sh} + \sum_{k \in hp_{inner}(m)} \left\lceil \frac{B^n_{shaper}(m)}{T_k} \right\rceil * T_{sh} \end{cases}$$

where,

$$X_m = \begin{cases} 0 & if \;\; lp_{inner}(m) = \emptyset \\ 1 & otherwise \end{cases}$$

and,

- $hp_{inner}(m) = \{k \in S_{sh} \;/\; P_k < P_m\}$: set of messages shaped with the same inner shaper sh, with priorities higher than m;
- $lp_{inner}(m) = \{j \in S_{sh} \;/\; P_j \geq P_m\}$: set of messages shaped with the same inner shaper sh, with priorities lower than m.

The stop condition is $B^{n+1}_{shaper}(m) = B^{n}_{shaper}(m)$.

This worst-case blocking delay is computed based on the classic model for Fixed Priority non-preemptive scheduler in [16]. We consider in our case that any message $m \in S_{sh}$ will occupy the shaper during T_{sh} because the shaper does not send more than one packet per T_{sh}.

Therefore, the upper bound of RDC delay $d_{RDC}(m)$ is the sum of: (i) the technological latency due to payload extraction and relaying process, ϵ; (ii) the maximum blocking time of inner shape in the HTS structure, $B_{shaper}(m)$.

$$d_{RDC}(m) = \epsilon + B_{shaper}(m) \quad (7)$$

In the other hand, the upper bound on worst-case response time on CAN for a downstream flow still is computed using Eq. 5. However, the $lep(m)$ and $hp(m)$ for each message $m \in S_{down}$ to integrate the impact of upstream flows are as follows:

- $hp(m) = \{k \in S_{up} \cup Sh_{inner} \;/\; P_k < P_m\}$: set of messages with priorities higher than m among upstream flows and inner shapers by considering $P\max_{sh}$ for each inner shaper $sh \in Sh_{inner}$;
- $lep(m) = \{j \in S_{up} \cup Sh_{inner} \;/\; P_j \geq P_m\}$: set of messages with priorities lower or equal than m among upstream flows and inner shapers by considering $P\max_{sh}$ for each inner shaper $sh \in Sh_{inner}$.

The schedulability test in 2 can be verified using Eq. 6 and 7.

D. Numerical Results

We consider an example of I/O CAN bus with downstream and upstream flows described in Tables I and II, respectively. This set of flows consists of 48 messages with payload equal to 8 Bytes and periods between 8 and 32 ms. We assume the worst-case priority assignment configuration for upstream flows, i.e., all downstream flows have higher priority than upstream flows.

TABLE I
DOWNSTREAM FLOWS CHARACTERISTICS

Message	Period (ms)	Payload (bytes)	Priority
$m^1 - m^8$	8	8	$1 - 8$
$m^9 - m^{16}$	16	8	$9 - 16$
$m^{17} - m^{24}$	32	8	$17 - 24$

TABLE II
UPSTREAM FLOWS CHARACTERISTICS

Message	Period (ms)	Payload (bytes)	Priority
$m^{25} - m^{32}$	8	8	$25 - 32$
$m^{33} - m^{40}$	16	8	$33 - 40$
$m^{41} - m^{48}$	32	8	$41 - 48$

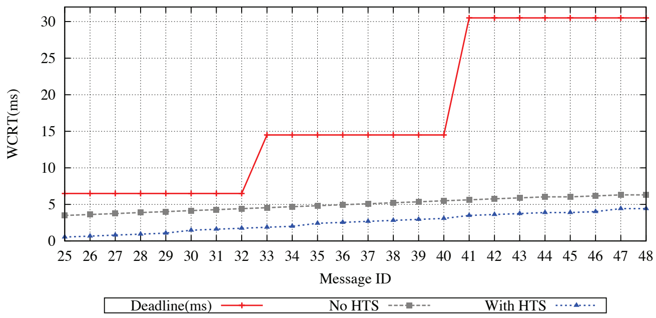

Fig. 9. *WCRTs of upstream flows on CAN*

TABLE III
BANDWIDTH UTILIZATION IN THE AFDX

Configuration	AFDX Bandwidth (in Mbps)
(1:1)	1.15
MSP + NO HTS	1.05
MSP + HTS	0.7

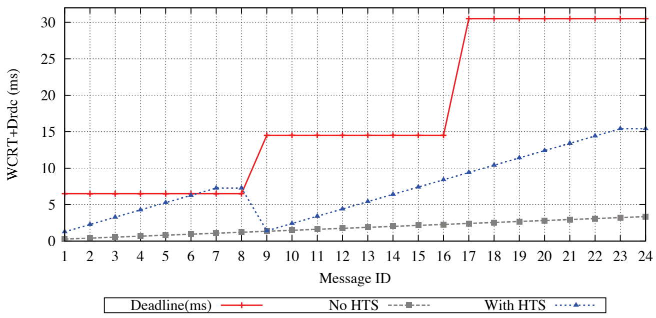

Fig. 10. *WCRTs of downstream flows on CAN*

As illustrated in Figure 9, the WCRTs of upstream flows on CAN decrease when HTS algorithm is implemented within the RDC device, compared to the basic RDC device. This reduction is in favor of performing MSP strategy within RDC device, and consequently reducing bandwidth utilization in the AFDX as described in Table III. Indeed, using HTS algorithm and MSP strategy within the optimized RDC device offers a noticeable improvement on the induced bandwidth in the AFDX, where reductions of 40% and 30% are obtained compared to the basic RDC device, i.e., implementing (1:1) strategy, and the optimized RDC device implementing only MSP strategy, respectively.

However, this improvement on WCRTs of upstream flows on CAN, and consequently bandwidth utilization in the AFDX, induces a degradation of WCRTs of downstream flows as shown in Figure 10. The blocking time due to inner shapers does not respect the timing requirements of downstream messages with identifiers 7 and 8. Therefore, we need a tuning process of HTS parameters to minimize as much as possible bandwidth utilization in the AFDX while ensuring downstream flows schedulability.

V. Hierarchical Traffic Shaping Tuning

In this section, we give a mathematical formulation of the HTS tuning problem. Then we detail the heuristic to find Hierarchical Traffic Shaping parameters which minimize bandwidth utilization in the AFDX while ensuring messages schedulability. Finally, some numerical results are illustrated for the same example described in Tables I and II.

A. Problem Formulation

To perform resource savings on the AFDX backbone and enhance margins for future function additions, an adapted optimization process is required to define the best HTS structure within RDC device. The bandwidth utilization in the AFDX induced by the RDC device is considered as a relevant metric to assess the resource saving margins in the AFDX network. Our objective is to find the best HTS structure, i.e. the number and parameters of inner shapers, which minimizes as much as possible the bandwidth utilization in the AFDX while guaranteeing the temporal constraints of upstream and downstream flows. The corresponding optimization problem is formulated as follows:

$$\begin{aligned} \underset{Sh_{inner}}{\text{minimize}} \quad & Bw(VL(Sh_{inner})) = \sum_{v_i \in VL(Sh_{inner})} \frac{L_{max}(v_i)}{BAG(v_i)} \\ \text{subject to} \quad & \forall m \in S_{up} \cup S_{down}, \\ & d_{CAN}(m) + d_{RDC}(m) \\ & \leq Dl_m - \max_m (d_{AFDX}(m)) \end{aligned} \tag{8}$$

where,

- $Bw(VL(Sh_{inner}))$ is the reserved bandwidth for upstream flows on the AFDX backbone; and it is equal to the sum of AFDX VLs rates depending on the parameters of inner shapers in Sh_{inner};
- the constraint corresponds to the schedulability test of upstream and downstream flows.

To solve this problem, we introduce in the next section an adequate heuristic approach to find the best HTS configuration which minimizes the objective function and respects the constraints.

B. HTS Heuristic Approach

The Figure 11 illustrates the proposed heuristic to find the optimal configuration of HTS. This heuristic will be processed only in the case where the basic RDC device or the optimized RDC device implementing only MSP strategy leads to a schedulable configuration of upstream and downstream flows. We are not looking through this heuristic to improve the

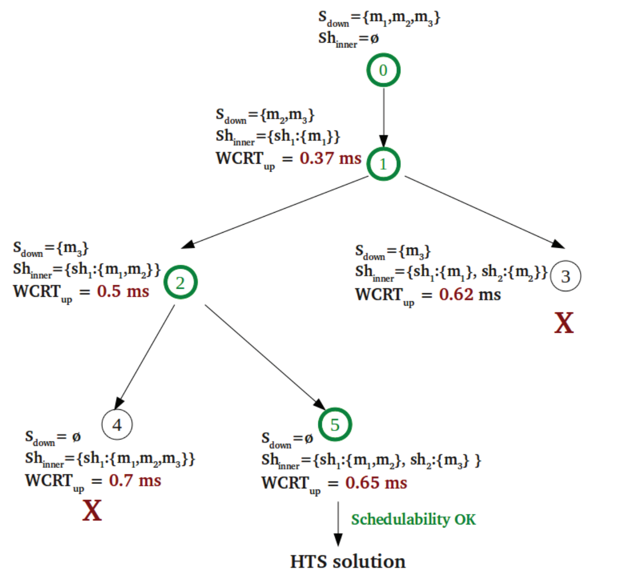

Fig. 11. *Example of The Heuristic Approach*

schedulability of the system but only the resource savings of the system to avoid a complexity explosion.

The different steps of this heuristic are as follows:

1) **Initialization**: First, the heuristic sorts downstream messages set S_{down} in non-decreasing order of periods. At this step, the set of inner shapers is empty, $Sh_{inner} = \emptyset$. The heuristic will start by allocating the first message in S_{down}.
2) **Iterative construction of inner shapers**: Then, the set Sh_{inner} is built iteratively. At the beginning, the first message in S_{down} is inserted in a new shaper sh that would be added to the list of HTS configurations $List^0_{Sh_{inner}}$. Then, for the next selected message in S_{down}, the heuristic is conducted as follows for each iteration $k \geq 1$:
   - (a) we add the selected message to each inner shaper in each HTS configuration in $List^{k-1}_{Sh_{inner}}$ and we build a new configuration by adding a new inner shaper containing only the selected message. Then, we update the inner shaper characteristics of each HTS configuration as defined in Section III-C. Furthermore, for each HTS configuration, we verify the schedulability condition of each upstream flow in S_{up} and of each selected downstream flow. Only feasible configurations if any are considered to form the list $List^k_{Sh_{inner}}$ and then go to step (b) until the stop condition is verified, i.e. each message in S_{down} has an associated inner shaper in the final HTS configuration. For $k \geq 2$ If there is no feasible configuration, go to step (c).
   - (b) for each configuration of inner shapers in the list $List^k_{Sh_{inner}}$, we compute the WCRTs of upstream flows on CAN. Then, we sort the list $List^k_{Sh_{inner}}$ in non-decreasing order of associated WCRTs ob-

tained for upstream flows. Then, we select the first inner shaper configuration Sh_{inner} in the sorted list $List^{k}_{Sh_{inner}}$ and we come back to step (a) by considering $List^{k-1}_{Sh_{inner}} = Sh_{inner}$ for the next selected message in S_{down}.

- (c) for each inner shaper configuration in $List^{k-1}_{Sh_{inner}}$, we compute the WCRTs of upstream flows on CAN. Then, we sort the list $List^{k-1}_{Sh_{inner}}$ in non-decreasing order of associated WCRTs obtained for upstream flows. Then, we select the next inner shaper configuration Sh_{inner} in the sorted list $List^{k-1}_{Sh_{inner}}$ and we come back to step (a) by considering $List^{k-1}_{Sh_{inner}} = Sh_{inner}$ for the next selected message in S_{down}.

C. Numerical Results

To illustrate the heuristic efficiency to find the optimal HTS structure which respects flows temporal constraints, we consider the same example described in Tables I and II. The optimal HTS structure consists of 7 inner shapers having the same period equal to 4ms and the associated downstream flows subsets are as following:
$Sh_1 : \{m^1, m^2\}$, $Sh_2 : \{m^3, m^4\}$,
$Sh_3 : \{m^5, m^6\}$, $Sh_4 : \{m^7, m^8\}$,
$Sh_5 : \{m^9, m^{10}, m^{11}, m^{12}\}$, $Sh_6 : \{m^{13}, m^{14}, m^{15}, m^{16}\}$,
$Sh_7 : \{m^{17}, m^{18}, m^{19}, m^{20}, m^{21}, m^{22}, m^{23}, m^{24}\}$.

This HTS configuration guarantees the schedulability of downstream flows as shown in Figure 12, unlike the one considered in the previous section. Furthermore, it reduces bandwidth utilization in the AFDX when using the optimized RDC device implementing frame packing and HTS algorithm, compared to the optimized RDC device implementingonly frame packing as illustrated in Table IV. However, as it can be noticed, there is a small difference in terms of bandwidth utilization in the AFDX compared to the results obtained in Table III due to the schedulability constraints.

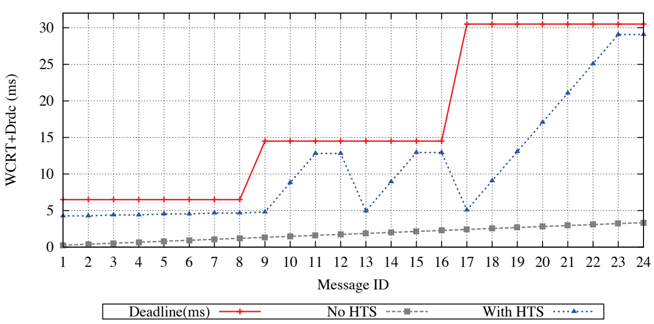

Fig. 12. *WCRT of downstream flows on CAN*

VI. Avionic Case Study

In this section, we validate the efficiency of the optimized RDC device implementing frame packing and HTS algorithm to reduce bandwidth utilization in the AFDX while guaranteeing the upstream and downstream schedulability for a realistic avionic case study.

TABLE IV
BANDWIDTH UTILIZATION IN THE AFDX AND SCHEDULABILITY TEST

Configuration	Bandwidth (in Mbps)	Schedulability
MSP + NO HTS	1.05	OK
MSP + HTS (heuristic)	0.75	OK

A. Description

Our case study is a representative avionic communication architecture based on a backbone network AFDX interconnected to one I/O CAN data bus via the optimized RDC device, as shown in Figure 13. The maximum delay bound on AFDX network considered herein is equal to 1.5ms and the CAN load is varied from 6% to 54%. Upstream and downstream flows are randomly generated with a payload of 8bytes and periods ranging in $\{8, 16, 32, 64, 128\}ms$. The generated CAN load due to Upstream flows is equal to the one due to downstream flows. For example, when we consider a CAN load of 6% this means that we have 3% due to upstream flows and 3% due to downstream flows.

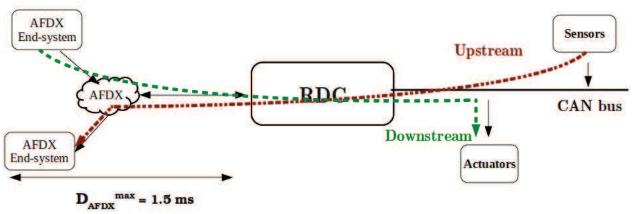

Fig. 13. *Avionic case study*

B. Performance Evaluation

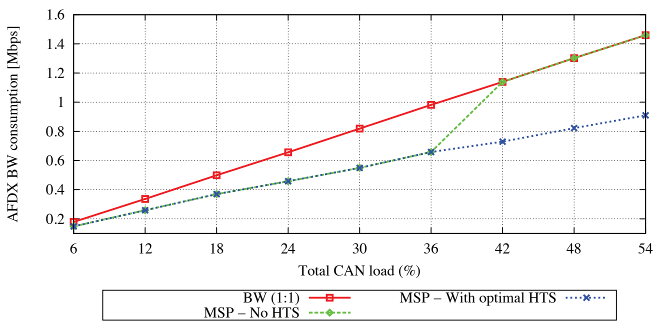

Fig. 14. *Bandwidth Utilization in the AFDX with optimal HTS structure*

Results illustrated in Figure 14 show the bandwidth utilization in the AFDX induced by upstream flows when using: a basic RDC device with (1:1) strategy, an optimized RDC device implementing only MSP strategy and an optimized RDC device implementing MSP strategy and an optimal HTS structure, i.e., the one obtained with the heuristic described in Section V-B.

As can be noticed, under low CAN loads (up to 30%), the two optimized RDC device configurations induce the same bandwidth utilization in the AFDX which is lower than the utilization obtained with the basic RDC device. These results validate the previous results proved in [3] where the MSP strategy offers a significant amelioration of bandwidth utilization in the AFDX, compared to the (1:1) strategy (up to 30% of reduction). However, in this case the HTS algorithm is useless for improving performances.

Under high CAN load (more than 36% and up to 54%), the optimized RDC device implementing MSP strategy and an optimal HTS structure shows significant enhancements in terms of bandwidth utilization in the AFDX, compared to the two first RDC device configurations. For example, under CAN load equal to 54%, we have a reduction of 50%. This fact validates the efficiency of HTS algorithm to guarantee a bandwidth isolation between upstream and downstream flows on CAN, and consequently minimizing WCRTs of upstream flows on CAN. This reduction is in favor of performing MSP strategy within RDC device and consequently reducing the bandwidth utilization in the AFDX.

VII. Conclusion

Since resource saving is inherently important to guarantee an easy incremental design process for avionics applications, the design of an optimized RDC device is proposed to interconnect an AFDX backbone to I/O CAN buses.

The optimized RDC device consists of: (i) accurate frame packing unit to manage upstream flows to reduce communication overheads, and consequently to minimize bandwidth utilization in the AFDX; (ii) Hierarchical Traffic Shaping (HTS) algorithm to control downstream flows, and consequently to minimize interferences and to guarantee bandwidth isolation on CAN bus; (iii) support of multiple I/O networks using partitioning to provide the required safety level, i.e., ensuring that an errant I/O network running in one partition will not affect the others.

First, modeling such RDC devices and timing analysis of upstream and downstream flows integrating the effects of frame packing and HTS were detailed. Then, the tuning process of HTS parameters, based on a heuristic approach, to minimize as much as possible bandwidth utilization in the AFDX and to guarantee at the same time upstream and downstream flows requirements was proposed. Finally, the validation of the optimized RDC device was conducted and the performance analysis highlighted its capabilities to improve resource savings for avionic networks, and particularly the use of the HTS algorithm.

The next step in our work consists in analyzing the adaptability of the proposed concepts to the specificities of other I/O data buses like MIL-STD-1553 [17] and TTP/C [18].

References

[1] A. E. E. Committee, "Remote Data Concentrator (RDC) Generic Description." Annapolis, Maryland: Aeronautical Radio, 1999.

[2] H. Ayed, A. Mifdaoui, and C. Fraboul, "Frame Packing Strategy within Gateways for Multi-cluster Avionics Embedded Networks," *ETFA*, 2012.

[3] ——, "Interconnection optimization for multi-cluster avionics networks," *ECRTS*, 2013.

[4] S. Zeng and N. Uzun, "A hierarchical traffic shaper for packet switches," *Global Telecommunications Conference, GLOBECOM '99*, vol. 2, no. 1, pp. 239–272, 1999.

[5] J. S. Turner, "New Directions in Communications (or Which Way to the Information Age?)," *IEEE Communications Magazine*, vol. 25, no. 10, pp. 8–15, 86.

[6] T. Nolte, M. Sjodin, and H. Hansson, "Server-based scheduling of the CAN bus," *Emerging Technologies and Factory AutomationProceedings. ETFA '03.*, vol. 1, 2003.

[7] Z. Iqbal, L. Almeida, R. Marau, M. Behnam, and T. Nolte, "Implementing hierarchical scheduling on cots ethernet switches using a master/slave approach," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'2012) Proceedings*, Karlsruhe, Germany, June 2012, pp. 76–84.

[8] E. WANDELER, A. MAXIAGUINE, and L. Thiele, "On the Use of Greedy Shapers in Real-Time Embedded Systems," *Embedded Computing Systems*, vol. 11, no. 1, pp. 239–272, 2012.

[9] R. Davis and N. Navet, "Traffic Shaping to Reduce Jitter in Controller Area Network (CAN)," *24th Euromicro Conference on Real-Time Systems*, vol. 9, no. 3, pp. 37–40, 2012.

[10] A. E. E. Committee, "Aircraft Data Network Part 7, AFDX NETWORK, ARINC Specification 664." Annapolis, Maryland: Aeronautical Radio, 2002.

[11] R. B. GmbH, "CAN specification Version 2,0," 1991.

[12] R. Marau, P. Pedreiras, and L. Almeida, "Enhancing Real-time communication over COTS Ethernet switches." Proceedings of the WFCS, 2006.

[13] A. A. Sheikh, O. Brun, M. Chramy, and P. Hladik, "Optimal Design of Virtual Links in AFDX Networks," *Laas*, 2012.

[14] R. Davis, A. Burns, R. Bril, and J. Lukkien, "CAN schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[15] J. Grieu, "Analyse et valuation de techniques de commutation ethernet pour l'interconnexion de systemes avioniques," Ph.D. dissertation, INP, Toulouse, 2004.

[16] K. Tindell, A. Burns, and A. J. Wellings, "Calculating Controller Area Network (CAN) message response times," *Control Engineering Practice*, vol. 3, August 1995.

[17] C. E. Incorporated, "MIL-STD-1553 Designer guide," 1982. [Online]. Available: http://www.condoreng.com/support/downloads/tutorials/MIL-STD-1553Tutorial

[18] H. Kopetz, "Event-Triggered Versus Time-Triggered Real-Time Systems ." Proceedings of the International Workshop on Operating Systems of the 90s and Beyond, 1991.