Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach

Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul

Abstract—**Avionics Full Duplex Switched Ethernet (AFDX) standardized as ARINC 664 is a major upgrade for avionics systems. The mandatory certification implies a worst-case delay analysis of all the flows transmitted on the AFDX network. Up to now, this analysis is done thanks to a tool based on a Network Calculus approach. The more recent Trajectory approach has been proposed for the computation of worst-case response time in distributed systems. This paper shows how the worst-case delay analysis of the AFDX can be improved using an optimized Trajectory approach. The Network Calculus and the Trajectory approaches are compared on a real avionics AFDX configuration. Moreover, an evaluation of an upper bound of the pessimism of each approach is proposed.**

Index Terms—**Aircraft, analytical model, Ethernet networks, real-time systems, upper bound.**

I. Introduction

DESIGNING and manufacturing new civilian aircraft has lead to an increase of the number of embedded systems and functions. In order to preserve performance and maintainability, the concept of common resource sharing has been introduced at platform level with the Integrated Modular Avionics (IMA) concept [1] and at network level with communication multiplexing.

The traditional mono-emitter ARINC 429 bus [2] with limited bandwidth cannot cope with new communication needs in terms of weight and complexity because of the number of needed buses. Military communication network uses master/slave technologies, such as MIL STD 1553B [11]. Such technologies are not adapted to civilian aircraft network, where the definition of a master would lead to tricky certification problems. The AFDX [3] brings an answer by multiplexing huge amount of communication flows over a full duplex switched Ethernet network. It has become the reference communication technology in the context of civilian avionics and provides a backbone network for the avionics platform.

Manuscript received October 30, 2009; revised April 01, 2010 and May 07, 2010; accepted June 17, 2010. Date of publication July 26, 2010; date of current version November 05, 2010. Paper no. TII-09-10-0281.

H. Bauer is with the Airbus France, 31300 Toulouse, France and also with the Université de Toulouse-IRIT-ENSEEIHT-INPT-2, rue Camichel 31000 Toulouse, France (e-mail: Henri.Bauer@enseeiht.fr).

J.-L. Scharbarg and C. Fraboul are with the Université de Toulouse-IRIT-ENSEEIHT-INPT-2, rue Camichel 31000 Toulouse, France (e-mail: Jean-Luc.Scharbarg@enseeiht.fr; Christian.Fraboul@enseeiht.fr).

Digital Object Identifier 10.1109/TII.2010.2055877

Full duplex switched Ethernet eliminates the inherent indeterminism of vintage (CSMA-CD) Ethernet. Nevertheless, it shifts the indeterminism problem to the switch level where various flows can enter in competition for sharing resources of a given switch (FIFO buffering of output ports).

Main AFDX specific assumptions deal with the static definition of avionics flows which are described as multicast links. All the flows are asynchronous, but have to respect a bandwidth envelope (burst and rate) at network ingress point. Each flow is statically mapped on the network of interconnected AFDX switches. These specific assumptions allow the analysis of the end-to-end delay of each flow of a given avionics configuration mapped on a given network of interconnected AFDX switches.

For a given flow, the end-to-end communication delay of a packet is the sum of transmission delays on links and latencies in switches. As the links are full duplex there is no packet collision on links [12]. The transmission delay only depends on the transmission rate (typically 100 Mb/s) and on the packet length (less or equal to the 1518 bytes Ethernet maximal packet size). On the other hand, the latency in switches is highly variable because of the confluence of asynchronous flows, which compete on each switch output port (according to a switch servicing policy).

Many work has been devoted to the analysis of end-to-end delays on an AFDX network. A simulation approach has been proposed in [21] for the computation of the distribution of the delay of a given flow. A probabilistic analysis of the network, using a stochastic Network Calculus approach, has been proposed in [21]. These two approaches do not cope with the worst-case analysis problem considered in this paper.

For certification reasons, a first tool has been proposed for the computation of an upper bound for the end-to-end delay of each flow. It is based on the Network Calculus theory. This approach models the traffic on the AFDX network as a set of sporadic flows with no further assumptions concerning the arrival time of packets. The input flows and the output ports are respectively modeled with traffic envelopes and service curves. Since these envelopes and curves are pessimistic, the obtained upper bounds are pessimistic. In fact, it has been proved that the obtained upper bounds can be reached only in the case of a mono-switch network. The Network Calculus approach has been improved in the context of AFDX by adding a grouping technique (flows sharing a common link are serialized and cannot arrive at the same time on a switch) [8], [10]. But, up to now, the pessimism of the obtained upper bounds was difficult to quantify, since the exact upper bounds are not known.

The model-checking approach presented in [7] computes the exact worst-case delay of each flow. Unfortunately, it cannot

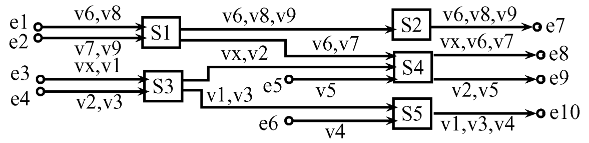


Fig. 1. An illustrative AFDX configuration.

cope with real AFDX configurations, due to the combinatorial explosion problem for large configurations. Nevertheless, it is used in this paper as a reference for exact worst-case computation on illustrative small configurations.

Another approach [4], [5] is based on the Trajectory concept [16]. It identifies for a packet m the busy periods and the packets impacting its end-to-end delay on all the nodes visited by m. Thus, it allows a worst-case delay computation.

A first contribution of this paper deals with the proof of how the Trajectory approach can be optimized by introducing the serialization of flows (similar to the grouping technique proposed in the Network Calculus context).

The second contribution of this paper is the comparison of the upper bounds obtained by Network Calculus and Trajectory approaches on a real avionics configuration.

Moreover, we propose in this paper a heuristic to generate an unfavorable reachable scenario (as close as possible from the exact worst-case). It gives a sure lower bound on the exact worst-case delay. The difference between this sure lower bound and the upper bounds computed by Network Calculus and Trajectory approaches gives an upper bound on the pessimism of each approach. This evaluation of the pessimism of Network Calculus and Trajectory approaches, as well as the explanation of the remaining pessimism, constitute the third contribution of this paper.

This paper is organized as follows. Section II shortly introduces the AFDX context. Section III summarizes the existing Network Calculus approach. Section IV presents the more recent Trajectory approach and its improvement in the context of AFDX. Section V compares the two approaches on a real avionics AFDX configuration and gives an evaluation of the pessimism of each approach. Section VI concludes and indicates directions for future research.

II. The AFDX Network Context

The AFDX is a switched Ethernet network taking into account avionics constraints. An illustrative example is depicted in Fig. 1. It is composed of five interconnected switches $S1$ to $S5$. Each switch has no input buffers on input ports and one FIFO buffer for each output port. The inputs and outputs of the network are called *End Systems* ($e1$ to $e10$ in Fig. 1). Each end system is connected to exactly one switch port and each switch port is connected to at most one end system. Links between switches are all full duplex.

The end-to-end traffic characterization is made by the definition of *Virtual Links*. As standardized by ARINC-664, Virtual Link (VL) is a concept of virtual communication channel. Thus, it is possible to statically define all the flows (VL) which enter the network [3].

End systems exchange packets through VLs. Switching a packet from a transmitting to a receiving end system is based on VL. The VL defines a logical unidirectional connection from one source end system to one or more destination end systems. Coming back to the example in Fig. 1, vx is a unicast VL with path $\{e3 - S3 - S4 - e8\}$, while $v6$ is a multicast VL with paths $\{e1 - S1 - S2 - e7\}$ and $\{e1 - S1 - S4 - e8\}$.

The routing is statically defined. Only one end system within the avionics network can be the source of one VL, (i.e., mono transmitter assumption). A VL definition also includes the Bandwidth Allocation Gap (BAG), the minimum and the maximum packet length ($s_{\min}$ and $s_{\max}$). BAG is the minimum delay between two consecutive packets of the associated VL (which actually defines a VL as a sporadic flow).

VL parameters $(\mathrm{BAG}, s_{\max})$ compliance is ensured by a shaping unit at end system level and a traffic policing unit at each switch entry port (specificity of AFDX switches, compared to standard Ethernet switches). The delay incurred by the switching fabric is upper bounded by a constant value, i.e., 16 μs.

All these constraints that the AFDX model adds to the vintage Ethernet enables a precise analysis of the network, especially the computation of an upper bound for the end-to-end delay of each flow and the dimensioning of output buffers so that no packet is lost. For the certification of the A380, this worst-case analysis has been conducted by a Network Calculus approach, which is summarized in the next section.

III. AFDX Worst-Case Delay Analysis With Network Calculus

Network Calculus [6], [14] has been proposed for the computation of an upper bound for the delay and the jitter of a flow transmitted over a network. It can be used on a set of sporadic flows with no assumption concerning the arrival time of packets.

The basic application of Network Calculus to the AFDX is presented in Section III-A. The improvement of this basic approach in the context of AFDX is described in Section III-B.

A. The Basic Network Calculus Approach for the AFDX

Network Calculus is mathematically based on the $(\min, +)$ dioid, for which the convolution $\otimes$ and the deconvolution $\oslash$ are defined as follows:

$$(f \otimes g)(t) = \inf_{0 \le u \le t} (f(t-u) + g(u))$$
$$(f \oslash g)(t) = \sup_{0 \le u} (f(t+u) - g(u)).$$

A flow is represented by its cumulative function R, where $R(t)$ is the total number of bits sent up to time t. A flow R is said to have a function α as *arrival curve* if and only if $R \le R \otimes \alpha$.

A server has a *service curve* β if and only if for all flow processed by the server, it holds that: $R' \ge R \otimes \beta$, where R is the input flow and R' is the output flow. In that case, $\alpha' = \alpha \oslash \beta$ is an arrival curve for R'.

The delay experienced by a flow R constrained by a service curve α in a node offering a service curve β is bounded by the maximum horizontal difference between curves α and β. This difference is formally defined by

$$h(\alpha, \beta) = \sup_{s \ge 0} (\inf\{\tau \ge 0 | \alpha(s) \le \beta(s+\tau)\}).$$

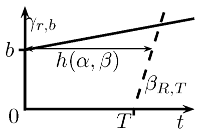


Fig. 2. The maximum delay $h(\alpha, \beta)$.

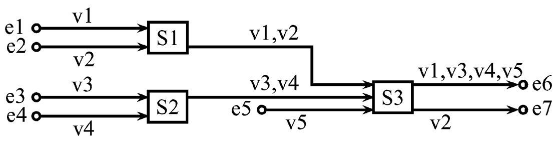


Fig. 3. Illustrative AFDX configuration.

Each VL of an AFDX network (a flow) is modeled by a *leaky bucket* $\gamma_{r,b}$, with $b = s_{\max}(v)$ and $r = s_{\max}(v)/BAG(v)$. The burst b is the capacity of the bucket and the rate r is the leak rate.

Each output port of an AFDX switch offers a service curve $\beta_{R,T} = R[t-T]^+$. T is the maximal technological latency of the switch, i.e., 16 μs. R is the servicing rate (100 Mb/s in our context) and $[x]^+ = max(0, x)$. Thus, in the context of this paper, the service curve which is offered by each output port is $\beta_{100,16}$.

Fig. 2 illustrates the delay $h(\alpha, \beta)$ experienced by a flow R constrained by a service curve $\alpha = \gamma_{r,b}$ in an output port of an AFDX switch, provided R is the only flow crossing this output port. The VLs which compete for a given output port are merged into a single flow by summing their respective arrival curves.

The overall computation is illustrated in the small example in Fig. 3. The five VLs $v1 \ldots v5$ which are transmitted on this AFDX network all have the same BAG (4000 μs) and the same $s_{\max}$ (4000 bits).

Let us consider the VL $v1$. Its arrival curve in $S1$ is $\alpha_1 = \gamma_{1,4000}$, since $r = s_{\max}(v1)/\mathrm{BAG}(v1) = 4000/4000 = 1$ Mb/s and $b = s_{\max}(v1) = 4000$ bits. $v1$ shares the output port of $S1$ with $v2$, whose arrival curve in $S1$ is $\alpha_2 = \gamma_{1,4000}$. Consequently, the overall arrival curve for the output port of switch $S1$ is $\alpha_1 + \alpha_2 = \gamma_{2,8000}$. As previously mentioned, the service curve of this port is $\beta_{100,16}$. Thus, the delay in this output port is bounded by the maximum horizontal difference between $\gamma_{2,8000}$ and $\beta_{100,16}$, which is 96 μs, as depicted in Fig. 4. It includes the technological latency (16 μs), the transmission time (40 μs) and the maximum waiting time in the output buffer (40 μs), since each packet of VL $v1$ or $v2$ can be delayed by at most one packet of the other VL.

Then, the computation proceeds to switch $S3$ and it needs the input curves of $v1$, $v3$, $v4$, and $v5$ in $S3$. These input curves are the output curves of the VLs in their previous crossed output port, i.e., $S1$ for $v1$, $S2$ for $v3$ and $v4$, $e5$ for $v5$. In the general case, the output curve α' of the flow is given by: $\alpha' = \alpha \oslash \delta_{\text{jitter}}$. α is the input curve of the flow in the port, jitter is the maximum jitter encountered by the flow in the port and δ_{jitter} is a guaranteed delay service curve ($\delta_d(t) = 0$ if $t \leq d$, ∞ otherwise). Graphically, it comes down to shift the arrival curve α to the left by the duration of the jitter. The maximum jitter in an output port clearly corresponds to the maximum waiting time in the corresponding buffer. Coming back to $v1$, its maximum jitter when leaving $S1$ is 40 μs, i.e., the maximum waiting time

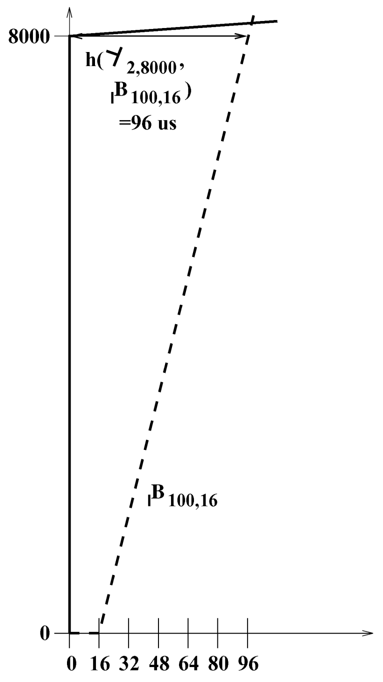


Fig. 4. Output port of switch $S1$.

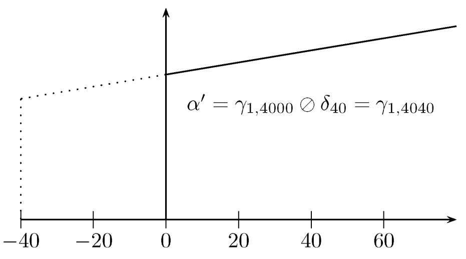


Fig. 5. Output curve after switch $S1$ or $S2$.

in the output buffer of $S1$. Then, the input curve α'_1 of $v1$ at $S3$ is obtained by shifting α_1 by 40 μs to the left: $\alpha'_1 = \alpha_1 \oslash \delta_{40} = \gamma_{1,4040}$. It is illustrated in Fig. 5.

Clearly, the input curves of $v3$, $v4$ and $v5$ at $S3$ are, respectively, $\gamma_{1,4040}$, $\gamma_{1,4040}$ and $\gamma_{1,4000}$. They lead to an overall arrival curve $\gamma_{4,16120}$ in the output port of $S3$. Then, the maximum delay for $v1$ in $S3$ is 177.2 μs, leading to a maximum end-to-end delay of 313.2 μs. It is composed of the transmission delay from $e1$ to $S1$ (40 μs) and the maximum delay computed for $S1$ and $S3$, i.e., 96 μs and 177.2 μs.

Column BNC in Table I summarizes the upper bounds computed with this method for the five VLs of Fig. 3. The exact worst-case end-to-end delay of each VL has been computed using the model checking approach presented in [7].

Results in Table I show that, on this small configuration, the basic Calculus approach is pessimistic (more than 40 μs for nearly all the VLs). The next section presents an improvement of the basic Network Calculus approach in the context of AFDX.

B. Optimizing the Network Calculus Approach With Grouping

The pessimism observed in Table I is partly due to the fact that the basic Network Calculus approach does not take into account the property that packets of different flows sharing a

TABLE I
UPPER END-TO-END DELAYS IN μS

VL	EWC	BNC	NCG
$v1$	272	313.2	273.6
$v2$	192	192.4	192.4
$v3$	272	313.2	273.6
$v4$	272	313.2	273.6
$v5$	176	217.2	177.6

EWC: exact worst-case (model checking approach).
BNC: basic Network Calculus approach.
NCG: Network Calculus approach with grouping.

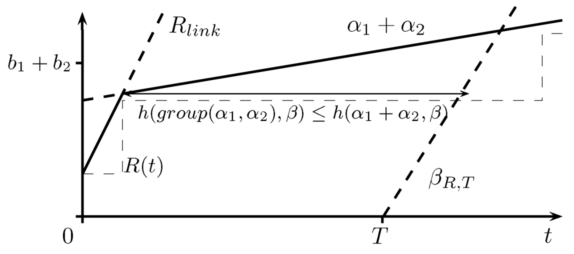


Fig. 6. Grouping flows reduces the maximum delay incurred in an output port.

link cannot be transmitted at the same time on this link (they are serialized). Consequently, the burst considered in the overall input curves of the basic Network Calculus approach can never happen, as soon as at least two flows share the same link. This problem is different from the classical "pay burst only once" case described in [14]. Indeed, the objective of "pay burst only once" is to aggregate successive switches in order to give an optimized aggregated service curve. The aggregation of nodes is not possible in our case, since flows can join and leave a path at any switch of the network.

In the example in Fig. 3, the input curve of the output port of $S3$ shared by $v1$, $v3$, $v4$, and $v5$ is $\gamma_{4,16120}$. The maximum burst (16120 bits) corresponds to the case where four packets—one for each VL—arrive at the same time in the output port. This is definitely impossible, since $v3$ and $v4$ share the same link. The grouping technique integrates this serialization. It proceeds in two steps. First, the overall arrival curve is computed for each link. It is the minimum between, on the one hand the sum of the arrival curves of all the flows sharing the considered link, on the other hand the curve bounding the burst to the maximum burst among the curves of the different flows sharing the link and the rate to the rate of the link. This first step is illustrated in Fig. 6 for a link shared by two flows with arrival curves γ_{r_1,b_1} and γ_{r_2,b_2}. In the second step, the curves obtained for the different links are added. The gain obtained with this technique is due to the reduction of the maximum burst.

Column NCG in Table I gives the upper bounds computed with this technique in the example in Fig. 3. Results are clearly improved, compared with the basic Network Calculus approach.

A more recent approach, based on trajectories, has been proposed for the worst-case delay analysis of distributed systems. The next section shows how this approach can be applied and optimized in the context of the AFDX. The main goal is to compare this new approach with the Network Calculus one.

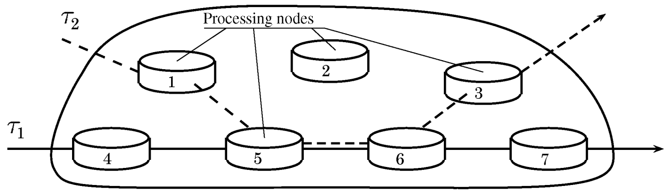


Fig. 7. A distributed system.

IV. AFDX WORST-CASE DELAY ANALYSIS WITH THE TRAJECTORY APPROACH

The Trajectory approach [15], [16], [18] has been developed to get deterministic upper bounds on end-to-end response time in distributed systems. This approach considers a set of sporadic flows with no assumption concerning the arrival time of packets. The principle of the application of the Trajectory approach to the AFDX has been presented in [4]. The improvement of the approach has been proposed in [5]. Main features of the Trajectory approach applied to AFDX are summarized and illustrated in Sections IV-A and IV-B. The proof of the optimization of the Trajectory approach computation is presented in Section IV-C.

A. The Main Features of the Trajectory Approach

The approach developed for the analysis of the AFDX considers the results from [16]. The general architecture of the distributed system considered in [16] is depicted in Fig. 7.

Such a system is composed of a set of interconnected processing nodes (seven in Fig. 7). Each flow crossing this system follows a static path which is an ordered sequence of nodes. In the example of Fig. 7, there are two flows τ_1 and τ_2. τ_1 follows the path $\mathcal{P}_1 = \{4, 5, 6, 7\}$. Node 4 is the ingress node of τ_1 in the system. The Trajectory approach assumes, with regards to any flow τ_i following path $\mathcal{P}_i$, that any flow τ_j following path $\mathcal{P}_j$, with $\mathcal{P}_j \neq \mathcal{P}_i$ and $\mathcal{P}_j \cap \mathcal{P}_i \neq \emptyset$, never visits a node of path $\mathcal{P}_i$ after having left this path. In the example of Fig. 7, $\mathcal{P}_2 = \{1, 5, 6, 3\}$ and $\mathcal{P}_1 \cap \mathcal{P}_2 = \{5, 6\}$.

Flows are scheduled with a FIFO algorithm in every visited node (non preemptive policy). Each flow τ_i has a minimum interarrival time between two consecutive packets at ingress node, denoted T_i, a maximum release jitter at the ingress node denoted J_i, an end-to-end deadline D_i that is the maximum end-to-end response time acceptable and a maximum processing time C_i^h on each node N_h, with $N_h \in \mathcal{P}_i$.

The transmission time of any packet on any link between nodes has known lower and upper bounds $L_{\min}$ and $L_{\max}$ and there are neither collisions nor packet losses on links.

The end-to-end response time of a packet is the sum of the times spent in each crossed node and the transmission delays on links. The transmission delays on links are upper bounded by $L_{\max}$. The time spent by a packet m in a node N_h depends on the pending packets in N_h at the arrival time of m in N_h (all these pending packets have a higher priority than m considering FIFO scheduling and, thus, will be processed before m). The problem is then to upper bound the overall time spent in the visited nodes.

The solution proposed by the Trajectory approach is based on the busy period concept. A busy period of level $\mathcal{L}$ is an interval $[t, t')$ such that t and t' are both idle times of level $\mathcal{L}$ and there is

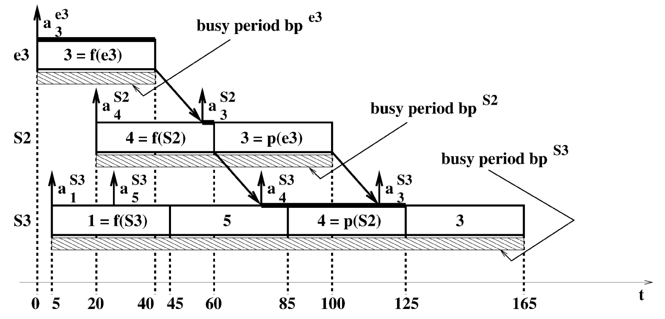


Fig. 8. An arbitrary scheduling of packets.

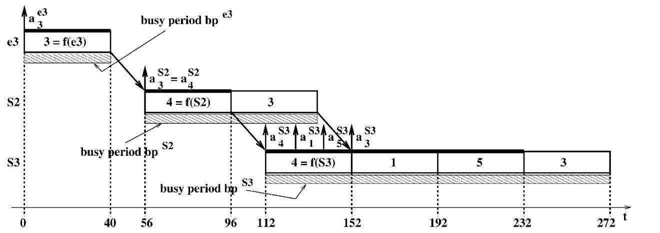


Fig. 9. Latest starting time of packet 3.

no idle time of level $\mathcal{L}$ in (t, t'). An idle time t of level $\mathcal{L}$ is a time such as all packets with priority greater than or equal to $\mathcal{L}$ generated before t have been processed at time t. With FIFO scheduling, no packet from the busy period of level corresponding to the priority of m could have arrived after m on the considered node.

The Trajectory approach considers a packet m from flow τ_i generated at time t. It identifies the busy period and the packets impacting its end-to-end delay on all the nodes visited by m (starting from the last visited node backward to the ingress node). This decomposition enables the computation of the latest starting time of m on its last node. This starting time can be computed recursively and leads to the worst case end-to-end response time of the flow τ_i. This computation will be illustrated in the context of AFDX.

The elements of the system considered in the Trajectory approach are instantiated in the following way in the context of AFDX:

- each node of the system corresponds to an AFDX switch output port, including the output link;
- each link of the system corresponds to the switching fabric;
- each flow corresponds to a VL path.

The assumptions of the Trajectory approach are verified by the AFDX (see Section II). Indeed, switch output ports implement FIFO service discipline. The switching fabric delay is upper bounded by a constant value (16 μs), thus $L = L_{\min} = L_{\max} =$ 16 μs. There are no collisions nor packet loss on AFDX networks. The routing of the VLs is statically defined.

VL parameters match the definition of sporadic flows in the following manner: $T_i = BAG$, $C_i^h = s_{\max}/R$, $J_i = 0$. Since all the AFDX ports work at the same rate $R = 100$ Mb/s, $C_i^h = C_i = s_{\max}/R$ for every node h in the network.

B. Illustration on a Sample AFDX Configuration

Let us consider the sample AFDX configuration depicted in Fig. 3. Fig. 8 shows an arbitrary scheduling of the packets, which are identified by their VL numbers (e.g., packet 3 is a packet from VL $v3$). The scheduling in Fig. 8 focuses on packet 3. The arrival time of a packet m in a node N_h is denoted $a_m^{N_h}$. Time origin is arbitrarily chosen as the arrival time of packet 3 in node $e3$. The processing time of a packet in a node is 40 μs. It corresponds to the transmission time of the packet on a link. The delay between the end of the processing of a packet by a node and its arrival in the next node corresponds to the 16 μs switch factory delay. In each node, the packets are processed with respect to the first come, first served policy. Consequently, packet 3 is delayed by packet 4 in $S2$. In node $S3$, packet 5 is delayed by packet 1 and delays packet 4, which delays packet 3.

Packet 3 from VL $v3$ crosses three busy periods (bp^{e3}, bp^{S2}, and bp^{S3}) on its trajectory, corresponding to the three nodes $N_1 = e3$, $N_2 = S2$, and $N_3 = S3$. Let $f(N_i)$ be the first packet which is processed in the busy period bp^{N_i} during which packet 3 is processed. Considering the scheduling in Fig. 8, we have $f(e3) = 3$, $f(S2) = 4$, and $f(S3) = 1$. As flows do not necessarily follow the same path in the network, it is possible that packet $f(N_i)$ does not come from the same previous node N_{i-1} as packet 3. This case occurs in node $S2$, where packet 4 comes from node $e4$. It also occurs in node $S3$, where packet 1 comes from node $S1$. Therefore, $p(N_{i-1})$ is defined as the first packet which is processed in bp^{N_i} and comes from node N_{i-1}. Considering the scheduling in Fig. 8, we have $p(e3) = 3$ and $p(S2) = 4$.

The starting time of packet 3 in node $S3$ is obtained by adding parts of the three busy periods bp^{e3}, bp^{S2}, and bp^{S3} to the delays between the nodes, i.e., 2×16 μs. From [16], the part of the busy period bp^{N_i} which has to be added is the processing time of packets between $f(N_i)$ and $p(N_i)$ minus the time elapsed between the arrivals of $f(N_i)$ and $p(N_{i-1})$, i.e., $(a_{p(N_{i-1})}^{N_i} - a_{f(N_i)}^{N_i})$. In the example in Fig. 8, the parts which have to be considered are the transmission of packet 3 in node $e3$, the time elapsed between the arrival of packet 3 and the end of processing of packet 4 in node $S2$, the time elapsed between the arrival of packet 4 and the end of processing of packet 5 in node $S3$. These parts are shown by thick lines on top of the packets in Fig. 8. Their durations are 40 μs for bp^{e3}, 4 μs for bp^{S2} and 49 μs for bp^{S3}. Thus, the starting time of packet 3 in node $S3$ in the example in Fig. 8 is

$$40 + 4 + 49 + (2 \times 16) = 125\ \mu\text{s}.$$

It has been shown [16] that the latest starting time of a packet m in its last node is reached when $(a_{p(N_{i-1})}^{N_i} - a_{f(N_i)}^{N_i}) = 0$ for every node N_i on the path of m. It comes to postpone the arrival time of every packet joining the path of m in the node N_i in order to maximize the waiting time of m in N_i.

The result of this postponing in the example in Fig. 8 is illustrated in Fig. 9. The arrival time of packet 4 at node $S2$ is postponed to the arrival time of packet 3 at node $S2$. In node $S3$, packets 1 and 5 have been postponed in order to arrive between packets 4 and 3.

Then, the worst case end-to-end delay of a packet is obtained by adding its latest starting time on its last visited node and its processing time in this last node. For packet 3 in Fig. 9, this

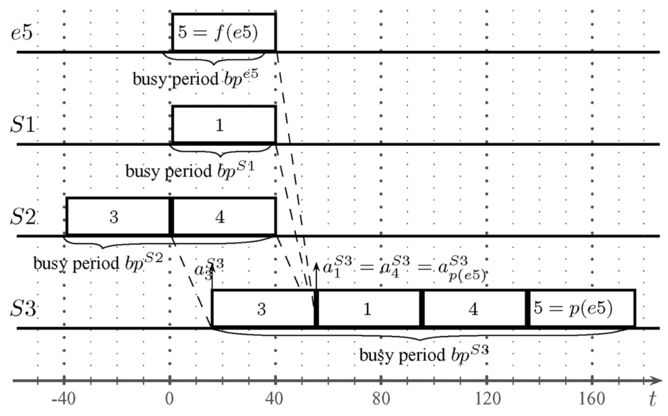


Fig. 10. Latest starting time of VL $v5$.

worst case end-to-end delay is $232 + 40 = 272\ \mu$s. More precisely, this delay includes the transmission times of packet 3 on node $e3$, packet 4 on node $S2$ and packets 4, 1, 5, and 3 on node $S3$. In this example, it can be seen that packets 3 and 4 are counted twice. Actually, it has been shown [16] that exactly one packet has to be counted twice in each node, except the slowest one. In the context of the AFDX, all the nodes work at the same speed. Thus, the slowest node is arbitrarily chosen as the last one. In the example in Fig. 9, packets 3 and 4 are, respectively, counted twice in nodes $e3$ and $S2$. Packet 3 is the longest one transmitted in nodes $e3$ and $S2$, while packet 4 is the longest one transmitted in node $S2$ and $S3$.

In the context of an AFDX network, it is not always possible to find a scheduling which cancels the term $(a_{p(N_{i-1})}^{N_i} - a_{f(N_i)}^{N_i})$ for every node N_i, as proposed in [16]. Let us consider VL $v5$ in the example depicted in Fig. 3. bp^{S3} is the busy period of level corresponding to the priority of packet 5. In order to maximize the delay of packet 5 in bp^{S3}, the arrival time of packets 3 and 4 in $S3$ have to be as large as possible, but not larger than the arrival time of packet 5 in node $S3$, because of the FIFO scheduling policy

$$a_3^{S3} \leq a_5^{S3} \ and \ a_4^{S3} \leq a_5^{S3}. \tag{1}$$

Since the two packets come from the same link, they are already serialized

$$\left|a_3^{S3} - a_4^{S3}\right| \geq C = 40\ \mu\text{s}. \tag{2}$$

Without loss of generality, let us consider that packet 3 arrives before packet 4. From (1), we have

$$a_4^{S3} = a_5^{S3}. \tag{3}$$

From (2) and (3), we have

$$a_3^{S3} = a_5^{S3} - 40\ \mu\text{s}. \tag{4}$$

The resulting worst-case scheduling is depicted in Fig. 10. $p(e5)$ is packet 5 and f(S3) is packet 3. From (4), we have $(a_{p(e5)}^{S3} - a_{f(S3)}^{S3}) \geq 40\ \mu$s for any possible scheduling. Thus, considering that $(a_{p(N_{i-1})}^{N_i} - a_{f(N_i)}^{N_i}) = 0$ for every node N_i is a pessimistic assumption in the context of the AFDX.

The next section presents the implementation of the Trajectory approach.

C. Optimization of the Trajectory Approach Computation

The computation of the worst-case end-to-end delay a packet of a flow τ_i has been formalized in [16]. In our context, all the nodes work at the same rate and the jitter in each emitting node is null. Thus, the worst case end-to-end response time of any flow τ_i is bounded by

$$R_i = \max_{t \geq 0}\left(W_{i,t}^{\text{last}_i} + C_i - t\right)$$

last_i is the last visited node of flow τ_i and $W_{i,t}^{\text{last}_i}$ is a bound on the latest starting time of a packet m generated at time t on its last visited node. The definition of $W_{i,t}^{\text{last}_i}$ given in [16] becomes

$$W_{i,t}^{\text{last}_i} = \sum_{\substack{j\in[1,n]\\ j\neq i\\ \mathcal{P}_j\cap\mathcal{P}_i\neq\emptyset}} \left(1 + \left\lfloor \frac{t + A_{i,j}}{T_j} \right\rfloor\right) \cdot C_j \tag{5}$$

$$+ \left(1 + \left\lfloor \frac{t}{T_i} \right\rfloor\right) \cdot C_i \tag{6}$$

$$+ \sum_{h\in\mathcal{P}_i h\neq last_i} \left(\max_{\substack{j\in[1,n]\\ h\in\mathcal{P}_j}} \{C_j\} \right) \tag{7}$$

$$+ (|\mathcal{P}_i| - 1) \cdot L_{\max} \tag{8}$$

$$- \sum_{\substack{N_h\in\mathcal{P}_i\\ N_h\neq first_i}} (\Delta_{N_h}) \tag{9}$$

$$- C_i \tag{10}$$

where

$$\Delta_{N_h} = a_{p(N_{h-1})}^{N_h} - a_{f(N_h)}^{N_h}. \tag{11}$$

Term (5) corresponds to the processing time of packets from every flow τ_j crossing the flow τ_i and transmitted in the same busy period as m. $A_{i,j}$ integrates the maximum jitter of packets from τ_i and τ_j on their first shared output port.

Term (6) is the processing time, on one node, of the packets of the flow τ_i which are transmitted in the same busy period as m.

Term (7) is the processing time of the longest packet for each node of path $\mathcal{P}_i$, except the last one. It represents the packets which have to be counted twice, as explained before.

Term (8) corresponds to the sum of switching delay.

Term (9) sums for each node N_h in $\mathcal{P}_i$ the duration between the beginning of the busy period and the arrival of the first packet coming from the preceding node in $\mathcal{P}_i$, i.e., N_{h-1}. This term is null in the context of [16].

C_i is subtracted, because $W_{i,t}^{\text{last}_i}$ is the latest starting time and not the ending time of the packet from τ_i on its last node.

Solving $R_i = \max(W_{i,t}^{\text{last}_i} + C_i - t)$ comes to find the maximum vertical deviation between the function $t \mapsto W_{i,t}^{\text{last}_i} + C_i$ and the identity function $(t \mapsto t)$.

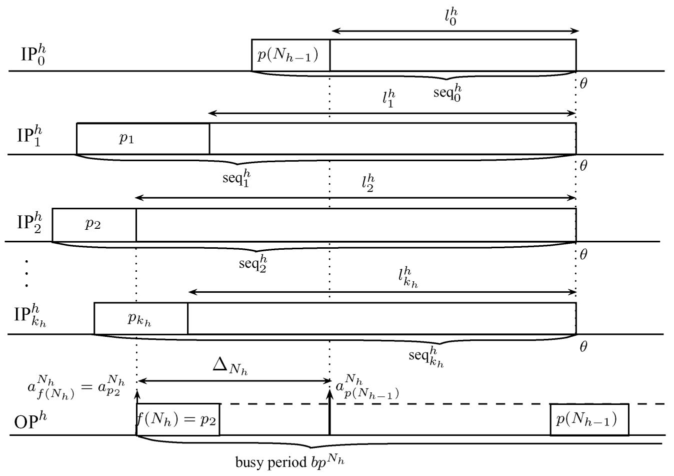


Fig. 11. Illustration of Δ_{N_h}.

This computation is illustrated on VL $v3$ of Fig. 3, considering that Term (9) is null

$$\begin{aligned}W_{3,t}^{S3} + C_3 = &\left(1 + \left\lfloor \frac{t + A_{3,1}}{T_1} \right\rfloor\right) \cdot C_1 \\ &+ \left(1 + \left\lfloor \frac{t + A_{3,4}}{T_4} \right\rfloor\right) \cdot C_4 \\ &+ \left(1 + \left\lfloor \frac{t + A_{3,5}}{T_5} \right\rfloor\right) \cdot C_5 \\ &+ \left(1 + \left\lfloor \frac{t}{T_3} \right\rfloor\right) \cdot C_3 + \max_{j \text{ with } e_3 \in \mathcal{P}_j} \{C_j\} \\ &+ \max_{j \text{ with } S_2 \in \mathcal{P}_j} \{C_j\} - C_3 + (|\mathcal{P}_3| - 1) \cdot L_{\max} \\ &+ C_3\end{aligned}$$

Consequently:

$$\begin{aligned}W_{3,t}^{S3} + C_3 = &\left(1 + \left\lfloor \frac{t + 40}{4000} \right\rfloor\right) \cdot 40 + \left(1 + \left\lfloor \frac{t + 40}{4000} \right\rfloor\right) \cdot 40 \\ &+ \left(1 + \left\lfloor \frac{t + 40}{4000} \right\rfloor\right) \cdot 40 + \left(1 + \left\lfloor \frac{t}{4000} \right\rfloor\right) \cdot 40 \\ &+ 40 + 40 - 40 + (3 - 1) \times 16 + 40 \\ = &\, 272 + 160 \left\lfloor \frac{t + 40}{4000} \right\rfloor\end{aligned}$$

The upper-bound on the end-to-end delay is reached for $t = 0$ and $R_5^{S3} = 272\ \mu$s.

The optimization of this computation in the context of the AFDX concerns Term (9). Indeed, it has been shown in Section IV-B that, for some VLs, there exists no scheduling leading to $\Delta_{N_h} = 0, \forall N_h \in \mathcal{P}_i$.

In the following, we describe the computation of a lower bound on $\Delta_{N_h} \forall N_h \in \mathcal{P}_i$ and we prove its correctness.

The value of Δ_{N_h} is illustrated in Fig. 11.

The packet m of flow τ_i under study is sent on the output link OP^h in a busy period bp^{N_h}. The packets which compose bp^{N_h} are determined thanks to terms (5) and (6). These packets are grouped by input link. IP_0^h is the input link of τ_i, while IP_x^h $(1 \leq x \leq k_h)$ are the other input links. Sequence seq_x^h $(0 \leq x \leq k_h)$ contains the packets of bp^{N_h} coming form IP_x^h.

TABLE II
UPPER END-TO-END DELAYS IN μS

VL	EWC	BNC	NCG	BT	OT
$v1$	272	313.2	273.6	312	272
$v2$	192	192.4	192.4	192	192
$v3$	272	313.2	273.6	272	272
$v4$	272	313.2	273.6	272	272
$v5$	176	217.2	177.6	216	176

EWC: exact worst-case (model checking approach).
BNC: basic Network Calculus approach.
NCG: Network Calculus approach with grouping.
BT: basic Trajectory approach.
OT: optimized Trajectory approach.

In order to maximize the delay of packet m in node N_h, each sequence seq_x^h $(1 \leq x \leq k_h)$ is postponed so that it finishes at the same time as sequence seq_0^h. This finishing time is denoted θ in Fig. 11. This construction is a generalization of the Trajectory approach presented in [16]: instead of postponing individually each packet, sequences of already serialized packets are postponed. As defined in (11), Δ_{N_h} is the delay between the earliest arrival of a packet of bp^{N_h} (i.e., the beginning of bp^{N_h}) and the arrival of the first packet of bp^{N_h} coming from IP_0^h. In Fig. 11, Δ_{N_h} is the difference between the arrival of $p2$ and the arrival of $p(N_{h-1})$.

The latest starting time of m in its last node is maximized when $\sum_{\substack{N_h \in \mathcal{P}_i \\ N_h \neq \text{first}_i}} (\Delta_{N_h})$ is minimized. It comes to determine the lower bound of each term Δ_{N_h} of the sum.

From (11), it is obvious that the minimum value of Δ_{N_h} is obtained by minimizing $a_{p(N_{h-1})}^{N_h}$ and maximizing $a_{f(N_h)}^{N_h}$.

Let us define $l_x^h (0 \leq x \leq k_h)$ as the duration of sequence seq_x^h without its first packet. Then, we have:

$$a_{f(N_h)}^{N_h} = \theta - \max_{1 \leq x \leq k_h} l_x^h \quad (12)$$

$$a_{p(N_{h-1})}^{N_h} = \theta - l_0^h. \quad (13)$$

Consequently, minimizing $a_{p(N_{h-1})}^{N_h}$ comes to maximize l_0^h. It is obtained when the smallest packet of sequence seq_0^h is transmitted at the beginning of seq_0^h.

Similarly, maximizing $a_{f(N_h)}^{N_h}$ comes to minimize each l_x^h for $1 \leq x \leq k_h$. It is obtained when the largest packet of sequence seq_x^h is transmitted at the beginning of seq_x^h, for $1 \leq x \leq k_h$.

To summarize, Δ_{N_h} is lower bounded by the maximum of 0 and

$$\max_{1 \leq x \leq k_h} \left(\min\left(l_x^h\right)\right) - \max\left(l_0^h\right). \quad (14)$$

D. Results on a Sample Configuration

The results for all the VLs of the configuration in Fig. 3 are presented in column BT in Table II. The results obtained with the Network Calculus approach are reminded in Table II for comparison purpose.

There are three VLs ($v2$, $v3$ and $v4$) for which the basic Trajectory approach gives the exact worst case. However, for VLs $v1$ and $v5$, the Trajectory approach introduces a 40 μs pessimism.

TABLE III
BAGS AND PACKET LENGTHS

BAG (ms)	Number of VL
2	18
4	498
8	1956
16	2203
32	3514
64	3842
128	3298

Packet length (bytes)	Number of VL
0-150	1811
151-300	5992
301-600	5426
601-900	937
901-1200	618
1201-1500	383
> 1500	162

TABLE IV
VL PATHS LENGTHS

Nb of crossed switches	Number of paths
1	4902
2	6621
3	3176
4	630

The next section presents a quantitative analysis of the Network Calculus and Trajectory approaches on a real avionics configuration.

V. COMPARATIVE EVALUATION OF THE TWO APPROACHES ON A REAL AVIONICS CONFIGURATION

The comparative analysis proceeds on a real avionics configuration. Worst-case end-to-end delays are computed with the different approaches. Then, the results are analyzed: the pessimism of the Network Calculus and the Trajectory approaches is upper bounded and some reasons for this pessimism are presented.

A. *Worst-Case End-to-End Delay on a Real Avionics Configuration*

The AFDX configuration considered in this evaluation is composed of two redundant networks. It includes 126 end systems, 16 switches, 1063 Virtual Links, and 15329 VL paths (due to VL multicast characteristics). The left part in Table III gives the dispatching of VLs among BAGs. It can be seen that BAGs are harmonic between 2 and 128. The right part in Table III gives the dispatching of VLs among packet lengths, considering the maximum length $s_{\max}$. The majority of VLs consider short packets. Table IV shows the number of VL paths per length (i.e., the number of crossed switches). The maximum path length is 4.

Both the Network Calculus and the Trajectory approaches have been implemented using Python programming language. Upper bounds of the end-to-end delays for each VL path of the industrial configuration have been computed with this tool. This computation takes less than two minutes for any approach on a Pentium 4 processor running at 2.8 GHz.

A comparison of the sure upper bounds obtained by the two approaches is depicted in Fig. 12.

For each path p_x of each VL, the upper bound nc_x computed by the Network Calculus approach is taken as the reference value and it is normalized to 100. The upper bound tr_x computed by the Trajectory approach is then normalized in a similar manner

$$Ntr_x = 100 + \left(\frac{tr_x - nc_x}{nc_x} \times 100\right).$$

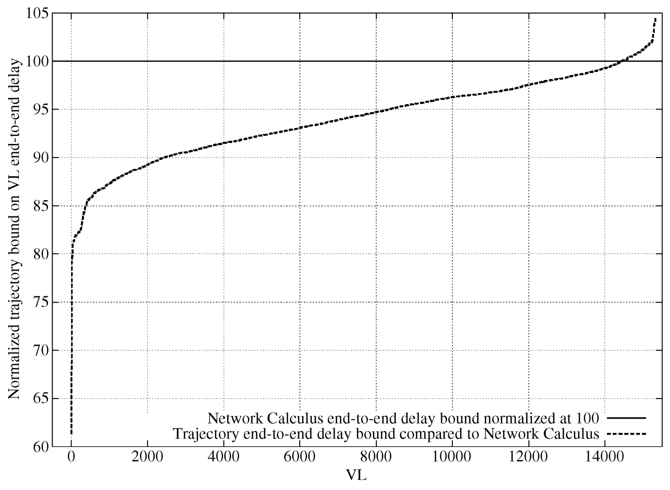


Fig. 12. Comparative results of the two approaches.

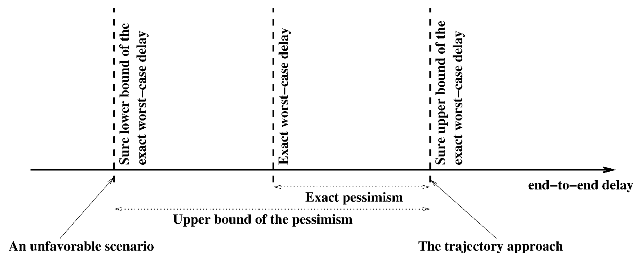

Sure lower bound of the exact worst-case delay
Exact worst-case delay
Sure upper bound of the exact worst-case delay
end-to-end delay
Exact pessimism
Upper bound of the pessimism
An unfavorable scenario
The trajectory approach

Fig. 13. Evaluation of the remaining pessimism.

The paths are sorted by increasing order of Ntr_x and the 15329 values are depicted in Fig. 12. Ntr_x values are between 60 and 105. It means that, depending on the path, the upper bound computed by the Trajectory approach can be 40% lower or 5% higher than the upper bound computed by the Network Calculus approach. The Trajectory approach gives a tighter upper bound for 14444 paths and a looser bound for the 885 remaining paths.

The results presented in Fig. 12 show that, on average, the Trajectory approach computes upper bounds which are tighter than the upper bounds computed by the Network Calculus approach. It means that, on average, the Trajectory approach is less pessimistic than the Network Calculus one, but it does not completely eliminate the pessimism. Thus, a first question is: is it possible to evaluate this remaining pessimism? A second one is: what are the reasons for this remaining pessimism? The following sections addresses these two points.

B. *Evaluation of the Remaining Pessimism*

For a given path of a given VL, the remaining pessimism is the difference between its exact worst-case end-to-end delay and the upper bound computed by the considered approach, e.g., the Trajectory approach. Obviously, the exact worst-case end-to-end delay is unknown. Then, the idea is to compute a sure upper bound of this difference. The principle is illustrated in Fig. 13. A sure lower bound of the worst-case end-to-end delay is computed. It is obtained by generating a scenario which is as close as possible from the exact worst-case scenario. It is called the unfavorable scenario in Fig. 13. Then, the difference between this sure lower bound and the sure upper bound computed by the Trajectory approach gives an upper bound on the pessimism of the Trajectory approach.

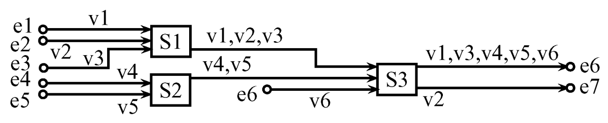


Fig. 14. Sample AFDX configuration.

TABLE V
TRANSMISSION TIME IN μs

VL	v1	v2	v3	v4	v5	v6
Tx time	40	80	160	40	80	40

The key point of this process is the generation of the unfavorable scenario. A scenario is unfavorable for a given path of a given VL if the packet of this flow is delayed as much as possible in each output port that the packet crosses. The generation of this unfavorable scenario is done by associating offsets to all the VLs which cross the VL vi under study. The offsets are assigned so that the sequences of packets in each input link of each node are synchronized in the same way as in Fig. 11. The generation proceeds in three steps.

First, it determines the packets contained in the sequence of each input link. In order to do that, it considers that vi and all the VLs which cross vi emit exactly one packet. All the other VLs of the configuration emit no packet. This assumption is valid, since VLs are defined as sporadic flows. Under this assumption, the packets contained in the sequence of each input link are known.

Second, it chooses an order of transmission of the packets in the sequence of each input link. This order is determined using the following criteria:

- the packets are first sorted by increasing number of nodes that they share with vi from the current node;
- second, the packets are sorted by decreasing size.

Third, it generates the sequence on the output link of the node. The packets are placed in this sequence in the order of their arrival time in the node (imposed by the FIFO scheduling policy). Packets with the same arrival time are ordered following the two previous criteria, i.e., the increasing number of shared nodes and then, the size.

The generation process is illustrated in the example in Fig. 14.

Table V gives the maximum packet transmission time of the six VLs of the configuration. First, let us consider the end-to-end delay of $v6$. On node $S3$, there are three input links: IP_0^{S3} with packet 6, IP_1^{S3} with packets 1 and 3 and IP_2^{S3} with packets 4 and 5. The ordering of the packets in sequences seq_1^{S3} and seq_2^{S3} has an impact on the end-to-end delay of $v6$. Indeed, when the packets are sorted by increasing size, the end-to-end delay of $v6$ is 336 μs, as depicted in Fig. 15. When they are sorted by decreasing size, the end-to-end delay of $v6$ is 376 μs, as depicted in Fig. 16. This is due to the fact that, when the largest packets are at the beginning of sequences seq_1^{S3} and seq_2^{S3}, the arrival time of the first packet of each sequence in $S3$ is postponed. Then, the transmission instant of packets on output link OP^{S3} are globally postponed. This property is true for most of the possible configurations of VLs.

Second, let us consider the end-to-end delay of $v1$. On node $S3$, there are three input links IP_0^{S3} with packets 2 and 3, IP_1^{S3} with packets 4 and 5 and IP_2^{S3} with packet 6. The ordering of the packets in sequences seq_1^{S3} and seq_2^{S3} have an impact on the end-to-end delay of $v1$. Indeed, when the packets are sorted solely by decreasing size, the end-to-end delay of $v1$ is 592 μs, as depicted in Fig. 17. This end-to-end delay is 672 μs in Fig. 18, where the packets are sorted by increasing order of nodes they share with $v1$ after the current node, then by decreasing order of their size. On the output link of node $S1$, packet 2 is before packet 3, because packet 2 leaves $v1$ after $S1$, while packet 3 shares node $S3$ with $v1$. Actually, the scheduling in Fig. 18 postpones the arrival time in $S3$ of packet 3. It leads to a longer delay for packet 1 in $S3$. This property is true for most of the possible configurations of VLs.

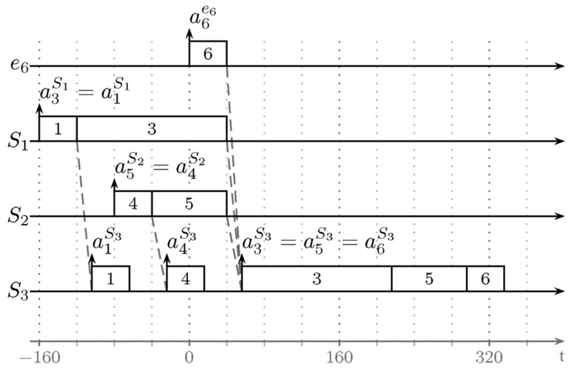


Fig. 15. $v6$ end-to-end delay: scheduling without any criterion.

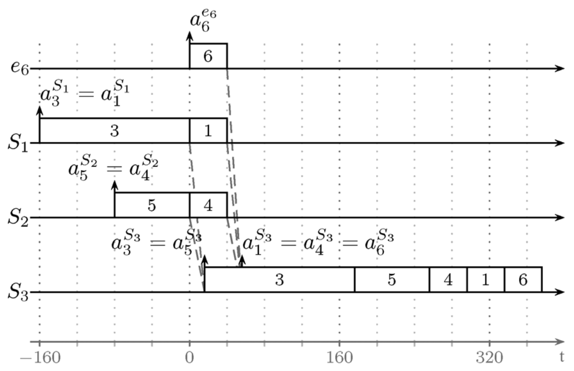


Fig. 16. $v6$ end-to-end delay: scheduling with decreasing size criterion.

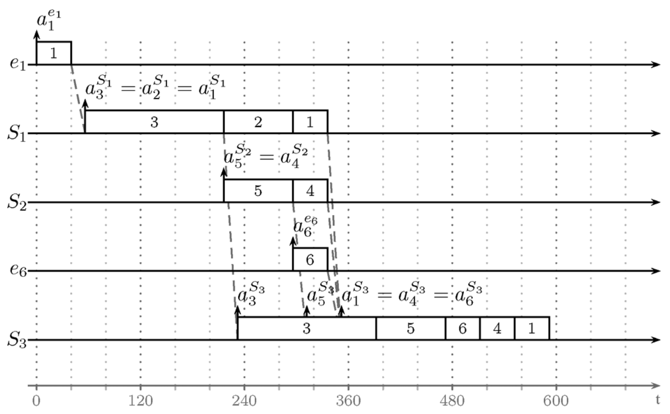


Fig. 17. $v1$ end-to-end delay: scheduling without hops criterion.

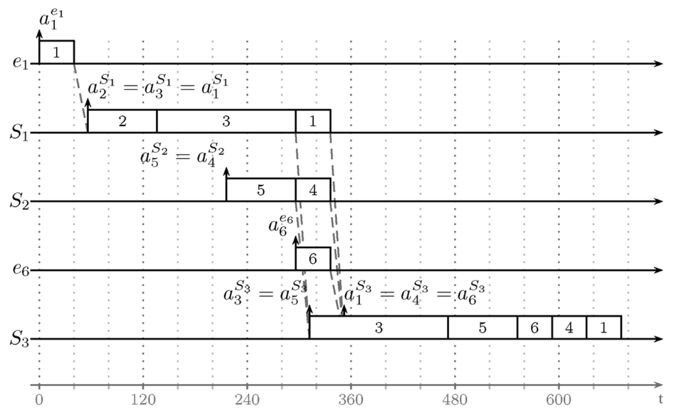


Fig. 18. $v1$ end-to-end delay: scheduling with both criteria.

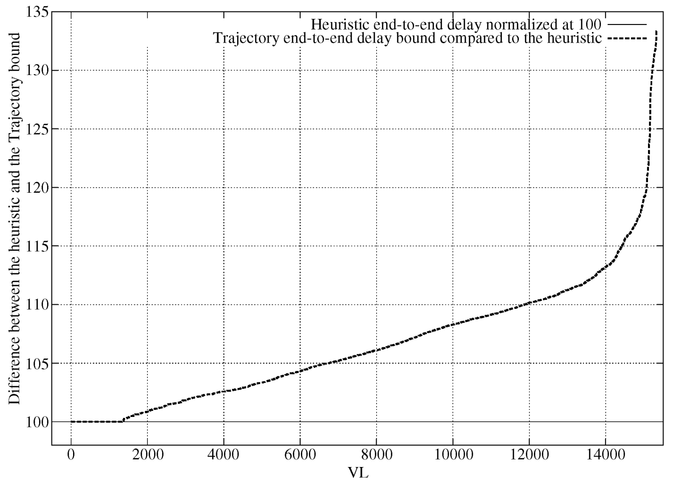


Fig. 19. Pessimism of the Trajectory approach.

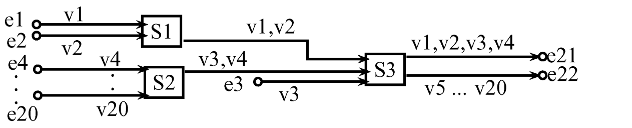


Fig. 20. Sample AFDX configuration.

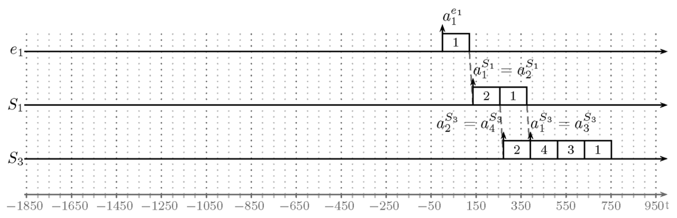


Fig. 21. The generated unfavorable scenario.

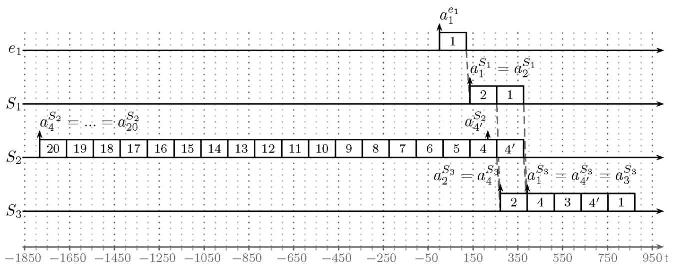


Fig. 22. A more unfavorable scenario.

A sure lower bound of the end-to-end delay has been computed for all the paths of the real avionics configuration evaluated in this paper, considering the unfavorable scenarios. Based on these sure lower bounds, the remaining pessimism of the Trajectory approach for this configuration is summarized in Fig. 19. The sure lower bound is considered as the reference and normalized to 100. The corresponding sure upper bounds computed by the Trajectory approach are depicted in Fig. 19. A value of 105 means that the sure upper bound is 5% larger than the sure lower bound. It means that, for the corresponding path, the pessimism of the upper bound computed by the Trajectory approach is at most 5%. On average, the upper bound on the pessimism of the Trajectory approach is 6.56%.

For 1332 paths, the sure lower and upper bounds are the same. It means that, for these paths, the Trajectory approach computes the exact upper end-to-end delay. For the 13 997 other paths, the upper bound on the pessimism is not null. It reaches 33% for some paths. This upper bound on the pessimism includes two parts:

- the difference between the exact worst-case and the sure upper bound, which is the effective pessimism of the approach;
- the difference between the sure lower bound and the exact worst-case, which is due to the fact that the unfavorable scenario is not necessarily the worst-case scenario (optimism of the sure lower bound).

The potential optimism of the sure lower bound is illustrated in the sample configuration in Fig. 20. This configuration includes 20 VLs $v1 \dots v20$ which all have the same $s_{\max}$, i.e., 1500 bytes. All the VLs have a 32 ms BAG, except $v4$ which has a BAG of 2 ms.

The unfavorable scenario leading to the sure lower bound considers that there is one single packet per VL. It leads to the scheduling depicted in Fig. 21. Thus, the sure lower bound on the end-to-end delay of $v1$ is 752 μs.

The scenario depicted in Fig. 22 considers that $v4$ emits two packets separated by one BAG, i.e., 2 ms. Due to the load of the output link of $S2$, the first packet of $v4$ is received by $S3$ immediately before the second one. Therefore, both packets of $v4$ delay in $S3$ the packet of $v1$ under study. In this scenario, the end-to-end delay of $v1$ is 872 μs. Thus, the optimism of the sure lower bound is at least 14% for $v1$.

Coming back to the results in Fig. 19, an analysis of the paths with the largest upper bounds on the pessimism shows that the sure lower bounds on their delay is often optimistic, because they have similar characteristics as the configuration of Fig. 20.

The pessimism of the Network Calculus approach is evaluated in the same way as for the Trajectory approach. The results are summarized in Fig. 23. The average upper bound of the pessimism is 13.57%. Depending on the path, this bound is between 0.8% and 63%.

The average value of this upper bound on the pessimism is twice the upper bound computed with the Trajectory approach. It means that the Trajectory approach is at least two times less pessimistic than the Network Calculus approach.

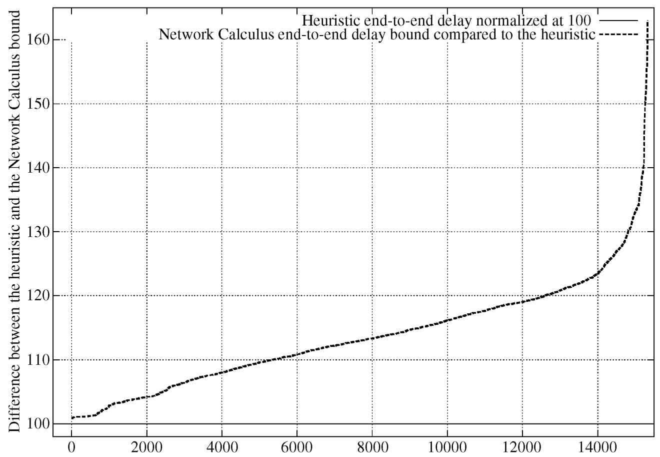


Fig. 23. Pessimism of the Network Calculus approach.

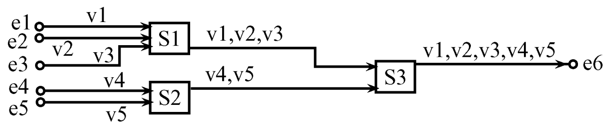
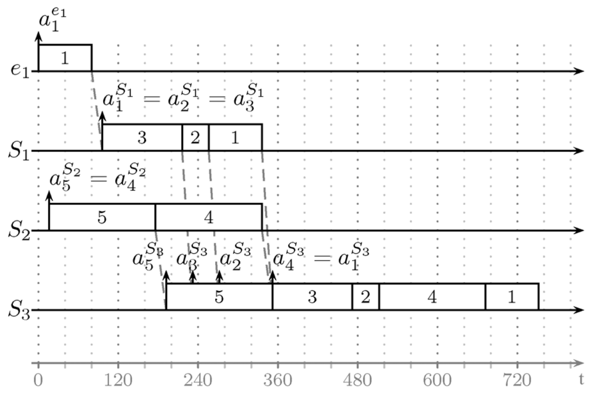


Fig. 24. Sample AFDX configuration.

TABLE VI
TRANSMISSION TIME IN μs

VL	v1	v2	v3	v4	v5
Tx time	80	40	120	160	160



$a_1^{e_1}$
e_1
$a_1^{S_1} = a_2^{S_1} = a_3^{S_1}$
S_1
$a_5^{S_2} = a_4^{S_2}$
S_2
$a_5^{S_3}$ $a_3^{S_3}$ $a_2^{S_3}$ $a_4^{S_3} = a_1^{S_3}$
S_3
0 120 240 360 480 600 720 t

Fig. 25. Worst-case scenario for $v1$.

C. Reasons for the Remaining Pessimism of the Trajectory Approach

This section shows an example where the Trajectory approach computes a sure upper bound of the worst-case end-to-end delay which is pessimistic. Let us consider the sample configuration depicted in Fig. 24.

Table VI gives the maximum packet transmission time of the five VLs of the configuration.

Based on the Trajectory approach, the worst-case end-to-end delay of VL $v1$ is 792 μs. The scenario leading to the exact worst case end-to-end delay for $v1$ has been determined by the model checking approach. This scenario is depicted in Fig. 25. It leads to an end-to-end delay of 752 μs for $v1$. With this scheduling, $\Delta_{S3} = a_3^{S3} - a_5^{S3} = 40$ μs.

The worst-case end-to-end delay of $v1$ computed by the Trajectory approach is 792 μs (pessimism of 40 μs). It includes the transmission time of each packet once, except for packets 1 and 3 which are counted twice, as they are the largest packets going, respectively, from node $e1$ to node $S1$ and from node $S1$ to node $S3$. Two link delays of 16 μs are also added. The extra 40 μs are due to the fact that Δ_{S3} is computed as 0. Actually, there are two input links: IP_0^{S3} with packets 1, 2, 3 and IP_1^{S3} with packets 4 and 5. From (14), we have

$$\delta_{S3} \geq \max(160 - 200, 0) = 0.$$

Indeed, the maximum value of l_0^{S3} is the sum of the transmission time of all the packets of $seq_0^{S3}(80 + 40 + 120)$, except the smallest one (40 for packet 2). The minimum value of l_1^{S3} is the sum of the transmission time of all the packets of $seq_1^{S3}(160 + 160)$, except the largest one (160 for packet 4).

Thus, for this configuration, the computation implemented by the Trajectory approach is pessimistic, because it considers that the first packet transmitted in seq_0^{S3} is the smallest one and this is a safe assumption. Unfortunately, in the worst-case scheduling depicted in Fig. 25, it is the biggest one.

More generally, the Trajectory approach becomes pessimistic when there are large differences between the length of the packets.

D. Reasons for the Pessimism of the Network Calculus Approach

The pessimism of the Network Calculus approach is illustrated on the configuration in Fig. 20. As presented in Section V-B, the 20 VLs of this configuration have the same packet size (1500 bytes) and BAG (32 ms), except $v4$ which has a 2 ms BAG.

In order to illustrate the pessimism of the Network Calculus approach, a set of computations is done with different loads on the output link of $S3$. The different loads are obtained with different values of BAG for $v4$. Each computation gives a sure upper bound of the end-to-end delay of $v1$ with both Trajectory and Network Calculus approaches. The results are given in Fig. 26.

This extra pessimism of the Network Calculus approach increases with the load until a load of 7.5%, corresponding to a 2 ms BAG for $v4$. At this point, the delay computed by the Trajectory approach integrates the transmission time of an additional packet from $v4$, as illustrated in Section V-B. The Network Calculus approach is based on a fluid model and does not integrate the concept of packet.

On this example, the delays computed by the Network Calculus approach increase continuously, whereas the delays computed by the Trajectory approach increase by step. On an arbitrary configuration, the probability that the load is close to a step is quite low. This explains why, most of the time, the Network Calculus approach is more pessimistic than the Trajectory approach. This extra pessimism is reduced for configurations with very different packet sizes, since the Network Calculus computation is less impacted by heterogeneous packet sizes.

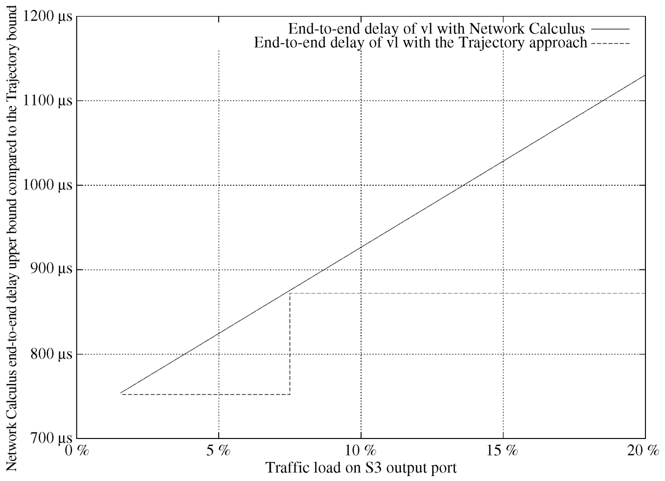


Fig. 26. Illustration of the pessimism of the Network Calculus approach.

VI. Conclusion

This paper shows how the Trajectory approach can be applied in the AFDX avionics network context. Unlike the Network Calculus approach that has been used for certification purpose, the Trajectory approach is based on the analysis of the worst-case scenario experienced by a packet on its Trajectory and not in each visited node. The resulting end-to-end delay computation is compared to the upper bounds obtained by the Network Calculus approach.

This paper emphasizes the importance of the grouping technique that has been proposed in the Network Calculus approach for configurations where the serialization effect of packets coming from the same link is substantial, as it is the case with AFDX networks.

A first contribution of this paper is the explanation of how the grouping technique can be introduced in the Trajectory approach. Moreover, a proof of the correctness of the corresponding computation is given.

A second contribution deals with the comparison of worst-case delay upper bounds obtained by Network Calculus approaches on a real avionics configuration. The Trajectory approach computes upper bounds which are tighter than the upper bounds computed by the Network Calculus one. It means that, on average, the Trajectory approach is less pessimistic than the Network Calculus one.

A third contribution is the computation of an upper bound of the pessimism of each approach on this avionics configuration. It has been shown that the Trajectory approach is at least two times less pessimistic than the Network Calculus one.

The worst-case analysis approaches presented in this paper consider a set of sporadic flows with no assumption concerning the arrival time of packets. This does not take into account the scheduling of the flows which are emitted by the same component. This scheduling could be integrated in the modeling by the mean of assumptions on the relative arrival time of packets, as it has been done in the automotive context [9]. The integration of this scheduling in the modeling of flows should distribute temporally the transmission of packets and very likely reduce the waiting time of packets in output buffers. Moreover, the sporadic characteristic of avionics flows could be taken into account with the help of a probabilistic modeling, as it has been proposed for the aperiodic traffic in the automotive context [13]. This leads to a probabilistic analysis of the worst-case delay of flows. Such an analysis has been proposed [21], based on a stochastic Network Calculus approach [22], [23].

For future aircraft, the addition of other type of flows (audio, video, best-effort, ...) on the AFDX network is envisioned. These different flows have different timing constraints and criticity levels. Thus, it is necessary to differentiate them and the FIFO policy on switch output ports is not suitable. Thus, it is necessary to consider other service disciplines, such as static priority queueing or weighted fair queueing [19]. The introduction of static priority queueing in the stochastic Network Calculus approach has been presented in [20]. The Trajectory approach is promising for handling heterogeneous flows needing QoS aware servicing policies at switches level [16], [17].

References

[1] *ARINC 653, Aeronautical Radio Inc., ARINC specification 653*, Avionics Application Software Standard Interface, 1997.

[2] *ARINC 429, Aeronautical Radio Inc.*, ARINC Specification 429. Digital Information Transfer System (DITS), Parts 1,2,3, 2001.

[3] ARINC Specification 664: Aircraft Data Network, Parts 1,2,7 Aeronotical Radio Inc., Tech. Rep., 2002–2005.

[4] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network," in *Proc. 21th ECRTS WiP Section*, Dublin, Ireland, Jul 2009, pp. 57, 60. [Online]. Available: http://gerfaut83.free.fr/article/Bauer-Scharbarg-Fraboul.pdf

[5] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying trajectory approach to AFDX avionics network," in *Proc. 14th Int. Conf. Emerging Technol. Factory Autom.*, Mallorca, Sep. 2009, pp. 1–8.

[6] C.-S. Chang, *Performance Guarantees in Communication Networks*. London, U.K.: Springer-Verlag, 2000, 1852332263.

[7] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an AFDX network," in *Proc. 18th ECRTS*, Dresden, Germany, Jul. 2006, pp. 193–202.

[8] F. Frances, C. Fraboul, and J. Grieu, "Using network calculus to optimize the AFDX network," in *Proc. ERTS*, Toulouse, France, Jan. 2006.

[9] M. Grenier, L. Havet, and N. Navet, "Scheduling messages with offsets on controller area network: A major performance boost," in *The Automotive Embedded Systems Handbook*, N. Navet and F. Simonot-Lion, Eds. New York: Taylor & Francis, Dec. 2008, ch. 14.

[10] J. Grieu, "Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques," Ph.D. dissertation, INP-ENSEEIHT, Toulouse, France, Sep. 2004.

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

[12] J. Jasperneite, P. Neumann, M. Theis, and K. Watson, "Deterministic real-time communication with switched ethernet," in *Proc. 4th IEEE Int. Workshop on Factory Commun. Syst.*, Västeras, Sweden, Aug. 2002, pp. 11–18.

[13] D. Khan, N. Navet, B. Bavoux, and J. Migge, "Aperiodic traffic in response time analysis with adjustable safety level," in *Proc. 14th Int. Conf. Emerging Technol. Factory Autom.*, Palma de Mallorca, Spain, Sep. 2009.

[14] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Germany: Springer-Verlag, 2001, vol. 2050, Lecture Notes in Computer Science, ISBN 3-540-42184-X.

[15] S. Martin, "Maîtrise de la dimension temporelle de la qualité de service dans les réseaux," Ph.D. dissertation, Université Paris XII, Paris, France, Jul. 2004.

[16] S. Martin and P. Minet, "Schedulability analysis of flows scheduled with FIFO: Application to the expedited forwarding class," in *Proc. 20th Int. Parallel and Distrib. Process. Symp.*, Rhodes Island, Greece, Apr. 2006.

[17] S. Martin and P. Minet, "Worst case end-to-end response times of flows scheduled with FP/FIFO," in *Proc. 5th IEEE Int. Conf. Networking*, Mauritius, Apr. 2006.

[18] J. Migge, "L'ordonnancement sous contraintes temps-réel: un modèle à base de trajectoires," Ph.D. dissertation, INRIA, Sophia Antipolis, France, Nov. 1999.

[19] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

[20] F. Ridouard, J.-L. Scharbarg, and C. Fraboul, "Probabilistic upper bounds for heterogeneous flows using a static priority queueing on an AFDX network," in *Proc. 13th Int. Conf. Emerging Technol. Factory Autom.*, Hambourg, Sep. 2008, pp. 1220–1227.

[21] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an AFDX network," *IEEE Trans. Ind. Informat.*, vol. 5, no. 1, pp. 38–49, Feb. 2009.

[22] M. Vojnović and J. Le Boudec, "Stochastic analysis of some expedited forwarding networks," in *Proc. Infocom*, New York, Jun. 2002.

[23] M. Vojnović and J. Le Boudec, "Bounds for independent regulated inputs multiplexed in a service curve network element," *IEEE Trans. Commun.*, vol. 51, no. 5, pp. 735–740 , May 2003.

Henri Bauer graduated from the Université de Poitiers, Poitiers, France, and the Ecole Nationale Supérieure de Mécanique et d'Aérotechnique (ENSMA), France, in 2006. He is currently working towards the Ph.D. degree in computer science at the Université de Toulouse, Toulouse, France, in partnership with the IRIT Laboratory and Airbus Operations.

His main research interest is embedded networks worst case analysis in avionics context.

Jean-Luc Scharbarg received the Ph.D. degree in computer science from the University of Rennes, Rennes, France, in 1990.

He has been an Associate Professor at the Universite de Toulouse (INPT/ENSEEIHT and IRIT Laboratory) since 2002. His current research interest concerns the analysis and performance evaluation of embedded networks, mainly in the context of avionics and automotive.

Christian Fraboul received the Engineer degree from INPT/ENSEEIHT, Toulouse, France, in 1974.

From 1974 to 1998, he worked as a Research Engineer at ONERA. Since 1998, he is full time Professor at INPT, where he is in charge of the Department of Telecommunications and Networks, ENSEEIHT, and of the IRT team of the IRIT Laboratory. His main fields of interest are embedded networks architectures and performance evaluation of such architectures (mainly in avionics context).