21

Methods and Tools for the Temporal Analysis of Avionic Networks

Jean-Luc Scharbarg and Christian Fraboul
Université de Toulouse - IRIT/ENSEEIHT/INPT
France

1. Introduction

Thanks to the Integrated Modular Avionics concept [ARI (1991; 1997)], functions developed for civilian aircraft share computation resources. However, the continual growing number of these functions implies a huge increase in the quantity of data exchanged and thus in the number of connections between functions. Consequently, traditional ARINC 429 buses [ARI (2001)] can't cope with the communication needs of modern aircraft. Indeed, ARINC 429 is a single-emitter bus with limited bandwidth and a huge number of buses would be required. Clearly, this is unacceptable in terms of weight and complexity.
In order to cope with this problem, the AFDX (Avionics Full DupleX Switched Ethernet) [ARI (2002-2005)] was defined and has become the reference communication technology in the context of avionics. AFDX is a full duplex switched Ethernet network to which new mechanisms have been added in order to guarantee the determinism of avionic communications. This determinism has to be proved for certification reasons and an important challenge is to demonstrate that an upper bound can be determined for end-to-end communication delays.
An important assumption is that all the avionics communication needs can be statically described: asynchronous multicast communication flows are identified and quantified. All these flows can be statically mapped on the network of AFDX switches. For a given flow, the end-to-end communication delay of a frame can be described as the sum of transmission delays on links and latencies in switches. Thanks to full duplex links characteristics, no collision can occur on links and transmission delays on links depend solely on bandwidth and frame length. But, as confluent asynchronous flows compete, on each switch output port, highly variable latencies can occur when a frame crosses a switch. Thus it is necessary to analyze these latencies in order to determine the upper bounds on end-to-end communication delays for each flow.
At least three approaches have been proposed in order to compute a worst-case bound for each communication flow of the avionic applications on an AFDX network configuration. They are based on network calculus, trajectories and model checking. Such a worst-case communication delay analysis allows the comparison between the computed upper bounds and the constraints on the communication delays of each flow. Moreover it allows the scaling of the switches memory buffers in order to avoid buffer overflow and frame losses. However, communication delays measured on a real configuration are much lower than the computed upper bound. This is mainly due to the fact that rare events are difficult to observe on a real configuration in a reasonable time.

In order to better understand the real behavior of the AFDX network, a simulation model of the network is proposed as a second step. Such a simulation approach allows the calculation, on the modeled network, of the end-to-end delay for each flow, according to a representative subset of possible scenarios. Thus an end-to-end delay distribution can be obtained for each flow, leading to a better understanding of communication delays. However such an approach cannot be used for certification needs as rare events can be missed by simulation.

This chapter summarizes the assumptions of the AFDX network technology and gives an overview of the different approaches which are used for the temporal analysis of such networks, i.e. the three approaches for the worst-case analysis and the simulation approach for the computation of the distributions of the end to end delays. The approaches are illustrated on a sample configuration and results on a realistic avionic configuration are shown.

2. The end-to-end delay analysis of an AFDX network

This section gives a short overview of an AFDX network and characterizes the end-to-end delay of a flow transmitted on such a network.

2.1 Overview of an AFDX network

The AFDX (Avionics Full DupleX Switched Ethernet) [ARI (2002-2005)] is a switched Ethernet network taking into account avionic constraints. An illustrative example is depicted in figure 1. It is composed of four interconnected switches $S1$ to $S4$. 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 the *End Systems* ($e1$ to $e8$ in Figure 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 [ARI (2002-2005)].

End systems exchange Ethernet frames through VLs. Switching a frame from a transmitting to a receiving end system is based on VL. The Virtual Link defines a logical unidirectional connection from one source end system to one or more destination end systems. Coming back to the example in Figure 1, $v4$ is an unicast VL with path {$e3$ – $S3$ – $S4$ – $e7$}, while $v6$ is a multicast VL with paths {$e1$ – $S1$ – $S2$ – $e6$} and {$e1$ – $S1$ – $S4$ – $e7$}.

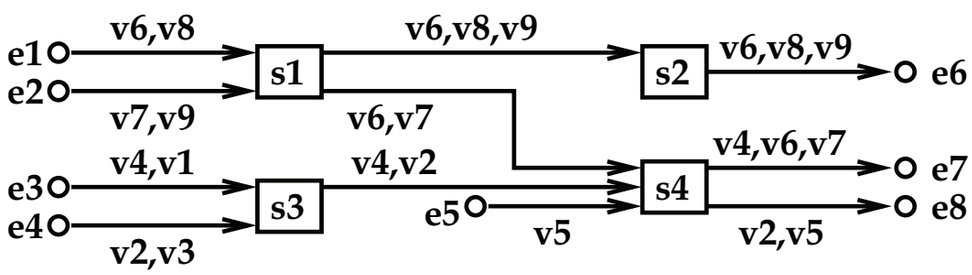


Fig. 1. An illustrative AFDX configuration

The routing is statically defined. Only one end system within the avionics network can be the source of one Virtual Link, (i.e. mono transmitter assumption). A VL v definition also includes the Bandwidth Allocation Gap ($BAG(v)$), the minimum and the maximum frame length ($s_{min}(v)$ and $s_{max}(v)$). $BAG(v)$ is the minimum delay between two consecutive frames of the associated VL (which actually defines a VL as a sporadic flow).
VL parameters ($BAG(v)$, $s_{max}(v)$) 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.
A realistic AFDX configuration is presented and analyzed in section 7. It includes nearby one thousand VLs. The next paragraph characterizes the end-to-end delay of a VL transmitted on an AFDX network.

2.2 Characterization of the end-to-end delay of a VL

Let's consider a path p_x of a VL v. The end-to-end delay $D(F_v, p_x)$ of a frame F_v transmitted on p_x is defined by

$$D(F_v, p_x) = LD(F_v, p_x) + SD(F_v, p_x) + WD(F_v, p_x) \quad (1)$$

where:

- $LD(F_v, p_x)$ is the transmission delay over the links: thanks to the full duplex characteristic of the AFDX, there are no collisions on the links. Thus, the transmission delay over a link is $t_{byte} \times s(F_v)$ where t_{byte} is the transmission time of one byte and $s(F_v)$ is F_v length. Therefore, considering that all the links have the same bandwidth c (consequently, t_{byte} is the same for all the links),

$$LD(F_v, p_x) = nbl(p_x) \times (t_{byte} \times s(F_v)) \quad (2)$$

  where $nbl(p_x)$ is the number of links in p_x.
- $SD(F_v, p_x)$ is the delay in switches between input and output ports: in the context of this presentation, the delay in a switch from an input port to an output port is considered as a constant td, since the only available information about this delay is a guaranteed upper bound of 16 μs. Thus

$$SD(F_v, p_x) = nbs(p_x) \times td \quad (3)$$

  where $nbs(p_x)$ is the number of switches in p_x.
- $WD(F_v, p_x)$ is the delay in switches and end system output buffers: this delay highly depends on each output port load at the time where F_v reaches it. Thus

$$WD(F_v, p_x) = WD(F_v, p_x, ev) + \sum_{sk \in \Psi_{p_x}} WD(F_v, p_x, sk) \quad (4)$$

  where ev is F_v source end system, Ψ_{p_x} is the set of switches in p_x, $WD(F_v, p_x, ev)$ is the delay in ev output buffer and $WD(F_v, p_x, sk)$ is the delay in sk output port buffer.

Consequently, $D(F_v, p_x)$ can be divided into a fixed part $LD(F_v, p_x) + SD(F_v, p_x)$ and a variable part $WD(F_v, p_x)$. The fixed part can be statically computed since it depends solely on the path p_x, the length of the frame F_v and the bandwidth o the links. The variable part depends on

dynamic characteristics, such as the sequence of frames which are emitted by each VL (the length of each frame) and the offsets between the different VLs, i.e. the emission instant of the first frame of each VL, as it is shown by the following example.

Let's consider the AFDX configuration in figure 2. This configuration includes five unicast VLs $v1 \ldots v5$. The parameters of these VLs - their BAGs, frames sizes and paths - are given in table 1. The bandwidth of every link is 100 Mb/s (t_{byte} = 0,08 μs). Figure 3 exhibits three possible scenarios for the transmission of the frames of the five VLs $v1 \ldots v5$ on the network in figure 2. The switching delay td is assumed to be null. This means that $SD(Fi, pi) = 0$ for every frame Fi on every path pi. One single BAG of the three considered scenarios is depicted in figure 3. The analysis focuses on the end-to-end delay of the frame F_1 of VL $v1$ (the path is $p1 = e1 - s1 - s3 - e6$). When the length of F_1 is $s_{max}(v1)$ (i.e. 500 bytes), the transmission delay on the links $LD(F1, p1)$ is $3 \times (0{,}08 \times 500) = 120\ \mu s$. When the length of F_1 is $s_{min}(v1)$ (i.e. 300 bytes), this transmission delay $LD(F1, p1)$ is $3 \times (0{,}08 \times 300) = 72\ \mu s$. In figure 3, the frame F_i from VL vi is denoted i.

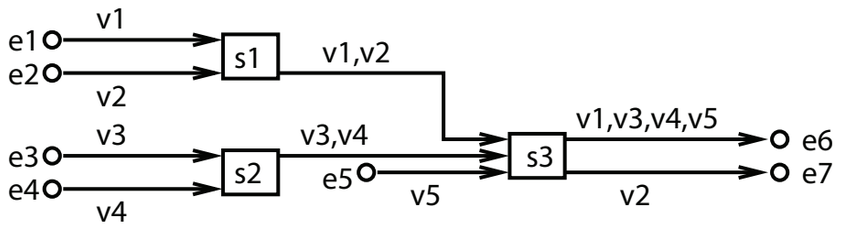


Fig. 2. A sample AFDX configuration

	$BAG(vi)$ (ms)	$s_{min}(vi)$ (byte)	$s_{max}(vi)$ (byte)	path pi
v1	4	300	500	$e1 - s1 - s3 - e6$
v2	4	300	500	$e2 - s1 - s3 - e7$
v3	4	300	500	$e3 - s2 - s3 - e6$
v4	4	300	500	$e4 - s2 - s3 - e6$
v5	4	300	500	$e5 - s3 - e6$

Table 1. Parameters of the sample AFDX configuration

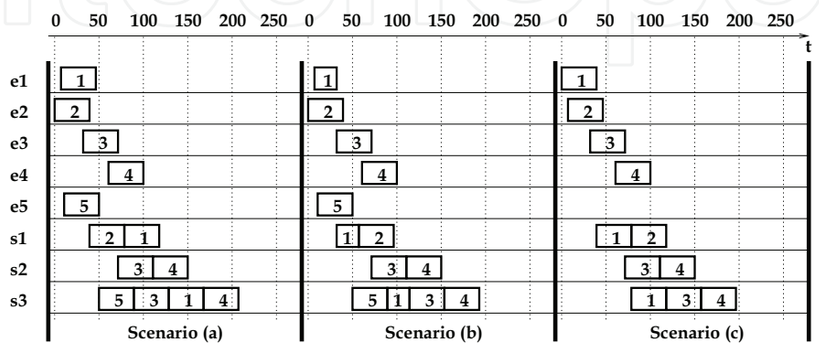


Fig. 3. Three possible scenarios for the sequence of frames

In the scenario **a**, each VL vi emits a frame with the maximal length $s_{max}(vi)$. The end-to-end delay of F_1 is 160 μs. It includes the transmission on links (120 μs) and the waiting time in output port buffers (40 μs). Indeed, F_1 waits for frame F_2 in switch $s1$ and it waits for frame F_3 in switch $s3$.
In the scenario **b**, the frames are generated at the same instants as in the scenario **a**, but the length of the frame F_1 of VL $v1$ is now 300 bytes. The end-to-end delay of F_1 is now 107 μs. It includes the transmissions on links (72 μs) and the waiting time in the output port buffer in switch $s3$ (35 μs). The scenarios **a** and **b** show that the length of a given frame can influence its waiting delay in output port buffers.
In the scenario **c**, $v1$, $v2$, $v3$ and $v4$ generate a frame with the maximal possible length (i.e. 500 bytes), while $v5$ does not generate a frame. The instant where the frames from $v1$ and $v2$ are generated are switched in comparison with the two previous scenarios, while these intants are not modified for $v3$ and $v4$. In this scenario **c**, the frame F_1 of $v1$ does not wait in output port buffers. Consequently, its end-to-end delay is 120 μs, i.e. the transmission time on links. This scenario shows that, for a given VL, its offset to the other VLs and the emission or non emission of frames by the other VLs influence its end-to-end delay.
The end-to-end delay analysis of a path p_x of a VL v has to take into account all the possible scenarios. This analysis should determine the following characteristics of this end-to-end delay.

- The smallest possible value of the end-to-end delay, which corresponds to the scenarios where the VL v emits a frame with minimal length $s_{min}(v)$ which never waits in output ports. This smallest possible value is denoted $D_{min}(v, p_x)$ and it is computed from equations 1, 2 and 3:

$$D_{min}(v, p_x) = nbl_{p_x} \times (t_{byte} \times s_{min}(v)) + nbs_{p_x} \times dt \tag{5}$$

- The highest possible value of the end-to-end delay, which corresponds to the worst case scenario. It is mandatory for the certification of the avionic network. In the general case, finding this worst-case scenario requires an exhaustive analysis of all the possible scenarios. Section 4 presents an approach which implement this exhaustive analysis. Such an exhaustive enumeration is impossible for any realistic configuration, since the number of possible scenarios is huge, due to the number of VLs (around 1000). An alternative solution is the computation of a safe upper bound of the end-to-end delay, based on a modelling of the configuration which over-estimate the traffic and/or underestimate the service offered by the network (pessimistic assumptions). Sections 5 and 6 present two approaches which compute a pessimistic safe upper bound of the end-to-end delay of any VL of an industrial AFDX configuration.
- The distribution of the end-to-end delay between its smallest and its highest possible values. This distribution is valuable when prototyping the whole system. This distribution can be obtained thanks to a simulation approach which is summarized in section 3.

Figure 4 summarizes the characteristics of the end-to-end delay. This delay is always between a minimum and a maximum value. Most of the time, the exact maximum value cannot be computed and it is lower-bounded by the maximum observed value and upper bounded by the computed safe upper bound.

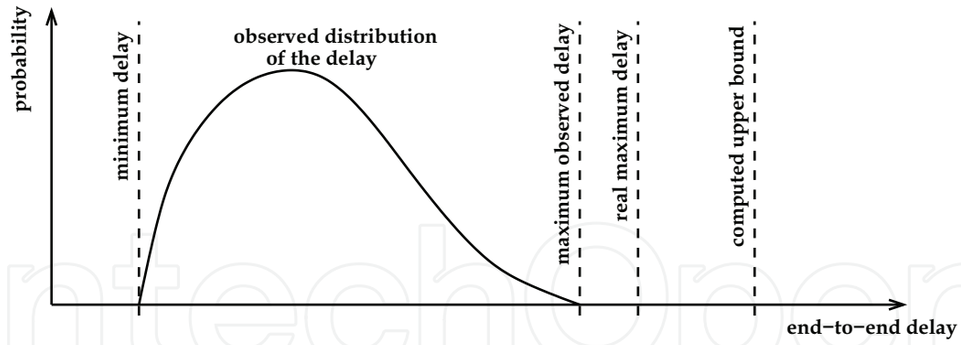


Fig. 4. The end-to-end delay characteristics

3. The simulation approach for the distribution of end-to-end delays

A simulation scenario is characterized by the sequence of frames emitted by each VL and the offsets between the different VLs. It has been previously noted that a typical AFDX network includes around 1000 VLs. Clearly, this leads to a huge set of possible scenarios from which it is difficult to extract a representative subset. The resulting challenge is, for each VL path, to focus on the part of the network that is relevant for this path's end-to-end delay distribution in order to reduce the simulation space. This is a mandatory requirement for the simulation approach. It is fulfilled by means of the VLs taxonomy that is presented in the next section.

3.1 A taxonomy of VLs

The basic idea of the taxonomy is that, given a path *px* of a VL *vx*, the other VLs do not have the same level of influence on it. For example, a *vx* frame can wait for the end of transmission of another frame only if the latter shares at least one output port with *px*. The application of this idea is to focus the simulation on the VLs that influence the end-to-end delay distribution of *vx* frames.

The taxonomy is illustrated considering the unicast VL *vx* in figure 5. Its path *px* is *e*3-*s*3-*s*4-*e*8. The paths or portions of paths of other VLs of this AFDX configuration can be divided into three classes [Charara, Scharbarg & Fraboul (2006)], as depicted in figure 5.

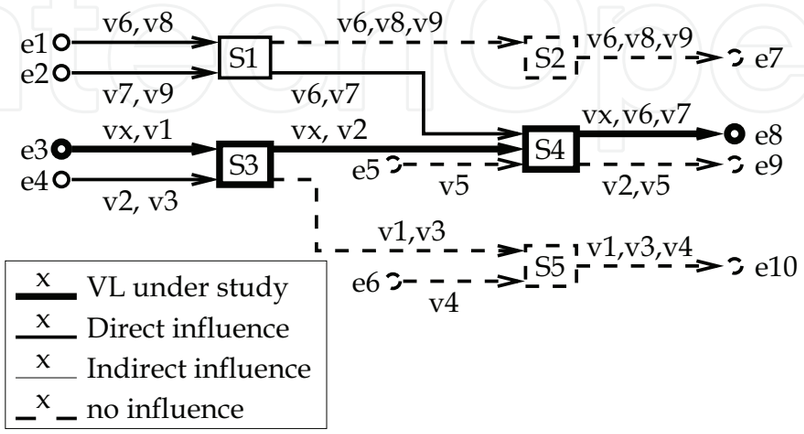


Fig. 5. Taxonomy of VLs

- Class *DI* (Direct Influence) contains all the paths that share at least one output buffer with *px*, truncated after the last output buffer shared with *px*. In figure 5, it contains the whole VL *v*7, path *e*1 - *s*1 - *s*4 - *e*8 of *v*6 and sub-paths *e*3 - *s*3 and *e*4 - *s*3 - *s*4 of *v*1 and *v*2 respectively.
- Class *II* (Indirect Influence) contains all the paths or portions of paths that share no output buffer with *px*, but at least one output buffer with a *DI* or an *I I* path. In figure 5, sub-paths *e*1 - *s*1 of *v*8, *e*2 - *s*1 of *v*9 and *e*4 - *s*3 of *v*3 are classed as indirect influence portions of VL paths.
- Class *NI* (No influence) contains all the paths or portions of paths that are not in class *DI* or class *II*. It contains all links represented with dashed lines in figure 5.

In this illustrative example containing ten VLs overall, classes *DI* and *II* each contain four and three VLs respectively. Figure 6 shows the partitioning between classes *DI*, *II* and *NI* for each VL path in a realistic network including 1000 VLs and 6400 paths. The continuous and dashed lines respectively give the number of VLs in class *DI* for each path and the number of VLs in classes *DI* or *II*. In this industrial network, on average, a VL path has 150, 650 and 200 *DI*, *II* and *NI* VLs respectively.

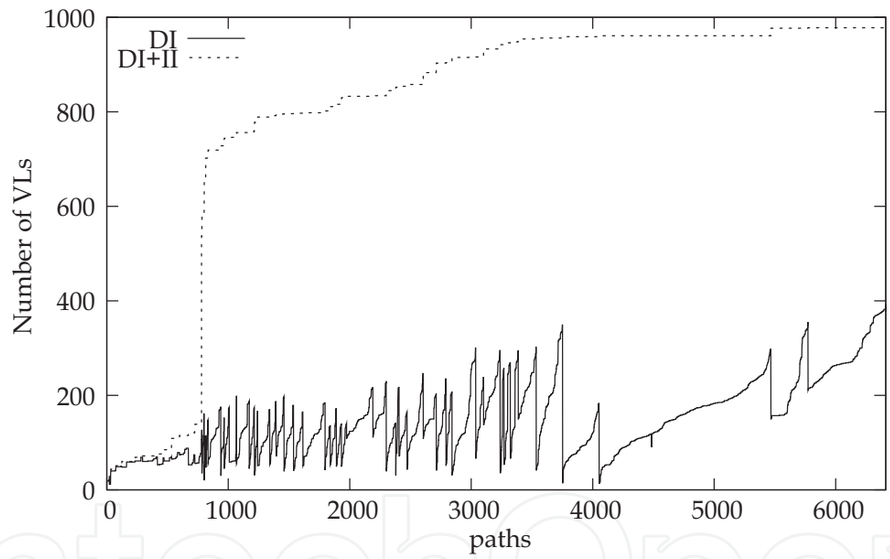


Fig. 6. Industrial configuration taxonomy

Considering this VL classification, VLs in class *NI* clearly have no impact on the end to end delay of their associated path *px*. Thus, VLs in class *NI* will not be considered in the definition of a scenario for a *px* end-to-end delay analysis. For the network analyzed in figure 6, this leads to a drastic reduction of the simulation space for approximately 800 VLs paths (each scenario includes less than 150 VLs instead of nearby 1000). Unfortunately, this reduction is quite poor for the 5600 remaining VLs paths (each scenario includes an average of 800 Vls). In order to obtain a larger reduction of the simulation space, the VL classification has to be exploited more effectively. The main idea concerns VLs in class *II*. They could be ignored in the definition of a scenario for a *px* end-to-end delay analysis provided they have no influence on *px* end-to-end delay distribution. The next section studies the effective influence of VLs in class *II*.

3.2 Effective influence of VLs in class II

The influence of a VL in class *II* on *px* is illustrated with the example depicted in figure 7. It includes one switch $s1$, four end systems $e1, \ldots, e4$ and three VLs vx, $v1$ and $v2$. These three VLs have identical BAGs and frame lengths. Using the taxonomy presented in section 3.1, unicast VL vx is directly influenced by $v1$ (class *DI*) and indirectly influenced by $v2$ (class *I I*). Depending on the scenario (phasings for vx, $v1$ and $v2$), $v2$ can have an influence on the vx end-to-end delay by modifying the $v1$ arrival time at the switch $s1$ output port. The three possible cases are illustrated in figure 8, considering three scenarios. For each of them, figure 8 shows the modification of the vx end-to-end delay due to $v2$ frames. For the three scenarios, $v1$ and $v2$ are ready for transmission simultaneously and each $v2$ frame is arbitrarily transmitted before the corresponding $v1$ frame. Thus, the non-transmission of a $v2$ frame advances the arrival time of the corresponding $v1$ frame at the switch $s1$ output port. In scenario *a* in figure 8, this leads to a shorter vx end-to-end delay because it allows the $v1$ frame to complete transmission on the $s1$ - $e3$ link before the arrival of the vx frame at the $s1$ output port. Conversely, it leads to a longer vx end-to-end delay in scenario *b*, because the arrival order of the vx and $v1$ frames at the $s1$ output port is inverted and consequently, the vx frame has to wait. Finally, the non-transmission has no influence in scenario *c*, because the vx frame arrives before the $v1$ one in both cases and as a result never waits.

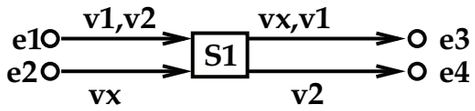


Fig. 7. Example of II Influence

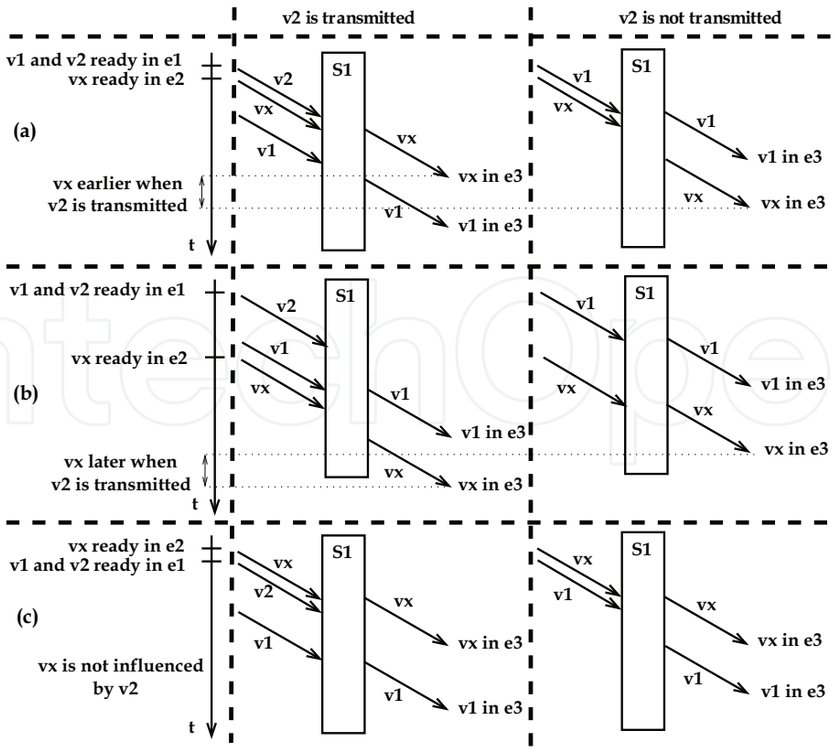


Fig. 8. Possible frame arrival times

Thus, depending on the application scenario, $v2$ frames can shorten, lengthen or have no influence on vx end-to-end delays. However, it remains to be seen if VLs in class *II* (e.g. $v2$) modify the end-to-end delay distribution of px, their associated VL path.
In order to answer this question, every possible VL path must be examined. The basic idea is to determine, for each VL path, the end-to-end delay distributions considering first, that VLs in class II are present, and second, that they are not present. The goal is to determine whether VLs in class II modify the end-to-end delay distributions (there is at least one VL path for which the two obtained distributions are different) or not (such a VL path does not exist). In the latter case, VLs in class II do not have to be taken into account when determining end-to-end delay distributions.
End-to-end delay distributions are obtained using a simulation approach. The simulation process is detailed in [Scharbarg et al. (2009)]. It considers all the possible kinds of VLs of a typical industrial AFDX configuration. For each considered VL, it compares the distribution of end-to-end delays obtained, first when VLs in class *II* are transmitted, second when VLs in class *II* are not transmitted. The two obtained distributions are the same for all the tested VLs. Thus the conclusion is that VLs in class *II* do not have to be taken into consideration for the computation of vx end-to-end delay distribution.
The resulting reduced simulation space makes it possible to determine an experimental probabilistic upper bound for every VL path in a realistic network. The simulation process considers a specific model for each VL path. Since an industrial network configuration includes more than 6000 paths, this leads to a heavy simulation process. A mean of speeding up this process has been presented in [Scharbarg & Fraboul (2007); Scharbarg et al. (2009)]. It consists in building a simplified model for each VL path. Such a model is depicted in Figure 9. It corresponds to a VL path which crosses two switches. The set of componants (switches and end systems) leading to each input port of a switch crossed by the path is modeled by one end system which emits all the VLs crossing this input port. It has been shown in [Scharbarg & Fraboul (2007); Scharbarg et al. (2009)] that this simplification does not modify the computed end-to-end delay distribution.

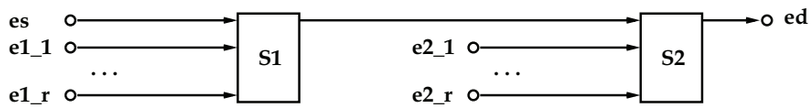


Fig. 9. A generic simulation model

The simplified flow model allows the evaluation of end-to-end delays by queueing network simulation mechanisms. The obtained end-to-end delay distributions give important information for the designer about the real behavior of the applications sharing the AFDX network configuration. Moreover, it provides both an experimental upper bound as well as an estimation of the probability to exceed a given bound.
These experimental upper bounds obtained by simulation are not safe, because simulation mechanisms are unable to efficiently take into account rare events. But safe upper bounds are needed for certification purposes. The next sections present different approaches which allow the computation of such safe upper bounds.

4. The model checking approach for computing exact worst-case delay

The proposed approach is based on a modeling in timed automata. Timed automata have been first proposed in [Alur & Dill (1994)] in order to describe systems behavior with time.

This section first gives a brief overview of timed automata. Then, the modeling of the AFDX network is presented. Finally, the verification process which computes the exact end-to-end delay upper bound is described and applied to the sample configuration in Figure 2.

4.1 Overview of timed automata

A timed automaton is a finite automaton with a set of clocks, *i.e.* real and positive variables increasing uniformly with time. Transitions labels can be:

- a guard, *i.e.* a condition on clock values,
- actions,
- updates, which assign new value to clocks.

The composition of timed automata is obtained by a synchronous product. Each action *a* executed by a first timed automaton corresponds to an action with the same name *a* executed in parallel by a second timed automaton. In other words, a transition which executes the action *a* can be fired only if another transition labeled *a* is possible. The two transitions are performed simultaneously. Thus communication uses the rendez-vous mechanism.

Performing transitions requires no time. Conversely, time can run in nodes. Each node is labeled by an invariant, that is a boolean condition on clocks. The node occupation is dependent of this invariant: the node is occupied if the invariant is true.

Several extensions of timed automata have been proposed. One of these extensions is timed automata with shared integer variables. The principle consists in defining a set of integer variables which are shared by different timed automata. Consequently, the values of these variables can be consulted and updated by the different timed automata [Larsen et al. (1997); Burgueño Arjona (1998)].

A system modelled with timed automata can be verified using a reachability analysis which is performed by model-checking. It consists in encoding each property in terms of the reachability of a given node of one of the automata. So, a property is verified by the reachability of the associated node if and only if this node is reachable from an initial configuration. Reachability is decidable and algorithms exist [Larsen et al. (1997)]. In the general case, reachability analysis is undecidable on timed automata with shared integer variables. In the particular case where the shared variables are represented by nodes of a timed automata, the reachability analysis is decidable.

The approach considered in this paper is based on timed automata with shared integer variables which are represented by nodes of a timed automaton. The modeling of the AFDX with timed automata is now presented.

4.2 The modeling of an AFDX network

The modeling of an AFDX network considers an automaton for each VL and an automaton for each output port of a switch. Figure 10 depicts the timed automaton of a VL with a BAG equal to *period*. This automaton sends a first message $send_i$ ($send_0$ in the example) delayed by a duration between 0 and *period*, and then sends periodically a new message $send_i$ (the period is equal to the BAG of the VL, *i.e. period*). So, this automaton models a periodic VL with an offset between 0 and its BAG.

Figure 11 shows an example of an output port of a switch. Each node of the automaton models a location in the FIFO queue associated to the port. Consequently, the number of nodes of the automaton equals the size of the queue (3 in the example of Figure 11). Each

transition from a node $Position_i$ to a node $Position_{i+1}$ of the automaton models the arrival of one frame at the transmit port while each transition from a node $Position_{i+1}$ to a node $Postion_i$ models the end of the transmission from this port. The automaton of the Figure 11 considers two flows (*i.e.* two VLs) received using signals *send*0 and *send*1 and transmitted using signals *send*4 and *send*5, corresponding respectively to *send*0 and *send*1. *delay* is the transmission time of the frame. In the considered example application, all the frames have the same length. *pos*1, *pos*2 and *pos*3 indicate the flows (0 or 1) corresponding to the frames waiting in each position of the queue.

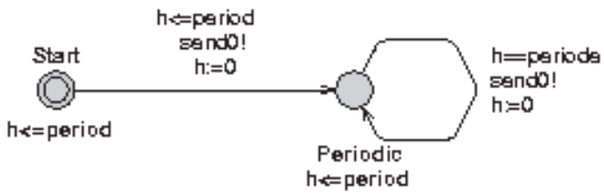


Fig. 10. Automaton of a VL

The global system is obtained by composing the timed automata of the VLs and the outpout ports of the switches. For instance, the network in Figure 2 is composed of 5 VLs and 4 output ports. So, the model is composed of 9 timed automata, as depicted in Figure 12. As an example, VL *v*2 is modelled by the timed automaton *v*2, which sends signal *send*1. This is received by the timed automaton *p1-1* which models the unique output port of the switch *s*1. *v*2 follows the path composed of signals *send*5 and *send*10.

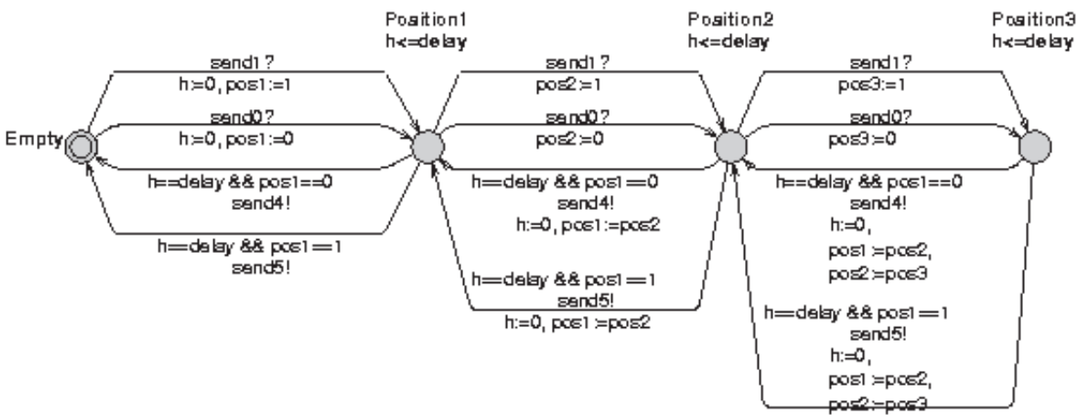


Fig. 11. Automaton of an output port of the switch

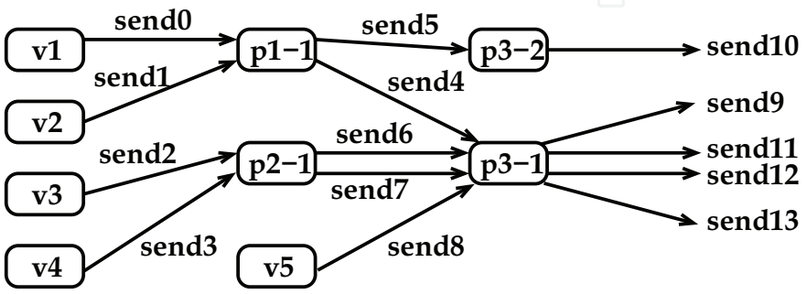


Fig. 12. The global modelled system

4.3 The computation of the exact worst-case end-to-end delay

Using the test automaton method [Burgueño Arjona (1998); Bérard et al. (2001)], the worst case end-to-end delay of each VL is obtained from the model previously described. The test automaton corresponding to the VL $v1$ is depicted in Figure 13. This automaton models the property "delay of $v1$ is less than *bound*". By receiving signal *send*0, it evolves to the node $s2$. Then the signal *send*9 is waited (transmission of $v1$ from the output port of the switch $s3$, see Figure 12). If this signal is not received before the delay of *bound*, the automaton evolves to the node *reject*. This behaviour corresponds to a scenario for which the transmission delay of $v1$ is greater than *bound*. So, the analysis consists in finding the lowest value of *bound* for which the node *reject* is reached. This value is the maximum end-to-end delay.
To verify the property, we use the model-checker U*PPAAL*. The calculation takes less than 1s on a Linux station with a Pentium 4 processor and 2GB of memory size. The exact worst case end-to-end delays obtained for each VLs in the Figure 2 are given in table 2.

VL	Exact worst-case
$v1$	272
$v2$	192
$v3$	272
$v4$	272
$v5$	176

Table 2. EWC end-to-end delays in μs

This approach exhibits the exact worst-case and it is valuable, since it helps to understand the worst-case behaviour of the network. However, it cannot be used for the certification of a realistic network (e.g. the AFDX of the A380), due to the well known combinatorial explosion problem. Therefore, methods which upper bound the end-to-end delay of each flow are used.

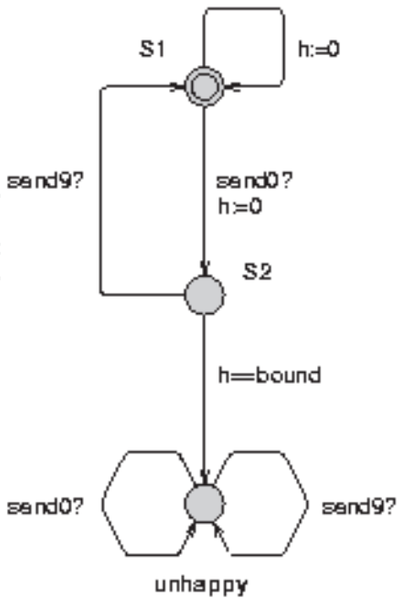


Fig. 13. Test automaton of $v1$

5. AFDX worst-case delay analysis with Network Calculus

Network Calculus [Chang (2000); Le Boudec & Thiran (2001)] 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 paragraph 5.1. The improvement of this basic approach in the context of AFDX is described in paragraph 5.2.

5.1 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:

$$(f \otimes g)(t) = \inf_{0 \le u \le t} \left(f(t-u) + g(u) \right) \quad and \quad (f \oslash g)(t) = \sup_{0 \le u} \left(f(t+u) - g(u) \right) \tag{6}$$

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:

$$h(\alpha, \beta) = \sup_{s \ge 0} \left(\inf \left\{ \tau \ge 0 \mid \alpha(s) \le \beta(s+\tau) \right\} \right) \tag{7}$$

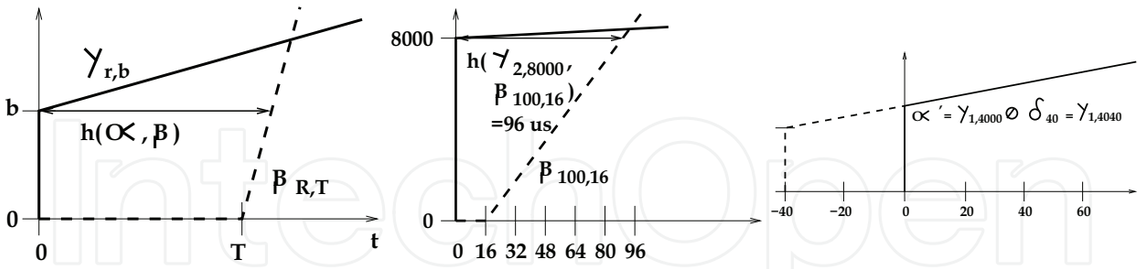
Each VL of an AFDX network (a flow) is modeled by a *leaky bucket* $\gamma_{r,b}$, with $b = s_{max}(v)$ and $r = \frac{s_{\max}(v)}{BAG(v)}$. The burst b is the capacity of the bucket and the rate r is the leak rate.

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. 16μs. R is the servicing rate (100 Mb/s in our context) and $[x]^+ = max(0, x)$. Thus, in the context of this chapter, the service curve which is offered by each output port is $\beta_{100,16}$.

Figure 14(a) 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 on the small example in Figure 2. Let's consider the VL $v1$. Its arrival curve in S1 is $\alpha_1 = \gamma_{1,4000}$, since $r = \frac{s_{max}(v1)}{BAG(v1)} = \frac{4000}{4000} = 1Mb/s$ and $b = s_{max}(v1) = 4000bits$. $v1$ 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 Figure 14(b). 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.



a) The maximum delay $h(\alpha, \beta)$ (b) Output port of switch $S1$ (c) Output curve after switch $S1$ or $S2$

Fig. 14. Curves for network calculus

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: $\alpha' = \alpha \oslash \delta_{jitter}$. 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 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 Figure 14(c).
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 3 summarizes the upper bounds computed with this method for the five VLs of Figure 2.

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

Table 3. Upper end-to-end delays in μs

Results in Table 3 show that, on this small configuration, the basic Calculus approach is pessimistic (more than 40 μs for nearly all the VLs). The next paragraph presents an improvement of the basic Network Calculus approach in the context of AFDX.

5.2 Optimizing the Network Calculus approach with grouping

The pessimism observed in Table 3 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 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 [Le Boudec & Thiran (2001)]. 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.

On the example in Figure 2, 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 Figure 15 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.

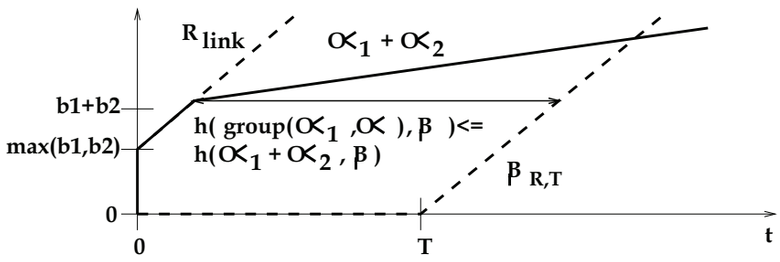


Fig. 15. Grouping flows reduces the maximum delay incurred in an output port

Column NCG in Table 3 gives the upper bounds computed with this technique on the example in Figure 2. 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.

6. AFDX worst-case delay analysis with the Trajectory approach

The Trajectory approach [Martin (2004); Martin & Minet (2006a); Migge (1999)] 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 [Bauer et al. (2009a)]. The improvement of the approach has

been proposed in [Bauer et al. (2009b)]. Main features of the Trajectory approach applied to AFDX are summarized and illustrated in Sections 6.1 and 6.2. The proof of the optimization of the Trajectory approach computation is presented in Section 6.3.

6.1 The main features of the Trajectory approach

The approach developed for the analysis of the AFDX considers the results from [Martin & Minet (2006a)]. A distributed system is composed of a set of interconnected processing nodes. Each flow crossing this system follows a static path which is an ordered sequence of nodes. The Trajectory approach assumes, with regards to any flow τ_i following path $\mathcal{P}_i$, that any flow 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.

Flows are scheduled with a FIFO algorithm in every visited node. Each flow τ_i has a minimum inter-arrival 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. 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 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:

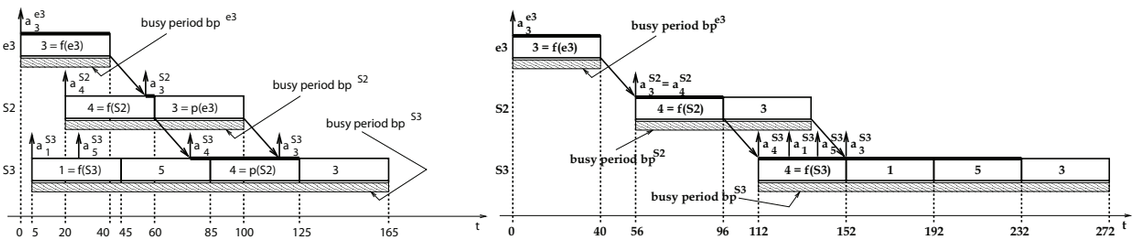
- each node of the system corresponds to an AFDX switch output port, including the output link,
- each link of the system corresponds to the switching fabric,
- each flow corresponds to a VL path.

The assumptions of the Trajectory approach are verified by the AFDX (see Section 2.1). Indeed, switch output ports implement FIFO service discipline. The switching fabric delay is upper bounded by a constant value (16 μs), thus $L = L_{min} = L_{max} = 16\ \mu s$. 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 $R = 100Mb/s$, $C_i^h = C_i = s_{max}/R$ for every node h in the network.

6.2 Illustration on a sample AFDX configuration

Let us consider the sample AFDX configuration depicted in Figure 2. Figure 16(a) 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 Figure 16(a) 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. Due to the FIFO policy, 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.



(a) An arbitrary scheduling of packets (b) Latest starting time of packet 3

Fig. 16. Two possible scheduling of packets

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 Figure 16(a), 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 Figure 16(a), 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 [Martin & Minet (2006a)], 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. $a_{p(N_{i-1})}^{N_i} - a_{f(N_i)}^{N_i}$. On the example in Figure 16(a), 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 Figure 16(a). The starting time of packet 3 in node $S3$ on the example in Figure 16(a) is 125 μs.

It has been shown [Martin & Minet (2006a)] that the latest starting time of a packet m in its last node is reached when $(a^{N_i}_{p(N_{i-1})} - a^{N_i}_{f(N_i)}) = 0$ 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 on the example in Figure 16(a) is illustrated in Figure 16(b). 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 packet 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 Figure 16(b), this worst case end-to-end delay is 232 + 40 = 272 μs. 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 $S3$. On this example, it can be seen that packets 3 and 4 are counted twice. Actually, it has been shown [Martin & Minet (2006a)] 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 Figure 16(b), packet 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 $(a^{N_i}_{p(N_{i-1})} - a^{N_i}_{f(N_i)})$ for every node N_i, as proposed in [Martin & Minet (2006a)]. Let us consider VL $v5$ of the example depicted in Figure 2. 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:

$$a_3^{S3} \leq a_5^{S3} \; and \; a_4^{S3} \leq a_5^{S3} \tag{8}$$

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

$$\left| a_3^{S3} - a_4^{S3} \right| \geq C = 40\mu s \tag{9}$$

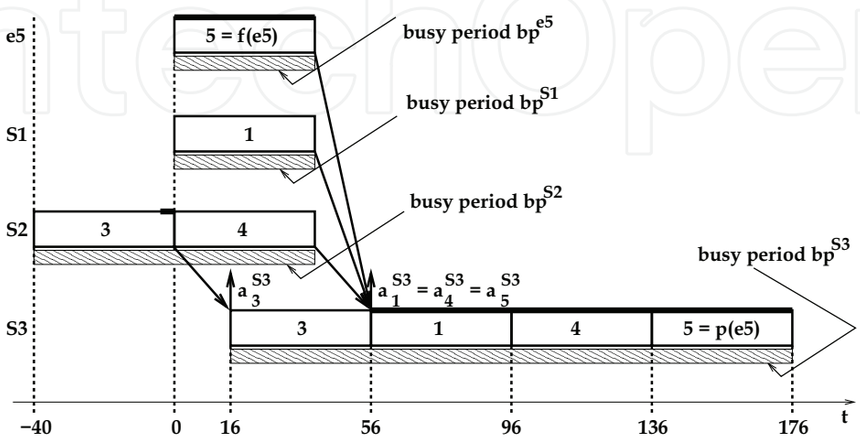


Fig. 17. Latest starting time of VL $v5$

Without loss of generality, let us consider that packet 3 arrives before packet 4. From (8) and (9), we have:

$$a_4^{S3} = a_5^{S3} \; and \; a_3^{S3} = a_5^{S3} - 40 \; \mu s \tag{10}$$

The resulting worst-case scheduling is depicted in Figure 17. $p(e5)$ is packet 5 and f(S3) is packet 3. From (10), 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.

6.3 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 [Martin & Minet (2006a)]. 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:

$$R_i = \max_{t \geq 0} \left(W_{i,t}^{last_i} + C_i - t \right) \tag{11}$$

$last_i$ is the last visited node of flow τ_i and $W_{i,t}^{last_i}$ is a bound on the latest starting time of a packet m generated at time t on its last visited node. The definition of $W_{i,t}^{last_i}$ given in [Martin & Minet (2006a)] becomes:

$$W_{i,t}^{last_i} =$$

$$\sum_{\substack{j \in [1,n] \\ j \neq i \\ \mathcal{P}_j \cap \mathcal{P}_i \neq \varnothing}} \left(1 + \left\lfloor \frac{t + A_{i,j}}{T_j} \right\rfloor \right) \cdot C_j \tag{11}$$

$$+ \left(1 + \left\lfloor \frac{t}{T_i} \right\rfloor \right) \cdot C_i \tag{12}$$

$$+ \sum_{\substack{h \in \mathcal{P}_i \\ h \neq last_i}} \left(\max_{\substack{j \in [1,n] \\ h \in \mathcal{P}_j}} \{ C_j \} \right) \tag{13}$$

$$+ (|\mathcal{P}_i| - 1) \cdot L_{max} \tag{14}$$

$$- \sum_{\substack{N_h \in \mathcal{P}_i \\ N_h \neq first_i}} (\Delta_{N_h}) \tag{15}$$

$$-C_i \tag{16}$$

Where

$$\Delta_{N_h} = a^{N_h}_{p(N_{h-1})} - a^{N_h}_{f(N_h)} \tag{17}$$

Term (11) corresponds to the processing time of packets from every flow τ_i 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 (12) is the processing time, on one node, of the packets of the flow τ_j which are transmitted in the same busy period as m.
Term (13) 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 (14) corresponds to the sum of switching delay.
Term (15) 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. $N_{h\text{-}1}$. This term is null in the context of [Martin & Minet (2006a)].
C_i is subtracted, because $W^{last_i}_{i,t}$ is the latest starting time and not the ending time of the packet from τ_i on its last node.
Solving $R_i = \max(W^{last_i}_{i,t} + C_i - t)$ comes to find the maximum vertical deviation between the function $t \mapsto W^{last_i}_{i,t} + C_i$ and the identity function $t \mapsto t$.
The optimization of this computation in the context of the AFDX concerns Term (15). Indeed, it has been shown in Section 6.2 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 Figure 18.

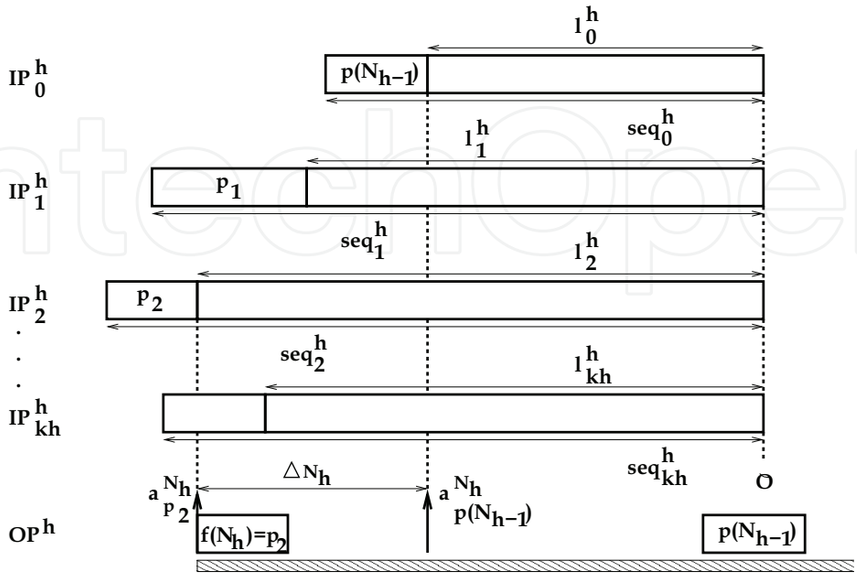


Fig. 18. Illustration of Δ_{N_h}

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 (11) and (12). These packets are grouped by input link. IP_0^h is the input link of τ_i, while $IP_x^h(1 \le x \le k_h)$ are the other input links. Sequence $seq_x^h(0 \le x \le k_h)$ contains the packets of bp^{N_h} coming form IP_x^h. In order to maximize the delay of packet m in node N_h, each sequence $seq_x^h(1 \le x \le k_h)$ is postponed so that it finishes at the same time as sequence seq_0^h. This finishing time is denoted θ in Figure 18. This construction is a generalization of the Trajectory approach presented in [Martin & Minet (2006a)]: instead of postponing individually each packet, sequences of already serialized packets are postponed. As defined in (17), 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 Figure 18, Δ_{N_h} is the difference between the arrival of $p2$ and the arrival of $p(N_{h\text{-}1})$.

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

From (17), 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 $l_x^h(0 \le x \le k_h)$ as the duration of sequence seq_x^h without its first packet. Then, we have:

$$a_{f(N_h)}^{N_h} = \theta - \max_{1 \le x \le k_h} l_x^h \quad and \quad a_{p(N_{h-1})}^{N_h} = \theta - l_0^h \tag{18}$$

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 $1 \le x \le k_h$. It is obtained when the largest packet of sequence seq_x^h is transmitted at the beginning of seq_x^h, for $1 \le x \le k_h$.

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

$$\max_{1 \le x \le k_h} \left(\min\left(l_x^h\right)\right) - \max\left(l_0^h\right) \tag{19}$$

7. Analysis of a realistic configuration

The realistic AFDX network considered in this chapter is composed of two redundant networks [Charara, Scharbarg, Ermont & Fraboul (2006)]. Each network includes 123 end systems, 8 switches, 964 Virtual Links and 6412 VL paths (due to VL multicast characteristics). The left part in Table 4 gives the dispatching of VLs among BAGs. It can be seen that BAGs are harmonic between 2 and 128. The right part in Table 4 gives the dispatching of VLs among frame lengths, considering the maximum length s_{max}. The majority of VLs consider short frames. Table 5 shows the number of VL paths per length (i.e. the number of crossed switches).

BAG (ms)	Number of VL
2	20
4	40
8	78
16	142
32	229
64	220
128	255

Frame length (bytes)	Number of VL
0-150	561
151-300	202
301-600	114
601-900	57
901-1200	12
1201-1500	35
> 1500	3

Table 4. BAGs and frame lengths

Nb of crossed switches	Number of paths
1	1797
2	2787
3	1537
4	291

Table 5. VL paths lengths

The temporal analysis of this realistic configuration has been conducted. Table 6 summarizes the results obtained by the safe upper bound computation. As previously mentioned, the model checking approach cannot be applied on such large scale configurations. 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 realistic 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. Table 6 gives for both approaches the average upper bounds among all the VLs of the configuration, as well as the minimum and the maximum upper bounds. The obtained upper bounds are slightly tighter with the trajectory approach.

	Network calculus	Trajectories
Minimum	0.386 *ms*	0.293 *ms*
Average	4.532 *ms*	4.247 *ms*
Maximum	13.194 *ms*	12.949 *ms*

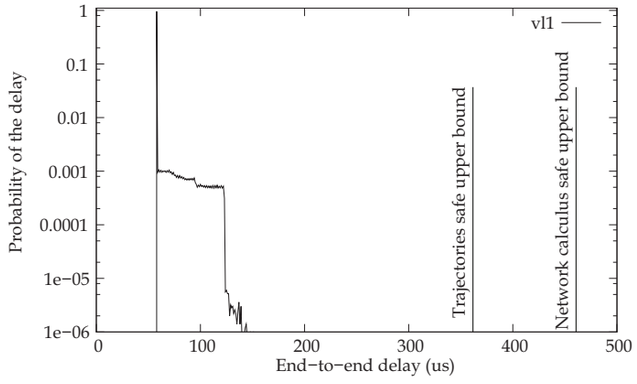
Table 6. Safe upper bounds on the industrial configuration

The end-to-end delay distributions have been computed for the VLs of the realistic configuration, thanks to a home made tool implementing the simulation approach presented in Section 3. Table 7 gives the results obtained for five VL paths. Table 7 gives, for each VL path, the BAG, the minimum and the maximum frame sizes, the number of crossed switches (hops) and the load in each output port. For instance, vl_4 crosses three switches. It shares its end system output port with 5 other VLs. Similarly, vl_4 shares the output port of its first crossed switch with 79 VLs and the output ports of the two consecutive ones with 34 and 49 VLs.

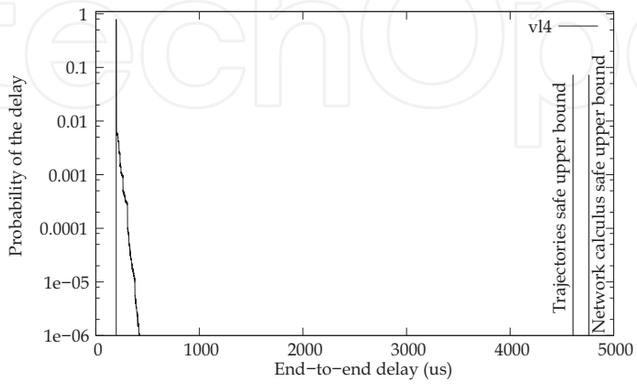
	Characteristics					Delay			
	BAG	s_{min}	s_{max}	Hops	Load	Min	Upper bound		
							Simu	NC	Traj
vl_1	128	263	263	1	8, 6	0.058	0.162	0.364	0.463
vl_2	8	84	467	1	5, 46	0.090	0.248	1.171	1.017
vl_3	64	91	91	1	8, 135	0.030	0.351	3.944	3.554
vl_4	8	84	467	3	5, 79, 34, 49	0.197	0.603	4.752	4.683
vl_5	128	107	107	4	17, 85, 96, 71, 46	0.106	0.759	7.610	7.727

Table 7. Analysis of five typical VLs

Table 7 gives the lower and upper delays observed by simulation for each VL (columns *Min* and *Simu*) and the safe upper bounds computed by the network calculus and trajectory approaches (columns *NC* and *Traj*). For each VL, the lower delay observed by simulation corresponds to the minimum possible value of the delay. The delay distributions for vl_1 and vl_4 are depicted in figures 19(a) and 19(b). It appears that the delay distribution is close to the minimum possible value of the delay and far from the the safe upper bound computed by either the network calculus or the trajectory approach. This is mainly due to the fact that the AFDX network is lightly loaded. Thus, the probability that several frames reach the same output port at the same time is very low.



(a) Delay analysis of vl_1



(b) Delay analysis of vl_4

Fig. 19. Delay analysis of two VLs

8. Conclusion

This chapter gives an overview of the temporal analysis of switched Ethernet avionic networks. Today, three approaches exist for the computation of a safe upper bound of the end-to-end delay of each flow transmitted on the avionic network. The first approach is based on model checking and allows the computation of the exact worst-case delay of each flow, but it is limited to small configurations, due to the combinatory explosion problem. The two other approaches are based on trajectories and network calculus and allow the computation of a safe upper bound of the end-to-end delay, which is most of the time larger than the exact worst-case, due to the pessimistic assumptions made by the two approaches. Nevertheless, these two approaches can be applied to industrial configurations. The computation of a safe upper bound is complemented by the evaluation of the end-to-end delay distribution, thanks to a simulation approach.

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 [Grenier et al. (2008)]. 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 a periodic traffic in the automotive context [Khan et al. (2009)]. This leads to a probabilistic analysis of the worst case delay of flows. Such an analysis has been proposed [Scharbarg et al. (2009)], based on a stochastic Network Calculus approach [Vojnović & Le Boudec (2002; 2003)].

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 [Parekh & Gallager (1993)]. The introduction of static priority queueing in the stochastic Network Calculus approach has been presented in [Ridouard et al. (2008)]. The Trajectory approach is promising for handling heterogeneous flows needing QoS aware servicing policies at switches level [Martin & Minet (2006a;b)].

9. References

Alur, R. & Dill, D. L. (1994). Theory of Timed Automata, *Theoritical Computer Science* 126(2): 183–235.

ARI (1991). ARINC 651, Aeronautical Radio Inc. ARINC specification 651. Design Guidance for Integrated Modular Avionics., *Technical report*, Aeronotical Radio Inc.

ARI (1997). ARINC 653, Aeronautical Radio Inc. ARINC specification 653. Avionic application Software Standard Interface., *Technical report*, Aeronotical Radio Inc.

ARI (2001). ARINC 429, Aeronautical Radio Inc. ARINC specification 429. Digital Information Transfer System (DITS) parts 1,2,3., *Technical report*, Aeronotical Radio Inc.

ARI (2002-2005). ARINC Specification 664: Aircraft Data Network, Parts 1,2,7, *Technical report*, Aeronotical Radio Inc.

Bauer, H., Scharbarg, J.-L. & Fraboul, C. (2009a). Applying and optimizing trajectory approach for performance evaluation of afdx avionics network, *Proceedings of the 21th ECRTS WiP section*, Dublin, pp. 57,60. http://gerfaut83.free.fr/article/Bauer-Scharbarg-Fraboul.pdf.

Bauer, H., Scharbarg, J.-L.&Fraboul, C. (2009b). Applying trajectory approach to afdx avionics network, *Proc. of the 14th International Conference on Emerging Technologies and Factory Automation*, IEEE, Mallorca, pp. 1–8.

Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L. & Schnoebelen, Ph. (2001). *Systems and Software Verification. Model-Checking Techniques and Tools*, Springer.

Burgueño Arjona, A. (1998). *Vérification et synthèse de systèmes temporisés par des méthodes d'observation et d'analyse paramétrique*, PhD thesis, Ecole Nationale Supérieur de l'Aéronautique et de l'Espace, Toulouse, France.

Chang, C.-S. (2000). *Performance Guarantees in Communication Networks*, Springer-Verlag, London, UK. isbn = {1852332263}.

Charara, H., Scharbarg, J.-L., Ermont, J. & Fraboul, C. (2006). Methods for bounding end-to-end delays on an AFDX network, *Proceedings of the 18th ECRTS*, Dresden, Germany, pp. 193–202.

Charara, H., Scharbarg, J.-L. & Fraboul, C. (2006). Focusing simulation for end-to-end delays analysis on a switched Ethernet, *Proceedings of the WiP session of RTSS*, Rio de Janeiro, Brasil.

Grenier, M., Havet, L. & Navet, N. (2008). Scheduling messages with offsets on Controller Area Network: a major performance boost, *in* N. Navet & F. c. Simonot-Lion (eds), *The automotive embedded systems handbook*, Taylor and Francis, chapter 14.

Khan, D., Navet, N., Bavoux, B. & Migge, J. (2009). Aperiodic traffic in response time analysis with adjustable safety level, *Proc. of the 14th International Conference on Emerging Technologies and Factory Automation*, IEEE, Palma de Mallorca, Spain.

Larsen, K. G., Pettersson, P. & Yi, W. (1997). UPPAAL in a Nutshell, *International Journal on Software Tools for Technology Transfer* 1(1–2): 134–152.

Le Boudec, J.-Y. & Thiran, P. (2001). *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, Vol. 2050 of *Lecture Notes in Computer Science*, Springer-Verlag. ISBN: 3-540-42184-X.

Martin, S. (2004). *Maîtrise de la dimension temporelle de la qualité de service dans les réseaux*, PhD thesis, Université Paris XII.

Martin, S. & Minet, P. (2006a). Schedulability analysis of flows scheduled with fifo: application to the expedited forwarding class, *20th International parallel and distributed processing symposium*, Rhodes Island, Greece.

Martin, S. & Minet, P. (2006b). Worst case end-to-end response times of flows scheduled with fp/fifo, *5th IEEE International Conference on Networking*, Mauritius.

Migge, J. (1999). *L'ordonnancement sous contraintes temps-réel: un modèle à base de trajectoires*, PhD thesis, INRIA, Sophia Antipolis France.

Parekh, A. & Gallager, R. (1993). A generalised processor sharing approach to flow control in integrated services networks: the single-node case, *IEEE/ACM transactions on networking* 1(3): 344–357.

Ridouard, F., Scharbarg, J.-L. & Fraboul, C. (2008). Probabilistic upper bounds for heterogeneous flows using a static priority queueuing on an AFDX network, *Proc. of the 13th International Conference on Emerging Technologies and Factory Automation*, IEEE, Hambourg, pp. 1220–1227.

Scharbarg, J.-L. & Fraboul, C. (2007). A generic simulation model for end-to-end delays evaluation on an avionics switched Ethernet, *Proc. of the 7th International Conference on Fieldbuses and networks in industrial and embedded systems*, IFAC, Toulouse, France.

Scharbarg, J.-L., Ridouard, F. & Fraboul, C. (2009). A probabilistic analysis of end-to-end delays on an AFDX network, *IEEE transactions on industrial informatics* 5(1).

Vojnović, M. & Le Boudec, J. (2002). Stochastic analysis of some expedited forwarding networks, *in Proceedings of Infocom, New-York*.

Vojnović, M. & Le Boudec, J. (2003). Bounds for independent regulated inputs multiplexed in a service curve network element, *IEEE Trans. on Communications* 51(5).

New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems
Edited by Meng Joo Er

ISBN 978-953-307-213-5
Hard cover, 438 pages
Publisher Sciyo
Published online 02, November, 2010
Published in print edition November, 2010

The grandest accomplishments of engineering took place in the twentieth century. The widespread development and distribution of electricity and clean water, automobiles and airplanes, radio and television, spacecraft and lasers, antibiotics and medical imaging, computers and the Internet are just some of the highlights from a century in which engineering revolutionized and improved virtually every aspect of human life. In this book, the authors provide a glimpse of the new trends of technologies pertaining to control, management, computational intelligence and network systems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Jean-Luc Scharbarg and Christian Fraboul (2010). Methods and Tools for the Temporal Analysis of Avionic Networks, New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems, Meng Joo Er (Ed.), ISBN: 978-953-307-213-5, InTech, Available from: http://www.intechopen.com/books/new-trends-in-technologies--control--management--computational-intelligence-and-network-systems/methods-and-tools-for-the-temporal-analysis-of-avionic-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen