Forward End-To-End delay Analysis for AFDX networks

Nassima Benammar, Frederic Ridouard, Henri Bauer
and Pascal Richard
e-mail: {nassima.benammar,frederic.ridouard,
pascal.richard}@univ-poitiers.fr henri.bauer@ensma.fr

April 14, 2017

Abstract

Packet switched networks and message multiplexing have been a major upgrade for industrial systems communications. In the avionics domain, this evolution was brought by the introduction of Avionics Full Duplex Switched Ethernet (AFDX). Guaranteed upper bounds of end-to-end delays for messages transmitted over an AFDX network are mandatory for certification reasons.

In this article, we present the Forward end-to-end delay Analysis (FA), which is scalable to both large and heavily loaded configurations. A formal proof of FA is detailed and the approach is compared with alternative methods (Network Calculus and the Trajectory approach) on two types of AFDX configurations (including a real industrial architecture).

1 Introduction

Over the last decades, the Real-Time systems have widely spread in many domains and their complexity has increased substantially. The augmentation of computation demand and cost reduction efforts has push forward distributed architectures, sharing resources in the form of processing units interconnected by communication channels.

A direct consequence of the evolution towards more distributed systems is a higher amount of data exchange. Real-time capable networks offering large bandwidth capacity over a shared medium have been proposed in the avionics field, such as the Avionics Full Duplex Switched Ethernet [1], used as a backbone network in most of recent civilian aircraft, or TTEthernet [2] at the core of the new NASA crew vehicle (ORION).

Network determinism is mandatory to ensure that end-to-end timing constraints are met. In particular, the worst-case end-to-end (ETE) latency of each frame sent over the network has to be determined. Such a worst-case is hard to track when the network components are asynchronous such as for AFDX. Several approaches have been developed in order to determine guaranteed upper bounds of ETE delays. Even though upper bounds are sufficient for certification purpose, they often imply oversizing when used for network dimensioning.

The contribution of this paper is to propose a new approach for computing ETE delays in AFDX networks, named *Forward Analysis*. It can be generalized to asynchronous packet networks with FIFO queuing. It has been improved in comparison with author's prior results [3] and is rigorously proved in this paper. Moreover, the approach is benchmarked against state of the art methods on a real-world avionics case-study.

This paper is organized as follows. In Section 2, we present existing worst-case end-to-end computation frameworks in the context of packet networks. The detail of AFDX and its modeling are given in Section 3. FA is presented in Section 4. Finally, in Section 5, experimentation and comparison are conducted on both sample and industrial configurations. Conclusion and further work are presented in Section 6.

2 Related work and motivations

Determining worst-case ETE delays of frames in a packet network is mandatory for temporal validation of distributed systems. However, with fully asynchronous sources and switching elements, tracking the worst-case scenario along a multi-hop path with several queuing elements can be hard. Thus, Model Checking (MC) approaches are not applicable as they do not scale up with large industrial configurations [4] due to combinatorial explosion. Some work has been conducted to improve the ambit of MC through state space reduction [4] or by considering timing correlation between traffic flows [5], but still not in line with full-fledged avionics configurations [6].

An alternative to worst-case scenarios tracking is to compute upper-bounds of the worst-case ETE delays by making conservative, and thus, pessimistic hypothesis. Finding a computation minimizing this pessimism will help keeping as low as possible the cost of overdimensioning. We compare hereafter several conservative computation techniques.

Network Calculus (NC) [7] is the current reference for certification of AFDX networks in civilian aircraft [8]. It can be applied [9] to other Ethernet based networks such as Audio Video Bridging (AVB) [10]. In NC, the upper bound is obtained as a sum of local worst-case scenarios computed with traffic envelopes. Since these scenarios cannot be experienced simultaneously, and since the envelope are conservative, the obtained upper bounds can be pessimistic. A stochastic extension of NC has been formalized in [11], but the results are not yet relevant for real-world systems.

The holistic approach (HA) also computes local delays in switches [12] and iteratively updates jitters until a fix point is reached (i.e., worst-case ETE upper bounds have been computed). Some work in the context of packet networks has been conducted to cope with ETE delays [13], but without formal proof of correctness. In [14] the holistic approach has been applied to analyze AFDX End Systems.

The Trajectory Approach (TA) proposed in [13,15] takes advantage of scheduling analysis results, by making an analogy between frames being served in a multiplexing point and non-preemptive task scheduling in a uniprocessor. The main idea is to overcome the "sum of local optimum" problem mentioned for NC and HA by concatenating all network elements crossed by a traffic flow into a single global node (hence the name of the approach). However, this assump-

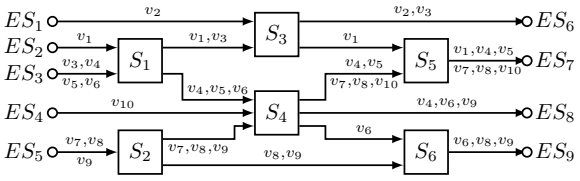

Figure 1: An AFDX configuration

tion induces an important limitation: it cannot determine an upper bound of the ETE delay for a flow with a cumulative load along its path higher than 100%. TA has not been formally proved in previously cited work, and Kemayo *et al.* [16] showed that TA can compute optimistic bounds in some corner cases. The method has been patched since [17], but the correction is also lacking a formal proof of correctness.

The limitations in existing methods motivated the research for another method for evaluating worst-case ETE delays in packet networks. Similarly to TA, the *Forward End-to-End delays Analysis* (FA) [3,18] leverages existing results about non-preempting task scheduling in the uniprocessor context. The most significant contributions of our approach compared with TA is the global load limitation being lifted and a formal proof of correctness.

Among the previously cited methods, NC and TA have been improved to take into account the so called *serialization effect*: frames transmitted via a same link cannot arrive simultaneously in the destination node of this link. They are necessarily delivered sequentially. This improvement has shown to produce less pessimistic ETE delays for both NC [8] and for TA [19,20]. We show (see Section 4.4) that our approach (FA) can benefit from this improvement within a same order of magnitude.

3 AFDX and network model

AFDX [1] is a switched packet network based on Ethernet. The AFDX ingress/Egress points are called End Systems (ES). The ESs are interconnected by a set of switches and physical links. The traffic is mapped on logical communication channels called Virtual Links (VL). A sample AFDX configuration is depicted in Fig. 1. It is made up of six switches and nine ESs exchanging frames through ten VLs.

A VL is an unidirectional connection between a source ES and one or more destination ESs. For example, in Fig. 1, multicast VL v_4 starts at ES_3, crosses switches S_1 and S_4, then splits towards two destination ESs: ES_7 (via S_5) and ES_8. A VL defines static routes with predetermined bandwidth allocation in the form of a maximum frame size ($Fmax$) and a minimum time between the generation of two consecutive frames in the source ES (BAG), also called *Bandwidth Allocation Gap*. The deterministic behavior of AFDX is ensured by traffic shaping and policing units in each node. Switches and End Systems are fully asynchronous: there is no global clock or synchronization protocol. Full Duplex links guarantee the absence of collisions between frames. The entire multiplexing cost is thus deferred to the servicing queues.

AFDX switches operate as *store-and-forward* between input and output

ports, applying traffic policing and routing according to the specification of the VLs. The switching fabric latency (L) is estimated at 16 μs [8]. An output port is a work-conserving queue, servicing frames with a First-In, First-Out (FIFO) policy. An ES has no input ports but has a single output port with a FIFO buffer. The servicing rate of an output port is constant (mostly 100 Mbps, 10 Mbps for a few of them).

FA has been thought for AFDX, but can be used for any kind of packet network compliant with the model defined hereafter. It considers a network composed of a set of nodes $\mathcal{S}$ interconnected through logical links. A node h has a set of input links, a single FIFO buffer with a servicing rate r^h connected to one or multiple destination nodes through output links. As for AFDX, each multiplexing point (*i.e.* a switch output port or an ES) corresponds to a node.

At its arrival time in the node, a frame is stored in an input buffer, before being forwarded to its destination nodes (see Fig. 2). The communications exchanged between nodes are modeled by a set of sporadic, unicast and unidirectional flows denoted $\Gamma = \{v_1, v_2, \ldots, v_n\}$. Let Γ_h be the set of flows crossing a node h. These flows are the counterpart of Virtual Links in the AFDX context. As VLs can be multicast, flows modeling a VL have to be duplicated at each splitting point of the multicast tree.

A frame from a flow v_i starts its transmission in a source node denoted $first_i$. It crosses successive nodes until its destination node, denoted $last_i$. The path $\mathcal{P}_i$ of a flow v_i is statically defined as the ordered list of nodes from $first_i$ to $last_i$. In the context of AFDX, $last_i$ often corresponds to the output port of last switch crossed by v_i, since the message processing delay in the destination ES (depending on the application running on it) is not accounted in the computation.

In addition to its path, a flow v_i is characterized by:

- $Fmax_i$, the maximum frame size (in bits) for a flow v_i, corresponding to a maximum transmission time in a node h (of servicing rate r^h) equal to $C_i^h = \frac{Fmax_i}{r^h}$;
- T_i, the minimum interval between the generation time of two consecutive frames from flow v_i in node $first_i$. It corresponds to the AFDX BAG.

The delay incurred by a frame of a flow v_i, as depicted in Fig. 2, is made up of :

- a **technological latency**, denoted L, including the electrical propagation time (fairly negligible) and the switching delay of 16 μs at each hop ;
- a highly variable **waiting time** spent in each node from its path $\mathcal{P}_i$, conditioned by the amount of pending frames (or *backlog*) in the buffer at its arrival time in the given node, including its own **transmission time** C_i^h.

The aim of the FA computation is to upper-bound these waiting times in order to compute worst-case ETE delays.

4 Forward Analysis

In this section, we present the *Forward End-to-End delay Analysis* (FA). The aim of this method is to determine upper bounds of worst-case ETE transmission

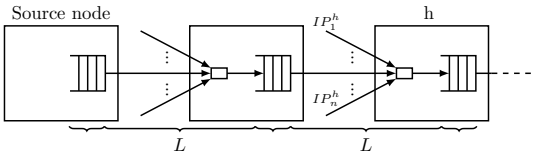

Figure 2: Elements of the ETE delay

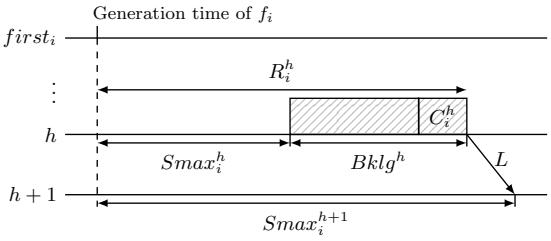

Figure 3: Iterative computation of R_i^h and $Smax_i^h$

delays, based on the packet network model exposed in Section 3.

4.1 Principles

To determine the ETE delay of a flow, all the nodes belonging to its path are analyzed sequentially. A worst-case ETE transmission delay of a frame from its departure node up to a given node is computed, and the computation is propagated in a data-flow manner, up to its destination node.

The worst-case traversal delay of a frame of a flow v_i from its ingress node ($first_i$) up to a given node h from $\mathcal{P}_i$, denoted R_i^h (see Fig. 3), is composed of a worst-case traversal time until node h, plus a worst-case delay incurred in node h :

$$R_i^h = Smax_i^h + Bklg^h \tag{1}$$

We detail hereafter the computation of both parts of R_i^h.

Definition 1 *A temporal definition of the backlog in an output buffer is the total transmission time of all the pending frames in that buffer at a given time t. We denote by $Bklg^h$ an upper bound on the worst-case backlog in the output buffer of node h at any time.*

Definition 2 *For a node h from $\mathcal{P}_i$, the term $Smax_i^h$ is the maximum delay incurred by a frame of v_i, from its generation time in node $first_i$, before entering node h.*

By definition: $Smax_i^{first_i} = 0$. $Smax_i^h$ is then determined iteratively. Let h and $h+1$ denote two consecutive nodes from $\mathcal{P}_i$. By construction, $Smax_i^{h+1}$ (see Fig. 3) is the sum of the delay suffered to reach h ($Smax_i^h$), the maximum queuing time in h ($Bklg^h$) and the technological latency L:

$$\begin{cases} Smax_i^{first_i} & = 0 \\ Smax_i^{h+1} & = Smax_i^h + Bklg^h + L \end{cases} \tag{2}$$

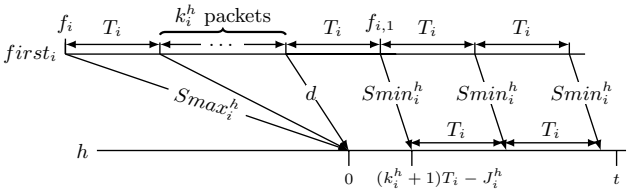

Figure 4: The worst-case scenario for the arrival of frames from $v_i \in \Gamma_h$ in a node h, maximizing the backlog during a time interval $[0, t]$.

It is worthy noting that $Bklg^h$ includes the transmission time C_i^h of the frame under study. Similarly, the shortest time to reach a node h for a frame from flow v_i considering no interference in each node is:

$$\begin{cases} Smin_i^{first_i} & = 0 \\ Smin_i^{h+1} & = Smin_i^h + C_i^h + L \end{cases} \tag{3}$$

Finally, the worst-case ETE delay for a flow v_i is denoted R_i, which corresponds, by definition to $R_i^{last_i}$.

The main difficulty is to determine $Bklg^h$ in each node h. Thus, for a node h, we establish the worst-case scenario maximizing the contribution of each flow v_i ($v_i \in \Gamma^h$) in $Bklg^h$. This problem is studied in Section 4.2. In Section 4.3, this scenario is described with the concept of *Request Bound Functions* (RBF). Further optimization considering the serialization effect is brought in Section 4.4. The final expression of the maximum backlog is given in Section 4.5 and a sufficient condition to stop the computation is given in Section 4.6. Finally, the whole method is summed up in Section 4.7 and expressed in the form of an algorithm.

4.2 Worst-case scenario to determine the maximum backlog

In order to determine $Bklg^h$, we consider the starting time of node h as time 0 (no frame arrives in h before time 0). $Bklg^h$ is the maximum backlog for any time $t \geq 0$. Therefore, we analyze the time interval $[0, t]$, looking for the maximum backlog that can be generated by each flow $v_i \in \Gamma_h$ in order to maximize the global amount of backlog for any $t \geq 0$.

Theorem 1 *In a FIFO buffer, the worst-case backlog generated by a flow v_i in a node h from $\mathcal{P}_i$ during a time interval $[0, t]$ ($t \geq 0$) is obtained when:*

(i) the frames of v_i are generated periodically every T_i in the source node ($first_i$);

(ii) the first frame, denoted f_i, generated by v_i arrives in h at time 0;

(iii) f_i reaches node h, suffering its maximum traversal delay ($Smax_i^h$) whereas all the subsequent frames from v_i arrive on h suffering their minimum traversal delay so that their arrival time is never before time 0 (see Fig. 4).

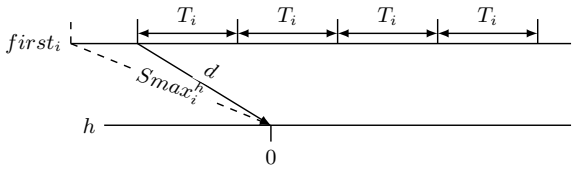

Figure 5: First frame reaching h suffering a delay $d < Smax_i^h$.

Proof: The maximum backlog is obtained when the flows generate the maximum interfering traffic. For sporadic flows, this is achieved when frames are generated with their minimal inter-generation time on their source node. Hence, for a flow v_i, this corresponds to the generation of a new frame every T_i in its source node $first_i$, which proves item (i).

To maximize its contribution in a time interval $[0, t]$, frames from v_i have to arrive in node h as soon as possible. Point (ii) can be proved by contradiction. If the first frame f_i arrives on h strictly after 0, it is always possible to find a worse scenario by moving f_i forward closer to 0. Therefore, to determine the maximum backlog, the first frame f_i has to enter node h exactly at time 0.

The case of the first frame f_i from point (iii) is also proved by contradiction: consider that f_i arrives at time 0 but suffers a delay $d < Smax_i^h$ to reach h, as depicted in Fig. 5. Since it suffers a delay shorter than $Smax_i^h$, its generation time in its source node $first_i$ has to be delayed (by $Smax_i^h - d$). Moreover, since the frame generation is strictly periodic, as stated in (i), the generation time of all the subsequent frames are also delayed. Considering that they cross the network with a minimum traversal time ($Smin_i^h$), their arrival time on h is also delayed. Therefore, it is possible to define a worse scenario by moving forward the generation time of f_i.

Frame f_i reaches node h at time 0 with a delay equal to $Smax_i^h$, but for the subsequent frames, this delay can range between $Smin_i^h$ and $Smax_i^h$. This means that one or more consecutive frames can catch up, but without overtaking, due to the FIFO constraint.

The scenario, where k_i^h frames (with $k_i^h \geq 0$) have caught up f_i, is illustrated in Fig. 4. These k_i^h frames have necessarily incurred shorter delays to catch up f_i. This delay cannot be shorter than $Smin_i^h$, but it may be larger for some of them as there can be no overtaking.

In Fig. 4, we denote by $f_{i,1}$ the first frame arriving after the k_i^h frames incoming at time 0. In order to maximize the backlog, that is, to minimize the time interval with previous frames, it has to arrive suffering the minimum delay $Smin_i^h$ to reach h. Moreover, since the following frame cannot suffer a delay shorter than $Smin_i^h$, they arrive periodically in h, with an interval of T_i. □

The number k_i^h of frames arriving together with frame f_i at time 0 depends on the flow characteristics (T_i, $Smin_i^h$ and $Smax_i^h$). It is computed in Theorem 2.

Theorem 2 *In a FIFO buffer, considering the worst-case scenario from Theorem 1, the first frame f_i generated by flow v_i ($v_i \in \Gamma_h$) reaches node h simulta-*

neously with k_i^h others frames. $k_i^h \in \mathbb{N}$ is defined such as:

$$k_i^h T_i \leq J_i^h < (k_i^h + 1)T_i \quad \textit{that is:} \quad k_i^h = \left\lfloor \frac{J_i^h}{T_i} \right\rfloor \tag{4}$$

with $J_i^h = Smax_i^h - Smin_i^h$ being the worst-case jitter of v_i in h.

Proof: We prove separately the two inequalities $k_i^h T_i \leq J_i^h$ and $J_i^h < (k_i^h + 1)T_i$:

- Let us consider the last frame from v_i arriving in node h at time 0. According to the periodic scenario defined in Theorem 1 and depicted in Fig. 4, its generation time in $first_i$ is $0 - Smax_i^h + k_i^h T_i$. As it suffers a delay $d \geq Smin_i^h$ in order to reaches h and its arrival time in h is still 0, we have:

$$\begin{aligned} -Smax_i^h + k_i^h T_i + d &= 0 \\ -Smax_i^h + k_i^h T_i + Smin_i^h &\leq -Smax_i^h + k_i^h T_i + d = 0 \\ -Smax_i^h + k_i^h T_i + Smin_i^h &\leq 0 \\ k_i^h T_i &\leq Smax_i^h - Smin_i^h = J_i^h \\ k_i^h T_i &\leq J_i^h \end{aligned}$$

- We now focus on $f_{i,1}$, the first frame arriving in node h strictly after 0. It is generated at time $0 - Smax_i^h + (k_i^h + 1)T_i$ in $first_i$ and suffers a delay equal to $Smin_i^h$ to reach h:

$$\begin{aligned} 0 &< -Smax_i^h + (k_i^h + 1)T_i + Smin_i^h \\ Smax_i^h - Smin_i^h &< (k_i^h + 1)T_i \\ J_i^h &< (k_i^h + 1)T_i \end{aligned}$$

Hence the proof. □

The arrival date of $f_{i,1}$ is deduced from the next corollary.

Corollary 1 *Considering the worst-case scenario described in Theorem 1, the arrival date, of the first frame from a flow $v_i \in \Gamma_h$ in node h after time 0, denoted $f_{i,1}$ in Fig. 4, is: $(k_i^h + 1)T_i - J_i^h$.*

Proof: This is a direct result of Theorems 1 and 2, since the generation time of $f_{i,1}$ in $first_i$ is equal to $-Smax_i^h + (k_i^h + 1)T_i$ and it suffers a delay equal to $Smin_i^h$ to reach h. □

In a FIFO buffer, the worst-case backlog generated by a flow v_i in a node h from $\mathcal{P}_i$ during a time interval $[0, t]$ $(t \geq 0)$ is maximized considering the scenario from Theorem 1.

In the next section, we express this worst-case in terms of Request Bound Functions (RBF), in order to determine the maximum possible backlog in the buffers.

4.3 Workload and Request Bound Functions

The Request Bound Function $rbf_i^h(t)$ defines the total transmission time of frames arriving in node h generated by a flow $v_i \in \Gamma_h$ during an interval of length t. We denote by $W^h(t)$ an upper bound on the workload, the cumulative transmission time of all the flows crossing h during a time interval of length t:

$$W^h(t) = \sum_{v_i \in \Gamma_h} rbf_i^h(t) \tag{5}$$

The expression of $rbf_i^h(t)$ can be derived directly from the worst-case scenario defined in Theorem 1.

Theorem 3 *Considering a FIFO node h and a flow v_i crossing h, the Request Bound Function (RBF) in an interval of length t is:*

$$rbf_i^h(t) = \left(1 + \left\lfloor \frac{t + J_i^h}{T_i} \right\rfloor\right) C_i^h \tag{6}$$

where J_i^h is the maximum jitter of the flow v_i on h ($J_i^h = Smax_i^h - Smin_i^h$).

Proof: We study several cases, according to the value of t:

- For $t = 0$, $(1+k_i^h)$ frames from v_i arrive at time 0, as proved in Theorem 1:

$$rbf_i^h(0) = \underbrace{\left(1 + \left\lfloor \frac{0 + J_i^h}{T_i} \right\rfloor\right)}_{k_i^h+1 \text{ frames}} C_i^h$$

- For $0 \leq t < (k_i^h + 1)T_i - J_i^h$, no additional frame is accounted since:

$$\begin{aligned} rbf_i^h(t) &= \left(1 + \left\lfloor \frac{t + J_i^h}{T_i} \right\rfloor\right) C_i^h \\ &< \left(1 + \left\lfloor \frac{(k_i^h + 1)T_i}{T_i} \right\rfloor\right) C_i^h \\ &< (k_i^h + 2)C_i^h \end{aligned}$$

- The next incoming frame from v_i arrives at time $(k_i^h + 1)T_i - J_i^h$. Finally, all the following frames arrive with an interval of T_i, as depicted in Fig. 4.

Summarizing these cases leads to Formula (6). □

It is worth noting that Formula (6), applied in a source node, also corresponds to the Request Bound Function for non-preemptive tasks in a uniprocessor system (*i.e.* $J_i^{first_i} = 0$ since $Smin_i^{first_i} = Smax_i^{first_i} = 0$).

The worst-case workload computed with RBFs can however be improved considering the effect of flow serialization.

4.4 The serialization effect

In packet networks, frames are necessarily serialized when being sent over a physical link at a given rate r. Frames sharing a common link cannot therefore arrive simultaneously in the subsequent node: their arrival times are delayed by at least their transmission time at rate r. This is what is commonly called the *serialization effect* [8].

For example, considering Fig. 4, the (k_i^h+1) first frames from $v_i \in \Gamma_h$ cannot arrive simultaneously in h since they share the same input link. Taking this into account can lead to a significant improvement for the tightness of the worst case bounds. Note that the serialization can only occur in non-source nodes, since it is accounted in the input links of the nodes.

We thus complete our model for non-source nodes, in order to distinguish flows based upon their input link. Such a node (let it be called h), with n input links denoted IP_x^h (with $1 \leq x \leq n$), is described in Fig. 2. The global workload $W^h(t)$ is then obtained by summing the contribution of each input link:

$$W^h(t) = \sum_{x=1}^{n} W_x^h(t) \tag{7}$$

where $W_x^h(t)$ denotes the cumulative transmission time of frames from flows incoming in node h through the input link IP_x^h during a time interval of length t.

The fixed rate of the link ensures a maximum rate of frame arrival and implies a shaping of the expression of $W_x^h(t)$, which can thus be determined as the minimum of two terms :

- the cumulative transmission time of frames during an interval of length t, regardless of the serialization effect:

$$W_x^h(t) \leq \sum_{v_i \in \Gamma_x^h} rbf_i^h(t) \tag{8}$$

  where Γ_x^h designates the set of flows entering node h through the input link IP_x^h (with $1 \leq x \leq n$);

- the maximum amount of incoming workload according to the rate of the input link, as determined in Lemma 1.

Lemma 1 *The worst-case cumulative transmission time originating from an input link IP_x in a node h during a time interval t, denoted $W_x^h(t)$, is:*

$$W_x^h(t) \leq \frac{r^{h_x\text{-}1}}{r^h} t + \max_{v_i \in \Gamma_x^h} C_i^h \tag{9}$$

where $(h_x\text{-}1)$ denotes the predecessor of node of h via IP_x^h.

Proof: Nodes $(h_x\text{-}1)$ and h are directly connected through a physical link. Therefore, the amount of traffic arriving on h via IP_x^h during a time interval of length t corresponds to the number of frames having been fully served at rate $r^{h_x\text{-}1}$, as depicted in Fig. 6:

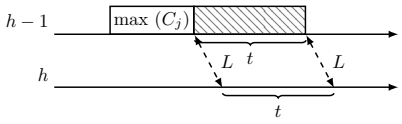

Figure 6: Worst-case cumulative transmission time of frames entering a node h through an input link during a time interval of length t.

(i) during a time interval t, the total amount of traffic served at rate $r^{h_x\text{-}1}$ is $r^{h_x\text{-}1} \times t$ (shaded area in Fig. 6);

(ii) in the worst-case, we consider that a frame of maximal size has been transmitted at the very start of the time interval: $r^{h_x\text{-}1} \times \max_{v_i \in \Gamma_x^h} C_i^{h_x\text{-}1}$

The corresponding workload in node h at rate r^h is thus: $\frac{r^{h_x\text{-}1}}{r^h} \times t$ for part (i), and the transmission time of the frame of maximal size arriving from IP_x^h, which is, by definition,
$\frac{r^{h_x\text{-}1}}{r^h} \times \max_{v_i \in \Gamma_x^h} C_i^{h_x\text{-}1} = \max_{v_i \in \Gamma_x^h} C_i^h$, for part (ii). □

Theorem 4 *The cumulative workload arriving in a non-source node h through the input link IP_x^h during a time interval t is bounded by:*

$$W_x^h(t) = \min\left(\sum_{v_i \in \Gamma_x^h} rbf_i^h(t),\ \frac{r^{h_x\text{-}1}}{r^h}t + \max_{v_i \in \Gamma_x^h} C_i^h\right)$$

Proof: Direct application of Formulas (8) and (9), which define two guaranteed upper bounds of the value $W_x^h(t)$. □

4.5 The maximum backlog

During a time interval $[0, t]$, the maximum cumulative workload $W^h(t)$ incoming in a node h is fully determined by the Formula (7) and Theorem 4. But, as the nodes are work-conserving, they start serving frames as soon as they arrive in the buffers. The maximum backlog in a node h is thus obtained by computing the difference between the incoming workload $W^h(t)$ and the amount of traffic serviced at the rate of the node during any time interval $[0, t]$, which is of length t:

$$Bklg^h = \max_{t \geq 0} \left(W^h(t) - t\right) \tag{10}$$

4.6 Sufficient condition to stop the computation

For each node h, all the intervals of length $t \geq 0$ have to be tested to determine the maximum backlog $Bklg^h$. Here we give a sufficient condition on the maximum value of t to be tested, in order to compute $Bklg^h$. To determine this condition, we use the concept of busy period.

Definition 3 *A busy period [21] is a time interval between two consecutive idle times. An idle time, is a time such as all previously arrived frames have been served at that time (no remaining backlog).*

No frame arrived before an idle time can impact the amount of backlog after it. If an idle time is present in a given time interval, the computation can be stopped, since for any scenario with a larger interval length no new frame arrival will generate more backlog. As a consequence, the computation is limited to the first busy period starting at time 0 (node start time)

An idle time occurs at a time t on a node h, if the amount of incoming workload $W^h(t)$ is less than or equal to the transmission time t: $W^h(t) \leq t$. We now define a sufficient condition to ensure the occurrence of such an idle time in the node h based on its load.

Theorem 5 *Let us consider a node h with a FIFO servicing policy where the frames arrive according the scenario defined in Theorem 1. The existence of an idle time in h is guaranteed if the following condition on its local load (U^h) is verified:*

$$U^h = \sum_{v_i \in \Gamma_h} \frac{C_i^h}{T_i} < 1 \tag{11}$$

Proof: The arrival pattern of every flow $v_i \in \Gamma^h$ starts with an initial burst at time 0 ($k_i^h + 1$ frames) and it becomes periodic from time $(k_i^h + 1)T_i - J_i^h$ (see Theorem 1). At time $\max_{v_i \in \Gamma^h}(k_i^h + 1)T_i - J_i^h$, the system is periodic and, in particular, the frames arrive in h as a repeating cycle. At this time instant, the initial bursts can be not fully transmitted. Considering that a node is similar to a non-preemptive uniprocessor scheduling problem, if $U^h < 1$, during a cycle, there are some idle times.

If $U^h < 1$, the overload generated by the initial burst will eventually be transmitted during the idle times present in the repeating cycle, regardless of its size. Thus, an idle time on node h will occur. □

4.7 Putting it all together and Algorithm

For every flow $v_i \in \Gamma$, we determine in each node from $\mathcal{P}_i$, an upper bound of the worst-case transversal delay R_i^h using:

$$R_i^h = Smax_i^h + Bklg^h$$

where the maximum backlog $Bklg^h$ is computed according to

$$Bklg^h = \max_{t \geq 0} \left\{ W^h(t) - t \right\} \tag{12}$$

with $W^h(t)$ being defined by Formula (7). The computation stops as soon as the first idle time is encountered. Its occurrence is ensured if $U^h < 1$.

The pseudo-code for computing the worst-case ETE delay of a set of flows in a network is given by Algorithm 1. This algorithm is pseudo-polynomial.

The computation can be done in a discrete manner by testing t only at arrival times of new frames.

Input: A network defined by $\mathcal{S}$ and Γ
For each flow $v_i \in \Gamma$: $Fmax_i$, T_i and $\mathcal{P}_i$
For each node $h \in \mathcal{S}$: r^h with $U^h < 1$ required
The technological latency L
Result: R_i for each $v_i \in \Gamma$
begin
  foreach *flow* $v_i \in \Gamma$ **do**
    $Smin_i^{first_i} \leftarrow 0$
    $Smax_i^{first_i} \leftarrow 0$
  end
  foreach *node* h **do**
    foreach *flow* $v_i \in \Gamma_h$ **do**
      $C_i^h \leftarrow \frac{Fmax_i}{r^h}$
      $J_i^h \leftarrow Smax_i^h - Smin_i^h$
    end
    if *h is a source node* **then**
      $W^h(t) \leftarrow \sum_{v_j \in \Gamma^h} rbf_j^h(t)$
    else
      foreach *input link* IP_x^h *with* $x \in \{1, \ldots, n\}$ **do**
        $W_x^h(t) \leftarrow \min\left\{\sum_{v_i \in \Gamma_x^h} rbf_i^h(t), \frac{r^{h_{x-1}}}{r^h} t + \max_{v_i \in \Gamma_x^h} C_i^h\right\}$
      end
      $W^h(t) \leftarrow \sum_{x=1}^{n} W_x^h(t)$
    end
    $Bklg^h \leftarrow 0$
    $t \leftarrow 0$
    do
      $Bklg^h \leftarrow \max\left\{Bklg^h, W^h(t) - t\right\}$
      $t \leftarrow NextArrivalDate(t)$
    while $W^h(t) > t$
    foreach *flow* $v_i \in \Gamma_h$ **do**
      if $h \neq last_i$ **then**
        $Smin_i^{h+1} \leftarrow Smin_i^h + C_i^h + L$
        $Smax_i^{h+1} \leftarrow Smax_i^h + Bklg^h + L$
      else
        $R_i \leftarrow Smax_i^h + Bklg^h$
      end
    end
  end
end

Algorithm 1: Computing worst-case ETE delays with FA.

	Characteristics		ETE delays	
	$Fmax_i$ (bits)	T_i (μs)	NC (μs)	FA (μs)
v_1	12144	4000	4270.48	3133.60
v_2	12144	4000	3678.54	3270.40
v_3	8256	8000	5404.05	4957.60
v_4 (ES_7)	8256	8000	6709.98	5564.16
v_4 (ES_8)			2955.28	2924.80
v_5 (ES_7)	8256	8000	6709.98	5564.16
v_6 (ES_8)	2048	1000	2955.28	2924.80
v_6 (ES_9)			3009.81	2880.26
v_7	2048	1000	4447.91	3309.44
v_8 (ES_7)	512	1200	4447.91	3309.44
v_8 (ES_9)			973.43	796.42
v_9 (ES_8)	2304	500	693.20	670.08
v_9 (ES_9)			973.43	796.42
v_{10}	512	250	3915.79	2789.12

Table 1: Flow characteristics and computed Worst-case ETE delays (by NC and FA) for the configuration depicted Fig. 1.

5 Experimentation

FA is applied hereafter on two types of configurations. We compare the upper bounds for worst-case ETE delays to the one obtained with Network Calculus (NC) and with the Trajectory Approach (TA). Our comparisons with NC are based on the version [8] used by Airbus for the certification of AFDX, that is: leaky bucket arrival curves, flow serialization optimization (rather than "Pay Burst Only Once" [7]). For TA, we use an implementation based on the patch proposed in [17]. These approaches are first applied on a sample case study, then on a real industrial configuration.

5.1 Case study

We first experiment with the configuration depicted in Fig. 1. The servicing rate of output ports is equal to $r^h = 100\,\text{Mbps}$, except for ES_1, ES_3, ES_4, S_2 (to S_4) S_3 (to ES_6) and S_5 which run at $r^h = 10\,\text{Mbps}$. The characteristics of the flows are given in Table 1.

The worst-case ETE delay bounds obtained with TA are not shown in Table 1, because the AFDX configuration from Fig. 1 cannot be tested with TA. In fact, the global load of traffic ($\sum \frac{C_i^h}{T_i}$) encountered by some flows (*e.g.* v_7) on their path is strictly larger than 1, which prevents the convergence of the computation with this method. The worst-case ETE delay computed with NC and FA are presented in the two last columns of Table 1. The worst-case ETE delays have been differentiated for each destination ES in the case of multicast VLs. For example, considering flow v_4, two worst-case ETE delays are determined with each method: one for destination node ES_7 and another for ES_8.

The ETE delay bounds obtained with FA are tighter than with NC. The

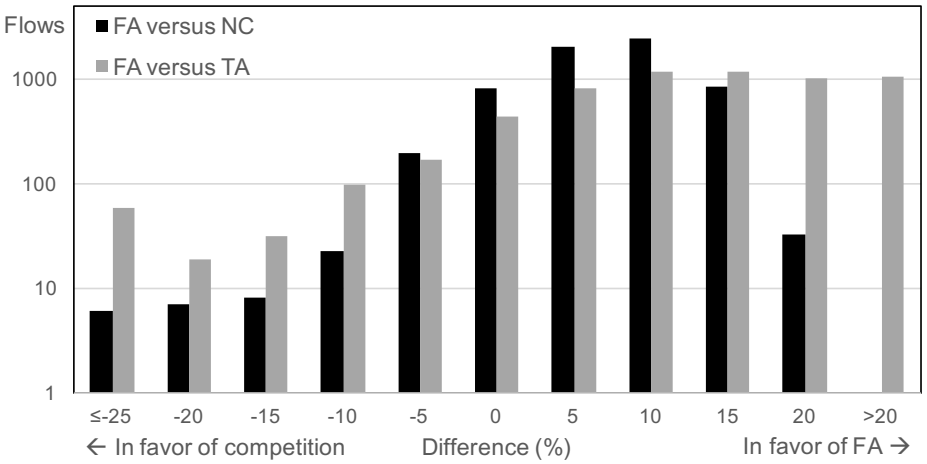

Figure 7: Statistical comparisons between the worst-case ETE delays determined using FA and the others approaches (NC and TA).

difference is up to 28.8% for flow v_{10}.

5.2 Industrial configuration

The three approaches (FA, NC and TA) have also been applied to an industrial configuration (an actual A380 type network configuration), thanks to an internally developed calculation tool. It is made up of 96 End Systems interconnected with 983 multicast VLs (corresponding to 6412 flows) through eight 24-ports switches.

FA is scalable to large configuration since its execution time on this configuration is 2.12s with our Python based computation tool on a 3.3Ghz quad-core 64 bits processor.

The worst-case ETE delay of flows have been computed with FA, NC and TA. We cannot compare the computed delays with exact worst-case ETE delays, since there is no scalable approach to determine exact worst-cases for such an industrial configuration. We compare FA with NC one hand, and with TA on the other hand. Due to the high number of flows, statistical results are showed in Fig. 7 in the form of a frequency histogram.

We determine, for each VL, the difference in percentage between the worst-case ETE delay obtained by FA and by the competing approaches (NC and TA). The frequency histogram is plotted with 5% intervals. For example, the grey bar at 20% indicates that approximately one thousand of flows have a worst-case ETE delay computed with FA 15% to 20% tighter, compared to the TA computation.

It has already been shown that there is no dominating approach between NC and TA [20], in the sense that none of them obtains the tightest bound for all flows of any configuration. Similarly, we see that there is neither a dominating method between FA, TA and NC, since we observe negative and positive values in the histogram depicted in Fig. 7. The performance variations between the methods can be explained each one is based on different conservative assumptions. Each hypothesis induces pessimism in the ETE delay upper bound, but

not necessarily under the same constraints, and thus, for the same flows.

It is still possible to combine two or three of the approaches, as each of them provides a sufficient condition, by taking the minimum of the upper-bounds, in order to mitigate the unfavorable scenarios of a single method.

Nevertheless, only a few and most of the lowest bars on the graph are in favor of TA, whereas the highest bars are in favor of FA. In the end, the upper bounds computed by FA are, on average 4.74% (*resp.* 11.78%) less pessimistic than the upper bounds computed by NC (*resp.* TA).

6 Conclusion

We have proposed a new approach for computing worst-case ETE delays of flows in an AFDX network. More generally, the forward ETE delay analysis (FA) is applicable to any packet network using FIFO queues, compliant with the model defined in Section 3. A formal proof of FA has been provided, covering the case of flow serialization in network links. FA has been compared to existing approach (Network Calculus and Trajectory Approach) on a case study and proved to be scalable on a real-world industrial configuration. The results obtained with FA are relevant, since tighter bounds can be obtained, compared to currently used NC. FA is also applicable without any constraint on the global load of a flow (overcoming the limitation of TA).

A short-term perspective is to extend the FA principle to other servicing policies, in particular, fixed priorities classes for flows. The tightness of the bounds should also be assessed toward the exact worst case ETE delay, which could be obtained thanks to Model Checking (at least on reasonably sized configurations). Another future area for analysis is the choice of the most suitable method based on topology and flow characteristics of a configuration, as non of the three competing approaches dominates another in general. Lastly, applying the FA method to other kinds of architectures, like IEEE Audio Video Bridging (AVB) or even newer IEEE Time Sensitive Networking (TSN) would be of highest interest.

References

[1] *ARINC 664, Aircraft Data Network, Parts 1,2 7. Technical report, ARINC specification 664.*, 2002-2005.

[2] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (tte) design," in *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, no. 18-20. IEEE, May 2005, pp. 22–33.

[3] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard, "A forward end-to-end delays analysis for packet switched networks," in *22nd International Conference on Real-Time Networks and Systems*. Versailles, France: RTNS 2014, October 2014.

[4] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an afdx network," *18th Euromicro Confer-*

ence on Real-Time Systems, ECRTS'06, pp. 193–202, July 2006, dresden, Germany.

[5] D. Thiele, P. Axer, R. Ernst, and J. R. Seyler, "Improving formal timing analysis of switched ethernet by exploiting traffic stream correlations," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, New Delhi, India, October 2014, to appear.

[6] M. Adnan, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows (regular paper)," in *Emerging Technologies and Factory Automation (ETFA), Krakow, 17/09/2012-21/09/2012*. IEEE, septembre 2012, pp. 1–8.

[7] J.-Y. L. Boudec and P. Thiran, *Network calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer Verlag, 2001, vol. 2050, iSBN: 3-540-42184-X.

[8] J. Grieu, "Study and evaluation of the switched ethernet technology for avionics systems interconnection," Ph.D. dissertation, Institut National Polytechnique de Toulouse, 2004.

[9] J. A. R. De Azua and M. Boyer, "Complete modelling of avb in network calculus framework," in *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems*, ser. RTNS '14. ACM, 2014, pp. 55–64.

[10] 802.1BA, "Ieee standard for local and metropolitan area networks–audio video bridging (avb) systems," IEEE, Tech. Rep., 2011.

[11] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an afdx avionic network," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 1, pp. 38–49, Février 2009.

[12] K. W. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming*, vol. 40, no. 2-3, pp. 117–134, Apr. 1994.

[13] S. Martin and P. Minet, "Holistic and trajectory approaches for distributed non-preemptive fp/dp* scheduling," in *ICN*. Berlin Heidelberg: Springer-verlag, 2005, pp. 296–305.

[14] J. J. Gutiérrez, J. C. Palencia, and M. González Harbour, "Response time analysis in afdx networks with sub-virtual links and prioritized switches," *XV Jornadas de Tiempo Real*, January-February 2012, santander.

[15] S. Martin, P. Minet, and L. George, "End-to-end response time with fixed priority scheduling : trajectory approach versus holistic approach," *International Journal of Communication Systems*, vol. 18, no. 1, pp. 37–56, February 2005, chichester, UK.

[16] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard, "Optimistic problems in the trajectory approach in fifo context," in *18th IEEE Int. Conf. on Emerging Technologies and Factory Automation*. Cagliari, Italy: ETFA 13, September 2013.

[17] X. Li, O. Cros, and L. George, "The trajectory approach for afdx fifo networks revisited and corrected," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*. IEEE, 2014, pp. 1–10.

[18] G. Kemayo, N. Benammar, F. Ridouard, H. Bauer, and P. Richard, "Improving afdx end-to-end delays analysis," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015, pp. 1–8.

[19] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying and optimizing Trajectory approach for performance evaluation of AFDX avionics network (regular paper)," in *Emerging Technologies and Factory Automation (ETFA), Palma de Mallorca, 22/09/2009-25/09/2009*. IEEE, septembre 2009, pp. 1–8.

[20] ——, "Improving the worst-case delay analysis of an afdx network using and optimized trajectory approach," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 521–533, November 2010.

[21] J. Lehoczky, "Fixed-priority scheduling of periodic task sets with arbitrary deadlines," *proc. Real-Time Systems Symposium*, no. 5-7, pp. 201–209, December 1990.