Temporal Analysis of AFDX Networks

João Louro
jd.louro@outlook.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

November 2018

Abstract

The recent advances in industrial systems’ communications have led, in the avionics domain, to the introduction of Avionics Full-Duplex Switched Ethernet (AFDX). It is today the information transmission system of the most modern civilian aircraft. For certification purposes, upper bounds to end-to-end delays of messages transmitted over AFDX networks must be guaranteed. The calculation of these bounds presents considerable challenges, and the development of methods for the computation of tight end-to-end delay upper bounds in switched networks is an open research field. In this thesis, the existing methods for AFDX worst-case delay calculation will be compared. The most capable ones will be selected and posteriorly implemented. The developed computation tool will later be added to GMV’s proprietary AFDX analysis software.

Keywords: Avionics Full-Duplex Switched Ethernet (AFDX), ARINC-664 Part 7, Forward Analysis, Network Calculus, end-to-end (ETE) delay worst-case transmission time, latency.

1. Introduction

The Avionics Full-Duplex Switched Ethernet (AFDX) protocol defines the latest generation of aviation data buses. The AFDX standard presents substantial improvements on its predecessors, allowing simultaneous bi-directional communication by default.

AFDX is patented by the international aircraft manufacturer Airbus, being in essence a specific implementation of the ARINC 664 Part 7 standard. Given the critical nature of the aeronautical domain, in which all systems aboard the aircraft require updated, complete data, the transmission of data must be performed according to a determined temporal boundary and the avionic network must be deterministic. This is all accomplished by the AFDX standard. Also due to the critical nature of the aeronautical sector, all systems aboard the aircraft must obligatorily pass several certification processes, both as individual systems and as part of the final assembled product. During this certification process, a temporal analysis of the network must be performed. In other words, an upper bound on the latency (as shown in figure 1) must be computed for every data flow in the network.

Figure 1: The end-to-end delay characteristics [6].

The calculated bounds on worst-case latencies can then lead to considerable cost savings when developing an avionic network. Algorithms for the calculation of worst-case end-to-end latencies in AFDX are relatively complex, and their development and improvement is an active research field.

The work described in this document was developed in partnership with GMV Innovating Solutions. GMV is a privately owned technological business group with an international presence. Founded in 1984, the group is based in Spain and employs more than 1600 people worldwide.

2. AFDX Networks

The components of AFDX networks include the end systems, which correspond to the ingress/egress points of AFDX networks; the switches, which forward the packets through the network according to the information on a static connection table and the virtual links, which define static routes across the shared physical medium of these networks. A sample AFDX configuration is depicted in figure 2.

Figure 2: Example AFDX configuration.

In AFDX, the concept of virtual link corresponds to a unidirectional linking between end systems. One vir-
tual link has a single origin end system, but it might have several destination end systems. The routes defined by virtual links are static, and have a predetermined bandwidth allocation handled through the maximum frame size ($L_{\text{max}}$) and the bandwidth allocation gap ($BAG$) which corresponds to the minimum allowed time between the generation of two consecutive data frames in the origin end system.

Being a safety-critical system, AFDX presents full determinism, which is enforced by traffic shaping and policing units in each element of the network. Since the physical links are full-duplex, collision between data packets is impossible.

Switches in AFDX apply traffic policing and routing according to the specification of the virtual links ($L_{\text{max}}$ and $BAG$), operating as store-and-forward between input and output ports. A switching latency of 16 $\mu$s must be considered in the operation of the switches. The output ports of switches are work-conserving queues, with a FIFO policy of service. End systems have a single output port (also with a FIFO buffer), and no input ports. The servicing rate of the output ports in AFDX is constant, being either 100 or 10 Mb/s.

3. Comparison of Latency Calculation Methods

The calculation of end-to-end latencies in AFDX networks presents a significant challenge, due to the considerable complexity of this protocol, and the fact that message packets can interfere with one another in the switches, forcing each other to undergo waiting periods before progressing. In this section, the most significant research on this subject will be summarized. Posteriorly, the method (or methods) with most potential will be chosen for implementation.

A number of methods have been used for the calculation of end-to-end latencies in AFDX. However, some of these methods have been proven not to be scalable to realistically sized networks, which makes them a bad choice for implementation. These methods are Model Checking [13], Holistic Approach [14] and Mathematical Programming [13]. Performing a simulation of the network is also not a good solution for obtaining worst-case latencies [7].

Network Calculus (NC) is the current reference for certification of AFDX networks in civilian aircraft, having been used to certify the A380 AFDX internal communications backbone [5]. In NC, the latency upper bound is calculated as a sum of local worst-case scenarios, computed in the switches and end-systems of the network. This makes for a relatively conservative upper bound [6], since it is frequent that the computed local worst-case scenarios can not be verified simultaneously. NC has been improved to take into account the serialization effect [16]. With this improvement in the implementation, data frames transmitted through the same physical link are counted as being delivered sequentially. This excludes the hypothesis of them arriving simultaneously to the link’s destination node, which corresponds to the worst delay, leading to less pessimistic time bounds. Despite not being the most recent method, Network Calculus is still the industry standard for the evaluation of end-to-end time delays in AFDX networks. For these reasons, Network Calculus is a very good choice for implementation.

A more recent method, the Trajectory Approach (TA), was conceived in order to try and reduce the pessimism relatively to Network Calculus. In Bauer et al. [2], TA was improved by taking into account the serialization effect. However, this method presents significant limitations: it can not bound the latency for paths with high information loads (cumulative load over 100%), and it lacks a mathematical, formal proof of robustness as a method for the computation of reliable upper bounds. In fact, it has been demonstrated in Kemayo et al. [10] that TA computes optimistic bounds in some cases. This is not acceptable, and effectively disqualifies this method from being used for certifying AFDX networks. This limitation has been patched in Li et al. [12], but this patch is also lacking a formal proof of correctness. TA is for these reasons a bad choice for implementation.

The latest method to have been proposed is Forward Analysis (FA) [4]. It was first proposed for AFDX in 2014 [11], and was posteriorly improved by taking advantage of the serialization effect [9]. It has been generalized for other networks in Benammar et al. [4] and support for fixed-priority flows was implemented in Benammar et al. [3]. This method shows potential, being able to scale to realistically-sized and heavily loaded networks, while often providing tighter bounds than NC. This method has also been formally verified, which improves confidence in the results. The drawbacks presented by this method have to do with its recent nature, and the absence of following research on it.

If only one method had to be chosen, the most sensible choice would be to implement Network Calculus, due to the fact that it is the current reference and the benchmark to which all new methods are compared, and the reduction in pessimism presented by Forward Analysis is not significant enough to select this method over Network Calculus [4]. Forward Analysis, despite its potential, is still a relatively recent tool, and has not undergone significant scrutiny. This would make it the second choice for implementation. All the other methods present significant drawbacks which make them unworthy of the implementation work, when compared to these two methods.

The results of two different methods can, however, be combined by running both implementations separately and choosing the smaller worst case end-to-end delay for each of the flows in the network. By implementing such a combination of methods, the best, most reliable possible tool for analysis of worst case latencies can be developed, according to the most recent available research. The choice of implementing two methods instead of one is ambitious, and the amount of additional implementation work is considerable, but the advantages of combining a time-proven, industry tested method with a state-of-the-art one make it a worthwhile choice. Work of implementation of such a recent method as Forward Analysis also presents scientific benefits. The implementation of Forward Analysis
4. Implemented Methods

The two methods selected for implementation (NC and FA) will now be described in detail, and their mathematical basis will be demonstrated.

A particularity of NC and FA is that, in order for these methods to be implemented, the AFDX network must be modeled slightly differently than the way it is presented. In this alternative modeling, the network consists of a set of nodes $S$ interconnected through logical links. Each node $h$ will have a set of $n$ input links denoted $IP_h^x$ (with $1 \leq x \leq n$), and single First-In-First-Out buffer with a servicing rate $r_h$. Each node will be connected to one or multiple destination nodes through output links. In the context of an AFDX network, each multiplexing point will correspond to a node, since nodes have a single output buffer. Nodes will then correspond to the output port of switches and end systems. In this model, only end systems which serve as the origin of virtual links are considered. End systems which function only as destination end system but to the output port of the last switch crossed by the flow $v_i$ in node $first_i$. A frame of the maximum size $L_{max}$ will be transmitted through node $h$ in time $C_h^v = L_{max}/r_h$.

5. Network Calculus

NC makes no assumptions concerning the arrival time of packets. Its main concept is to over-estimate the quantity of data that flows deliver to nodes and to under-estimate the capacity of the network elements to serve this data. This is achieved through the arrival and service curves. This method has its mathematical basis on the min-plus ($\land,+$) dioid, whose convolution $\odot$ and deconvolution $\oslash$ can be defined as

$$
(f \odot g)(t) = \inf_{0 \leq u \leq t} f(t - u) + g(u) \\
(f \oslash g)(t) = \sup_{0 \leq u \leq t} f(t + u) - g(u)
$$

In NC, flows are modeled by their cumulative function $R(t)$, corresponding to the total number of bits sent through the flow up to time $t$. $\alpha$ is an arrival function of flow $R$ if $R \leq R \odot \alpha$. A node is said to have a service curve $\beta$ if, for all the flows passing through the node $R^* \geq R \odot \beta$. $R$ and $R^*$ correspond respectively, to the input and output cumulative functions of the analyzed node. $\alpha^* = \alpha \odot \beta$ is therefore an arrival curve for $R^*$.

The worst-case delay suffered by a flow $R$ bounded by an arrival curve $\alpha$ in a node whose service curve is $\beta$ corresponds do the maximum horizontal difference between the functions $\alpha$ and $\beta$, formally defined as

$$
d(\alpha, \beta) = \sup_{s \geq 0} (\inf \{ \tau \geq 0 | \alpha(s) \leq \beta(s + \tau) \})
$$

Each flow is modeled by a leaky bucket function $\gamma_{r,b}$, with $b = L_{max}$, and $r = \frac{max}{\text{MAC}}$. Each node offers a service curve $\beta_{r,b,L}$, with $L = 16\mu s$ (technological latency) and $r^h$ being the servicing rate of the node. The virtual links which compete for a given output port are merged into a single flow by summing their respective arrival curves. The delay is then calculated as shown in figure 4.
curves have been tried [16] [6], leading to variable pessimism. In this work, left shifts equal to \( d(\alpha, \beta) \) and \( d(\alpha, \beta) - C_i^h \) were implemented.

The implementation of the serialization effect in NC proceeds in two steps. First, the overall arrival curve is computed for each link. It is the minimum between the sum of the arrival curves of all the flows sharing the considered link, and 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 step is illustrated in figure 5 for a link shared by two flows with arrival curves. In the second step, the curves obtained for the different links are added, obtaining the overall arrival curve of a node.

The complete pseudo-code for the NC method can be found in algorithm 1.

6. Forward Analysis

In the Forward Analysis method, it is considered that frame of flow \( v_i \) will incur in a delay in a node \( h \) made up of a technological latency \( L = 16\mu s \) and a waiting time (highly variable, corresponds to the amount of time spent in each of the nodes in the path \( P_i \)). The Forward Analysis approach was developed with the goal of computing an upper bound to these waiting times, in order to bound the end-to-end latency of flows.

When calculating \( R_i^h \), which corresponds to the worst-case delay of a frame of flow \( v_i \) from \( \text{first}_i \) until a node \( h \) in the path \( P_i \), we can separate the calculation between a worst-case transversal time until \( h \) and the delay incurred in the node \( h \), which amounts to having \( R_i^h = S_{max}^h + Bklg_i^h \).

\( Bklg_i^h \) corresponds to an upper bound on the worst-case backlog in the output buffer of the node \( h \) at any time, \( S_{max}^h \) denotes the maximum possible delay suffered by a frame of flow \( v_i \) from the time it is generated in \( \text{first}_i \) until before entering node \( h \). \( S_{max}^h \) is determined iteratively (a shown in the bottom part of algorithm 2, with \( S_{max}^{\text{first}_i} = 0 \) by definition. The shortest possible time for a frame of flow \( v_i \) to reach node \( h \) (\( S_{min}^{\text{last}_i} \)) is calculated in a similar way. The worst-case end-to-end latency for a flow \( v_i \) corresponds to its value of \( R_i \).

FA is not as intuitive as NC, and its understanding implies some level of familiarity with AFDX networks, as it makes assumptions regarding the arrival times of separate packets. The method was proposed as a set of theorems by Kemayo et al. [4]. The formal proof of these theorems can be found in the original publication, but was omitted for simplicity’s sake.

Theorem 1. In a FIFO buffer, the worst-case backlog generated by a flow \( v_i \) in a node \( h \) from path \( P_i \) during a time interval \([0, t]\) is obtained when:

1. the frames of \( v_i \) are generated periodically every \( BAG_i \) in the source node \( \text{first}_i \);
2. the first frame, denoted \( f_i \), generated by \( v_i \) arrives at time \( 0 \);
3. \( f_i \) reaches node \( h \) after suffering its maximum traversal delay (\( S_{max}^h \)) whereas all the subse-
quent frames from \(v_i\) arrive on \(h\) suffering their minimum traversal delay so that their arrival time is never before time 0.

**Theorem 2.** In a FIFO buffer, considering the worst-case scenario from Theorem 1, the first frame \(f_i\) generated by flow \(v_i \in \Gamma^h\) reaches node \(h\) simultaneously with \(k^h\) others frames. \(k^h\) can be defined as:

\[
k^h = \left\lfloor \frac{J^h}{BAG_i} \right\rfloor \tag{3}
\]

which leads to

\[
k^h = \left\lfloor \frac{J^h}{BAG_i} \right\rfloor \tag{4}
\]

In the previous theorem, the worst-case jitter of flow \(v_i\) in node \(h\) is represented by \(J^h = Smax^h - Smin^h\). Taking into account theorems 1 and 2, the arrival date of \(f_{i,1}\), the first frame arriving at \(h\) after \(t = 0\) is \((k^h + 1)T^i - J^h\).

The request bound function \(rbf^h(t)\) defines the total transmission time of frames generated by flow \(v_i\) arriving on node \(h\) during an interval of length \(t\). \(W^h(t)\) denotes an upper bound on the workload, which can be defined as the cumulative transmission time of all the flows crossing \(h\) during a time interval with duration \(t\).

\[
W^h(t) = \sum_{v_i \in \Gamma_i} rbf^h(t) \tag{5}
\]

**Theorem 3.** Taking into account a flow \(v_i\) crossing a node \(h\) with a FIFO buffer, the RBF in an interval with duration \(t\) can be defined as:

\[
rbf^h(t) = \left(1 + \left\lfloor \frac{t + J^h}{BAG_i} \right\rfloor \right) C^h \tag{6}
\]

As is the case with other methods for the bounding of worst-case latencies in switched networks, FA can have its pessimism reduced if the serialization effect is taken into account. In the scheme to figure 6, the first \(k^h + 1\) frames from \(v_i \in \Gamma_h\) can’t arrive in node \(h\) at the same time, due to the fact that they share the same input link. The model previously presented for the computation of the worst-case workload (with \(rbf\)) can then be improved, through the distinction of flows based on their input links. In order to obtain the global workload in the node, the contribution of each of the input links is summed:

\[
W^h(t) = \sum_{x=1}^{n} W^h_x(t) \tag{7}
\]

In equation 7, \(W^h_x(t)\) denotes the cumulative transmission time of frames from flows coming through the input link \(IP^h_x\) during a time interval of duration \(t\).

**Lemma 4.** The worst-case cumulative transmission time, originated in an input link \(IP^h_x\) in a node \(h\) during a time interval of duration \(t\) can be defined as:

\[
W^h_x \leq \frac{r^{h_{x-1}}}{r^h} t + \max_{v_i \in \Gamma^h_x} C^h_i \tag{8}
\]

in which \(h_{x-1}\) refers to the predecessor node of \(h\) via \(IP^h_x\).

![Figure 6: 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 \([0, t]\).](image)

**Theorem 5.** The cumulative workload which arrives at a node \(h\) through the \(IP^h\) input link during a time interval of duration \(t\) is bounded by the expression

\[
W^h = \min \left\{ \sum_{v_i \in \Gamma^h} \frac{rbf^h_x}{r^h} + \max_{v_i \in \Gamma^h_x} C^h \right\} \tag{9}
\]

The maximum cumulative workload \(W^h(t)\) incoming in a node \(h\) during a time interval \([0, t]\) is determined by formula 7 and theorem 5.

The nodes start serving frames as soon as they arrive in the buffers. The maximum backlog (maximum waiting time in the node) in a node \(h\) is then obtained by computing the difference between the incoming workload \(W^h(t)\) and the amount of traffic served at the rate of the node during any time interval \([0, t]\):

\[
Bklg^h = \max_{t \geq 0} (W^h(t) - t) \tag{10}
\]

Only the times of arrival of new nodes will be tested, plus the times of equality between the two portions of the expression in theorem 5. This computation is guaranteed to stop at the occurrence of an idle time, which occurs at a time \(t\) on a node \(h\), if it is verified that \(W^h(t) \leq t\).

**Theorem 6.** In a node \(h\) with FIFO servicing, and in which the arriving frames follow the scenario described in theorem 1, the existence of an idle time is guaranteed if the condition on local load \((U^h)\) is verified:

\[
U^h = \sum_{v_i \in \Gamma^h} C^h_i < 1 \tag{11}
\]

For each of the flows \(v_i \in \Gamma\), in each of the nodes in the path \(P_i\), an upper bound of the worst-case end-to-end delay is calculated as \(R^h_i = Smax_i^h + Bklg^h\). In the previous expression, the maximum backlog is computed according to \(Bklg^h = \max_{t \geq 0} \{W^h(t) - t\}\) where \(W^h(t)\) is defined by formula 7.

The computation will be performed iteratively, stopping as soon as the first idle time is encountered, which is guaranteed to happen if \(U^h < 1\).

The complete pseudo-code for the Forward Analysis method can be found in algorithm 2. The function nextArrivalDate\((t)\) refers to the arrival pattern of frames described in the proof of theorem 3.
Input: A network defined by $S$ and $\Gamma$
For each flow $v_i \in \Gamma$: $F_{\text{max}}$, $BAG_i$, and $P_i$
For each node $h \in S$: $r^h$ with $U^h < 1$
The technological latency $L$

Result: $R_i$ for each $v_i \in \Gamma$

begin
for each flow $v_i \in \Gamma$
do
  $S_{\text{min}}^{\text{first}}_i ← 0$
  $S_{\text{max}}^{\text{first}}_i ← 0$
end

for each node $h$
do
for each flow $v_i \in \Gamma$
do
  $C_h^i ← F_{\text{max}}$
  $J_i^h ← S_{\text{max}}^h - S_{\text{min}}^h$
end
if $h$ is a source node then
  $W_h^b(t) ← \sum_{v_i \in \Gamma_h} rbf_i^h(t)$
else
  foreach input link $IP_x^h$ with $x \in \{1, \ldots, n\}$
do
    $W_h^b(t) ← \min\{ \sum_{v_i \in \Gamma_x} rbf_i^h(t), \}
               \frac{S_{\text{max}} - S_{\text{min}}}{\sum_{x=1}^n C_h^i} \}$
end
$W_h^g(t) ← \sum_{x=1}^n W_x^h(t)$
end

$B{\text{klg}}^b ← 0$
$t ← 0$
do
  $B{\text{klg}}^b ← \max\{ B{\text{klg}}^b, W_h^b(t) - t \}$
  $t ← \text{NextArrivalDate}(t)$
while $W_h^b(t) > t$;

for each flow $v_i \in \Gamma_h$
do
if $h \neq \text{last}_h$ then
  $S_{\text{min}}^{h+1} ← S_{\text{min}}^h + C_h^i + L$
  $S_{\text{max}}^{h+1} ← S_{\text{max}}^h + B{\text{klg}}^i + L$
else
  $R_i ← S_{\text{max}}^h + B{\text{klg}}^h$
end
end

end

Algorithm 2: Computing worst-case end-to-end delays with FA [4].

7. Implementation of the Methods

With the major goals of the work defined (implementation of NC and FA), the programming portion of this dissertation must then be undertook. The development of the code accounted for the majority of the work described in this dissertation, and presented considerable challenges.

The input of a program for the calculation of end-to-end delays in AFDX is relatively complex: it must include a complete description of the network, with all its elements and interconnections. Fortunately, GMV already possessed some capabilities which simplified the choice of input format. The A664 Network Configurator, used for the creation and manipulation of virtual AFDX networks, can export a file which describes in detail both the physical components and the virtual links of an AFDX network previously designed by the user of the tool. The files exported by the tool are documents in the XML (Extensible Markup Language) format. In order for the worst-case latencies to be calculated, the information present in the XML files must then be parsed, inter-related and organized for easy access. This would mean creating virtual objects, such as, e.g., a virtual link object, which aggregates all the information present in the XML file regarding a particular virtual link. GMV’s A664 Network Configurator already includes capabilities for calculation of end-to-end latencies, adapted from a tool based on the Trajectory Approach method developed by Pavel Vdovin [15]. However, this tool provides erroneous results, and the TA method presents several limitations. Nonetheless, the parsing of the XML files was already handled by the tool, as well as the aggregation of the gathered information into interconnecting classes of objects representing the whole network. The work could be mostly focused on the implementation of the new algorithms. It must be noted, however, that the model of the network created by Vdovin’s implementation was somewhat more complex than necessary, and some of his programming choices led to a relatively high code complexity and computational cost in the developed program. Since the model of an AFDX network described by Vdovin [15], was implemented in C++, both the NC and FA methods were implemented in this programming language.

7.1. pum.cc library

One of the most complex elements on an implementation of NC are the arrival and service curves when considering the serialization effect, as well as the functions that must be computed from them. In the case of this work, the curves used in NC were implemented by Eugene Chemeritskiy, and their source code was published on Github under a permissive license. The finding of this library was a very welcome contribution to this work. If the affine curves had to be implemented, there is a strong probability the code would not be finished in the predicted time-frame. This is simply an example of the benefits of the open-source community for both scientific work and enterprise.

7.2. “João&Maria” Method

The work described in this thesis was developed in communications with the Brazilian manufacturer Embraer, the third largest producer of civil aircraft. Embraer is one of the parties interested in the network analysis suite in development at GMV (of which this work is an important part), and constant feedback was provided by Sérgio Duarte Penna, an engineer at Embraer with expertise in AFDX networks and latency analysis methods. Besides developing implementations of recognized methods such as NC and FA, there were attempts at developing a new, simpler method for the calculation of bounds in worst case end-to-end latencies. The new method, originally conceived by Sérgio Penna, was named "João&Maria". A partial implementation of the method in C Language was even developed by
Penna. However, it was found that "João&Maria" could not be generalized as an algorithm (the method could not deal with some corner cases).

7.3. Result Comparison Tool
Despite the simplicity of most of the networks to be tested, it would be significantly time consuming to repeatedly compare the obtained results, presented on a command line, with the latency bounds obtained from research articles. In order save time, a tool for the direct and quick comparison of results was developed during the work presented in this thesis. The tool has several capabilities which were useful during this work: comparison of results obtained for one of the methods (NC or FA) with the data published in research papers, and comparison between the results obtained for the two methods when analyzing a specific network. A third function of this tool allowed for comparisons between the results of a method with and without the serialization effect. This tool was essential for the analysis of the results, especially when considering realistically-sized networks, which are interconnected by thousands of flows.

7.4. Assembly of Realistic Networks
It is not enough to test the implementations on the simple networks described in research articles. A more realistic test must be prepared in order to evaluate the developed program. For that purpose, a large network was assembled, based on descriptive schematics of the AFDX network installed in the Airbus A380. Based on this A380-inspired network, some simpler networks were built, by reducing the number of virtual links, while maintaining the network components.

8. Corrections to the Literature
Throughout this work, some inaccuracies were found on relevant research to this field. Since some of them have significant impact on the results of important articles, a separate section was created in order to present a more detailed description of these findings.

8.1. Network Servicing Rates in Benammar et al. [4]
The implementation of the FA algorithm was marked by considerable difficulties. Despite matching the data present in Benammar et al. [11] and [9], the initial results did not match the ones present in one of the reference papers, presenting a considerable error. The authors of the paper were then contacted. The implementation of FA was from that moment on conducted while in contact with Frédéric Ridouard, an Associate Professor at the LIAS Laboratory of the Université de Poitiers, France responsible for the development of the theoretical basis for Forward Analysis. A critical error in the description of the network analyzed in the research article by Benammar et al.[4] was immediately identified: the service nodes for some of the output ports of switches were erroneously described. If this configuration of the network were to be analyzed by Forward Analysis, the upper bounds for the worst-case latencies would never match the results of the paper. Professor Ridouard admitted that the network was wrongly described in the paper, and provided the correct description of the network.

8.2. Ordering of the Nodes Described in Kemayo et al. [9]
In [9], a method which describes the correct order of propagation of the computations in FA is described. It introduces the concept of a virtual link’s rank and the nodes’ order. The rank $h_i$ of a flow $e_i$ in node $h$ corresponds to the number of nodes, including $h$, crossed by $e_i$ to reach $h$. The order $O(h)$ of a node $h$ is the maximum rank of $h$ in the paths of the flows crossing it. The calculation is propagated by analyzing the nodes in a sequence of growing order, starting by the origin end systems, which will have order 1. It was found that, in some corner cases, the method for the propagation of the calculations is not robust. Such a case is presented in figure 2. In the model of figure 3, the number on the lower right corner of each of the nodes corresponds to its order. As can be seen, both $S_5$ and $S_{61}$ (a node corresponding to one of the output ports of $S_0$) have order equal to 4. The calculations related to them will then be performed in an indiscriminate sequence. However, the calculations in $S_{61}$ are dependent from the ones of $S_5$. If it happens that the calculation reaches $S_{61}$ before $S_5$ (which would be possible if the propagation algorithm described in [9] is used), the program will try to access non-existent information, and the computation will stop. This limitation of the propagation algorithm is easy to miss, since it is only present in networks with specific configurations, which can be considered corner cases. The approach was then modified, and the implemented propagation algorithm was simpler but less efficient: the computation only propagates to a node if all nodes before it have been analyzed or if it is an origin end system. This method is robust, but was verified to increase the computation time of the implementations an average of around 13%.

8.3. Lacking Testing Times in Benammar et al. [4]
In Benammar et al. [4], it is explained that, in order to find the maximum values for $W(x)$ necessary for the calculation of the bounds on worst-case end-to-end delays, only the arrival time of new frames need to be tested. However, when applied to the network described in the article, the developed program’s results did not match the ones present in the research article, even with the corrections to the network described previously. It was then explained by the author that, besides testing the time $t$ for every new arrival frame, as described in Benammar et al. [4], the times of equality between the two constituent functions of $W(t)$ must also be tested. Professor Ridouard provided an algorithm which calculates all the times in which $t$ must be tested. This algorithm was programmed in a slightly modified manner, since the pseudo-code received was not implementable in its original form.

8.4. Bug in the Author’s Implementation
Even after implementing the two corrections previously described, a significant amount of error (in some flows close to a 20%) was still present in the results of the implementation, when compared to the results in Benam-
mar et al. [4]. Such an error would not be acceptable in an implementation of this nature, and considerable effort was dedicated to trying to solve it. An extensive set of hand-made calculation was sent to Professor Ridonard, which matched the results of the program developed during this thesis. Upon comparison between the results of the author’s implementation and the one described in this work, problems were found with the implementation of Forward Analysis developed at the LIAS Laboratory. Upon correction of the detected bugs, the authors provided an as of yet unpublished updated version of Benammar et al. [4], taking into account the inaccuracies described in this section, whose results matched perfectly those obtained by the implementation developed in this work.

This section describes how inaccuracies detected during this work had significant effect on the results of the most recent research regarding Forward Analysis. If this algorithm ends up gathering mainstream recognition in the certification of AFDX networks, the work developed during this thesis can be said to have made a small contribution.

9. Obtained Results
It is common practice in the literature related to methods for the calculation of worst-case end-to-end delays to present results obtained using different computation methods on small sample networks. A source of data to verify the outputs of the developed program can then be found across the published articles on this subject. However, it was soon noted that very little data regarding worst-case latencies in AFDX networks is available in the literature. Due to this fact, some additional tests were devised, based on the realistically-sized network which was assembled.

9.1. Results of the Network Calculus Implementation
The only data found on latency bounds in AFDX calculated using NC was found in papers related to Trajectory Approach and Forward Analysis. In these papers, the results for NC were of secondary importance, serving only as a benchmark for comparison of other methods.

A comparison between the bounds calculated by the developed implementation of NC and the data in Scharbarg et al. [6] was made, and a negligible difference between the values was verified. This difference amounted to an average of 0.13% of the latency values presented in the article. This is a good first indication of the quality of the implementation.

As mentioned previously, there are several methods for the calculation of output curves in NC. Since the data analyzed stems form an implementation of NC developed by Jrome Grieu for his 2004 thesis [8] (for which the method for computation of output curves could not be confirmed), an exact match between the data and the output of the program may not be a realistic goal. A relatively small difference between the results is then expected. Since two approaches for the calculations of arrival curve were tried during the work described in this dissertation, the one which provides the most approximate result will be presented in this document. The results using the two methods for the calculation of the output curve have consistently been similar, presenting relatively small differences.

Regarding the network analyzed in Kemayo et al. [11], the output of the program matched the article data very closely, with an average difference corresponding to 1.9% of the latency values presented in the article. As in the previous case, some variance in the results is present, although not too significant. This is also the case for the network analyzed in Kemayo et al. [9]. The average difference between the output latency of the developed implementation and the data in this article was 2.62%.

Regarding the network presented in Benammar et al. [4], the difference between the article data and the outputs of the developed program is more considerable, amounting to 9.26%. This error is more significant than expected. The authors were contacted, since there is a suspicion that the corrections performed to the description of the network (described in section 8.1) may have led to differences in the network considered for FA and NC in Benammar et al. [4], but no information was provided. It must also be noted that the difference between the results for this network obtained using Network Calculus and Forward Analysis is very small (0.93%), with some additional pessimism for Network Calculus. This increases the suspicion that there might be some inconsistency present in the calculation for NC in this network.

Since the implemented methods are relatively complex, the possibility of bugs in this implementation is of course not excluded. A great deal of effort went into preventing them during this work. For the shortest paths in networks, some bounds for end-to-end latencies can be calculated by hand and compared with the algorithm results. This was done in several occasions, and the results of the hand-made calculations matched the outputs of the implemented programs for Network Calculus. Since the authors of the analyzed articles are not trying to match any already existing results, it is expected that the obtained latencies do not undergo as much scrutiny as they would if the goal was to match some preexisting data, as was the case during the work described in this dissertation. The fact that the implementation of Network Calculus was not the main focus of Grieu’s Ph.D thesis, but merely a part of the developed research must also be held into consideration.

These comparisons of results were presented to Sérgio Penna, and were considered considered very satisfactory. Sérgio claimed that differences of this nature were to be expected. In fact, in his experience analyzing similar articles on AFDX latency bounding methods, occasional calculation errors are normally found.

9.2. Results of the Forward Analysis Implementation
Since the FA implementation was developed in close communication with the developers of the method, the results obtained presented a perfect match of all the data published in the literature so far [11] [9] [4]. For the network presented in Kemayo et al. [11], the data does not consider the serialization effect. However, this effect can be easily disabled for both the NC and FA
implementations, allowing for the testing of the results presented in all the available articles. The data which was matched by the outputs of the developed program is not correctly presented in Benammar et al. [4]. In fact, the correct latency results for network analyzed in this research article were extracted from the updated version of the document previously mentioned, which is complete with the corrections described in this document.

These are very positive results, only made possible by the direct communication with the developers of the Forward Analysis method. They present a good indication of the robustness of the developed implementation of FA.

9.3. Analysis of Realistically-Sized Networks
Given the relatively small quantity of data for testing present in the literature, some other tests must be conducted. Results will be presented in graphical form for the realistically-sized network, composed of 633 virtual links, distributed by 1447 flows, running across 9 switches and 80 end systems, and for a "semi-realistic" network, based on the same configuration, but linked by 69 virtual links, corresponding to 159 flows.

The results obtained for this "semi-realistic" network can be seen in the graph of figure 7. In the graph, information regarding both NC and FA (both considering and ignoring the serialization effect) is presented. The results were as expected, with FA presenting less pessimistic bounds than NC, and a significant reduction in pessimism due to the serialization effect. This reduction is greater than the ones verified in smaller sample networks, due to the specificities of the devised network. For this configuration, the bounds for FA received a reduction of pessimism of 54.2% due to serialization effect. In the case of NC, this reduction amounted to 54.1%. Without considering the serialization effect, FA presents 2.8% less pessimism than NC. If the serialization effect is considered, this reduction corresponds to 25%.

The same results obtained for this "A380-inspired" (realistically sized) network can be seen in the graph of figure 8. In this network, the bounds for NC received a reduction of pessimism of 57.8% due to serialization effect. In the case of FA, this reduction amounted to 79.4%. Without considering the serialization effect, FA presents 11.8% less pessimism than NC. If the serialization effect is considered, this reduction corresponds to 60.3%. At first glance, the results correspond to what is expected, although the differences between NC and FA with serialization are very pronounced. This is probably simply due to the specificities of the developed network. The main concern when assembling this network was to conform with the local load limit mentioned in theorem 6, which might have led to an unrealistic or strange configuration.

The results were very positive, and similar to what has been presented in previous research. The obtained latencies were considered very satisfactory by both GMV and Sérgio Penna, and the data presented in the literature was matched for all the expectable cases.

9.4. Computational Cost Assessment
The data found in the literature regarding the computational cost of these algorithms when applied to AFDX is relatively sparse. In the case of FA, Benammar et al. [4] claimed that the method the execution time on a Python-based implementation was 2.12 seconds on a 3.3 GHz quad-core 64 bits processor for a realistically-sized network. These low computational costs were not verified in the implementation developed during this work: in fact, run times around the two second mark were verified only for smaller networks. When attempting to calculate latency bounds in more realistically-sized networks, the computation takes much longer. Computation time for both implemented algorithms across several networks can be found in table 9.4. In the table, when an article is mentioned in the "Network" field, it refers to the network analyzed in the document. These values are relatively high. Some effort was taken towards reducing computational cost in this work. The network model used in this work (implemented by Vdovin) made some programming choices which increased complexity and runtime. Nonetheless, this initial work was essential for the success of this implementation.
Table 1: Running times of the implemented algorithms for different networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>NC (s)</th>
<th>FA (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bauer et al. [1]</td>
<td>0.02</td>
<td>0.51</td>
</tr>
<tr>
<td>Kemayo et al. [11]</td>
<td>0.02</td>
<td>0.50</td>
</tr>
<tr>
<td>Kemayo et al. [9]</td>
<td>0.02</td>
<td>0.761</td>
</tr>
<tr>
<td>Benammar et al. [4]</td>
<td>0.05</td>
<td>1.12</td>
</tr>
<tr>
<td>Semi-realistic</td>
<td>31.30</td>
<td>50.32</td>
</tr>
<tr>
<td>A380-inspired</td>
<td>662.69</td>
<td>1391.41</td>
</tr>
</tbody>
</table>

10. Conclusions

In this work, capabilities for the calculation of worst-case end-to-end latencies in AFDX networks were developed, which will be integrated in GMV’s tool for AFDX networks’ analysis. An ambitious objective was set: instead of choosing a single method for the computation, the two most capable algorithms would be implemented: Network Calculus and Forward Analysis. These algorithms were successively implemented, with good results. Some difference between published data and the latencies calculated by the implemented tools was detected for NC. This was to be expected, and is not a indication that the implemented results are not correct. In the case of FA, all the results presented in the literature were matched with no error.

This thesis also made some contributions to academic research, leading to significant corrections on a state-of-the-art article on end-to-end delay calculations in AFDX.

10.1. Future Work

After concluding this implementation work, some recommendations for future work can be given, based on the results obtained and the experience gathered:

- Implementation of the developed algorithms in GMV’s graphical AFDX analysis suite;
- Subsequent study of the pessimism reduction achieved by the serialization effect in the A380-inspired network.

Acknowledgements

I would first like to thank my thesis advisors, André Rocha at GMV and Agostinho Fonseca at Instituto Superior Técnico. A word of appreciation must be expressed to GMV, which provided me with an excellent work environment during the development of this thesis.

I would also like to thank the experts who were essential during the testing process for this implementation project - Frédéric Ridouard and Henri Bauer, both professors at the LIAS Laboratory of the Université de Poitiers, France.

A note of gratitude must also be expressed to Sérgio Duarte Penna, engineer at Embraer. The communication maintained during the work was enlightening, helpful and motivational.

My best friends Diogo Sanches, José Pereira and Samuel Cardoso also deserve a mention, for all the good moments we shared in the last five years.

Finally, I must express my deep gratitude to my parents and to my brother for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of implementation described in this thesis. This accomplishment would not have been possible without them. Thank you.

References


