<http://www.diva-portal.org>

Preprint

This is the submitted version of a paper published in *Journal of Interconnection Networks (JOIN)*.

Citation for the original published paper (version of record):

Jonsson, M., Kunert, K., Kallerdahl, A. (2015)
Analysing AFDX Networks Using End-to-end Response Time Analysis.
Journal of Interconnection Networks (JOIN), 14(4)
<http://dx.doi.org/10.1142/S0219265913500175>

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-29206>

Jonsson, M., K. Kunert, and A. Kallerdahl, "Analysing AFDX networks using end-to-end response time analysis," Journal of Interconnection Networks, vol. 4, no. 4, 2013 (accepted Dec. 2014 and published online Feb. 2015).

Preprint of an article published in Journal of Interconnection Networks, vol. 4, no. 4, 2013 (accepted Dec. 2014 and published online Feb. 2015) dx.doi.org/10.1142/S0219265913500175 © World Scientific Publishing Company <http://www.worldscientific.com/worldscinet/join>

ANALYSING AFDX NETWORKS USING END-TO-END RESPONSE TIME ANALYSIS

MAGNUS JONSSON, KRISTINA KUNERT

*Halmstad University, Box 823,
SE-301 18 Halmstad, Sweden
{magnus.jonsson;kristina.kunert}@hh.se*

ANDERS KALLERDAHL

*Affiliation at the time of this work: Mentor Graphics Scandinavia AB, Therese Svenssons gata 15,
SE-417 55 Gothenburg, Sweden
anders@kallerdahl.se*

Received Day Month Year

Accepted Day Month Year

In this paper, we present a novel real-time analysis framework for AFDX (Avionics Full Duplex Switched Ethernet) networks. The framework, based on end-to-end response time analysis, calculates not only delay bounds, but also the maximum jitter for each VL (Virtual Link) at each hop, which is necessary according to the AFDX standard. Moreover, the framework supports multicasting, i.e., VLs with several paths, and VLs with arbitrary delay bounds, i.e., shorter, longer, or equal to their periods. An analysis method to calculate the worst-case buffer population is included in the framework, as it is important to guarantee that no buffer-overflow occurs. With a performance surpassing that of Network Calculus and comparable with Trajectory Approach, our framework presents a good choice due to its many features and its foundation in well-accepted analysis methods.

Keywords: AFDX, response time analysis

1. Introduction

AFDX (Avionics Full Duplex Switched Ethernet) networks, standardized through ARINC 664 Part 7 [1], adds real-time and fault-tolerance possibilities to switched Ethernet. Even though the standard dictates how to make the network deterministic, it does not describe how to analyse the real-time performance of a typical large set of traffic flows (VLs, Virtual Links, according to the notation in the AFDX standard) traveling through the network. Some attempts have been made (see, e.g. [2]), often based on Network Calculus [3] [4] [5], but Network Calculus has been shown to be pessimistic leading to underutilized networks [2] [6]. Attempts to reduce or eliminate the pessimism have been made [7] [8], but seem not to be practically realizable, due to computational complexity for realistic sets of hundreds or thousands of traffic flows. In [9], the use of offsets is proposed to reduce the upper bounds on end-to-end delay. This is a promising method when it is appropriate and allowed, but the current proposal is still based on the pessimistic Network Calculus. Another delay analysis for switched networks with traffic

with offsets was proposed in [10], but it does not support several priority levels as done by AFDX. Modifications of the scheduling in the switches have also been proposed [11] [12], but such changes are not allowed considering the standard. Moreover, it is assumed in [12] that the local deadline for each hop equals the period of the respective traffic flow, which at least seems like an impractical restriction. In [2], three analysis methods to analyse AFDX networks are compared. One of the methods is quite pessimistic (Network Calculus), one has high computational complexity (model checking), while one does not give guarantees (simulation). The Trajectory approach [13] [14] has shown promise compared to Network Calculus, but the improvements are not significant, and sometimes it even performs worse than Network Calculus [14]. Recently, the Trajectory Approach with static priority support [15] has been adapted to AFDX [16]. Two other recent papers also report methods for delay analysis [17] [18]. The method proposed in [17] is similar to our method in the response-time analysis and in the sense that all delay calculations for the system are iterated until convergence. However, they use a compositional approach to divide the analysis into a separate calculation for each network component. The method reported in [18] also relies on a similar response-time analysis, but again implements the analysis for each network component instead of doing it in an end-to-end fashion as in our case. The analysis method is able to consider not only VLs, but also sub-VLs and several traffic flows over each sub-VL. We believe this can be incorporated easily into our method, utilizing their suggestions, if needed.

In this paper, we present an analysis framework able to calculate end-to-end delay bounds for traffic with arbitrary number of priority levels over an AFDX network. The performance is shown to be better than standard Network Calculus, i.e., our analysis framework is less pessimistic, and comparable with the performance of Trajectory Approach. The framework also calculates the maximum jitter in each switch for each VL, which is necessary according to the AFDX standard, but often missed in other proposed analysis methods. Arbitrary delay bounds are supported, i.e., they can be shorter, longer, or equal to the VL periods. The analysis framework also supports multicasting, i.e., VLs with several paths. Moreover, we have included an analysis method to calculate a worst-case measure of the buffer population, which is important to guarantee that no buffer-overflow occurs.

Our approach relies on the well-known response time analysis for fixed-priority uniprocessor systems, not only being adapted for preemptive scheduling [19], but also supporting arbitrary deadlines [20] (see also [21] for a good summary). Schedulability analysis under the assumption of fixed-priority preemptive scheduling with arbitrary deadlines for arbitrary/worst-case task period phasing was first presented in [22], but the paper did not show how to calculate the response time (the topic was merely touched upon). Fixed-priority scheduling analysis has been used for communication networks before, both for shared-medium networks [23] (extended in [24-26]) and switched networks as elaborated on in the following. In [27], response time analysis for switched networks is presented, but assuming deadline scheduling of the traffic in the switches, which is not allowed in AFDX switches. In [28], response time analysis for ATM networks is presented, but it assumes traffic shaping in the switches, which is only

allowed in AFDX for rejection of traffic from misbehaving ESs or switches. Moreover, rate-monotonic scheduling, i.e., shorter period traffic gets higher priority, is assumed for the analysis. Response time analysis for switched Ethernet is presented in [29]. However, only a single switch is considered. The group extended their results to support multihop networks, but did this by adding the use of pessimistic Network Calculus [30]. Delay analysis for single-switch networks [31] and larger packet-switched networks [6], exemplified with Ethernet, has been proposed, but the focus is on FCFS (First Come First Served) queuing, where priority support is only used to separate guaranteed real-time traffic from other traffic-classes. In [32], scheduling analysis for ATM networks with priority queuing is presented. However, rate monotonic scheduling is assumed, which restricts the usefulness of the analysis. A specific queuing discipline supporting delay and throughput guarantees for priority queued switched networks is presented in [33], but requires specific traffic shapers in the switches, which is not allowed for regular traffic shaping in AFDX switches.

In [34], the authors present response time analysis for ATM networks with fixed-priority scheduling. Even though ATM differs significantly from AFDX, their work still serves as an important base for our work. Our analysis, however, is not implementing response time analysis hop by hop, but instead for the whole path of the VL in consideration, as done in [35] (a source of inspiration). We believe this is an interesting track that opens up for continued research in the field. The analysis method presented in [35] is also based on response time analysis and is developed for AFDX, but it assumes deadlines equal to periods and treats equal-priority VLs in the same manner as higher-priority VLs, which, e.g., for systems with two priority levels means no priority treatment at all in the analysis. Related results by the same group are presented in [36], but only considering ESs.

The remaining paper is structured as follows. This introduction is followed by the specification of the assumed system model in section 2, and the theoretic response time analysis, containing the description of a utilization check, a delay analysis, a hop-dependent response time calculation and the maximum queuing delay jitter at each hop, for an AFDX network is given in section 3. A way of calculating the buffer requirements in the network is described in section 4. A performance evaluation for a given traffic case, comparing our approach with both standard Network Calculus and the Trajectory Approach, is done in section 5, and the paper is concluded in section 6.

2. System model

An AFDX network consists of End Systems (ES) and switches connected by full-duplex point-to-point links (all assumed to have the same bit rate during this paper). Traffic through the network is defined in terms of Virtual Links (VLs), i.e., logical channels or Real-Time Virtual Channels [37]. Each VL v_i is characterized by a period, or in the case of sporadic traffic a minimum interarrival time, called BAG (Bandwidth Allocation Gap), denoted $T_{BAG,i}$ in this paper, a loop-free path (or several in the multicast case, as explained below), a minimum frame size S_i^{min} , and a maximum frame size S_i^{max} , in this paper denoted as $L_{min,i}$ and $L_{max,i}$, respectively, where the frame sizes are expressed in bytes.

BAGs are only allowed to have values equal to 2^n ms, where $0 \leq n \leq 7$, giving BAG values in the range 1 ms to 128 ms. The frame sizes include 20 bytes in total of interframe gap, preamble and start frame delimiter.

Each source ES shapes the traffic to ensure that at most one frame per period (BAG) is injected to the network. Moreover, each VL must have a maximum jitter $T_{J,max}$ defined for each hop (source ES and each switch). The switches on a VL's path should be configured with corresponding maximum jitter values to be able to act (by traffic policing) if any node or switch misbehaves by injecting nonconforming traffic. In other words, those values must be calculated (see next section). The jitter (maximum delay variation) of a VL at the output of a source ES must also be calculated and comply with both of the following:

$$T_{J,max} \leq 40\mu s + \frac{\sum_{\substack{\text{set of VLs} \\ \text{from ES}}} 8 \cdot L_{max,i}}{r} \quad (2.1)$$

and

$$T_{J,max} \leq 500\mu s \quad (2.2)$$

where r is the physical link bit rate. Even though the maximum jitter values must be obtained, the standard does not define how to calculate them.

The standard states that traffic prioritization with two priority levels should be supported, i.e., strict priority queuing should be implemented, but where FCFS (First Come First Served) queues are used for each priority level. Moreover, multicast is supported by the possibility of having several paths per VL. In this paper, we assume that such paths, belonging to the same VL, never join again after being split.

3. Response time analysis

Our approach for analysing the network has its origin in the area of response time analysis for uniprocessor systems [19], a well-known and widely acknowledged method used for providing deadline guarantees in a real-time system context. In order to decide upon the feasibility of a given traffic specification over a given network, two demands have to be fulfilled. Firstly, the utilization has to be checked so that the traffic demands do not exceed the available bandwidth on any physical link. Secondly, a delay analysis has to be conducted in order to confirm that the potential traffic can be serviced in time, i.e., without any deadline misses.

3.1. Utilization check

Given the characterization of a VL v_i , the utilization U_i of one such traffic flow is calculated according to the following equation.

$$U_i = \frac{\frac{L_{max,i} \cdot 8}{T_{BAG,i}}}{r} \quad (3.1)$$

In this, r denotes the bit rate over the physical link for which the utilization is calculated, and $T_{BAG,i}$ signifies the period for v_i . In order to pass the utilization check, the summed utilizations, U , of all VLs over any physical link must not exceed 1, i.e.,:

$$U = \sum_{i=1}^V U_i \quad (3.2)$$

where V is the number of all VLs over the studied hop (output queue plus following physical link), and

$$U \leq 1 \quad (3.3)$$

This condition must hold in order to not have overflow on any link, but does not give any delay guarantees by itself.

3.2. Delay analysis

When studying the workload, i.e., the bandwidth demand over a certain interval of time, that the VLs are imposing on the network, we take an end-to-end approach to the analysis, not looking at one hop at a time, but examining a whole VL, and the VLs interfering with it at any instance in time, as one virtual resource. We are calculating end-to-end response times, indicating the longest time a frame can maximally take from arriving at the queue at its source ES to arriving at the queue at its destination ES. Technological latencies, as, e.g., processing delays, in the source and destination ES are not considered in this paper, but can easily be added as constants. The response time $T_{R,i}$ for v_i is given by:

$$T_{R,i} = \max_{\substack{\forall n_i \in [0; n_{max,i}] \\ \forall i \in [0; V]}} \{T_{r,i,n_i} - n_i \cdot T_{BAG,i}\} \quad (3.4)$$

where T_{r,i,n_i} is the response time for the frame sent during instance n_i of v_i since the beginning of the level- i busy-period T_{busy} (defined by [22] and explained below). n_i denotes the instance of v_i , i.e., how many periods have passed since the beginning of T_{busy} . The maximum number $n_{max,i}$ of instances for v_i is calculated as:

$$n_{max,i} = \left\lfloor \frac{T_{busy}}{T_{BAG,i}} \right\rfloor \quad (3.5)$$

A busy-period is the interval of time from when all VLs' periods start simultaneously until no more frames are queued up for sending and the physical link becomes idle. In a level- i busy-period, the calculation of the busy-period is based only on VLs with priority i or higher. According to [22], but mapped on a communication context, the frames belonging to v_i will experience their longest end-to-end response time during a level- i busy-period when all VLs' periods start simultaneously. As stated above, we consider in our context a virtual resource with all the physical links along the path of the considered VL. T_{busy} is therefore referring to the use of this virtual resource. From an aggregated worst-case analysis point-of-view, we assume the periods for all VLs are synchronized in a way so they (can) start using the virtual resource at the same time. When analysing how

each interfering VL contributes to the aggregated delay, as described later in the paper, a specific worst-case situation for a specific hop is analysed instead. Aggregating worst-case delay components in this way constitutes a potential source for pessimism in the analysis, but ensures that no incorrect delay guarantees are given. As T_{busy} is unknown, the upper limit for n_i , i.e., $n_{max,i}$, has to be found by means of an iterative calculation instead. n_i is incremented by one until the following inequality holds (as explained and motivated in [34]):

$$T_{R,i} \leq (n_i + 1) \cdot T_{BAG,i} \quad (3.6)$$

Eq. (3.4) calculates the worst case response time for v_i by selecting the maximum relative response time (from the beginning of any period) amongst all relative response times, where each relative response time is calculated by taking an absolute response time (measured from the beginning of T_{busy}) and subtracting the number of periods already passed. The absolute (not relative to the start of the current period but to the start of the first period) response time T_{r,i,n_i} , modeled as the workload on the link, is given by:

$$\begin{aligned} T_{r,i,n_i} = & n_i \cdot T_{max,i} + S_i \cdot T_{switch} + (S_i + 1) \cdot (T_{max,i} + T_{prop}) \\ & + T_{iflp,S_i,i} + T_{ifsp,S_i,i} + T_{ifhp,S_i,i} \end{aligned} \quad (3.7)$$

The first summand takes the number of passed instances before the start of the current period, n_i , times the maximum length of a data frame $T_{max,i}$ of v_i , resulting in the maximum workload put on the link by v_i during the periods before the start of the current one. The second summand is the delay for any packet introduced by the switches along its way, where S_i is the number of switches v_i traverses and T_{switch} indicates the upper-bounded processing and switching time in each switch. The third summand denotes the workload introduced by the frame in the current period instance n_i , multiplying the number of hops for v_i between its source and destination ES with the sum of the maximum transmission time of its data and the propagation delay T_{prop} . The propagation delay is assumed to be identical for each hop, but can easily be generalised.

Up to now, only delays introduced by v_i itself have been explained for the calculation of the response time; however, each packet of v_i can experience a delay introduced by other VLs as well. Summand number four in eq. (3.7) accounts for interference with VLs that have lower priority than v_i , but can block the transmission of a data frame belonging to v_i due to the nonpreemptiveness of communication on a packet level. This means that even if a data frame with higher priority than the one being sent at the moment arrives at the queue, the currently on-going frame transmission will not be stopped in order to give way for the newly arrived packet. The maximum time the new, higher priority, packet can be blocked is the time it takes to send one maximum size data frame on all hops, i.e.,

$$T_{iflp,S_i,i} = \sum_{h=0}^{S_i} \max_{\forall v_j \in lp(i,h)} \{T_{max,j}\} \quad (3.8)$$

where h denotes which hop along the path of v_i , and $lp(i, h)$ is the set of all VLs with lower priority than v_i that share hop h with v_i . Hop zero is defined to be the hop from the source ES to the first switch.

Theorem 3.1. The delay v_i is experiencing due to the interference by other VLs with lower priority than v_i will never be longer than $T_{\text{inf}, S_i, i}$.

Proof. Since strict priority queuing is enforced, except for the nonpreemptive transmission of lower-priority frames, a frame belonging to v_i will never be delayed again in the same output port queue by any lower-priority packet when the transmission of the first interfering lower-priority frame is finished. The total delay contribution by lower-priority frames on the whole path of v_i can obviously never be higher than the sum of the delay contributions by lower-priority frames for all hops along the path. $\square$

Summand five in eq. (3.7) accounts for the interference of v_i with other VLs with the same priority as itself. The worst case in this situation is that all VLs with the same priority as v_i are ahead of v_i in its queue. However, only the last shared hop for every interfering VL is included. This is due to the “pay multiplexing only once principle” introduced in [38]. According to this principle, a VL v_x can only be blocked by another VL (of the same or lower priority) the very first time they meet each other at a switch. Once it has been delayed due to their first meeting it will experience no further delay, instead the delay will stay the same, as the packets pertaining to v_x will always arrive after the ones pertaining to the VLs encountered earlier. The reason to include the last shared hop is that the accumulated jitter, which directly affects the interference, is always equal or higher in this hop as compared to former hops.

So the interference T_{ifj} (as defined in eq. (3.10)) experienced by a frame belonging to v_i is calculated as a function of the maximum queuing delay $T_{Q,j,hop(v_j,v_i)}$ of a frame pertaining to v_j at $hop(v_j,v_i)$. Generally, $hop(v_x,v_y)$ denotes the index of the last hop on the path of v_x that v_y shares with v_x . $T_{ifsp, S_i, i}$ is therefore calculated as the sum of the interferences of all interfering VLs v_j that have the same priority as v_i , i.e., :

$$T_{ifsp, S_i, i} = \sum_{\forall v_j \in sp(i, S_i)} T_{if, j} \left(T_{Q, j, hop(v_j, v_i)} \right) \quad \text{where } i \neq j \quad (3.9)$$

where $sp(i, S_i)$ is the set of all VLs with the same priority as v_i , sharing at least one hop with v_i . The interference with frames belonging to v_i itself ($i = j$) can be omitted since eq. (9) adds $T_{max, i}$ for each instance. The transmission time of the maximum amount of information belonging to v_x that might be stored in the queue during any time interval of length T_L is given by:

$$T_{if, x}(T_L) = \left\lfloor \frac{T_L}{T_{BAG, x}} \right\rfloor \cdot T_{max, x} + \min(T_L \bmod T_{BAG, x}, T_{max, x}) \quad (3.10)$$

Lemma 3.1. The maximum interference on a frame belonging to v_i by another VL v_x , with the same priority as v_i , in a FCFS queue (such as a specific priority-level queue) can never be larger than the maximum amount of traffic belonging to v_x that can reside in the queue when the frame belonging to v_i arrives.

Proof. This is obvious when considering that frames in a FCFS queue are never reordered after arrival. Eq. (3.10), with the maximum queuing delay for a specific hop and interfering VL as the argument, will thereby calculate the maximum possible interfering traffic by this specific VL in this specific hop. $\square$

Theorem 3.2. The delay by which v_i is delayed through the interference by other VLs with the same priority as v_i will never be longer than $T_{ifsp,S_i,i}$.

Proof. As can be seen in the derivation of $T_{Q,k,i}$ below, the following always holds true: $T_{Q,k,x} \leq T_{Q,(k+1),x}$. The maximum interference by v_x is therefore reached in the last interfering hop. According to the “pay multiplexing only once principle” [38], as mentioned above, we thereby only need to consider the interference in the last interfering hop for each interfering VL. The use of eq. (3.10) in eq. (3.9) is, according to Lemma 1, a valid calculation of the maximum interference in this context. $\square$

Summand six in eq. (3.7) finally signifies the interference that v_i experiences by all other VLs with higher priority than itself sharing at least one hop with v_i . With the argument being equivalent as for same priority VLs, the interference by higher priority links is given as:

$$T_{ifhp,S_i,i} = \sum_{\forall v_j \in hp(i,S_i)} T_{if,j} \left(\max \left(T_{Q,i,hop(v_i,v_j)}, T_{Q,j,hop(v_j,v_i)} \right) \right) \quad (3.11)$$

Comparing eq. (3.9) and eq. (3.11), it should be noticed that the argument for the interference delay function is different, i.e., the maximum queuing delay is in one case Q_j , the one of the interfering VL v_j , and in the other case Q_i , the one of the studied VL v_i . The reason for this is that in the case of same priority interfering links the amount of interfering traffic depends on the maximum queuing delay of the interfering VL, while for the case of higher priority interfering links, it depends on the maximum queuing delay of the VL in consideration.

Lemma 3.2. The maximum interference on a frame belonging to v_i by another VL v_x , with higher priority than v_i , in a priority queue (e.g., a priority-queued output port with several priority-level queues) can never be larger than the maximum amount of traffic belonging to v_x that can reside in the priority queue anytime under the duration equal to the maximum queuing delay of a frame belonging to either v_i or v_x in the same priority queue.

Proof. This is obvious when considering that frames belonging to v_x that already left the priority queue when the frame belonging to v_i (or v_x) arrives, or are arriving after the instance when the frame belonging to v_i (or v_x) has left the priority queue (i.e., at the latest at the instance equal to its arrival time plus its maximum queuing delay) cannot contribute to the interference. Eq. (3.10), with the maximum queuing delay for a specific hop and VL, v_i , under delay consideration as the argument, but in a max function together with the corresponding argument for v_x , will thereby calculate the maximum possible interfering traffic by v_x (v_j in eq. (3.11)) in this specific hop.

Theorem 3.3. The delay v_i created through the interference by other VLs with higher priority than v_i will never be longer than $T_{ifhp,S,i}$.

Proof: In line with the proof of Theorem 2, the use of eq. (3.10) in eq. (3.11) is, according to Lemma 2, a valid calculation of the maximum interference in this context. $\square$

3.3. Response time from ES to hop k

While the equations given above merely calculate the response time from the source ES to the destination ES, a generalization can be made in order to be able to calculate the response time for a frame from its source ES until hop k , i.e., to the output queue of the switch at hop k , on the path of the VL. (Actually, this is needed even to calculate eq. (3.9) and eq. (3.11) above, since they rely on the knowledge of the queuing delay at an arbitrary queue.) However, as hops are individual to each VL, i.e., hop k for v_1 might be to a different switch than for v_2 , k_i is introduced to denote VL specific hops. k_i therefore signifies hop k for v_i . The relative response time for VL v_i is then, in accordance with eq. (3.4), given by:

$$T_{R,k_i,i} = \max_{\substack{\forall n_i \in [0; n_{max,i}] \\ \forall i \in [0; V]}} \{ T_{r,k_i,i,n_i} - n \cdot T_{BAG,i} \} \quad (3.12)$$

and $n_{max,i}$ is still calculated as defined in eq. (3.6). The absolute response time to hop k_i can be obtained as:

$$\begin{aligned} T_{r,i,n_i,k_i} = & n_i \cdot T_{max,i} + k_i \cdot T_{switch} + (k_i + 1) \cdot (T_{max,i} + T_{prop}) \\ & + T_{iflp,k_i,i} + T_{ifsp,k_i,i} + T_{ifhp,k_i,i} \end{aligned} \quad (3.13)$$

While the first term in eq. (3.13) is identical to the corresponding one in eq. (3.7), the second term adds the delay introduced to every packet by the k_i switches along its path, and the third term the corresponding transmission and propagation delay for the current instance n_i . Term four to six are the interference experienced by a packet of v_i at hop k_i or earlier by other VLs with lower, equal, or higher priority, respectively. The calculations of these interference delays are given by eq. (3.14) to eq. (3.16).

$T_{iflp,k_i,i}$ calculates the interference created by VLs with lower priority than v_i over any hop h , up to hop k_i as:

$$T_{iflp,k_i,i} = \sum_{h=0}^{k_i} \max_{\forall v_j \in lp(i,h)} \{T_{max,j}\} \quad (3.14)$$

The equation summarizes the transmission times for maximum sized data packets of any VL v_j that is a member of the set $lp(i,h)$ which was defined in connection with eq. (3.8). In contrast to eq. (3.8), the summation only includes interference experienced up to hop k_i .

A corresponding set $sp(i,h)$ was defined for eq. (3.9), containing all VLs with the same priority as v_i that share any hop with v_i . When calculating the interference $T_{ifsp,k_i,i}$ by same priority traffic, as given in

$$T_{ifsp,k_i,i} = \sum_{\forall v_j \in sp(i,k_i)} T_{if,j} \left(T_{Q,j,hop(v_j,v_i)} \right) \quad \text{where } i \neq j \quad (3.15)$$

the set has to be decreased to $sp(i,k_i)$, merely containing all VLs that share any hop with v_i along the path of v_i , but only up to hop k_i , which denoted hop k for v_i . Otherwise eq. (3.15) is equivalent to eq. (3.9), and eq. (3.10) still holds as the definition of $T_{if,i}(T_L)$.

The interference by higher priority traffic given by

$$T_{ifhp,k_i,i} = \sum_{\forall v_j \in hp(i,k_i)} T_{if,j} \left(T_{Q,i,hop(v_i,v_j)} \right) \quad (3.16)$$

corresponds to eq. (3.11), but with the set of higher priority VLs given by $hp(i,k_i)$, containing all VLs with a higher priority than v_i that share any hop with v_i along its path up to hop k_i . The usage of the interference delay function in eq. (3.15) and eq. (3.16) corresponds to the usage in eq. (3.9) and eq. (3.11), still with arguments $T_{Q,i,hop(v_i,v)}$ and $T_{Q,j,hop(v_i,v)}$, respectively.

3.4. Maximum queuing delay jitter at hop k

In order to be able to correctly configure the switches in an AFDX network, the maximum jitter delay at any given hop k has to be calculable. Knowing the relative response time $T_{R,k,i}$ for v_i at hop k , as given by eq. (3.12), the maximum queuing delay jitter experienced by a packet at hop k can be obtained as:

$$T_{J,k_i,i} = T_{R,k_i,i} - (k_i + 1) \cdot (T_{min,i} + T_{prop}) - k_i \cdot T_{switch} \quad (3.17)$$

The jitter is calculated by subtracting the minimum transmission time and propagation time for a frame pertaining to v_i for all hops up to hop k_i , including hop $k_i = 0$ referring to the transmission from the source ES. Furthermore, the (constant) delay introduced by the k_i switches along the path of v_i has to be subtracted as well. If the switch delay is not constant, it should here be exchanged with the minimum switch delay instead of the maximum. The maximum queuing delay for any frame of v_i , consists of the maximum transmission time for the frame plus the maximum jitter for the hop:

$$T_{Q,k_i,i} = T_{J,k_i,i} + T_{max,i} \quad (3.18)$$

3.5. Calculating the equations

To solve the equations, we start with an initialization phase (see Algorithm (1)), where eq. (3.13) is initialized as follows:

$$T_{r,i,n_i,k_i}^* = n_i \cdot T_{max,i} + k_i \cdot T_{switch} + (k_i + 1) \cdot (T_{max,i} + T_{prop}) + T_{iflp,k_i,i} + \sum_{\forall v_j \in hp(i,k_i)} T_{max,j} \quad (3.19)$$

Since the equations in the delay analysis are monotonically increasing, they can be solved through iteration (as done in [17] and [34]). The delay equations are hence calculated for all VLs, paths (in case of multicast), and hops until constant values are reached for all sought after parameters. For interfering multicast VLs, only interference from the path with the longest interference on the currently analysed VL and path need to be considered.

```

/*INITIALIZATION*/
For all v(i)
  For all paths
    For all hops (0..S_i)
      Calculate T_r_init using (3.19)
      Calculate T_Q_init using (3.18)
    End
  End
End

/*ITERATIVE CALCULATION*/
Do
  For all v(i)
    For all paths
      For all hops (0..S_i)
        n(i)=-1
        T_R_max=0
        Do
          n(i)=n(i)+1
          Calculate T_r(i,n(i),k(i)) using T_Q
          If (T_r(i,n(i),k(i)) > T_R_max)
            T_R_max=T_r(i,n(i),k(i))-n(i)* T_BAG(i)
          End
          While (T_r(i,n_i,k_i) > (n(i)+1)*T_BAG(i))
            T_R(i,k(i))=T_R_max
            T_J(i,k(i))= T_R(i,k(i))-(k(i)+1)*(T_min(i)+T_prop)-k(i)*T_switch
            T_Q_new=T_J(i,k(i))+T_max(i)
            If (T_Q_new != T_Q(i,k))
              T_Q(i,k)=T_Q_new
            End
          End
        End
      End
    End
  End
While (any T_Q has changed)

```

Algorithm 1: Calculation of relative response time T_R , queuing delay T_Q , and jitter T_J .

For each combination of VL, path, and hop, the algorithm starts at instance $n_i = 0$, and increases with a step size of one in each iteration, until the following inequality holds true:

$$T_{r,i,n_i,k_i} \leq (n_i + 1) \cdot T_{BAG,i} \quad (3.20)$$

4. Buffer requirements

The maximum buffer population B_{i,k_i} of a specific VL v_i in a specific queue (a specific hop k_i) is calculated (using eq. (3.10)) as:

$$B_{i,k_i} = T_{if,i}(T_{Q,i,k_i}) \cdot r \quad (4.1)$$

The maximum total buffer population $B_{p,k}$ for a specific priority-level queue p for a specific hop k is calculated by considering the maximum queuing delay of any traversing VL:

$$B_{p,k} = \max_{\forall v_i, pri(p)} (T_{Q,i,k_i}) \cdot r \quad (4.2)$$

where $pri(p)$ indicates the priority of the VLs in priority queue p .

Since the maximum queuing delay for a FCFS queue (in this case one of the priority-level queues) is directly proportional to the maximum queuing population, the equation holds true. The maximum total buffer population B_k for the whole output port of a specific hop k , i.e., for all priority levels, is calculated as:

$$B_k = \sum_{\forall p} B_{p,k} \quad (4.3)$$

This aggregation is obviously a potential source for the introduction of pessimism. However, we do not consider buffer memory to normally be a bottleneck since buffer sizes usually are large, and, mainly, because the traffic is well specified and only admitted into the network after admission control. Instead, it is important to be able to guarantee that no buffer overflow occurs. The maximum aggregated buffer population for all output ports of a switch can easily be summed up in the same manner.

5. Performance evaluation

We have compared the performance of our real-time analysis framework to both Network Calculus (standard version as implemented in [34]) and Trajectory Approach (implemented based on [15] [16]). Moreover, we have simulated the network to see the experienced delay. The topology used is shown in Fig. 1, together with the multicast path of one high priority VL, while the traffic parameters and remaining VL unicast or multicast paths are shown in Table (1). The bit rate is assumed to be 100 Mbit/s, the propagation delay is assumed to be negligible, while the switch latency is set to 16 μ s. The period for all VLs is 1 ms, and their transmission time is 80 μ s, including, e.g., headers and interframe gap. The periods of all VLs are started synchronously.

Fig. 1: Topology (-) used in the evaluation, and multicast path (---) of VL1000.

Table 1: Traffic parameters.

VL	Tx ES	Rx ES	Path	Priority
Id	Id	Id		
VL0100	ES01	ES02; ES06	ES01-SW1-ES02; ES01-SW1-SW2-SW5-SW6-ES06	Low
VL0101	ES01	ES04; ES05	ES01-SW1-SW2-SW3-ES04/ES05	Low
VL0301	ES03	ES06	ES03-SW2-SW5-SW6-ES06	Low
VL0800	ES08	ES06	ES08-SW5-SW6-ES06	Low
VL0801	ES08	ES06		Low
VL0802	ES08	ES06		Low
VL1000	ES10	ES01- ES09	See Fig. (1)	High

As seen in Fig. (2), the simulated experienced delay never exceeds the calculated delay bounds, as expected and required. Our analysis framework and Trajectory Approach perform equally well (Trajectory Approach is better for one VL), and they both outperform Network Calculus. We have also tested our analysis framework for a large

system with thousands of VLs, completing the analysis on a normal PC in less than an hour. This means that the framework is practically usable.

Fig. 2. Delay vs VL identifier in delay order.

6. Conclusions

In this paper, we have presented a novel real-time analysis framework for AFDX that shows similar performance in terms of delay bounds as Trajectory Approach. This, in combination with its support of multicast traffic, buffer population calculation, arbitrary number of priority levels, arbitrary delay bounds relative to periods, and per hop maximum jitter calculation, all important for AFDX networks, makes the framework very attractive.

References

1. Aircraft Data Network, Part 7 – Avionics Full Duplex Switched Ethernet Network. *ARINC Specification 664 P7-1*, Aeronautical Radio, Inc., Annapolis, MD, USA, Sept. 2009.
2. H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, “Methods for bounding end-to-end delays on an AFDX network,” *Proc. 18th Euromicro Conf. on Real-Time Systems (ECRTS’06)*, Dresden, Germany, July 2006.
3. J.Y. Le Boudec and P. Thiran, *Network Calculus, Springer Verlag Lecture Notes in Computer Science*, vol. 2050, July 2001.

4. R.L. Cruz, A calculus for network delay, I: network elements in isolation, *IEEE Trans. on Information Theory*, vol. 37(1), pp. 114-131, Jan. 1991.
5. R.L. Cruz, "A calculus for network delay, II: network analysis," *IEEE Trans. on Information Theory*, vol. 37(1), pp. 132-141, Jan. 1991.
6. X. Fan, M. Jonsson, and J. Jonsson, "Guaranteed real-time communication in packet-switched networks with FCFS queuing," *Computer Networks*, vol. 53(3), pp. 400-417, Feb. 2009.
7. M. Adnan, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Model for worst case delay analysis of an AFDX network using timed automata," *Proc. 15th IEEE Int. Conf. on Emerging Techn. & Factory Automation (ETFA 2010)*, Bilbao, Spain, Sept. 2010.
8. M. Adnan, J.-L. Scharbarg, and C. Fraboul, "Minimizing the search space for computing exact worst-case delays of AFDX periodic flows," *Proc. 6th IEEE Int. Symp. on Ind. Emb. Syst. (SIES 2011)*, Västerås, Sweden, June 2011, pp. 294-301.
9. X. Li, J.-L. Scharbarg, and C. Fraboul, "Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis," *Proc. 15th IEEE Int. Conf. on Emerging Techn. & Factory Automation (ETFA 2010)*, Bilbao, Spain, Sept. 2010.
10. M. Weckstén and M. Jonsson, "Less pessimistic worst-case delay analysis for packet-switched networks," *Proc. 13th IEEE Int. Conf. on Emerging Techn. & Factory Automation (ETFA 2008)*, Hamburg, Germany, Sept. 2008, pp. 1213-1219.
11. [11] Y. Hua and X. Liu, "Scheduling design and analysis for end-to-end heterogeneous flows in an avionics network," *Proc IEEE INFOCOM 2011*, Shanghai, China, Apr 2011
12. Z. Qiang, Q. Zhenliang, and L. Hengqing, "Admission control of VL in AFDX under HRT constraints," *Chinese J. of Aeronautics*, vol. 24(2), pp. 195-201, Apr. 2011.
13. H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying and optimizing Trajectory approach for performance evaluation of AFDX avionics network," *Proc. 14th IEEE Int. Conf. on Emerging Techn. & Factory Automation (ETFA 2009)*, Mallorca, Spain, Sept. 2009.
14. H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Worst-case end-to-end analysis of an avionics AFDX network," *Proc Conf. on Design, Automation & Test in Europe (DATE 2010)*, Dresden, Germany, Mar. 2010, pp. 1220-1224.
15. S. Martin and P. Minet, "Worst case end-to-end response times of flows scheduled with FP/FIFO," *Proc. Int. Conf. on Networking, Int. Conf. on Systems and Int. Conf. on Mobile Comm. and Learning Techn. (ICN/ ICONS/MCL '06)*, Morne, Mauritius, April 2006.
16. H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying Trajectory approach with static priority queuing for improving the use of available AFDX resources," *Real-Time Systems*, vol. 48(1), pp. 101-133, 2012.
17. Rox, J. and Ernst, R., "Formal timing analysis of full duplex switched based ethernet network architectures," *SAE Technical Paper 2010-01-0455*, 2010.
18. 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*, Santander, Jan.-Feb. 2012.
19. M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29(5), pp. 390-395, 1986.
20. K.W. Tindell, A. Burns, and A.J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Systems*, vol. 6(2), pp. 133-151, 1994.
21. L. George, N. Rivierre, and M. Spuri, "Preemptive and non-preemptive real-time uni-processor scheduling," *Technical Report 2966, INRIA*, France, Sept. 1996.
22. J.P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," *Proc. 11th IEEE Real-Time Systems Symposium*, 1990, pp. 201-209.

23. K.W. Tindell, H. Hansson, and A.J. Wellings, "Analysing real-time communications: controller area network (CAN)," *Proc. 15th IEEE Real-Time Systems Symposium*, pp. 259-263, 1994.
24. K. Tindell, A. Burns, and A.J. Wellings, "Analysis of hard real-time communications," *Real-Time Systems*, vol. 9(2), pp. 147-171, 1995.
25. R.J. Bril, J.J. Lukkien, R.I. Davis, and A. Burns, "Message response time analysis for ideal controller area network (CAN) refuted," *Proc. 5th Int. Workshop on Real Time Networks (RTN'06)*, Dresden, Germany, July 2006, pp. 13-18.
26. R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35(3), pp. 239-272, 2007.
27. D.D. Kandlur, K.G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5(10), pp. 1044-1056, 1994.
28. S.-K. Kweon and K.G. Shin, "Traffic-controlled rate-monotonic priority scheduling of ATM cells," *Proc. INFOCOM 1996*, vol. 2, pp. 655-662, 1996.
29. Y. Song, A. Koubaa, and F. Simonot, "Switched Ethernet for real-time industrial communication modelling and message buffering delay evaluation," *Proc. of 4th Int. Workshop on Factory Comm. Syst. (WFCS 2002)*, Västerås, Sweden, Aug. 2002, pp. 27-35.
30. A. Koubaa and Y. Song, "Evaluation and improvement of response time bounds for real-time applications under non-pre-emptive fixed priority scheduling," *Int. J. of Production Research*, vol. 42(14), pp. 2899-2913, July 2004.
31. X. Fan and M. Jonsson, "Guaranteed real-time services over standard switched Ethernet," *Proc. of the 30th Annual IEEE Conference on Local Computer Networks (LCN'2005)*, Sydney, Australia, Nov. 2005.
32. T.-H. Lee and A.-B. Chang, "Admission control for ATM networks using rate monotonic traffic scheduling algorithm," *Proc. GLOBECOM 1997*, vol. 3, pp. 1355-1360, 1997.
33. H. Zhang and D. Ferrari, "Rate-controlled static-priority queuing," *INFOCOM 1993*, vol. 1, pp. 227-236, 1993.
34. A. Ermedahl, H. Hansson, and M. Sjödin, "Response-time guarantees in ATM networks," *Proc. 18th IEEE Real-Time Systems Symp.*, pp. 274-284, 1997.
35. M. Tawk, G. Zhu, Y. Savaria, X. Liu, L. Jian, and F. Hu, "A tight end-to-end delay bound and scheduling optimization of an avionics AFDX network," *Proc. IEEE/AIAA 30th Digital Avionics Systems Conf. (DASC)*, Seattle, WA, USA, Oct. 2011.
36. M. Tawk, X. Liu, L. Jian, G. Zhu, Y. Savaria, and F. Hu, "Optimal scheduling and delay analysis for AFDX end-systems," *SAE Technical Paper 2011-01-2751*, presented at SAE 2011 AeroTech Congress & Exhibition, Toulouse, France, Oct. 2011.
37. D. Ferrari and D.C. Verma, "A scheme for real-time channel establishment in wide area network," *IEEE J. on Selected Areas in Comm*, vol 8(3), pp 368-379, Apr 1990
38. J.B. Schmitt, F.A. Zdarsky, and I. Martinovic, "Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once", *14th GI/ITG Conf. Measuring, Modelling and Evaluation of Computer and Comm. Systems (MMB)*, 2008, pp. 1-15.