A Reality-Conforming Approach for QoS Performance Analysis of AFDX in Cyber-Physical Avionics Systems

Boyang Zhou, Liang Cheng
Department of Computer Science and Engineering, Lehigh University, USA.
boz319@lehigh.edu, cheng@lehigh.edu

***Abstract*—AFDX (Avionics Full Duplex Switched Ethernet) is developed to support mission-critical communications while providing deterministic Quality of Service (QoS) across cyber-physical avionics systems. Currently, AFDX utilizes FP/FIFO QoS mechanisms to guarantee its real-time performance. To analyze the real-time performance of avionic systems in their design processes, existing work analyzes the deterministic delay bound of AFDX using NC (Network Calculus). However, existing analytical work is based on an unrealistic assumption leading to assumed worst cases that may not be achievable in reality. In this paper, we present a family of algorithms that can search for realistic worst-case delay scenarios in both preemptive and non-preemptive situations. Then we integrate the proposed algorithms with NC and apply our approach to analyzing tandem AFDX networks. Our reality-conforming approach yields tighter delay bound estimations than the state of the art. When there are 100 virtual links in AFDX networks, our method can provide delay bounds more than 25% tighter than those calculated by the state of the art in our evaluation. Moreover, when using our reality-conforming method in the design process, it leads to 27.2% increase in the number of virtual links accommodated by the network in the tandem scenario.**

I. Introduction

With the development of the avionic industry, an increasing number of devices and functions have been added to the aircraft Cyber-Physical Systems (CPS), which results in more and more data transmissions across the network within aircraft CPS [1]. Airbus has developed AFDX (Avionics Full Duplex Switched Ethernet), standardized as ARINC 664, based on the Ethernet technology to replace ARINC 429 that cannot support a large number of end systems due to its limited bandwidth. AFDX is required to provide deterministic QoS to data flows using FP/FIFO (Fixed-Priority/First-In-First-Out) QoS mechanisms.

Aircraft CPS components such as fly-by-wire, passenger entertainment systems, and visionics devices take advantage of AFDX to implement data transmission, which then affect the performance of the aircraft CPS [2]. For example, a large latency can seriously degrade the utility, usability, and acceptability of visionics devices [3]. Therefore, it is critical to (i) analyze the delay performance of AFDX to ensure that the end-to-end communication delay in aircraft CPS can be bounded, and (ii) increase network utilization under the resource constraint in the process of designing the aircraft CPS. Low network utilization may result in a non-optimal hardware footprint leading to flight weight increases and fuel wastes [2].

The state of the art in delay analysis of AFDX uses NC to estimate the delay bounds of traffic flows. However, it is based on an unrealistic assumption as illustrated in Section III, which introduces pessimism to the analytical results. Thus, the motivation of our work is finding a reality-conforming method to provide accurate delay bound estimations of AFDX, which also helps the design of AFDX networks.

In this research, based on QoS mechanisms including preemptive and non-preemptive scheduling used by AFDX switches in practice, we design a family of algorithms that can search for realistic worst-case delay scenarios. We integrate the proposed algorithms with NC, which enable realistic per-flow modeling of services for accurate delay bound estimations. Our approach is then applied to analyzing tandem AFDX networks and the results show that it can obtain tighter delay bounds than those identified by the state of the art. Using our method in the design process, we can improve QoS in AFDX in terms of delay and network utilization.

In this paper, Section II discusses the background and related work of AFDX and NC. Section III describes the research problems and explains why the analysis by the state of the art deviates from reality. Section IV presents algorithms of worst-case scenario identification, and our reality-conforming methods. Section V evaluates the performance of our approach and its impact on AFDX networks. Finally, we conclude our research and discuss the future work in Section VI.

II. Background Knowledge and Related Work

A. AFDX

AFDX networks take responsibility of airborne data transmission in modern cyber-physical avionics systems. There are two types of devices in AFDX, which are end systems and switches. In AFDX, data exchanges between end systems are performed through virtual links. A virtual link is a unidirectional logic path from one source end system to one or more destination end systems.

Each virtual link has a Bandwidth Allocation Gap (BAG) and a maximum frame size. A BAG defines the minimum time interval between two consecutive frames of a virtual link [4]. The value of the BAG is a power of 2, and the unit is milliseconds. The maximum frame size l_{max} of a virtual link

ranges from 64 bytes to 1518 bytes. The frame size of a virtual link is constrained between 64 bytes and l_{max} bytes.

B. Network Calculus

NC is a well-known tool to analyze network performance based on min-plus algebra. Convolution and deconvolution of functions f_1 and f_2 are basic operations in min-plus algebra, which are defined as

$$convolution \quad (f_1 \otimes f_2)(d) = \inf_{0 \le s \le d} \{f_1(d-s) + f_2(s)\}, \quad (1)$$

$$deconvolution \quad (f_1 \oslash f_2)(d) = \sup_{u \ge 0} \{\{f_1(d+u) - f_2(u)\}. \quad (2)$$

The basic principle of calculating maximum end-to-end delay bounds is to find the upper bound of the incoming cumulative function and the minimum service that the switch can provide. The cumulative function of a flow shows the cumulative data amount varying over time, which is non-negative and non-decreasing. The upper bound of the cumulative function of an incoming flow during any backlogged period is called its arrival curve.

Given that $A(t)$ is the cumulative function of the incoming flow, the arrival curve $\alpha(t)$ of the flow is given by

$$\forall 0 \le s \le t, A(t) - A(s) \le \alpha(t-s). \quad (3)$$

Equation 3 describes the arrival process in the worst case. In this paper, we use leaky bucket arrival curves $\alpha_{\rho,\sigma}(t) = \rho t + \sigma$, where ρ is the data rate and σ is the burst, which profile periodic flows well.

The service curve is a property of the switch. It defines the minimum amount of service that the switch can provide. Suppose $A(t)$ is the cumulative function of the incoming flow, and $A'(t)$ is the cumulative function of the switch's outgoing flow. The service curve $\beta(t)$ of a switch is defined if and only if

$$A'(t) \ge (A \otimes \beta)(t). \quad (4)$$

In this paper, we use a rate-latency service curve $\beta(t)_{R,T} = R(t-T)$, where R is the rate of the link and T is the latency in the switch. When a flow with an arrival curve α passes a switch with a service curve β, the delay bound of the flow is:

$$delay : \forall t \ge 0 : D(t) \le inf\{d \ge 0 | (\alpha \oslash \beta)(-d) \le 0\}. \quad (5)$$

When there are several flows passing the same switch, each flow is assigned a part of the service curve based on the scheduling algorithm. That part of the service owned by the flow is defined as the leftover service curve of the flow.

III. Research Problems

In this section, we first summarize the assumption used by the state of the art for leftover service curve calculations. Then an AFDX example is studied to reveal that such assumption leads to impossible frame processing scenarios in reality. Lastly, we present the realistic scenario and research questions.

A. The Assumption Made by the State of the Art

In the existing work of analyzing AFDX networks using NC [5] [6] [7], the way to calculate the leftover service curve $\beta_i(t)$ of VL_i can be expressed in the following equation:

$$\beta_i(t) = [\beta(t) - \sum_{P(n) \le P(i)} \alpha_n(t) - \max_{P(k) > P(i)} \sigma_k]^+ \quad (6)$$

In Equation 6, $\beta(t)$ is the service curve of the switch, and $\alpha_n(t)$ is the arrival curve of VL_n. σ_n is the burst of the arrival curve of VL_n. $P(n)$ is the priority of VL_n. VL_n has a higher priority than VL_i if $P(n) < P(i)$.

Suppose $\beta(t) = r(t-T)$ and $\alpha_n(t) = \rho_n t + \sigma_n$. The delay bound of VL_i derived from Equation 6 is $D_i = \frac{(\sigma_i + \sum_{P(n) \le P(i)} \sigma_n + \max_{P(k) > P(i)} \sigma_k + rT)}{(r - \sum_{P(n) \le P(i)} \rho_n)}$. The physical meaning of the result assumes that all bits from virtual links with priorities higher than or equal to VL_i arrive in the period of $(\sigma_i + \sum_{P(n) \le P(i)} \sigma_n + \max_{P(k) > P(i)} \sigma_k + rT)/r$ should be transmitted before the first frame of the VL_i. This assumption has two defects, which do not conform to the reality. First, the state of the treat the delay of the switch as a virtual burst rT. According to the result of D_i in the previous paragraph, this virtual burst rT introduces a delay of $\frac{rT}{r - \sum_{P(n) \le P(i)} \rho_n}$, which is larger than T. However, since T is the delay of the switch, which should be a constant, Equation 6 makes the result conservative. Another defect is that the arrival curve uses the fluid model, which means that there are bits arriving at any time point. In AFDX networks, flows are periodic, and incoming cumulative curves are step functions. Thus, the fluid model makes the result conservative. In Section III-B, we present an example showing the difference between the outgoing processing order derived from the state of the art and the reality.

B. The Processing Order Assumed by the State of the Art

Figure 1 shows an AFDX example where the assumption described by Equation 6 does not hold in reality. Figure 1(a) shows the properties of three virtual links. In this example, all three virtual links have the same priority, and VL_1 is the virtual link of interest, which is the flow or virtual link that we want to find the delay bound. The BAG values of VL_1, VL_2 and VL_3 are 2 ms, 4 ms and 8 ms, and the maximum frame size of VL_1, VL_2 and VL_3 is 512 bits. We define the hyperperiod as the least common multiple of periods of all flows, which is 8 ms in this scenario. We define the jth frame of VL_i as VL_i^j in each hyperperiod.

In this example, all three virtual links pass the same switch to the same destination. For the simplicity of illustration, we use 1 Mbps as the rate of the service curve and 1 ms as the latency of the service curve. Thus, the service curve is $\beta(t) = 1 * (t - 1 * 10^{-3})$, where the unit of the rate is Mbps, and the unit of the delay is second.

In each hyperperiod VL_1, VL_2, and VL_3 sends 4 frames, 2 frames, and 1 frame, respectively. Knowing the maximum

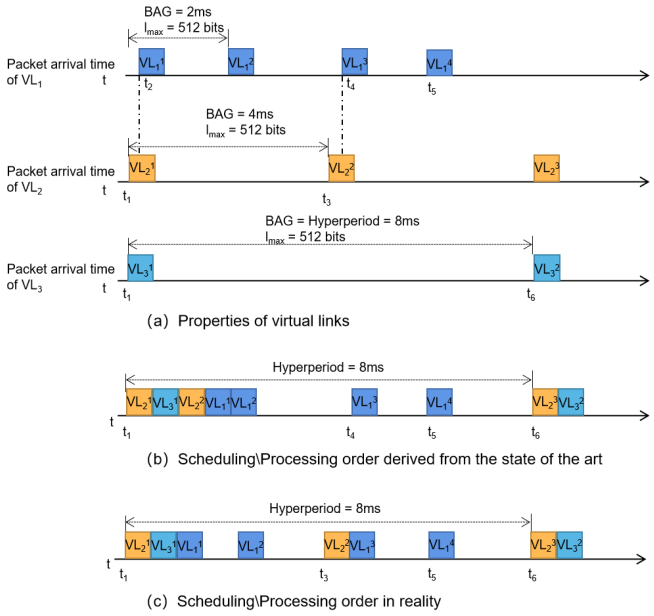

Fig. 1: Worst-case scenarios for frames in VL_1 when all virtual links have the same priority passing through the same switch

frame size l^i_{max} and the BAG value B_i of VL_i, the arrival curve of VL_i can be calculated using Equation 7.

$$\alpha_i(t) = l^i_{max}/B_i * t + l^i_{max} \tag{7}$$

The leftover service curves of VL_i can be derived using Equation 6. Then, using Equation 5, we can get the maximum delay bound of VL_i. For VL_1, the delay bound is about 3.1 ms. The total amount of data being transmitted when VL_1^1 finishes its transmission can be calculated by substituting $t = 3.1 * 10^{-3}$ into $\beta(t)$, which is 2100 bits. The maximum frame size of VL_1^1 itself should be subtracted from 2100 bits. Thus, the amount of data transmitted before VL_1^1 is $(2100 - 512 = 1588)$ bits.

This may imply a worst-case scenario illustrated as Figure 1(b), which shows the processing order of frames assumed by the model (i.e., Equation 6) used by the state of the art. VL_1^1 needs to wait for VL_2^1, VL_2^2, VL_3^1.

However, this worst-case scenario cannot be realized. For example, VL_1^1 must arrive at the switch ahead of VL_2^2. Since the switch has a FIFO policy and VL_1 and VL_2 have the same priority, VL_1^1 must be processed before VL_2^2. The reason why VL_2^2 arrives later than VL_1^1 is that AFDX switch has a maximum allowed jitter. According to the AFDX standard [8], the maximum jitter should be less than 500 μs [9].

C. The Reality and Research Questions

Since the smallest BAG value is 2 ms and the maximum jitter is 500 μs, the first frame of any virtual link should arrive before all second frames from other virtual links if all virtual links start the transmissions at the same time. Therefore, before the transmission of the first frame of the virtual link of interest, there is exactly one frame from all other virtual links that can be transmitted in the worst case. Figure 1(c) shows the worst-case condition encountered by VL_1 in reality. It is clear that the assumed scheduling or processing order in Figure 1(b) leads to an unnecessarily conservative delay estimation.

Based on the previous analysis, we can define the research problems of this paper. (i) *How should we identify reality-based worst-case scenarios for AFDX network analyses?* (ii) *How should the worst-case delay bounds be calculated to reflect the reality for NC analysis?*

IV. Algorithms to find worst-case scenarios and their integration with NC

To solve the first research question mentioned in Section III, which is how to identify worst-case scenarios, we propose two algorithms in AFDX networks under both preemptive and non-preemptive situations. Although frame preemption is not mandatory in AFDX, we develop an algorithm for it because it may be implemented in switches to guarantee the service for high-priority virtual links [10]. We use dynamic programming to find all s_i^j of VL_i, and then find s_i^{max}, where $s_i^{max} = \max(s_i^1, s_i^2, ..., s_i^n)$, and s_i^j is the maximum number of bits from other virtual links transmitted between the arrival time of VL_i^j and the time finishing the transmission of the frame. n is the number of frames of VL_i in each hyperperiod. Thus, s_i^j represents the worst-case scenario encountered by VL_i^j, and s_i^{max} represents the worst-case scenario for VL_i.

A. An algorithm for the worst-case scenario with frame preemption

To find s_i^{max} under preemptive scenarios, we develop an algorithm as shown in Algorithm 1, which is called Seeking Frames with Frame Preemption (SFFP), to identify the worst-case scenario experienced by the virtual link of interest. In this algorithm, VL_i is the virtual link of interest. hp is the set containing all virtual links that have priorities higher than VL_i. ep contains all virtual links having the same priority as VL_i except VL_i itself, and jt is the minimum jitter of VL_i and all virtual links that influence VL_i.

$VL.BAG$, $VL.jitter$, and $VL.l_{max}$ represent the BAG value, the jitter, and the maximum frame size of virtual link VL, respectively. $temp$ is the total size of all the frames transmitted ahead of VL_i^j plus the size of VL_i^j itself because the transmission of VL_i^j can be interrupted by high-priority frames. Array l stores the number of frames of virtual links in hp that can be transmitted before VL_i^j. For example, $l[2] = 4$ means that there are 4 frames belonging to $hp[2]$ transmitted before VL_i^j.

To find s_i^{max}, we need to check all frames of VL_i in each hyperperiod. The *for* loop in Line 8 is used to find all s_i^j for n frames of VL_i. Since $temp$ is used to record the total size of all the frames transmitted ahead of VL_i^j plus the size of VL_i^j itself, we can derive s_i^j from $temp$ in every iteration. We initialize $temp$ to the sum of all maximum frame sizes of virtual links in hp and ep plus the maximum frame size of VL_i as there is at least one frame from each virtual link that

Algorithm 1: Seeking Frames with Frame Preemption

Input: Set hp, Set ep, Set lp, VL VL_i, Bandwidth R, Hyperperiod H.
Output: s_i^{max}, which is the data quantity w.r.t to the largest delay experienced by VL_i.

1 $k \leftarrow hp.length$, $l \leftarrow$ an array of k element, $n \leftarrow \frac{H}{VL_i.BAG}$
2 $l[1...k] = 1$, $jt \leftarrow VL_i.jitter$, $flag \leftarrow 1$, $res \leftarrow 0$
3 $temp \leftarrow VL_i.l_{max}$
4 **for** VL *in* hp **do**
5 $\quad temp \leftarrow temp + VL.l_{max}$, $jt \leftarrow \min\{jt, VL.jitter\}$
6 **for** VL *in* ep **do**
7 $\quad temp \leftarrow temp + VL.l_{max}$, $jt \leftarrow \min\{jt, VL.jitter\}$
8 **for** $j \leftarrow 1$ *to* n **do**
9 $\quad$ **if** $j \neq 1$ **then**
10 $\quad\quad$ **for** VL *in* ep **do**
11 $\quad\quad\quad$ **if** $((j-1) * VL_i.BAG \% VL.BAG == 0)$ **then**
12 $\quad\quad\quad\quad temp = temp + VL.l_{max}$
13 $\quad$ **while** $flag$ **do**
14 $\quad\quad flag \leftarrow 0$, $m \leftarrow 0$
15 $\quad\quad$ **while** $m < k$ **do**
16 $\quad\quad\quad$ **if** $(temp/R) + jt \geq hp[m].BAG * l[m]$ **then**
17 $\quad\quad\quad\quad flag \leftarrow 1$, $l[m] \leftarrow l[m] + 1$
18 $\quad\quad\quad\quad temp \leftarrow temp + hp[m].l_{max}$
19 $\quad\quad m \leftarrow m + 1$
20 $\quad res \leftarrow \max\{res, temp - VL_i.BAG * (j-1) * R - VL_i.l_{max}\}$
21 $\quad temp \leftarrow \max\{temp + VL_i.l_{max}, j * VL_i.BAG * R + VL_i.l_{max}\}$
22 **return** res

has an equal or higher priority than VL_i transmitted ahead of VL_i^1 in the worst case. All elements in l are initialized to 1.

We need to search whether there is more than one frame from each virtual link that can be transmitted before VL_i^j iteratively. Since each virtual link having the same priority as VL_i can contribute at most one frame to s_i^j, we only need to search virtual links in hp. The inner *while* loop in line 15 is used to check whether there are more than $l[m]$ frames that can be transmitted ahead of VL_i^j in $hp[m]$. $(temp/R) + jt$ is the time to finish transmitting VL_i^j and all frames before it. $hp[m].BAG * l[m]$ is the earliest arrival time of the $(l[m]+1)th$ frame of $hp[m]$. Thus, $(temp/R) + jt > hp[m].BAG * l[m]$ represents whether the $(l[m]+1)$th frame of $hp[m]$ transmitted before VL_i^j. If it does, we need to update $l[m]$, $temp$ and $flag$. $flag$ is used to control the outer *while* loop in Line 13. $flag$ is true meaning that there was an update in the previous iteration.

Line 20 is used to compute s_i^j from $temp$ and compare the current s_i^j with the previous maximum value. Because the earliest arrival time of VL_i^j is $VL_i.BAG * (j-1)$, the first $VL_i.BAG * (j-1) * R$ bits of data does not influence the delay experienced by VL_i^j. Then, we also need to subtract $VL_i.l_{max}$ from $temp$ because this term is the size of VL_i^j itself.

Line 21 is used to update $temp$ for the next iteration. The initial number of bits transmitted before $VL_i^{(j+1)}$ equals to $temp + VL_i.l_{max}$. Since the earliest arrival time of $VL_i^{(j+1)}$ is $j * VL_i.BAG$ and the transmission of frame can be interrupted due to the frame preemption, $temp$ must be larger than or equal to $j * VL_i.BAG * R + VL_i.l_{max}$. When the algorithm finishes its execution, res is s_i^{max} that we want to find.

B. An algorithms for the worst-case scenario in the non-preemptive situation

The algorithm for the non-preemptive situation is called Seeking Frames without Frame Preemption (SFNFP). The main difference between Algorithm SFNFP and Algorithm SFFP is that the frame size of VL_i is not considered in $temp$ because the transmission cannot be interrupted. Thus, in Algorithm SFNFP, we remove $VL_i.l_{max}$ from $temp$, and add $mSize$, which is the maximum burst of flows having lower priorities than VL_i, to $temp$. The final result returned by the algorithm is s_i^{max} in the non-frame preemption scenario.

Both algorithms have a time complexity O($numVL * w$), where $numVL$ is the number of virtual links, and w is the average number of frames of all virtual links in each hyper-period. Thus, it is not time-consuming when using them to derive reality-conforming leftover service curves. We discuss how to integrate the results of the two algorithms with NC to calculate delay bounds of virtual links in Section IV-C.

C. Integration of the Algorithms with NC

We put forward an approach to analyzing the delay performance of virtual links using s_i^{max} obtained by the previous algorithms. To analyze the delay performance of virtual links, we first need to calculate the leftover service curve for each virtual link. Theorem 1 provides a way to calculate leftover service curves using s_i^{max}.

Theorem 1: Suppose m time-synchronized periodic virtual links, meaning that all periodic virtual links start the transmission at the same time, pass the same switch, and VL_i is the virtual link of interest. The maximum frame size of VL_i is l_{max}^i. The service curve of the switch is $\beta(t)$. Then, the leftover service curve $\beta_i(t)$ of VL_i can be computed by

$$\beta_i(t) = [\beta(t) - s_i^{max}]^+ \tag{8}$$

, where $[a]^+$ means $\max\{a, 0\}$.
Knowing the leftover service curves, delay bounds can be calculated using Equation 5.

V. Evaluation

In this section, we evaluate our reality-conforming approach by comparing its delay bound estimations with those obtained by the state of the art (i.e. Equation 6). We use a tandem network with different numbers of virtual links as shown in Figure 2 to evaluate the performance of our approach. In the evaluation, we use the DiscoDNC toolbox to perform the calculation of NC [11].

The network consists of 10 switches and n virtual links of the same path. Virtual links are assigned with BAG values from 2 ms to 128 ms, and frame sizes range from 256 bits to

12000 bits iteratively. We vary the number of virtual links n. The bandwidth of switches is set as 100 Mbps, which is the bandwidth of real AFDX switches. The latency of switches is assumed to be 0.1 ms.

Current implementations of AFDX networks use two priorities [12]. Since avionics CPS becomes more and more complicated, additional priorities may be needed in AFDX networks to provide deterministic QoS [13]. Thus, we discuss scenarios where systems have two or eight priorities. The reason for choosing eight priorities is that the same number of priorities is used in TSN standards [14], which are specifically designed for traffic with different QoS requirements, including hard real-time ones.

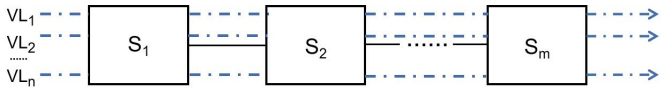

Fig. 2: Network topology and virtual links for evaluations

We compare the delay bounds obtained by our approach and the state of the art. Figure 3 shows the results of the one-priority scenario and the two-priority scenario. We compare two methods when there are 20 and 100 virtual links in AFDX networks.

The previous work (i.e. the state of the art) did not study how to derive leftover service curves when there exists frame preemption in AFDX networks. Since frame preemption eliminates the influence of the low-priority flows on the flow of interest, we use Equation 9 to calculate the leftover service curves when switches support frame preemption [15].

$$\beta_m(t) = [\beta(t) - \sum_{P(n)\leq P(m)} \alpha_n(t)]^+ \tag{9}$$

Virtual links are assigned priorities based on the parity of their ID. For example, VL_1, VL_3, and VL_5 have a low priority, and VL_2, VL_4, and VL_6 have a high priority in the two-priority scenario. Results of two-priority and one-priority scenarios are shown in Figure 3. The x-axis of the figures is the ID of the VL, and the y-axis is the delay bound.

From Figure 3, we can conclude that whatever the scenario is, our approach always derives tighter delay bounds than the state of the art. The reason for it is that our reality-conforming method removes the pessimism introduced by the fluid model and the unrealistic processing order of frames in AFDX. Since we can draw the same conclusion from the results of eight-priority scenarios, we do not show the results of eight-priority scenarios in the evaluation due to the page limit. We calculate the percentage decrease of the delay bounds derived from our method compared with the state of the art to the decrement of delay bounds. The percentage decrease p is defined as

$$p = \frac{md_s - md_r}{md_s}. \tag{10}$$

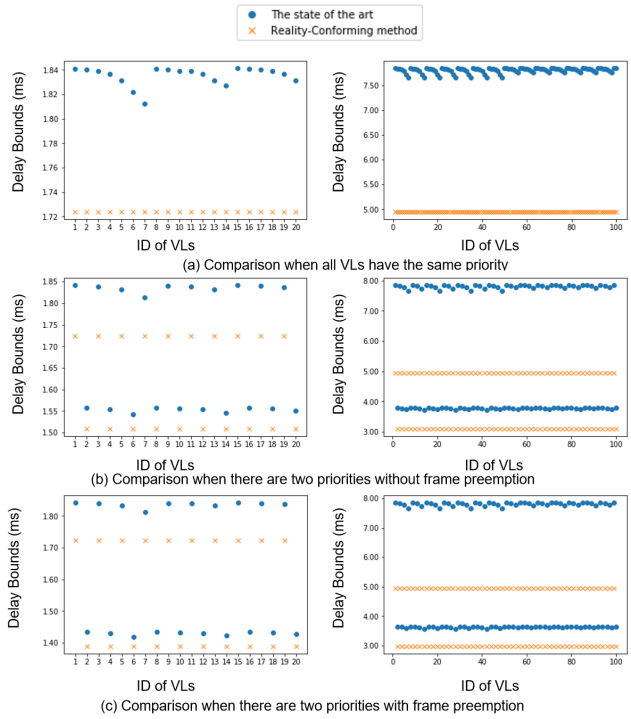

Fig. 3: Comparison between our reality-conforming method and the state of the art in the one-priority and the two-priority scenarios

, where md_s is the average delay bound of the state of the art, and md_r is the average delay bound of our reality-conforming method.

Table I shows the percentage decrease in different scenarios. We can see that when all flows have the same priority, our method achieves the largest improvement. The percentage decrease is not significant when there are only 20 virtual links in the network. However, if we increase the number of virtual links to 100, the percentage decrease increases significantly. In the two-priority and the one-priority scenarios, the percentage decrease is more than 30% when there are 100 virtual links because more pessimism is introduced by Equation 6 when the number of virtual links is large. The decrease of delay bounds of virtual links can help design AFDX networks more efficiently. Using our method, the same network can be designed to accommodate more virtual links.

To show the improvement in the design process, we compare the average delay bounds of our method and the state of the art when the network has different numbers of virtual links. We evaluate the average delay bounds in the two-priority scenario without frame preemption, which is the most popular implementation of AFDX. Figure 4 shows the result. The x-axis is the number of virtual links, and the y-axis is the average delay bounds. The blue line shows the average delay bounds computed by the state of the art, and the orange line shows the average delay bounds calculated using our reality-conforming method. The average delay bounds represent the

TABLE I: Average percentage decrease of the delay bounds using our method compared with those of the state of the art

	One-priority Scenario		Two-priority Scenario		Eight-priority Scenario	
Number of virtual links	With Frame Preemption	Without Frame Preemption	With Frame Preemption	Without Frame Preemption	With Frame Preemption	Without Frame Preemption
20	6.1%	6.1%	4.7%	4.6%	4.7%	4.5%
100	36.6%	36.6%	30.7%	30.5%	25.3%	25.2%

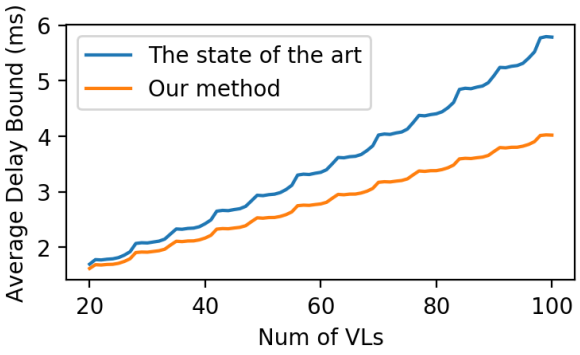

Fig. 4: Average delay bounds of the reality-conforming method and the state of the art when the network has different numbers of virtual links

delay performance of the network.

As we observe from Figure 4, the average delay bound calculated using the state of the art when there are 60 virtual links in the network is 3.39 ms, which is similar to the average delay bound when there are 80 virtual links computed by our reality conforming method. The network utilization increases using our method because the same network can accommodate 20 more virtual links based on analytical estimations in the design process. Thus, our method can help reduce network resource needs in the design, which means that the number of switches used by the network might decrease, leading to the cost and weight reduction of the system [2]. When designing the two-priority non-frame preemption network using our method, the network can accommodate 27.2% more virtual links than the one designed by the state of the art on average according to Figure 4.

VI. Conclusion

In this paper, we have designed a family of algorithms and integrate them with NC to analyze the delay performance of cyber-physical avionics systems backboned by AFDX networks. Using our method, we can achieve more than a 25% decrease of delay bound estimations in tandem scenarios with 100 virtual links compared with the state of the art. Tighter delay bound estimations contribute to a larger number of flows accommodated by the network in the CPS design. Thus, using the same amount of resource, we can achieve a significant increase in accommodating the number of virtual links by using our reality-conforming method in the design process. This QoS performance analysis of AFDX conforming to the reality of cyber-physical avionics systems results in the decrease of the number of switches needed by the AFDX network and the associated cost and weight reduction of the system.

VII. Acknowledgement

This work is supported by NSF Award No. 1646458. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the sponsors of the research.

References

[1] K. Sampigethaya and R. Poovendran, "Cyber-physical system framework for future aircraft and air traffic control," in *2012 IEEE Aerospace Conference*, pp. 1–9.

[2] ——, "Aviation cyber–physical systems: Foundations for future aircraft and air transport," *Proceedings of the IEEE*, vol. 101, no. 8, pp. 1834–1855, 2013.

[3] R. E. Bailey, J. Arthur III, S. P. Williams, and L. J. Kramer, "Latency in visionic systems: Test methods and requirements," in *RTO HFM Workshop*, 2005.

[4] R. Scarduelli, P.-A. Bourdil, S. D. Zilio, D. L. Botlan, and P.-A. Bourdil, "Time-accurate Middleware for the Virtualization of Communication Protocols," *arXiv preprint arXiv:1805.09256*, 2018.

[5] 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, Tech. Rep., 2011.

[6] F. Frances, C. Fraboul, and J. Grieu, "Using network calculus to optimize the afdx network," in *2006 Proceedings of ERTS*.

[7] A. Finzi, A. Mifdaoui, F. Frances, and E. Lochin, "Network calculus-based timing analysis of AFDX networks with strict priority and TSN/BLS shapers," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, pp. 1–10.

[8] R. Mancuso, A. V. Louis, and M. Caccamo, "Improving bandwidth utilization with deterministic delivery guarantees in AFDX through traffic phase-shifting," Tech. Rep., 2015.

[9] J. Ermont, S. Mouysset, J.-L. Scharbarg, and C. Fraboul, "Message scheduling to reduce AFDX jitter in a mixed NoC/AFDX architecture," in *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, 2018, pp. 234–242.

[10] D. Thiele and R. Ernst, "Formal worst-case performance analysis of time-sensitive Ethernet with frame preemption," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–9.

[11] S. Bondorf and J. B. Schmitt, "The discodnc v2: a comprehensive tool for deterministic network calculus," in *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, 2014, pp. 44–49.

[12] T. Hamza, J.-L. Scharbarg, and C. Fraboul, "Priority assignment on an avionics switched ethernet network (qos afdx)," in *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*. IEEE, 2014, pp. 1–8.

[13] O. Hotescu, K. Jaffrès-Runser, J.-L. Scharbarg, and C. Fraboul, "Multiplexing avionics and additional flows on a qos-aware afdx network," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 282–289.

[14] J. L. Messenger, "Time-sensitive networking: An introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, 2018.

[15] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer Science & Business Media, 2001, vol. 2050.