Improving Bandwidth Utilization With Deterministic Delivery Guarantees in AFDX through Traffic Phase-Shifting

Renato Mancuso*, Andrew V. Louis*, Marco Caccamo* *University of Illinois at Urbana-Champaign, USA,
{rmancus2, avlouis, mcaccamo}@illinois.edu

Abstract

The Avionic Full-Duplex Switched Ethernet (AFDX) is a data network certified for avionic operations. AFDX closely follows the IEEE 802.3 (Ethernet) standard for packet forwarding. On top of that, bandwidth enforcement using traffic shaping is performed to provide deterministic delivery guarantees. The design of an AFDX network, however, imposes that bandwidth enforcement is performed at a coarse granularity. This, together with the tight requirements on transmission jitter, determines a low utilization of the physical links.

In this work, we propose traffic phase shifting (TPS) as a way to increase the granularity of bandwidth assignment to nodes of an AFDX network using logic time synchronization among traffic sources. Specifically, we leverage the periodic nature of real-time traffic and use phase-shifing to prevent link congestion. This in turns allows a more fine-grained bandwidth control via the AFDX protocol. We show that TPS leads to significant improvements in terms of per-link utilization without violating predictability.

I. Introduction

Modern aircraft are becoming increasingly complex both in terms of on-board deployed hardware and from a software perspective. Avionic systems are comprised of processing units, sensors and actuators (end-systems) that are deployed across the aircraft body and communicate over a reliable network. From a networking perspective the growth in complexity is both horizontal, as a higher number of interconnected end-systems become part of the avionic network, and vertical, as end-systems transmit higher volumes of data across the network.

The emerging bandwidth requirements, together with the need for deterministic delivery guarantees have determined the adoption of Avionic Full-Duplex Switched Ethernet (a.k.a. AFDX) [1]. AFDX is capable of sustaining a bandwidth that is three orders of magnitude higher than older avionic network standards [2, 3]. AFDX reuses a consolidated packet switching technology by conforming to IEEE 802.3 (Ethernet). On top of that, full redundancy and per-flow bandwidth control is enforced to ensure time predictability, delivery guarantees and fault tolerance.

Thanks to the additional traffic regulation mechanisms, AFDX is able to provide hard real-time guarantees as long as the network is operated below its saturation point. As such, the AFDX network specification imposes strict configuration constraints to prevent: a) overloading of network nodes which may lead to packet drop and b) excessive packet queuing time which may result in unacceptable transmission jitter. However, these constraints heavily impact the achievable channel utilization and thereby negatively affect the number of end-systems that can be configured on the network.

In this work, we propose a technique that exploits traffic phase shifting and buffering (TPS) to significantly improve channel utilization in AFDX networks, while ensuring deterministic packet delivery guarantees. Specifically, TPS allows different traffic flows that have a bursty periodic behavior to be aggregated by time-shifting their packet release time. As such, a) uncertainty about packet arrival at the AFDX switches is reduced, and b) the granularity of bandwidth assignment is increased. These two properties allow for smaller queues to be necessary at AFDX switches, while lowering the bandwidth waste. TPS can be implemented in software by exploiting time synchronization among traffic sources and without modifying neither the underlying protocol nor the AFDX switch hardware.

In summary, this work makes the following contributions:

- Propose TPS as a novel technique to increase the number of configurable network flows in an AFDX network through traffic aggregation;
- Demonstrate that the proposed technique increases the achievable network bandwidth while decreasing packet backlog at the AFDX switches;
- Demonstrate that strict delivery guarantees are met when TPS is used to perform traffic aggregation.

The rest of the paper is organized as follows. Section II provides an overview of the related work. Background information about AFDX networks is provided in Section III, while Section IV introduces the terminology used in this paper as well as the assumptions made to describe TPS. Next, Section V analyzes bandwidth usage in AFDX and describes the main properties of TPS. A heuristic algorithm to perform flow aggregation using TPS is presented in Section VI, while a simulation-based evaluation of TPS is provided in Section IX. Finally, the paper concludes in Section X.

II. Related Work

AFDX has become a commonly used network solution in avionics systems due to its performance and standardized design. This has resulted in many investigations and experiments seeking to improve analysis and performance of AFDX. Charara *et al.* [4] provided a proof of AFDX delivery guarantees and performed an extensive study to bound AFDX end-to-end delay. This ensures the robustness and timing of AFDX. By optimizing flow priorities, [5] proved tighter bounds on the latency of flows and the queue size of network nodes.

The work in [6] uses frame insertion to improve the determinism of frame arrival. To decrease network load due to frame insertion, a Sub-Virtual-Link (Sub-VL) aggregation technique was introduced. While this technique shows some similarities with our approach, it remains profoundly different from the proposed TPS. In fact, it relies on round-robin multiplexing of Sub-VL packets to aggregate different virtual-links. Conversely, the proposed TPS enforces a deterministic packet scheduling at the sources by restricting packet emission times within specific conflict-free windows. This way, packets from different flows can be aggregated within the same virtual-link, realizing both bandwidth utilization improvements and reducing packet backlog at the AFDX switches.

By investigating the design of VLs, [7] proposed methods of saving the valuable bandwidth of an AFDX network. The first method involves computing the optimal Bandwidth Allocation Gap and Maximum Frame Size for a given network flow. The second method provides an algorithm for aggregating messages into so called "super-messages." The final method is concerned with optimizing the problem of routing VLs through the network.

Through the implementation of various strategies, [8] improved the bandwidth that can be utilized by flows in systems composed of multiple networks. The first strategy involved placing a Remote Data Concentrator (RDC) between a CAN bus and the AFDX network to implement frame packing, which reduced AFDX bandwidth utilization of streams from the CAN bus to the AFDX network. The second strategy added Hierarchical Traffic Shaping to the RDC to improve the performance of the CAN bus for streams from the AFDX network to the CAN bus. Work has even been done to improve the interconnection of AFDX with other network architectures used in avionics systems [9]. It is worth noting that many of these works have used Network Calculus [10] as a framework for analysis.

Our work features some similarities with *stream desynchronization*, also known as *offset optimization*. These techniques represent industry practices applied to CAN networks to improve the worst-case response time (WCRT) of periodic data frames by assigning offsets to the transmission of periodic network flows. Methods to analyze WCRT of CAN frames with offset relationships have been analyzed in [11, 12]. Although the proposed TPS performs offset assignment to periodic flows, the mentioned techniques are profoundly different in terms of applicability and goals. First, phase shifting in TPS is used to aggregate traffic flows within the same virtual-link in order to increase bandwidth utilization while providing deterministic delivery guarantees. Conversely, the works on CAN focus on WCRT minimization which represents a secondary objective in AFDX networks. Second, due to significant dissimilarities in the nature of the networks (e.g. broadcast vs. switched, priority-based vs. bandwidth-regulated), different analysis methodologies are required to derive conclusions about traffic behavior.

This work introduces TPS: a novel technique to improve bandwidth utilization in AFDX using traffic phase shifting and buffering. To the best of our knowledge, this is the first work that explores the possibility to aggregate multiple compatible flows over the same VL by enforcing strict packet scheduling at the sources. We demonstrate that, apart from achieving better bandwidth utilization, TPS reduces packet backlog at the switches while providing deterministic delivery guarantees.

III. Background

The evolution in the design process of modern aircraft has determined that increasingly more electronic devices are in control of critical functionalities, often replacing old mechanic, hydraulic, and pneumatic systems. For example, the electronic fly-by-wire [13] system initially introduced on the Enterprise space shuttle in 1977 became part of the standard equipment of the Airbus A320 in 1988. Such equipment is now the only control system in most of the recently produced civil and military aircraft. Digital control systems have often been coupled with high-resolution sensors deployed all over the aircraft body. As a result, the capability of moving a large amount of data across an aircraft in an efficient and reliable way has become a fundamental design constraint for avionic systems.

These needs led to the development of the Avionics Full-Duplex Switched Ethernet (AFDX) [1] whose electrical and protocol specifications are defined in the two standards: ARINC 664 [1] and IEEE 802.3 (Ethernet), respectively. In its simplest formulation, the AFDX specification builds upon a standard Ethernet network by adding explicit traffic bandwidth enforcement and redundancy. Being based on the well established Ethernet standard has determined two main advantages. First, by leveraging on a consolidated technology base, it is able to deliver a bandwidth that is three orders of magnitude higher than previous solutions, such as the ARINC 429 [2] and the MIL-STD-1533 [3] bus. Second, by reusing well understood packet switching strategies, it allows a faster design-to-production cycle.

Figure 1 provides an overview of the elements in an AFDX network. From the figure, it can be noted that legacy processing units can be used to perform control, sampling, and processing of sensor and actuation data. If the produced/required data need to be transferred to/from the AFDX network, the traffic source/destination is paired with an AFDX *end system*. The main purpose of the end system is to perform AFDX encapsulation of outgoing traffic and to unencapsulate incoming traffic. Next,

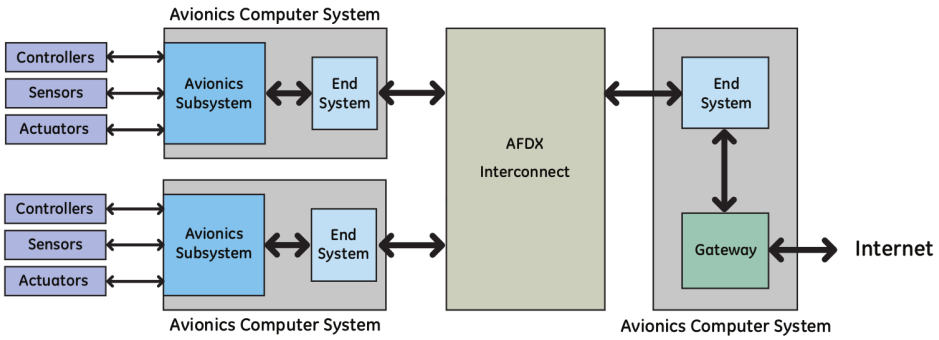

Fig. 1. Overview of AFDX network components [14].

the AFDX interconnect is responsible for routing and packet delivery between groups of end systems. A gateway, paired with an end system, can also be added for Internet connectivity.

What sets AFDX apart from the underlying Ethernet standards and makes it suitable for avionics operations is the mechanism of *virtual links* (VL). A virtual link is an abstraction provided by AFDX that allows the definition of statically routed, unidirectional data flows from a given source to a group of destination hosts (multi-cast). The mechanism of VLs allows aggregating and dispatching different data flows on the same physical channel, while at the same time enforcing explicit bandwidth control. The goal is to provide deterministic packet delivery guarantees. At the edge of each end system, a static configuration groups the traffic generated by different applications into virtual links. Similarly, at the AFDX interconnect, each VL is statically configured to be directed toward a given number of hosts. As a result, the traffic generated by an application will be encapsulated inside a given VL and routed to the configured destinations.

In order to perform per-VL routing, each AFDX end system produces Ethernet frames in which the 48-bit destination field is structured in the following way: the most significant 24 bits contain an AFDX-specific constant; while the least significant bits carry a unique identification number for the VL (the VL ID). Each VL is subject to differential bandwidth control on each AFDX link. This allows partitioning of the physical channel in order to isolate transmissions belonging to different VLs. Bandwidth control is performed by enforcing a minimum inter-spacing (BAG) in the transmission of any two packets belonging to the same VL [1]. This operation is known in Network Calculus as traffic shaping [10]. As shown in Figure 2b, an AFDX switch considers the instant of transmission of the first byte of a packet as the beginning and the end of a bandwidth allocation gap (BAG). In order to perform bandwidth control, two parameters are statically assigned to each VL:

1) **Bandwidth Allocation Gap** (BAG): minimal inter-spacing time of two packets belonging to the same VL
2) **Maximum Length** (L^{max}): maximum size in bytes for a packet transmitted on the considered VL

The BAG parameter specifies the minimal interval of time I that must elapse between the transmission of two AFDX frames on the same VL. According to the AFDX standard [1], the BAG is a 3 bits field whose content BAG is interpreted as:

$$I = 2^{BAG} \text{ ms} \tag{1}$$

Possible BAG values for a given VL range from 1 ms to 128 ms with power-of-two increments. From Equation 1, it follows that a VL with a given BAG parameter will be allowed to forward one packet every I ms on the AFDX interconnect. The achievable bandwidth b of the VL can be calculated once the second parameter, L^{max}, is known for the VL. Equation 2 can be used to calculate the achievable bandwidth given the maximum size of a packet L^{max} and the assigned BAG.

$$b = \frac{8 \cdot 10^3 \cdot L^{max}}{I} \text{ b/s} \tag{2}$$

Thanks to the incorporated bandwidth control mechanism, an AFDX network is capable of enforcing a strict periodicity over packets of the same VL. However packets belonging to different VLs need to be multiplexed on the same physical line. Thereby, if the switch is currently transmitting a packet for VL A, a ready packet from VL B is queued and delayed even if I_B ms have elapsed from the last transmitted packet of VL B. As a result, potentially unpredictable jitter could be accumulated when packets traverse AFDX switches. According to the specification [1], an AFDX network must be designed so that packets do not experience more than a maximum amount of jitter $J^{max} = 500\mu s$.

Finally, AFDX networks use traditional IEEE 802.3 (Ethernet) physical links. However, the maximum bandwidth BW^{max} of currently certified and implemented solutions is equal to 100 Mb/s (i.e. IEEE 802.3 100BASE-TX).

IV. Terminology And Assumptions

Our solution allows to aggregation of *compatible* network flows while preserving deterministic delivery guarantees on the generated traffic. This can be done by performing traffic phase-shifting (TPS) and buffering of packets. In other words, we

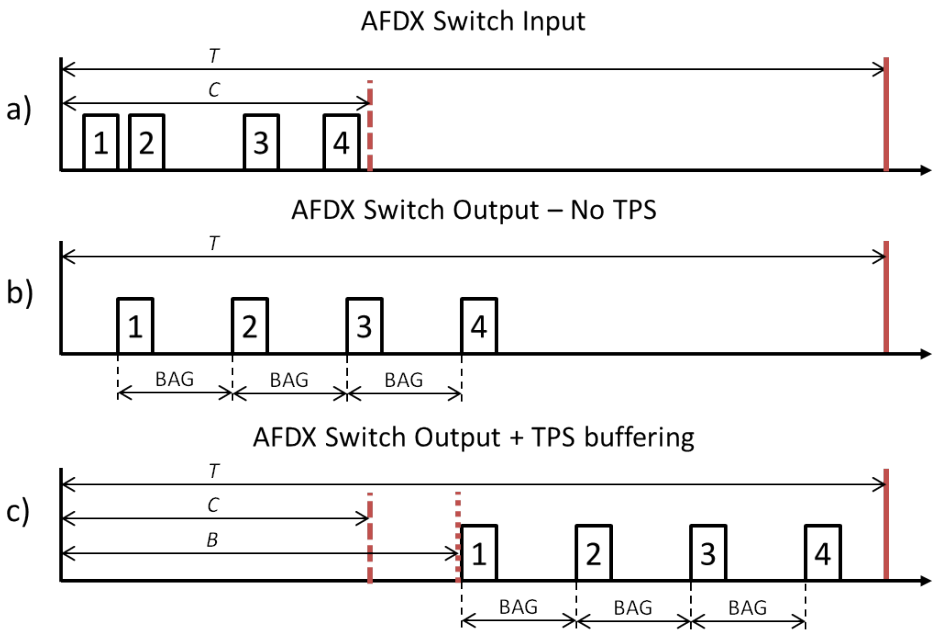

Fig. 2. Bandwidth regulation for a bursty periodic flow (a) crossing an AFDX switch. Both outputs from the switch without TPS (b) and with TPS buffering (c) are shown.

allow two or more traffic flows to appear as a single VL to the AFDX switch, while TPS ensures that packets originated at the sources of the aggregated flows will not reach the switch at the same time. Aggregation of several flows in the same VL does not require additional support at the switch level since AFDX natively supports packet multi-casting.

Specifically, we show how TPS can be used to aggregate multiple network flows that are characterized by *bursty periodic traffic*. The described technique: a) allows the allocation of bandwidth that would remain unused in standard AFDX configurations; and b) ensures deterministic delivery and timing guarantees. This work describes how TPS can be applied at the level of a single AFDX switch, i.e. focusing on a single-hop AFDX network topology. While we restrict the description of TPS on single-hop topologies, our technique can also be extended to work on multi-hop topologies, as discussed in Section VIII. However, we note that specific network topologies may allow additional optimizations. Such optimizations are currently out of the scope of this work and left as a part of future work. Moreover, throughout our discussion we will interchangeably refer to AFDX (single-hop) network and AFDX switch.

In order to capture the behavior of bursty periodic traffic, we adopt a traffic model where each network flow has the following characteristics: a) it generates at most s packets exactly every T time units; and b) there exists a constant C such that all the s packets are emitted before C within each period T, given that $C \leq T$. An example of bursty periodic flow is depicted in Figure 2a. This formulation is rather generic and can be applied to a number of network elements and sensors that produce their output in bursty sequences of packets inter-spaced by the device-specific computation (or sampling) time.

As we detail in Section V, the proposed TPS modifies a given network flow in two ways. First, it shifts the beginning of the first period by a given amount ϕ such that $0 \leq \phi \leq T$. In other words, TPS can modify the phase of a periodic network flow. Second, within each period, it can buffer and delay at the sources all the s packets of the considered flow for an amount of time B, where $C \leq B \leq T$. Figure 2c depicts the effects of TPS controlled buffering on packet transmission.

Since network flows are configured according to the VL abstraction, each flow is also assigned a value of BAG I when configured on the AFDX network. Thus, we can write a flow F as a 6-tuple of the form: $F = \{T, s, C, I, \phi, B\}$. However we use the simplified notation $F = \{T, s, C, I\}$ whenever the parameters ϕ and B are not used and equal to 0. We use S_F to indicate the set of all flows configured on the network, while $N_F = |S_F|$ represents the total number of configured flows.

We assume that intermediate logic is placed between the traffic sources and the AFDX switch, and that time synchronization at a coarse granularity is maintained throughout the system lifetime. Both assumptions can be easily satisfied in real systems, considering two main aspects. First, AFDX networks already require additional logic at the sources to correctly encapsulate traffic[1]. Second, many avionic systems already require time-synchronization to achieve logical correctness. In fact, solutions such as PALS [16] and TTA [17] achieve time-triggered logical synchronization in a system of distributed nodes communicating over a reliable network [18].

Note that the VL mechanism used by AFDX not only allows per-flow bandwidth control, but it also provides a layer of

[1]The ARINC 655 standard [15] introduces Remote Data Concentrators (RDC). RDCs are devices designed to aggregate traffic from clustered sources connected to the AFDX network and to perform proper packet encapsulation/unencapsulation.

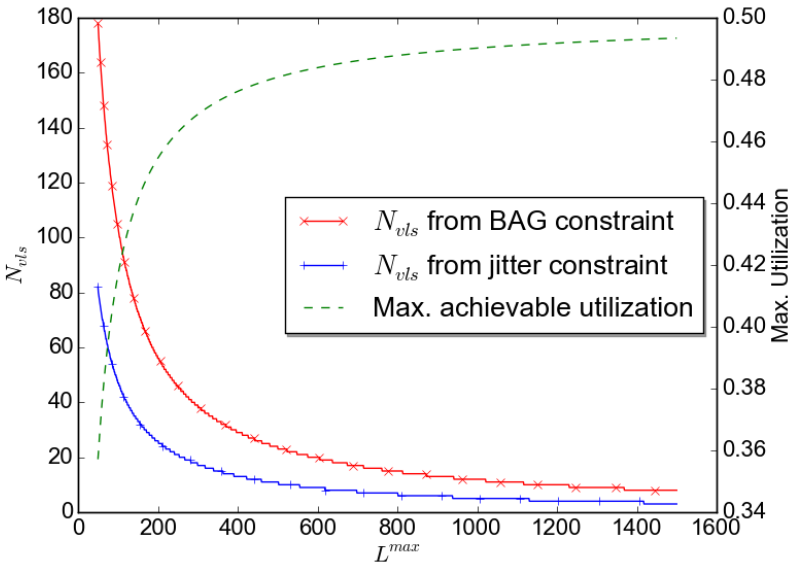

Fig. 3. N_F as a function of L^{max} (left y-axis), and maximum achievable channel utilization as a function of L^{max} (right y-axis).

temporal isolation among flows. When TPS is used to aggregate different flows together, we show that delivery guarantees are ensured, as long as the behavior of the aggregated flows reflects the design-time parameters. Notice that, to make an AFDX network that uses TPS robust against flow misbehavior, it is fundamental that the additional logic at each source discards any packet violating the design-time parameters.

V. Phase Shifting to Improve Utilization

In this section, we identify the characteristics of network traffic that lead to suboptimal channel utilization due to the described AFDX mechanisms. Next we detail our solution, where traffic sources are clustered together providing an increased amount of achievable bandwidth utilization. At the same time, phase-shifting is exploited to decrease unregulated packet queuing, ultimately preventing network congestion.

A. Underutilized Bandwidth in AFDX

As mentioned in Section III, the maximum bandwidth of the physical medium in AFDX networks is $BW^{max} = 100$ Mb/s. Moreover, since AFDX frames follow IEEE 802.3 encapsulation, each packet is transmitted after a preamble of $Pre = 4$ bytes and is followed by an inter-frame sequence of $Ifs = 12$ bytes.

Due to the way bandwidth is enforced at the level of VL, it follows that there exist an upper bound on the number of VLs that can be configured on an AFDX network to ensure deterministic packet delivery guarantees. Let us assume for sake of simplicity that all the flows in S_F have the same value of L^{max} and need to be configured with the minimum value of BAG $I = 1$ ms. Under these assumptions, the maximum number N_F of different flows/VLs configurable on a AFDX switch while ensuring deterministic delivery guarantees can be calculated according to Equation 3.

$$N_F^{bw} = \left\lfloor \frac{10^{-3} \cdot BW^{max}}{8 \cdot (Pre + Ifs + L^{max})} \right\rfloor \tag{3}$$

On the other hand, the constraint on the maximum jitter $J^{max} = 500\mu s$ imposes an additional constraint on the number of configurable VLs. Specifically, the jitter J caused by queuing time at the switch can be calculated according to Equation 4, while the resulting constraint on the number of configurable VLs can be obtained using Equation 5.

$$J = \frac{8 \cdot \sum_{j=1}^{N_F - 1} (Pre + Ifs + L_j^{max})}{BW^{max}} \mu s \leq J^{max} \tag{4}$$

$$N^{jitter} = \left\lfloor \frac{0.5 \cdot 10^{-3} \cdot BW^{max}}{8 \cdot (Pre + Ifs + L^{max})} \right\rfloor + 1 \tag{5}$$

The plot in Figure 3 depicts how the number of configurable 1 ms BAG VLs varies as a function of L^{max}, subject to both bandwidth and jitter constraints. It emerges from the plot and the mentioned equations that: a) the constraint on the jitter is always stricter than bandwidth constraint; and b) the number of configurable VLs rapidly decreases as the maximum packet

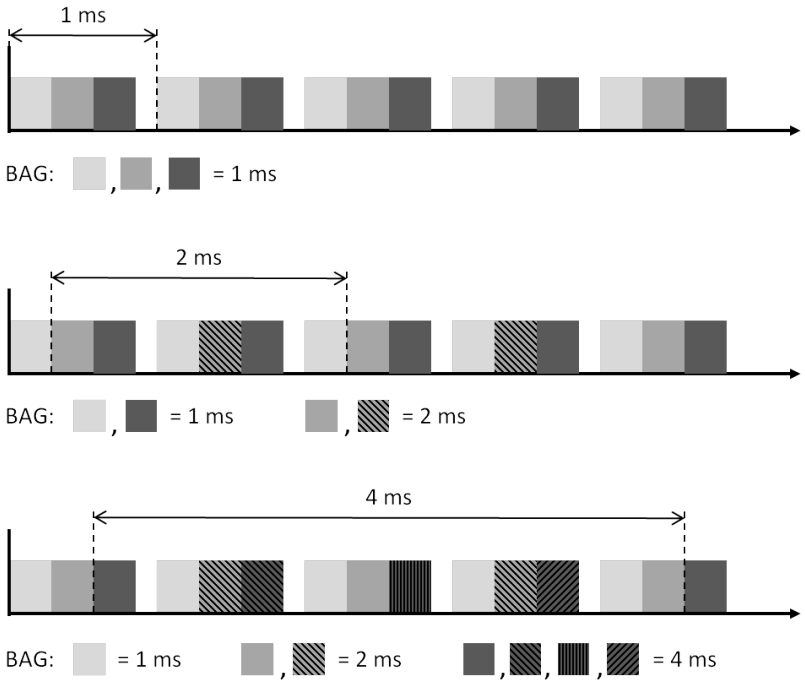

Fig. 4. Possible feasible configurations of AFDX VLs when $N_F^{jitter} = 3$

size increases. In the worst case, with devices that generate packets of maximum allowed size (1500 bytes), only 4 VLs with BAG equal to 1 ms can be configured.

Consequently, meeting both bandwidth and jitter constraints negatively impacts the achievable utilization of the AFDX network. The dotted line in Figure 3 depicts the trend of the maximum channel utilization that can be achieved under the discussed constrains. From the picture, it can be noted that it ranges from 40% when $L^{max} = 80$ bytes to 49% when $L^{max} = 1500$ bytes.

The maximum number of configurable VLs with 1 ms BAG calculated in Equation 5 can also be interpreted as the number of *slots* available for the transmission of packets at the highest achievable bandwidth. In fact, when $N_F > N_F^{jitter}$ VLs are configured, a total of N_F packets can be queued for transmission at the switch at the same time, leading to a violation of the jitter constraint.

Note that even if all the VLs are configured with the maximum BAG of 128 ms, still no more than N_F^{jitter} flows can be configured on the network if no constraint is enforced on their arrival time. Conversely, if additional knowledge about packet arrival time is exploited, it is possible to produce additional feasible configurations. For example, the AFDX network could be configured to have $(N_F^{jitter} - 1)$ VLs with 1 ms BAG and 2 VLs of 2 ms BAG; or $(N_F^{jitter} - 2)$ 1 ms BAG VLs, two 2 ms BAG VLs and four 4 ms BAG VLs, and so on. Figure 4 describes these three possible configurations when $N_F^{jitter} = 3$.

B. BAG Assignment for Bursty Periodic Traffic

Given a generic flow F with the characteristics described in Section IV, we now study how the BAG parameter can be configured to meet delivery constraints.

Recall that AFDX considers the first bit of a packet as the beginning and end of a bandwidth allocation gap (see Figure 2). It must always hold that within each period T, the last of the s packets is transmitted no later than the end of the period. This is fundamental to prevent the queue at the AFDX switch from growing indefinitely. In order to assign a value of I to a flow, we consider a single period T. Next, we calculate the time $\hat{P}$ from the beginning of the period at which the last of the s packets will start being transmitted.

$$\hat{P} = \max(C, I) + J^{max} + I \cdot (s - 1) \tag{6}$$

The equation captures that s packets can be emitted by the source at exactly C. If I is entirely contained inside C, the first packet can start right after C and be queued for at most J^{max}. Otherwise, in the worst case, a packet of the previous period was transmitted exactly at the end of the period. Thus, AFDX will allow the first transmission for the current period only at $I + J^{max}$. Finally, since packets are inter-spaced by I, a total of $I \cdot (s - 1)$ ms are required to start the transmission of the last packet. However, for sake of simplicity and without loss of generality, we will assume in the rest of the paper that $C \geq I$.

It follows that each flow must respect the constraint: $\hat{P} \leq T$. This constraint ensures that a finite amount of backlog is present at the AFDX switch. Thereby, we can write:

$$\hat{P} = C + J^{max} + I \cdot (s-1) \leq T \tag{7}$$

$$\Rightarrow \quad T - C \geq I \cdot s - I + J^{max} \tag{8}$$

since $-I + J^{max} < 0$:

$$\Rightarrow \quad T - C \geq I \cdot s \tag{9}$$

Thus, $T - C \geq I \cdot s$ represents a sufficient condition for flow feasibility. Note that this condition does not necessarily mean that the traffic will be delivered on time from the point of view of the application.

Let us consider a generic stream $F = \{T, s, C, I\}$. The stream F needs to be configured on the AFDX network to have a given BAG I. Given the values of s, T, and C it is possible to derive the largest value of BAG I^{ideal} (i.e. the lowest bandwidth) that can be assigned to the flow while satisfying the delivery constraints:

$$I^{ideal} = \left\lfloor \frac{T - C}{s} \right\rfloor \tag{10}$$

However, AFDX nodes can only express 8 values of BAG ranging from 1 to 128 in power-of-two increments. Thereby, the bandwidth allocation gap I^{AFDX} under traditional AFDX will be assigned as:

$$I^{AFDX} = 2^{\lfloor \log_2 \frac{T-C}{s} \rfloor} \tag{11}$$

To understand how the difference in granularity impacts the bandwidth utilization, let us perform numeric considerations. Suppose that a given flow F_A is periodic every $T_A = 80$ ms, releasing $s_A = 8$ packets no later than $C_A = 17$ every period. In this case, an $I_A^{ideal} = 7$ ms would be sufficient to correctly deliver packets to destination. Under AFDX, however, the largest configurable BAG is $I_A^{AFDX} = 4$ ms. Under this latter assignment, the last packet of each sequence is transmitted no later than $C_A + I_A^{AFDX} \cdot (s_A + 1) = 53$ ms, i.e. 27 ms before the end of the period. Every T_A, The remaining 6 slots with 4 ms BAG remain unutilized and cannot be assigned to a different VL.

In other words, the difference between Equation 10 and Equation 11 reveals an intrinsic waste of resources in the way AFDX is capable of handling periodic bursty traffic. This, in addition to the dimensioning constraints discussed in Section V-A, further reduces the achievable channel utilization.

C. Key insight

To understand how TPS can be used to aggregate multiple periodic flows, consider Figure 5. In the plot, we analyze two periodic flows F_A and F_B. The first flow F_A has parameters $F_A = \{T_A = 7, s_A = 2, C_A = 1, I_A = 2\}$, while $F_B = \{T_B = 7, s_B = 1, C_B = 1, I_B = 4\}$. From Equation 10 we have that BAGs of $I_A = 3$ ms and $I_B = 6$ ms respectively would be enough to multiplex the flows, but in AFDX BAGs of 2 ms and 4 ms need to be assigned (see Equation 11) instead.

Figure 5a illustrates the case of traditional AFDX assignment where wasted packet slots are marked as "e" (empty). Unused resources can be allocated if phase-shifting with buffering at the sources is enforced so to allocate both the flows using only one single 2 ms slot. Specifically, Figure 5b shows how: a) buffering can be used to defer the arrival of packets, and b) phase-shifting on the period of the traffic flows can be used to prevent congestion from hitting the channel. By doing so, packets in flows A and B are regulated at a higher level, allowing them to both be configured with a shared 2 ms BAG slot. This in turn frees slots for additional traffic, whose packets are marked as "2" in the figure.

TPS can be used in two ways. First, as shown in Figure 5, the unused packet slots in a "master" flow are used to aggregate one or more "slave" flows. In this case, the I parameter of the master flow is left unchanged. A second possibility consists in assigning a smaller value I of BAG to the master flow. The result is a compression of the time needed to dispatch the flow packets within the period. Consequently, a larger number of empty slots are freed and made available for additional slave flows.

For instance, consider the master flow $F_A = \{s_A = 8, T_A = 80, C_A = 17, I_A\}$. F_A requires to be configured with $I_A \leq 4$ ms. When $I_A = 4$ ms, no more than 3 slave flows with same parameters could be aggregated. However, if a BAG of $I_A = 1$ ms is assigned instead, the transmission of the flow packets can be buffered and postponed with TPS to occupy 1/10 of the 80 ms period. Therefore, 9 additional slave flows with same parameters can be aggregated. In other words, the number of VLs that can be additionally multiplexed can be significantly increased by efficiently utilizing a single 1 ms slot.

As we show in the following section, the benefit of the proposed technique is twofold: a) reduction in the bandwidth waste originating from the low granularity of BAG values configurable in AFDX; b) reduction in the size of buffers required at the switch, i.e. AFDX node backlog. As we discuss in Section V-E, different flows can be aggregated if they satisfy compatibility requirements. Nonetheless, a good design practice is to aggregate flows that have the same criticality level.

Given a set of flows and the discussed constraints, the problem of assigning values of ϕ and B to compatible flows in order to perform aggregation can be formulated as an optimization problem. A heuristic algorithm to perform flow aggregation is provided and described in Section VI.

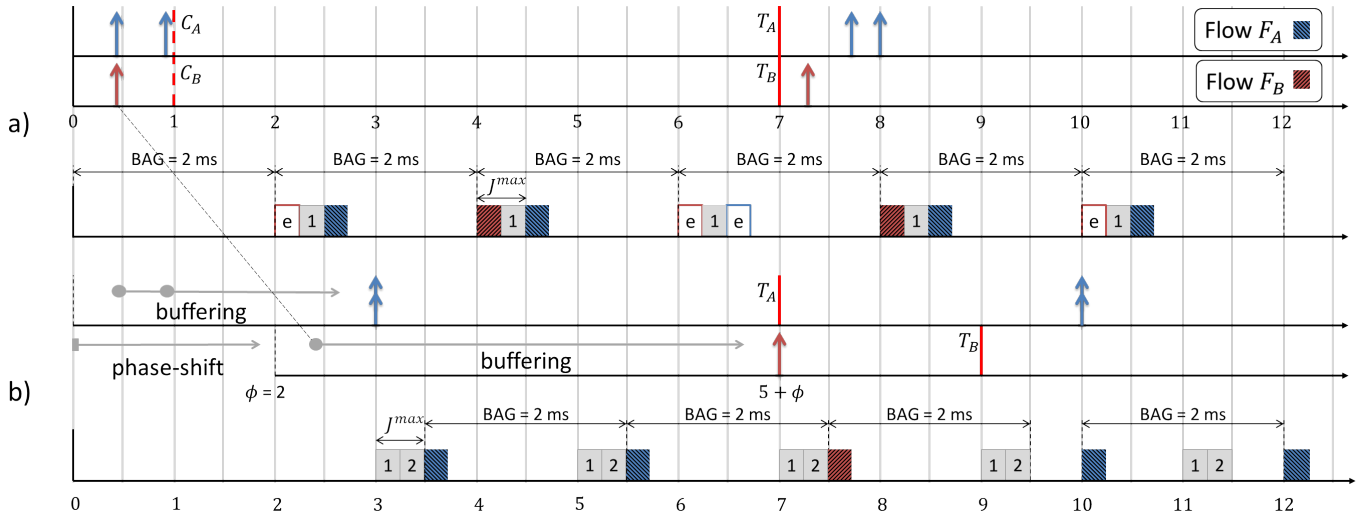

Fig. 5. Buffering and traffic phase shifting can be used to enforce BAG slot sharing on flows (b) compared to traditional AFDX BAG assignment with waste of resources (a).

D. AFDX model

To perform the analysis of our system and introduce the proposed solution, we base our model on the Network Calculus framework [10] and use some of the results obtained for its extension to real-time systems, namely Real-Time Calculus [19]. This mathematical framework has been used to model a wide range of network nodes and traffic scenarios and well fits our problem domain.

In Network Calculus, a flow of incoming packets can be seen as a trace of events $R(t)$, counting how many packets have arrived at the server, an AFDX switch in our case, at time t. Since reasoning for all the possible traces of a system is infeasible, it is possible to reason about the bounds on the number of packets that can reach the server in a time window of length t. This takes the name of arrival curve $\alpha(t) = \{\alpha^u(t), \alpha^l(t)\}$. Where $\alpha^u(t)$ represents the upper bound on the number of emitted packets during an interval of length t, while the lower bound is represented by $\alpha^l(t)$. Formally, it is said that $\alpha(t)$ is an arrival curve for a given flow if and only if for any given flow trace $R(t)$ it holds that:

$$\forall s \leq t : \alpha^l(t-s) \leq R(t) - R(s) \leq \alpha^u(t-s) \tag{12}$$

Similarly to the definition of arrival curve, the number of packets that are processed/forwarded by a server in any given time interval of length t takes the name of service curve $\beta(t) = \{\beta^u(t), \beta^l(t)\}$. Given a backlogged traffic flow, the modeled server will process a maximum of $\beta^u(t)$ packets and a minimum of $\beta^l(u)$ packets during a time window of length t. As in Real-Time Calculus, we will only consider strict service curves.

In this framework, a periodic traffic flow as described at the beginning of the section can be modeled with a stepped arrival curve of the form:

$$\begin{aligned} \alpha^u_{src}(t) &= s\left\lceil \frac{t}{T} \right\rceil \\ \alpha^l_{src}(t) &= s\left\lfloor \frac{t+T-C}{T} \right\rfloor \end{aligned} \tag{13}$$

In order to model the AFDX switch, we consider the time at which the first bit of each packet is transmitted, which corresponds to the beginning/end of the bandwidth allocation gap. Thus, a switch operating according to traditional AFDX with negligible processing delay, will offer to a flow configured with a BAG equal to I a service curve of the form:

$$\begin{aligned} \beta^u_{AFDX}(t) &= \left\lceil \frac{t}{I} \right\rceil \\ \beta^l_{AFDX}(t) &= \left\lfloor \frac{t}{I} \right\rfloor \end{aligned} \tag{14}$$

Consider a flow F with parameters $F = \{T = 80, s = 8, C = 17, I\}$. The plot in Figure 6 depicts the upper, lower bound and a possible trace for such periodic flow. The highlighted region between the $\alpha^u(t)$ and $\alpha^l(t)$ is the region in which a trace for the flow can be contained.

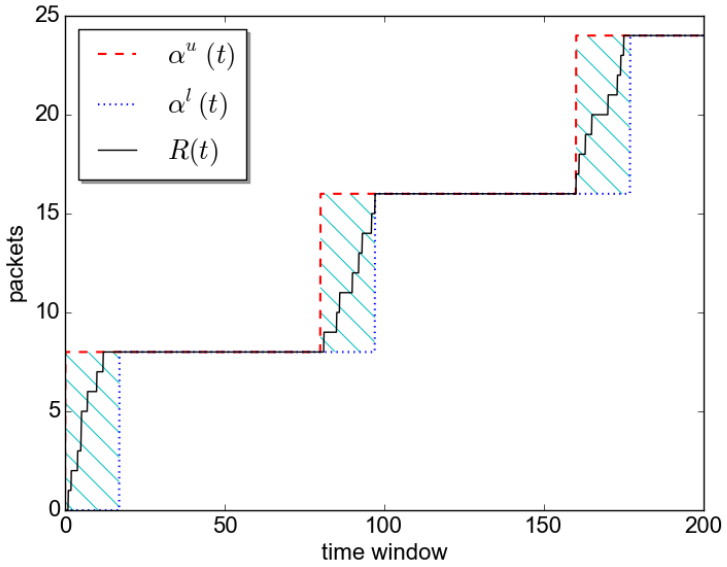

Fig. 6. Upper, lower arrival curve and possible trace for periodic flow $F = \{T = 80, s = 8, C = 17, I\}$.

E. Theoretical results

By performing phase-shifting and buffering of packets at the sources, we effectively transform the original arrival curve of the periodic flow (see Figure 6) into a new arrival curve with the following characteristics:

1) packets are released at the latest instant of time inside each flow period that guarantees a full delivery before the expiration of the period using the assigned BAG;
2) flows that are aggregated on the same slot are shifted with respect to each other so that any two packets from two different flows are never emitted at the same time.

In order to derive the new resulting arrival curve, we assign two additional parameters for each considered flow: ϕ and B. As mentioned in Section IV, ϕ represents the amount of shifting within the period of a flow, while B represents the instant of time within T at which buffered packets are released to the switch. The parameters need to satisfy the following constraints:

1) $\phi < T$ since a flow cannot be shifted more than an entire period T;
2) $B \geq C$ since non-arrived packets cannot be buffered and thus released;
3) $T - B \geq I \cdot s$ to make sure that once the packets are released, there is enough time to transmit them with a BAG I.

Moreover, for a set of aggregated flows with BAG I and period T where F_0 is the master flow and $F_1, \ldots, F_n$ are the slave flows, it must hold that:

1) $\phi_0 = 0$ since the master flow is never shifted;
2) $B_i = T - I \cdot s_i$ to always ensure the transmission of packets from a given flow;
3) $\phi_i = \phi_{i-1} + I \cdot s_i$ to deconflict arrival time of packets from aggregated flows;
4) $I \cdot \sum_{i=0}^{n} s_i \leq T$ to ensure enough time for the transmission of all the packets from the aggregated flows.

The effect of phase shifting and buffering can be modeled as a service node that introduces a maximum processing delay of $\phi + B$, i.e. featuring a lower service curve of the form:

$$\beta^l_{buf}(t) = s\left\lceil \frac{t - \phi - B}{T} \right\rceil \tag{15}$$

Thus, it is possible to chain the effects of buffer and AFDX switch by calculating: $\beta^l_{buf+AFDX} = \beta^l_{buf} \otimes \beta^l_{AFDX}$. The result is provided in Equation 16.

$$\beta^l_{buf+AFDX}(t) = \beta^l_{buf} \otimes \beta^l_{AFDX} = s\left\lfloor \frac{t-\phi}{T} \right\rfloor + \left[\lfloor t - \phi \rfloor - T\left\lfloor \frac{t-\phi}{T} \right\rfloor - r \right] \cdot \left[\left\lceil \frac{t-\phi-B}{T} \right\rceil - \left\lfloor \frac{t-\phi}{T} \right\rfloor \right] \tag{16}$$

Figure 7 depicts $\beta^l_{buf}(t)$ and $\beta^l_{buf+AFDX}$ for two flows F_A and F_B of parameters $s_A = s_B = 8$ packets, $T_A = T_B = 80$ ms, $I_A = I_B = 1$ ms, $\phi_A = 0, \phi_B = 8$ ms and $B_A = B_B = 72$ ms. The value of B is selected by meeting the constraint $B = T - I \cdot s$. Some important features can be highlighted. First, the region in which packets can be emitted to the channel

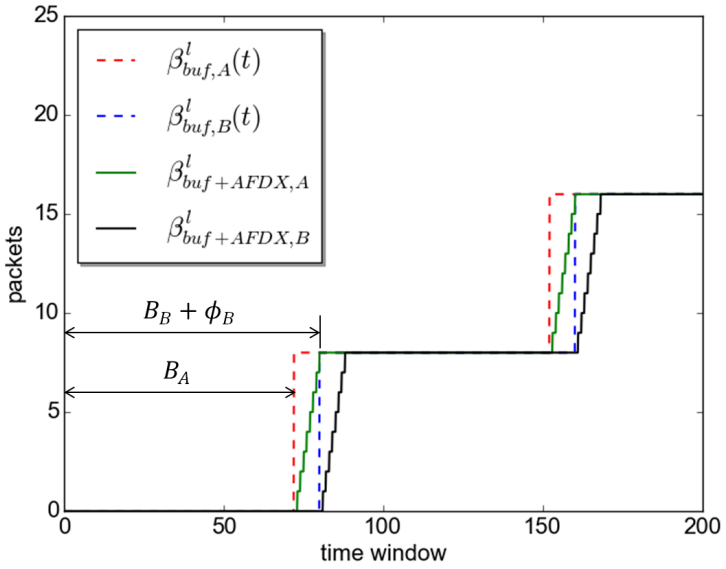

Fig. 7. $\beta^l_{buf}(t)$ and $\beta^l buf + AFDX$ flows $F_A = \{T_A = 80, s_A = 8, C_A = 1, \phi_A = 0, B_A = 72\}$ and $F_B = \{T_B = 80, s_B = 8, C_B = 1, \phi_B = 8, B_B = 72\}$.

is significantly reduced thanks to the controlled buffering that moves their arrival toward the end of each period. Second, the release time B of packets from the buffer ensures that packets, if buffered, are always released in time to be transmitted, with the selected BAG, before the end of each period. Third, that by properly picking the amount of shifting for each flow it is possible to resolve packet collisions at the sources so to a) maintain delivery guarantees, and b) maximize the number of flows allocated on the same BAG slot.

By employing buffering and shifting, we effectively allow different network flows to share the same BAG slot. This can be done only with *compatible* flows. The first requirement for two (or more) flows to be compatible is to have either the same period, or periods that are multiple of each other (harmonic). Additional constraints need to be considered in order to aggregate harmonic flows, as we discuss in Section VII. In avionic systems, this is often true since for ease of design and validation all the safety-critical components are configured with a limited number of different rates. Second, it is fundamental that the total maximum number of packets after the aggregation can be transmitted on time with the selected BAG. Finally, the discussed constraints on ϕ and B must be satisfied for all the aggregated flows.

F. TPS Delay and Backlog

Given the service curve provided in Equation 16 and the upper arrival curve in Equation 13, it is possible to reuse some of the Real-Time Calculus results to compute delay and backlog requirements for the AFDX switch. From [19] we have that the maximum amount of delay on emitted packets can be calculated as:

$$d^{max} = \sup_{\lambda \geq 0}\{\inf\{t \geq 0 : \alpha^u_{src}(\lambda) \leq \beta^l_{buf+AFDX}(\lambda + t)\}\} \tag{17}$$

From Equation 17 it follows that in the worst case, the beginning of the transmission for a packet in flow with arrival curve of Equation 13 will be delayed by at most $T + \phi$. As a consequence, all the packets generated during a (shifted) period will be transmitted before the beginning of the next period. Hence the discussed delivery guarantees follow.

Similarly, the Real-Time calculus framework provides a formula to calculate the maximum amount of backlog at the network nodes of the system. The backlog calculation can be performed according to Equation 18.

$$b^{max} = \sup_{\lambda \geq 0}\{\alpha^u_{src}(\lambda) - \beta^l_{buf+AFDX}(\lambda)\}\} \tag{18}$$

The formula provides the total amount of memory (maximum backlog) required between the source buffer and the AFDX switch. Per each flow, no more than s packets need to be buffered within each period T. However, since the aggregation of flows prevents packets from aggregated flows from hitting the AFDX switch at the same time, the proposed technique has two main benefits:

1) Packets are held in less expensive memories at the sources, while waiting for the buffering time B to elapse;
2) Since only the packets of one aggregated flow at the time hit the AFDX switch, the buffer requirements for a set S_F of aggregated flows is $\max_{S_F}\{s\}$, instead of $\sum_{S_F} s$.

VI. Additional Considerations

In this section, we present an euristic algorithm to perform flow aggregation by using TPS. Moreover, we provide an high-level description about how TPS can be extended to work on more complex network topologies.

A. TPS Aggregation Algorithm

Given a set of flows and the discussed constraints, the problem of finding suitable aggregations can be formulated as an optimization problem. However, in Algorithm 1 we provide a heuristic to perform the aggregation in time $O(n \log n)$, thus making it suitable for online operations and acceptance-test for additional flows. This heuristic produces as an output the value of ϕ, B and I for each flow, given a set of n flows $\widehat{F} = \{F_1, \ldots, F_n\}$.

Initially, empty buckets are created to organize flows based on their period (line 1). This in fact is the fundamental requirement for flow compatibility. Next, at lines 2-5, initial values of BAG are assigned to flows according to Equation 11, and buckets are initialized. The main loop of the algorithm goes from line 6 to 25. For each different value of period in the flow parameters, aggregation is performed on a bucket basis.

First, flows in the same bucket are sorted by initial BAG in ascending order, and by number of packets s in descending order. The flow at the top of the bucket, thus, is used as master flow and its BAG is considered. The loop at lines 11-21 implements the aggregation logic.

The master flows do not change as long as it is possible to aggregate an additional flow with the master. For this reason, line 12 checks that all the packets from all the previously aggregated flows can be transmitted within the period. If the check is passed, the current flow is added to the set aggr_{flows} of flows to be aggregated together. If the condition at line 12 is not met, an appropriate amount of shifting ϕ and buffering B is assigned to all the flows in aggr_{flows} through the $ShiftFlows$ procedure (lines 27-38). In this case, a new master is also selected as the next flow in the bucket. The inner loop terminates when no more flows are left in the bucket, and $ShiftFlows$ is invoked on the leftover flows in aggr_{flows} (lines 23-25).

Given a set of flows to aggregate (aggr_{flows}), the $ShiftFlows$ routine assigns values of phase shifting and buffering. The first element in the set is the master flow. For the master, a shift of $\phi = 0$ is selected (line 31) and a value of $B = T - I \cdot s$ is assigned (line 32). Next, for each flow F a cumulative shift factor cur_ϕ is calculated in line 34 and used as a value of shift ϕ for the slave flows (line 35). Finally, B is calculated, at line 36, as the latest time such that all slave-flow packets can start transmission before T.

VII. Aggregation of Harmonic Flows using TPS

For ease of design and certification, avionic systems are often designed to operate with harmonic tasksets. In an harmonic taskset, all the periods of the tasks are multiple of each other. Assume that the periods of the tasks performing data acquisition at the sensors, processing and actuation are harmonic. Consequently, the periodic traffic generated by the AFDX nodes will also be harmonic. In this section, we discus how TPS can be applied to a set of bursty periodic network flows (as described in Section IV). with harmonic characteristics.

Without loss of generality, we can consider how to aggregate two harmonic flows $F_A = \{T_A, s_A, C_A, I_A\}$ and $F_B = \{T_B, s_B, C_B, I_B\}$ and easily extend the methodology to an arbitrary set of harmonic flows. Since F_A and F_B are harmonic, and assuming that $T_B \geq T_A$, there exists an integer k such that $T_B = k \cdot T_A$. Thus, the key insight to perform aggregation with TPS is to transform the lower rate flow F_B into a set of k sub-flows, so that: a) the sequence of sub-flows is periodic at T_B; and b) the packet transmission deadline of each sub-flow is exactly T_A.

For ease of understanding, let us first assume that the parameter $C_B = 0$ and that s_B is multiple of k. Under these assumptions, F_B can be split in a set of k homogeneous sub-flows. Specifically, if we indicate the sub-flows with the notation $F_{B,i}$, each sub-flow will need transmit s_B/k packets before T_A. Furthermore, the sequence $F_{B,1}, \ldots, F_{B,k}$ will repeat after T_B. In this case, thanks to the homogeneity in the parameters of all the sub-flows, it is possible to simply reason in terms of one single periodic flow $F'_B = \{T_A, s_B/k, 0, I_B\}$. Next, we can relax the assumption on s_B being a multiple of k. In fact, homogeneity can be maintained by considering a number of packets $s'_B = \lfloor \frac{s_B}{k} \rfloor + k > s_B$. As a result, aggregation will be performed between F_A and $F''_B = \{T_A, s'_B/k, 0, I_B\}$.

Next, let us consider values of $C_B > 0$. In this case, when flow B is divided into a sequence of sub-flows $F_{B,1}, \ldots, F_{B,k}$, we identify the specific sub-flow in which C_B falls. We call this sub-flow the *merging sub-flow* and we refer to it as $F_{B,h}$. The index h in the sequence of sub-flows can be calculated as $h = \lfloor \frac{C_B}{T_A} \rfloor + 1$.

When the merging flow is the last in the sequence, i.e. when $h = k$, then all s_B packets need to be transmitted within the merging sub-flow. Moreover, within the merging sub-flow, the relative value of $C_{B,h} = C_B \% T_A$[2]. As a result, the TPS aggregation will be performed between flow F_A and $F'_B = T_A, s_B, C_{B,h}, I_B$.

Finally, when the merging flow is not the last in the sequence $F_{B,1}, \ldots, F_{B,k}$, i.e. when $h < k$, aggregation can be done in two ways. First, the merging flow can be excluded and packets will be transmitted within the sub-flows $F_{B,i}$ such that

[2]Where the operation $a\%b$ (modulo operation) denotes the remainder of the euclidean division of a by b

Data: $\widehat{F} = \{F_1, \ldots, F_n\}, N_{vls}$
Result: $\widehat{F}$ with updated ϕ_i, B_i, I_i

/* Create buckets for flows based on periods */
$\widehat{F}_T = \emptyset$ for $T \in \{T_1, \ldots, T_n\}$;
/* Split flows and assign initial BAG */
for $i \leftarrow 1$ **to** n **do**
  $I_i = 2^{\lfloor \log_2 \frac{T_i - C_i}{s_i} \rfloor}$
  $\widehat{F}_{T_i} \leftarrow \widehat{F}_{T_i} + F_i$
end
/* For each group of flows with same period */
foreach $T \in \{T_1, \ldots, T_n\}$ **do**
  Sort($\widehat{F}_T$ by: I ascending, s descending)
  $\text{cnt}_{pkts} \leftarrow 0$
  $\text{I}_{master} \leftarrow$ FirstElement($\widehat{F}_T$).I
  $\text{aggr}_{flows} \leftarrow \emptyset$
  /* Aggregate flows until the total number of packets can still be transmitted with the same BAG */
  foreach $F_k \in \widehat{F}_T$ **do**
    if $(cnt_{pkts} + s_k) \cdot I_{master} \leq T$ **then**
      $\text{aggr}_{flows} \leftarrow \text{aggr}_{flows} + F_k$
      $\text{cnt}_{pkts} \leftarrow \text{cnt}_{pkts} + s_k$
    end
    else
      ShiftFlows($T, \text{cnt}_{pkts}, \text{aggr}_{flows}$)
      $\text{aggr}_{flows} \leftarrow F_k$
      $\text{I}_{master} \leftarrow I_k$
      $\text{cnt}_{pkts} \leftarrow s_k$
    end
  end
  /* Aggregate the leftover in a single flow */
  if $aggr_{flows} \neq \emptyset$ **then**
    ShiftFlows($T, \text{cnt}_{pkts}, \text{aggr}_{flows}$)
  end
end
/* Given a set of flows to aggregate, assign appropriate values of phase shifting ϕ and buffering B */
Function *ShiftFlows($T, cnt_{pkts}, aggr_{flows}$)* **is**
  $F_{master} = \text{FirstElement}(\text{aggr}_{flows})$
  $\text{min}_{bag} = F_{master}.I$
  $\text{cur}_{\phi} = 0$
  $F_{master}.\phi \leftarrow 0$
  $F_{master}.B \leftarrow T - F_{master}.s \cdot \text{min}_{bag}$
  foreach $F \in aggr_{flows} - \{F_{master}\}$ **do**
    $\text{cur}_{\phi} \leftarrow \text{cur}_{\phi} + F.s \cdot \text{min}_{bag}$
    $F.\phi \leftarrow \text{cur}_{\phi}$
    $F.B = T - F.s \cdot \text{min}_{bag}$
  end
end

Algorithm 1: Heuristic algorithm for flow aggregation

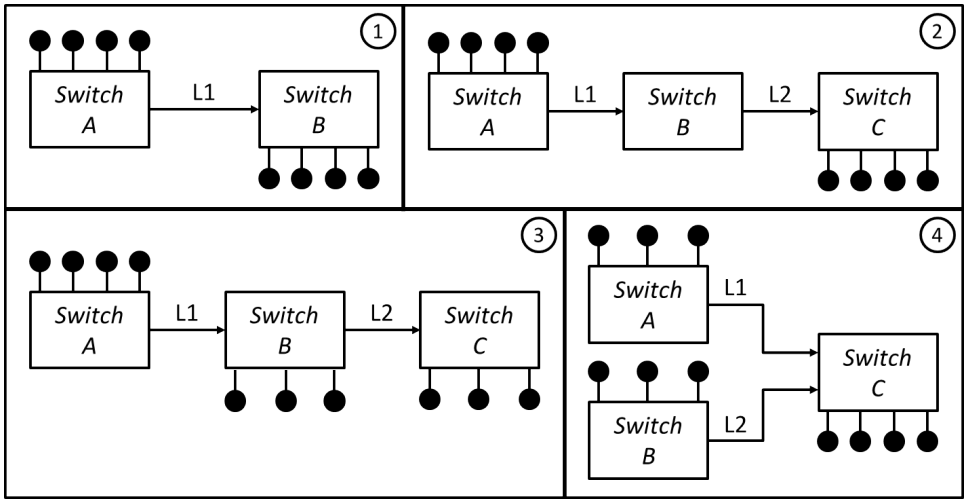

Fig. 8. Considered topologies on which TPS extension is considered.

$i > h \leq k$. The number of packets $s_{sf,1}$ to transmit in each sub-flow will be given by Equation 19. Thus, TPS will be used to perform aggregation between F_A and $F'_B = T_A, s_{sf,1}, 0, I_B$.

$$s_{sf,1} = \begin{cases} \frac{s_B}{k-h} & \text{if } s_B \% (k-h) = 0 \\ \left\lfloor \frac{s_B}{k-h} \right\rfloor + 1 & \text{otherwise} \end{cases} \tag{19}$$

In the second case, packets are partially transmitted during the merging sub-flow, and partially during the remaining $k-h$ sub-flows. But in order to make all the flows homogeneous for aggregation purposes, each sub-flow i will have a value of $C_{B,i} = C_{B,h}$ will need to be used. Finally, the number of packets $s_{sf,2}$ to transmit in each sub-flow will be derived according to Equation 20 and TPS will be used to aggregate F_A and $F'_B = T_A, s_{sf,2}, C_{B,h}, I_B$.

$$s_{sf,2} = \begin{cases} \frac{s_B}{k-h+1} & \text{if } s_B \% (k-h+1) = 0 \\ \left\lfloor \frac{s_B}{k-h+1} \right\rfloor + 1 & \text{otherwise} \end{cases} \tag{20}$$

VIII. TPS on Multi-hop Network Topologies

As studied in Section V, TPS can improve bandwidth utilization while providing traffic delivery guarantees on single-hop AFDX networks. However, it is important to note that these results can be extended to multi-hop network topologies. In this section, we discuss how TPS can be applied to the four basic multi-hop topologies depicted in Figure 8. Such topologies can be considered as basic-blocks to derive more complex multi-hop topologies.

It is important to note that since AFDX is a full-duplex network, it is enough to analyze the behavior of packets in one direction, as if the packets traveling in the other direction were routed through a separate network. Thus, all the topologies in Figure 8 are depicted with directed links.

The first considered extension is depicted in Figure 8-1. In this case, traffic entering Switch A reaches Switch-B through link L1 before being forwarded to all the destinations. Since the two switches exhibit identical behavior, TPS can be applied by reasoning only in terms of Switch A. In fact, if the design is correctly performed according to Switch A, the traffic reaching Switch B will already be shaped and inherently meet delivery constraints. However, the transmission over link L1 may introduce a non-negligible transmission delay d. If such a delay is found to be unacceptable at an application level, TPS aggregation can be performed using a stricter relative deadline D for each flow, i.e. by beginning the transmission of the last packet in each period no later then $D \leq T - d$. This will ensure that the last packet of each period will always reach Switch B before (or exactly at) T.

Figure 8-2 shows that an additional Switch B can be placed between the switches that aggregate sources and destinations. The considerations made for Figure 8-1 directly apply to this case if the packets scheduling policy at Switch B is known to be FIFO. In this case: a) shaped traffic will reach both Switch B and Switch C, while no additional queuing delay will be suffered; and b) given the cumulative delay for traversing links L1 and L2, the deadline of the flows can be adjusted to ensure that packets will always reach Switch C at T or earlier. Conversely, if Switch B features a packets scheduling policy that is not FIFO, arriving packets may suffer additional queuing delay. In this case, the single-hop jitter J calculated in Equation 4 can be suffered at every hop. Thereby, if packets traverse N_{hops} switches, the following constraint needs to be satisfied:

$$N_{hops} \cdot J \leq J^{max} \tag{21}$$

In Figure 8-3, traffic originated at nodes connected to Switch A will be split among destinations connected to Switch B and C. In order to ensure delivery guarantees, it is enough to apply the same considerations as the topology in Figure 8-2. The main difference is that flows traversing both L1 and L2 will be delayed by $d_{1,2}$, which is strictly greater than the delay d_1 seen by the flows traversing only L1. Thus, the adjustment of flows' relative deadlines to account for intermediate link delays will have to be performed accordingly.

Finally, in Figure 8-4, outgoing traffic from Switch A and B merges at the input of Switch C, where all the destinations are connected. In this case, the simplest approach consists in applying TPS locally for each traffic entry point (Switch A and B). In this case, flows will be aggregated only if (1) they are compatible and if (2) their sources are connected to the same switch. Moreover, without any assumption about the packet scheduling policies at the switches, the total number of flows after the aggregation originated at Switch A $N_{F,A}$ and Switch B $N_{F,B}$ will need to satisfy the following jitter constraints:

- $2 \cdot N_{F,A} + N_{F,B} \leq N_F^{jitter}$
- $2 \cdot N_{F,B} + N_{F,A} \leq N_F^{jitter}$

The latter conditions ensure that traffic travelling across the path Switch A-Switch C, and Switch B-Switch C will be transmitted enough without violating the jitter (and bandwidth) constraints.

IX. Evaluation

In order to evaluate the benefits of the proposed technique, we have added support for AFDX nodes and VLs in Network Simulator 2 (NS2) [20]. NS2 is a discrete event simulator that provides support for a number of network elements and protocols, including: TCP, UDP, routing, unicast and multicast protocols over both wired and wireless networks. In order to carry out our evaluation, we have additionally implemented a configurable AFDX node and queue, as well as a periodic traffic source over UDP with the specifications and parameters discussed in Section V-B[3].

In our evaluation, we investigate those quantifiable aspects of our technique that cannot be easily extracted from the theoretical model because they mainly depend on the characteristics of the considered network flows. These aspects involve: (A) what is the saving in terms of BAG slots that arise from the discussed aggregation; (B) what are the improvements in terms of channel utilization that can be achieved by performing flow aggregation; and (C) what is the trend in the amount of required memory at the AFDX switch.

A. BAG Score

We have discussed how phase-shifting and buffering can be used to perform flow aggregation. To quantify the benefits of the proposed technique, we consider the number of BAG slots that are freed for multiplexing additional network flows when aggregation is in use. Specifically, we design our experiment to generate a set of network flows with random characteristics. The generation is performed until all the 1 ms slots are in use. Next, TPS is run on the same set of network flows using the Algorithm 1. As a result of the aggregation procedure, some of the flows in the generated set may be clustered together, resulting in previously occupied BAG slots to be freed. To quantify the saving in terms of slots with a single number, we calculate a *BAG score*. This score summarizes the number of available BAG slots using a single index where higher weight is assigned to higher bandwidth slots. The BAG score can be calculated according to Equation 22.

$$BAG_{score} = \sum_{i=0}^{7} N_{free}^{\{2^i\}} \cdot \frac{1}{2^i} \tag{22}$$

Where $N_{free}^{\{2^i\}}$ represents the number of free slots with 2^i ms BAG. Figure 9 depicts the trend of resulting BAG score for the random sets of flows before and after performing aggregation, as a function of the maximum packet length L^{max}. In this experiment, flows are randomly generated to have a period T ranging between 200 ms and 1000 ms, with a number of packets s and a maximum emission time C uniformly distributed across the length of the period. Unfeasible flows are discarded.

As emerges from the picture, the proposed techniques offer a remarkable improvement in the number of freed BAG slots with smaller values of L^{max}, while providing less benefits for values of L^{max} around 1000 bytes. This is not surprising because, as depicted in Figure 3, the number of available 1 ms BAG slots for VLs is very low (below 10) for values of L^{max} beyond the 600 bytes boundary. This intuitively means that less flows can be generated before exhausting the available 1 ms BAG slots, and that, consequently, less are the compatible flows that can be aggregated.

[3]The developed code is available upon request.

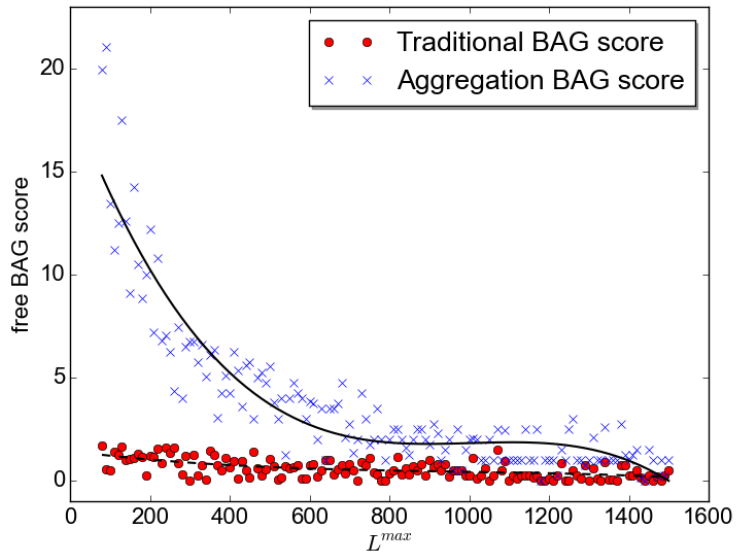

Fig. 9. BAG score for system with and without no flow aggregation.

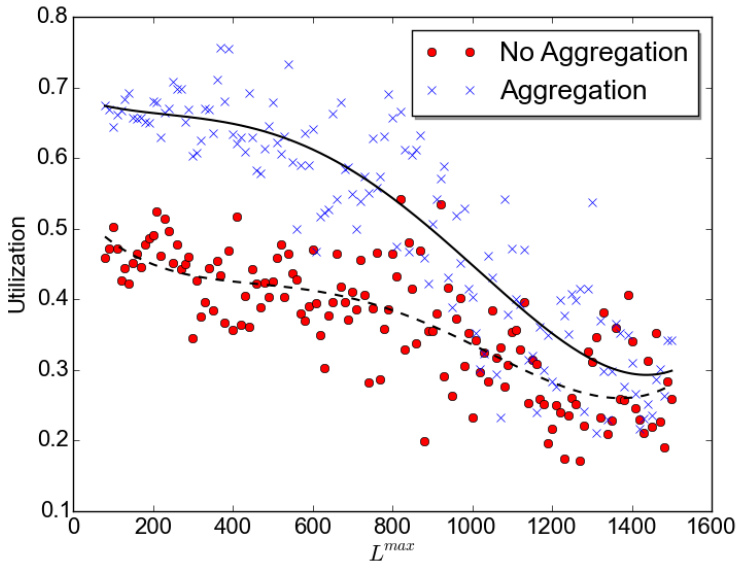

Fig. 10. Resulting normalized channel utilization for random flows with periods between 200 ms and 1000 ms.

B. Channel Utilization

Next, we proceed to study the increase in the utilization that flow aggregation can determine in comparison with what is achievable on a standard AFDX network. Similarly to the previous experiment, we randomly generate a series of network flows that saturate all the available 1 ms BAG slots according to traditional AFDX and its jitter constraint. The obtained set of flows is used to study the achieved utilization of traditional AFDX. On the same set of flows, aggregation is performed like in the previous experiment. This time, however, if the aggregation procedure results in a 1 ms BAG slot being freed, an additional flow is generated and added to the set. The aggregation procedure is invoked until no 1 ms BAG can be freed. Given the two obtained networks, we simulate each system for 20 seconds of virtual time and extract channel utilization from the obtained event trace.

Figure 10 and Figure 11 report the obtained values for channel utilization normalized with respect to the theoretically achievable value of utilization, calculated as in Figure 3. Both the figures plot the resulting normalized utilization as a function of the L^{max} parameter. Moreover, Figure 10 was obtained using flows with periods between 200 ms and 1000 ms, while periods in the range 500 ms to 1000 ms are considered for Figure 11.

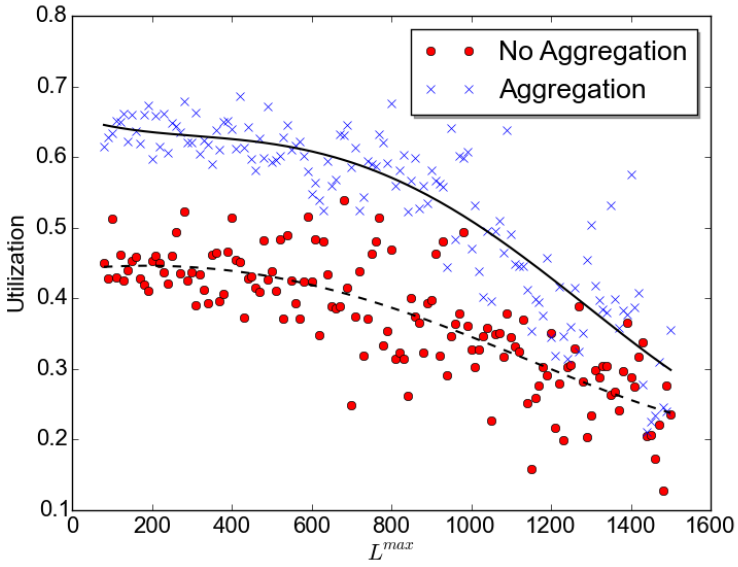

Fig. 11. Resulting normalized channel utilization for random flows with periods between 500 ms and 1000 ms.

The figures show three main features. First, we note that a significant increase in the achievable channel utilization is possible by using flow aggregation. The increase is between 40% to 70% with values of L^{max} between 80 bytes and 600 bytes. Second, we observe that the cutoff value of L^{max} after which the benefits in terms of utilization become less substantial is the same as in the previous experiment. This supports the idea that a limited number of BAG slots translated directly into less opportunities for aggregation and thus less improvements altogether. Finally, by comparing the two figures, we note that the achieved utilization benefits do not depend heavily on how similar are the characteristics of the flows. In fact, the same trend emerges even if the possible range of flow periods varies consistently across the two experiments.

C. AFDX Switch Backlog

Next, we study the trend in the amount of backlog at the AFDX link under two different situations. In this case, like in our first experiment, we randomly generate flows until saturation of a traditional AFDX network. Then, we study the backlog when aggregation is not performed compared to the case where the proposed technique is in use. The status of the queue is depicted across the 20 seconds of simulation with a 0.1 second interval between each measurement. The maximum number of queued packets in each 0.1 second interval is considered.

Figure 12 depicts the trend for the queue size with values of L^{max} equal 80 and 300 bytes. A sharp improvement in the amount of packets queued at the AFDX switch is obtained for smaller values of L^{max} throughout the simulation time. The gain in terms of backlog becomes less substantial but still significant with a value $L^{max} = 300$ bytes. The system exhibits more comparable performances with high values of L^{max}, as depicted in Figure 13, even though the configuration featuring aggregated flows still achieves better performances. An interesting feature can be noted on both figures: when no aggregation is performed, at time 0 a peak in the number of queued packets is visible, which periodically reappears during the simulated time. This is not surprising since periods are in phase and the transmission of packets may occur toward the beginning of each period. Conversely, the phase-shifting necessary for flow aggregation limits the amount of packets simultaneously reaching the AFDX switch. Thus, effectively, packets are retained at the buffers introduced between sources and AFDX interconnect, where memory is less expensive and can be distributed among network nodes.

X. Conclusions and Future Work

The steady growth in the complexity of avionic networks requires high-bandwidth solutions to be employed. On the other hand, the correct behavior of safety-critical software components communicating over the network represents a key design-time aspect. As a result, reliability and packet delivery guarantees constitute a strict requirement for an avionic network. In the context of avionic networks, AFDX represents a trade-off between performance and necessary determinism.

In our work, we first consider specification-imposed constraints on the configuration of an AFDX network. Our analysis about their impact on the number of configurable network flows and achievable bandwidth reveals an intrinsic resource underutilization. With this in mind, we propose TPS: a technique that allows the aggregation of network flows through traffic shifting and buffering at the sources. We demonstrate that TPS leads to better channel utilization, allows for an increased

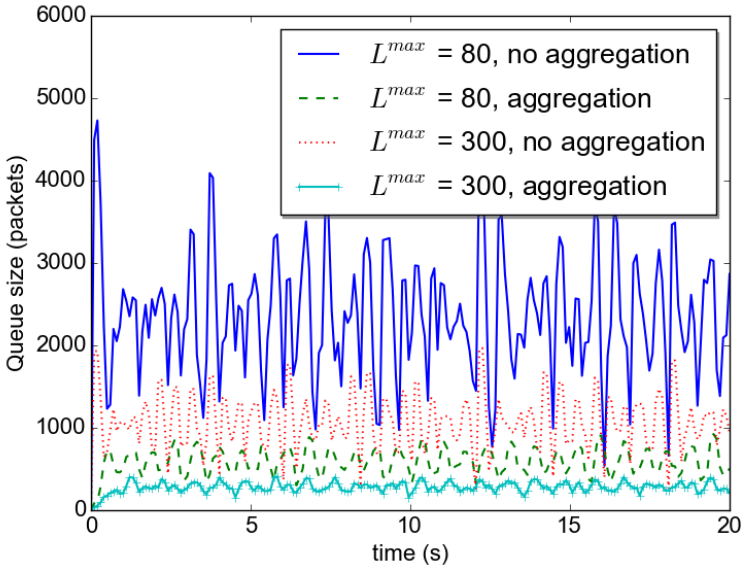

Fig. 12. Queue length in packets with $L^{max} = 80, 300$ bytes.

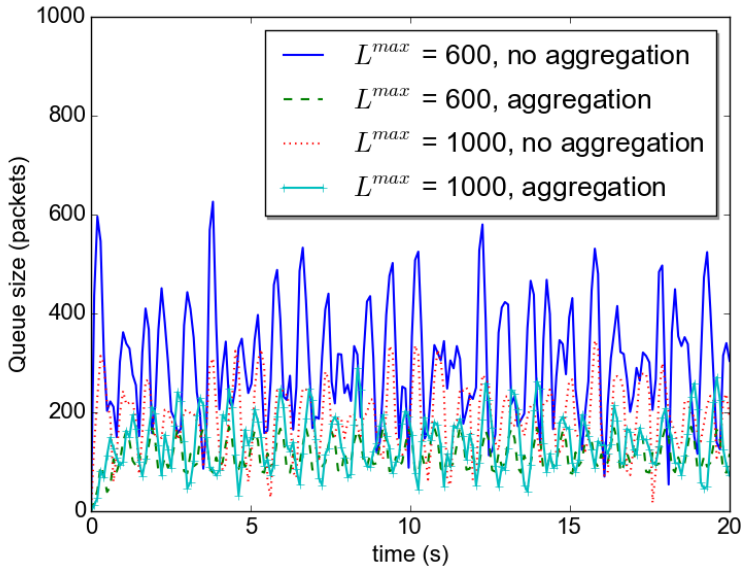

Fig. 13. Queue length in packets with $L^{max} = 600, 1000$ bytes.

number of nodes to communicate over the network, and reduces the queue size at the AFDX switches. At the same time, we show that delivery guarantees are always met, making TPS suitable for hard real-time operations.

As a part of our future work, we plan to extend TPS to further increase achievable bandwidth through packet-level aggregation. Moreover, we intend to investigate whether a similar technique can be used to aggregate traffic flows with non-harmonic rates. Finally our plans include the evaluation of TPS using real AFDX switches and PALS for time-synchronization of traffic sources [16].

ACKNOWLEDGMENT

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS-1219064, and CNS-1302563. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Airlines Electronic Engineering Committee. *Aeronautical Radio Inc. ARINC specification 664P7: aircraft data network, part 7 – avionics Full Duplex Switched Ethernet (AFDX) network*. June 2005.

[2] Airlines Electronic Engineering Committee. *Aeronautical Radio Inc. ARINC specification 429: Digital Information Transfer System (DITS)*. December 2004.

[3] *MIL-STD-1553B: Aircraft Internal Time Division Command/Response Multiplex Data Bus*. September 1978.

[4] H. Charara, J. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an AFDX network. In *Real-Time Systems, 2006. 18th Euromicro Conference on*, pages 10 pp.–202, 2006.

[5] F. Frances, C. Fraboul, and J. Grieu. Using network calculus to optimize the AFDX network. In *ERTS 2006 : 3rd European Congress ERTS Embedded real-time software*, pages pp. 1–8, 2006.

[6] M. Li, M. Lauer, G. Zhu, and Y. Savaria. Determinism enhancement of AFDX networks via frame insertion and sub-virtual link aggregation. *Industrial Informatics, IEEE Transactions on*, 10(3):1684–1695, Aug 2014.

[7] A. Al Sheikh, O. Brun, M. Chéramy, and P. E. Hladik. Optimal design of virtual links in AFDX networks. *Real-Time Systems*, 49(3):308–336, 2013.

[8] H. Ayed, A. Mifdaoui, and C. Fraboul. Hierarchical traffic shaping and frame packing to reduce bandwidth utilization in the AFDX. In *Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on*, pages 77–86, June 2014.

[9] H. Ayed, A. Mifdaoui, and C. Fraboul. Interconnection optimization for multi-cluster avionics networks. In *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*, pages 145–154, July 2013.

[10] J. Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, Berlin, Heidelberg, 2001. ISBN 3-540-42184-X.

[11] M. Grenier, L. Havet, and N. Navet. Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost. In *Embedded Real Time Software (ERTS), 4th European Congress on*, Toulouse, France, 2008.

[12] P.M. Yomsi, D. Bertrand, N. Navet, and R.I. Davis. Controller area network (can): Response time analysis with offsets. In *Factory Communication Systems (WFCS), 9th IEEE International Workshop on*, pages 43–52, May 2012.

[13] R.P.G. Collinson. Fly-by-wire flight control. *Computing Control Engineering Journal*, 10(4):141–152, Aug 1999.

[14] GE/Fanuc Embedded Systems Information Centers. AFDX/ARINC 664 protocol tutorial. 2007.

[15] Airlines Electronic Engineering Committee. *Aeronautical Radio Inc. ARINC Report 655: Remote Data Concentrator (RDC) Generic Description*. April 1999.

[16] S.P. Miller, D.D. Cofer, Lui Sha, J. Meseguer, and A. Al-Nayeem. Implementing logical synchrony in integrated modular avionics. In *Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th*, pages 1.A.3–1–1.A.3–12, Oct 2009.

[17] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, Jan 2003.

[18] W. Steiner and J. Rushby. TTA and PALS: Formally verified design patterns for distributed cyber-physical systems. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 7B5–1–7B5–15, Oct 2011.

[19] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 4, pages 101–104 vol.4, 2000.

[20] University of Southern Carolina. The network simulator - NS-2. URL `http://www.isi.edu/nsnam/ns/`.