







Article

An Event-Driven Link-Level Simulator for the Validation of AFDX and Ethernet Avionics Networks

Pablo Vera-Soto ¹, Javier Villegas ¹, Sergio Fortes ^{1,*}, José Pulido ¹, Vicente Escaño ², Rafael Ortiz ²
and Raquel Barco ¹

¹ Telecommunication Research Institute (TELMA), E.T.S. Ingeniería de Telecomunicación, University of Málaga, Boulevard Louis Pasteur 35, 29010 Málaga, Spain; pvera@ic.uma.es (P.V.-S.); jvc@ic.uma.es (J.V.); josepa@ic.uma.es (J.P.); rbm@ic.uma.es (R.B.)

² Aerospace and Defence Systems, Aertec Solutions, 29590 Málaga, Spain; vescano@aertecsolutions.com (V.E.); rortiz@aertecsolutions.com (R.O.)

* Correspondence: sfr@ic.uma.es

Abstract: Aircraft are composed of many electronic systems: sensors, displays, navigation equipment, and communication elements. These elements require a reliable interconnection, which is a major challenge for communication networks since high reliability and predictability requirements must be verified for safe operation. In addition, their verification via hardware deployments is limited because these are costly and it is difficult to try different architectures and configurations, thus delaying design and development in this area. Therefore, verification at early stages in the design process is of great importance and must be supported with simulation. In this context, this work presents an event-driven link-level framework and simulator for the validation of avionics networks. The tool presented supports communication protocols commonly used in avionics, such as Avionics Full-Duplex Switched Ethernet (AFDX), as well as Ethernet, which is used with static routing. Also, the simulator provides accurate results by employing realistic models for various devices. The proposed platform was evaluated in the Clean Sky's Disruptive Cockpit for Large Passenger Aircraft architecture scenario, showing the capabilities of the simulator. Verification speed is a key factor in its application, so the computational cost was analyzed, proving that the execution time is linearly dependent on the number of messages sent and that the increase in the number of nodes has few quadratic components.

Keywords: avionics; ARINC664; AFDX; Ethernet; communications; verification and validation; protocols



Citation: Vera-Soto, P.; Villegas, J.; Fortes, S.; Pulido, J.; Escaño, V.; Ortiz, R.; Barco, R. An Event-Driven Link-Level Simulator for the Validation of AFDX and Ethernet Avionics Networks. *Aerospace* **2024**, *11*, 247. <https://doi.org/10.3390/aerospace11040247>

Academic Editor: Spiros Pantelakis

Received: 31 January 2024

Revised: 16 March 2024

Accepted: 19 March 2024

Published: 22 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The aerospace industry has made significant progress since its inception over a century ago by the Wright brothers. The introduction of *avionics* (a term derived from the combination of *aviation* and *electronics*) has been of great importance in these advances. Avionics encompasses all the electronic systems that have been added to aircraft, including a wide range of equipment such as actuators, sensors, and communication systems. These systems make up the majority of the safety-critical elements in an aircraft. The concept of Integrated Modular Avionics (IMA) and Distributed IMA (DIMA) [1] has been extended to commercial aviation with the design of airplanes such as the Airbus A380 [2] and Boeing 777 [3]. This approach has advanced avionics significantly. The approach distributes safety-critical functions into separate independent modules, placing them closer to the components they monitor and connecting them within an avionics network.

Ethernet-based protocols, such as Avionics Full-Duplex Switch Ethernet (AFDX) [4], are currently the most widely used among the various protocols and buses available for establishing these types of networks. Other protocols and buses, such as the CAN bus and serial bus, are also used, but Ethernet-based protocols are the most prevalent. AFDX is an implementation of the ARINC 664 Part 7 standard that provides dedicated bandwidth and

a fixed Quality of Service (QoS). The authors of this work previously presented an AFDX framework and simulator in [5] to facilitate the validation process during the Software-in-the-loop (SIL) step of the Model-Based Systems Engineering (MBSE) design process.

Although this protocol is widely used in avionics, there are efforts to implement other protocols, such as Ethernet networks with static routing. In this matter, some works are starting to propose Ethernet topologies instead of AFDX for avionics networks. For instance, refs. [6,7] explored new topologies for Ethernet-based avionics networks with a focus on ring topologies. The authors compared the AFDX topology of the Airbus A380 with different versions of an equivalent Ethernet ring topology, some of which achieved better delays. This shift away from AFDX has also been seen in the market. For example, [8] determined that a custom Ethernet implementation is more flexible and suitable for enterprise interests. However, it is noted that Ethernet is not as reliable as AFDX.

Additionally, the Time Sensitive Networking (TSN) [9] standard is another Ethernet-based option that is expected to become the standard for future generations of aircraft. Currently, a working group is developing a TSN profile tailored specifically to the avionics sector, covering aspects such as shapers, scheduling, and stream isolation [10].

In the aerospace market, achieving great reliability is crucial for avionics networks since they must satisfy the Design Assurance Level-A (DAL-A) of the DO-254 [11] and DO-178 [12] documents for certification. However, the industry is also seeking to reduce costs. As mentioned, one way to achieve this is by replacing the AFDX network with a less expensive alternative that requires fewer devices, resulting in less fuel consumption per flight. The flexibility of Ethernet-based networks provides designers with greater freedom to design redundancy strategies. Therefore, the choice to implement redundancy management when working with the Ethernet protocol in avionics topologies becomes optional, as it may or may not be necessary, based on the specific network design and redundancy requirements.

Therefore, it was deemed necessary to update the simulator in order to meet market demands. In this regard, the Ethernet protocol was added as an option to the simulator. Additionally, improvements have been made to the simulator, such as a more realistic memory structure in the switch model and the separation of packet generation and Band Allocation Gap (BAG) scheduling. In addition, the simulator now includes switch capacity as an output to facilitate analysis of use cases. The TSN standard has also been studied for a possible future update of the simulator. Finally, a validation of the simulator was carried out.

Therefore, the present work is structured as follows. Section 2 provides a brief summary of the supported protocols. Secondly, Section 3 introduces the developed simulator with an insight into the implementation. Thirdly, Section 4 analyzes the correctness of the simulator results and the computational performance. Then, Section 5 presents a discussion of the integration of the simulator in the design process. Finally, Section 6 presents the main conclusions of this work.

2. Avionic Protocols

This section analyzes the two most important protocols used in avionics networks nowadays; namely ARINC 664 or AFDX, and TSN.

2.1. ARINC 664

AFDX is a packet-switching protocol layered over Ethernet networks that provides deterministic timing and redundancy management. It uses Internet Protocol (IP) and User Datagram Protocol (UDP) as its upper-layer protocols. AFDX provides determinism to the network with static virtual paths called Virtual Links (VL), limited bandwidth through the so-called BAG, and duplicity in the network for redundancy. AFDX networks consist of two types of devices: End Systems (ES), which are the end points of the network, and switches for interconnecting the ES. Further insight into this protocol can be found in the authors' previous work [5].

2.2. Time-Sensitive Networking

TSN is a set of standards of IEEE 802.1 [13] based on Ethernet to provide communications with real-time requirements. It includes several profiles, including Audio Video Bridging (802.1 BA [14]), Fronthaul (802.1 CM/de [15]), Industrial Automation (IEC/IEEE 60802 [16]), and Automotive in-Vehicle (P802.1 DG [17]). Recently, TSN has emerged as a promising protocol for avionics networks. The IEEE 802.1 Task Group is actively developing a TSN profile specifically tailored for avionics networks (IEEE P802.1 DP [10]), which need specifications slightly different from the other profiles.

In order to create the aerospace profile, the Task Group is adapting the structure of the AFDX protocol with IEEE 802.1 substandards. The IEEE 802.1Q [18] Stream is introduced instead of the AFDX VL, which would be implemented on top of Virtual Local Area Networks (VLANs). The AFDX ES is replaced with the IEEE 802.1Q End Station, and the AFDX Switch is substituted with the IEEE 802.1Q Bridge.

A significant difference between TSN and AFDX is the ease of configuration. TSN networks benefit from simplified configuration using the YANG data models developed by IEEE. Additionally, TSN is expected to be less expensive than AFDX, as it can use cheaper Commercial off-the-shelf (COTS) L2 switches, while AFDX equipment is quite costly.

3. Proposed System

As highlighted in Section 1, the acquisition of metrics to evaluate the performance of avionics networks during their development, validation, and verification processes is essential. The main objective is to ensure that the specified delay thresholds critical to the proper operation of the aircraft are consistently met. The logical behavior of the simulator is explained in detail in [5]. This section focuses on the implementation of the simulator in Matlab/Simulink and its enhancements.

3.1. General Framework

On the one hand, The simulator creates a simulation model by taking a series of inputs. These inputs, which are summarized in Table 1, include the simulation time, the Bit Error Rate (BER), and the topology of the network. The topology includes the choice of protocol (Ethernet or AFDX), the connections between network elements (via adjacency matrix), the routing of each flow/VL (manually or randomly configured), periodicity/BAG, frame length, and certain switch parameters such as the switching delay and internal memory.

Table 1. Input configuration parameters.

Parameter	Fields
Simulation time	Duration in seconds
BER	Bit error rate
Topology	Protocol Identifier ESs Route A Route B Cable length (m) Link speed (bps) BAG/periodicity (ms) Min/max packet length (B) Switch characteristics (delay and memory)

On the other hand, the simulator outputs the following Figures of Merit (FoM), which are valuable for network validation and for integrating the simulator into Validation & Verification (V&V) frameworks:

- **Delay.** Includes the maximum, minimum, mean, and standard deviation values of each flow/VL in milliseconds. The delay is set as the time from departure to arrival.
- **Jitter.** Includes the maximum, minimum, mean, and standard deviation values of each flow/VL in milliseconds.
- **Throughput.** Includes the maximum, minimum, mean, and standard deviation values of each flow/VL in bits per second (bps).
- **Packet loss.** Specified for each flow/VL as a percentage.
- **Switch capacity.** General capacity of each switch through the simulation in percentage.

The network model generation process utilizes these inputs to construct the network model, which includes all specified ES and switches. In addition, the model links each VL to its respective ES and establishes all necessary connections between ES and switches.

The simulator was developed using Matlab/Simulink, and it performs event-driven simulations by modeling the packets as entities. This approach prevents timestamp errors and unnecessary computation when there are no events [19]. Also, timestamps are taken from the simulation time, so the lack of synchronization in the network is not taken into account.

Additionally, the simulator manages packet entities in the ES and switches generated models to simulate the transmission of packets in the Data Link Layer. Thus, the simulator's models for these network elements play a crucial role, as explained in the following subsections.

To sum up, the simulator operates according to the scheme depicted in Figure 1. It begins by obtaining the inputs from the configuration files. These inputs are then used to create the routing configuration, which is saved for future simulation replication. Thirdly, it utilizes the routing configuration to generate the Simulink model, link the ESs and switches, and set the parameters for the ES and the switch's blocks. Lastly, the simulation is executed, and after the simulation finishes, the simulator extracts the FoMs from the results and logs them.

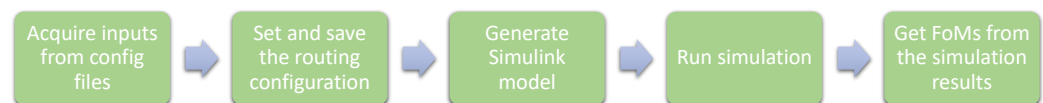


Figure 1. Simulator logical process.

3.2. Multiple VL/Path Configuration

In large real-world network topologies, it is impractical to expect that every data flow will have the same configuration. Data flows for different purposes will have different periodicity and packet length configurations. For this reason, the simulator can automatically generate different configurations for the various VL/paths to understand the normal behavior of an avionics network. This configuration generation is implemented using Orthogonal Arrays (OA), as described in [20].

OAs are mathematical tools utilized for designing an optimal combination of multiple variables in a set of experiments. The input variables, also known as factors, have a discrete set of possible values, or levels. These levels are combined to create an array of representative combinations. These combinations are then used as configurations for the different data flows.

3.3. End System Model

The ES consists of two components: a receiver and a transmitter. When emulating on-board equipment, the frames are generated by the transmitter, which includes a packet generator, a redundancy management module, and a route selection module. Figure 2a shows the Simulink model of the transmitter, which consists of three main components modeled by code: the *Packet Generator*, the *Input Selector*, and the *Route Selector*. Meanwhile, the Simulink model of the receiver is displayed in Figure 2b, where the input streams are combined into the output and stored with a timestamp for later analysis.

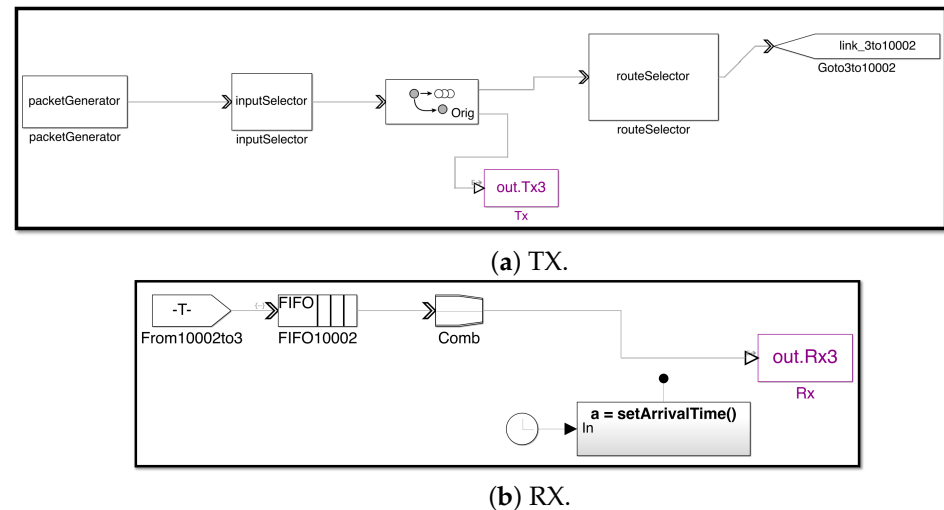


Figure 2. Simulink models within the ES: (a) TX module and (b) RX module.

The *Packet Generator* module replaces the IMA device within the network and generates the frames. In the previous version, packets were directly generated in their corresponding BAG. In this version, the BAG-based scheduling has been separated from the packet generation process and moved to the *Route Selector* module. This facilitates the testing of various traffic patterns to observe network behavior.

Figure 3a illustrates the behavior of this module. In the initial setup, the module programs the first packet-generation event of each data flow /VL. Subsequently, a packet entity is generated for each VL. These entities store all relevant data, such as the BAG value, the frame data, the VL ID, and the payload size. The packet entity is then sent to the module output, and the next packet-generation event for the corresponding VL is programmed based on the traffic pattern. In case of needing redundancy, two packets with the same sequence number are generated.

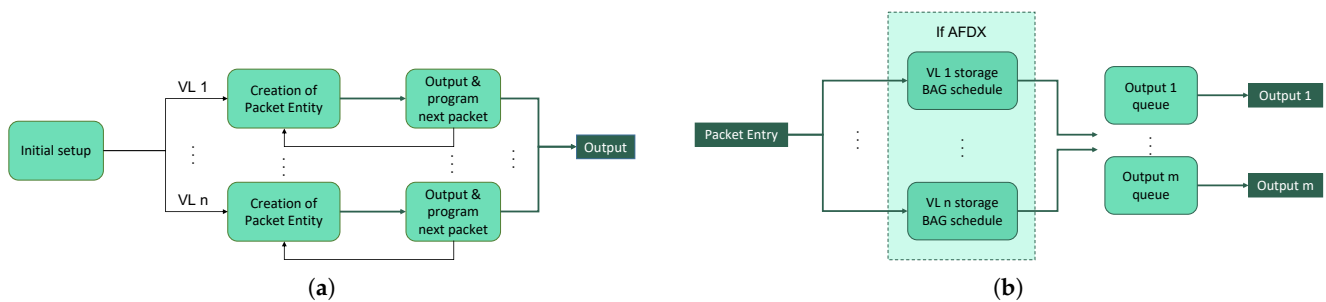


Figure 3. Internal logic and implementation of ES modules: (a) Packet Generator and (b) Route Selector.

The *Input Selector* module retrieves packets from the available inputs, which, in this case, is only the *Packet Generator*, and it forwards them to the *Route Selector* via a single link.

The *Route Selector* is responsible for addressing the packet entities of the output connected to the corresponding switch. The module's behavior is illustrated in Figure 3b. If the AFDX protocol is used, the initial packet is directed to the storage corresponding to its VL, and it is held until the next available BAG. Subsequently, it will be transmitted to the corresponding output queue, where it will wait to be dispatched for the calculated transmission delay. Additionally, the Cyclic Redundancy Check (CRC) is set by the *Route Selector* module based on the BER input, modeling transmission errors.

3.4. Switch Model

As described in [5], the switch's operation revolves around two main processes, scheduling and filtering, which implement the Round Robin and Token Bucket algorithms,

respectively. Figure 4 shows the Simulink model of the switch, which consists of two main modules: an *Input Selector Switch* and *Route Selector Switch*.

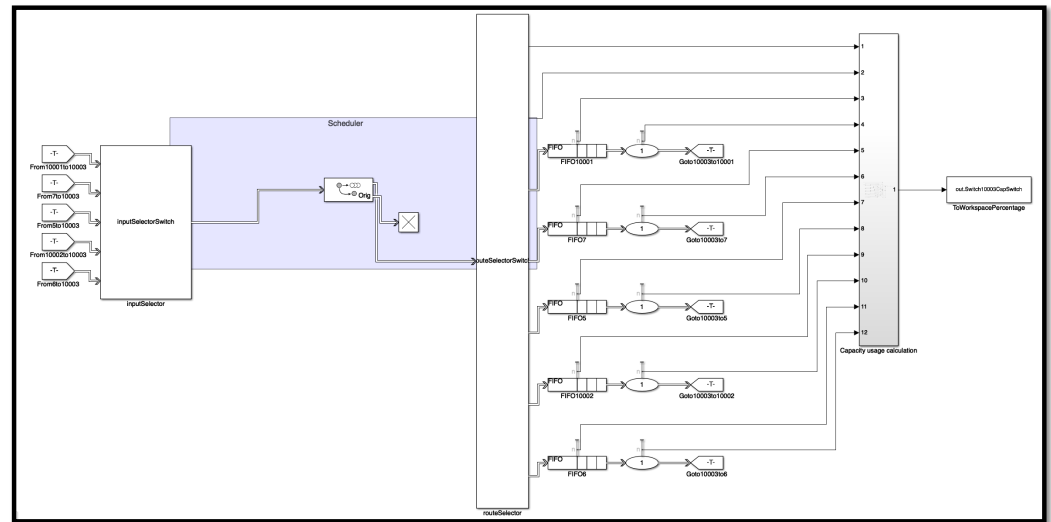


Figure 4. Simulink model for the switch.

The internal memory configuration uses a shared queue system, which is a common approach found in commercial switches, such as the one described in [21]. As shown in Figure 5, this system comprises individual First In First Out (FIFO) queues assigned to each port, providing dedicated memory space. In addition to this, there is a shared memory that is used when a particular FIFO reaches full capacity, ensuring that it does not compromise the reserved memory of other ports. This design protects each port from the saturation effects of burst traffic from other ports. This behavior is illustrated in Figure 5, showing two case scenarios: one without the shared memory full and the other with the shared memory full. In Figure 5a, the green flow fills its FIFO queue while the red flow, characterized as burst traffic, utilizes the shared queue once its dedicated FIFO queue reaches full capacity. In Figure 5b, despite the switch memory being saturated with burst traffic from two flows (blue and red), the green traffic retains its dedicated memory and can transmit without issue, whereas messages from the red and blue traffic are dropped.

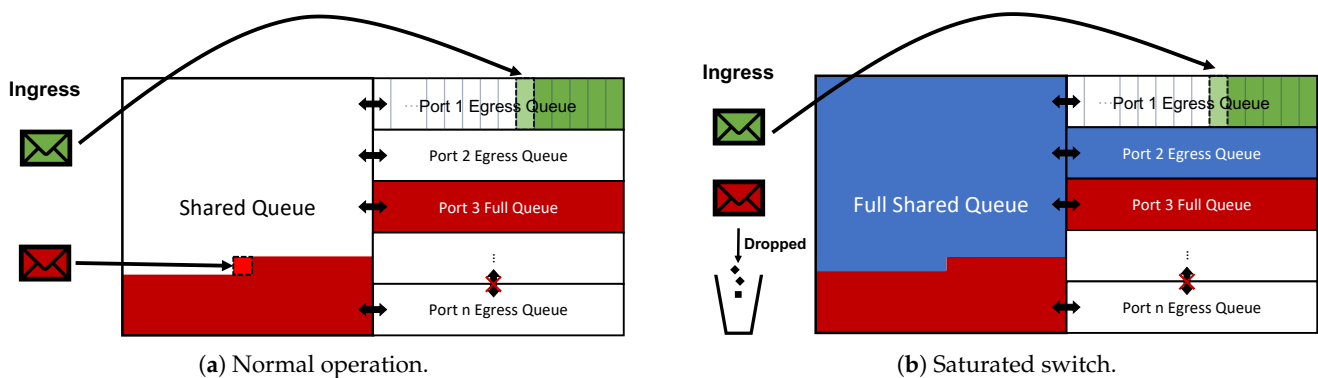


Figure 5. Architecture of the switch queues during (a) normal operation and (b) switch saturation.

Figure 6a illustrates the behavior of the *Input Selector Switch* module. When a packet entity enters the module, it is placed in the appropriate input queue if the CRC is correct. Otherwise, it is dropped. Then, the packets are transmitted from the module using the Round Robin algorithm. This algorithm sequentially processes the queues, transmitting one packet per queue before proceeding to the next. This approach allows messages to bypass queues congested with burst traffic, enabling packets to be moved to the next module without an excessive delay.

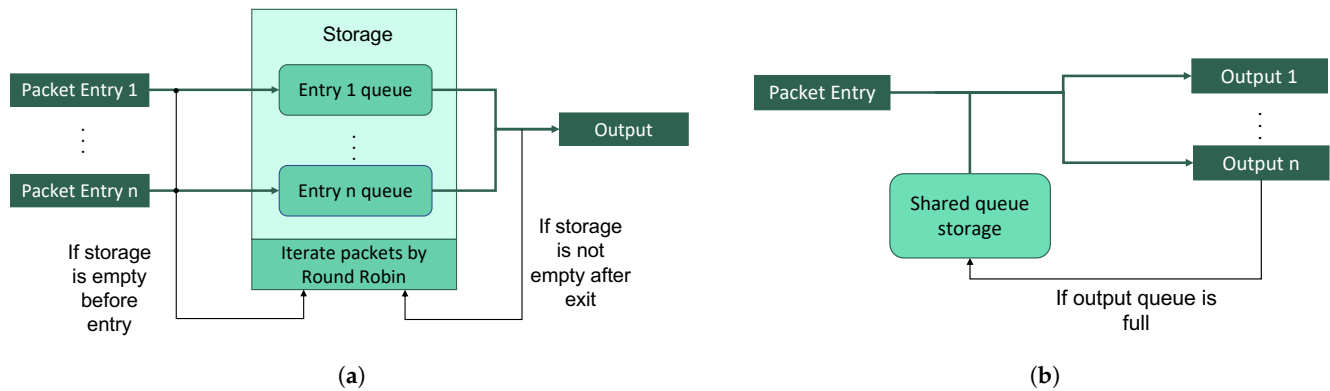


Figure 6. Switch modules' internal logic and implementation: (a) Input Selector Switch and (b) Route Selector Switch.

Figure 6b shows the behavior of the *Route Selector Switch*. When a packet entity enters the module, it is directed to the corresponding output after a switching delay. In the Simulink model, this output is connected to a FIFO queue. If the queue is full, the packet is redirected to a shared memory queue, provided that memory is available. The packet then waits until a gap in the output FIFO queue is available. If the protocol used is AFDX, the module checks whether there is enough credit available to send the packet. If there is not enough credit, the packet is dropped. Following the FIFO queue shown in Figure 4 is an Entity Server block. The packet remains in this block for the duration of the calculated transmission delay before advancing to the next device in the network. This block has been configured to hold only one packet entity at a time. Once the current packet leaves, the next one in the FIFO queue replaces it.

A local Simulink library was created to enable the easy reuse of models or blocks for building new use cases. The simulator was designed to automatically generate the Simulink model using simple configuration files and set block parameters accordingly.

4. Evaluation

Regarding the evaluation of the simulator, two main areas were analyzed in this work. First, the accuracy of the simulation results was verified to ensure that the simulator can be trusted. Secondly, the computational performance of the simulator was analyzed to determine its usefulness and provide an example of its results.

4.1. Correctness of the Results

In order to check the correctness of the simulator results, a comparison with the work presented in [22] was made. This work presents the use of an analytical method derived from Network Calculus to determine the worst possible delays in an AFDX network. Then, the method was validated using a simple use case. The results of this use case were replicated with the simulator presented in this work in order to ensure that the simulator is capable of providing reliable results.

The topology consists of seven ES, three switches, and five VLs. The configuration of the VLs in this use case is depicted in Figure 7 and described in Table 2, where it can be observed that two VLs go through switch S1, another two VLs go through switch S2, and all five VLs go through switch S3. Also, the length of the packets sent is 500 Bytes and the BAG configured is 4 ms.

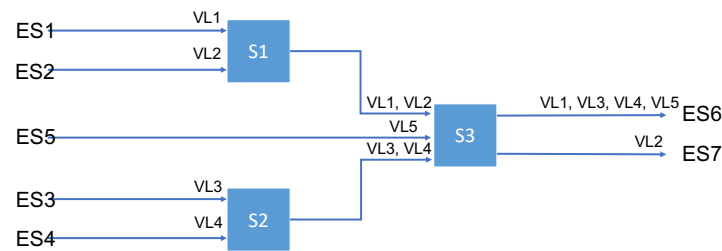


Figure 7. Use case topology for the validation of the simulator, adapted from [22].

Table 2. Use case configuration for the validation of the simulator [22].

Transmitter	VL	Receiver	Path	Packet Length	BAG
ES1	VL1	ES6	ES1 → S1 → S3 → ES6	500 B	4 ms
ES2	VL2	ES7	ES2 → S1 → S3 → ES7	500 B	4 ms
ES3	VL3	ES6	ES3 → S2 → S3 → ES6	500 B	4 ms
ES4	VL4	ES6	ES4 → S2 → S3 → ES6	500 B	4 ms
ES5	VL5	ES6	ES5 → S3 → ES6	500 B	4 ms

Table 3 summarizes the results of this use case. Each row represents the worst-case delay of each VL and the necessary transmission start time of each ES transmission to obtain it, where Δt is an insignificant delta of the time used to establish the packet order in the queues. For instance, in the worst case of VL1, the defined transmission start times cause the packet of VL1 (which departs from ES1) to be processed inside Switch 1 in the second place after the packet of VL2 (which departs from ES2). In Switch 3, packets from VL1, 4, and 5 (departing from ES1, 4, and 5, respectively) arrive simultaneously. The VL1 packet is the last to be processed before reaching the destination ES. The transmission start time for each VL shown in the table was configured in the traffic pattern section of the Packet Generator block. The simulation results match those given in [22], as can be seen in the three columns on the right.

Table 3. Use case results comparison for the validation of the simulator.

VL	Transmission Start					Evaluation Method		
	ES1	ES2	ES3	ES4	ES5	EPL	BNCOG	Simulation
VL1	$2\Delta t \text{ } \mu\text{s}$	$\Delta t \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$96 \text{ } \mu\text{s}$	$272 \text{ } \mu\text{s}$	$272.8 \text{ } \mu\text{s}$	$272 \text{ } \mu\text{s}$
VL2	$0 \text{ } \mu\text{s}$	$\Delta t \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$96 \text{ } \mu\text{s}$	$192 \text{ } \mu\text{s}$	$192 \text{ } \mu\text{s}$	$192 \text{ } \mu\text{s}$
VL3	$\Delta t \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$2\Delta t \text{ } \mu\text{s}$	$\Delta t \text{ } \mu\text{s}$	$96 \text{ } \mu\text{s}$	$272 \text{ } \mu\text{s}$	$272.8 \text{ } \mu\text{s}$	$272 \text{ } \mu\text{s}$
VL4	$\Delta t \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$\Delta t \text{ } \mu\text{s}$	$2\Delta t \text{ } \mu\text{s}$	$96 \text{ } \mu\text{s}$	$272 \text{ } \mu\text{s}$	$272.8 \text{ } \mu\text{s}$	$272 \text{ } \mu\text{s}$
VL5	$\Delta t \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$0 \text{ } \mu\text{s}$	$96 + 2\Delta t \text{ } \mu\text{s}$	$176 \text{ } \mu\text{s}$	$176.8 \text{ } \mu\text{s}$	$176 \text{ } \mu\text{s}$

The simulator offers the possibility to analyze the causes of various delays in detail by examining the FoM of switch capacity. This can be done by monitoring the memory usage of the switch throughout the simulation. For example, the usage of memory of the three switches during the collision of packets is represented in Figure 8. In particular, these data correspond with the simulation of the worst case for VL1 presented in Table 3. In both Switch 1 and Switch 2 (Figure 8a and Figure 8b, respectively), it can be observed that two packets arrive at the same time and leave one by one. Meanwhile, in Switch 3 (Figure 8c), more packets arrive at the same time: at time $t = 112 \text{ } \mu\text{s}$, the packets of VL2 and VL3 arrive at the switch and go to different queues; at time $t = 152 \text{ } \mu\text{s}$, both packets leave the switch, and three packets from VL4, VL5, and VL1 arrive at the same queue (being processed in that order). At time $t = 192 \text{ } \mu\text{s}$, the first packet in the queue leaves, at time $t = 232 \text{ } \mu\text{s}$, the second packet leaves, and, at time $t = 272 \text{ } \mu\text{s}$, the packet corresponding to VL1 leaves and reaches its destination, as shown in Table 3. Also, in Figure 8d, the time axis is zoomed out to show the BAG of 4 ms.

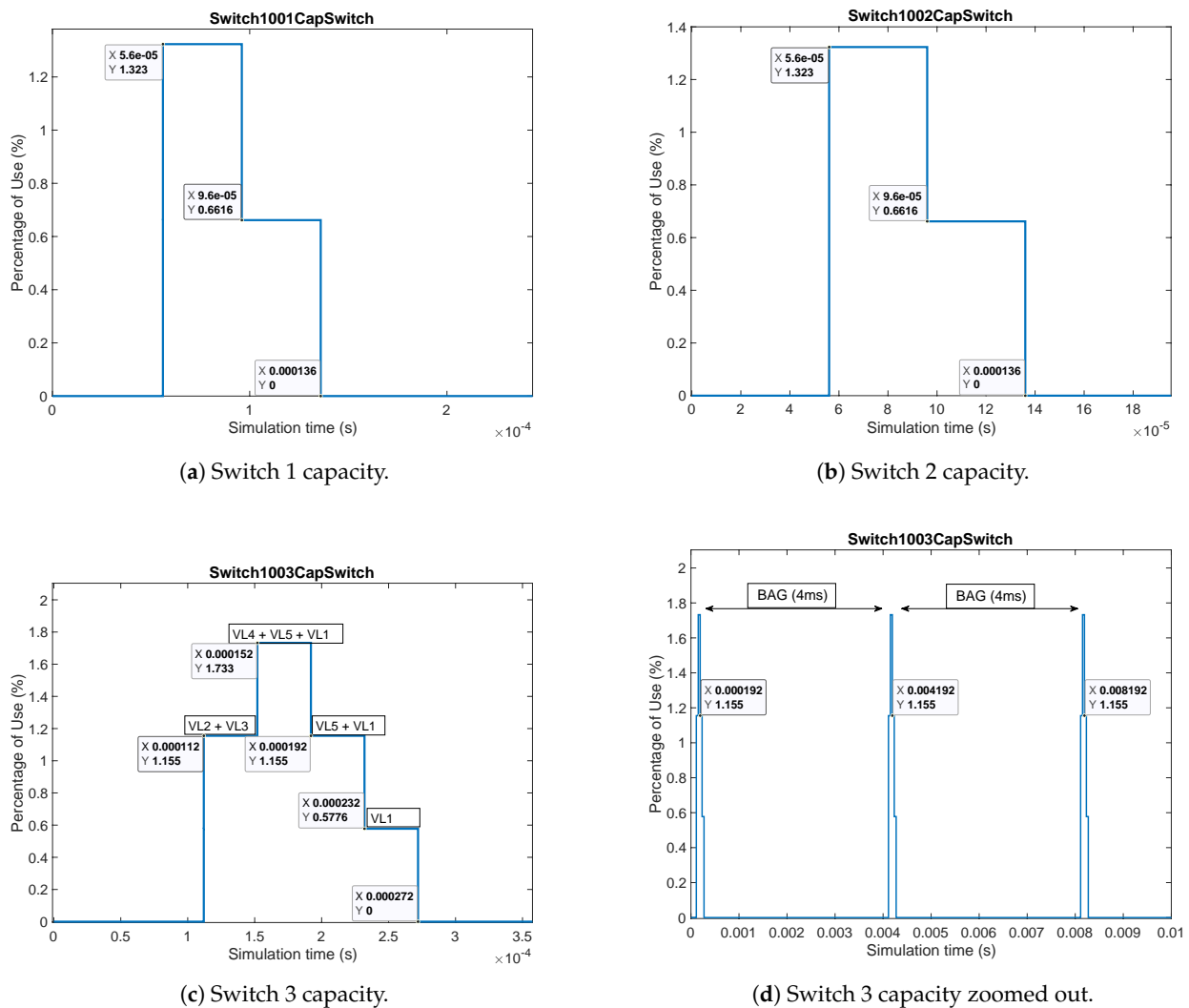


Figure 8. Switch memory usage during packet collisions.

4.2. Computational Performance Analysis

In order to analyze the computational performance, the execution time of the simulations was studied. For this, the network topology of a real airplane, specifically the Airbus A350, was simulated with different packet periodicity configurations. The Airbus A350 flight control architecture, which was adapted from [23], is depicted in Figure 9. This architecture is composed of 37 ESs, of which 6 are Calculator Unit (CU)s and 7 are switches. The switches L2, L1, C, R1, and R2 are connected to six or seven ESs each. The computation and data processing are carried out via the CUs, while the remaining ESs work as sensors or actuators. As a result, communication flows between the CUs and the other ESs in both directions.

Then, this topology was simulated for 1 s with 60 VL configured with a packet periodicity of 0.5 ms, 1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 6 ms, 7 ms, and 8 ms, meaning a total of 120,000, 60,000, 30,000, 20,000, 15,000, 12,000, 10,000, 8570, and 7500 messages sent, respectively. The rest of the configuration parameters are summarized in Table 4. Each configuration was simulated 100 times in order to obtain statistically significant results. The mean execution time of these configurations (running on a Mac with an Apple M1 chip and 16 GB of RAM) resulted in 13.5, 6.4, 3.5, 2.6, 2.2, 1.8, 1.6, 1.5, and 1.4 min, respectively, as shown in Figure 10. These results show that the duration of the simulations has a clear linear dependence on the messages sent, resulting in the linear expression of Equation (1)

with a correlation value of $R^2 = 0.97869$, where EX_time refers to the execution time of the simulation in minutes, and $N_Packets$ refers to the number of packets sent during the simulation.

$$Ex_time[min] = 0.000106 \cdot N_Packets + 0.5 \quad (1)$$

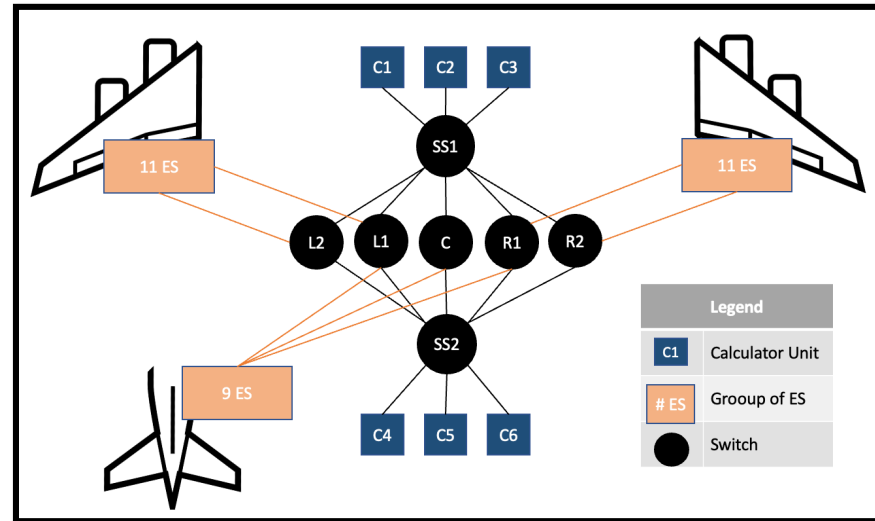


Figure 9. Airbus A350 architecture used for the performance analysis, adapted from [23].

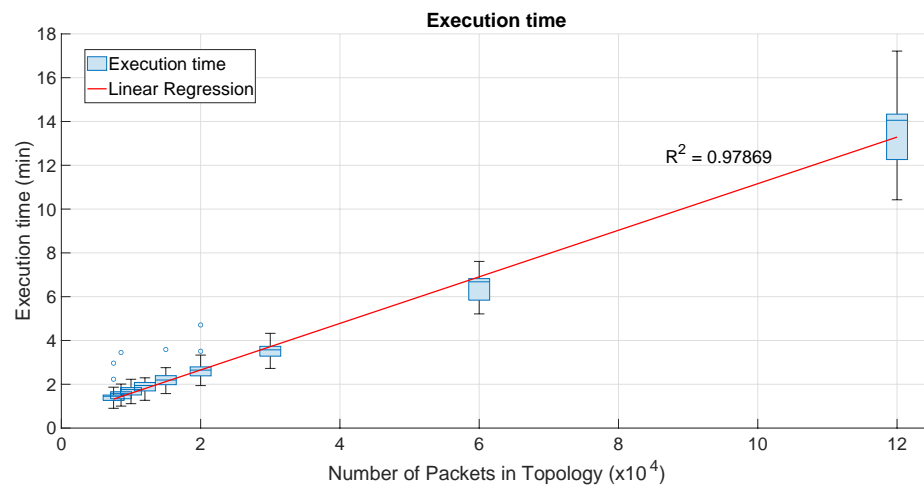


Figure 10. Computational performance of the simulator: execution time in minutes vs. number of messages sent.

Additionally, a comparison of execution times for different topologies was conducted to observe the impact of the number of nodes (ES and switches) in the network. Three topologies were used: the 10-node topology from Section 4.1 (7 ES and 3 switches), the Airbus A350 topology with 44 nodes (37 ES and 7 switches), and the Airbus A380 topology with 132 nodes (123 ES and 9 switches), which is a typical example of an AFDX topology. Each of these topologies was simulated five times using the input parameters summarized in Table 5.

Table 4. Input configuration parameters for the computational analysis.

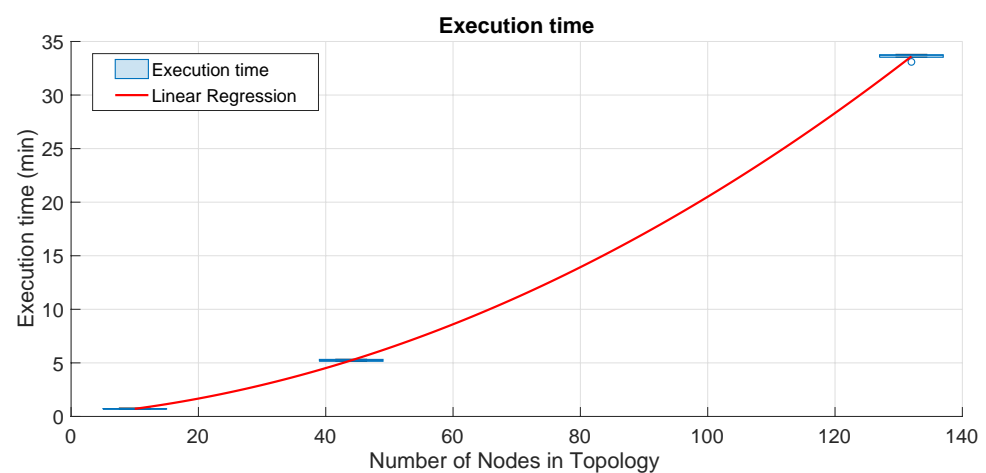
Parameter	Fields
Simulation time	1 s
Protocol	Ethernet
Link speed	1 Gbps
Packet length	1280 B
Periodicity	[0.5, 1, 2, 3, 4, 5, 6, 7, 8] ms
Topology	A350
#VLs	60

Table 5. Input configuration parameters for the topology comparison.

Parameter	Fields
Simulation time	0.5 s
Protocol	AFDX
Link speed	1 Gbps
Packet length	1280 B
BAG	1 ms
#VLs	3 per ES

Figure 11 shows the execution times of these topologies. The mean execution time for each topology is 42.63 s, 5.23 min, and 33.6 min, respectively. The execution time increases polynomially, as described by Equation (2), where EX_time represents the simulation execution time in minutes, and N_Nodes represents the number of nodes in the topology with the configuration of Table 5. However, the linear component is approximately 30 times greater than the quadratic component in the quadratic expression. This, along with the A380 topology being one of the largest available, suggests that the quadratic term would have little impact on the execution time of real networks, ensuring prompt evaluation.

$$Ex_time[min] = 0.001552 \cdot N_Nodes^2 + 0.049187 \cdot N_Nodes + 0.063406 \quad (2)$$

**Figure 11.** Computational performance of the simulator: execution time in minutes vs. number of nodes in the topology.

4.3. Results Comparison

Furthermore, the simulation-derived packet traces, such as the ones presented in Table 6, can serve as a valuable data source for generating time series metrics. This feedback is crucial in the design process, offering insights into network performance during normal operation. It allows the efficiency of the network to be evaluated and metrics other than worst-case delays, which are typically used for certification purposes, to be obtained.

Table 6. Traces at the reception of an ES.

	Timestamp (s)	Delay (s)	Arrival Time (s)	Depart Time (s)	Payload Size (B)	Tx Address	Rx Address	VL	Seq ($\times 10^{14}$)
1	6.3344×10^{-5}	6.3344×10^{-5}	0	6.3344×10^{-5}	1280	7	1	1	9.1667
2	7.3896×10^{-5}	6.3344×10^{-5}	1.0552×10^{-5}	7.3896×10^{-5}	1280	7	1	2	10.768
3	9.5001×10^{-5}	6.3344×10^{-5}	3.1657×10^{-5}	9.5001×10^{-5}	1280	7	1	15	15.317
4	1.1611×10^{-4}	6.3344×10^{-5}	5.2762×10^{-5}	1.1611×10^{-4}	1280	7	1	32	40.854
5	1.0633×10^{-3}	6.3344×10^{-5}	1.0000×10^{-3}	1.0633×10^{-3}	1280	7	1	1	2.9073
6	1.0738×10^{-3}	6.3344×10^{-5}	1.0205×10^{-3}	1.0738×10^{-3}	1280	7	1	2	4.0522
7	1.0950×10^{-3}	6.3344×10^{-5}	1.0316×10^{-3}	1.0950×10^{-3}	1280	7	1	15	35.144
8	1.1161×10^{-3}	6.3344×10^{-5}	1.0527×10^{-3}	1.1161×10^{-3}	1280	7	1	32	36.255

The simulation's packet entities store the payload contents. However, the payload in the simulations conducted for this work was randomly generated, so it was not included in the traces in Table 6. Nonetheless, the payload can be included in the traces if the packets contain useful information.

In this way, in order to show some representative statistics, the A350 topology was simulated using an automatically generated configuration of 80 VLs, as commented upon in Section 3.2. This VL configuration is shown in Table 7, where the path, the packet length, and the BAG/periodicity of each VL/data flow are shown. The possible packet length levels were 64, 128, 256, 400, 512, 750, 900, 1100, and 1280 Bytes, while the possible BAG levels were 0.5, 1, 2, 4, and 8 ms. Also, the bitrate was configured as follows: 100 Mbps between the ESs and the switches, and 1 Gbps between different switches.

The A350 topology was simulated twice with this configuration: once with Ethernet and once with AFDX. To compare the protocols, a burst traffic pattern was configured in the *Packet Generator*. The generation time of the next packet entity was modeled as a uniform distribution between the calculated transmission delay of the current packet and the BAG or periodicity value.

Then, the results of the Ethernet simulation are summarized in Table 8, while the results of the AFDX simulation are summarized in Table 9. The differences are significant, mainly due to the application of the BAG-based scheduler. On one hand, the jitter in the AFDX simulation is bounded and never exceeds 500 μ s, which is the maximum jitter allowed in AFDX networks. In most cases, it is close to 0. On the other hand, the Ethernet simulation exhibits higher jitter, with a standard deviation that sometimes exceeds 500 μ s. Additionally, the delay in the Ethernet simulation is generally higher than in the AFDX simulation due to higher network saturation.

However, in the AFDX simulation, packet loss is consistently around 50% in every VL due to the constant waiting for an available BAG. Packet loss is calculated as the number of packets that do not reach their destination at the end of the simulation. In contrast, the Ethernet simulation undergoes a low or close to 0% packet loss in most data flows.

The results of the Ethernet simulation show that, in overly demanding situations (i.e., all data flows with a bursty traffic pattern), the A350 Ethernet topology can result in unacceptable delays and jitter. For this reason, to validate a topology to be used with Ethernet, it must be carefully evaluated within the expected operating conditions.

Table 7. A350 testing use-case-generated configuration.

VL	Transmitter	Receiver	Path	Packet Length	BAG/Periodicity	VL	Transmitter	Receiver	Path	Packet Length	BAG/Periodicity
VL1	ES29	C1	ES29 → R2 → SS1 → C1	1280 B	1 ms	VL41	ES30	C1	ES30 → R2 → SS1 → C1	400 B	1 ms
VL2	C4	ES4	C4 → SS2 → L2 → ES4	400 B	0.5 ms	VL42	ES26	C6	ES26 → R2 → SS2 → C6	128 B	1 ms
VL3	ES17	C6	ES17 → C → SS2 → C6	512 B	8 ms	VL43	ES13	C2	ES13 → C → SS1 → C2	256 B	1 ms
VL4	C1	ES31	C1 → SS1 → R2 → ES31	256 B	2 ms	VL44	ES14	C6	ES14 → C → SS2 → C6	512 B	8 ms
VL5	C3	ES25	C3 → SS1 → R1 → ES25	1280 B	8 ms	VL45	ES9	C1	ES9 → L1 → SS1 → C1	1100 B	0.5 ms
VL6	ES14	C6	ES14 → C → SS2 → C6	900 B	8 ms	VL46	ES27	C4	ES27 → R2 → SS2 → C4	64 B	8 ms
VL7	ES30	C4	ES30 → R2 → SS2 → C4	64 B	0.5 ms	VL47	ES5	C6	ES5 → L2 → SS2 → C6	128 B	0.5 ms
VL8	ES27	C6	ES27 → R2 → SS2 → C6	900 B	1 ms	VL48	ES11	C4	ES11 → L1 → SS2 → C4	900 B	1 ms
VL9	ES24	C5	ES24 → R1 → SS2 → C5	64 B	0.5 ms	VL49	ES3	C2	ES3 → L2 → SS1 → C2	128 B	4 ms
VL10	ES21	C2	ES21 → R1 → SS1 → C2	256 B	2 ms	VL50	ES6	C2	ES6 → L2 → SS1 → C2	512 B	4 ms
VL11	ES1	C2	ES1 → L2 → SS1 → C2	900 B	2 ms	VL51	ES2	C6	ES2 → L2 → SS2 → C6	1100 B	1 ms
VL12	ES4	C5	ES4 → L2 → SS2 → C5	400 B	4 ms	VL52	C3	ES16	C3 → SS1 → C → ES16	1100 B	1 ms
VL13	ES10	C6	ES10 → L1 → SS2 → C6	1100 B	2 ms	VL53	ES28	C3	ES28 → R2 → SS1 → C3	64 B	4 ms
VL14	ES14	C3	ES14 → C → SS1 → C3	128 B	4 ms	VL54	ES25	C3	ES25 → R1 → SS1 → C3	1100 B	2 ms
VL15	ES25	C2	ES25 → R1 → SS1 → C2	900 B	4 ms	VL55	ES13	C1	ES13 → C → SS1 → C1	256 B	8 ms
VL16	ES14	C4	ES14 → C → SS2 → C4	128 B	4 ms	VL56	ES30	C6	ES30 → R2 → SS2 → C6	400 B	2 ms
VL17	ES24	C2	ES24 → R1 → SS1 → C2	1280 B	1 ms	VL57	ES2	C2	ES2 → L2 → SS1 → C2	1100 B	4 ms
VL18	ES21	C1	ES21 → R1 → SS1 → C1	750 B	2 ms	VL58	ES26	C1	ES26 → R2 → SS1 → C1	256 B	4 ms
VL19	ES16	C6	ES16 → C → SS2 → C6	400 B	8 ms	VL59	ES6	C4	ES6 → L2 → SS2 → C4	750 B	8 ms
VL20	ES19	C2	ES19 → R1 → SS1 → C2	512 B	1 ms	VL60	ES21	C3	ES21 → R1 → SS1 → C3	750 B	0.5 ms
VL21	ES8	C4	ES8 → L1 → SS2 → C4	1280 B	0.5 ms	VL61	ES10	C5	ES10 → L1 → SS2 → C5	750 B	8 ms
VL22	ES28	C6	ES28 → R2 → SS2 → C6	900 B	0.5 ms	VL62	ES22	C2	ES22 → R1 → SS1 → C2	900 B	4 ms
VL23	ES5	C1	ES5 → L2 → SS1 → C1	1100 B	2 ms	VL63	ES20	C5	ES20 → R1 → SS2 → C5	750 B	2 ms
VL24	ES27	C2	ES27 → R2 → SS1 → C2	128 B	0.5 ms	VL64	ES29	C5	ES29 → R2 → SS2 → C5	256 B	2 ms
VL25	ES8	C6	ES8 → L1 → SS2 → C6	128 B	8 ms	VL65	ES14	C3	ES14 → C → SS1 → C3	256 B	0.5 ms
VL26	ES7	C2	ES7 → L1 → SS1 → C2	1280 B	2 ms	VL66	ES16	C4	ES16 → C → SS2 → C4	400 B	8 ms
VL27	ES15	C3	ES15 → C → SS1 → C3	64 B	2 ms	VL67	ES25	C4	ES25 → R1 → SS2 → C4	512 B	1 ms
VL28	ES19	C4	ES19 → R1 → SS2 → C4	1280 B	1 ms	VL68	ES26	C4	ES26 → R2 → SS2 → C4	400 B	4 ms
VL29	C2	ES24	C2 → SS1 → R1 → ES24	512 B	8 ms	VL69	ES30	C6	ES30 → R2 → SS2 → C6	750 B	4 ms
VL30	ES12	C4	ES12 → L1 → SS2 → C4	400 B	0.5 ms	VL70	ES20	C4	ES20 → R1 → SS2 → C4	256 B	2 ms
VL31	ES2	C4	ES2 → L2 → SS2 → C4	64 B	1 ms	VL71	ES10	C3	ES10 → L1 → SS1 → C3	400 B	2 ms
VL32	ES29	C1	ES29 → R2 → SS1 → C1	750 B	0.5 ms	VL72	ES27	C2	ES27 → R2 → SS1 → C2	1100 B	4 ms
VL33	ES15	C1	ES15 → C → SS1 → C1	1100 B	8 ms	VL73	ES6	C2	ES6 → L2 → SS1 → C2	64 B	8 ms
VL34	ES6	C5	ES6 → L2 → SS2 → C5	900 B	0.5 ms	VL74	ES10	C6	ES10 → L1 → SS2 → C6	1280 B	4 ms
VL35	ES17	C1	ES17 → C → SS1 → C1	750 B	1 ms	VL75	ES6	C6	ES6 → L2 → SS2 → C6	1280 B	8 ms
VL36	ES9	C4	ES9 → L1 → SS2 → C4	64 B	1 ms	VL76	C3	ES4	C3 → SS1 → L2 → ES4	512 B	2 ms
VL37	ES24	C3	ES24 → R1 → SS1 → C3	750 B	4 ms	VL77	ES13	C4	ES13 → C → SS2 → C4	512 B	0.5 ms
VL38	ES8	C6	ES8 → L1 → SS2 → C6	900 B	8 ms	VL78	ES19	C5	ES19 → R1 → SS2 → C5	128 B	1 ms
VL39	ES26	C4	ES26 → R2 → SS2 → C4	512 B	0.5 ms	VL79	ES4	C2	ES4 → L2 → SS1 → C2	64 B	4 ms
VL40	C1	ES14	C1 → SS1 → C → ES14	1280 B	2 ms	VL80	ES14	C4	ES14 → C → SS2 → C4	256 B	0.5 ms

Table 8. Results of A350 simulation with Ethernet protocol for bursty traffic.

VL	Delay (μs)		Jitter (μs)		Throughput (Kbps)	Packet Loss (%)	VL	Delay (μs)		Jitter (μs)		Throughput (Kbps)	Packet Loss (%)
	Mean	Std	Mean	Std				Mean	Std	Mean	Std		
1	5246.1	683.19	253.21	634.5	1717.1	20.842	41	5159.3	671.83	238.07	628.21	1717.1	6.0666
2	105.57	5.1888	1.9086	4.8249	130.8	0	42	336.27	164.01	136.55	90.807	130.8	0.65524
3	376.63	154.67	128.68	85.426	284.95	1.2245	43	171.04	95.664	73.45	61.27	284.95	0.050429
4	80.376	5.438×10^{-11}	5.4246×10^{-11}	3.4102×10^{-12}	332.75	0	44	379.1	161.01	131.04	93.183	332.75	2.3166
5	252.42	0.14869	0.018622	0.14751	220.58	0.3937	45	5238	651.23	236.15	606.89	220.58	17.863
6	450.96	159.33	132.89	87.466	350.55	1.2448	46	5065	328.19	85.074	316.93	350.55	4.2969
7	5079.3	232.52	79.154	218.63	1774.3	3.5661	47	334.78	163.41	135.8	90.877	1774.3	0.39448
8	434.56	156.75	130.2	87.224	360.87	2.2959	48	5175.4	323.96	94.756	309.78	360.87	43.489
9	66.762	28.359	22.739	16.943	278.82	0	49	156.93	95.261	75.865	57.513	278.82	0
10	175.72	94.467	73.892	58.809	902.75	0.10121	50	219.92	95.22	75.732	57.617	902.75	0
11	273.02	89.147	68.386	57.147	212.54	0.30738	51	461.48	155.32	129.52	85.688	212.54	3.0153
12	125.4	29.706	24.834	16.263	1052.3	0	52	222.24	0.77226	0.15096	0.75736	1052.3	0
13	459.12	150.28	125.6	82.409	78.884	3.4161	53	68.718	30.636	23.826	19.228	78.884	0
14	83.284	37.041	28.161	24.029	484.62	0	54	238.55	25.889	19.565	16.943	484.62	0
15	273.62	93.307	72.391	58.785	73.192	0	55	5116	720.05	276.48	664.62	73.192	4.6512
16	5088.1	274.15	83.657	261.05	2598.3	6.8762	56	372.66	158.25	130.5	89.416	2598.3	1.8127
17	323.19	87.64	67.082	56.378	680.97	0.15068	57	307.16	99.28	73.571	66.583	680.97	0.38462
18	5215.2	635.2	225.04	593.95	101.74	12.475	58	5128.7	688.91	252.31	640.93	101.74	4.2857
19	361.68	162.62	134.49	90.998	1065.7	0.41841	59	5135.5	506.11	136.84	487.13	1065.7	40.249
20	210.25	94.524	72.41	60.737	2415.3	0.05048	60	176.83	27.085	18.578	19.707	2415.3	0.025233
21	5198.5	384.58	115.22	366.91	3529.2	51.888	61	184.73	29.439	24.547	16.176	3529.2	0
22	433.59	157.57	131.8	86.326	873.18	2.3584	62	269.89	88.026	66.646	57.43	873.18	0.1938
23	5226.4	714.1	265.19	662.96	604.05	22.577	63	180.14	25.908	21.136	14.969	604.05	0
24	155.28	97.908	76.265	61.386	35.878	0.02551	64	99.826	29.231	23.326	17.602	35.878	0
25	336.43	171.81	141.43	97.109	1300.2	0.8547	65	100.9	32.568	25.072	20.783	1300.2	0
26	331.4	86.477	66.28	55.505	91.133	0.1004	66	5104.6	381.03	97.424	368.3	91.133	19.672
27	70.835	32.715	25.578	20.381	1128.5	0	67	5123.2	322.39	95.251	307.99	1128.5	25.92
28	5205.2	407.25	117.75	389.83	135.48	56.148	68	5106.9	356.87	101.34	342.14	135.48	20.468
29	123.38	6.0004×10^{11}	5.9264×10^{-11}	8.611×10^{-12}	1347.5	0.39841	69	409.57	156.21	128.67	88.373	1347.5	2.5794
30	5118.1	267.17	84.246	253.54	173.47	19.11	70	5093	335.08	95.365	321.2	173.47	13.636
31	5081.8	233.7	78.755	220.02	2682.9	3.8423	71	126.88	34.304	25.984	22.381	2682.9	0
32	5199	658.93	243.02	612.46	226.51	11.231	72	302.71	93.6	73.386	58.002	226.51	0
33	5234.2	702.61	256.02	654.05	3667.2	19.679	73	150.84	106.29	78.409	71.58	3667.2	0
34	195.6	17.139	10.603	13.464	1367.5	0.025259	74	490.67	153.7	130.98	80.197	1367.5	3.5785
35	5190.8	685.53	254.66	636.44	171.8	12.475	75	513.11	158.11	131.68	87.093	171.8	4.5082
36	5085.7	219.69	75.663	206.24	388.56	3.443	76	125.95	7.3683	4.3839	5.9207	388.56	0.099701
37	185.76	34.521	26.466	22.133	220.48	0	77	5131.2	275.03	85.297	261.46	220.48	23.711
38	447.78	141.19	116.29	79.72	1618.2	3.2653	78	79.191	29.744	23.919	17.672	1618.2	0
39	5127.2	272.66	87.871	258.11	1300.2	24.085	79	136.55	91.567	71.269	57.406	1300.2	0.19342
40	252.5	0.80937	0.18706	0.78743	805.38	0	80	5097	262.88	82.952	249.44	805.38	12.598

Table 9. Results of A350 simulation with AFDX protocol for bursty traffic.

VL	Delay (μs)		Jitter (μs)		Throughput (Kbps)	PacketLoss (%)	VL	Delay (μs)		Jitter (μs)		Throughput (Kbps)	Packet Loss (%)
	Mean	Std	Mean	Std				Mean	Std	Mean	Std		
1	482.37	93.349	85.696	36.918	852.43	49.135	41	104.57	3.525×10^{-11}	2.6026×10^{-11}	2.3759×10^{-11}	852.43	49.9
2	104.57	3.5776×10^{-11}	2.6058×10^{-11}	2.4507×10^{-11}	67.792	49.571	42	67.393	19.032	12.782	14.095	67.792	50.348
3	147.86	3.1272×10^{-11}	2.6501×10^{-11}	1.6431×10^{-11}	141.28	51.923	43	84.785	8.9162	6.8199	5.7394	141.28	50.025
4	80.376	5.7043×10^{-11}	5.4875×10^{-11}	1.5381×10^{-11}	164.57	50.642	44	629.21	1.3022×10^{-10}	7.185×10^{-11}	1.0841×10^{-10}	164.57	50.98
5	252.41	2.139×10^{-11}	1.8294×10^{-11}	1.0962×10^{-11}	116.67	50.98	45	267.02	71.414	61.674	35.976	116.67	48.24
6	307.9	9.3641	2.3405	9.0644	180.09	48.56	46	242.68	6.5281×10^{-11}	5.3866×10^{-11}	3.6561×10^{-11}	180.09	50.98
7	70.213	42.853	25.281	34.596	927.04	49.686	47	258.81	144.7	124.66	73.411	927.04	49.837
8	195.46	18.465	11.904	14.111	180.09	49.975	48	226.48	11.359	7.5097	8.5194	180.09	49.52
9	48.12	5.5794×10^{-11}	4.3391×10^{-11}	3.5061×10^{-11}	141.28	50.199	49	58.872	1.2788×10^{-11}	6.7322×10^{-12}	1.0864×10^{-11}	141.28	48.665
10	95.673	10.969	10.488	3.1775	463.93	48.454	50	463.99	1.0155×10^{-10}	9.8199×10^{-11}	2.513×10^{-11}	463.93	51.55
11	261.31	41.462	41.391	1.547	106.93	49.749	51	423.17	85.785	77.395	36.92	106.93	49.29
12	104.57	3.5217×10^{-11}	2.5894×10^{-11}	2.3812×10^{-11}	564.2	49.698	52	222.17	4.117×10^{-11}	3.4699×10^{-11}	2.2129×10^{-11}	564.2	49.469
13	417.69	52.239	45.195	26.119	38.655	52.153	53	48.12	5.0921×10^{-11}	4.2708×10^{-11}	2.7598×10^{-11}	38.655	49.084
14	60.69	4.9127×10^{-11}	4.0216×10^{-11}	2.81×10^{-11}	232.43	51.644	54	287.17	3.7347×10^{-11}	3.0937×10^{-11}	2.0877×10^{-11}	232.43	49.341
15	377.89	43.828	23.733	36.816	38.656	48.98	55	112.02	4.3846×10^{-11}	3.1502×10^{-11}	3.0365×10^{-11}	38.656	51.362
16	234.36	20.932	20.89	6.4511×10^{-11}	1307.8	51.362	56	285.58	52.239	45.195	26.119	1307.8	50.397
17	394.07	158.52	141.67	70.996	388.79	48.533	57	729.67	3.7325	3.725	1.2011×10^{-10}	388.79	51.172
18	174.96	13.82	11.59	7.5105	53.679	50.1	58	90.987	10.632	10.611	6.4576×10^{-11}	53.679	48.98
19	104.57	3.5487×10^{-11}	2.5998×10^{-11}	2.4042×10^{-11}	538.56	47.699	59	530.47	6.2479×10^{-11}	5.1852×10^{-11}	3.4546×10^{-11}	538.56	50.593
20	137.26	13.949	13.594	3.0973	2613.7	50.372	60	163.37	3.0549×10^{-11}	2.5627×10^{-11}	1.6618×10^{-11}	2613.7	49.393
21	355.56	52.685	45.964	25.73	1853.3	46.921	61	212.58	5.356×10^{-11}	4.4554×10^{-11}	2.9456×10^{-11}	1853.3	49.597
22	260.41	98.178	58.265	79.008	564.15	48.901	62	244.54	44.922	26.724	36.069	564.15	49.9
23	288.1	13.82	11.59	7.5105	308.21	48.98	63	165.13	1.7668	1.765	3.2911×10^{-11}	308.21	50.348
24	120.66	83.379	75.969	34.32	19.405	50.311	64	122.2	28.074	24.256	14.092	19.405	49.85
25	518.02	1.2485×10^{-10}	5.5473×10^{-11}	1.1173×10^{-10}	654.41	50.593	65	128.48	0.017723	0.0085781	0.015508	654.41	50.483
26	416.81	78.787	78.708	8.8648×10^{-11}	45.09	49.9	66	147.77	5.0018×10^{-11}	4.159×10^{-11}	2.7535×10^{-11}	45.09	48.347
27	48.12	6.8471×10^{-11}	4.5613×10^{-11}	5.1025×10^{-11}	1307.5	49.799	67	411.38	56.666	42.811	37.1	1307.5	50.298
28	459.82	38.586	31.124	22.787	67.792	48.901	68	381.96	20.932	20.89	4.4607×10^{-11}	67.792	50
29	123.38	5.9998×10^{-11}	5.9148×10^{-11}	8.552×10^{-12}	852.43	55.986	69	543.22	66.678	66.545	1.0492×10^{-10}	852.43	52.107
30	117.2	37.82	21.845	30.869	90.09	50.087	70	190.88	14.88	12.874	7.4401	90.09	49.393
31	53.387	5.2872	4.6456	2.52	1553.4	49.316	71	273.81	31.196	31.165	4.3855×10^{-11}	1553.4	49.597
32	317.79	139.49	121.39	68.673	141.89	49.058	72	642.91	3.5798	3.5726	1.0965×10^{-10}	141.89	49.698
33	442.32	45.171	44.815	3.975	1853	50.199	73	470	1.0382×10^{-10}	1.0236×10^{-10}	1.4718×10^{-11}	1853	50.397
34	201.29	22.061	19.086	11.057	776.98	48.546	74	594.77	125.61	125.36	9.6547×10^{11}	776.98	47.146
35	334.17	93.349	85.696	36.918	90.09	49.85	75	824.52	1.2254×10^{10}	5.9082×10^{11}	1.0723×10^{-10}	90.09	48.133
36	57.529	5.2167	4.5133	2.6122	194.78	49.52	76	123.38	5.9724×10^{-11}	5.907×10^{-11}	8.4126×10^{-12}	194.78	48.718
37	311.53	4.2774×10^{-11}	3.6576×10^{-11}	2.2055×10^{-11}	116.68	50.593	77	203.22	49.976	44.179	23.342	116.68	49.367
38	685.4	1.2464×10^{-10}	6.4195×10^{-11}	1.0668×10^{-10}	1076.5	48.56	78	88.746	35.875	29.874	19.841	1076.5	49.444
39	160.49	37.82	21.845	30.869	654.31	49.774	79	77.521	2.7128×10^{-11}	2.5423×10^{-11}	9.3295×10^{-12}	654.31	52.471
40	252.41	2.1124×10^{-11}	1.797×10^{-11}	1.1074×10^{-11}	426.43	50.249	80	272.93	109.07	98.943	45.847	426.43	49.52

5. Discussion

As mentioned in this work, the simulator presented in this work was designed to be used in the SIL step of the MBSE design process. This step is particularly important in the aerospace sector because hardware implementation is highly expensive and lacks the flexibility of simulators for network testing. It is, therefore, essential to be sure that the network will function correctly before proceeding to the Hardware-in-the-loop (HIL) stage of the MBSE design process.

Simulators, like the one proposed in this study, play a critical role in this sector. They facilitate the easy reuse of models by storing them in a library and programmatically generating use cases. The utilization of Matlab/Simulink for implementation also offers advantages, including the ability to conduct parallel simulations. The automation of generating use cases from configuration files, as demonstrated in the presented simulator, is a crucial aspect. This functionality enables the seamless integration of the simulator into validation frameworks, such as the one discussed in [5], or automatic validation platforms. These platforms use the FoMs to generate new use cases with minimal human intervention, a current focus of the authors.

6. Conclusions and Outlook

This work is a follow-up to the work presented in [5], which presented an AFDX simulator. Therefore, a brief analysis of the main protocols and standards used to communicate between the different elements of an avionics network, such as Ethernet and Ethernet-based AFDX and TSN, was presented. Efforts to replace AFDX with lower-cost Ethernet-based devices have also been observed in the market. Thus, the AFDX simulator was updated to adapt to the market needs, including the Ethernet protocol, as well as new inputs and outputs, while also improving the models to be more realistic. A deeper examination of the Matlab/Simulink implementation of the simulation was also conducted. In addition, this simulator can be easily integrated into validation frameworks and platforms. Furthermore, the simulator was successfully tested by replicating the results of a known use case, also showing the analysis possibilities that the simulator's FoMs provide. Then, a computational performance analysis was carried out. This analysis showed that the time required for each simulation is linearly dependent on the number of messages sent. Secondly, the comparison of various topologies showed that the time required for each simulation has a slight quadratic correlation with the number of nodes in the topology. This computational complexity allows for the evaluation of real networks in a timely manner. Additionally, the simulator's event-driven design makes it more efficient than other simulators with fixed-step solvers. The versatility of the results also facilitates informed decision-making and the refinement of avionics networks.

Further work will include the study of the TSN standards for their implementation in the simulator, as TSN is anticipated to become the standard for future generations of aircraft. Also, validation frameworks and platforms would be developed in order to automate the avionics design process.

Author Contributions: Conceptualization, P.V.-S., J.V. and S.F.; methodology, P.V.-S. and J.V.; software, P.V.-S., J.V. and J.P.; validation, P.V.-S., J.V. and V.E.; formal analysis, J.V.; resources, S.F., V.E., R.O. and R.B.; writing—original draft preparation, P.V.-S., J.V. and S.F.; writing—review and editing, J.P., V.E., R.O. and R.B.; supervision, S.F.; project administration, S.F. and R.B.; funding acquisition, S.F., V.E., R.O. and R.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Junta de Andalucía and the ERDF (European Regional Development Fund) Operational Programme in the framework of the CAPTOR project: “advanCed Avionics communications validation and verification PlatfORM” (Ref. PYC20 RE 077 UMA), AERTEC Solutions (reference 8.06/5.59.5543, 8.06/5.59.5715, 806/59.5974 and 806/59.5715) in the framework of project 2020 AS-DISCO: “Audio Suite for Disruptive Cockpit Demonstrator” (this project has received funding from the Clean Sky 2 Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under Grant Agreement n°: 865416), the Ministry of Economic Affairs and Digital Transformation and the European Union—NextGenerationEU in the framework of the

Recovery, Transformation and Resilience Plan and the Recovery and Resilience Mechanism under the MAORI project, the Ministry of Science and Innovation (grant FPU21/04472), and the University of Malaga through the “II Plan Propio de Investigación, Transferencia y Divulgación Científica”.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors are grateful to Aertec Solutions for its support and collaboration in this project.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Butz, H. Open Integrated Modular Avionic (IMA): State of the Art and future Development Road Map at Airbus Deutschland. 2008. Available online: [https://www.semanticscholar.org/paper/Open-Integrated-Modular-Avionic-\(-IMA-\)-%3A-State-of-Butz/042d7fc1a86cfd72d8e33f8e2ed93bb7c54a9ffd](https://www.semanticscholar.org/paper/Open-Integrated-Modular-Avionic-(-IMA-)-%3A-State-of-Butz/042d7fc1a86cfd72d8e33f8e2ed93bb7c54a9ffd) (accessed on 5 May 2023).
- Brajou, F.; Ricco, P. The Airbus A380—An AFDX-based flight test computer concept. In Proceedings of the AUTOTESTCON 2004, San Antonio, TX, USA, 20–23 September 2004; pp. 460–463.
- Boeing. Boeing-777. Available online: <https://www.boeing.es/productos-y-servicios/commercial-airplanes/777.page> (accessed on 5 May 2023).
- Kazi, S.I. Architecting of Avionics Full Duplex Ethernet (AFDX) Aerospace Communication Network. 2013. Available online: <https://www.iject.org/vol4/spl4/c0140.pdf> (accessed on 5 May 2023).
- Villegas, J.; Fortes, S.; Escano, V.; Baena, C.; Colomer, B.; Barco, R. Verification and Validation Framework for AFDX Avionics Networks. *IEEE Access* **2022**, *10*, 66743–66756. [CrossRef]
- Mifdaoui, A.; Amari, A. Real-time ethernet solutions supporting ring topology from an avionics perspective: A short survey. In Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017; pp. 1–8. [CrossRef]
- Amari, A.; Mifdaoui, A. Specification and Performance Indicators of AeroRing—A Multiple-Ring Ethernet Network for Avionics Embedded Systems. *Sensors* **2018**, *18*, 3871. [CrossRef] [PubMed]
- Doverfelt, R.; Skolan, K.; Elektroteknik, F.; Datavetenskap, O. An Evaluation of Ethernet as Data Transport System in Avionics 2020. Available online: <https://www.diva-portal.org/smash/get/diva2:1484743/FULLTEXT01.pdf> (accessed on 5 May 2023).
- Deng, L.; Xie, G.; Liu, H.; Han, Y.; Li, R.; Li, K. A Survey of Real-Time Ethernet Modeling and Design Methodologies: From AVB to TSN. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
- IEEE P802.1DP; TSN for Aerospace Onboard Ethernet Communications. IEEE: Piscataway, NJ, USA, 2023. Available online: <https://1.ieee802.org/tsn/802-1dp/> (accessed on 5 May 2023).
- AC 20-152-RTCA; Document RTCA/DO-254, Design Assurance Guidance for Airborne Electronic Hardware—Document Information. Federal Aviation Administration: Washington, DC, USA, 2005.
- DO-178C RTCA; Software Considerations in Airborne Systems and Equipment Certification. Federal Aviation Administration: Washington, DC, USA, 2012.
- IEEE. Welcome to the IEEE 802.1 Working Group. Available online: <https://1.ieee802.org/> (accessed on 5 May 2023).
- IEEE. IEEE SA—IEEE 802.1BA-2021. Available online: <https://standards.ieee.org/ieee/802.1BA/10547/> (accessed on 5 May 2023).
- IEEE. IEEE SA—IEEE 802.1CMde-2020. Available online: <https://standards.ieee.org/ieee/802.1CMde/7478/> (accessed on 5 May 2023).
- IEEE. IEC/IEEE 60802 TSN Profile for Industrial Automation. Available online: <https://1.ieee802.org/tsn/iec-ieee-60802/> (accessed on 5 May 2023).
- IEEE. P802.1DG—TSN Profile for Automotive In-Vehicle Ethernet Communications. Available online: <https://1.ieee802.org/tsn/802-1dg/> (accessed on 5 May 2023).
- IEEE. IEEE SA—IEEE 802.1Q-2018. Available online: <https://standards.ieee.org/ieee/802.1Q/6844/> (accessed on 5 May 2023).
- Mathworks. Solvers for Discrete-Event Systems—MATLAB & Simulink. Available online: <https://es.mathworks.com/help/simevents/ug/solvers-for-simevents-models.html> (accessed on 20 January 2023).
- Altuntaş, M.; Eker, M.; Kışla, P.; Can, E.; Demir, M.; Hokelek, I.; Akdogan, E. Testing Deterministic Avionics Networks Using Orthogonal Arrays. In Proceedings of the The Fifteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Barcelona, Spain, 3–7 October 2021.
- Curtiss-Wright Defense Solutions. Parvus DuraNET 20-10. Available online: <https://www.curtisswrightds.com/products/networking-communications/rugged-systems/duranet-20-10> (accessed on 5 May 2023).

22. Xu, Q.; Yang, X. Performance Analysis on Transmission Estimation for Avionics Real-Time System Using Optimized Network Calculus. *Int. J. Aeronaut. Space Sci.* **2019**, *20*, 506–517. [[CrossRef](#)]
23. Finzi, A.; Mifdaoui, A.; Frances, F.; Lochin, E. Network Calculus-based Timing Analysis of AFDX networks incorporating multiple TSN/BLS traffic classes. *arXiv* **2019**, arXiv:1905.00399.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.