On the integration of DDS and AFDX standards

Héctor Pérez
Software Engineering and Real-Time Group
Universidad de Cantabria
Santander, España
perezh@unican.es

J. Javier Gutiérrez
Software Engineering and Real-Time Group
Universidad de Cantabria
Santander, España
gutierjj@unican.es

***Abstract*—Standard distribution middleware has traditionally been perceived as complex software which is not suitable for satisfying the highest certification criteria in safety-critical environments. However, this idea is slowly changing and there are efforts such as the Future Airborne Capability Environment (FACE) consortium to integrate standard distribution middleware into the development of avionic systems. This integration facilitates the interoperability and portability of avionic applications, but there are still challenges that need to be addressed before full success can be achieved. To this end, this paper explores the usage of the Data Distribution Service for Real-Time Systems (DDS) on top of a partitioned system with a communication network based on the ARINC 664 specification (precisely, the AFDX network). This work specifically identifies the incompatibilities between the two standards and also proposes potential solutions. A set of overhead metrics of using DDS in a distributed partitioned platform is also provided.**

***Index Terms*—distributed systems, distribution middleware, vehicular networks, real-time systems, AFDX, DDS, ARINC 653**

I. Introduction

Over the last years, airborne systems have migrated from federated to integrated modular avionics (IMA) architecture in order to simplify the development of onboard software. To further develop the competitiveness of IMA, this architecture is still evolving and being investigated [1]. A core concept in IMA architectures is partitioning, which enables applications to be strictly isolated from each other in terms of time and space. This isolation allows applications running on a common hardware platform to be certified separately, even if they have been developed by different companies. The partitioning concept is fully supported by the ARINC 653 specification [2], which represents the reference standard in airborne systems. Although partitioning can be implemented in different ways [3], one increasingly important approach is based on hypervisors, which allow multiple operating systems to be run on the same hardware platform.

The adoption of IMA architectures provoked the need for new requirements to be fulfilled [4], such as robust and flexible data communications. ARINC 653 communication ports [2] provide standardized inter-partition communication services in avionics systems, ensuring robust and predictable data exchange between software modules. The Avionics Full Duplex Switched Ethernet (AFDX) enhances these capabilities by offering a deterministic Ethernet-based network, enabling high-speed data transfer and improved integration of complex avionics systems. The AFDX network is defined in the ARINC 664 standard, Part 7 [5], and it relies on the use of standard Ethernet links along with special-purpose switches and network interface cards.

Traditionally, the avionics industry has relied on custom communication software for the exchange of data among avionic subsystems. These proprietary, closed developments usually lead to high life cycle costs, as maintenance and system upgrades are bounded to the original manufacturers. Furthermore, other relevant capabilities such as portability across platforms or interoperability among subsystems are also restricted. To address these issues, multiple initiatives in the industry [6] [7] are supporting the use of open standards as a key element to preclude vendor lock-in and reduce software development and integration costs.

These initiatives together with the new technologies adopted by the industry (i.e., virtualization based on a hypervisor and Ethernet-based networks) facilitate the integration of commercial-off-the-shelf (COTS) distribution middleware. One major effort in this direction is the Technical Standard for Future Airborne Capability Environment (FACE) [7], which supports the use of middleware built upon open distribution standards to facilitate the interoperability and portability of software. This integration is not yet exempt from open challenges, as typical communication middleware usually aims at non-critical applications and thus it does not meet the safety requirements to be used in the avionic domain. However, this scenario is slowly changing, and some vendors are specifically developing products for the airborne market. For instance, this is the case of [8] and [9] for the Data Distribution Service for Real-Time Systems (DDS) standard [10].

The DDS standard is explicitly designed to build distributed real-time systems, as it supports a rich set of quality of service (QoS) parameters for fine control of non-functional properties. For instance, DDS has been used in robotics [11] or smart-grid [12] systems. Furthermore, the standard is evolving towards a safety-critical subset of its full distribution facilities [13][14].

From a communication viewpoint, AFDX complements ARINC 653 systems. Typically, the communication between partitions belonging to different end-systems is performed through ARINC 653 ports, which in turn are connected to the

This work was supported in part by MCIN/ AEI /10.13039/501100011033/ FEDER "Una manera de hacer Europa" under grant PID2021-124502OB-C42 (PRESECREL)

Accepted version of the paper published in the Proceedings of the 30th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2024), Sokcho (South Korea), pp. 73-84, August 2024
DOI: 10.1109/RTCSA62462.2024.00020

communication ports defined by AFDX. Consequently, both standards should be analysed in order to integrate DDS into avionics systems. There are several analysis for integrating DDS and ARINC 653 [15][16][17] that represent the basis for the integration of DDS in distributed avionics systems interconnected through AFDX. However, it is worth noting that the integration of DDS and AFDX is not straightforward, thus requiring a detailed analysis to determine which features of DDS can be used in this domain. This paper fills this gap by making the following contributions:

- Identification of a set of issues that may compromise the integration of DDS and AFDX technologies.
- Definition of a set of recommendations for the usage of DDS in avionics, which can be of relevance for the future safety-critical profile of DDS in this domain.
- Development of a software platform in order to estimate the overhead of using DDS with AFDX.

It is also worth considering that in this paper, we examine the integration from both standard perspectives, although the choice of a particular DDS implementation may require extending the analysis to include implementation-dependent aspects. Furthermore, the design choices made by each implementation may affect the response time of DDS applications [18].

This document is structured as follows. Section II introduces a brief review of the DDS and AFDX standards. Section III analyses the differences between these standards from the communication perspective and suggests ways of integrating them. Section IV outlines recommendations that could form part of the future safety-critical profile of DDS in the avionics domain. The implementation of a partitioned distributed platform that combines the use of DDS with an emulated AFDX network is described in Section V, together with some overhead and performance estimations. The related work is presented in Section VI. Finally, Section VII summarizes the main contributions and the lines of future work.

II. Background

A. Overview of DDS

The DDS standard [10] provides connectivity, portability and interoperability features in distributed real-time systems. The communications in DDS are based on the publisher - subscriber paradigm in which *DataWriter* and *DataReader* communication entities respectively write (produce) and read (consume) data through a fully distributed global data space. To support the requirements of distributed real-time systems, the standard also provides applications with QoS policies to control non-functional properties such as persistence, durability or timeliness.

In addition to user data, DDS applications also exchange middleware internal data in order to obtain information about the presence and characteristics of the DDS entities available in the distributed system (i.e., the *discovery* process). Furthermore, satisfying some of the QoS configurations in DDS also produces additional overhead in the network. Henceforth and in the context of this paper, this special kind of network traffic that is internal and automatically managed by middleware will be called *metatraffic*.

Another key feature of the standard is the interoperability among middleware implementations, which is provided by the DDS Interoperability Wire Protocol (DDSI-RTPS) [19]. It defines a set of exchange information protocols and message formats capable of supporting the main features of DDS (e.g., decentralized architecture, QoS configurations, automatic discovery, etc.). This protocol is based on a modular message design in which messages are composed of a fixed-size header followed by a variable number of sub-messages. This special design maximizes interoperability, as nodes can still participate in the network even though they are running different versions or they only implement a subset of the protocol (i.e., unknown sub-messages are ignored).

B. Overview of AFDX

AFDX is a communication network defined for ARINC systems and included in the ARINC 664 standard [5]. Although this communication network is based on standard Ethernet technologies, it adds significant restrictions to ensure deterministic timing behaviour [5], such as dedicated full-duplex Ethernet links, pre-defined network routing or bandwidth preservation through a traffic shaping mechanism.

The AFDX network mainly consists of two kinds of devices: end-systems and switches. While the former represents a source or destination of data on a network, the latter provides different network functionalities such as frame filtering, traffic policing, frame switching or network monitoring.

The AFDX specification should be interpreted in the context of IMA systems. Hence, partitions exchange messages via the application ports defined in the ARINC 653 standard [2], which in turn are connected to AFDX communication ports, as can be seen in Fig. 1. For both kinds of port, two types of services are defined: *sampling*, where only the last received message is available for reading (i.e., messages are always overwritten); and *queueing*, which allows multiple messages to be buffered.

A central feature in AFDX networks is the *virtual link*. It defines a logical unidirectional connection between end-systems. An important feature of AFDX networks is that a particular virtual link originates at one source end-system, but also at a single partition. However, messages can be originated from multiple communication ports, as long as they are allocated to the same source partition. Regarding the receiving end-systems, a virtual link can deliver messages to multiple partitions, belonging or not to the same end-system.

To prevent interference between virtual links that could lead to non-deterministic behaviour, virtual links are isolated from each other. This is accomplished by limiting the size and the transmission rate of messages associated with a virtual link. To this end, each virtual link is characterized by two parameters: the Bandwidth Allocation Gap (*BAG*) and the maximum frame size (*Lmax*). Data written to a communication port is only transmitted when the virtual link complies with the required

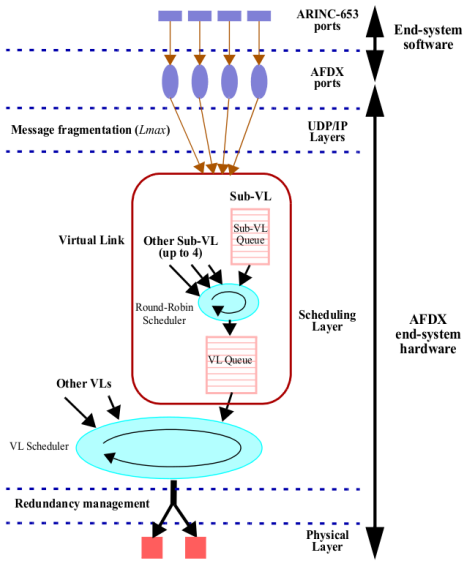

Fig. 1. Transmission stack in AFDX

bandwidth regulation for transmission. This task is performed by the virtual link scheduler, which is responsible for selecting the first packet from the FIFO queue associated with the virtual link (see Fig. 1). Optionally, a single virtual link can be comprised by up to four sub-virtual links. Each sub-virtual link has a dedicated FIFO queue, and all the queues are read on a round robin basis by the main FIFO queue associated with the virtual link.

To increase the robustness of the system and prevent the failure of network devices, AFDX relies on two independent and redundant networks operated on a per virtual link basis. The management of redundant transmission and reception of data is performed at the end-systems before data is delivered to the transmitting network or the receiving partition. Therefore, partitions are unaware of the underlying network redundancy so any error in either network is transparent to them [5].

Finally, the IMA development process requires the virtual links and their corresponding features to be statically defined in a configuration table. This task is performed by the system integrator, who is responsible for the integration and configuration of all components in the system.

III. Integration Analysis

The use of DDS in ARINC 653 systems may provide significant benefits for developing partitioned distributed systems [15][16][17]. For instance, it may provide a unified data distribution environment without concern for the actual physical location, the programming language or the underlying network services. This latter aspect is of paramount importance in modern mixed-criticality systems, where connectivity remains a core part of the system, even with open system architectures. Furthermore, DDS provides high-level mechanisms for the management of data [10][7] which not only reduces the application logic but also facilitates software integration and distribution.

A preliminar analisys for the integration of DDS and ARINC 653 was presented in [15]. This analysis presented three different integration architectures (see Fig. 2):

- *Configuration #1*: DDS as a transport layer for the ARINC 653 communication service
- *Configuration #2*: Standard DDS on top of ARINC 653
- *Configuration #3*: Safety-critical subset of DDS on top of the ARINC 653 communication service

The work in [15] argued that *Configuration #3* is a promising option for developing future heterogeneous partitioned systems, as it preserves most DDS features and it allows certification at high criticality levels. As a result, our research focuses on using the architecture depicted in *Configuration #3* to integrate DDS and AFDX.

Furthermore, the DDS distribution model resembles the one proposed for avionic systems. Similarly to ARINC 653, communication among partitions using AFDX networks is performed through sampling and queueing services [5]. These services are similar to the decentralized architecture proposed by DDS, in which distribution entities produce and consume data regardless of their location. However, the use of data-centric middleware allows the application code to raise the abstraction level, as applications do not need to directly write to the application ports and manage the logic related to the connectivity process.

Despite these similarities, some features of AFDX networks may compromise the integration of the two technologies. The remainder of this section deals with a set of integration issues and proposes possible solutions to address them. To better organize this analysis, the integration of DDS is discussed with regard to the main features of AFDX networks as follows:

- *Communication model*, or how data can be transmitted among end-systems.
- *Frame format and communication protocols*, or how messages are structured and exchanged in the network.
- *Bandwidth regulation*, which deals with the regulation of transmitted data on a per virtual link basis.
- *Failure protection mechanisms* to minimize or eliminate the impact of network failures.
- *Traffic prioritization mechanisms* to differentiate between traffic classes.

A. Communication model

As shown in Fig. 1, the queueing and sampling communication ports defined in AFDX have a direct correspondence to those defined in ARINC 653. Consequently, the analysis of the integration of DDS and the ARINC 653 communication service performed in [15] and [17] remains valid for AFDX. Nevertheless, AFDX includes some additional features that should be taken into account, all of them related to the virtual link mechanism.

Queueing and sampling ports are the communication points for the virtual link entity defined by AFDX. The virtual

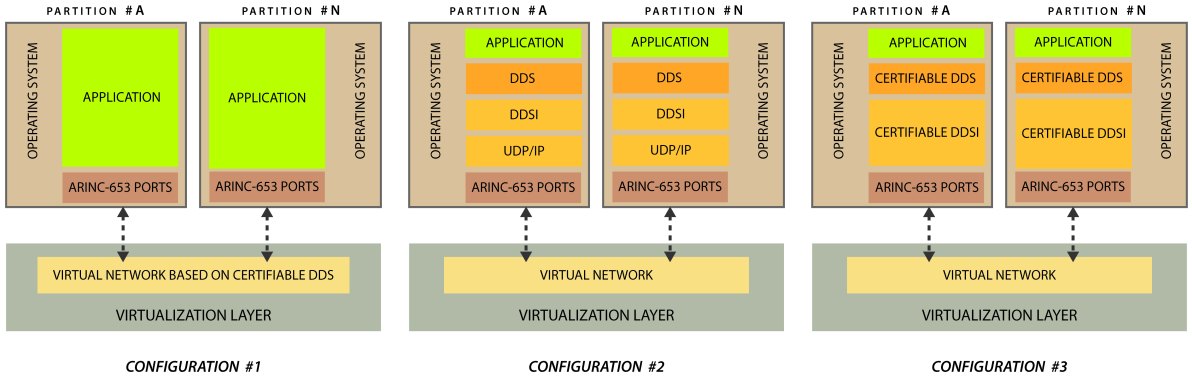

Fig. 2. Integration architectures for DDS and ARINC 653

link represents a logical unidirectional connection from one source end-system to one or more destination end-system/s. Hence, AFDX provides support for unicast and multicast communications. According to the AFDX standard, one virtual link cannot be shared by two or more source partitions within the same end-system, but a single source partition can have multiple communication ports using the same virtual link.

The limitations on the type of communications supported by AFDX restrict how DDS may share data. Both standards allow unicast and multicast communications, but DDS also provides support for many-to-one communications. This is a common scenario in DDS, where for instance a set of DataWriters can be responsible for updating the same *Topic* so that a single DataReader receives data from multiple sources. This scenario can be supported in partitioned systems using different application ports. In the particular case of AFDX networks, this means that many-to-one communication is logically managed at DDS level. Hence, a single DataReader can receive data from multiple communication ports, which are allocated per incoming data flow to the end-system. Furthermore, redundant DataWriters should use different virtual links, as they are commonly located in different partitions or end-systems. This scenario is illustrated in Fig. 3. It is worth noting that this approach is transparent to the end-user, as many-to-one communications over AFDX should be internally managed by middleware.

On the other hand, a single virtual link can be shared by different DataWriters as long as they (1) are allocated to the same partition and (2) share the receiving end-system/s. It is worth noting that the latter does not imply the same receiving partition at the end-system.

It is important to remark that this paper does not analyze whether multiple DataWriters should use different virtual links or a single shared virtual link, as it would depend on the system's real-time requirements. In any case, sharing a virtual link among several DataWriters may imply additional restrictions in the configuration of DDS. These restrictions will be discussed throughout the next subsections.

B. *Frame format and communication protocols*

To enable the communication among avionic applications, the AFDX specification identifies two types of message structures for building and decoding messages from the application layer [5]. In the context of DDS, this task is carried out by the standard DDSI-RTPS protocol [19], which provides interoperability among different implementations.

Similarly to AFDX, DDSI-RTPS is primarily based on UDP transport, although the specification allows other transport mechanisms to be used. Thus, several strategies can be applied to integrate DDS and ARINC-compliant systems.

This work focuses on the use of a safety-critical subset of DDS directly on top of the ARINC 653 communication service, as it is suitable for applications where a high-level of criticality is required: on the one hand, communications among partitions belonging to the same end-systems are directly built on top of the ARINC 653 communication service; on the other hand, communications among partitions belonging to different end-systems rely on the UDP/IP stack provided by the AFDX hardware. The proposed architecture is illustrated in Fig 4.

Under the proposed architecture, the use of DDS with AFDX networks would require adapting the DDSI-RTPS protocol to use a transport based on the ARINC 653 communica-

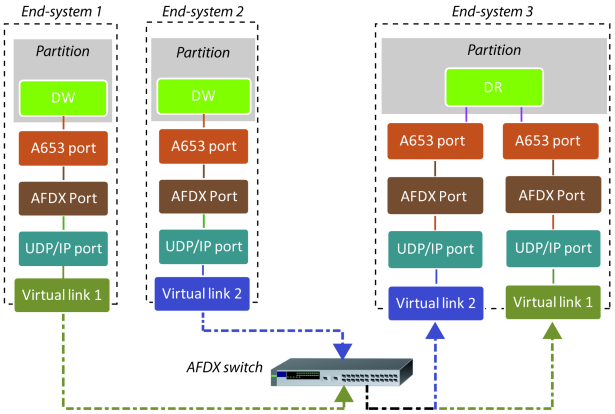

Fig. 3. Many-to-one communication in AFDX

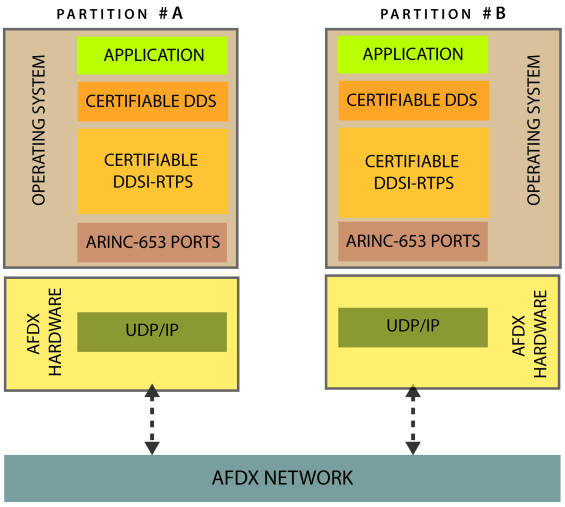

Fig. 4. Integration between DDSI-RTPS and AFDX

tion service. This new transport should modify some common behaviour of regular distribution middleware as follows:

- *The use of pre-defined ports to receive data*. DDS defines a set of pre-defined transport ports in order to facilitate interoperability and the automatic discovery of entities. However, the AFDX specification requires ports to be statically assigned by the system integrator, who is responsible for the assignment of ports throughout the distributed system. Consequently, DDS implementations cannot rely on pre-defined transport ports, but should determine which ports are available in each partition (e.g., via a static configuration table).
- *Sharing a single port to receive data from multiple sources*. As commented in subsection III.A, the many-to-one communication model is not directly supported. Therefore, this issue is addressed in the reader partition by creating as many ports as matched DataWriters (i.e., per data flow), as can be seen in Fig. 3.

To communicate via an AFDX network, each partition is assigned a unique IP address. From a DDS perspective, this means that inter-partition communications should be identical regardless of whether the partitions are allocated to the same end-system (i.e., messages are directly exchanged through the hypervisor) or not (i.e., messages are exchanged via the AFDX network).

In general-purpose systems based on Ethernet, a large block of data usually fragments at the IP level. For AFDX networks, the fragmentation service is also performed at the IP level, but it depends on each virtual link and its corresponding *Lmax* parameter. Specifically, while sampling communication ports do not support fragmentation (i.e., the size of messages and protocol overheads must be less than or equal to *Lmax*), queueing ports can only manage up to 8 Kbytes which will be fragmented according to the *Lmax* parameter. Therefore, the maximum payload size coming from partitions must be bounded and it cannot exceed these values.

The DDSI-RTPS protocol provides support for fragmenting messages whose size is over the maximum size allowed in UDP (i.e., 64Kbytes). Furthermore, the modular design of DDSI-RTPS implies that the size of messages is not pre-determined, as multiple sub-messages including user and/or protocol data may be combined into a single message. Therefore, DDS for AFDX should take into account the restrictions of AFDX in terms of message size and fragmentation. This issue can be addressed by pre-configuring the maximum size of messages written by a DataWriter.

Finally, and since DDS for AFDX would be built on top of the ARINC 653 communication service, the management of the UDP/IP stack is left to the specialized AFDX hardware, as shown in Fig. 1 and Fig. 4.

C. Bandwidth regulation

The virtual link concept provides partitioning at the network layer through the traffic shaping mechanism, which regulates the flow of data generated by different sources belonging to the same end-system.

To ensure determinism in the whole distributed system, all the network traffic should be regulated using the traffic shaping mechanism. In the case of DDS, middleware implementations exchange messages which could include not only user data but also protocol data (i.e., *metadata* or *metatraffic*). In AFDX networks, this additional traffic or overhead should be included in the corresponding virtual link.

It is worth remarking that the impact is even higher when middleware uses additional messages for this *metatraffic*. In this case, two options are available: (1) the creation of new virtual links for this protocol data or (2) the reuse of the virtual links defined for user data. On the one hand, the former option strongly depends on the DDS implementation and configuration and therefore it adds complexity. Furthermore, adding a new virtual link impacts the worst-case latency of the remaining virtual links. On the other hand, the latter option is simpler since the number of virtual links is unchanged. However, it strongly impacts the worst-case latency of the corresponding virtual link, as it increases the number of messages. In DDS, part of this extra latency could be controlled by means of the Transport_Priority and Latency_Budget QoS parameters to prioritize communications, but the standard is not very precise about the aim and implementation requirements of these QoS parameters. In any case, this extra latency cannot be fully removed since the virtual link relies on a FIFO queue for all the packets ready to be transmitted. As a result, the DDS metatraffic should be minimized whenever possible. To this end, the following DDS configurations should be taken into account:

- *Discovery metadata*: DDS exchanges data between built-in DataWriters and DataReaders. However, a static approach is more suitable for partitioned systems [20]. Under this approach, all the information about DDS entities can be statically configured before runtime (i.e., no discovery data is sent through the network).

- *Liveliness metadata*: The Liveliness QoS parameter determines whether or not a DDS entity is still alive. In this case, DDS supports two approaches to assert liveliness: (1) automatic, in which middleware is responsible for signaling the liveliness using a built-in DataWriter; or (2) manual, in which the user must explicitly or implicitly signal the liveliness of DDS entities. Furthermore, two possible manual settings are defined to control the granularity at which the application must assert liveliness: *manual by participant*, in which the liveliness of a set of DataWriters can be maintained at the same time by using a middleware built-in DataWriter; and *manual by topic*, in which the liveliness in maintained per DataWriter. For AFDX networks, the manual by topic setting is preferred as it minimizes the overhead in the network and it does not require adding extra virtual links for the communications associated with the built-in entities.
- *Reliable metadata*. The Reliability QoS parameter addresses the loss of messages in the network. To this end, periodic messages are sent to the reliable readers in order to inform about the last available samples. In response to these periodic messages, readers must also acknowledge the reception of all available samples or they must indicate that some samples are missing. On the other hand, best-effort readers/writers do not send this kind of metadata. As this QoS parameter is directly related to the failure protection mechanisms, it will be further discussed in subsection III.D.
- *Time_Based_Filter metadata*: The Time_Based_Filter QoS parameter provides support for limiting the number of data samples received in a period of time. This parameter applies per DataReader, even those subscribing to the same topic. The filtering can be applied on both writer and reader side. The former option requires the transmission of metadata to denote that the sample was filtered and not simply lost. This option also increases the complexity, as it may require specifying a variable number of filtered samples in the metadata or applying multiple filters from different readers at the same time [19]. Therefore, filtering on the reader side is simpler and it does not require transmitting metadata through the network.
- *Durability metadata*: The Durability QoS parameter provides support for making historical data available to any potential late-joining DataReaders. Although this parameter can take up to four different settings, the DDSI-RTPS specification only covers two of them for interoperability purposes: volatile (i.e., historical data does not need to be maintained for late-joiners) and transient-local (i.e., historical data needs to be maintained in memory as long as the associated DataWriter is active). As commented before, the AFDX network requires the static configuration of communications and therefore late-joiners are not allowed. However, this parameter may still be useful to facilitate the startup sequences or increase the tolerance to failure conditions (for instance, a partition reset). In addition to the operational overhead for maintaining the historical data, this QoS parameter also produces an increase of the network traffic when the transmission of the historical data takes place. The effect of this extra traffic on the rest of the system is bounded by the use of the virtual link mechanism, which regulates the flow of data by setting a maximum frame size (i.e., *Lmax*) and a minimum interval between frames transmitted on the corresponding virtual link (i.e., *BAG*). Other QoS parameters defined by DDS can be used to limit the number of historical data samples stored and transmitted [10].

The traffic shaping mechanism defined by AFDX may also influence the QoS configurations in DDS. In particular, the QoS parameters related to some kind of temporal deadline may be restricted by AFDX networks. Hence, the configuration of the following parameters should take into account the traffic shaping mechanism as follows:

- *Deadline*: This QoS parameter indicates the maximum amount of time available to send/receive data samples belonging to a particular Topic. This QoS parameter is restricted by several factors in partitioned systems, such as the temporal isolation, the shared access to the network device or the underlying communication network. Regarding the AFDX network, the Deadline parameter is influenced by the selected BAG and *Lmax* even for the simplest scenario, that is, without fragmentation or sub-virtual links usage. Therefore, the Deadline parameter could be guaranteed through the worst-case end-to-end latency (L_{VL}) for the message stream sent through the virtual link or an upper bound for it. There are analytical techniques which can compute upper bounds of L_{VL} such as those based on forward analysis [21] or worst-case response times [22].
- *Liveliness*: As commented earlier, this parameter determines whether or not a DDS entity is still alive. To this end, DataWriters signal that it is alive with a delay lower than a specific *lease_duration*. Similarly to the Deadline parameter, this value should be greater than the end-to-end latency for the message stream sent through the virtual link.

D. Failure protection mechanisms

AFDX provides support for redundancy to protect against network failures. In general, each message is sent through two networks. Upon reception, only the first valid message received from either network is accepted and delivered to the receiving partition. As a consequence of using redundant AFDX networks, both queueing and sampling communication services may be simple, connectionless and without acknowledgements.

In the case of DDS, the standard defines three main mechanisms to provide protection against failures: firstly, the Reliability QoS parameter, where middleware will automatically manage the communication to address the loss of messages at network level; secondly, the Ownership QoS parameter, which

enables transparent failover between redundant DataWriters and thus is applied at partition level; and thirdly, the Durability QoS parameter, which allows the retransmission of data to temporally out-of-contact DataReaders, and thus it is also applied at partition level.

As a result, the redundancy implemented in the AFDX specification and the Reliability QoS parameter work at network level to provide protection against network failures. It is worth remarking that the loss of messages is not fully avoided when using the redundancy management mechanisms in AFDX networks [5]. For instance, the loss of messages could happen if message "A" is lost on one of the redundant networks and the subsequent message "B" arrives earlier at the destination than the copy of message "A" transmitted by the other network. Under this scenario, the *Reliable* setting could address the loss of messages in AFDX at the cost of not only increasing overhead but also complexity, which may lead to search for simpler approaches: for instance, this issue could also be addressed using other approaches such as the proposed in [23].

On the other hand, the Ownership and Durability QoS parameters work at the partition level and thus they can cooperate with the redundancy mechanism provided by AFDX, which works at the network level. In this case, both *Exclusive* and *Shared* settings are available for the Ownership QoS parameter, while only *Volatile* and *Transient_Local* settings are available in the case of the Durability QoS parameter.

E. Traffic prioritization mechanisms

The AFDX standard enforces appropriate switching mechanisms to guarantee deterministic behaviour. One of these mechanisms is the traffic prioritization at output ports of the AFDX switch. The switch should be able to differentiate between traffic classes (high priority and low priority) on a per virtual link basis.

In the case of DDS, traffic prioritization is performed through the Transport_Priority QoS parameter. This parameter allows the priority of the underlying transport used to send data from a particular DataWriter to be set. However, this parameter is considered a hint in the specification, which makes it hard to determine whether it may or may not be used in AFDX networks.

Nowadays, most high-end Ethernet switches support traffic prioritization based on the type of service (ToS) field in the IP header. Hence, some DDS implementations could take advantage of this feature to resolve data contention at output ports in the switches. However, it is worth noting that the ToS field must not be used in AFDX networks, where traffic prioritization should be statically defined on a per virtual link basis (i.e., during the system integration phase). Furthermore, several DDS topics with different priorities could be sent through the same virtual link (e.g., using multiple sub-virtual links), which is not allowed either. However, the loose definition of this QoS parameter provides implementations with great flexibility to apply it in a compatible way to AFDX. For instance, some implementations could use this parameter to prioritize the transmission of messages before their delivery to the network stack. Consequently, the use of the Transport_Priority QoS parameter with AFDX networks depends on the middleware implementation.

IV. Towards the Safety-Critical Profile

The integration of standard distribution middleware in partitioned systems represents a promising approach for the future development of distributed applications in avionics. In the case of the DDS specification, it is already envisioned by the FACE standard [7], and by the ongoing development of a safety-critical profile for DDS [13][14].

As a result, the previous analysis for the AFDX avionics network can be of interest for this new profile for DDS. This analysis has shown that the use of virtual links restricts the use of DDS in terms of the communication model, the underlying transport, the communication protocol and the QoS parameters. In the context of the DDS specification, this means that the standard should not only be restricted at the configuration level, but it should also add restrictions from the operational point of view.

Although the proper configuration of the QoS parameters is already a hard task in distributed real-time systems, it is even more complex when DDS entities produce/consume data using the same virtual link. In this scenario, these entities must also share some of their QoS parameters, as the traffic shaping mechanism is defined per virtual link.

Unlike traditional DDS applications, it is also worth noting that the usage of DDS in AFDX networks requires the static configuration of the QoS parameters. Once the system integrator has dealt with the IMA system configuration and transferred the collected data into the configuration tables, middleware should determine the corresponding QoS settings from them and disable their reconfiguration at runtime.

Therefore, while some QoS parameters may not be necessary in the context of AFDX due to their loose definition in the current DDS standard (e.g., Transport_Priority), others can only be configured with a restricted set of their available settings (e.g., Durability or Liveliness). Furthermore, the QoS parameters which depend on the end-to-end latency should have a value according to the underlying traffic shaping mechanism (e.g., Deadline or Liveliness). The Reliability QoS parameter represents a complex case since the *Best-Effort* is the preferred setting for AFDX, but the *Reliable* setting could be of interest for scenarios where the loss of any sample cannot be tolerated. However, the extra complexity and overhead introduced by this setting may lead to search for simpler alternatives. Table 1 summarizes the QoS restrictions for DDS identified when using an AFDX network.

In addition to these restrictions, the implementation of the DDS distribution model should take into account the following considerations. Firstly, the DDSI-RTPS protocol requires the definition of a new transport based on the ARINC 653 communication ports. Unlike traditional UDP/IP transport used in DDSI-RTPS, the use of ARINC 653 and AFDX implies that the communication ports can no longer be predefined

Table 1. QoS restrictions for DDS in AFDX networks

QoS	Supported	Applicable settings	Additional restrictions
Reliability	Partially	Best-Effort	-
Ownership	Yes	Shared/Exclusive	-
Deadline	Yes	-	$deadline \geq L_{VL}$
Liveliness	Partially	Manual by topic	$lease\_duration \geq L_{VL}$
Time based filter	Yes	-	Filtering on the reader side
Durability	Partially	Volatile/Transient_Local	-
Transport priority	-	-	Implementation-dependent

or shared among multiple DDS entities. Furthermore, DDSI-RTPS messages should be restricted in size to comply with the requirements of the underlying communication service. Secondly, the discovery of entities should be statically performed, for example, by obtaining the location data together with the QoS configurations from the system integration phase. A comprehensive summary of the DDS and AFDX integration challenges is shown in Table 2, which denotes the list of issues and proposed solutions.

Finally, one key objective of using open standards is supporting the addition or substitution of software components from different vendors. Hence, interoperability between different DDS implementations supporting this new profile should be guaranteed as long as they are compliant with the requirements of the DDSI-RTPS protocol, including the aforementioned restrictions.

V. Evaluation

This Section aims to assess the proposed integration by implementing a distributed partitioned platform from which some performance metrics can be obtained.

A. The distributed partitioned platform

The proposed platform relies on DDS for the distribution of data, which in turn relies on the ARINC 653 communication service to interconnect partitions within the same end-system. Additionally, end-systems are interconnected through an AFDX emulator [24]. Beyond the integration, our development has also focused on adapting the partitioned platform towards multi-core systems in order to minimize the overhead associated with time partitioning.

The main points addressed in the development of the distributed partitioned platform are briefly described next.

Extensions to the hypervisor: The proposed platform relies on the XtratuM hypervisor [25], version 3.7.3, to enforce spatial and temporal isolation as required by ARINC 653. It provides different services to partitions such as timing, interrupt management, scheduling or inter-partition communications.

When executing on multi-core systems, XtratuM can also provide partitions with one or more virtual CPUs. To enable communication and event notification between different virtual CPUs, a special kind of interrupt called IPVI (Inter-Partition Virtual Interrupt) is defined. This special interrupt is triggered by a partition to inform about some relevant event to partitions allocated in different virtual CPUs.

XtratuM handles shared devices by means of special partitions called I/O partitions. In the proposed distributed partitioned platform, for instance, multiple partitions may require access to the AFDX network and therefore the driver contention is handled through the corresponding I/O partition. Under this approach, all the partitions belonging to the same node are interconnected through the standard ARINC transport mechanism; and communications between partitions allocated to different nodes must be performed via the pseudo-partition, which will be responsible for redirecting messages to the communications network.

Finally, XtratuM only considers communications among partitions within the same end-system. Thus, communications among remote partitions require extending the hypervisor with new configuration parameters. These new parameters are used

Table 2. Distribution model restrictions for DDS in AFDX networks

Feature	AFDX	DDS	DDS for AFDX
Pre-defined communication ports	No	Yes	Ports defined by system integrator
Shared communication port	No	Yes	Port per DW/DR entity
Transport protocol	UDP	UDP	New ARINC-653 transport
Message size	Bounded	Unbounded	Restrictions on DDSI-RTPS protocol
Discovery	Static	Dynamic	Static
Communication model	Unicast, Multicast	Unicast, Multicast, Many-to-one	Many-to-one implemented at DDS level

to generate the discovery information required to interconnect remote partitions through DDS.

Extensions to the real-time operating system: The real-time operating system used in the platform is called MaRTE OS [26]. It is a real-time kernel which follows the POSIX.13 minimal real-time system profile and provides support for concurrent applications, any of which could be written in C or Ada programming languages.

In the context of the platform, applications running in a partition on top of MaRTE OS do not need to be directly virtualized and they use the services provided by the operating system. However, the hardware abstraction layer of MaRTE OS need to be modified in order to access the system resources by means of the interface provided by XtratuM.

Since distributed applications based on distribution middleware usually rely on a direct access to the network interface and XtratuM enforces the use of the I/O partition mechanism for shared devices, MaRTE OS has been extended to provide partitions with a virtual network interface. From the distributed application viewpoint, the virtual network interface features the same basic functionality as network devices (open, send, receive, close, etc.) but inter-partition communications are internally based on the ARINC 653 communication service provided by XtratuM. In particular, the proposed communication mechanism present two major features:

- *Blocking and non-blocking receiving operations.* While the latter is directly implemented on top of the communication services provided by XtratuM, the former is based on extended interrupts to determine on which port the message is received.
- *Management of IPVIs*. In multi-core scenarios, the virtual network interface is responsible for triggering an IPVI to notify the delivery of an inter-partition message whose destination is allocated in another virtual CPU as soon as it is transmitted.

Extensions to the DDS implementation: RTI Connext Micro is a minimal DDS implementation aimed at resource-constrained devices which has been designed as a certifiable component for safety-critical systems. In the context of this work, version 3.0.3 has been ported to the distributed partitioned platform. In particular, our work has focused on two developments:

- *Development of a port to MaRTE RTOS*: This middleware provides a kernel abstraction layer to facilitate the access to different kernel services such as threading, memory management, timing or protecting shared resources.
- *Development of a transport plugin for ARINC 653 communications*. This plugin implements an abstract transport API defined by the middleware that allows the ARINC-based transport to be registered in middleware, announcing it to other participants and sending and receiving messages over the ARINC 653 communication service provided by XtratuM via the virtual network interface previously described.

Integration of the AFDX Emulator: The authors in [24] presents a software written in Ada to emulate the behaviour of an AFDX network on top of standard Ethernet hardware. This emulator does not comply with all the timing requirements specified by the AFDX standard, but it is considered to be suitable for training or research purposes as it provides the basic functionality of AFDX networks such as the definition and configuration of virtual and sub-virtual links, the transmission and reception of messages at the end systems, fragmentation, etc.

In this case, our development focused on adapting the software to be distributed as a library and thus facilitating the integration in the building process. Furthermore, some minor bugs identified in the reception of data has also been fixed.

B. Performance metrics

This subsection provides some overhead metrics of using DDS for information dissemination in the avionics domain. To this end, we revisit the simulated Traffic alert and Collision Avoidance System (TCAS) described in [27]. The TCAS system is sufficiently complex to estimate the overhead of using DDS with AFDX while keeping the scale appropriate for the available development platform. The main objective of this surveillance system is to prevent mid-air collisions by monitoring the airspace for nearby aircrafts. When a threat of mid-air collision is detected, the TCAS system shows proximity warnings in the target display. Fig. 5 shows a simplified TCAS system which is composed of two end-systems:

- ***Air-to-air surveillance end-system***: It is divided into three main subsystems: (1) the *data acquisition subsystem*, which provides the position and altitude of aircrafts within the monitored area with a data acquisition rate of 500 samples per second; (2) the *control panel subsystem*, which allows the pilot to configure the sensitivity level that should be used to determine possible threats; and (3) the *threat detection subsystem*, which examines the surveillance data generated by the data acquisition subsystem in order to determine the existence of nearby aircrafts. If a potential threat is detected or a risk of collision exists, it provides the user interface subsystem with traffic and resolution advisories, respectively.
- ***Alarm triggering end-system***: It comprises the *user interface subsystem*, which is responsible for describing the state of the TCAS. It also reports the necessary warnings to the pilot according to the selected operation mode. In this case, the available information is reported by means of two kinds of displays: (1) the traffic display and (2) the resolution advisories (RA) display. In the case of RA, it may be required to have one display per pilot.

In this system, all the data flows are asynchronous. The *data acquisition subsystem* periodically send data samples about nearby aircraft/s to the *threat detection subsystem* in order to process them. Then, the threat detection subsystem sends warnings to the corresponding displays managed by the *user interface subsystem*. Additionally, the control panel subsystem

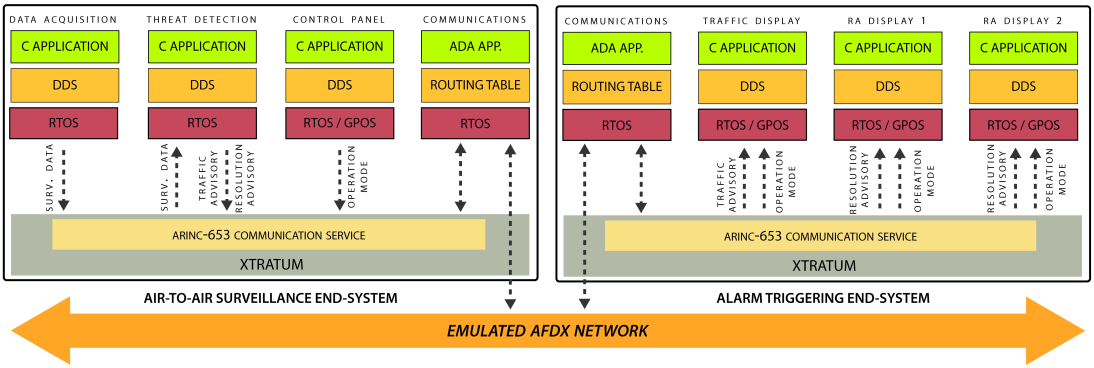

Fig. 5. The TCAS system

aperiodically sends the sensitivity levels to configure the *user interface subsystem*. Since this example aims at evaluating the overhead of the proposed DDS and AFDX integration, two different scenarios has been implemented for the distribution of data:

- ***DDS distribution***: This requires the mapping of the TCAS system to a DDS architecture by defining the data structures that are exchanged within the distributed system. For instance, 4 different topics have been defined: *Surveillance_Data*, *Traffic_Advisory*, *Resolution_Advisory* and *Control_Panel*.
- ***Adhoc distribution***: This is the reference case in which the data distribution does not rely on middleware. Partitions interconnect via predefined communication channels statically configured before runtime.

The hardware platform is composed of two quad-core processors with a clock rate of 2.8 and 3.2 Ghz. The system has been configured to have a dedicated core for each partition in both end-systems, thus minimizing the overhead associated with time partitioning. Threads running on each partition are scheduled using a fixed-priority preemptive scheduler provided by MaRTE OS.

Since the whole set of data flows are asynchronous, an oscilloscope has been used to obtain the response times by measuring the delay between two digital signals. Without loss of generality, the evaluation consists of obtaining the response time of triggering a resolution advisory in one display. To this end, two different data flows are evaluated: the first data flow is between the data acquisition and the threat detection subsystems, and the second data flow is between the threat detection and the resolution advisory displays. In order to obtain the metrics, the *Data Acquisition* partition sets/clears the first digital signal under evaluation and publishes a *Surveillance_Data* sample; then, this sample is captured and processed by the *Threat Detection* partition, which in turn sends traffic and resolution advisories; once the advisory is shown on the display, the *RA Display 2* partition sets/clears the second digital signal under evaluation, and therefore the delay between both signals can be measured. It is worth remarking that the remaining data flows could be evaluated using a similar procedure.

To better estimate the performance figures, the operation under evaluation is executed a minimum of 10,000 times and the payload is bounded in order to avoid any fragmentation at transport-level. Furthermore, response times are only measured after initialization and discovery processes have been completed (i.e., the distributed partitioned system is in a steady state). It is worthy of consideration that emulating the AFDX network increases the response times compared to using specialized AFDX hardware [24]. However, this overhead is added to both scenarios and therefore the temporal cost of using DDS can still be evaluated.

The images shown in Fig. 6 and Fig. 7 were generated using the oscilloscope, where the y-axis represents voltages (2 volts per division) and the x-axis represents times (200 microseconds per division). Both figures show the minimum, average and maximum response times obtained in the evaluation for each proposed scenario, together with the standard deviation. The *DDS distribution* scenario obtains slightly higher values (about 380 microseconds for the maximum value) than the *Adhoc distribution* scenario due to the extra overhead of using a distribution middleware. For this particular example, the data flow under evaluation includes at least 4 send / receive operations which are handled by middleware in the *DDS distribution* scenario. As a result, it takes less than 50 microseconds per send / receive middleware operation on average. Additionally, both scenarios present a standard deviation below 10 microseconds, which shows that middleware adds extra overhead but without increasing the dispersion. As a trade-off for this extra overhead, DDS facilitates the integration of heterogeneous software and it provides several quality-of-service (QoS) policies, among other benefits.

VI. Related work

Although the integration of DDS in safety-critical scenarios is still at an early stage in the avionics domain, a few research works have already been published. For instance, the use of DDS and time-triggered networks in avionic systems is discussed in [28]. Furthermore, the authors in [29] evaluate the impact of using DDS in a simulated automatic flight control system from the application performance perspective.

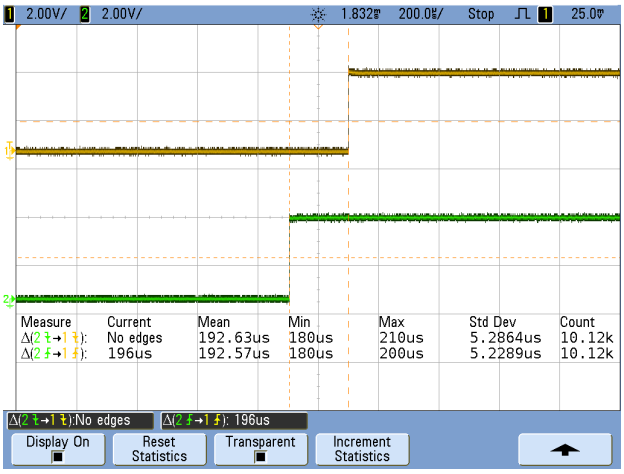

Fig. 6. Response times obtained for the *Adhoc distribution*

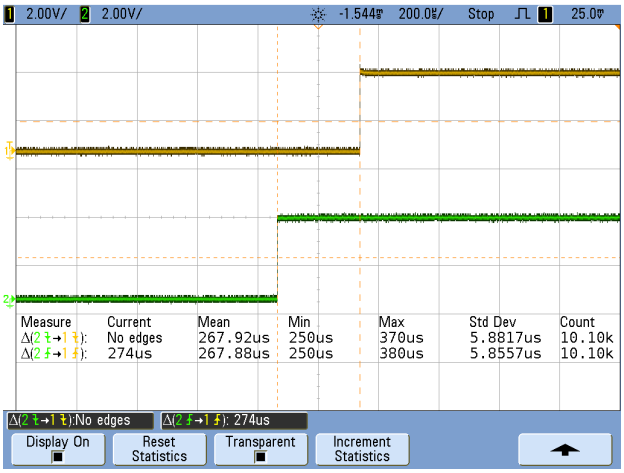

Fig. 7. Response times obtained for the *DDS distribution*

In the context of combat management systems, the work in [30] proposes a new discovery protocol of DDS entities in order to reduce the data exchanged in large distributed real-time systems. This protocol guarantees that the number of messages sent to the network does not depend on the number of DataWriter or DataReader entities, but Participants. However, the static nature of AFDX communications prevents its application in partitioned systems, where static discovery is more suitable.

The execution of DDS in a virtual environment was addressed in previous works. For instance, the authors in [31] explore the performance of DDS over general-purpose virtualization software, while DDS is used to interconnect virtual resources on heterogeneous hypervisors in [32]. The integration with ARINC 653 partitioned systems was dealt with in [17] and in [15]. While the former explores the integration of DDS for the communication between end-systems (i.e., DDS is only executed in the pseudo-partition or the partition which interacts with the external system), the latter not only integrates the distribution model proposed by DDS for communications between end-systems, but also considers the communication between partitions within the same end-systems (i.e., DDS is executed in all the partitions with communication requirements). The authors in [33] also deal with the integration of DDS into partitioned systems by proposing an architecture in which each partition has direct access to the shared network interface. Furthermore, [34] uses DDS to interconnect partitioned and non-partitioned nodes. Our paper represents a step forward to these works as it relies on the use of the special-purpose AFDX network, instead of the traditional UDP/IP network, while DDS is executed in all the required partitions.

Similarly to our work, the use of other networks in DDS has been addressed for automotive systems, where [35] and [36] integrate the CAN bus and the FlexRay communications system, respectively. Furthermore, [37] proposes a model-based design block in Simulink to facilitate the use of DDS with vehicular controllers.

VII. Conclusions and Future Work

New approaches are being investigated by the avionics industry to reduce times and costs in software development. This is the case of the FACE standard initiative, which aims at promoting interoperability and software reuse by applying open standard solutions such as DDS.

The use of DDS in the avionics domain is a promising approach towards more cost-effective software development and integration. However, the current DDS specification is not suitable for safety-critical environments so the development of a new profile for DDS suitable for certification is essential. To contribute in the development of the safety-critical profile, this paper has explored the features of the DDS and ARINC 664 standards that may compromise the integration of both technologies. Hence, the paper has analysed the communication models proposed by these standards, their frame formats and communications protocols, their available failure protection mechanisms, and their capabilities for bandwidth regulation and traffic prioritization.

As a result, a set of incompatibilities and recommendations has been identified, which can be part of the future safety-critical profile of DDS in the avionics domain.

Additionally, this paper has presented a distributed partitioned platform and the required extensions to integrate DDS and an AFDX network emulator. Some performance metrics were also presented and the results show the feasibility of the proposed integration given the overhead and dispersion values obtained.

In the short term, we plan to continue our investigation into DDS for the avionics domain. For instance, further research is required to fully determine which advanced features of DDS can be applied in future airborne systems, such as the use of keys.

REFERENCES

[1] Avionics Systems Hosted on a distributed modular electronics Large scale dEmonstrator for multiple tYpe of aircraft (ASHLEY) EU Project, 7th Framework Progr., 2017.

[2] Airlines Electronic Engineering Committee. “Avionics Application Software Interface, required Services”. ARINC Specification 653-1. 2010.

[3] S. Han and H. Jin. “Resource partitioning for Integrated Modular Avionics: comparative study of implementation alternatives”. Software: Practice and Experience, http://dx.doi.org/10.1002/spe.2210. 2014.

[4] R. L. Alena, J. P. Ossenfort, K. I. Laws, A. Goforth and F. Figueroa. “Communications for Integrated Modular Avionics”. Proc. of the IEEE Aerospace Conf., pp.1-18. doi:10.1109/AERO.2007.352639, 2007.

[5] Airlines Electronic Engineering Committee, Aeronautical Radio INC. “ARINC Specification 664 P7-1: Aircraft Data Network, Part 7 - Avionics Full Duplex Switched Ethernet Network”. 2009.

[6] Systems Engineering Guidebook, Defense Acquisition University, 2022.

[7] The Open Group. Technical Standard for Future Airborne Capability Environment, v 3.2. FACE Consortium, 2023.

[8] Real-Time Innovations (RTI) Connext Cert. https://www.rti.com/products/connext-cert. Last access: March 2024.

[9] Twin Oaks Comp. CoreDX DDS-SE. https://www.twinoakscomputing.com/CoreDX_DDS_Safety-critical_Edition. Last access: March 2024.

[10] Object Management Group. Data Distribution Service for Real-time Systems. OMG Document, v1.4. 2015.

[11] Z. Xiaowen, Z. Xiaogang, W. Shaoyuan and X. Ping, "Design and Implementation of Robot Middleware Service Integration Framework Based on DDS". IEEE International Conference on Real-time Computing and Robotics (RCAR), Guiyang, China, pp. 588-593, doi: 10.1109/RCAR54675.2022.9872212, 2022.

[12] S. Saxena, H.E.Z. Farag, N. El-Taweel, "A distributed communication framework for smart Grid control applications based on data distribution service" Journal of Electric Power Systems Research, Vol. 201, doi: 10.1016/j.epsr.2021.107547, 2021.

[13] R. Wahlin, and G. Hunt, “Towards a Safety Critical profile for DDS,” Real-time and Embedded Systems Workshop, Arlington, USA, 2009.

[14] R. Karoui, and A. Corsaro. “Real time Data Distribution for Airborne Systems,” Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems, Washington DC, (USA), 2011.

[15] H. Pérez and J. J. Gutiérrez. “Handling heterogeneous partitioned systems through ARINC-653 and DDS” Comput. Stand. Interfaces 50, C, 258-268. doi:https://doi.org/10.1016/j.csi.2016.10.012. 2017.

[16] M. García-Valls, J. Domínguez-Poblete, I. E. Touahria and C. Lu. “Integration of Data Distribution Service and distributed partitioned systems” Journal of Systems Architecture 83, 23-31. doi:https://doi.org/10.1016/j.sysarc.2017.11.001. 2018.

[17] T. Chen, X. Hu, G. Zhang and J. Xiao. “Implementation of data distribution service interface based on ARINC653 system” 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, pp. 428-432, doi:10.1109/ICIEA.2018.8397755. 2018.

[18] G. Sciangula, D. Casini, A. Biondi, C. Scordino, and M. Di Natale. "Bounding the Data-Delivery Latency of DDS Messages in Real-Time Applications". 35th Euromicro Conference on Real-Time Systems (ECRTS). Leibniz International Proceedings in Informatics (LIPIcs), Vol 262, pp. 9:1-9:26, 2023.

[19] Object Management Group. The Real-time Publish - Subscribe Wire Protocol. DDS Interoperability Wire Protocol (DDSI-RTPS) Specification. OMG Document, v2.5, formal/2021-03-03, 2021.

[20] H. Pérez, and J. J. Gutiérrez, “Enabling data-centric distribution technology for partitioned embedded systems”, IEEE Trans. on Parallel and Distributed Systems, vol. 27, no. 11, pp. 3186-3198, doi:10.1109/TPDS.2016.2531695, 2016.

[21] N. Benammar, F. Ridouard, H. Bauer and P. Richard. “Forward End-To-End delay Analysis for AFDX networks” IEEE Trans Ind Inform, In Press, doi:10.1109/TII.2017.2720799. 2017.

[22] J. J. Gutiérrez, J. C. Palencia, M. G. Harbour. “Holistic schedulability analysis for multipacket messages in AFDX networks”, Journal of Real-Time Systems 50(2), Springer, pp. 230-269, 2014.

[23] M. Li, G. Zhu, Y. Savaria and M. Lauer. “Reliability Enhancement of Redundancy Management in AFDX Networks”, IEEE Trans. Ind Inform, vol. 13, pp.2118-2129, doi:10.1109/TII.2017.2732345. 2017.

[24] J. Fernández, H. Pérez, J. J. Gutiérrez and M.G. Harbour. “AFDX Emulator for an ARINC-Based Training Platform”. In: Reliable Software Technologies – Ada-Europe. Lecture Notes in Computer Science, vol 9111. Springer, doi:10.1007/978-3-319-19584-1_14. 2015.

[25] M. Masmano, I. Ripoll, A. Crespo, and J.J.Metge, “Xtratum a hypervisor for safety critical embedded systems”, in; Proceedings of the 11th Real-Time Linux Workshop, Dresden, Germany, 2009.

[26] M. Aldea and M. González Harbour. “MaRTE OS: An Ada Kernel for Real-Time Embedded Applications”. Proc. of the International Conference on Reliable Software Technologies, Ada-Europe 2001, Leuven, Belgium, in Lecture Notes in Computer Science, LNCS 2043. 2001.

[27] RTCA / DO-185B, Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System (TCAS II) Airborne Equipment. 2008.

[28] Y. Gao, F. He, S. Yu and H. Xiong. "Publish/Subscribe Architecture for Airborne Time-triggered Network in Avionics System". IEEE/AIAA 41st Digital Avionics Systems Conference (DASC), Portsmouth, USA, pp. 1-8, doi:10.1109/DASC55683.2022.9925729, 2022.

[29] K. Landry, J. F. Boland and G. Bois, “Integration and Performances Analysis of a Data Distribution Service Middleware in Avionics”, AeroTech Congress and Exhibition, SAE Tech. Paper, doi:10.4271/2015-01-2554, 2015.

[30] H. A. Putra and D.S. Kim, “Node discovery scheme of DDS for combat management system”, Comput. Stand. Interfaces, Vol 37, pp. 20-28, doi:10.1016/j.csi.2014.05.002, 2015.

[31] M. García-Valls and P. Basanta-Val. “Analyzing point-to-point DDS communication over desktop virtualization software”, Comput. Stand. Interfaces, Vol 49, pp. 11-21, doi:10.1016/j.csi.2016.06.007, 2017.

[32] Y. Cho, J. Choi, and J. Choi, “An integrated management system of virtual resources based on virtualization API and data distribution service,” ACM Cloud and Autonomic Computing Conf., USA, 2013.

[33] Q. Zhou, Z. Xiong, Z. Zhan, T. You and N. Jiang, “The mapping mechanism between Distributed Integrated Modular Avionics and data distribution service”, Proc. of the 12th Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, pp. 2502-2507. doi: 10.1109/FSKD.2015.7382348, 2015.

[34] B. Song, X. Hu, J. Xiao, G. Zhang, S. Wang and Q. Zhou, “Integration of Data Distribution Service into Partitioned Real-time Embedded Systems” 15th IEEE Conf. on Industrial Electronics and Applications, Norway, pp.1606-1611, doi:10.1109/ICIEA48937.2020.9248298. 2020.

[35] R. Ekik and S. Hasnaoui, “Application of a CAN bus transport for DDS middleware”, In 2nd Int. Conference on the Applications of Digital Information and Web Technologies (ICADIWT), pp.766 –771. 2009.

[36] R. Bouhouch, H. Jaouani, A.B. Ncira and S. Hasnaoui. “DDS on Top of FlexRay Vehicle Networks: Scheduling Analysis”, Int. Journal of Computer Science and Artificial Intelligence, Vol. 3 - 1, pp.10-26. 2013.

[37] M. Takrouni, R. Bouhouch, S. Hasnaoui. “Simulink Implementation of the Data Distribution Service for Vehicular Controllers on Top of GBE and AFDX” The Computer Journal, Vol 64 - 6, pp.860-879, doi: 10.1093/comjnl/bxaa176, 2021.