Data Distribution Service (DDS) over Time Sensitive Networking (TSN) for launchers

José Miguel Sánchez Martínez
On board Software
(GMV)
Madrid, Spain
jomsanchez@gmv.com

Pablo Galeano Sánchez
Space Avionics and Equipment
(GMV)
Madrid, Spain
pgaleano@gmv.com

Carlos Domínguez Sánchez
On board Software
(GMV)
Madrid, Spain
cdsanchez@gmv.com

Jaime Martin Losa
(eProsima)
Madrid, Spain
jaimemartin@eprosima.com

***Abstract*—The Data Distribution Service (DDS) is a middleware protocol to facilitate real-time data exchange. A recent DDS standard extension [3] has been published to leverage the deterministic and highly reliable communication provided by Time Sensitive Networking (TSN).**

This paper analyses the feasibility of using DDS over TSN considering a micro-launcher representative use case, including the trade-off analysis of different DDS implementations, the design of a DDS based on-board software (OBSW) for launchers and the results obtained in an Avionic Test Bench (ATB) for closed loop tests.

***Index Terms*—DDS, TSN. Avionic, micro-launcher, Ethernet, FPGA, RTEMS, SafeDDS**

I. Introduction

The avionic system in a launcher can be understood as a data centric distributed system in which different entities are sharing data to control the subsystems in the launcher and to provide visibility to ground regarding the health status or the performances. The communication between the entities is critical, if the data is lost or the data is not received in time the consequences can be up to catastrophic. The absence of established standards leads to increased development time and higher costs, as additional efforts are required to ensure system compatibility and integration.

The Object Management Group (OMG) [1] created the Data Distribution Service for Real-Time Systems (DDS) [2], the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems. Additionally, the OMG has recently published a specification: "DDS Extensions for Time Sensitive Networking" to standardize the usage of DDS over TSN networks.

The availability of a standard middleware relying on a trustful communication bus such as TSN enables the decoupling between data sources and data subscribers such as sensors, actuators, guidance, navigation and control algorithms, pressure or temperature control logic. Each entity only interfaces with the middleware and this interface is open with the abovementioned standards.

The DDS allows reducing the coupling in the communication between different elements, improving the productivity, interoperability, and the scalability of the systems. DDS is a very popular technology in robotics, i.e., it is included in the widely used Robotic Operating System (ROS) [6], it is also very popular in automotive, e.g. in the AUTOSAR [7] framework and there are already some space use cases such as the core Flight System cFS from NASA [8].

DDS is designed to facilitate real-time data exchange across distributed systems. It enables applications to communicate with minimal latency while ensuring scalability and high reliability, making it particularly suitable for mission-critical domains such as aerospace, defense, healthcare, industrial automation, and automotive industries.

Time Sensitive Networking (TSN) is a collection of standards developed by the TSN Task Group of the IEEE 802.1 Work Group. Their purpose is to enable deterministic, highly reliable communications on standard Ethernet.

TSN is more and more being used in launchers. This paper analyzes usage of DDS over TSN for launchers to reduce the development time and maximize the re-use maintaining the determinism and the reliability required.

II. DDS & Functional Safety

DDS is designed to facilitate real-time data exchange across distributed systems. It enables applications to communicate with minimal latency while ensuring scalability and high reliability, making it particularly suitable for mission-critical domains such as aerospace, defense, healthcare, industrial automation, and automotive industries.

At its foundation, DDS operates on a publish-subscribe communication model, wherein data producers (publishers) disseminate information to data consumers (subscribers) without requiring direct knowledge of each other's presence. This architectural decoupling enhances system flexibility and scalability. Furthermore, DDS provides fine-grained control over data transmission through Quality of Service (QoS) policies, which regulate parameters such as reliability, latency, and message ordering to meet specific application requirements.

A key characteristic of DDS is its capability for automatic discovery of data producers and consumers, ensuring efficient data routing in dynamic and evolving system architectures. This feature is relevant for launchers as the elements on the DDS network vary depending on the mission phase e.g,. during interstage separation some elements are disconnected from the network. Additionally, DDS supports efficient, scalable, and

fault-tolerant data exchange, accommodating both point-to-point and many-to-many communication patterns. Its support for complex data types and real-time constraints makes it a preferred choice for applications demanding deterministic and reliable data dissemination.

III. DDS IMPLEMENTATION TRADE-OFF FOR LAUNCHERS

Several DDS solutions are on the market, some of them open-source and some others with proprietary licensing. The trade-off analysis of the different solutions shall consider several factors like the programming language, dynamic memory usage, code size, external dependencies, portability, abstraction on operating system dependencies, licensing policy, effort towards qualification, ...

The analysis phase was conducted using the previous criteria and considering the most used solutions in the market as:

- Fast DDS [9]: open-source and widely adopted specially in robotics, chosen as the default middleware supported by the Robot Operating System 2 (ROS 2) [6]. With many features and an extensive code base, it is designed to be versatile and targets general-purpose hardware.
- Cyclone DDS [10] similar solution as Fast DDS, in this case written in C, and also extensively used in robotics systems.
- RTIConnext [11], extensively used as previous solutions, in this case this is a proprietary license with some evaluation versions and it is intended for both general-purpose hardware and embedded systems.
- EmbeddedRTPS [12] is an open-source solution of the Real-Time Publish-Subscribe Protocol (RTPS) [13], the wire protocol normally used by DDS systems.
- Safe DDS [5] a licensed DDS solution designed for embedded systems, written in C++, and designed in a modular way that allows to configure it to be used in different platforms and operating systems.

After all these analyses, it was decided to use Safe DDS to evaluate the feasibility of using DDS over TSN, due to its maturity level, specially coming from the experience of developing a widely used solution as Fast DDS, its design that facilitates the integration with different applications and the modular design that helps to adapt the software to TSN usage.

IV. SAFE DDS

Among the various DDS implementations, eProsima Safe DDS [5] has been specifically developed to address the needs of safety-critical applications. As an implementation of the Data Distribution Service (DDS), Safe DDS is ISO 26262 [4] ASIL D certified, conforming to one of the most rigorous functional safety standards. Its primary objective is to facilitate reliable, deterministic, and high-performance communication in environments where operational failure is not permissible.

In contrast to conventional DDS implementations, Safe DDS prioritizes certification readiness, making it particularly well-suited for applications in aerospace, automotive, industrial automation, and healthcare. It is structured upon a robust architectural framework that emphasizes fault tolerance, redundancy, and strict adherence to real-time constraints. Moreover, Safe DDS seamlessly integrates with eProsima Fast DDS, an extensively utilized open-source DDS implementation, thereby enabling developers to transition between non-safety-critical and safety-critical systems with minimal reconfiguration.

Safe DDS is characterized by its modular architecture, which allows for portability across diverse platforms, compatibility with transport protocols such as Time-Sensitive Networking (TSN), and extensive configurability to meet specific application demands. This design ensures adaptability, enabling Safe DDS to conform to the stringent requirements of safety-critical environments while maintaining compliance with the highest industry standards.

V. DDS OVER TSN

By using the new standard, TSN adds the low latency, deterministic communication and real-time operations to the system network, all these capabilities are used by the DDS Quality of Service policies to determine different types of traffic inside of the system depending on their criticality level.

Both standards incorporate the modularity and interoperability features so that new, both software and hardware, modules can be added to the full system reducing the development, maintenance and integration costs, e.g. reusing modules between projects that have been previously developed, unitary tested and with a clear interface defined that helps during the integration process in the new system where it will be used.

The DDS over TSN standard adds new software entities to the DDS to handle the network connections, where the DDS writer and reader streams are linked to TSN Talkers and Listeners respectively that will manage the streams between the different participant entities. These entities manage the low-level layer of this new standard and allow the application developers to manage the different characteristic of each part of the system like the Quality of Service policies.

DDS over TSN standard defines a platform-independent DDS-TSN Configuration Model ensuring compatibility across various different TSN implementations. This model allows the definition of the nodes inside the application and the deployment scenarios matching the DDS Application with the Nodes inside. It is also used to define TSN Talkers and Listeners, TSN SW entities that manage the physical send and reception, resources for streams management.

VI. TESTING ENVIRONMENT

The DDS middleware will be integrated with a well-established TSN implementation, which has undergone extensive validation through a series of incremental qualification tests in both emulated and real-world environments [14].

Thus, the DDS over TSN will be directly validated on a realistic environment according to the considered use case: a micro-launcher architecture.

The Avionics Test Bench (ATB) consists of a series of hardware and software modules that creates a multi-unit avionics system architecture interconnected through a TSN daisy

chain topology network, replicating a real-case scenario. Other peripherals, both data producers and consumers as generic sensors (temperature, pressure...), Inertial Measurement Unit (IMU), Thrust Vector Control (TVC), etc. are also integrated via different protocols (CANbus, UART...).

This architecture will be sufficient to test the DDS network and evaluate TSN capabilities in terms of QoS, ensuring that each system element receives the right information at the right time.

An incremental test approach will be followed, going from a single hardware unit implementing pure software capabilities to a full architecture with all the units, data producers and consumers integrated, performing a full-chain validation.

VII. Results

A. Experimental results

For the test campaign, a C wrapper library has been created, offering a simple API for use in embedded applications. This library contains the compilation of the C++ SafeDDS library, adding a series of modifications to its base code and overriding some classes in order to avoid the use of dynamic memory and add the capability of communications via TSN using ethernet frames where DDS data can be transmitted according to the serialization standard.

For the development of this library, both Linux, using GCC version 11.4.0, to enable simpler development and the execution of both unit and integration tests more easily, and RTEMS, to demonstrate the applicability of this library and different use cases on embedded systems used in space environments and more specifically in launchers, have been used. The embedded hardware used is Zynq 7000 with its ARM Cortex-A9 processor, in this case in a simple single-core configuration.

The developed API allows an embedded software application to use the union of both standards, both DDS and TSN over DDS, in a simple and dynamic way where each node that wants to join a DDS network can publish or subscribe to different topics during execution in order to have better control of the consumption of both memory and CPU of the system.

To avoid the use of standard dynamic memory that is, calls to functions such as calloc(), malloc(), realloc(), etc., a mechanism has been developed by which all the memory that the DDS-TSN library needs is being reserved in a memory zone statically reserved prior to the initial call of the library so that in this way it is the user who has the control of how much memory is going to be available for the library.

B. Functionality validation

A phase of continuous integration and continuous development was carried out, thus testing both at the unit level and in a more integrated manner the new functionalities that were being added to the library since it allows a high number of configurable components. Once there was a prototype that could be executed, it was verified that it worked both in the Linux simulation environment and in the real environment of the embedded hardware with RTEMS 5.3. With these initialization tests it was verified at the unit level that the library could be executed in a real environment without problems of memory and basic CPU usage. Little by little, the functionality provided by the library and the use case was gradually increased in order to test all the main functionalities defined in both standards and obtain a prototype where there are several remote instances connected to the same DDS network where each node is able to publish and subscribe to both local and remote topics, and this information is exchanged between the nodes and SW modules that need it. When there was a prototype where the main functionalities of both standards for the use of DDS over TSN were tested and validated, performance tests were carried out where the traffic was increased both in the size of the topics and in the publication rate of these topics to verify that the system continued working without performance problems and fulfilling the real-time requirements defined for each instance. Following the incremental approach, HW-in-the-loop testing is first implemented in a point-to-point architecture using just a two nodes configuration.

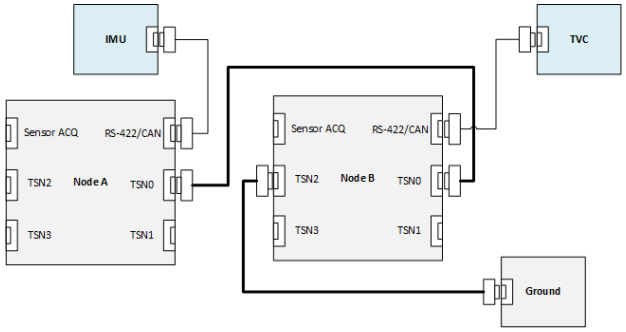

Fig. 1. First HW-in-the-loop test with two nodes configuration.

After successful validation of a basic setup, the configuration of the final test environment is implemented, which consists of 4 nodes in a TSN daisy chain network. All nodes send TSN traffic to the network with different periods, sizes, and priorities to verify that the DDS network does not experience delays or collisions with other packets.

In order to test the library and show the following results, two nodes are participating in the DDS network. Three topics are defined with different periodicities of publishing their data, from 10ms to 30ms, which simulate a real traffic and real time needs in real embedded systems.

TABLE I
DDS TOPICS AND PUBLISH/SUBSCRIBE OPERATION PER NODE

Node	Topic	Operation	Frequency
Node A	IMU_TM	Publish	100 Hz
Node A	TVC_TC	Publish	33 Hz
Node A	IMU_TM	Subscribe	100 Hz
Node A	TVC_TM	Subscribe	33 Hz
Node B	TVC_TC	Subscribe	33 Hz
Node B	TVC_TM	Publish	33 Hz

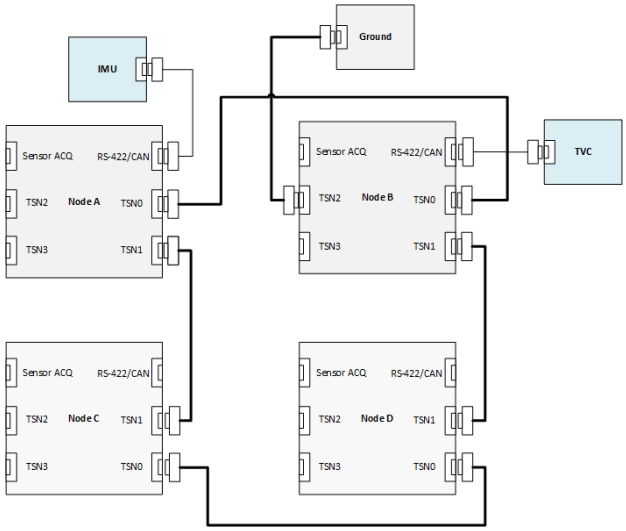

Fig. 2. ATB complete test environment architecture with 4 nodes in TSN daisy chain network.ATB complete test environment architecture with 4 nodes in TSN daisy chain network.

C. Comparative analysis

The inclusion of this library in embedded software with high load both on the CPU and TSN network bandwidth does not cause any degradation in overall system performance, nor does it introduce any delays or collisions with other packets on the same network.

The TSN bandwidth is managed by the IPCore developed in a previous activity, which allows traffic to be managed in a simple and transparent way for the application software.

Regarding the pure behaviour of the software, no behaviour has been detected that would delay other system tasks, since the execution cycle of the complete library is very fast, with an average response time of 0.3 ms which represent a 3% of CPU load in a 100Hz cycle in our analysis case, as observed in the results previously reported for the Zynq 7000 + RTEMS 5.3 system.

Long CPU cycles are only observed during the system initialization phase, when all instances are enabled and the maximum memory required throughout program execution is reserved. Anyway, this is not an issue because the library allows to define a timeout to the processing task cycle to stop processing during this cycle and queue the remaining task to be executed in the following cycles.

TABLE II
CPU TIME STATISTICS (IN MS) FOR 120 S EXECUTION

Statistic	Node A	Node B
Maximum	2.089	1.883
Mean	0.277	0.185

The memory used by the DDS library was measured, results are shown in table III. The memory overhead is affordable for current space systems. For reference, in the select use case it represents an increment below 5% in the memory used.

TABLE III
LIBRARY SIZE (IN BYTES) FOR DDS OVER TSN IMPLEMENTATION

Case	text	data	bss	dec
RTEMS	484451	68	4843	489362

Regarding the interaction with the TSN network, only DDS will send periodic data, configurable from the application through the C API, for the handshake between nodes and to verify that the network maintains the same configuration in case any node has disconnected or newly joined. In addition, it will send a TSN message to each remote node that needs the data of a new sample published in some topic; therefore, this behaviour does not add significant overhead to the network, as it depends entirely on the data update period defined by the application.

In addition, through the use of the DDS standard, we add a series of quality of service policies to the application, allowing the selection of message priorities according to the topic being written to by configuring the TSN VLAN and priority of the message, to receive alerts if a new sample takes longer than expected to be received, to define how many of the latest samples to store, to monitor the instances active in the network, and to send historical data to new nodes that join the network, and so on.

VIII. CONCLUSIONS

This study presents the design, implementation, and evaluation of a library for embedded software, enabling interoperability between the DDS and TSN standards. The library has been validated in a realistic case study, where it is verified that the functionality is correct and efficient for space environments, and more specifically, for launch vehicles. With the tests campaign carried out, it is demonstrated that, with minimal overhead in terms of memory, CPU usage, and TSN traffic, multiple functionalities implemented by the DDS standard can be added. DDS provides useful capabilities for modular and reusable application software, thanks to the ability to communicate between local nodes, software modules, and remote nodes connected to the same TSN network. Integration and test campaign has proven seamless, demonstrating that DDS over TSN is a robust and viable control and communication backbone for launcher avionics systems. The combination of both standards ensures end-to-end deterministic and reliable data handling by combining TSN's time-aware scheduling, frame preemtpion and VLAN tagging with DDS's QoS policies while maintaining the modularity and scalability required for avionics and launchers architectures. The results indicate that DDS over TSN is not only a viable solution for launcher data handling subsystems but also a reliable framework for future avionics platforms requiring predictable timing, scalable integration, resource-efficient and high-reliability.

IX. Acknowledgement

The activities were carried out under a programme of and funded by the European Space Agency through the Future Launchers Preparatory Programme (FLPP). The authors would like to thank the European Space Agency – ESA for the financial and the technical support. They also extend their appreciation to Sirius Space Services for endorsing the concept during its presentation to the ESA FIRST! Initiative.

The activities were carried out under a programme of and funded by the European Space Agency through the Future Launchers Preparatory Programme (FLPP).

To be completed for final paper (no names of people or companies can be included in the digest).

References

[1] (2025, March) Object Management Group website. [Online]. Available: https://www.omg.org

[2] (2025, March) OMG DDS Standard website. [Online]. Available: https://www.omg.org/spec/DDS/1.4/About-DDS

[3] (2025, March) DDS Extensions for Time Sensitive Networking. [Online]. Available: https://www.omg.org/spec/DDS-TSN/1.0/Beta2/About-DDS-TSN

[4] (2025, March) ISO 26262 Standard website. [Online]. Available: https://www.iso.org/standard/68383.html

[5] (2025, March) eProsima Safe DDS website. [Online]. Available: https://www.eprosima.com/middleware/safe-dds

[6] (2025, March) Robotic Operating System (ROS) website. [Online]. Available: https://www.ros.org

[7] (2025, March) AUTomotive Open System ARchitecture (AUTOSAR) website. [Online]. Available: https://www.autosar.org

[8] (2025, March) core Flight System by NASA website. [Online]. Available: https://cfs.gsfc.nasa.gov/index.html

[9] (2025, March) Fast DDS website. [Online]. Available: https://fast-dds.docs.eprosima.com/en/latest

[10] (2025, March) Cyclone DDS website. [Online]. Available: https://cyclonedds.io

[11] (2025, March) RTI Connext professional website. [Online]. Available: https://www.rti.com/products/connext-professional

[12] (2025, March) EmbeddedRTPS repository website. [Online]. Available: https://github.com/embedded-software-laboratory/embeddedRTPS

[13] (2025, March) OMG DDS Interoperability Wire Protocol website. [Online]. Available: https://www.omg.org/spec/DDSI-RTPS/2.5/About-DDSI-RTPS

[14] C. Dominguez et al, "Time Sensitive Networking (TSN) as reliable communications bus for micro-launchers," 2023 European Data Handling & Data Processing Conference (EDHPC), Juan Les Pins, France, 2023, pp. 1-5, doi: 10.23919/EDHPC59100.2023.10396540.