TECHNICAL INSIGHT

Bridging CAN and DDS for High-Performance Distributed Systems

Integrating CAN signals into a DDS data space enables real-time embedded systems to seamlessly connect with scalable distributed architectures, unlocking new possibilities in automotive, industrial automation, and autonomous systems. For today's complex systems, a well-designed CAN-DDS bridge can deliver the best of both worlds: the device-level reliability of CAN plus the system-level scalability of DDS.

In modern applications, communication between distinct systems and networks is crucial. **Controller Area Network (CAN)** and **Data Distribution Service (DDS®)** are two widely-used communication solutions serving different purposes. CAN is a lower-level protocol dominant in real-time embedded systems such as automotive and industrial automation. For wider communication requirements, DDS provides a middleware architecture that supports scalable, high-performance, and distributed systems.

Integrating CAN and DDS allows for seamless communication between real-time embedded devices and large-scale distributed systems. This technical insight explores the fundamentals of CAN and DDS, why bridging them is beneficial, and how to design a DDS system to efficiently transfer CAN signals from the physical and data link layers of communication to anywhere in the system and up to the cloud.

Understanding CAN and DDS

Controller Area Network (CAN)

CAN is a message-based protocol designed for real-time communication between embedded devices. It is widely used in critical systems due to its **high reliability, deterministic communication, and low-cost implementation.**

Key Features of CAN:

- Low latency & real-time performance
- Error detection & fault tolerance
- Multi-master, priority-based arbitration
- Limited data payload (8 bytes per frame in CAN, 64 bytes in CAN FD, 2048 bytes in CAN XL)
- Seamless integration in existing technology stacks such as AUTOSAR

Data Distribution Service

DDS is an open standard for data-centric communication. With DDS, the data itself is the interface between system components. It uses a data-centric publish-subscribe or request-reply pattern, with automatic discovery, independent of the underlying transport (network, memory, radio, serial), and granular security that enables it to operate in the harshest of environments. DDS offers extremely high performance using efficient binary data encoding (RTPS), and is able to take advantage of the underlying transport capabilities (i.e., multicast, zero-copy, lossless compression, etc.) while maintaining code portability.

Key Features of DDS:

- High scalability to support both local and wide-area networks (WANs)
- Quality of Service (QoS) policies to control data flow
- Dynamic decentralized communication
- Large message support
- Platform support for dozens of processor architectures, operating systems, and compilers

Combining DDS with CAN for Robust, Scalable Communication

As industries move towards smart automation, integrating embedded CAN networks with scalable DDS-based distributed systems becomes essential for the following reasons:

1. Real-Time Embedded-to-Cloud Connectivity

Though extremely fast, CAN's limited range may restrict its use to localized networks. To capitalize on today's evolving use cases, DDS supports everything from small ECUs and high performance CPUs to WANs and cloud connectivity, enabling remote monitoring and control of CAN-based systems.

A compelling advantage of CAN-DDS integration is allowing real-time CAN devices (e.g., sensors, actuators) to communicate with cloud-based applications, analytics platforms, or enterprise systems via DDS.

2. System Interoperability

Modern vehicles already feature numerous CAN buses for various subsystems (engine control, braking, body electronics, etc.). Although those systems are slowly migrating to Ethernet, they are unlikely to change in the short term. Therefore, there is a need for integration with newer systems that are compatible with higher-level communication solutions.

Integrating legacy subsystems with a broader vehicle network using Ethernet or cloud-based services requires a bridge to a more scalable and flexible technology such as DDS. Moreover, in heterogeneous systems that span multiple hardware platforms, operating systems, and programming languages, it is crucial to employ a connectivity framework that provides broad platform support and ensures seamless communication across all components.

3. Scalability and Remote Monitoring

As systems grow in complexity, DDS provides a more scalable and adaptable architecture that can accommodate future expansion, while extending the flexibility of legacy CAN devices.

With DDS, CAN devices can be monitored and controlled remotely over **Ethernet, Wi-Fi, or 5G**, enabling scalable deployment across autonomous systems that run factories, advanced medical equipment, smart cities, transportation systems, and more.

4. Enabling Data Fusion

As markets move toward software-defined architectures, it is critical to transition from a signal-oriented data model to a data-centric model that captures relationships, context, and semantics across the entire system. This enables the necessary scalability and consistent data integration as systems grow in complexity. The data-centric approach combines data from multiple CAN buses with other sensor data or external sources.

DDS allows CAN data to be **aggregated, processed,** and **analyzed** in real-time, enabling advanced functionalities such as predictive maintenance, autonomous driving, and remote diagnostics. DDS facilitates this data fusion by providing a unified data space where signals are combined based on the data needed by the software.

How to Integrate CAN and DDS

The best way to integrate CAN and DDS is to use a **CAN-DDS gateway**, for translating the transferred information between the two protocols. There are different aspects to consider when implementing such a gateway:

1. Hardware & Software Requirements

- A **CAN interface** (e.g., MCP2515, Peak CAN, or SocketCAN-compatible device)
- A **DDS middleware** (e.g., RTI Connext®)
- A **microcontroller or embedded system** (e.g., IFX TC3, STM32, NXP S32G, NVIDIA Jetson, Raspberry Pi or an industrial PC)

2. Message Translation

The bridge should map CAN messages to DDS samples and vice versa. However, CAN is a low-level protocol with a very limited throughput, while DDS is a high-level middleware able to achieve much higher throughputs. This is reflected on the maximum payloads and typical overheads per message, as shown in the table below.

Protocol	Max Payload (Bytes)	Typical Overhead (Bytes)
Classic CAN	8 Bytes	**~8 Bytes**
CAN FD	64 Bytes	**~9 Bytes**
CAN XL	2,048 Bytes	**~19 Bytes**
DDS (RTPS over UDP/IPv4)	65,507 Bytes*	**~72–80 Bytes**

* DDS can handle massive payloads (2GB) through built-in fragmentation, but each fragment is limited by the underlying transport protocol (typically UDP).

- CAN has a limited payload size, with a relatively small overhead. DDS payloads are substantially larger, but each message has larger overhead.

Therefore, multiple CAN messages may map into a single DDS sample, and bit-packing compression techniques used in CAN are unnecessary in DDS.

3. QoS and Timing Considerations

- CAN is **real-time but limited in bandwidth**, so DDS messages should be optimized for efficiency. DDS offers a rich set of QoS policies, such as deadline, reliability, and liveliness, which can be configured to achieve a similar communication robustness as CAN, but avoids sending unchanged periodic data.
- **Prioritization and mapping of CAN IDs to DDS topics** ensures real-time critical messages are handled correctly. Developers can use flow controllers or other QoS to map high-priority topics in DDS to high-priority CAN signals.
- **Latency constraints** should be considered, especially for control applications. QoS settings such as deadline or lifespan in DDS could help enforce and monitor different latency requirements. In CAN it is usually solved by a combination of applying a higher priority and also applying a faster sending cycle to ensure that the data will arrive on time.

4. Implementing the Bridge

This section proposes different approaches to solve the interconnection between CAN and DDS.

- **Direct Signal to Topic Mapping:** This implementation is very easy to apply in legacy systems, but will not scale.
- **Simple Domain-Oriented Approach:** This implementation has better scalability, but still doesn't leverage all the advantages of DDS to achieve good performance.
- **Data-Centric Architecture:** This implementation offers good scalability and takes advantage of DDS features to improve overall performance.

Direct Signal to Topic Mapping

A direct approach would be using a middleware application that reads from a CAN bus and republishes data to DDS and vice versa without any additional processing. This implies mapping each unique CAN message to a unique DDS sample. Although this is an easy approach, it is not scalable as illustrated with the following example:

Example implementation flow:

- CAN to DDS:
  - CAN messages are received by the bridge over the CAN channel.
  - The message is parsed and translated into a DDS topic.
  - The DDS topic is sent over the DDS databus.
- DDS to CAN:
  - DDS samples are received by the bridge over the DDS databus.
  - The sample is translated into a CAN message.
  - The CAN message is sent over the CAN channel.

Although this is an easy approach, it is **not scalable** for several reasons:

- **Network Constraints:** Ethernet throughput generally improves as packet size increases and similarly as the ratio of packet overhead to payload decreases. In the case of very small CAN messages (CAN: 8 bytes or CAN-FD: 64 bytes), the overhead from various headers (IP/UDP/DDS) can result in packet overhead that is nearly 75%. The effective throughput of a network that is only 25% efficient is not scalable for high-performance applications.
- **CPU Constraints:** Like the network constraints, a high rate of very small packets requires significant CPU processing, and it is common for the processor to become the throughput bottleneck.
- **Memory Constraints:** Every DDS DataWriter and DataReader consumes memory, so creating potentially more than one hundred of these entities isn't possible on many low-power and low-cost ECUs.

Therefore, this approach would only make sense when the architect doesn't want to use advanced DDS features (such as filtering) and are using DDS just to tunnel a few CAN messages between two different parts of a system that will produce and consume CAN signals. In any case, this should be considered a bad practice in general, and the Simple Domain-Oriented Implementation should be considered instead when legacy parts of the system are to be kept.

Simple Domain-Oriented Approach

A more efficient alternative is to encapsulate several CAN frames in the same DDS sample, for instance based on a common characteristic such as their sample rate. A different example would be to combine domain-oriented data into a single DDS message.

Although this is more efficient than directly mapping each CAN message to a DDS sample, it lacks the **data-centric architecture** required to maximize DDS performance. Some of those signals may be sensor/actuator signals that change at a fairly high rate, and other signals may be status signals that never change for the life of the application. However, the entire sample, including many signals that are not changing, must be sent at the sensor/actuator rate. Similarly, although the sensor/actuator data doesn't require reliable communication, reliable communication must be used to support the status data, creating an unnecessary overhead for periodic data.

As noted in these two examples, indiscriminately combining signals into a single sample often results in inefficient DDS communication, since the aggregate QoS requirements for the entire sample rarely match the efficiency of the QoS configuration for individual components.

Compared to the previous approach, this strategy would allow for better interconnection of multiple parts of the system, tunneling CAN messages between different sub-systems that still produce and consume CAN signals. This is a good approach if the system still has many legacy applications that don't need to be updated and are designed to work with CAN signals.

Data-Centric Architecture

Although a simpler domain-oriented implementation is possible and might seem attractive (as explained above), that option doesn't offer the scalability required. The recommended implementation is a data-centric architecture, which enables users to leverage their understanding of the data content in CAN and apply a data-centric DDS design that defines appropriate datatypes and topics.

This involves intentionally categorizing signals into dataflow patterns such as periodic, status, and command/event. Each pattern requires distinct QoS configurations, and by isolating them, developers can optimize QoS for the specific data flow. While not strictly required for all messages, a data-centric architecture approach is particularly effective in zonal architectures, where each ECU exposes a large set of signals that can be organized into different data flow patterns for optimal efficiency.

For legacy systems, applications can be refactored using DDS to improve performance, scalability, and modularity by restructuring code without altering external behavior. Moreover, adopting a Data-Centric Architecture approach to bridge CAN and DDS is the best option to follow for newly developed systems that don't carry the burden of legacy applications.

5. Conclusion

The combination of CAN and DDS provides a robust foundation for building resilient, scalable, and interconnected systems that connect from edge to cloud, enabling engineers to create innovative applications that meet the demands of today's complex systems.

However, choosing the right approach to combine those two technologies is key to achieving a performant and scalable solution. RTI Professional Services provides tailored architectural and system design services to integrate CAN into a DDS data space.

For more information, please visit rti.com/automotive or contact your RTI representative.