Journal of Innovation
February 2025 | 26th Edition

The Role of Data Centricity in Smart, Connected Systems

Author:

Andy Krassowski
Real-Time Innovations (RTI)
andyk@rti.com

CONTENTS

FIGURES

TABLES

This article discusses how data centricity, enabled by core connectivity standards, enables real-time communications for advanced AI and autonomous systems.

Data-centricity is the underlying mechanism for developing a data-centric architecture. This approach emphasizes the central role of data in designing, implementing, and managing distributed systems. While there are a few different architectural approaches that use data centricity, this paper will focus on the Object Management Group® (OMG®) Data Distribution Service™ (DDS) protocol. DDS is widely used in complex AI and Machine Learning (ML) use cases, providing the communication layer that keeps data as the focal point, thereby enhancing system flexibility, scalability, reliability, data interoperability, and digital connectivity for next-generation systems.

Readers will learn how DDS decouples data from the application, which enables real-time, scalable data exchange for complex, high-performance systems. This paper concludes with an overview of how data centricity works in three industry use cases.

1 CONNECTIVITY CHALLENGES IN SMART, CONNECTED SYSTEMS

Next-generation AI and ML applications are beginning to deliver transformative value through intelligent decision-making, automation, and optimization. These advanced applications require rapid, reliable data exchange across a complex interconnected system of devices and subsystems to achieve stringent performance and safety requirements. Examples include surgical robotics, software-defined vehicles, autonomous systems, and military defense systems, where unreliable data flow could have severe consequences. In addition to low latency, the data must be protected against unauthorized access and be able to exchange information across systems, regardless of platform. At their core, smart applications require access to data - lots of data - as well as reliable connectivity to process that data rapidly, robustly, and at scale.

The complexity and scale of this effort introduces data connectivity challenges for developers of AI-enabled applications. To start with, the data in AI/ML subsystems can reside in various locations, including edge devices, centralized cloud platforms, and/or in central compute locations. Data exchange between these subsystems can occur internally within a system, across different operational technology (OT) systems, or between IT and OT systems (e.g., analytics platforms). While all smart applications face challenges such as data integration, latency, and security, there are additional interoperability requirements when bridging IT and OT systems, due to differing transport protocols, hardware architectures and programming languages. A data-centric architecture can provide the framework to ensure cohesion which bridges these divides.

For applications where data delays pose safety risks, such as self-driving cars, it’s critical to ensure low-latency, high-throughput data exchange to meet demand for processing data in real time. Modern systems require seamless communication between multiple nodes, which can become problematic in environments with unreliable or high-latency networks. Scalability is another concern, as the addition of devices, sensors, or applications often introduces bottlenecks or communication breakdowns, especially when legacy systems are not designed to handle dynamic growth effectively.

System interoperability is also a concern. Smart applications often involve integrating hardware and software from different vendors, each using proprietary protocols or standards. It's an ongoing challenge to ensure that all components communicate efficiently without extensive custom coding, which can slow down progress and future expansion. Additionally, developers must address security risks associated with data transmission, such as unauthorized access, spoofing, or breaches. Finally, developers must contend with CPU, memory, and bandwidth limitations, especially in remote or resource-constrained environments, which can impede the timely flow of critical data that is required for generating and leveraging AI-driven insights.

Together, these challenges demand an architecture that is tailored to the demands of smart, interconnected applications. Here's one approach that is gaining traction: Rather than designing the system around the process, instead design the system around its data flow. This approach to putting data at the core of the architecture is called data centricity. Data-centric architectures are now enabling a new generation of AI systems, fueled by core connectivity standards.

2 DATA CENTRICITY: ENABLING SCALABILITY, DESIGN FLEXIBILITY, AND REAL-TIME DATA DELIVERY

Data-centric architectures are system designs that prioritize data as the central element, around which all components and processes are organized. Unlike traditional architectures that focus on applications or infrastructure, a data-centric architecture designs the system around the data to emphasize data creation, storage, accessibility, and governance. It leverages scalable data sources by unifying structured and unstructured data and employing advanced pipelines for seamless data integration and transformation. Interoperability is a key feature of any data-centric architecture, enabling systems to share and utilize data effectively across domains through APIs, standard protocols, and semantic layers. Governance frameworks ensure data quality, security, and compliance, while feedback loops facilitate the continuous improvement of processes and analytics. Crucially, this approach enables real-time insights, AI and ML workflows, and cross-system collaboration.

The basic concept of data centricity in connected applications is not new. In the 1980s for example, customer relationship management (CRM) software became popular, due to its ability to manage a "single source of truth" for each customer or prospect's data. Each transaction was captured and made accessible, and the storage/retrieval of such data was used for future workflows, such as reports, campaigns or even AI-assisted analysis. In other words, data was at the heart of the system; so technically, they were early data-centric systems. Today, we call this architecture client-server or message centric. The message-centric architecture is still popular today for predictable, stable and non-critical environments.

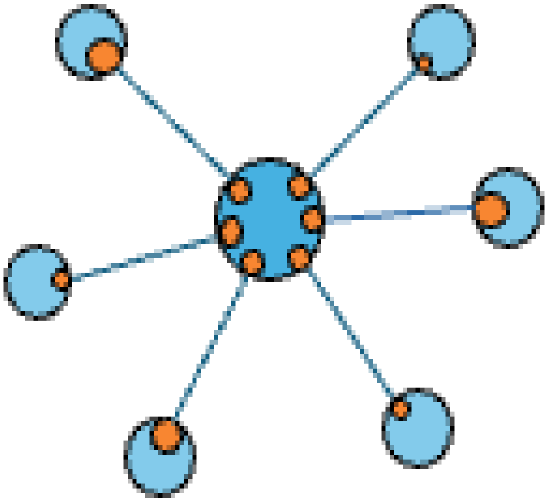

Figure 2-1: Message-centric architecture.

The outer blue circles each connect to a central hub and each contains a bit of communications logic shown in orange. Each added node complicates the architecture and impacts existing functionality. Message-centric architectures face challenges in systems that process much higher volumes of data at more rapid rates, while ensuring future flexibility to accommodate expansions, new technologies, and other unknown changes. Data centricity operates at a much more sophisticated level, capturing data in motion across subsystems to enable real-time communications for a new generation of AI-enabled systems. The data's structure is formally defined with enough precision for a tool to generate the code which serializes the data on and off the databus, such that it is interoperable across different nodes' CPU architectures.

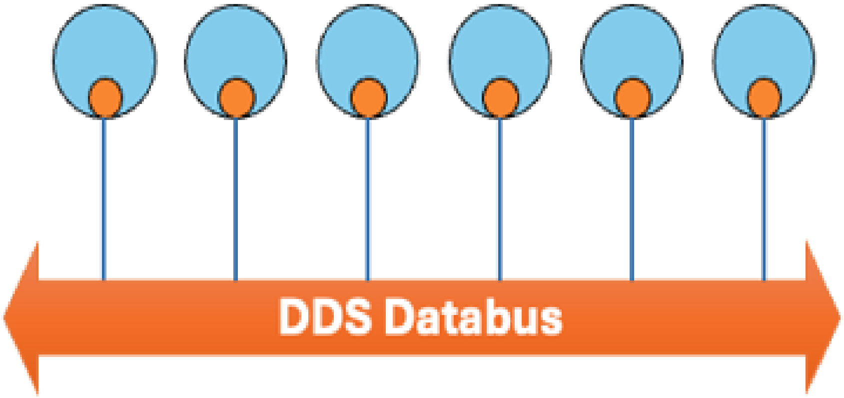

Figure 2-2: Data-centric architecture.

Here, the nodes can easily inter-communicate via an abstract databus. Adding nodes is a simple linear extension of the databus and has minimal impact on existing behavior.

In DDS-based systems, data centricity is an abstraction for a software databus that is focused on the actual data, not the mechanics of its distribution nor the details of its transmission. To construct a data-centric system, developers first analyze and categorize what data is needed and then group fields of that data into Topics – this defines the "what." Since the "how" and "when" are captured by a standards-based set of Quality of Service (QoS) parameters, they can be all but ignored early-on in the design process. The "who" portion of the design is implicit – one or more Publishers of each Topic is readily identified as producing the data, along with at least one Subscriber that consumes that data. Also implicit is the "where" – Publishers and Subscribers only need to be reachable by some abstract Transport, which can be Ethernet or wireless UDP, TCP, Shared Memory, or even the Internet (WAN).

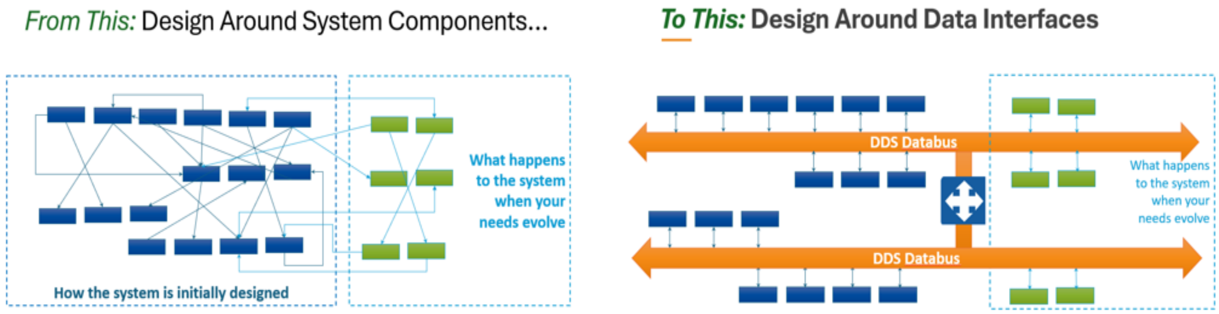

Figure 2-3: Data centricity designs the system around the data flow.
(Source: Real-Time Innovations (RTI), 2024)

Several important design benefits are achieved by adopting a data-centric DDS approach. Since Publishers and Subscribers can be developed independently, additional Subscribers with different QoS settings can be added as the system evolves. Coping with ever-improving CPUs and network hardware is enabled by the abstraction layer, which makes data-centric designs agnostic to transport and compute resources.

3 AN INTRODUCTION TO DDS

DDS is an OMG® open middleware standard that uses a Publish/Subscribe communication mode to facilitate real-time data exchange in distributed systems. Data producers (Publishers) and consumers (Subscribers) exchange information without needing a prior knowledge of each other.

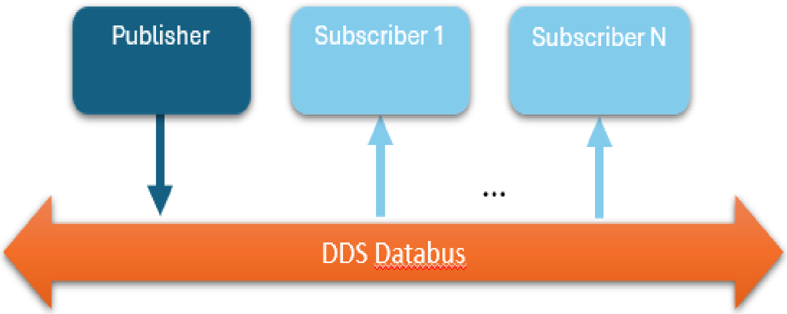

Figure 3-1: How data subscribers and publishers work in a data-centric DDS architecture.

DDS decouples data from the application, which enables the real-time, scalable data exchange within complex, high-performing systems. Unlike traditional client-server architectures, DDS provides direct access to structured data and fosters real-time decision-making. This makes DDS particularly well suited for industry applications where low and stable latency, high throughput, and deterministic performance are critical. Examples include unmanned aerial vehicles, connected surgical devices, intelligent robots, power generation & distribution, and industrial IoT. This approach eliminates the need for point-to-point connections, reducing communication overhead and enabling systems to scale dynamically.

Additionally, DDS includes advanced features such as multicast communication, which optimizes network bandwidth usage by delivering data to multiple Subscribers simultaneously. It also can easily implement redundancy, by arbitrating the gapless delivery of samples from multiple "hot" publishers. Delegating removes custom application logic to handle failover and failback, allowing data to get to the right place continually via DDS.

One of the defining advantages of DDS is its Quality of Service (QoS) abstraction. Setting QoS policies allows developers to fine-tune system behavior based on specific application requirements. Developers can configure data reliability, fault tolerance, and performance parameters, thereby ensuring that critical data is delivered on time and without loss, even under adverse network conditions.

QoS supports data-centric communication, where applications interact directly with structured data rather than relying on complex protocols or message parsing – this approach simplifies development, improves maintainability, and enhances real-time decision making. DDS also complies with other industry standards, fostering interoperability across platforms and devices, reducing vendor lock-in and slashing integration costs. Overall, DDS empowers businesses to build scalable, reliable, and future-ready systems in increasingly data-driven markets.

4 DDS: THE DATA-CENTRIC CONNECTIVITY STANDARD

DDS serves as the backbone to a data-centric architecture, ensuring that data remains the focal point in developing smart, autonomous systems. The data centricity enabled by DDS is achieved by introducing a Discovery process which works as follows: Nodes with a need to communicate will exchange their publisher/subscriber details at startup, then leverage those details to communicate via a highly-efficient, binary structure using the DDS Real-Time Publish Subscribe (DDS-RTPS) protocol. Inside the payload of that protocol is a Sample along with associated metadata, such as the send and receive timestamps, sequence number, length, retry statistics, failover details, and even coherency information.

Data centricity therefore provides capabilities that are impossible to achieve with opaque blob messaging systems. Through its QoS properties, the DDS framework can understand the complex filtered needs of multiple Subscribers. It can then deliver only the data that qualifies (samples that pass the filter) to each subscribing node. By not sending unqualified samples, it reduces application complexity, conserves network bandwidth, and saves Publisher CPU cycles.

Another data-centric design feature is the ability of DDS to handle extended datatypes. When a datatype is declared to be an extension of a base type, Subscribers can robustly handle new samples with fields unknown to them, without crashing or misinterpreting the new data. The new fields are simply ignored, and legacy behavior continues as before.

This is particularly useful for systems that require future-forward flexibility. Since evolving a complex system over time is commonplace and systems often need to be gradually updated in the field, the extended datatypes feature can radically improve an engineer's ability to work in real-world design and deployment scenarios.

The following table compares DDS to a Message-Centric (Opaque Messaging) approach.

Attribute	Message-Centric	Data-Centric
Byte efficiency	Poor: Each sample must be self-describing requiring redundant dataflow	Ideal: Only binary data needs to flow, as the format/meaning was already exchanged during Discovery
Coupling	Poor: Requires adding a subscriber, as well as recoding the publisher	Ideal: Just add a subscriber to the needed Topic(s)
Ease of Programming	Poor: Requires dealing with addressing, as well as filtering with bespoke application code	Ideal: The architecture enables the framework to handle addressing and filtering seamlessly

Table 4-1: Comparing DDS to a message-centric approach. (Source: Real-Time Innovations (RTI), 2024)

5 EXAMPLES

This section briefly describes a few key industry use cases that leverage DDS. In addition, the OMG® DDS Foundation maintains a broad library of use cases with additional detail.

5.1 SOFTWARE-DEFINED VEHICLES

Automotive manufacturers are turning to DDS to achieve robust, scalable, and real-time data exchange that is critical for the next generation of connected and autonomous vehicles. A DDS-enabled, data-centric Publish/Subscribe architecture facilitates seamless communication between sensors, control units, and actuators within the vehicle. Sample applications include Zonal architectures, High Performance Compute, Digital Cockpit, and Testing and Simulation. For example, DDS ensures that sensor data from LiDAR, cameras and radar is distributed in real-time to various subsystems like perception, decision-making, and motion control. Low-latency, high-throughput data points are rapidly transmitted, providing intelligent dataflow that results in safe vehicle navigation.

5.2 CONNECTED MEDICAL DEVICES

In complex connected medical devices such as surgical robotics, DDS provides the data-centric communication backbone for precise, coordinated movements between various subsystems, including robotic arms, control units, imaging systems, and monitoring devices. For example, DDS ensures that data from sensors that track position, pressure, and more can be immediately transmitted to the control system.

Forming a context from the enabled sensor fusion, invalid alarm conditions can be filtered out, allowing medical staff to respond only to properly correlated sensor conditions. In addition, the data-centric architecture simplifies the integration of various components, regardless of vendor or platform, enabling interoperability across imaging systems, robotic controllers, and other supporting systems. A scalable, fault-tolerant and secure DDS communication framework enhances the precision, reliability, and safety of surgical robotic systems, ultimately improving patient outcomes. Current deployed system vendors include GE Healthcare and Moon Surgical.

5.3 UAVS

In unmanned Defense systems, DDS provides the data-centric backbone for efficient, reliable, and secure operations across a distributed network of autonomous vehicles, drones, and ground systems. The Publish/Subscribe model enables seamless data exchange among sensors, control systems, decision-making algorithms, and mission planners.

For example, in a fleet of autonomous drones conducting reconnaissance, DDS ensures that sensor data is distributed in real time to processing units and other drones in the network. This rapid, intelligent approach allows for collaborative decision-making and rapid adaptation to dynamic battlefield conditions. DDS provides the platform-agnostic design with interoperability, allowing seamless integration of diverse systems including air, land, and sea-based vehicles, regardless of vendor or CPU architecture. DDS also incorporates robust security features, including encryption and authentication to protect sensitive military data from interception, false data injection, or other cyberattacks.

6 SUMMARY

Complex, AI-enabled systems rely heavily on intelligent dataflow. By changing the system architecture from process-oriented to data centric, systems can be scalable, flexible, and more secure by design. DDS is the middleware standard that supports data centricity, using a Publish/Subscribe protocol to reliably enable rapid data exchange throughout smart, interconnected systems.

7 ACKNOWLEDGEMENTS

- Return to *OMG Journal of Innovation landing page* for more articles and past editions.