DDS Spec Outfits Publish-Subscribe Technology for the GIG

Recently finalized, the OMG's Data-Distribution Service (DDS) for Real-Time Systems enables real-time, data-critical applications for the Global Information Grid.

Gerardo Pardo-Castellote, Chief Technology Officer
Real-Time Innovations

Easy access to information ranks as a central goal of the military's networked future. The DoD's vision, called the Global Information Grid (GIG), calls for connecting systems across the globe. These systems range from enterprise business servers to battlefield tactical systems. The challenge is to get the right data to the right place at the right time.

The enterprise portion of the GIG relies on commercial standards like Web services. However, Web services can't address the real-time requirements of a tactical system like fire-control. These real-time systems need targeted technology, and that technology must be specified by targeted standards.

Responding to those demands, the Object Management Group (OMG) finalized the Data-Distribution Service (DDS) for Real-Time Systems Specification in June 2004. The OMG is the standards body responsible for many modern software standards, including CORBA, UML and MDA. The new DDS standard directly addresses the communication needs of real-time systems via network middleware—the software fabric that allows computer programs to "talk" and easily exchange information over the network (Figure 1). DDS is the most significant middleware addition that OMG has made in recent years.

DDS Communications Model

DDS is well suited to address the needs of real-time applications. The DDS publish-subscribe architecture lets applications communicate simply by publishing the information they have and subscribing to the information they need. DDS handles automatic discovery, reliability and redundancy over otherwise unreliable networks.

Publish-subscribe is a powerful paradigm. But the real key to DDS's power is its ability to "flexibly-but-precisely" specify performance requirements between all the different parts of the system. DDS achieves this power through the pervasive use of Quality of Service (QoS) parameters. QoS parameters configure the system and establish "contracts" between publishers and subscribers that specify exactly how information should flow between those nodes. QoS contracts provide the performance predictability and resource control required by real-time systems, while preserving the modularity, scalability and robustness inherent to the anonymous publish-subscribe model.

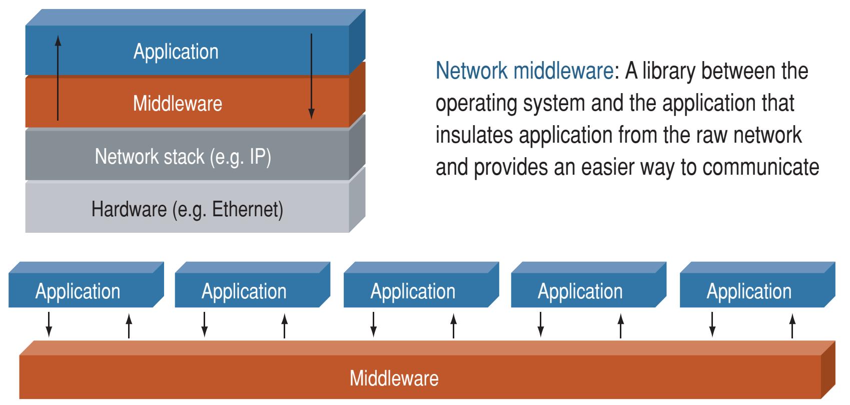
Application
Middleware
Network stack (e.g. IP)
Hardware (e.g. Ethernet)
Network middleware: A library between the operating system and the application that insulates application from the raw network and provides an easier way to communicate
Application
Application
Application
Application
Application
Middleware

Figure 1

The new DDS standard relies on network middleware—the software fabric that allows computer programs to "talk" and easily exchange information over the network.

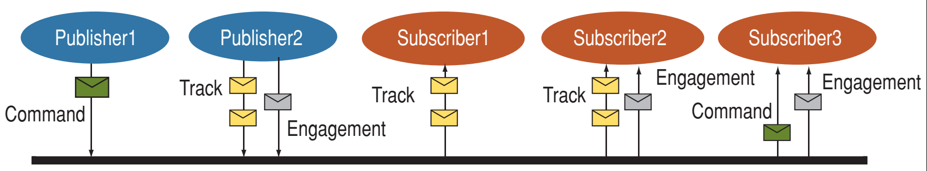

Figure 2

DDS uses Topics to identify the data-flows between publishers and subscribers.

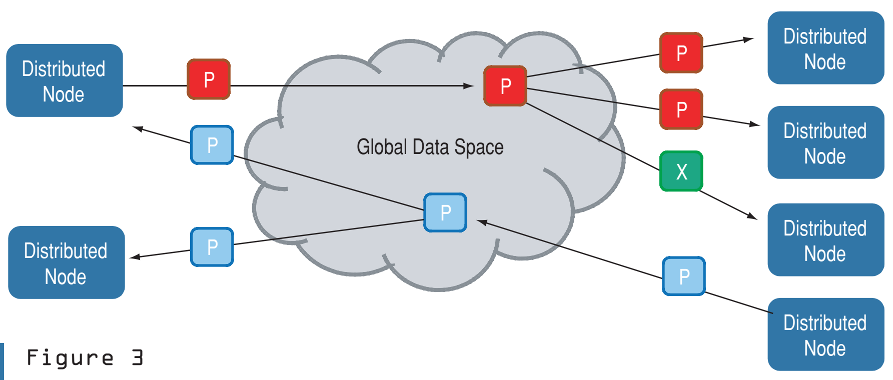

Figure 3

DDS creates the illusion of a Global Data Space where applications can read and write data.

DDS is well suited for heterogeneous networks, as it handles format conversion across operating systems, processor architectures and programming languages.

These characteristics make it ideally suited for the heterogeneous dynamic environments introduced by the increased use of wireless devices at the edge of the GIG.

Despite its novelty the technology is well proven. The DDS standard unifies some of the best practices present in successfully deployed real-time data-distribution middleware such as NDDS from Real-Time Innovations and SPLICE from Thales. Since its finalization DDS has gained broad adoption. It is now mandated for data distribution by the Navy Open Architecture Computer Environment (Navy OACE) and DISR and has already been adopted by programs such as FCS, DD(X), LCS and SSDS.

A Simpler Programming Model

DDS enables applications to use a much simpler programming model when dealing with distributed-data applications. Rather than developing custom event/messaging schemes or creating wrapper CORBA objects to access data remotely, the application can identify the data it wishes to read and write using a simple topic-name, and use a data-centric API to directly read and write the data (Figure 2).

The DDS publish-subscribe model connects anonymous information publishers (writers) with information subscribers (readers). A distributed application is composed of processes called "participants", each running possibly in a separate computer. A participant may simultaneously publish (write) and subscribe (read) data-flows identified by topic-name. This operational view can be pictured as a "data-flow bus." The data model means the developer

DoD Mandates DDS

In 2001, the Object Management Group (OMG) responded to industry's call for a publish-subscribe standard. The Data-Distribution Service (DDS) for Real-Time Systems Specification was adopted at the OMG in 2003 and finalized in 2004. DDS is the first publish-subscribe standard for real-time, data-critical systems.

In an effort to reduce costs and increase speed and flexibility of system procurement, the DoD has been pushing the industry to establish and use open architectures. The idea behind these efforts is to make it simpler and cheaper to integrate systems together by clearly defining the infrastructure software and electronics that "glue" the subsystems or systems together. One of the early efforts was the Navy Open Architecture out of PEO IWS and the Naval Surface Warfare Center in Dahlgren. They selected DDS as their publish-subscribe standard for moving data in real time.

In July of 2004, the Department of Defense Information Technology Standards Registry (DISR) selected DDS as its mandated publish-subscribe standard for the GIG. That same month, the DISR replaced the Joint Technical Architecture as the DoD's standards registry. DDS is now mandated across the DoD.

A more recent open architecture effort is being led by Boeing in their role as the Lead System Integrator for the Future Combat System program. They selected DDS as the key middleware technology for the System of Systems Common Operating Environment (SoSCOE). SoSCOE is the infrastructure through which all of the various systems and subsystems communicate. It is the key integration middleware in the FCS.

Other programs have stepped up to use DDS. These include the Single Integrated Air Picture; the Navy's Littoral Combat Ship, Aegis, E2C and Ship Self-Defense System; the Air Force's Joint Strike Fighter and AWACS; and the Army's Apache program. In addition, the Army's Weapon Systems Technical Architecture is integrating DDS into its solution. The DDS standard was created to fulfill an industry need for a standard interface for distributing data in real-time systems. Its quick adoption and wide use, especially in the DoD, is testimony to its importance in data-critical systems.

or system integrator can essentially ignore the complexity of the data flow and rely on each participant getting the data it needs from the bus.

Participants using DDS can read and write data efficiently and naturally. Underneath, the DDS middleware will distribute the data so that each reading participant can access the "most-current" values. In effect, the service creates a global "data space" that any participant can read and write (Figure 3). It also creates a name space to allow participants to find and share objects. In reality the data does not really "live" in any one computer. Rather it lives in the local caches of all the applications that have an interest in it. Here is where the publish-subscribe aspect becomes key.

DDS also provides a "state propagation" model. This model allows nodes to treat DDS-provided data structures like distributed shared-memory structures, with values efficiently updated only when they change. There are facilities to ensure coherent and ordered state updates.

Intuitive and Scalability

DDS is designed to automatically discover publishers and subscribers for each topic and autonomously establish data flows between them as permitted by the settings of the quality of service parameters. That makes it well suited for wireless environments where dynamic configuration changes are common. Unlike Jini, CORBA and other client-server technologies, DDS does not rely on centralized name servers and is therefore highly resilient to partial failures in the network.

At a fundamental level, DDS is designed to work over unreliable transports such as UDP, multicast or wireless networks. No facilities require central servers or special nodes. Efficient, direct, peer-to-peer communications, or even multicasting, are used to implement every part of the model.

Guaranteeing Predictable Operation

DDS allows for fine control over Quality of Service (QoS) on a "per-data-flow basis." Each publisher-subscriber pair can establish independent QoS agreements. This aspect, unique to DDS, enables application designs that easily support extremely complex, flexible data-flow requirements.

QoS parameters control virtually every aspect of the DDS model and the underlying communications mechanisms. Many QoS parameters are implemented as "contracts" between publishers and subscribers; publishers offer and subscribers request levels of service. The middleware is responsible for determining if the offer can satisfy the request, thereby establishing the communication or indicating an incompatibility error. Ensuring that participants meet the level-of-service contracts guarantees predictable operation necessary for real-time systems.

For example, periodic DataWriters can indicate the rate at which they can publish by offering guaranteed update deadlines via the "Deadline" QoSPolicy. By setting a deadline, a compliant DataWriter promises to send a new update at a minimum rate. Similarly, DataReaders can also specify a deadline for receiving the next sample update. If the deadline is missed, the middleware can notify a listener installed by the application. DataReaders may also request data at that or any slower rate via the TimeBasedFilter QosPolicy.

DataWriters may offer levels of reliability (via the Reliability QoSPolicy), parameterized by the number of past issues they can store to retry transmissions (via the History QoSPolicy). DataReaders may request differing levels of reliable delivery, ranging from fast-but-unreliable "best efforts" to highly reliable in-order delivery. This provides per-data-flow reliability control.

The middleware can also arbitrate between multiple DataWriters of the same Topic as specified by the Ownership and OwnershipStrength QosPolicies. Under exclusive ownership, each data-object can only be updated by a single DataWriter. DataReaders will only see the changes made by the active DataWriter with the highest ownership-strength. The built-in detection of DataWriter liveliness also provides automatic fail-over; if a strong DataWriter fails, all DataReaders immediately receive updates from the backup (weaker) DataWriter.

DDS: The Real-Time in Net-Centric

The need for data-distribution is fundamental to tactical military systems as well as other domains such as traffic monitoring and industrial sensing. Tactical systems at the edge of the GIG have complex data-flows—one-to-many, many-to-one, many-to-many—that require real-time performance, reliability, fault-tolerance, dynamic configuration changes and scalability. The GIG environment must reflect the speed and ever-changing nature of the real world.

The lack of established standards has led to the development of ad-hoc, custom, one-off solutions with sub-optimal performance and robustness that end up being very expensive to maintain. Commercial implementations of publish-subscribe data-distribution middleware have existed and have been deployed with success. However, each vendor provided a different API and slightly different semantics for equivalent concepts.

This situation was similar to the one prior to the adoption of UML. Successful modeling tools existed but widespread acceptance, availability of trained developers and broad knowledge exchange did not come until the UML standard consolidated terminology and notation. The recent adoption of the Data-Distribution Service (DDS) for Real-Time Systems Specification by the Object Management Group (OMG) is doing for data-distribution what UML did for modeling technologies.

Why DDS?

DDS is the first networking standard specifically targeting distributed real-time data-centric systems. It combines a simple data model, efficient distribution, extreme configurability and natural fault-tolerance. It holds the promise of a new generation of networking technology that will enable large, distributed, high-performance systems. More detailed information about the finalized Data-Distribution Service (DDS) for Real-Time Systems Specification can be found at www.omg.org/cgi-bin/doc?ptc/2004-03-07. *This article's author, Gerardo Pardo-Castellote, serves as Chairman of the DDS Standards Committee.*

Real-Time Innovations
Santa Clara, CA.
(408) 200-4700.
[www.rti.com].