The **Data Distribution Service** for real-time systems (**DDS**) is an Object Management Group (OMG) machine-to-machine (sometimes called middleware) standard that aims to enable scalable, real-time, dependable, high-performance and interoperable data exchanges using a publish–subscribe pattern. DDS addresses the needs of applications like financial trading, air-traffic control, smart grid management, and other big data applications. The standard is used in applications such as smartphone operating systems, transportation systems and vehicles, software-defined radio, and by healthcare providers. DDS was promoted for use in the Internet of things.

Architecture

Model

DDS is networking middleware that simplifies complex network programming. It implements a publish–subscribe pattern for sending and receiving data, events, and commands among the nodes. Nodes that produce information (publishers) create "topics" (e.g., temperature, location, pressure) and publish "samples". DDS delivers the samples to subscribers that declare an interest in that topic.

DDS handles transfer chores: message addressing, data marshalling and demarshalling (so subscribers can be on different platforms from the publisher), delivery, flow control, retries, etc. Any node can be a publisher, subscriber, or both simultaneously.

The DDS publish-subscribe model virtually eliminates complex network programming for distributed applications.

DDS supports mechanisms that go beyond the basic publish-subscribe model. The key benefit is that applications that use DDS for their communications are decoupled. Little design time needs be spent on handling their mutual interactions. In particular, the applications never need information about the other participating applications, including their existence or locations. DDS transparently handles message delivery without requiring intervention from the user applications, including:

- determining who should receive the messages
- where recipients are located
- what happens if messages cannot be delivered

DDS allows the user to specify quality of service (QoS) parameters to configure discovery and behavior mechanisms up-front. By exchanging messages anonymously, DDS simplifies distributed applications and encourages modular, well-structured programs. DDS also automatically handles hot-swapping redundant publishers if the primary fails. Subscribers always get the sample with the highest priority whose data is still valid (that is, whose publisher-specified validity period has not expired). It automatically switches back to the primary when it recovers, too.

Interoperability

Both commercial and open-source software implementations of DDS are available. These include application programming interfaces (APIs) and libraries of implementations in Ada, C, C++, C#, Java, Scala, Lua, Pharo and Ruby. Some implementations are shown in the table at the end of this article.

DDS vendors participated in interoperability demonstrations at the OMG Spring technical meetings from 2009 to 2013.

During demos, each vendor published and subscribed to each other's topics using a test suite called the shapes demo. For example, one vendor publishes information about a shape and the other vendors can subscribe to the topic and display the results on their own shapes display. Each vendor takes turns publishing the information and the other subscribe. Two things made the demos possible: the DDS-I or Real-Time Publish-Subscribe (RTPS) protocol,[12] and the agreement to use a common model.