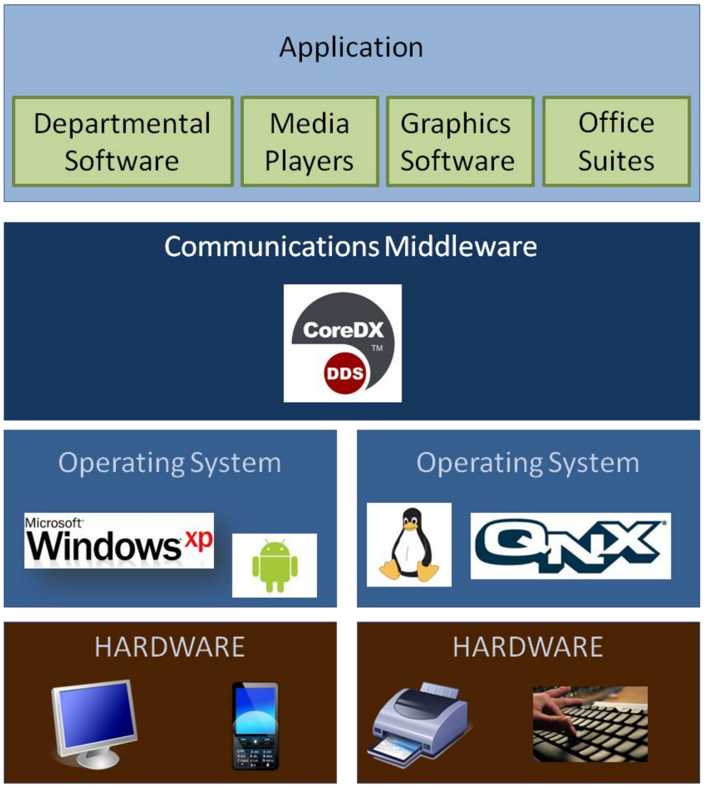
TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

Communications Middleware and DDS

December 2011

Contents

What is Communications Middleware?



The bottom layer consists of two or more different devices, each using their own version of an operating system. When we add middleware, those different devices can now communicate and use the same application, even though they use different operating systems.

Communications Middleware is computer software that enables two otherwise separate software components, processes, and/or applications to exchange information, either within one device, or between multiple devices. It is a specific kind of Middleware: the layer that lies between the operating system (Linux, MAC OS, Unix, Windows, etc.) and system applications (accounting software, media players, office productivity suites, etc.), that allows for communications. Communications Middleware may be built into or added to one or both of the applications. Sometimes Communications Middleware is referred to as "plumbing" because it is the piece that connects two (or more) applications and allows data to pass through.

The purpose of Communications Middleware is to simplify the designing, programming, and managing of software applications by streamlining the way these applications receive and process data. Communications Middleware simplifies writing communications software while providing sophisticated built-in features, reducing development costs.

Communications Middleware is used in a wide variety of software systems, from mobile devices (PDAs, Android phones, iPads, etc.) to enterprise and database systems. The equipment in these systems varies in screen and visual display capabilities, bandwidth capacities, and processing power. Communications Middleware facilitates communications between these differing devices. Communications Middleware can understand and support multiple programming languages (C, C++, Java, PHP, Ruby on Rails, etc.). We can use a cell phone and a PC here as an example. They both function in vastly different capacities, but with Communications Middleware are able to “talk” to and “work with” each other. This holds true for devices of similar capacities with different operating systems as well (say a Mac talking to a PC, or your accounting software communicating with your word processing software).

Companies and organizations are increasingly integrating previously independent applications with new developments and technologies; building enterprise-wide information systems. This integration process surrounds legacy applications; old or out of date software. Many of these legacy applications can be used only through their specific interface, and modifications prove costly or otherwise prohibitive. This can occur when software needs to be upgraded, or when one company acquires or takes over another company and their different systems now need to work together. Communications Middleware can link information from departmental databases such as payroll, sales, and accounting, or databases housed in multiple geographic locations into one centralized system, even if these databases store and process information differently.

Why use Middleware?

A wide variety of operating systems are being used in today's software development efforts: Windows, Android, Linux, and QNX, just to name a few. These operating systems communicate data differently, just as different hardware types (computers, cell phones, printers, etc.) store and retrieve information in a variety of ways. This translates into an expensive problem when your project wants to exchange information between two diverse systems, costing you and your business precious time, money, and resources.

There are a number of factors that contribute to the complexity of a software system, all of which increase schedule budget and risk. Examples of these factors include: size of the system, the number of different hardware architectures and/or Operating Systems involved, the number of nodes that must communicate together, the distance over which they must communicate, and the length of time a system must be maintained.

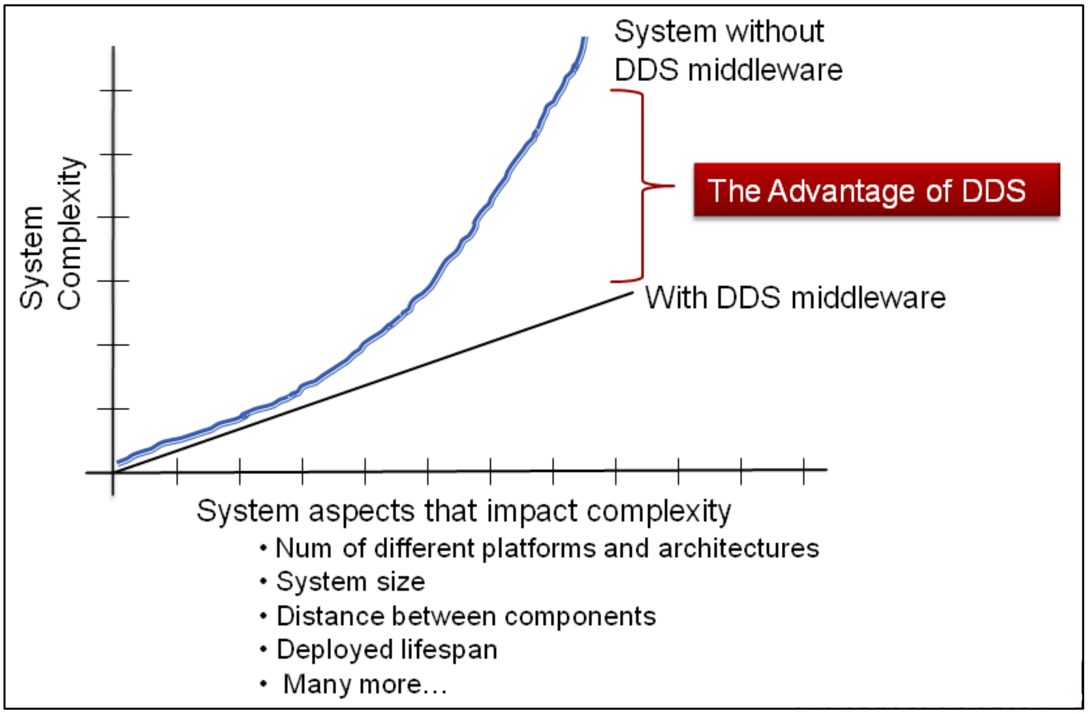
Using a Communications Middleware reduces system complexity. While different Communication Middleware technologies provide different features and benefits, they all strive to provide application portability across different operating systems and hardware, reduce development cost, and simply the resulting application code.

The Solution to this problem is DDS

What is DDS?

Data Distribution Service (DDS) is a type of communications middleware whose concept was standardized and is currently managed by the Object Management Group (OMG).

DDS simplifies software systems. DDS reduces risk and costs through development, integration, deployment, and the lifetime maintenance of distributed software systems.



DDS simplifies communications processes among different system types, making distributed development easier, faster, and more reliable. A DDS Communications Middleware simplifies your software project from development through initial deployment and maintenance over the life of the system.

How does DDS work?

DDS is in charge of transferring information: Information is transferred from publishers (producers and senders of messages) to subscribers (consumers and receivers of messages). Subscribers and publishers employing DDS can use different platforms or operating systems and still communicate with each other. Exchanges can take place through tens of thousands of devices at the same time, each one of which can be publishers, subscribers, or both simultaneously.

Systems that use DDS to communicate can do so independently of each other: They do not rely on each other's systems to send and process information. A publisher can still publish information even if there is no subscriber seeking the information. A subscriber can receive information from other publishers if the original publisher it was getting information from fails[1].

[1] http://www.ucsarchitecture.org/downloads/DDS%20Exec%20Brief%20v20l-public.pdf

DDS automatically knows how to send and receive messages with other DDS users: By design DDS is able to conclude which users should receive messages, where these users are located, and what to do if the receiver is unavailable. This simplifies data distribution, lessens the code required to perform message delivery (and less code means more efficiency), and thus saves time.

Each version of DDS can perform the same minimum set of functions in the same way with the same results. This is referred to as an "open standard system": system components from different manufacturers can be replaced and/or take over for each other with minimal or no changes to the larger systems in which they operate. This saves costs and avoids vendor lock-in.

DDS works in "real time": With very low overhead and efficient processing, messages are sent with minimal latencies (generally measured in the microseconds). It has a flexible architecture that is also *scalable*: it can adapt to processing both large and small amounts of data.

What are the Benefits of DDS?

DDS Reduces Risk:

DDS ensures consistency: Users of DDS can make changes to one system without the other system being adversely affected. Time is saved as less design time is allocated to determining how to get these systems to "talk" to each other.

DDS is interoperable: Every implementation of DDS can "talk" with every other type of DDS. The wire protocol is standardized; ensuring programs using different DDS products can discover each other and exchange data and can communicate. Referred to as *interoperability*, this is the ability of two or more systems to exchange information.[2]

*DDS automatically switches between publishers if the primary publisher fail*s. For example, once programmed, a publisher knows to "re-try" in 10 milliseconds, 10 minutes, every hour, or to drop the message all together, etc. if it is unable to reach a subscriber, and vice versa. In addition, subscribers always get the information that most closely matches their needs. If the information they seek is unavailable, they get

[2] http://www.omg.org/technology/documents/dds_spec_catalog.htm

the next best information. The system will automatically switch back to the information that most closely matches their needs when it becomes available.

DDS has no single point of failure: Systems that use DDS to communicate can do so independent of a server or service, and independently of each other. They do not rely on each other's systems to send and process information. A publisher can still publish information even if there is no subscriber seeking the information, or if a subscriber becomes "lost" for any reason. A subscriber can search for other publishers if the publisher it is getting information from fails or is lost.

DDS filters data for unique users: Each user only receives the information they need (or are intended) to receive. Consider online banking. The information is available to anyone who can access the web, as long as they have the correct username and password.

DDS can be used wirelessly to communicate information: For example: handling secure transactions via Smartphones and financial institutions, or scanning and tracking systems for package delivery systems. DDS provides high performing, **reliable** communications over un-reliable wireless networks.

DDS is reliable and always available **-** interactions with other services or application's are independent from network services, meaning they are always available for users (the server can't be "down" because of too many users, etc.) Data is cached by the publisher until all subscribers have received the information, so even if the network is unavailable the information is not lost. The publisher and subscriber merely try again.

DDs has the ability to tailor communication behavior – Quality of Service (QoS) policies allow the user to configure over 22 distinct items of communications behavior, providing fine-grained control to meet your communication requirements[3]. For example: reliability requirements, storage requirements, data presentation requirements, data filtering requirements, and redundancy or failover requirements (more than one path to communicate information).

DDS Reduces Cost:

DDS cuts development lifecycle cost: When disparate systems need to be integrated, instead of building a new system from the beginning, DDS can be deployed to facilitate communications and the project can continue. This saves both time and

[3] http://portals.omg.org/dds/category/keywords/qos

labor cost[4].

DDS shortens deployment timeline: Dynamic (automatic) discovery of communication end-points and optimization algorithms allow for easy deployment without time-consuming and error-prone site-specific reconfigurations. This is done by "hiding" lower level programming details with an intelligent and dynamic configuration capability.

Administration and maintenance expenses are reduced with DDS: Standardized programming and communication interfaces simplify administration and maintenance of DDS-enabled systems. It is easy to replace a system component because the other components don't have to change; the new component is simply accepted. It is easy to remove a component because other components will continue to exchange information when the removed component is gone. It is easy to add a device: new publishers and subscribers can be added with no change to existing components.

DDS is widely adopted across a variety of industries, including some of the most mission-critical systems within the United States Department of Defense[5]. DDS is also being used in a growing number of commercial applications, including smart vehicle control, high-speed stock trading, consumer electronics, telecommunications, manufacturing, power generation, medical devices, and simulation.

Key Points:

► DDS simplifies transferring information from one source to another, reducing risk and cost.

► DDS streamlines communications processes among different system types, making program development easier, faster, and more reliable.

► Subscribers and publishers employing DDS can use different platforms or operating systems and still communicate with each other.

► Exchanges can take place through countless devices at the same time, each one of which can be publishers, subscribers, or both simultaneously.

► DDS saves your business precious time, money, and resources through streamlining the communications process.

[4] http://www.ucsarchitecture.org/downloads/DDS%20Exec%20Brief%20v20l-public.pdf

[5] http://www.ucsarchitecture.org/downloads/DDS%20Exec%20Brief%20v20l-public.pdf

About Twin Oaks Computing

With corporate headquarters located in Castle Rock, Colorado, USA, Twin Oaks Computing is a company dedicated to developing and delivering quality software solutions. We leverage our technical experience and abilities to provide innovative and useful services in the domain of data communications. Founded in 2005, Twin Oaks Computing, Inc delivered the first version of CoreDX DDS in 2008. The next two years saw deliveries to over 100 customers around the world. We continue to provide world class support to these customers while ever expanding.

Contact

Twin Oaks Computing, Inc.
(720) 733-7906
+33 (0)9 62 23 72 20
755 Maleta Lane
Suite 203
Castle Rock, CO. 80108
www.twinoakscomputing.com