Middleware: The last roadblock to distributed systems development

By Dr. Stan Schneider

Both embedded systems and networking are high growth fields; combining them creates a world of opportunities. The list of applications goes on and on, including military systems, telecommunications, factory automation, traffic control, financial trading, medical imaging, building automation, consumer electronics, and more. Each of these industries struggles daily with how to create a single system from many distributed parts.

The information is out there. The network is fast, cheap, and reliable. We could build whole classes of new distributed applications, if we could just get the data. Unfortunately, today, getting the data isn't that simple.

The last roadblock

So what's holding distributed systems back? Network software architecture has not kept pace with easy-to-use operating systems and network hardware. The fundamental information models built for office and enterprise networks are fine for sharing files and executing transactions, but they simply don't fill the needs of embedded systems. In a real-time network, the main problem is to find and disseminate information quickly to many nodes; that problem is not yet completely solved.

Middleware, a class of software serving distributed applications by delivering data, is a layer between the ubiquitous network *stack* and the user application. The stack provides fundamental access to the network hardware and low-level routing and connection management. The application software, a set of custom-written modules, implements the various parts of the particular system. The middleware delivers data to the application software modules by using the underlying stack. However, finding the right data, knowing where to send it, and delivering it to the right place at the right time is a real challenge.

Embedded system developers, for the most part, have written their own in-house middleware layers. These range from simple, crude stack interfaces to sophisticated connection management and delivery services. They are often efficient, customized implementations that directly map to, or even merge with, the application. But, each one is unique and thus expensive to maintain and slow to adapt. Without a general solution, each application becomes complex, costly, and unreliable.

Recently, general embedded middleware technologies have begun emerging. The pattern follows the classic technological infrastructure evolution path: A few scrappy vendors generalize from custom implementations, gradually building a solution that will work for all. The benefit comes when standards (de facto or not) emerge, allowing industries to concentrate on the next, higher-level problem. Infrastructure software that clearly has adhered to this pattern includes operating systems, compilers, debuggers, network stacks, and so on. Middleware is positioned to be the next major area of standardization.

Generic middleware shows promise for standardized, easy distributed data access. If middleware can deliver on that promise, it has the potential to fuel an explosion in embedded applications that parallels the enterprise growth of IT.

Publish-subscribe networks

The fastest-growing embedded middleware technologies are publish-subscribe architectures. In contrast to the *central server with many clients* model of enterprise middleware, publish-subscribe nodes simply *subscribe* to data they need and *publish* information they produce. Messages pass directly between the communicating nodes. This design mirrors time-critical information delivery systems in everyday life, including television, magazines, and newspapers. Publish-subscribe systems excel at distributing large quantities of time-critical information quickly, even in the presence of unreliable delivery mechanisms.

Publish-subscribe architectures map well to the embedded communications problem. Finding and sending the right data is straightforward; nodes just request what they want and the system delivers it. Sending the data at the right time is also natural; publishers simply send data whenever new information is available. Publish-subscribe is efficient because the data flows directly from sender to receiver without intermediate servers. Multiple senders and receivers are easily supported, making redundancy and fault tolerance natural. Finally, and perhaps most importantly, each publisher-subscriber pair can establish independent Quality of Service (QoS) agreements. Thus, publish-subscribe designs can support extremely complex, flexible dataflow requirements. Publish-subscribe networks deliver the right data to the right place at the right time.

Emerging standards

Publish-subscribe designs are not new. Custom, in-house publish-subscribe layers abound. Industrial automation *fieldbus* networks have used publish-subscribe designs for decades. Commercial middleware products today control ships, digital television systems, large traffic grids, flight simulators, military systems, and thousands of other real-world applications. The technology is proven and reliable.

What is new is that standards are evolving. The Object Management Group (OMG), the standards body responsible for technologies like CORBA and UML, recently recognized the importance of publish-subscribe architectures. The newly adopted OMG standard, Data Distribution Service (DDS), is the first open international standard directly addressing publish-subscribe middleware for embedded systems. DDS, outlined in Figure 1, features extensive, fine control of QoS parameters, including reliability, bandwidth control, delivery deadlines, and resource limits. Several industry groups are rallying around DDS; for example, the U.S. Navy's Open Architecture specification stipulates DDS for all future Navy platforms. DDS represents the combined experience of many applications, and could be an important technology milestone.

Embedded middleware is crossing the threshold from specialized point solution to widely adopted infrastructure.

Of course, many difficult problems remain in middleware. For instance, complex systems require merged data distribution and client-server services. Guaranteed bandwidth reservations and prioritized message delivery is not yet available. Highly intermittent transports, such as wireless networks, must be supported. Scalability to large networks with thousands or millions of nodes presents an unsolved challenge.

Nonetheless, embedded middleware is crossing the threshold from specialized point solution to widely adopted infrastructure. When it completes that transition, the technology could render bold new distributed systems with much greater capabilities than are practical today. ECD

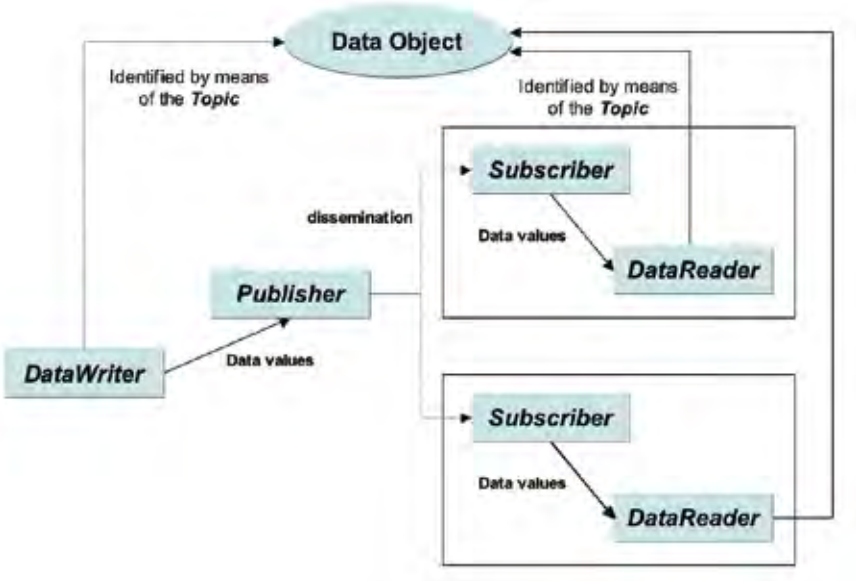

Figure 1

More information:
www.omg.org/news/whitepapers/IsDDS4U.pdf

Dr. Stan Schneider *is CEO of Real-Time Innovations, Inc., a leader in software infrastructure and tools for embedded systems.*

Immediately before RTI, Stan was an independent technical and management consultant, working with companies in medical products, digital signal processing, aerospace, semiconductor manufacturing, video and television, and networking. Before that, Stan managed one of the largest laboratories at Stanford University, focusing on intelligent mechanical systems. At Sperry Computer Systems (now Unisys), Stan developed networked communications systems and led the software team responsible for Sperry's personal computer product line. Stan began his career building data acquisition systems for automotive impact testing.

Stan holds a PhD in Electrical Engineering and Computer Science from Stanford, an MS in Computer Engineering, and a BS in Applied Mathematics.

To learn more, contact Stan at:

Real-Time Innovations, Inc.
3975 Freedom Circle, Sixth Floor
Santa Clara, CA 95054
Tel: 408-200-4715
Fax: 408-200-4702
E-mail: Stan.Schneider@rti.com
Website: www.rti.com