Application portability across RTOSs and connection media

<Written by> Terry Wright, RTI </W>

Middleware for embedded systems is beginning to make an impact on the embedded space.

TODAY'S EMBEDDED devices are more connected than ever before. Indeed, it's hard to conceive of a device that does not require connection capabilities in today's world. There is a plethora of connection mechanisms, from the ubiquitous Ethernet and serial connections, to USB, Wi-Fi and fabric technologies. RTOS vendors deliver an essential service to developers by providing an efficient and productive environment for system design and integration. As well as providing an optimized platform for single node application development, the modern RTOS environment must integrate the huge range of device drivers and protocol stacks needed to meet increasingly complex distributed system requirements, and also facilitate hardware to software integration.

What has changed in recent year's is that system developers now frequently have to deal with applications which must span multiple connected nodes; they also have to run across multiple hardware transport mechanisms connecting those nodes, and even across multiple different OS's, from the deeply embedded RTOS through RT Linux and up into the enterprise space where standard Unix and Windows systems are running. This is where middleware solutions are needed, providing the simplifying model of a single API that spans multiple OS's and CPU types.

Much as in the enterprise space, where middleware has been a key application enabler for many years, the embedded device space is now recognizing its increasing importance to cost effective application development and deployment.

Open standard

Data Distribution Services (DDS) is an embedded middleware solution that delivers common data distribution capabilities for almost any connection mechanism and RTOS or enterprise OS, but tuned to the real-time performance and memory requirements of what are, after all, highly demanding embedded devices and systems. Even better, the DDS mechanism is a published open standard developed and supported by the Object Management Group (OMG), which is already being adopted by a number of embedded software developers and vendors.

Of course there is more to developing embedded devices than just putting the components together; embedded systems designers need to manage the available resources, be it the amount of work the CPU can be expected to do or the amount of memory available and required. Embedded devices are usually expected to operate 24/7 without failure, and some safety critical systems need to have built in redundancy and automatic failover.

Effective Data Distribution middleware must also cater for real-time requirements. How is tasking controlled, how is the task priority set in Data Distribution middleware and how is the task stack size determined? How is the memory required by the middleware for data buffering allocated? How is data delivered? What overheads on the transport bandwidth does the data distribution method imply? All these questions and more must be considered by system designers.

> **"Developers now frequently have to deal with applications which span multiple connected nodes"**

The DDS standard (DDS 1.0) was released in 2005 and details a Data Distribution API using the Publish-Subscribe data distribution paradigm, as opposed to the more widely known Client-Server paradigm. DDS provides for a Peer to Peer, loosely coupled network of connected devices with no single point of failure. Best Effort or Reliable data delivery semantics and automatic discovery of DDS nodes facilitates a "self-healing" network. Asynchronous notification of the data arrival and early detection of failing nodes are a few of the many features available within the DDS specification. Application-level security validation can also be achieved using DDS features.

Vendor support

There are already a number of software vendors who have decided to provide support this new standard, Real Time Innovations (RTI) being one of them. RTI have a long history in working with Real Time Systems and RTOS vendors. Wind River recently purchased the RTI tools division which developed some of the most widely used real time diagnostic tools available (Stethoscope, MemoryScope, ProfileScope ,CoverageScope and TraceScope) so their experience in the real-time environment is widely acknowledged. Such experience, coupled with their long-standing involvement in Publish-Subscribe middleware and the drive for open standards, motivated RTI to chair the OMG standardization committee for the DDS specification.

RTI designed their implementation of the DDS specification (NDDS4.0) to be RTOS-friendly by providing support for standard RTOS offerings popular in today's heterogeneous networks; notably VxWorks from Wind River, Integrity from Greenhills and LynxOS from LynuxWorks and more recently QNX.

Fine Control

Quality of Service (QoS) settings within NDDS provide the fine control over system resources that system designers demand, such as NDDS task priorities, and NDDS task stack types and sizes. "Ahead-of-time" memory allocation for data buffers, no mallocs at run time, network redundancy and multiple concurrent transports are all standard supported features, in addition to the QoS built in to the DDS specification that handles automatic failover and data delivery semantics. RTI's DDS implementation features a pluggable transport mechanism for NDDS that provides the necessary abstraction for application developers who need to move data over any transport. e.g Ethernet, VME backplanes, PCI Express, Shared Memory or indeed StarFabric transports.

The Open DDS middleware standard, combined with RTI's NDDS implementation features and extensive RTOS support delivers an open standards based solution to the most demanding and complex of distributed system development today. <End/>

www.rti.com/resources.html
www.omg.org/technology/