The Global Leader in DDS

May 2011

# Limited-Bandwidth Plug-ins for DDS

## Integrating Applications over Low Bandwidth, Unreliable and Constrained Networks using RTI Data Distribution Service

*Edwin de Jong, Director of Product Management*

www.rti.com

US HEADQUARTERS
**Real-Time Innovations, Inc.**
385 Moffett Park Drive
Sunnyvale, CA 94089
Tel: (408) 990-7400
Fax: (408) 990-7402
info@rti.com

This paper describes a number of additional capabilities RTI has created for its OMG Data Distribution Service (DDS) compliant middleware, specifically to address the challenges of constrained network communication. It will discuss some sample network types and present solutions using RTI Data Distribution Service.

So what do we mean when we talk about "constrained" networks? This paper will discuss the issues and solutions for network connections which have one or more of the following characteristics:

- Low bandwidth – for example, HF radio links often have speeds as low as 1200 baud
- Limited bandwidth – where it is known ahead of time that peak traffic loads will exceed available bandwidth. This requires bandwidth management combined with improved wire level data transmission efficiencies.
- Very high signal latency – for example, satellite links can have high bandwidth but also very high latency for round trip communications. HF radios commonly have very high latency, too.

A network or its communications links may be constrained in other ways, such as:

- Highly error prone – sometimes referred to as DIL (Disconnected, Intermittent, Limited or Lossy) networks, such as radio data-links to Unmanned Vehicles
- Variable bandwidth – modern HF radio connections respond to varying noise levels by dynamically adjusting to appropriate waveforms and forward error correction techniques that may vary bandwidth
- Receive only - sometimes referred to as EMCOM (Emission Control) in defense, whereby a receiver does not wish to broadcast, usually in order to avoid location detection
- Simplex – HF radios commonly cannot detect incoming signals when they transmit, so if more than one transmits at a time nothing gets through and none of the radios can detect that issue

Three optional modules available for RTI Data Distribution Service are described in this paper:

- Compression
- Optimization of the DDS wire protocol (RTPS)
- Enhanced discovery, optimising application boot time over the network

**Background Information**

This paper assumes a reasonable level of understanding of the principles of DDS. A useful whitepaper to provide this background is "Is DDS for You?" at http://www.rti.com/mk/DDS.html.

Many low bandwidth or bandwidth constrained networks often exhibit disadvantaged, intermittent or lossy (DIL) communication. This paper does not fully explore how RTI DDS addresses these issues at a system level. However a previous RTI paper published at http://www.cotsjournalonline.com/articles/view/100940 provides a good introduction.

## Associated Applications

The technology described in this paper is applicable to many different types of applications. Applications commonly requiring these sorts of features and capabilities include:

- Advanced Combat Net Radio systems, such as the Thales PR4G
- Asset Tracking systems, like the Wi-Tronix vehicle tracking system http://www.rti.com/docs/witronix.pdf
- Air Traffic Management Systems, such as the ATLANTIDA consortium led investment http://www.rti.com/company/news/air-traffic-management.html
- Unmanned Systems such as http://www.rti.com/docs/GenAtomics_Snapshot.pdf , http://www.rti.com/docs/Insitu_Snapshot_2.pdf and http://www.rti.com/docs/Base10_RoboScout.pdf
- Wide Area Network based applications such as those using satellite communications

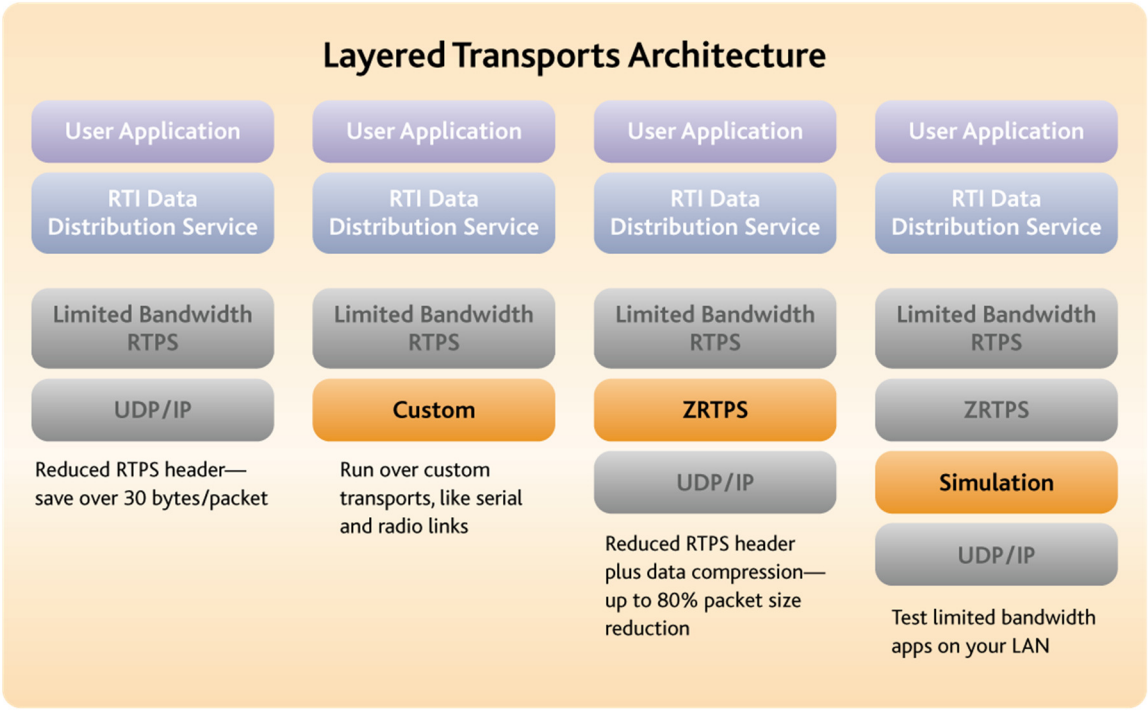
## The Constrained Network Problem

Whether your planned network has low or limited bandwidth, high latencies or all three, the additional requirement upon the messaging middleware is to place the minimum strain upon the communication links. This is usually expressed in terms of even more efficient messaging mechanisms, minimized packet overheads and maximized data to packet size ratio.

### Maximized Data to Packet Ratio

The DDS Wire protocol was designed for distributed real-time applications. It was therefore designed from day one to be highly efficient on the wire. For example data type information is exchanged once at discovery time, not continuously at run-time. In addition the open OMG standard wire protocol of DDS (called the DDS-RTPS Interoperability Wire Protocol) uses a very compact and efficient binary wire data representation (Common Data Representation—CDR). RTI has numerous benchmarks that support its throughput capabilities (see http://www.rti.com/products/dds/benchmarks-cpp-linux.html ).

The application developer is also given fine grain control of packet sizes, which is particularly important in low bandwidth and high latency networks where application data needs to be placed into large data packets to minimize the impact of latency on the application. RTI Data Distribution Service also uses type information intelligently in order to put minimum data on the wire. For example, it supports sparse data types so that only those fields of a type that have changed are transmitted on the wire. For more information on standard product capabilities see http://www.rti.com/products/index.html




### *Optimizing Packet Size on the Wire*

As mentioned earlier, DDS uses a standard on the wire protocol called RTPS. This open wire protocol is designed to guarantee interoperability between vendor implementations of DDS and is versatile enough to run over many types of transport and evolve over time without breaking interoperability with deployed systems. Although RTPS is already highly efficient for real-time networks, it does include a 20 byte packet header designed to meet the needs of a wide variety of network types, a 12 byte timestamp, and an additional 24 byte sub-message header that precedes each data-payload. The data overhead these 56 bytes introduce is usually insignificant for high-speed networks (indeed, it is very efficient when compared to most other messaging middleware). However, its impact on throughput increases as bandwidth or bandwidth availability reduces.

The packet headers contain information that can be optimized out if the application is running over transports that are extremely low bandwidth, where every bit is precious; in small networks where addressing requirements are minimized; or in high latency environments where certain types of time oriented information become irrelevant and unusable at the application layer.

For network transports using UDP/IP, RTI has created an even further optimized version of RTPS that enables the developer to strip out certain header information for networks and applications with the aforementioned attributes. The configurable options enable up to 30 bytes to be stripped out of the typical 56 byte standard RTPS header. For example, the RTPS *GUID* (Global Unique Identifier) is a 12 byte field consisting of 3 fields of 4 bytes each. The first is an address field, which in most networks is masked down to the last byte (for example in an IP network the address mask might be 192.168.1.x) and can therefore be reduced to 1, 2 or 3 bytes depending on your IP

address mask. The second is a local application identifier (within the computer at that IP address), but many distributed real-time systems run a single application on each computer, so this field can be ignored and defaulted to 1. Lastly, a participant index is used to identity multiple DDS participants on the same computer. However, many applications running on radios or other highly-constrained devices have only one participant per node, so again this field can be stripped out. Overall this RTPS 12 byte header field can be reduced down as low as 1 byte.

Using similar logic, up to 18 further bytes of information can be stripped from the header without reducing the utilized functionality of DDS at all. Removing this packet content can deliver a 53% efficiency saving on the packet overhead of the standard RTPS protocol.

RTI's optimized wire protocol can be layered with the compression plug-in to create a powerful combination of optimizations for constrained or low bandwidth and/or high latency networks. However, to fully optimize messaging middleware for these sorts of networks, we need to optimize the way DDS itself works, in particular its discovery operation.

### *Packet Compression*

However, when the network link is very low bandwidth or already close to its capacity, application specific efficiencies may be sought. Compression is a common approach to address the issue, but it comes at a cost – CPU use and latency. The time taken to compress a packet will add to the latency of the communication. In many systems, the available processing power versus the available bandwidth is a simple trade off to make: the latency in low bandwidth networks is often high anyway, so the proportional impact of compression can be minimal. This is certainly true in HF radio networks, where latency is measured in seconds. In addition, on today's low cost and high performance processors, even the most comprehensive compression techniques will take a small fraction of a second. On more powerful processor platforms, compression offload engines may be available to minimize latency impact even further.

RTI has developed a compression transport plug-in that addresses all of these issues and more. Compression algorithms work at varying levels of efficiency dependent on the size and type of data being compressed (images versus text, for example). As DDS gives fine grain control over packet sizes to the application, it becomes very beneficial if the compression plug-in provides flexibility to control which algorithm it applies to which packets. This plug-in allows you to define three packet sizes—small, medium and large—with your own size parameters. You can then assign a compression algorithm to each packet size.

Alternatively, if you have processing power or latency to spare, you can have the plug-in run in an automatic selection mode. In this mode, all compression algorithms run in parallel for each packet and the best result is selected to send the packet to the underlying transport.

The compression plug-in comes with two integrated compression algorithms as well as facilities for users to integrate their own compression algorithm:

- Open Compression Plug-in: a framework for developers to integrate their own compression algorithm or algorithms
- Zlib: an abstraction of the DEFLATE compression algorithm used in gzip
- Bzip2: a lossless compression algorithm which, depending on the data, can be far more effective than Zlib but takes approximately 12 times longer on typical data

The flexibility of the Compression Transport Plug-in combined with the inherent fine grain data communication capabilities of DDS enables users to fine tune the bandwidth utilization and latency to best meet their distributed application requirements.

## Efficient Data Distribution

This paper does not go into depth on the general features of DDS that enable application developers and systems integrators to make the most efficient use of their underlying transports. However we will briefly summarize some of those DDS and RTI Data Distribution Service features that tend to be of importance to application developers in constrained networks:

- Multicast support - static multicast (multicast address IP assigned to a fixed set of recipients) and dynamic multicast (each sender has a set of IP multicast addresses reserved for dynamic multicast to allow negotiation of the multicast recipients). DDS makes no distinction between these and pushes the exact methodology down to the transport level implementation.
- Message prioritization – can be a key issue especially in high latency networks. DDS supports several Quality of Service (QoS) features such as TransportPriority and LatencyBudget that enable fine grain control of prioritization of bandwidth utilization to take advantage of the underlying network infrastructure provides.
- Latency-aware packet batching – particularly important in high latency networks. This is transparently supported by RTI Data Distribution Service and configured by setting a QoS.
- Multiple transport support – DDS supports packet sequencing. So if you have multiple connections between two nodes (an HF radio and a satellite link for example), you can transmit on both and the recipient will ignore packets already received.
- Flow control – RTI Data Distribution Services allows applications to configure the shape of the traffic load the middleware places on the network, limiting maximum throughput, peak bursts, etc. It also allows applications to tag different data-flows with their relative latency needs so that the middleware can police the access to the transport and ensure that the most urgent traffic gets precedence.

Of course any messaging middleware has to discover and set up communication paths between its active participants. How efficiently this is done is important to any network at start up, but it's critically important that this is achieved optimally in low bandwidth and high latency distributed systems.

### DDS Discovery

A key attribute of RTI Data Distribution Service is its inherently fault tolerant and distributed system friendly design. It not only tolerates any node failing at any time, but also enables network nodes and applications to start up and terminate (planned or unplanned) in any order at any time across a network or on a single computer. It even caters to incremental version upgrades of DDS on live networks. All of these capabilities are under-pinned by the DDS-RTPS protocol and Discovery mechanism.

This fault tolerant integration flexibility requires DDS to discover each publish-subscribe communication channel between participants. DDS typically supports a completely anonymous publish-subscribe environment, where no knowledge of any part of the application is known ahead of time in order to facilitate integration in highly dynamic environments. Each participant merely knows that another participant somewhere may be able to meet the data needs it has or may be interested in the data it has and how it can be provided. This means any deployed distributed system can be changed, added to, or reduced in terms of active participants—without recourse to re-builds, recompilation or even re-starts of any other part of the system.

To support this capability, each participant maintains a local knowledge cache of other participants in the domain and their entities. To maintain their presence in the distributed application, participants must refresh their presence in the domain or they will be aged out of the cache. To sustain the cache, participants announce their presence at a period set up at the application layer. Participants may also change their QoS as well, and this information may need to be propagated, too.

For most networks, this overhead is minimal, fairly steady state, and does not have any significant detrimental effect on the application performance or network throughput. However, in low bandwidth and/or high latency networks, this can consume enough bandwidth to have a noticeable effect on the applications in terms of the time it takes for the communication infrastructure to be set up.

### Introducing Quasi-Static Discovery

To meet the challenges of the low bandwidth and/or high latency over DIL type networks, RTI has created a discovery plug-in that enables highly efficient utilization of available bandwidth.

One key design issue that distributed application developers need to consider is the nature of application *presence*. DDS typically makes no assumptions at all about the set of applications that may or may not be present on the network at any time nor about how that application is distributed across the network. Hence, discovery works in the background to continuously maintain address tables and resolves publish-subscribe QoS negotiations. However, in many deployed systems, the application or set of applications may be known ahead of time and will not change while the network is in service, nor will the QoS parameters vary – the application presence is pre-determined and stable. However, the order of node and participant start up may still be arbitrary, and nodes and participants may often fail, become inaccessible, or otherwise need a fall back node or

participant to continue the application. The quasi-static discovery plug-in of RTI Data Distribution Service uses these assumptions to achieve a highly optimal discovery phase for DDS which is particularly beneficial in constrained and low bandwidth networks or networks requiring very fast boot up.

### *Principles of Quasi-Static Discovery*

The DDS discovery process is executed in two stages. There is a DDS Participant[1] discovery phase and a DDS Endpoint[2] discovery phase. With quasi-static discovery, the second phase is suppressed, and is replaced by an XML file that acts as a look up table. When a DDS Participant announces it has joined the distributed environment, RTI Data Distribution Service looks up the set of end-points that must now exist and establishes the publish-subscribe communication channels. No QoS negotiation is needed because this has already been validated in the development environment and is part of the XML look up table, further reducing traffic on the network.

The net effect of this optimized discovery is a significant reduction in the time taken to establish channels of communication in systems in which application constituents are known a priori, while maintaining DDS's robustness in DIL environments.

### *Quasi-Static Discovery Performance*

In early tests on a 2400 baud HF radio network of 8 nodes (1 participant per node), quasi-static discovery reduced discovery time (and thus application set up time) to less than 40 seconds. To put this in perspective, for 8 nodes to simultaneously start up and seek to set up their connections over a 2400 baud HF radio network means that each radio has 300 baud available to it. However, due to collisions and other issues in HF radio, you end up with a more realistic 150 baud connection—in other words, an effective throughput of 18 bytes per second. So, a complete 8 node application-level messaging framework with one publisher and one subscriber per node is set up by quasi-static discovery through the successful transmission of under 6000 bytes of information!

These tests were made on the Thales PR4G Combat Net radio. This HF radio system is widely adopted across many nations for portable infantry tactical radio requirements.

## Summary

DDS has proven its applicability to challenged DIL distributed environments with its successful deployment in many tough environments (see http://www.rti.com/company/customers.html).With the addition of compression, an optimized wire protocol, and quasi-static discovery, customers of RTI Data Distribution Service can now confidently apply this resilient integration middleware to highly constrained network environments.

[1] A DDS Participant is a process which may or may not be on another node
[2] A DDS Endpoint is a publisher or subscriber contained within a DDS Participant