Out-of-Band Flow Control for Reliable Multicast

Harry Wolfson

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02421
781-981-5996
HarryWolfson@LL.MIT.EDU

Keywords:
HLA, RTI, Reliable, Multicast, flow control, squelch, throughput, out of band.

ABSTRACT: *The High Level Architecture (HLA) Runtime Infrastructure (RTI) provides a variety of message Transport Services for different communication needs. These Transport Services include Best Effort and Reliable. While Best Effort Multicast messaging is fairly well standardized using UDP (User Datagram Protocol), a protocol for Reliable Multicast is still a technology in development.*

The Reliable Multicast service that was first implemented for the STOW Program has had significant development over the years to enhance its utility for a wide variety of users, including improved robustness and reliability for delivery of messages. The most recent version of this Reliable message service is currently being used in the DMSO RTI 1.3v7.

This paper will describe the recent addition of Flow Control to the RTI Reliable service. End to end flow control prevents rapid, or heavy, producers of data from overwhelming and clogging the network. It insures that subscribers are able to receive and process that data in a timely manner without overflowing their own buffers, or preventing other traffic on the network from continuing to flow.

1. Introduction

Prior implementations of Reliable Multicast for the DMSO RTI 1.3, as well as for its predecessor for the STOW program [1], handled extremely fast producers of data combined with slower consumers of data, in a less than ideal manner. Cached data would fill available buffers; or socket-based connections would stall and/or die; or data would simply be dropped. This was due primarily to the lack of any coordination of data rates, or throughput, between producers and consumers of data, i.e. previous versions of this implementation of Reliable Multicast lacked any end-to-end flow control.

This paper will describe a new out-of-band, flow control protocol that allows all producers and consumers of data in a Federation Execution to coordinate the flow of data sent via the Reliable Multicast service. This flow control: helps prevent clogged networks (deadlock, deadly embrace, etc.); prevents the uncontrolled filling of buffers and available memory; prevents the discarding of data due to a single Federate, or the network, being overwhelmed with too much data. In other words, Flow Control for Reliable Multicast in the DMSO RTI 1.3v7 provides a control on the throughput of data to match the capabilities of the various Federates within an Execution.

2. Motivation

The original design of the STOW RTI Reliable Multicast focused on high throughput and low latency performance, and the design specification allowed for the occasional dropped message (i.e. not 100% reliable), therefore flow control was not a requirement. As that implementation was modified for the DMSO RTI 1.3, it became apparent that a more ***reliable*** Reliable Multicast was necessary! In addition, the

This work was sponsored by the Department of the Air Force under Air Force Contract F-19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

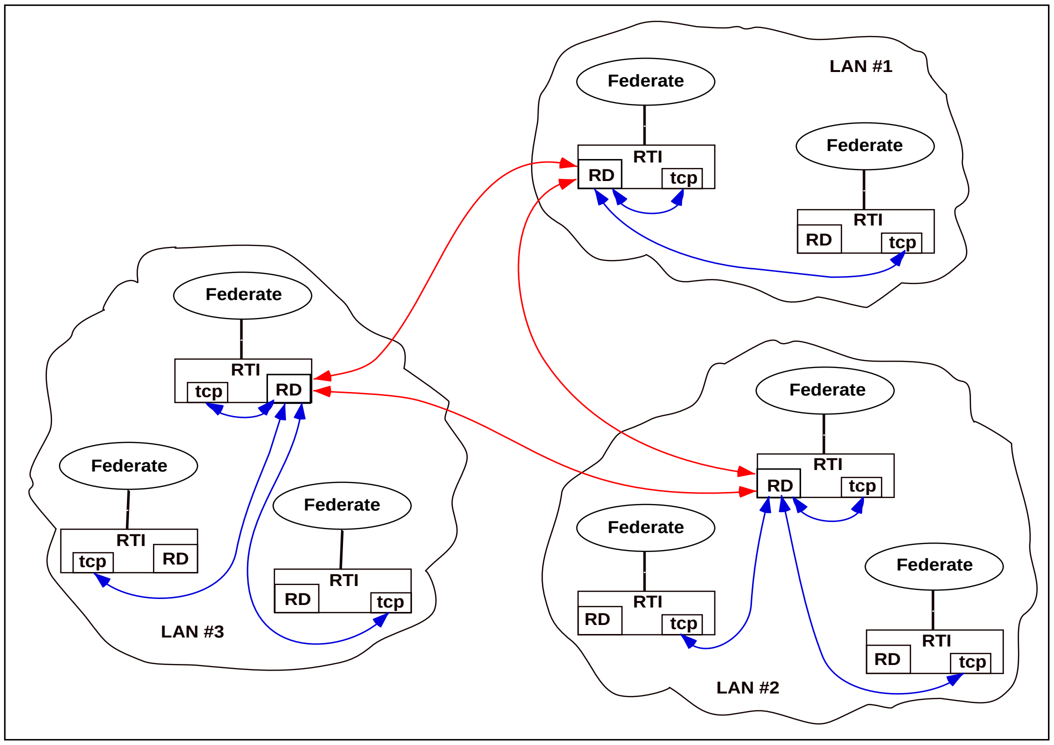

Figure 3.1: Sample Topology of Reliable Servers and Clients

DMSO RTI required the use of tick(min, max), which meant that the low level Transport Services of the RTI would not have an unlimited amount of time to process incoming and outgoing messages. These conflicting requirements of better reliability, together with limited processing time, led to the need for end-to-end flow control. This would allow for Federation wide "throttling" of the data rates and prevent dropped messages, or the dreaded "lock up".

Most other design concepts and implementations of Reliable Multicast (RM) are based on UDP communication, and deal with Flow Control in a variety of ways, including those RM implementations that rely on monitoring the counts or status of ACK [2] and NAK [3] signaling. These approaches were not applicable to our design since our RM is based on top of TCP/IP, which hides all sender/receiver acknowledgments, retransmission and error correction. Monitor Based Flow Control [4] utilizes an independent hand-shaking protocol between the sender and its receivers to monitor their state, and control the transmission rate. A RM flow control design concept that is somewhat similar is described as Polling Based Reliable Multicast (PBRM) [5].

3. Reliable Multicast Implementation

The design and implementation of the Reliable Multicast that is used in the DMSO RTI 1.3 is very closely based on that described in [1]. What follows is a very brief summary description.

This implementation of a Reliable Multicast Transport Service is built on top of TCP/IP. This takes advantage of the inherent advantages of TCP that are required for a reliable message service, namely: reliable delivery; sequential ordering of messages; and a measure of robustness and fault-tolerance. Of course, since TCP is a point-to-point communication protocol, we need to provide a way to distribute messages to numerous recipients (point-to-multipoint). This is accomplished by implementing a client/server approach, where the client (TCPconn) only needs to maintain a single connection with one server (Reliable Distributor), and the server is responsible for distributing messages to all intended recipients.

As shown in Figure 3.1, each Federate is tightly coupled to the RTI. The RTI provides Reliable Message Transport to the Federate via its TCPconn client (labeled *tcp* in the Figure). Each TCPconn client connects to a single Reliable Distributor server (labeled *RD* in the Figure). All Reliable Distributors communicate with each other in a fully connected mesh. Each reliable connection (shown in the Figure as a double sided arrow) is implemented with a point-to-point TCP connection.

4. Flow Control Protocol

Since our RM implementation sits on top of TCP/IP [1], we face the additional constraint that if the network, or our message queues, become blocked or sluggish due to excessive transmission rates combined with slow receive processing, we loose the ability to perform "in-band" hand shaking to form a flow control protocol. We briefly considered basing our flow control strictly on timeouts (in essence, a "blind" protocol) but realized that a more robust design would require communication between senders and receivers to coordinate traffic throughput. Our flow control is based on a combination of monitoring of a Federate's internal state, as well as inter-Federate coordination via out-of-band (OOB) communication.

Flow control is accomplished by preventing a Federate from sending new data out to the Federation Execution when receiving Federates become unable to keep up with the volume or rapidity of the traffic. The RTI makes the decision to restrict new data from being propagated from a Federate based on "ClearToSend" (CTS) or "Squelch" (SQ) mode. The algorithm for choosing CTS or SQ is based on monitoring of four internal states of each Federate: receive queues; send queues; receipt of a remote SQ message; or the absence of receipt of a remote CTS message within a timeout period.

4.1 Monitoring of Internal Queues

The RTI in each Federate maintains separate queues for buffering of incoming and outgoing data. The Receive Message Queue buffers incoming data from both Best Effort and Reliable Transport. The number of messages in this queue is reduced as the Federate processes incoming data. The number of messages in this queue increases if the RTI is required to return control to the Federate as the result of reaching the time limit imposed by tick(min, max). Obviously, this buffer has the potential to fill up if the Federate is unable to keep pace with incoming data retrieved from the wire before the tick(min, max) time limit has expired.

Each connection in support of Reliable Multicast maintains its own Send Message Queue to buffer outgoing data. The number of messages in these queues can also increase as the tick(min, max) time limit expires. In addition, the stored message level in these queues will increase if the TCP buffers in the kernel fill up, or become sluggish, due to network congestion or if one of the multicast receivers is unable to process the level of traffic.

If the number of messages in the RTI's Receive Message Queue, or in ***any*** of the Send Message Queues, exceeds a configurable maximum threshold, then the Squelch Mode flag for ***that*** buffer is set. If the number of messages is less than the minimum threshold, then the ClearToSend Mode flag for that buffer is set. (Note that the minimum and maximum message thresholds are independently configurable to allow for hysteresis in the system.) Separate counters are incremented and decremented to keep track of CTS/SQ flags for the various buffers.

4.2 Out-of-Band Messaging Link

To overcome the risk of congestion in our TCP connections preventing inter-Federate communication in support of flow control handshaking, we implemented an out-of-band (OOB) protocol. Since this messaging path does not need to be reliable, we establish independent, UDP (Best Effort) "connectionless" links to carry our protocol traffic. A separate, 2-way, point-to-point, unicast UDP link is established in parallel with each TCP connection (i.e. parallel connections both between a Federate and its Reliable Distributor, as well as between connected pairs of Reliable Distributors).

If all the internal buffers in a Federate are in the ClearToSend mode, then a CTS message is sent on this OOB link to the Federate's Reliable Distributor. If any one of the internal buffers is in Squelch Mode, then a SQ message is sent to the Reliable Distributor. Similarly, a Reliable Distributor monitors the state of the internal buffers that it maintains for each of its Clients, as well as listening for received CTS/SQ messages from its Clients. If a Reliable Distributor goes into Squelch Mode based on the state of its internal buffers, or due to receipt of a SQ message, then it propagates SQ messages to ***all*** of its Clients, including other Reliable Distributors.

These protocol messages are very short (essentially just a RTI Reliable header with a flag set) and are sent out periodically based on the parameters described below in Section 4.4.

4.3 Squelch Mode

A Federate's RTI will go into Squelch Mode if ***any*** of the following conditions are true (see Figure 4.1): the Squelch flag is set on its Receive Message Queue; the Squelch flag is set for ***any*** of its Send Message Queues; if it has received a Squelch message from a

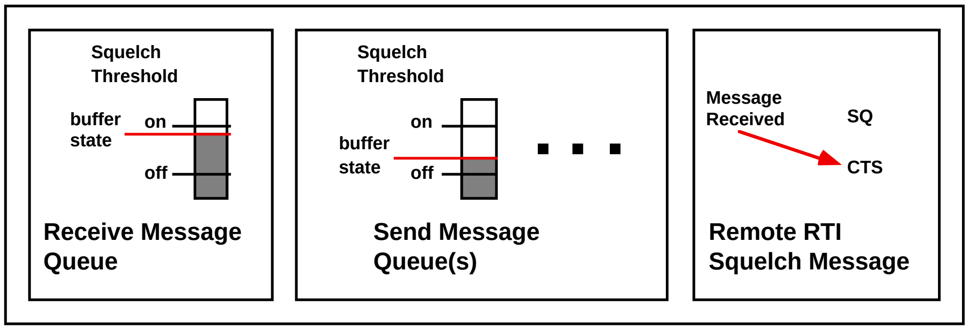

Figure 4.1: Conceptual Model of Squelch Mode

Stream_manager_squelch_enabled	Enables flow control in a Federate. Boolean. Default is 1 (on).
Stream_manager_squelch_off_threshold	ClearToSend Mode is activated if the number of messages in a queue is less than this percentage of the queue's maximum capacity. Float: Defaults to 0.
Stream_manager_squelch_on_threshold	Squelch Mode is activated if the number of messages in a queue exceeds this percentage of the queue's maximum capacity. Float: Defaults to 1.
Stream_manager_cts_poll_interval	Minimum time interval between sending ClearToSend protocol messages. Milliseconds: Defaults to 3000.
Stream_manager_squelch_poll_interval	Minimum time interval between sending Squelch protocol messages. Milliseconds: Defaults to 1000.

Table 4.1: Flow Control RID Configuration Parameters

remote RTI (either a Reliable Distributor receiving a SQ from any of its clients, or a Client receiving a SQ from its Reliable Distributor); or it has not received a CTS message from a remote RTI within the designated timeout period (either a Reliable Distributor receiving a CTS from each of its clients, or a Client receiving a CTS from its Reliable Distributor). Hysteresis in the queue threshold settings, as well as relatively long time intervals for sending CTS/SQ protocol messages, allow for excellent stability in the system and prevents the RTI from constantly slamming in and out of Squelch mode.

When an RTI is in Squelch Mode, it prevents its Federate from generating new outgoing data. This is simply accomplished by grabbing control of tick and not returning control to the Federate until the conditions for ClearToSend Mode are achieved. Of course this means that the time limits for tick(min, max) are violated when a Federate is in Squelch Mode. But as we all learned from a beheaded French queen, you can't have your cake and eat it too! This was the tradeoff that was made to balance the needs for flow control, with the expectations of the Federate. If a Federate generates data faster than consumer Federates can process that data, then something has to give. We regulate the data throughput so that all Federates in an Execution can keep up with the data they need to receive, without causing a complete meltdown of the Execution.

4.4 Flow Control Configuration

A set of configurable parameters are contained in the RTI Initialization Date (RID) configuration file to control the Reliable Multicast Flow Control. Table 4.1 lists these parameters, along with brief explanations. In addition to these parameters, the configurable capacities for the various send and receive buffers (which have always been part of the RID) will also affect the performance of flow control.

5. Summary

A flow control protocol for Reliable Multicast, based on monitoring the state of internal buffers, in conjunction with out-of-band signaling between senders and receivers, has been described. This protocol has been successfully incorporated in the DMSO RTI 1.3v7 and is currently being used by end users around the world in a variety of simulations, both large and small. This Reliable Multicast implementation sits on top of standard TCP/IP communications and uses a configurable number of repeaters, or "exploders," to forward messages from many senders to many receivers. This Reliable Multicast implementation has proven to be robust and scalable to large simulations.

6. Acknowledgments

My sincere thanks to Dan Van Hook and Jim Calvin for their motivation and suggestions for the design of the OOB Flow Control. Also my gratitude is owed to Carol Chiang who was the primary person who went out into the field to demonstrate the DMSO RTI 1.3 to sponsors and end users while kinks in the protocol were still being "worked out." Her analysis and suggestions for improvements and optimizations were instrumental in the successful completion of this work.

7. References

[1] H. Wolfson, S. Boswell, D. Van Hook, S. McGarry: “Reliable Multicast in the STOW RTI Prototype”, 97S-SIW-119, 1997 Spring Simulation Interoperability Workshop Papers, Vol. 2, pp. 771-777, March 1997.

[2] K. Lee, S. Ha, V. Bharghavan: "IRMA: A Reliable Multicast Architecture for the Internet", Proceedings IEEE Infocom '99, Vol. 3, pp. 1274-1281, March 1999.

[3] M. Yamamoto, Y. Sawa, S. Fukatsu, H. Ikeda: "NAK-based Flow Control Scheme for Reliable Multicast Communications", Proceedings IEEE Globecom 1998, Vol. 5, pp. 2611-2616, November 1998.

[4] T. Sano, T. Shiroshita, O. Takashi, N. Yamanouchi: "Flow/Congestion Control for bulk reliable multicast", SPIE Conference on Routing in the Internet, Vol. 3529, pp. 329-338, November 1998.

[5] M. Barcellos, P. Ezhilchelvan: "An End-to-End Reliable Multicast Protocol Using Polling for Scaleability", Proceedings IEEE Infocom '98, Vol. 3, pp. 1180-1187, March/April 1998.

Author Biography

HARRY WOLFSON has been a Member of the Technical Staff at MIT Lincoln Laboratory since 1985. He was responsible for the design and implementation of the Reliable Message Transport Service, and its underlying protocol, for the STOW RTI Prototype and the DMSO RTI v1.3. Currently, Harry splits his time between the RTI, and development of real time Intrusion Detection Systems. Prior to joining Lincoln, Harry was a Staff Member of TRW in Redondo Beach, Ca. He received a Master of Science in Electrical Engineering from the University of Southern California with a major in Microwave Design and Communication Systems. He received his Bachelor of Science in Electrical Engineering from Northeastern University.