

Network Working Group
Request for Comments: 4410
Category: Experimental

M. Pullen
F. Zhao
George Mason Univ
D. Cohen
Sun Microsystems
February 2006

Selectively Reliable Multicast Protocol (SRMP)

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

The Selectively Reliable Multicast Protocol (SRMP) is a transport protocol, intended to deliver a mix of reliable and best-effort messages in an any-to-any multicast environment, where the best-effort traffic occurs in significantly greater volume than the reliable traffic and therefore can carry sequence numbers of reliable messages for loss detection. SRMP is intended for use in a distributed simulation application environment, where only the latest value of reliable transmission for any particular data identifier requires delivery. SRMP has two sublayers: a bundling sublayer handling message aggregation and congestion control, and a Selectively Reliable Transport (SRT) sublayer. Selection between reliable and best-effort messages is performed by the application.

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. Protocol Description	4
3. Message Formats	6
3.1. Bundle Message Format:	6
3.2. Bundle Header Format	7
3.3. Feedback Message Format	9
3.4. SRT Mode 0 Header Format	10
3.5. SRT Mode 1 Header Format	11
3.6. SRT Mode 2 Header Format	11
3.7. SRT NACK Format	12
3.8. User-Configurable Parameters	13
4. TFMCC Operation	13
4.1. TCP Rate Prediction Equation for TFMCC	13
4.2. Bundling	13
4.3. Congestion Control	14
4.4. Any-Source Multicast	14
4.5. Multiple Sources	14
4.6. Bundle Size	15
4.7. Data Rate Control	15
4.8. Mode 1 Loss Detection	16
4.8.1. Sending a Negative Acknowledgement	16
4.9. Unbundling	17
4.10. Heartbeat Bundle	17
5. SRT Operation	17
5.1. Mode 0 Operation	18
5.1.1. Sending Mode 0 Messages	18
5.1.2. Receiving Mode 0 Messages	18
5.2. Mode 1 Operation	18
5.2.1. Sending Mode 1 Data Messages	19
5.2.2. Receiving Mode 1 Data Messages	19
5.2.3. Sending a Negative Acknowledgement	20
5.2.4. Receiving a Negative Acknowledgement	21
5.3. Mode 2 Operation	21
5.3.1. Sending Mode 2 Data Messages	21
5.3.2. Receiving Mode 2 Data Messages	22
5.3.3. Sending a Positive Acknowledgement	23
5.3.4. Receiving a Positive Acknowledgement	23
6. RFC 2357 Analysis	23
6.1. Scalability	23
6.2. Congestion	24
7. Security Considerations	25
8. List of Acronyms Used	26
9. Contributions	27
10. References	27

1. Introduction

There is no viable generic approach to achieving reliable transport over multicast networks. Existing successful approaches require that the transport protocol take advantage of special properties of the traffic in a way originally proposed by Cohen [10]. The protocol described here is applicable to real-time traffic containing a mix of two categories of messages: a small fraction requiring reliable delivery, mixed with a predominating flow of best-effort messages. This sort of traffic is associated with distributed virtual simulation (RFC 2502 [4]) and also with some forms of distributed multimedia conferencing. These applications typically have some data that changes rarely, or not at all, so the best efficiency will be achieved by transmitting that data reliably (the external appearance of a simulated vehicle is an excellent example). They also require real-time transmission of a best-effort stream (for example, the position and orientation of the vehicle). There is no value to reliable transmission of this stream because typically new updates arrive faster than loss identification and retransmission could take place. By piggy-backing the sequence number (SN) of the latest reliable transmission on each bundle of traffic, the reliable and best-effort traffic can co-exist synergistically. This approach is implemented in the Selectively Reliable Multicast Protocol (SRMP).

The IETF has conducted a successful working group on Reliable Multicast Transport (RMT) that has produced RFCs 2357 [6], 2887 [11], and 3450 through 3453 [12 - 15], which define building block protocols for reliable multicast. Selectively reliable multicast is similar in spirit to these protocols and in fact uses one of them, TCP-Friendly Multicast Congestion Control (TFMCC). This document provides the basis for specifying SRMP with TFMCC for use on an experimental basis. Key requirements of the RMT process that is carried forward here are specified in RFC 2357 [6]. These generally relate to scalability and congestion control, and are addressed in section 6 of this document.

1.1. Terminology

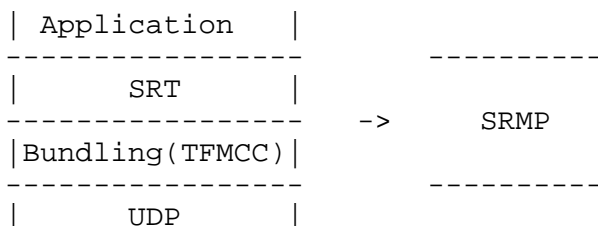
In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

2. Protocol Description

The Selectively Reliable Multicast Protocol (SRMP) has two major components: Selectively Reliable Transport (SRT) and a "bundling sublayer" that implements TCP-Friendly Multicast Congestion Control (TFMCC), as proposed by Widmer and Handley [2], in order to meet the requirements of RFC 2357 [6] for congestion avoidance.

SRMP is capable of reliable message delivery over multicast networks, when the messages to be delivered reliably represent a fraction of a larger, associated best-effort flow and only the latest reliable message must be delivered. The basic strategy for SRMP is to trade as little network capacity as possible for reliability by buffering the most recently sent reliable message at each sender and piggy-backing its sequence number on associated best-effort messages. For this purpose, three modes of sending are defined:

- o Mode 0 messages. These will be delivered best-effort; if lost, no retransmission will be done.
- o Mode 1 messages. When a Mode 1 message loss is detected, the receiver will send back a NACK to the sender, where SRMP will retransmit the latest reliable message from that sender. Senders define data identifiers (dataIDs), allowing multiple reliable message streams to be supported. Mode 1 messages may be up to 131,071 bytes long; SRMP provides for segmentation and reassembly, but only for the latest Mode 1 message for any given <sourceAddress, multicastAddress, dataID>.
- o Mode 2 messages. Through Mode 2 messages, SRMP provides for a lightweight, reliable, connectionless peer-to-peer unicast transaction exchange between any two members of the multicast group. This is a unicast message requiring positive acknowledgement (ACK).



The bundling sublayer is transparent to the Selectively Reliable Transport (SRT) sublayer. It implements congestion control both by dropping Mode 0 messages at the source when needed and by bundling multiple short messages that are presented by applications within a short time window. It also performs NACK suppression.

A bundling sublayer data unit is called a bundle. A bundle is made up of a bundle header and one or more Mode 0 and Mode 1 SRMP messages. Retransmission of Mode 1 messages does not imply retransmission of the original bundle; the retransmitted message becomes part of a new bundle.

The TFMCC layer's behavior follows the mechanism described by Widmer and Handley. This is an equation-based multicast congestion control mechanism: in a multicast group, each receiver determines its loss rate with regard to the sender, and calculates a desired source sending rate based on an equation that models the steady-state sending rate of TCP. A distributed feedback suppression mechanism restricts feedback to those receivers likely to report the lowest desired rates. Congestion control is achieved by dropping best-effort (Mode 0) messages at random. For example, in distributed simulation, Mode 0 messages are part of a stream of state updates for dynamic data such as geographic location; therefore, the application can continue to function (with lower fidelity) when they are dropped.

As described by its authors, TFMCC's congestion control mechanism works as follows:

- o Each receiver measures the loss event rate and its Round-Trip Time (RTT) to the sender.
- o Each receiver then uses this information, together with an equation for TCP throughput, to derive a TCP-friendly sending rate.
- o Through a distributed feedback suppression mechanism, only a subset of the receivers is allowed to give feedback to prevent a feedback implosion at the sender. The feedback mechanism ensures that receivers reporting a low desired transmission rate have a high probability of sending feedback.
- o Receivers whose feedback is not suppressed report the calculated transmission rate back to the sender in so-called receiver reports. The receiver reports serve two purposes: they inform the sender about the appropriate transmit rate, and they allow the receivers to measure their RTT.
- o The sender selects the receiver that reports the lowest rate as the current limiting receiver (CLR). Whenever feedback with an even lower rate reaches the sender, the corresponding receiver becomes the CLR and the sending rate is reduced to match that receiver's calculated rate. The sending rate increases when the CLR reports a calculated rate higher than the current sending rate.

TFMCC was intended for fixed-size packets with variable rate. SRMP applies it to variable-size SRMP messages that are mostly the same size because the best-effort updates typically all represent the same sort of simulation information and are grouped into bundles of size just under one MTU during periods of heavy network activity. Future developments in TFMCC for variable-size messages will be of high value for inclusion in SRMP if, as expected, they prove to be appropriate for the types of traffic SRMP is intended to support.

SRMP is intended for general use under applications that need its services and may exist in parallel instances on the same host. The UDP port is therefore established ad hoc from available application ports; accordingly, it would not be appropriate to have a well-known port for SRMP.

3. Message Formats

3.1. Bundle Message Format:

```
-----
| bundle header | SRT Message 0 | SRT message 1 | SRT message 2 | ...
-----
```

A bundle is an aggregation of multiple SRMP messages destined for the same multicast address. A bundle can contain only Mode 0 and Mode 1 messages; Mode 2 messages are exchanged using unicast addresses.

SRMP identifies the sender and receiver using their 32-bit Sender_ID, which may be an IPv4 address. For use with IPv6, a user group will need to establish a unique identifier per host. There is no requirement for this identifier to be unique in the Internet; it only needs to be unique in the communicating group.

3.2. Bundle Header Format

0	8	16	24	32
+	+	+	+	+
	Version	Type fb_nr	flag	bundle_SN
+	+	+	+	+
	Sender_ID			
+	+	+	+	+
	Receiver_ID			
+	+	+	+	+
	Sender_Timestamp		Receiver_Timestamp	
+	+	+	+	+
	x_supp		R_max	
+	+	+	+	+
	DSN_count	padding	Length	
+	+	+	+	+
	0 to 255 DSN: <dataID, SN, NoSegs> of this sender			
+	+	+	+	+

Version:

4 bits currently 0010

Type:

4 bits 0000 - indicates bundle

fb_nr:

4 bits feedback round, range 0-15

flag:

4 bits 0001 Is_CLR
other bits reserved

bundle_SN:

16 bits range 0-65535

Sender_Timestamp:

16 bits Representing the time that the bundle was sent out (in
milliseconds) based on the sender's local clock.

Receiver_Timestamp:

16 bits Echo of the Receiver_Time_Stamp field (in milliseconds)
of the receiver feedback message. If the sender has
time delay between receiving the feedback and echoing
the timestamp, it MUST adjust the Receiver_Timestamp
value to compensate.

Receiver_ID
32 bits Unique identifier for the receiver within the multicast group. IPv4 addresses may be used.

Sender_ID:
32 bits Unique identifier for the sender within the multicast group. IPv4 addresses may be used.

X_supp:
16 bits The suppression rate corresponding to the sender, in bits/s. Only those receivers whose desired rate is less than the suppression rate, or whose RTT is larger than R_max, may send feedback information to the sender. The suppression rate is represented as a 16-bit floating point value with 8 bits for the unsigned exponent and 8 bits for the unsigned mantissa.

R_max:
16 bits The maximum of the RTTs of all receivers, in milliseconds. The Maximum RTT should be represented as a 16-bit floating point value with 8 bits for the unsigned exponent and 8 bits for the unsigned mantissa.

DSN_count:
8 bits The count of DSN blocks following the header.

Length:
16 bits Range from 0~65535. The total length of the bundle in octets (including the header).

DSN:
32 bits There can be up to 256 of these in a header. An SRMP implementation MUST support a minimum of 1. Each DSN consists of three fields:

dataID:
16 bits A unique number associated with a particular data element on the sending host, used to identify a Mode 1 message.

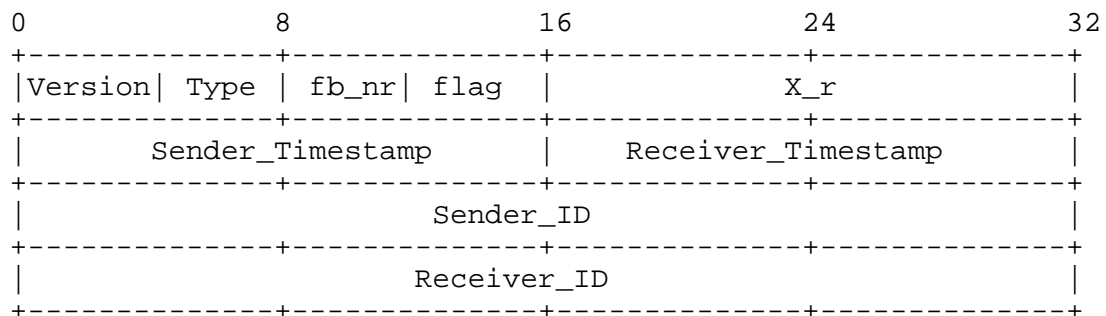
SN:
9 bits Sequence number associated with a particular Mode 1 transmission of a particular dataID.

NoSegs:
7 bits Number of segments, if the dataID was long enough to require segmentation; otherwise 0x0.

Note that the number of DSNs reflects the number of different Mode 1 DataIDs being supported at this time by this instance of SRMP, and is not the count of SRMP messages bundled in this transmission.

Note also that 16-bit timestamps will wrap around in 65536 milliseconds. This should not be a problem unless an RTT is greater than 65 seconds. If a timestamp is less than its predecessor (treating the 16 bits as an unsigned integer), its value must be increased by 65536 for comparisons against the predecessor.

3.3. Feedback Message Format



Version:

4 bits currently 0010

Type:

4 bits value 0001

fb_nr:

4 bits current feedback round of the sender

flag:

4 bits
 0001 - have_RTT
 0010 - have_loss
 0100 - receiver_leave
 other values reserved

X_r:

16 bits desired sending rate X_r in bits/s, calculated by the receiver to be TCP-friendly, 16 bit floating point value with 8 bits for the unsigned exponent and 8 bits for the unsigned mantissa.

Sender_Timestamp:

16 bits Echo of the Sender_Timestamp in bundle header. If the receiver has time delay between receiving the bundle and echoing the timestamp, it MUST adjust the Sender_Timestamp value correspondently.

Receiver_Timestamp:

16 bits The time when the feedback message was sent out from the receiver.

Receiver_ID:

32 bits Unique identifier for the receiver within the multicast group. IPv4 addresses may be used. (Identifies the receiver that sends the feedback message).

Sender_ID:

32 bits Unique identifier for the sender within the multicast group. IPv4 addresses may be used. (Identifies the sender that is the destination of the current feedback message.)

3.4. SRT Mode 0 Header Format

0	8	16	24	32
+-----+-----+-----+-----+				
Version	Type	000	00000000	Length
+-----+-----+-----+-----+				

Version:

4 bits currently 0010

Type:

4 bits 0000

Mode:

3 bits 000

Padding:

8 bits 00000000

Length:

11 bits Length of the payload data in octets (does not include the header).

3.5. SRT Mode 1 Header Format

0	8	16	24	32
+-----+-----+-----+-----+				
Version	Type	001	SegNo	Length
+-----+-----+-----+-----+				
	DSN			
+-----+-----+-----+-----+				

Version:

4 bits currently 0010

Type:

4 bits 0000

Mode:

3 bits 001

SegNo:

7 bits The index number of this segment.

Length:

14 bits Length of the payload data in octets (does not include the header).

DSN:

32 bits Same as in the bundle header. Note that this contains NoSegs, whereas SegNo is a separate element.

3.6. SRT Mode 2 Header Format

0	8	16	24	32
+-----+-----+-----+-----+				
Version	Type	010 00000		Length
+-----+-----+-----+-----+				
	SN			
+-----+-----+-----+-----+				

Version:

4 bits currently 0010

Type:

4 bits 0010

Mode:

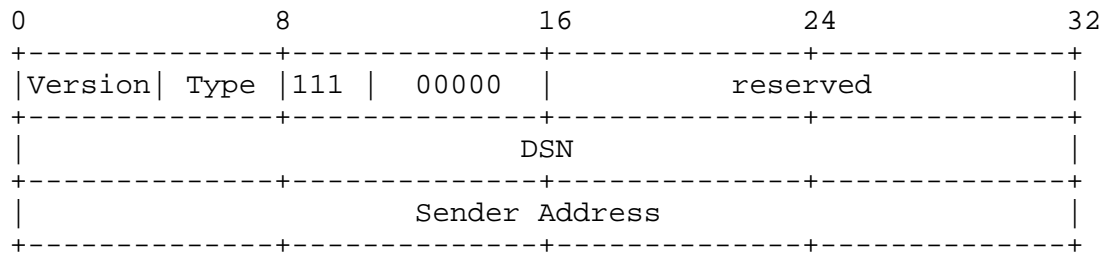
3 bits 010

Padding:
5 bits 00000

Length:
16 bits Length of the payload data in octets (does not the include header).

SN:
32 bits Same as in bundle header.

3.7. SRT NACK Format



Version:
4 bits currently 0010

Type:
4 bits 0010

Mode:
3 bits 111

Padding:
5 bits 00000

Reserved:
16 bits

DSN:
32 bits sequence number

Sender Address:
The IP address of the sender of the message being NACKed.

3.8. User-Configurable Parameters

Name	Minimum Value	Recommended Value	Units
DSN_Max	1	32	messages
dataID_Timeout	none	none	ms
Segment_Timeout	50	250	ms
Bundle_Timeout	1	10	ms
Heartbeat_Interval	1	none	s
Mode2_Max	1	none	messages
ACK_Threshold	none	worst RTT in group	ms

4. TFMCC Operation

4.1. TCP Rate Prediction Equation for TFMCC

The RECOMMENDED throughput equation for SRMP is a slightly simplified version of the throughput equation for Reno TCP from [5]:

$$X = \frac{8*s}{R * (\text{sqrt}(2*p/3) + (3*\text{sqrt}(6*p) * p * (1+32*p^2)))} \quad (1)$$

(the formula may be simplified for implementation), where

X is the transmit rate in bits/second.

s is the message size in bytes.

R is the round-trip time in seconds.

p is the loss event rate, between 0.0 and 1.0, of the number of loss events as a fraction of the number of messages transmitted.

In the future, different TCP formulas may be substituted for this equation. The requirement is that the throughput equation be a reasonable approximation of the sending rate of TCP for conformant TCP congestion control.

4.2. Bundling

Multiple SRMP messages will be encapsulated into a bundle. When a new SRMP message (Mode 0 or Mode 1) arrives, the SRMP daemon will try to add the new message into the current bundle.

The SRMP daemon MUST keep a timer, which will be reset when the first SRMP message is added into the bundle. After Bundle_Timeout, the timer will time out, and the current bundle should be transmitted

immediately. A new bundle will then be initialized to hold new SRMP messages. Bundle_Timeout SHALL NOT be less than 1 ms. The recommended value is 10 ms.

Also, the bundle length MUST NOT exceed LENGTH_MAX. If adding a new SRMP message will produce a greater length, the SRMP daemon MUST initialize a new bundle for the new SRMP messages, and the current bundle should be transmitted immediately. The recommended value for LENGTH_MAX is 1454 bytes (Ethernet MTU minus IP and UDP header lengths).

In a bundle, there may exist multiple SRMP messages with the same dataID. In this case, only the latest version of that dataID is useful. SRMP may check for duplicate dataIDs in the same bundle and delete all but the latest one. If a Mode 1 message appears in the outgoing bundle, then the corresponding DSN should not appear in the bundle header.

The bundle header contains the DSN <dataID,SN,NoSegs> for Mode 1 messages from this sender. The absolute maximum number of DSN is 255; however, an implementation may apply a user-specified DSN_Max, no smaller than 1. An implementation may support a user-defined dataID_Timeout, after which a given dataID will not be announced in the bundle header unless a new Mode 1 message has been sent. If the sender has more dataIDs sent (and not timed out) than will fit in the bundle header, the DSNs MUST be announced on a round-robin basis, with the exception that no bundle header will announce a DSN for a Mode 1 message contained within that bundle. If a duplicate DSN is received, it may be silently discarded.

4.3. Congestion Control

The congestion control mechanism operates as described in [7].

4.4. Any-Source Multicast

SRMP uses the Any-Source Multicast Mode. Each sender will determine its maximum RTT, suppression data rate, and sending rate with respect to each sender. Each receiver will measure its RTT and desired rate to each sender in the group, and send feedback to every sender by sending to the multicast group.

4.5. Multiple Sources

Under SRMP, each group member in a multicast group is a sender as well as a receiver. Each receiver may need to participate in TFMCC information exchange with all senders. Thus, when a receiver sends a

feedback message, it must identify to which source the message should be sent using the "Sender ID" field in the header.

The feedback is multicast to the group. Depending on the network situation, senders may select different receivers to provide feedback. Feedback messages from receivers that are not among those selected by the local TFMCC to provide feedback should be silently discarded.

4.6. Bundle Size

TFMCC is designed for traffic with a fixed message size. The maximum bundle size (including header) for SRMP is set to a configurable maximum, typically 1454 bytes (Ethernet MTU minus IP and UDP header lengths). The bundle size will be used in a TCP throughput equation, to get a desired source rate. However, in SRMP, the message size is variable because:

1. After bundle time out, the current bundle will not wait for new SRMP messages. This happens with sources sending at a slow rate.
2. In long messages, there is no further space in the current bundle for new SRMP messages. This will happen with sources sending at a high rate or sending messages with a length over half of the bundle payload size.

The case 1 bundle size is likely to be much smaller than that of case 2.

Therefore, in SRMP, the mean value of the 10 most recent bundles' sizes will be used as the bundle size in the TCP throughput equation. This mean value is independent from the network condition and reflects current activity of the source.

4.7. Data Rate Control

Each host will have a single instance of SRMP supporting all of its applications. Thus, the sender's source rate is the sum of the rates of all the clients of the same multicast group.

If the source rate is larger than the sender's desired transmission rate, it is the sender's responsibility to do traffic shaping. Any method that conforms to the target sending rate may be used. The RECOMMENDED method is to randomly discard enough Mode 0 messages to meet the target rate.

4.8. Mode 1 Loss Detection

Bundle header processing includes checking each DSN in the bundle header and scheduling a NACK for each DSN bearing a dataID for which some application has indicated interest, if the SN/SegNo in that DSN indicates that a NACK is needed. NACKs are sent in bundles and may be bundled with data messages. A NACK is required if:

- o the SN is one or more greater (mod 512) than the latest received Mode 1 message for that dataID, or
- o the SegNo has not been received, some segment of the <dataID,SN> has been received, and a user-defined Segment_Timeout, which SHALL NOT be less than 50 ms, has expired since receipt of the first SegNo for the <dataID,SN>.

The bundling sublayer will pass the DSN list in any received bundle header to the SRT sublayer. It also will suppress NACKs in outgoing bundles, as described in the next section.

4.8.1. Sending a Negative Acknowledgement

Negative acknowledgements are used by SRMP for multicast messages in order to avoid the congestion of an "ACK implosion" at the original sender that would likely occur if positive acknowledgements were used instead. However, with a large multicast group spread out over a congested wide-area network, there is the potential for enough members of the multicast group to fail to receive the message and generate NACKs to cause considerable congestion at the original sender despite the use of negative acknowledgements instead of positive acknowledgements. For this reason, SRMP uses a NACK suppression mechanism to reduce the number of NACKs generated in response to any single lost message.

The NACK suppression mechanism uses the Bundle_Timeout to distribute NACKs over an appropriate time window. This assumes that the user has selected a bundle timeout appropriate for the needs of the application for real-time responsiveness.

When the bundling sublayer is ready to send a bundle, it removes from the bundle any NACKs for which a response has been sent by another member of the multicast group within the NACK_Repeat_Timeout window. If the original Bundle_Timeout has not expired, transmission of the bundle may then be delayed until the original Bundle_Timeout expires or the bundle is full, whichever happens first.

4.9. Unbundling

After a receiver completes congestion control processing on a bundle, it parses the bundle into SRT messages and sends these to the SRT sublayer.

4.10. Heartbeat Bundle

SRMP implementations may support a user-defined Heartbeat_Interval, which SHALL NOT be less than one second. At the end of each heartbeat interval, if the sender has not sent any bundle, an empty bundle will be sent in order to trigger Mode 1 loss detection.

5. SRT Operation

SRMP operates in three distinct transmission modes in order to deliver varying levels of reliability: Mode 0 for multicast data that does not require reliable transmission, Mode 1 for data that must be received reliably by all members of a multicast group, and Mode 2 for data that must be received reliably by a single dynamically determined member of a multicast group.

Mode 0 operates as a pure best-effort service. Mode 1 operates with negative acknowledgements only, triggered by bundle arrivals that indicate loss of a Mode 1 message. Mode 2 uses a positive acknowledgement for each message to provide reliability and low latency. Mode 2 is used where a transaction between two members of a multicast group is needed. Because there can be many members in such a group, use of a transaction protocol, with reliability achieved by SRMP retransmission, avoids the potentially large amount of connection setup and associated state that would be required if each pair of hosts in the group established a separate TCP connection.

Use of SRMP anticipates that only a small fraction of messages will require reliable multicast, and a comparably small fraction will require reliable unicast. This is due to a property of distributed virtual simulation: the preponderance of messages consist of state update streams for object attributes such as position and orientation. SRMP is unlikely to provide effective reliable multicast if the traffic does not have this property.

In SRMP, "dataID" is used to associate related messages with each other. Typically, all messages with the same dataID are associated with the same application entity. All the messages with the same dataID must be transmitted in the same mode. Among all the messages with the same dataID, the latest version will obsolete all older messages.

5.1. Mode 0 Operation

Mode 0 is for multicast messages that do not require reliable transmission because they are part of a real-time stream of data that is periodically updated with high frequency. Any such message is very likely to have been superceded by a more recent update before retransmission could be completed.

5.1.1. Sending Mode 0 Messages

When an application requests transmission of Mode 0 data, a destination multicast group must be provided to SRMP along with the data to be sent. After verifying the data length and multicast group, the following steps **MUST** be performed by the SRT sublayer:

1. An SRT message **MUST** be generated with the following characteristics:

the version is set to the current version, the message type is set to 0x0, the mode is set to 0x0. User data is included after the message header. If the message cannot be generated as described above, the user data is discarded and the error **MUST** be reported to the application.

2. If step 1 was completed without error, the newly generated message **MUST** be sent to the bundling sublayer. The implementation **MUST** report to the application whether the message was ultimately accepted by UDP.

5.1.2. Receiving Mode 0 Messages

When a Mode 0 message is received by SRMP, it **MUST** be processed as follows: after verifying the version, message type, and destination multicast address fields, the user data **MUST** be delivered to all applications that are associated with the multicast group in the message. If the SRMP receiver has never received any Mode 1 messages before the Mode 0 message is received, the Mode 0 message should be silently discarded.

It is **RECOMMENDED** that the following information be provided to the receiving applications: message body, multicast address.

5.2. Mode 1 Operation

Mode 1 is for multicast data that requires reliable transmission. A Mode 1 message can be either a data message or a NACK. Mode 1 data messages are expected to be part of a data stream. This data stream is likely to contain Mode 0 messages as well (see section 5.1.1), but

it is possible for a data stream to be comprised solely of Mode 1 messages.

5.2.1. Sending Mode 1 Data Messages

After the data length, dataID, and destination multicast group are verified, SRT MUST take the following steps:

1. If the message will not fit in an empty bundle with DSN_Max DSN in the header, the message MUST be segmented. The remaining steps pertain to each segment of the message. Each segment receives a unique SegNo, starting with 0 and ending with (NoSegs-1).
2. An SRT message is generated with the following characteristics: the version is set to 0x02, the message type is set to 0x0, the transmission mode is set to 0x01, the SN is set equal to the SN of the most recently sent Mode 1 complete message of the same dataID, incremented by 1 modulo 512. If no such Mode 1 message exists, the SN is set to 0x0.
3. The newly generated message (all segments) must then be buffered, replacing any formerly buffered Mode 1 message of the same dataID, destination multicast address. If the message cannot be buffered, the user data is discarded and the error is reported to the application.
4. If step 2 was completed without error, the newly generated message is sent to the TFMCC sublayer.

5.2.2. Receiving Mode 1 Data Messages

When a Mode 1 data message is received by SRT, it will be processed as follows (assuming that the version field has already been verified to be 0x02):

1. The destination address MUST be verified to be a valid IP multicast address on which this instance of SRMP is a member. If this is not the case, the message should be silently discarded.
2. The destination address MUST be verified to be one for which some application has indicated interest. Otherwise, the message should be silently discarded.
3. The SN, SegNo, source_ip_address, and the body of the received message MUST be buffered, and the user data MUST then be delivered to all applications that have indicated interest in the multicast group of the received message.

4. When a new DSN value is received with NoSegs greater than zero, a timer should be set for Segment_Timeout, after which a NACK should be sent to the bundling sublayer and the timer should be restarted for Segment_Timeout.
5. If NoSegs in the received message is not 0, a reassembly process MUST be started. Each segment MUST be buffered. If receipt of the current message completes the segment, the reassembled message MUST be released to the application and the Segment_Timeout timer cancelled.
6. If a new DSN is received before all segments of the previous DSN are received, the segments that have been received should be dropped silently.
7. It is RECOMMENDED that the following information be provided to the receiving applications: message body, dataID, source_ip_address, multicast_group address.
8. When a client signs on to a new multicast group, all locally buffered Mode 1 messages related to that multicast group should be delivered to the client immediately.

5.2.3. Sending a Negative Acknowledgement

Whenever a bundle is received, the bundling sublayer will forward the DSN list from the bundle header to the SRT sublayer. The SRT sublayer will examine buffered values of <SenderID,dataID,SN,SegNo> to determine whether a NACK is required. If so, it will generate a NACK message and send it to the bundling sublayer. The NACK message will have version set to 0x2, message type set to 0x2, and transmission mode set to 0x7. dataID, SN, and destination address are set to that of the Mode 1 message for which the NACK is being sent. If a NACK has been received from any member of the destination multicast group for the Mode 1 message in question within the NACK threshold, no NACK is generated.

For segmented messages, there are two possible types of NACKs:

- o Based on the DSN list in the bundle header, the SRT implementation may determine that an entire segmented Mode 1 message was lost. In this case, the NACK MUST carry SegNo=0x7F (all in one field).
- o Based on the Segment Timeout, the SRT implementation may determine that one or more segments of a message have not been delivered. In this case, a NACK will be sent for each missing segment.

5.2.4. Receiving a Negative Acknowledgement

When a NACK is received by SRT, it MUST be processed as follows, after verifying the multicast address, dataID, source IP address, and transmission mode:

1. If this instance of SRT's most recent Mode 1 message of the dataID indicated in the NACK has an SN newer than the SN in the NACK, that message (which is buffered) should be immediately retransmitted to the multicast address indicated in the received NACK. If the most recent Mode 1 message has an SN equal to the SN indicated in the NACK, and if the SegNo field in the NACK contains 0x7F, all segments of the buffered Mode 1 message MUST be retransmitted; if the SegNo has some other value, only the indicated segment should be retransmitted.
2. Whether or not step 1 results in the retransmission of a message, the event of receiving the NACK and the (local machine) time at which the NACK was received should be buffered. Each instance of SRT MUST buffer the number of NACKs that have been received for each dataID-multicast address pair, since the most recent Mode 1 message of the same pair was received and the time at which the most recent of these NACKs was received.

5.3. Mode 2 Operation

Mode 2 is for infrequent reliable transaction-oriented communication between two dynamically determined members of a multicast group. TCP could be used for such communication, but there would be unnecessary overhead and delay in establishing a stream-oriented connection for a single exchange of data, whereas there is already an ongoing stream of best-effort data between the hosts that require Mode 2 transmission. An example is a Distributed Interactive Simulation (DIS) collision PDU.

5.3.1. Sending Mode 2 Data Messages

When an application requests transmission of Mode 2 data, a dataID and a destination unicast IP address MUST be provided to SRT along with the data to be sent. After verifying the data length, dataID, and destination address, SRT MUST perform the following steps:

1. An SRT message is generated with the following characteristics: the version is set to 0x02, the message type is set to 0x02, the transmission mode is set to 0x2, the dataID is set to the application-provided value, and the destination address is set to the application-provided IP address. The SN is set equal to the SN of the most recently sent Mode 2 message of the same dataID

incremented by 1 modulo 65536. If no such Mode 1 message exists, it is set to 0x0.

2. The newly generated message is buffered. This new message does not replace any formerly buffered Mode 2 messages. An implementation MUST provide a Mode 2 message buffer that can hold one or more Mode 2 messages. Mode 2 messages are expected to be infrequent (less than 1 percent of total traffic), but it is still strongly RECOMMENDED that an implementation provide a buffer of user-configurable size Mode2_Max that can hold more than a single Mode 2 message. If the message cannot be buffered, the user data is discarded and the error MUST be reported to the application. If the message can be buffered, it should be sent to UDP immediately after being buffered.
3. If step 2 was completed without error, the newly generated message MUST be sent to the IP address contained in its destination address field, encapsulated within a UDP datagram. If the UDP interface on the sending system reports an error to SRT when the attempt to send the SRT message is made, an implementation may attempt to resend the message any finite number of times. However, every implementation MUST provide a mode in which no retries are attempted. Implementations should default to this latter mode of operation. The implementation MUST report to the application whether the message was ultimately accepted by UDP.
4. If some user-configurable "ACK_Threshold" (which should be greater than the worst-case round-trip time for the multicast group) elapses without receipt of an ACK for the Mode 2 message, it is retransmitted. An implementation may define a maximum number of retransmissions to be attempted before the Mode 2 message is removed from the buffer.

5.3.2. Receiving Mode 2 Data Messages

When a Mode 2 data message is received by SRT, it should be processed as follows after verifying version, dataID, sender address, and SN:

1. For Mode 2 messages, the sequence number field is used to associate the required positive acknowledgement with a specific Mode 2 message. If the message passes verification, the encapsulated user data is delivered to all applications that have indicated interest in the dataID and multicast address of the received message, regardless of the value of the SN field.
2. Additionally, an ACK MUST be sent to the host from which the Mode 2 data message originated. See section 5.3.3. below for details.

5.3.3. Sending a Positive Acknowledgement

A positive acknowledgement (ACK) is triggered by the receipt of a Mode 2 data message. To send an ACK, a new SRT message is generated with version set to 0x02, message type set to 0x2, and transmission mode set to 0x2. The dataID and SN are those of the Mode 2 data message being acknowledged. The destination address field is set to the source IP address from which the data message was received. Since Mode 2 data messages are unicast, there is little concern about an ACK implosion causing excessive congestion at the original sender, so no suppression mechanism is necessary.

5.3.4. Receiving a Positive Acknowledgement

When an ACK is received by SRT, after verifying the transmission mode, dataID, and source IP address against outstanding Mode 2 transmission, SRT MUST remove the pending transmission from its buffer.

6. RFC 2357 Analysis

This section provides answers to the questions posed by RFC 2357 for reliable multicast protocols, which are quoted.

6.1. Scalability

"How scalable is the protocol to the number of senders or receivers in a group, the number of groups, and wide dispersion of group members?"

SRMP is intended to scale at least to hundreds of group members. It has been designed not to impose limitations on the scalability of the underlying multicast network. No problems have been identified in its mechanisms that would preclude this on uncongested networks.

"Identify the mechanisms which limit scalability and estimate those limits."

There is a practical concern with use of TFMCC, in that the receiver with the most congested path constrains delivery to the entire group. Distributed virtual simulation requires data delivery at rates perceived as continuous by humans. Therefore, it may prove necessary to assign such receivers to different, lower-fidelity groups as a practical means of sustaining performance to the majority of participating hosts. SRMP does not have a mechanism to support such pruning at this time.

6.2. Congestion

"How does the protocol protect the Internet from congestion? How well does it perform? When does it fail? Under what circumstances will the protocol fail to perform the functions needed by the applications it serves? Is there a congestion control mechanism? How well does it perform? When does it fail?"

Both simulations and tests indicate that SRMP with TFMCC displays backoff comparable to that of TCP under conditions of significant packet loss. The mechanism fails in a network-friendly way, in that under severe congestion, it reduces sending of the best-effort traffic to a very small rate that typically is unsatisfactory to support a virtual simulation. This is possible because the reliable traffic typically is a small percentage of the overall traffic and SRMP is NACK oriented, with NACK suppression, so that reliable traffic loss adds little traffic to the total. If the traffic mix assumption is not met, the reliable traffic (which does not back off under increased RTT) could produce a higher level of traffic than a comparable TCP connection. However, levels of reliable traffic this large are not in the intended application domain of SRMP.

"Include a description of trials and/or simulations which support the development of the protocol and the answers to the above questions."

SRMP has been simulated using a discrete event simulator developed for academic use [8]. The design assumptions were validated by the results. It also has been emulated in a LAN-based cluster and application-tested in a wide-area testbed under its intended traffic mix (distributed virtual simulation) and using a traffic generator with losses emulated by random dropping of packets [9].

"Include an analysis of whether the protocol has congestion avoidance mechanisms strong enough to cope with deployment in the Global Internet, and if not, clearly document the circumstances in which congestion harm can occur. How are these circumstances to be prevented?"

Because it provides sending backoff comparable to TCP, SRMP is able to function as well as TCP for congestion avoidance, even in the Global Internet. The only way an SRMP sender can generate congestion is to use the protocol for unintended purposes, for example, reliable transmission of a large fraction of the traffic. Doing this would produce unsatisfactory results for the application, as SRMP's mechanism for providing reliability will not function well if the best-effort traffic does not constitute the majority of the total traffic.

"Include a description of any mechanisms which contain the traffic within limited network environments."

SRMP has no such mechanisms, as it is intended for use over the open Internet.

"Reliable multicast protocols must include an analysis of how they address a number of security and privacy concerns."

See section 7 below.

7. Security Considerations

As a transport protocol, SRMP is subject to denial of service by hostile third parties sending conflicting values of its parameters on the multicast address. SRMP could attempt to protect itself from this sort of behavior. However, it can be shielded from such attacks by traffic authentication at the network layer, as described below. A comparable level of authentication also could be obtained by a message using MD5, or a similar message hash in each bundle, and using the SRMP bundle header to detect duplicate transmissions from a given host. However, this would duplicate the function of existing network layer authentication protocols.

Specific threats that can be eliminated by packet-level authentication are as follows:

- a. Amplification attack: SRMP receivers could be manipulated into sending large amounts of NACK traffic, which could cause network congestion or overwhelm the processing capabilities of a sender. This could be done by sending them faked traffic indicating that a reliable transmission has been lost. SRMP's NACK suppression limits the effect of such manipulation. However, true protection requires authentication of each bundle.
- b. Denial-of-service attack: If an SRMP sender accepts a large number of forged NACKs, it will flood the multicast group with repair messages. This attack also is stopped by per-bundle authentication.
- c. Replay attack: The attacker could copy a valid, authenticated bundle containing a NACK and send it repeatedly to the original sender of the NACKed data. Protection against this attack requires a sequence number per transmission per source host. The SRMP bundle header sequence number would satisfy this need. However, the SN also can be applied at a lower layer.

- d. Reverse path forwarding attack (spoofing): If checks are not enabled in all network routers and switches along the path from each sender to all receivers, forged packets can be injected into the multicast tree data path to manipulate the protocol into sending a large volume of repairs. Packet-level authentication can eliminate this possibility.
- e. Inadvertent errors: A receiver with an incorrect or corrupted implementation of TFMCC could respond with values of RTT that might stimulate a TFMCC sender to create or increase congestion in the path to that sender. It is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the Session Description needed to join the session. How receivers identify themselves as legitimate is outside the scope of this document.

The required authentication could become part of SRMP or could be accomplished by a lower layer protocol. In any case, it needs to be (1) scalable and (2) not very computationally demanding so it can be performed with minimal delay on a real-time virtual simulation stream. Public-key encryption meets the first requirement but not the second. Using the IPsec Authentication Header (AH) (RFC 4302 [3]) meets the second requirement using symmetric-key cryptography. See RFC 4302 [3] for guidance on multicast per-packet authentication. In practice, users of distributed simulation are likely to work over a (possibly virtual) private network and thus will not need special authentication for SRMP.

8. List of Acronyms Used

ACK	- positive acknowledgement
AH	- Authentication Header
CLR	- current limiting receiver
IPSEC	- Internet Protocol Security
MTU	- maximum transmission unit
NACK	- negative acknowledgement
RTT	- round-trip time
SA	- security association
SRMP	- Selectively Reliable Multicast Protocol
SRT	- Selectively Reliable Transport
TFMCC	- TCP-Friendly Multicast Congestion Control

9. Contributions

We gratefully acknowledge the significant contributions of two people without whom this RFC would not have been developed. Vincent Laviano created the first specification and implementation of SRMP (at that time called SRTP). Babu Shanmugam employed SRMP in a sizable distributed virtual simulation environment, where he revised the implementation and helped revise the design to support distributed virtual simulation workload effectively.

10. References

10.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] J. Widmer, M. Handley, Extending Equation-Based Congestion Control to Multicast Applications, ACM SIGCOMM Conference, San Diego, August 2001. <<http://www.sigcomm.org/sigcomm2001/p22-widmer.pdf>>
- [3] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

10.2. Informative References

- [4] Pullen, M., Myjak, M., and C. Bouwens, "Limitations of Internet Protocol Suite for Distributed Simulation the Large Multicast Environment", RFC 2502, February 1999.
- [5] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proceedings of ACM SIGCOMM 1998.
- [6] Mankin, A., Romanow, A., Bradner, S., and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, June 1998.
- [7] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.
- [8] J. M. Pullen, "The Network Workbench: Network Simulation Software for Academic Investigation of Internet Concepts," Computer Networks Vol 32 No 3 pp 365-378, March 2000.

- [9] J. M. Pullen, R. Simon, F. Zhao and W. Chang, "NGI-FOM over RTI-NG and SRMP: Lessons Learned," Proceedings of the IEEE Fall Simulation Interoperability Workshop, paper 03F-SIW-111, Orlando, FL, September 2003.
- [10] D. Cohen, "NG-DIS-PDU: The Next Generation of DIS-PDU (IEEE-P1278)", 10th Workshop on Standards for Interoperability of Distributed Simulations, March 1994.
- [11] Handley, M., Floyd, S., Whetten, B., Kermode, R., Vicisano, L., and M. Luby, "The Reliable Multicast Design Space for Bulk Data Transfer", RFC 2887, August 2000.
- [12] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., and J. Crowcroft, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 3450, December 2002.
- [13] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., Handley, M., and J. Crowcroft, "Layered Coding Transport (LCT) Building Block", RFC 3451, December 2002.
- [14] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "Forward Error Correction (FEC) Building Block", RFC 3452, December 2002.
- [15] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast", RFC 3453, December 2002.

Authors' Addresses

J. Mark Pullen
C4I Center
George Mason University
Fairfax, VA 22030
USA

EMail: mpullen@gmu.edu

Fei Zhao
C4I Center
George Mason University
Fairfax, VA 22030
USA

EMail: fzhao@netlab.gmu.edu

Danny Cohen
Sun Microsystems
M/S UMPK16-160
16 Network Circle
Menlo Park, CA 94025
USA

EMail: danny.cohen@sun.com

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

