Internet Draft: RMTP-II Specification
Document: draft-whetten-rmtp-ii-00.txt
Expires: Six months

B. Whetten, M. Basavaiah
GlobalCast Communications, Inc.
S. Paul
Lucent Technologies, Bell Labs
T. Montgomery
West Virginia University
N. Rastogi, J. Conlan, T. Yeh
GlobalCast Communications, Inc.
April 8, 1998

THE RMTP-II PROTOCOL

Status of this Memo

Abstract

The Reliable Multicast Transport Protocol II, RMTP-II, is a reliable multicast protocol, designed to reliably and efficiently send data from a few senders to large groups of simultaneous recipients. It is designed primarily for use over controlled network topologies. It works over both symmetric networks, as well as over asymmetrical network topologies such as those provided by satellite, cable modem or Asymmetrical Digital Subscriber Line (ADSL) carriers. Before sending, each sender must connect with a trusted Top Node, to receive permission and control parameters for its data stream. The top node provides network managers with a single point of control for the senders, allowing them to monitor and control the traffic being sent.

Table of Contents

1 INTRODUCTION 4
2 ENTITIES 6
2.1 Node 6
2.2 RMTP-II Tree 6
2.3 Multicast Group Address 6
2.4 Data Channels 6
2.5 Control Channels 6
2.6 Data Stream 7
2.7 StreamID 7
2.8 Packet Sequence Numbers 7
2.9 Data Queue 7
2.10 RMTP-II Packets 7
3 NODE FUNCTIONS 10
3.1 Sender Node 10
3.2 Receiver Node 11
3.3 Top Node 12
3.4 Aggregator Node 13
3.5 Designated Receiver Node 13
4 OPERATION OF THE RMTP-II PROTOCOL 14
4.1 Basic Operation of the Protocol 14
4.2 Global Tree Parameters 15
4.3 Data and Control Paths 16
4.4 Data Transmission 16
4.5 Retransmission 16
4.6 Options 17
4.7 Tree Connections 18
4.8 Flow and Congestion Control 18
4.9 Membership 20
4.10 Fault Detection and Recovery 20
4.11 Quality of Service (QoS) 21
4.12 SNMP Support 22
4.13 Session Manager Support 22
4.14 Forward Error Correction 22
5 DETAILS: 23
5.1 RMTP-II Global Parameters 23
5.2 Tree Configuration 24
5.3 Tree Membership Algorithms 26
5.4 Data Transmission 28
5.5 Data Reception 32
5.6 HACK Generation 34
5.7 Retransmissions in response to a HACK 39
5.8 NACK Generation 40
5.9 NACK Response 41
5.10 Congestion Control 41
5.11 Failure Detection and Recovery 41
5.12 Control tree discovery 44

5.13 Control Tree Round Trip Times 46
6 RMTP-II PACKET FORMATS 48
6.1 RMTP-II Fixed Header 48
6.2 Data 50
6.3 Retransmission Packet 52
6.4 HACK 54
6.5 JoinConfirm 56
6.6 JoinStream 58
6.7 JoinAck 59
6.8 LeaveStream 60
6.9 Heartbeat 61
6.10 HeartbeatResponse 62
6.11 NullData 63
6.12 Eject 64
6.13 EOS 65
6.14 LeaveConfirm 66
6.15 RMTP-II Options Header 67
7 ACKNOWLEDGEMENTS 81
8 REFERENCES 83

1 Introduction

IP Multicast (RFC 1112) provides highly efficient delivery of information to many receivers at once, with each packet going over any given link no more than a single time. To date, there are no standardized, reliable multicast protocols that provide the equivalent of TCP for IP Multicast. It is widely recognized that, unlike TCP, no single reliable multicast protocol can meet the needs of all application types over all network types.

The Reliable Multicast Transport Protocol II, RMTP-II, is a reliable multicast protocol which reliably and efficiently sends data from a few senders to large groups of simultaneous recipients. RMTP-II works over all types of networks, including symmetric networks as well as asymmetrical networks such as those provided by satellite, cable modem or Asymmetrical Digital Subscriber Line (ADSL) carriers. RMTP-II requires some configuration of topology.

The objectives of RMTP-II are guaranteed reliability, high throughput, and low end-to-end delay on any network topology, while providing the network manager with control over transmission traffic.

RMTP-II is based on a hierarchical tree structure in which receivers are grouped into local regions. In each region a special control node is responsible for maintaining receiver membership and for aggregating the acknowledgments from the receivers in its region and forwarding them to the sender. The control node may also take responsibility for retransmitting dropped packets to the local receivers.

RMTP-II is a sender-reliable multicast protocol which uses a fully distributed membership protocol to keep track of the current membership of the tree. This allows senders to determine when packets become stable and may be deleted. RMTP-II provides strong guarantees on packet delivery and gives the senders a count of the number of receivers that successfully received each packet. It also supports optional negative acknowledgements of missed packets for faster recovery of data and lower control traffic on low loss networks.

The Internet is highly dependent on the TCP congestion control mechanisms which allow all streams to share bandwidth fairly. Widespread deployment of a transport protocol that does not interact gracefully with the TCP protocol has the potential to do significant damage to the Internet. This has been an obstacle to the standardization of a reliable multicast protocol. RMTP-II is designed to interact closely with congestion control algorithms and provide these algorithms with constant information about the loss

rates and round trip times in the tree. The RMTP-II congestion control algorithms are under development and will be presented in a companion document.

RMTP-II complements its congestion control algorithms with additional centralized management control over all RMTP-II streams running on the network. RMTP requires senders to interact with a trusted Top Node and accept configuration information from this node. The Top Node provides the network manager with an SNMP interface for monitoring and controlling the streams with which it is associated. The network manager can place bandwidth limits on each stream and can specify the congestion control parameters that each sender must use. This provides graceful, controlled deployment of reliable multicast that protects the network both through explicit management and automatic congestion control policies.

RMTP-II requires some topology configuration which is left to an external Session Manager. A Session Manager can be implemented as either a centralized tool controlled by the network manager, as static configuration files, or as a set of fully distributed algorithms. The Session Manager is also responsible for optional session advertisements and optional total group membership tracking.

RMTP-II provides the following additional features:

- Optional, integrated Forward Error Correction (FEC)is provided.

- Receivers may join or leave a stream already in progress.

- SNMP support is furnished all nodes.

- Senders and receivers may be ejected.

- Many-to-many multicast or special upgrades to routers is not required.

- Time Bounded Reliability is provided for bounded recovery of data for synchronous real-time streaming applications.

- A fault tolerant top node is supported.

- Dynamic fault detection and recovery are supported.

- The rate of generation of acknowledgement traffic can be controlled.

Appendix A contains a design rationale for RMTP-II.

2 Entities

2.1 Node

A node is an entity with a network address which provides transmission, control, or reception services for the network.

There are five types of nodes: sender nodes which transmit data, receiver nodes which receive data, and three types of control nodes which provide acknowledgment, aggregation and communication services for control functions. The control nodes are: aggregator nodes which aggregate and forward control information, designated receiver nodes which function as aggregators and also receive and retransmit data, and top nodes which perform aggregation and other control functions at the top of the RMTP communication tree. Node functionality is described later.

2.2 RMTP-II Tree

An RMTP-II tree is a collection of nodes connected into a tree-like communication network with the top node at the root. The primary function of the tree is to efficiently distribute acknowledgement packets.

2.3 Multicast Group Address

A multicast group address is a pair consisting of an IP multicast address and a UDP port number. Multicast group addresses are used for sending data to receivers, and for sending control information to local groups.

2.4 Data Channels

A data channel is a communication path used to send data streams. Each data channel is associated with a multicast group address used for sending and receiving the data streams. A sender sends a data stream on a data channel. Receivers must join the data channel to receive the data stream.

One RMTP-II Tree supports multiple data channels. A single data channel can be used by one or more senders to send data. A receiver must join all data channels corresponding to the streams it wishes to receive.

2.5 Control Channels

A control channel is a communication path used to send control information. A control channel may be associated with a unicast IP

address and port or may be associated with a multicast group address.

2.6 Data Stream

A data stream is a sequence of Data packets having a unique StreamID and a data channel.

2.7 StreamID

A StreamID is a 16-bit number, chosen either by the application that creates the stream or by RMTP-II. Senders and receivers use the StreamID to distinguish data streams. A sender may specify a StreamID in the range from 32768 (2^15) to 65535 (2^16)-1. Numbers in the range from 0 to 32768 are reserved. If a sender specifies 0 as the StreamID, then RMTP-II assigns a unique StreamID in the range from 1 to 32768. If a sender application specifies a StreamID that is in use, RMTP-II will not allow the sender to join the RMTP tree.

2.8 Packet Sequence Numbers

A packet sequence number is a 32 bit number in the range from 1 through 2^32 - 1 which is used to specify the sequential order of a Data packet in a stream of Data packets. A sender node assigns consecutive sequence numbers to the Data packets provided by the sender application for the data stream. Zero is reserved.

2.9 Data Queue

A data queue is a buffer, maintained by a sender or a designated receiver node, for transmission and retransmission of the Data packets provided by the sender application. New Data packets are added to the data queue as they arrive from the application, up to a specified buffer limit. The admission rate of packets to the network is controlled by congestion control algorithms. Once a packet has been received by its target recipients, it may be deleted from the buffer under appropriate conditions.

2.10 RMTP-II Packets

All RMTP-II packets consist of a fixed header, optional option headers, followed by data or control information.

Data is carried by Data packets and Retransmission packets. Control information is carried in the following types of control packets: HACK, NACK, JoinStream, JoinAck, JoinConfirm, LeaveStream, Heartbeat, HeartbeatResponse, NullData, Eject, EOS, and LeaveConfirm.

2.10.1 Data Packet

The sender application provides Data packets to the sender node. The sender node assigns consecutive sequence numbers to the Data packets and multicasts the packets on the data channel.

Each Data packet has a StreamID and a sequence number which identify the packet and allow a receiver application to reconstruct the data stream from the Data packets. A receiver learns about Data packets that have not yet arrived by receiving a packet later in the sequence or from a NullData packet which mentions later packets.

A Data packet is said to be "stable", if it has been acknowledged as received by all its target recipients.

2.10.2 Retransmission Packet

Retransmitted data is sent in Retransmission packets. A separate packet type is used for retransmission to more easily enable use of subtree multicast when this option is eventually supported by routers.

2.10.3 HACK Packets

A HACK packet is unicast from a child node to its parent node to indicate the status of the Data packets which have arrived and to furnish statistics about the state of data reception at the node.

A HACK requests retransmission of Data packets that have not been received. A HACK also acknowledges packets that have become stable. A packet becomes stable when it has been received by sufficient, critical nodes so that the sender is no longer required to hold the packet for retransmission.

2.10.4 NACK

A NACK is a HACK that is used to request immediate recovery of lost Data packets.

2.10.5 JoinStream Packet

All nodes except the top node send a JoinStream request to join a data stream. The first JoinStream packet implicitly joins the RMTP-II tree.

2.10.6 JoinConfirm Packet

A parent sends a JoinConfirm after it processes a JoinStream packet. The JoinConfirm indicates the success or failure of the JoinStream request.

2.10.7 JoinAck Packet

If a control node is unable to respond immediately to a JoinStream request by a child with a JoinConfirm packet, it unicasts a JoinAck packet to that child.

2.10.8 LeaveStream Packet

All nodes, except the top node, send a LeaveStream request when the node leaves the data stream.

2.10.9 Heartbeat Packet

A control node periodically sends Heartbeat packets to notify its child nodes that it is alive.

If a child node does not receive a Heartbeat packet within a specified time limit, it detects that the parent has failed and joins another parent node.

2.10.10 HeartbeatResponse Packet

All nodes, except the top node, unicast a HeartbeatResponse packet to its parent node in response to a Heartbeat packet. The HeartbeatResponse packet indicates that the child node is still alive.

If a parent does not receive a HeartbeatResponse packet or a HACK packet for a specified interval of time, then it detects that the child node has failed and removes that child from its list of child nodes.

2.10.11 NullData Packet

If a sender node has no data to send for a stream, it periodically multicasts a NullData packet on the data channel. NullData packets inform receivers about the state of the data stream and the sender.

A NullData packet contains the sequence number of the last packet sent, which allows the receivers to determine missing packets. If a receiver does not receive any Data or NullData packets from the sender for a specified time, then the receiver will declare the sender failed.

2.10.12 Eject Packet

If a child node is not operating normally, or a parent node restarts after a failure and receives a packet from a child not in its child

list, then the parent node unicasts an Eject packet to the child node. This child node can be another control node, a sender, or a receiver.

2.10.13 LeaveConfirm Packet

A parent node unicasts a LeaveConfirm packet in response to a LeaveStream packet from a child node.

3 Node Functions

3.1 Sender Node

Stream Variables

A sender node maintains the values of variables relating to the data stream: the size of the data queue, the packet admission rate, the sequence number of the lowest numbered unstable packet, and the sequence number of the highest numbered packet in the data queue.

Join Stream

A sender node sends a JoinStream request to the top node for each stream it intends to send. The sender provides a data channel with the JoinStream request and either provides a unique StreamID or else requests that the top node generate a unique StreamID.

Data Transmission

A sender multicasts Data packets to the receivers on the data channel at a sending rate determined by the flow and congestion control algorithms. If there is no data to send for a stream, the sender periodically sends NullData packets to indicate that the data stream is active.

Data Queue

When a sender receives a HACK from the top node, it uses the information to delete packets from the data stream's data queue. A packet is deleted from the queue if that packet is stable and all lower numbered packets are also stable.

Data Retransmission

A HACK or NACK indicates which packets have not been received. A sender retransmits the missed packets on the data channel. The time interval between successive retransmissions for a data packet is doubled for each retransmission, with an upper limit of 64 seconds(exponential backoff).

Data retransmission has higher priority than new data transmission. The sending rate is controlled by the flow and congestion control algorithms.

Tree Integrity

A sender gets Heartbeat packets from the top node's multicast control channel and sends HeartbeatResponse packets to the top node on the unicast control channel. This allows the sender and the top node to determine whether the other is functioning. The NullData packets also inform the receivers that the sender is functioning.

Receivers send HACKs in response to NullData packets, which informs the sender of the status of the receivers.

3.2 Receiver Node

Join Stream

A receiver node sends a JoinStream request to its parent node for each of the data streams that it wishes to receive.

Data Reception

A receiver node receives the data packets on the data channels and delivers the data to its application.

Data Reliability

A receiver delivers the Data packets to the receiver application in accordance with quality of service specified for the data stream.

A receiver sends HACKs to its parent to indicate the status of the Data packets received and missed.

A receiver may also send NACKs to expedite the recovery of missing Data packets.

Tree Integrity

A receiver receives Heartbeats from its parent on the parent's multicast control channel to insure that the parent is operating.

A receiver sends HACKs and NACKs to its parent for each of its active streams to indicate its availability. If none of the streams that a receiver has joined are active, it periodically sends a HeartbeatResponse packet to its parent.

3.3 Top Node

Tree Creation

The top node is assigned administratively and is the core of the tree.

Sender Control

The top node controls transmission parameters and congestion control parameters to each sender, and can change these dynamically while the sender is transmitting.

Data Reliability

The top node aggregates HACKs received from its children and forwards the aggregated HACK to the sender node.

Group Membership

The top node aggregates the membership count of the receivers, and passes this to each sender.

Data Transmission

The top node may optionally accept unicast data from senders and provide multicast transmission to the group. This is needed if the sender node cannot multicast data.

Stream Identification

The top node allocates new StreamID values and guarantees the uniqueness of the StreamIDs. If any sender requires a system generated StreamID, the top node provides a unique value. If a sender provides a StreamID value, the top node verifies that the value is unique. If the value is not unique, the sender is not allowed to join the RMTP-II tree.

Tree Integrity

The top node sends Heartbeats on its multicast control channel to notify its child nodes and senders that it is available.

The child nodes of the top node respond to its Heartbeats with HACKs or Heartbeat response packets.

The sender nodes respond to the top node Heartbeats with HeartbeatResponse packets.

3.4 Aggregator Node

Data Reliability

An aggregator node aggregates HACKs from its child nodes and sends them to its parent node on the unicast control channel.

NACK Forwarding

An aggregator node forwards NACKs from its child nodes to its parent node.

Retransmission Forwarding

An aggregator node forwards retransmissions from its parent to its children.

Group Membership

The aggregator node accumulates the membership count of its receivers, and passes this count to its parent. An aggregator node also maintains a list of its receivers.

Tree Integrity

An aggregator sends Heartbeats on its multicast control channel to notify its child nodes of its availability.

An aggregator receives Heartbeats from its parent to insure that its parent is available.

An aggregator receives HACKs or HeartbeatResponse packets from its child nodes to insure that the child modes are available.

3.5 Designated Receiver Node

Aggregation

A designated receiver node has all the functionality of the aggregator node.

Data Retransmission

A designated receiver receives data from all the data streams that are joined by its child receiver nodes. A designated receiver buffers the data for potential retransmission to its child receiver nodes.

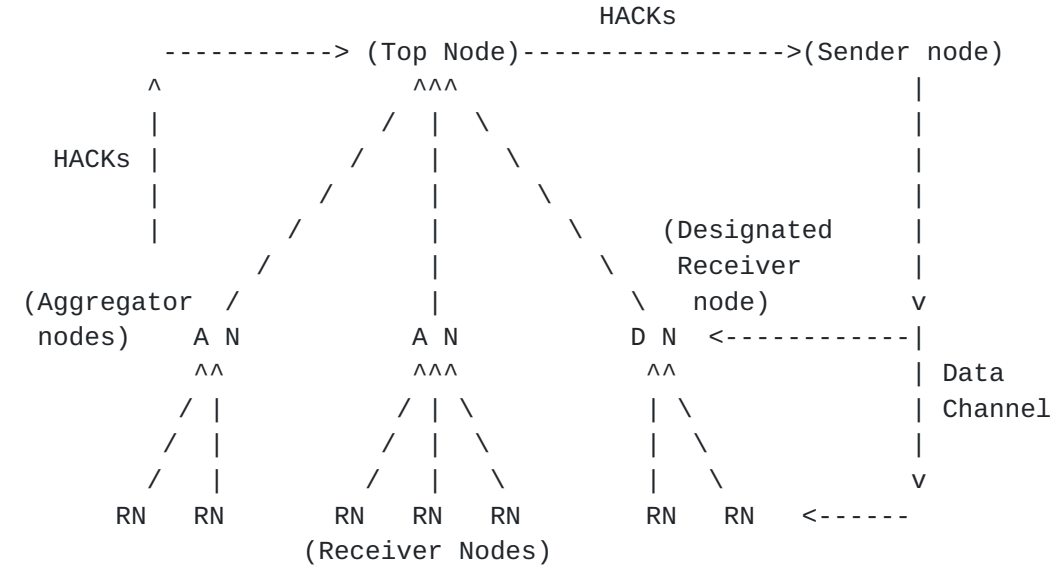
4 Operation of the RMTP-II protocol

4.1 Basic Operation of the Protocol

RMTP-II provides sequenced, reliable delivery of data from a few senders to a large group of receivers. RMTP-II consists of a network that has one or more sender nodes, many receiver nodes and one or more control nodes.

The simplest configuration consists of a single sender and a top node and receivers on multiple hosts connected to the network. An implementation could allow multiple nodes of different types to run in a single process, allowing a host to act as a sender and a receiver, a sender and a top node, a receiver and a designated receiver, or other combinations.

The figure below illustrates an RMTP tree with multiple control nodes.



A sender joins the RMTP tree and multicasts Data packets on the data channel.

In the case of a symmetrical network whose control tree topology

congruent to the multicast routing tree topology, the data channel may have the same path as the control channel. In this case, a data packet would be multicast from the sender through the control nodes to the receivers.

In the case of an asymmetrical network such as a one way satellite network with a terrestrial return path, the receivers and designated receivers would receive the multicast data directly, but the aggregators and top nodes will only receive control traffic.

A receiver joins a data channel to receive data. A receiver periodically informs its parent about the packets that it has or has not received by unicasting a HACK packet to the parent. Each parent node aggregates the HACKs from its child nodes and unicasts a single aggregated HACK to its parent. The top node aggregates the HACKs from its child nodes and unicasts a single HACK to the sender.

Each control node multicasts Heartbeat packets that inform their child nodes that the parent node is still functioning.

A tree forms a loop from the sender to the receivers, and back to the sender. Data and NullData regularly exercise the downward data direction. Heartbeat packets exercise the downward control direction. HACKs, NACKs, and HeartbeatResponse packets regularly exercise it in the upward direction. This combination constantly checks that all of the nodes in the tree are still functioning correctly, and initiates fault recovery when required.

4.2 Global Tree Parameters

A collection of tree-wide parameters are set and controlled at the top node. All the nodes of the RMTP tree acquire these parameters when they join the RMTP tree for the first time and receive a JoinConfirm packet.

These global parameters can be changed at the top node. The changes are propagated to the nodes of the tree in optional fields of the Heartbeat packets.

The top node sends the modified global parameters in Heartbeat packets and waits for all of its child nodes to confirm the change.

A child node receives and applies the changed parameters and sends a confirmation to its parent in a special HeartbeatResponse packet.

When a control node receives a Heartbeat packet with modified global parameter options, it propagates the modified global parameters to its child nodes in its Heartbeat packets. A control node sends the

confirmation HeartbeatResponse packet to its parent only after it receives HeartbeatResponse confirmations from all its child nodes.

When the top node gets confirmations from all its child nodes, it is guaranteed that all the nodes in the tree have updated their global parameters.

4.3 Data and Control Paths

Data is multicast by a sender on the data channel. Data may be retransmitted by the sender on the data channel or be multicast or unicast on a local control channel by a designated receiver.

The control channels are organized into a tree with the top node at the top of the tree and the receivers at the bottom of the tree. Aggregators and designated receivers are always in the middle of the tree.

4.4 Data Transmission

In order to receive Data packets from a sender, a receiver must join the multicast group for the stream to be received. Each multicast group is called a data channel. A receiver only joins the data channels for the streams it will receive.

A designated receiver joins all of the multicast groups that its descendants have joined.

A sender transmits Data packets to a group using IP Multicast. The intermediate nodes in the RMTP-II tree structure are not responsible for forwarding these Data packets. The standard IP Multicast routers forward the packets to all receivers and designated receivers.

4.5 Retransmission

There are three types of retransmissions: local retransmission, global retransmission, and subtree retransmission.

4.5.1 Local Retransmission

If a designated receiver determines from its child node's HACKs or NACKs that a Data packet was missed, the designated receiver retransmits the Data packet. The designated receiver multicasts the retransmission on its multicast control channel. Each control node child forwards a retransmission, as required, to its children.

4.5.2 Global Retransmission

If a sender node receives a HACK or NACK that indicates missing packets, the sender performs global retransmission. The unacknowledged packets are multicast as Retransmission packets on the data channel.

4.5.3 Subtree Retransmission

On a network whose routers support subtree multicast transmission, a designated receiver at the top of a subtree may multicast retransmitted data to its subtree. In this case, it is not necessary for the control nodes below the designated receiver in the subtree to forward the retransmissions. Currently, no commercial routers support this feature.

4.6 Options

4.6.1 Expedited Recovery

Expedited Recovery is an optional feature, which can be enabled on a per-stream basis. If the NACK option is enabled for a Data stream, an explicit request, called a negative acknowledgment, or NACK, may be sent by a child node to its parent requesting retransmission of missed data packets. Expedited recovery is used for data streams that require very low latency for data recovery or for networks that have low loss rates and wish to reduce the amount of control traffic sent.

4.6.2 NACK Suppression

A receiver waits a random interval, within a specified range, before sending a NACK. If the receiver receives the retransmitted data before the NACK timer expires, the receiver cancels the NACK. This reduces the chance that multiple receivers generate a NACK for the same packet.

A designated receiver node multicasts a Data packet to its children as soon as it gets a NACK request for that packet.

An aggregator node or a top node forwards only one NACK for a missing Data packet within a specified period of time. Other NACKs for a missing packet are ignored because the up-stream designated receiver or Sender will multicast the retransmission.

4.6.3 Control Tree discovery

RMTP supports an option which allows the nodes in the RMTP tree to acquire the addresses and location of its ancestors in the tree and the addresses of its parent's siblings

If an RMTP-II node's parent fails, then the node can use the acquired information to join an alternate control node.

4.6.4 Forward Error Correction

Option fields in RMTP-II can be used to implement Forward Error Correction. RMTP-II supports both proactive FEC, in which a fixed amount of parity is sent with the data, and reactive FEC, in which parity is sent in retransmission packets to repair multiple independent losses. See the Appendix - Forward Error Correction for details.

4.6.5 Fault Tolerant Top Node

RMTP-II provides support for the implementation of a fault tolerant top node. The top node uses the options field of its Heartbeat packets to send the address of a backup top node to all of its children. The top node sets this option when a backup top node joins or leaves the RMTP tree.

4.6.6 Congestion Control

Congestion Control option fields are used to propagate the loss rate and other congestion control parameters. RMTP-II proposes one congestion control mechanism. See the Appendix - Congestion Control for details.

4.6.7 Time Bounded Reliability

A bound for delivery time may be optionally specified for a stream. Packets are delivered exactly once and in the same order in which they were sent from the source, but if the time bound on a packet expires it is dropped. See the Appendix - Time Bounded Reliability for details.

4.7 Tree Connections

In order to join a tree, each node, except top node, sends a JoinStream packet to its parent node and sets a Join timer for the confirmation.

If the parent node can immediately confirm the JoinStream request, it responds with a JoinConfirm packet. If the parent node needs more time to process the JoinStream request, it responds with a JoinAck packet. After it fully processes the JoinStream, it sends a JoinConfirm packet.

The top node does additional processing for a JoinStream request from a sender node. The top node validates the uniqueness of the StreamID of the JoinStream request. If a sender requests a system generated StreamID, the top node generates a unique StreamID.

4.8 Flow and Congestion Control

Flow and congestion control algorithms act to prevent the senders from overloading the receivers.

RMTP-II uses a send-ahead mechanism to allow continuous transmission of data without waiting for packet acknowledgments. This is used in conjunction with flow and rate control algorithms.

Research in congestion control for large scale reliable multicast is not yet mature. RMTP-II is currently designed to support a range of window based and rate based flow and congestion control mechanisms. Each of these mechanisms can be specified as an option and is controlled by the top node of the tree. For general Internet use, specific congestion control policies will eventually be mandated.

4.8.1 Fixed Rate Based Control

Fixed, rate-based flow control limits the transmission speed to a predefined value. The flow control rate includes the global multicast retransmissions from the sender.

4.8.2 Congestion Control

RMTP-II uses a Loss Tolerant Rate Controller algorithm (LTRC) for congestion control. See Appendix B. This controller uses loss report information from the receivers and perceived state at the sender to perform congestion control. The goal of the controller is to be responsive to congestion, but not overly reactive to spurious independent loss.

LTRC allows the sender application to dynamically adjust the admit rate between specified limits. A sender gets loss rate information for the receivers from the HACK packets. The bottom level control nodes calculate loss rate for each of its receiver child nodes from the HACK packet information. The control nodes send the maximum loss rate for the child nodes to its parent in a HACK packet. The next level control nodes do not aggregate the loss rate, but send the maximum loss rate for all the child nodes.

4.8.3 Ejection of Slow Receivers

Each sender node is responsible for ejecting slow receivers.

The sender application provides the minimum and maximum admit rate limits. If the admit rate falls below the minimum rate, the sender maintains the admit rate at the minimum rate and unicasts an Eject packet to the top node. The Eject packet contains the loss rate threshold for the stream and a flag that indicates whether all receivers that exceed the threshold should be ejected or only the receiver(s) with the maximum loss rate exceeding the threshold. The top node sends one or more Eject packets to its children based on the loss rate threshold and bit flag in the Eject packet. If the flag is set then it will send Eject packet to all of its children having loss more than the threshold else only to the child having maximum lass over threshold. If the child node is a control node it uses the same mechanism to send Eject packet to its children. The receiver(s) having loss rates at or above the threshold get Eject packets and are required to leave the stream. The receivers may rejoin the stream at a later time.

4.9 Membership

RMTP provides a simple counted membership for each stream, available at the top node and sender node. This counted membership is a simple count of the receiver nodes that have joined a stream.

The detailed RMTP tree and stream membership information is distributed across all the control nodes in the RMTP tree. An external membership server can be used to acquire the full tree and per stream membership from all the control nodes.

4.10 Fault Detection and Recovery

4.10.1 Sender node failure

A sender node that has no data to send will periodically send NullData packets on the data channel. If a receiver fails to receive Data packets or NullData packets for a stream sent by the sender, the receiver detects a sender failure.

A sender node sends HeartbeatResponse packets to the top node. If the top node fails to receive a HeartbeatResponse from a sender, it detects the sender's failure.

4.10.2 Receiver node failure

A receiver node sends HACKs for each of the active streams that it has joined. If none of the streams are active, then the receiver sends HeartbeatResponse packets to its parent.

If a receiver's parent node does not receive a HACK or a

HeartbeatResponse within a specified time interval, the parent detects the failure of the receiver and removes the child from its child list.

4.10.3 Top node failure

The top node sends Heartbeat packets to its child nodes on its multicast control channel. If a child node does not receive Heartbeats from the top node, it detects a failure of the top node.

4.10.3.1 Fault Tolerant Top Node

RMTP-II provides for the implementation of a fault tolerant top node. The top node sends the address of a backup top node to all of its children in the Heartbeat packets. A backup top node monitors the Heartbeat packets of the top node. If the top node fails, the backup top node takes over.

If a child node of the top node detects a failure of the top node, it reconnects to the backup top node.

4.10.4 Aggregator/Designated Receiver failure

Each aggregator and designated receiver node sends Heartbeat packets to its child nodes on its multicast control channel. If the child nodes do not receive any Heartbeats from the parent node, they detect failure of the parent.

4.10.4.1 Recovery

When a child node detects failure of its parent node, it can try to reconnect to a randomly chosen alternate aggregator, designated receiver, or the top node of the RMTP-II tree.

4.11 Quality of Service (QoS)

RMTP-II supports three levels of QoS:

Reliable Unordered: packets are delivered exactly once, but packet order is not guaranteed.

Reliable Source Ordered: Packets are delivered exactly once and in the same order in which they were sent from the source.

Time Bounded Reliability: This is an optional QoS in which packets are delivered exactly once and in the same order in which they were sent from the source, but packets that cannot be delivered within a specified time bound are dropped. See Appendix D - Time

Bounded Reliability.

4.12 SNMP Support

The control nodes, and the top node in particular, are designed to interact with SNMP management tools. In cooperation with the top node's centralized control of the senders, this allows network managers to easily monitor and control the sessions being transmitted.

All nodes of RMTP-II provide SNMP support. SNMP support is optional for sender and receiver nodes, but is required for all control nodes. See the Appendix - SNMP MIBs for details.

4.13 Session Manager Support

The Session Manager is an optional component associated with RMTP-II which is responsible for providing each node with topology configuration information. A session manager can be implemented either as a set of static configuration files, as a fully distributed algorithm, or as a centralized management tool that interfaces with each of the nodes.

A session manager can optionally be responsible for providing total group membership listings by aggregating the membership lists at each control node.

A session manager can optionally provide an interface with directory services for the announcement of sessions.

A session manager be responsible for partitioning the data at the sender, selecting the appropriate sets of layers at each receiver, and for reassembling the data at the receiver. This work is currently in progress.

4.14 Forward Error Correction

Recent work [NB96, NBT97, Rizzo97] has shown the benefits of incorporating reactive forward error correction (FEC) into reliable multicast protocols. This feature encodes data packets with FEC algorithms, but does not transmit the parity packets until a loss is detected. The parity packets are then transmitted and are able to repair different lost packets at different receivers. This is a powerful tool for providing scalability in the face of independent loss. When implemented, it is a simple matter to also provide proactive FEC which automatically transmits a certain percentage of parity packets along with the data. This is particularly useful when

a high minimum error rate is expected, or when low latency is particularly important. This can be implemented as an option to RMTP- II. See Appendix D for FEC details.

5 Details:

5.1 RMTP-II Global Parameters

5.1.1 The Tree-Wide Parameters

The following global, or tree-wide, parameters are defined for an RMTP-II tree. These parameters are configured at the top node. The other nodes acquire the parameters when they connect to the RMTP-II tree.

The parameters can be modified at the top node. The changes will be propagated to the rest of the tree.

B: The branching factor B denotes the maximum number of children for any node in the tree.

C: The constant C is a scaling coefficient for Thack. The constant C determines how quickly Thack increases when no data packets are being sent.

R: The number R specifies the maximum number of HACKs which should be received, on average, for each Data packet sent, at any node. The stored value of R is divided by 100 to get an R value of the form dd.dd.

Thack_max: The number Thack_max specifies the maximum time allowed between HACK transmissions for each receiver.

Tjoin_response: The number Tjoin_response is the maximum time to wait for a response to a JoinStream request from the parent node. The response should be either a JoinAck or a JoinConfirm.

Rjoin: The number Rjoin is the number of retries of the JoinStream request before declaring the parent unreachable.

Thb: the number Thb is the time interval at which control nodes multicast Heartbeat packets.

F: The failure threshold constant, F, determines the threshold time for failure detection.

Tnulldata_max: The number Tnulldata_max is the maximum time interval for sending NullData packets.

Optimistic: All the designated receivers should use optimistic HACK mechanism.

RxMax: The number RxMax is the maximum number of retransmission that can be done for a data packet.

Radmit_rate_min: This is the minimum admission rate that the sender can use for Data packets.

Radmit_rate_max: This is the maximum admission rate that the sender can use for Data packets.

5.1.2 Global Tree Parameter Distribution

When a node joins an RMTP tree for the first time, it receives the global parameters from the options extension of the JoinConfirm packet.

A node receives changes to global parameters in Heartbeat packets whose option flag, Global Parameters Modified, is set to 1.

If a control node receives a Heartbeat packet containing modified variables, it sends Heartbeat packets to its children with the option flag, global parameters modified, set to 1. The control node continues sending the Heartbeat packet until it receives confirming HeartbeatResponse packets from all of its children. The control node then applies the new values of global parameters and sends the confirming HeartbeatResponse packet to its parent.

When a receiver node receives a Heartbeat with the option flag, global parameter modified, set to 1, it applies the new parameter values and sends confirmation in the next HeartbeatResponse packet to its parent.

Each time modifications are made to global parameters at the top node, it increments a 16-bit sequence number and inserts this sequence number in the Heartbeat packets it sends to its child nodes. The nodes of the RMTP tree identify global parameter modifications based on the sequence number in the Heartbeat packet.

The top node does not allow additional modifications to the global parameters, until it gets confirmations from all of its children that the current modifications have been received.

5.2 Tree Configuration

A node requires information about the RMTP-II tree and the data stream it wishes to join. The mechanism for acquiring the configuration information is not part of the protocol, but is implemented by the associated Session Manager. The RMTP-II protocol specifies only the rules by which a node joins the tree.

A node requires the following information to join an RMTP-II tree. The information depends on the type of the node.

5.2.1 Top Node Configuration

A top node requires:

tree ID: a unique identifier for the RMTP-II tree

UDP listen port: the number of the port on which the top node will listen for its children's control messages

Local Multicast Control Channel: the address on which the top node sends Heartbeat packets to its children.

5.2.2 Backup Top Node Configuration

A backup top node requires:

tree ID: a unique identifier for the RMTP-II tree

UDP listen port: the number of the port on which the top node will listen for its children's control messages

Local Multicast Control Channel: the address on which the top node sends Heartbeat packets to its children.

5.2.3 Aggregator node Configuration

An aggregator requires:

tree ID: the unique identifier for the RMTP-II tree to join

parent address: the address and port of the parent node to which the node should connect

UDP listen port: the number of the port on which the node will listen for its children's control messages

Local Multicast Control Channel: the address on which this node sends Heartbeat packets to its children.

5.2.4 Designated Receiver node Configuration

A designated receiver requires:

tree ID: the unique identifier for the RMTP-II tree to join

parent address: the address and port of the parent node to which the node should connect

UDP listen port: the number of the port on which the node will listen for its children's control messages

Local Multicast Control Channel: the address on which this node sends Heartbeat packets to its children.

5.2.5 Sender or Receiver node

A sender or receiver requires:

tree ID: the unique identifier for the RMTP-II tree to join

parent address: the address and port of the parent node to which the node should connect.

StreamID: the unique identifier of the stream the node wants to send or receive

data multicast group address: the multicast address on which the data stream is sent.

5.3 Tree Membership Algorithms

5.3.1 Join Algorithm

A sender node makes a JoinStream request for every data stream that it intends to send. The parent for a sender is the top node.

When a receiver node is created, it sends a separate JoinStream packet for each of the streams that it intends to receive. When a receiver node rejoins a stream after a parent failure, it may send a single JoinStream packet to join multiple streams.

Aggregator and designated receiver nodes send a JoinStream packet with StreamID value zero to join the RMTP tree, without joining any stream. An aggregator or designated receiver later joins the streams that their children join.

A node sends a JoinStream request to its parent node. It waits a

time interval, Tjoin_response, for a response from the parent node. A parent node responds to a JoinStream request with a JoinAck packet or a JoinConfirm packet. If the parent node cannot process the request immediately, it responds with a JoinAck packet. Otherwise, it sends a JoinConfirm packet.

If no response is received, the node retransmits the JoinStream request. The node retransmits the JoinStream request a maximum number of times, Rjoin. If a node does not receive a response for the JoinStream requests after Rjoin tries, it reports a Parent-Unreachable failure after Rjoin retries.

If a node receives a JoinAck packet, it continues transmitting JoinStream requests at exponentially longer times, until it receives a JoinConfirm rather than a JoinAck. Every time that a node does not receive a response from its JoinStream request, it increments a local variable NumberFailures. Every time that it does not receive a JoinAck in response to its JoinStream request, it increments NumberFailures. If NumberFailures exceeds Rjoin, it reports a Parent- Unreachable failure.

A node receives the tree's constant parameters and the parent's multicast control channel address from the JoinConfirm.

When joining, the sender can request that RMTP-II provide the StreamID, in which case the top node generates a unique StreamID and sends it in the JoinConfirm packet.

If a sender fails and than attempts to rejoin the tree, the top node will not allow this new sender to join until a period 6*F*Thb, the sender failure detect interval, has elapsed since the failure.

A control node can proactively join the RMTP-II tree without joining a stream by issuing a JoinStream request with a StreamID value of zero. A control node can reactively join the RMTP-II tree when it receives a JoinStream request from any of its children. The control node joins all the streams that its children join.

When a node issues a JoinStream request during a failure recovery, it sets the rejoin flag in the JoinStream request. During a rejoin, a node can specify all the streams that it wants to join in a single JoinStream request.

5.3.2 Sender TimeStamp

When a sender is created, it records the current time, expressed in elapsed seconds and microseconds since 00:00 Universal Coordinated Time, January 1, 1970. The sender uses the seconds part to

initialize the value of TimeStamp. TimeStamp is a 32 bit number which is unique within approximately 136 years.

The resolution of the TimeStamp is 1 second. Because this is UTC, there is no counter discontinuity when daylight savings occurs.

The TimeStamp value is sent in every Data, NullData, Retransmission, and HACK packet for the data stream, and does not change over time for a stream.

The value of TimeStamp allows nodes to distinguish different incarnations of the sender. A new TimeStamp on a data stream indicates that the sender has restarted.

5.3.3 Leave Algorithm

A node sends a LeaveStream request when it wishes to leave the data stream. The node waits for a period Tjoin_response to receive the LeaveConfirm packet from its parent node. If the LeaveConfirm packet is not received within the time Tjoin_response, then the node resends the LeaveStream request. The node sends the LeaveStream request a maximum of Rjoin times.

A control node can leave a stream only if it has no children.

To leave a stream, a sender node sends an EOS packet and waits for the EOS in the HACK before leaving the stream.

5.4 Data Transmission

An application provides Data packets to a sender node. The sender node assigns consecutive sequence numbers to the Data packets and buffers the Data packets in the data queue.

A sender can use any sequence number other than 0 as the starting sequence number for the stream. Sequence number 0 indicates an inactive data stream and should not be used for other purposes.

The sender initializes LastStable as StartSequenceNumber-1. LastStable is the sequence number of the last stable packet which the sender has removed from its data queue. All lower numbered packets are stable. The sender cannot retransmit any packet having sequence number less than or equal to LastStable.

LastStable in conjunction with the sequence number is used by receivers to detect the first Data packet for the stream as well as missing packets.

For example, suppose that a sender's StartSequenceNumber = 10 and the receiver's first Data packet has sequence number 15. LastStable = 9 (10-1). The missing packets are 10, 11, 12, 13 and 14.

5.4.1 Stable Packets

If the lowest numbered Data packet in the data queue becomes stable, that is, all the target recipients have received the packet, then the sender deletes the packet from the data queue. The Sender receives stability information from the Stable Sequence Number of the HACKs it receives.

5.4.2 Stream Parameters

The sender maintains the following values for each of its data streams:

Ndata_size: This is the data queue size, the number of Data packets the sender can buffer for transmission or retransmission. A sender can admit a packet to the data queue only if the number of packets in the data queue is less than Ndata_size.

Radmit_rate_max: A sender uses Radmit_rate_max to control the maximum sending rate. The sending rate is determined by congestion control algorithms, and can fluctuate between Radmit_rate_min and Radmit_rate_max.

Nseq_low: This is the sequence number of the lowest numbered packet in the data queue which is not yet stable. It is the sequence number of the oldest Data packet that can be retransmitted.

Nseq_high: This is the sequence number of the highest numbered packet in the data queue.

Tsend_alarm: This is the interval between multicasts of Data packets by the sender.

Tnulldata_min: This is the minimum interval between the transmission of NullData packets by the sender.

Tnulldata_alarm: This is the current interval between transmissions of NullData packets by the sender.

5.4.3 Example: Packet Stability and Transmission

Suppose that: Ndata_size equals 150, Nseq_low equals 200, Nseq_high equals 300, and all the Data packets numbered from 200 to 230 have gone stable. In this case, the sender will delete the stable packets

from 200 to 230 from the data queue. The new value of Nseq_low will be 231. The sender can transmit Data packets until Nseq_high reaches 380 (Nseq_low + Ndata_size).

5.4.4 Example: Packet Rate

Suppose the sender is using fixed size packets and Radmit_rate = 512 Kbps = 65536 Bps, Ndata_size = 200 packets, PACKET_SIZE = 1024 bytes, Tsend_alarm = 50 ms = 20 ticks per second.

The maximum number of packets that can be sent per interval Tsend_alarm is: (Radmit_rate*1024)/(8*(1000 / Tsend_alarm)*PACKET_SIZE) For this example the sender can at most send (512 * 1024)/(8 * (1000/50) * 1024) = 512/160 which is approximately 3 packets per interval

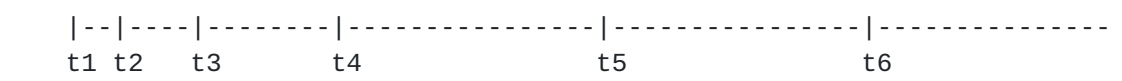
5.4.5 NullData Packets

If a sender has no Data packets to transmit for a data stream, it transmits NullData packets on the data channel. A NullData packet contains a sender time stamp, TimeStamp, the sequence number of the last packet, LastSequence, and the sequence number of the last stable packet, LastStable.

NullData messages keep the stream alive, inform the receivers about the state of the sender, and detect packets missed by receivers.

When there are no Data packet to send, a sender sets Tnulldata_alarm to Tnulldata_min. If Tnulldata_alarm expires, then the sender transmits a NullData packet and sets Tnulldata_alarm to twice its current value. The maximum value of Tnulldata_alarm is Tnulldata_max. This is done to utilize minimum bandwidth and insure timely recovery of missing Data packets.

Example: Increase of Tnulldata_alarm
Suppose that:
Tnulldata_min = 2000 ms
Tnulldata_max = 16000 ms
Time of transmission of last Data packet = t1
Let s = Tnulldata_min
If there is no data to send, the sender transmits NullData packets at times t2 = t1+s, t3 = t2+2s, t4 = t3+4s, t5 = t4+8s, t6 = t5+16s



5.4.6 End Of Stream

The end of a data stream is indicated by the flag EOS in the final Data packet of the data stream. The sender transmits a Data packet with the EOS bit set to 1 and waits for the packet to become stable.

If a receiver receives a Data packet with EOS set to 1 and it is not missing any Data packets, it continuously generates HACK packets, based on Thack, with the EOS bit set to 1, until it gets an EOS packet from its parent confirming the end of stream. The receiver can then leave the stream.

If a parent node receives a HACK with the EOS bit set to 1, it responds with an EOS packet to that child node confirming the end of stream.

If a parent node has received HACKs for a data stream with the EOS bit set to 1 from all of its child nodes, then it sends a HACK with EOS set to 1 to its parent node. Once a sender indicates the end of the stream, it must wait until it gets a confirming EOS packet from the top node before it can leave the group.

5.4.7 Computation of Feedback Round Trip Time

The round trip time, RTT, is the difference between the time a Data packet is sent by the sender and the time that a HACK is received by the sender indicating that the packet was received or requesting retransmission of the packet.

The sender measures the RTT for the first packet of a stream and every packet with sequence number equivalent to 1 mod H where

```
H = Ceiling(B/R)
B = MaxChildren
R is the overhead ratio.
```

For example, if H = 200, the sender will calculate RTT on packet number 1, then on packet 201, 401, 601. The sender will receive, on average, one HACK per 200 Data packets.

5.4.8 Retransmission Timeout

The Retransmission Timeout, RTO, is the time interval between retransmissions, in milliseconds.

The RTO value is calculated by the sender node using the algorithm of [Jacobson88], [Jacobson90] which depends on both the smoothed RTT and the smoothed mean deviation. The algorithm can be implemented with integer arithmetic.

M = measured RTT at the sender
A = smoothed RTT
D = smoothed mean deviation

Initially, let

g = 1/8, g is the gain for the RTT average.
h = 1/4, h is the gain for deviation
A = 0, current RTT estimator
D = 3 seconds

Err <= M - A
A <= A + g*Err
D <= D + h*(|Err| - D)
RTO = A + 4D

5.5 Data Reception

5.5.1 New Stream

When a receiver joins a data stream it receives a JoinConfirm packet which contains the StreamID, the TimeStamp, and the value of LastStable.

If the data stream is not active, then LastStable equals 0. If the data stream is active, LastStable is the number of the stable packet with the greatest sequence number. The receiver can only receive packets with higher numbers than LastStable. The receiver calculates the starting sequence number, StartSequenceNumber, for the stream by adding 1 to the value LastStable. If the result equals 0 because of wrap-around, then the value is incremented to 1, since sequence number 0 is reserved.

5.5.2 Quality of Service

A receiver node delivers Data packets based on the quality of service value, QoS, specified for the data stream. If the QoS value is Source Ordered, then the Data packets must be delivered to the application in the order defined by the sequence numbers of the packets. If the QoS is Unordered Reliable, the packets are delivered to the application in the order they are received, but are never delivered more than once.

An optional QoS level, Time Bounded Reliability, is detailed in Appendix - Time Bounded Reliability.

5.5.3 Detection of Missing Packets

Data, NullData, and Retransmission packets contain information that a receiver uses to detect missing packets.

A NullData packet for a stream contains the sequence number of the last Data packet transmitted for the stream. A receiver determines whether any intervening packets in the sequence are missing.

Each Data packet and Retransmission packet contains its own sequence number and the number of the last stable packet, LastStable. A receiver determines whether any packets have been missed between the last stable packet and the highest numbered packet received.

5.5.4 Recovery of Missing Packets

Both HACKs and NACKs are used for recovery of missing packets.

If a receiver detects a missing Data packet, it sets to zero the bit of the HACK bitmap which corresponds to the sequence number of the missing Data packet.

If the bit flag N of Data packets, NullData Packets, or Retransmission packets for a data stream is set to 1, then NACKs are enabled for the data stream. If a receiver detects a missing Data packet and NACKs are enabled for the stream, then the receiver sends a NACK based on the NACK generation algorithm.

The HACK and NACK generation algorithms are described later.

5.5.5 Stream failure detection

A receiver detects the failure of a stream by the following mechanisms:

Timeout Detection If a receiver does not receive any Data, NullData, or Retransmission packets within an interval 2*F*Tnulldata_max, then it detects the failure of the data stream

Timestamp Detection If a receiver receives a Data packet whose TimeStamp value is greater than the current TimeStamp value for the data stream, then the receiver detects the failure of the stream.

The receiver changes the current TimeStamp value to the TimeStamp value of the received packet. The receiver discards any Data packet with a TimeStamp value that is less than the current TimeStamp value.

For example, suppose the current StreamID equals 0x1000 and TimeStamp equals 0x1234. The receiver receives a new Data packet with StreamID 0x1000 and TimeStamp 0x1300. The new TimeStamp value is higher than

the current value. The receiver detects that the stream has been abnormally terminated and restarted with the TimeStamp 0x1300.

If a control node receives a HACK packet from a child node for a data stream which has a higher TimeStamp value than the current TimeStamp value for the data stream, then the node detects that the data stream has failed. The control node updates its information for the data stream with the new TimeStamp value and it ignores all HACK packets with TimeStamp value less than the new TimeStamp value.

5.5.6 End of Stream

If a receiver receives a confirming EOS packet from its parent for a data stream, it discards any further Data packets for that data stream.

5.6 HACK Generation

HACK packets are used to acknowledge received Data packets and to detect missed packets. The purpose of a HACK is to acknowledge the packets that are considered stable, and to request retransmission of packets that have not been received.

The following fields of a HACK packet are used to determine the status of the received data:

High Sequence Number: If the node is a receiver, the High Sequence Number is the sequence number of the highest numbered packet that the node has received. If the node is a control node, then the High Sequence Number is the sequence number of the highest numbered packet received by any of its child nodes.

Low Sequence Number: If the node is a receiver, the Low Sequence Number is the sequence number of the lowest numbered packet that the node has not yet received. If the node is a control node, then the Low Sequence Number is the sequence number of the lowest numbered packet that has not yet been received by any of its child nodes.

Stable Sequence Number: The stable sequence number is the sequence number of the highest contiguous stable packet known to the node. If the node is a receiver, this number is the Low Sequence Number minus 1. If the node is an aggregator or top node then it is the aggregated Low Sequence number minus 1. If the node is a designated receiver then the Stable Sequence Number has different meaning depending on the HACK mechanism used.

Bitmap: The bitmap consists of a sequence of single bit

"acknowledged" flags, corresponding to the sequence of packets with sequence numbers between Low Sequence Number and the High Sequence Number. A bit value 0 indicates that the corresponding packet is missing and requires retransmission. For reasons of efficiency, the bit corresponding to Low Sequence Number is at the bit position Low Sequence Number Modulo 32. The remaining bits are assigned to consecutive sequence numbers up to High Sequence Number. Any remaining initial or terminal bits are ignored.

Example for details of HACK
LSN, the Lowest Sequence Number not yet received by this node = 40
HSN, the Highest Sequence Number received by this node = 72
Bitmap[0] = 0xFF7EDC7F
Bitmap[1] = FF800000
Here is the bitmap array written as 0s and 1s:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1|1 1 1 1|0 1 1 1|1 1 1 0|1 1 0 1|1 1 0 0|0 1 1 1|1 1 1 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1|1 1 1 1|1 0 0 0|0 0 0 0|0 0 0 0|0 0 0 0|0 0 0 0|0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The bitmap begins at position LSN modulo 32, which is position 8. Bits 0-7 in Bitmap[0] are ignored. The bitmap extends for HSN - LSN = 32 positions. The final position is at position 8 of Bitmap[1]. Bits 9 - 31 of bitmap[1] are ignored. The bitmap indicates that the following packets are missing: 40, 47, 50, 54, 55, 56.

The receiver nodes and control nodes use different algorithms for generating HACKs.

5.6.1 Receiver node HACK generation

Congestion caused by bursts of ACK or HACK traffic limits the scalability of ACK or HACK based protocols. RMTP resolves this problem by allowing the system to control the level of HACK overhead by dynamically adjusting the rate of control packets to match the rate of data packets, and by limiting control packet generation with the rotating HACK algorithm.

Control rate of HACK

Congestion due to control packet traffic is most likely to occur when data packets are sent at a high rate and the control channel has significantly lower bandwidth than the data channel. RMTP-II controls the rate by controlling the response ratio R = C/D. C is the

number of control packets a parent receives from its children in response to the number D of data packets received. This ratio is controlled by limiting the acknowledgments returned by the children.

The Rotating HACK Algorithm

Each receiver is informed of its index number, M, in its parent's child list from the JoinConfirm packet received when the receiver joins the data stream. The receiver calculates the value of H defined by: H = Ceiling(B/R). B and R are tree wide constants.

A receiver only sends a HACK for a packet whose sequence number Modulo H equals its M number modulo H. The value of H insures that the receiver child nodes of a single parent send a total of no more than R HACKs for each Data packet

If a packet is dropped in transit, and the packet would have generated a HACK for the rotating HACK algorithm, then the first packet received with a sequence number greater than the missing trigger packet will cause a HACK to be generated.

5.6.1.1 Example

Assume a parent has B = 6 children, and the desired ratio of HACK packets received to Data packets sent is R = 2. Then the children will respond to every H = 6/2 = 3 Data packets. The children are separated into three groups by their M numbers (Modulo H), 0, 1, 2. The children with number 0 will respond to sequence numbers that equal 0 (Modulo 3). The children with number 1 will respond to sequence numbers that equal 1 (Modulo 3). The children with number 2 will respond to sequence numbers that equal 2 (Modulo 3). The 6 children are partitioned into three response classes of size 6/3 = 2. There will be B/H = 2 children responding to each Data packet.

Sequence #:	101	102	103	104	105	106	107	108
Sequence #, Mod 3:	2	0	1	2	0	1	2	0

Receiver Index #:	1, 2, 3, 4, 5, 6
Receiver Index (Modulo 3):	1, 2, 0, 1, 2, 0

A small R value makes H large, and decreases the volume of control traffic. However, since the children do not send HACKs immediately, the time to reach a stable state is longer.

5.6.1.2 Guaranteed HACK Generation

When data traffic is low, a receiver may not send a packet for a long time. This could cause long waits for packet stability and could also

make the receiver appear to have failed. The tree wide constant Thack_max guarantees that receivers respond in a timely manner. Thack_max specifies the maximum time that can elapse between HACKs while the stream is active. If the stream is active, a receiver sends at least one HACK within the interval Thack_max.

The value of the variable Thack dynamically controls the time between HACKs. Thack varies as the average of the previous two inter-HACK times, Thack1 and Thack2. The values of Thack1 and Thack2 are initially set to Thack_max. Thack = Min((Thack1 + Thack2)*C, Thack_max) where C is a tree-wide constant that determines the responsiveness of Thack.

The Thack timer is initialized when a receiver receives the first packet of a data stream. A new Thack value is calculated and the Thack timer is reset each time the receiver sends a HACK packet.

If the data traffic load is high, H packets are received in time Thack or less and the rotating HACK algorithm determines the generation of HACKs.

If the load is low, the Thack timer expires before H packets are received and Thack determines the generation of HACKs. Under low load conditions, the value of Thack grows in an approximately exponential manner to Thack_max.

5.6.2 Control node HACK generation

A control node has two functions in the HACK algorithm, HACK aggregation and HACK generation. Designated receivers use a modified HACK mechanism.

Aggregator and top node HACK mechanism

An aggregated HACK from a control node acknowledges that a packet has gone stable only if all descendants of the control node have received the packet. The Lowest Sequence Number refers to the lowest numbered packet that has not been received by at least one of the control node's child nodes. The Highest sequence number is the highest numbered packet that has been received by all of the control node's child nodes. The Stable Sequence Number is the aggregated Lowest Sequence Number minus 1. The parent bitmap is a logical AND of corresponding bits of all the child bitmaps. The bitmap indicates the packets missed by all of the control node's descendents.

Example for HACK aggregation
HACK information from receiver-1
Lowest sequence number = 40

Highest sequence number = 72
Stable sequence number = 39
Bitmap[0] = 0xFF7EDC7F Bitmap[1] = 0xFF800000
missing packets: 40, 47, 50, 54, 55, 56

HACK information from receiver-2
Lowest sequence number = 38
Highest sequence number = 74
Stable sequence number = 37
Bitmap[0] = 0xFDFEDD7F Bitmap[1] = 0xFF600000
missing packets: 38, 47, 50, 54, 56, 72

Aggregated HACK
Lowest sequence number = 38
Highest sequence number = 71
Stable sequence number = 37
Bitmap[0] = 0xFD7EDC7F Bitmap[1] = 0xFF600000
The control node generates a HACK when it receives HACKs from all its child nodes. To guarantee HACK generation the control nodes sends a HACK if its Thack timer expires. The algorithm for Thack timer is identical to the receiver, refer to "Guaranteed HACK generation".

```
receiver-1:       40  47  50  54 55 56       (72)
receiver-2:   38      47  50  54    56        72   (74)
aggregated:   38  40  47  50  54 55 56   (71)
```

Packet 38 is missing from receiver-2 and is the lowest missing packet. If a packet is missing in either bitmap, it is missing in the aggregated HACK. Packet 71 is the highest packet received by both receivers.

Designated receiver HACK mechanism

The designated receiver has two kinds of HACK mechanisms: Pessimistic and Optimistic.

Pessimistic HACK

The pessimistic HACK mechanism is very similar to the aggregator's HACK mechanism. The Stable Sequence Number is the aggregated lowest Stable Sequence Number of all the child HACKs. The Lowest Sequence Number is the lowest missing packet above the Stable Sequence Number of the designated receiver. The Highest Sequence Number is the Highest Sequence Number received by the designated receiver. The bitmap only contains information about the data packets missed by the designated receiver.

Optimistic HACK

An optimistic HACK acknowledges that a packet is stable if the parent node receives it. It is the responsibility of the parent node to insure that every child receives the packet. A node generates an optimistic HACK based on its own list of received packets and uses the same rotating HACK algorithm used by the receiver nodes.

With optimistic HACKs, the designated receiver still needs to aggregate HACKs from its child nodes because it needs to discard buffered data packets. It discards a packet as soon as all of its children have received it.

The advantage of the Optimistic HACK mechanism is that the sender gets stability of packets before the packets are actually stable at the receivers. This decreases the buffering requirements at the receiver. The disadvantage of the Optimistic HACK mechanism is the resulting decrease in fault recovery.

Consider a RMTP tree in which a receiver has a designated receiver as parent and a designated receiver as the next ancestor node. The parent node receives all the packets, but the receiver misses a packet. If the receiver's parent dies before it can retransmit the missing packet and the receiver rejoins the ancestor node, the receiver cannot recover the mission packet. This is because the packet has already gone stable at the ancestor and at the sender because of the optimistic HACK mechanism.

5.7 Retransmissions in response to a HACK

HACK packets prompt retransmission of missing packets. This retransmission can be done by the sender or by designated receiver nodes.

A designated receiver node retransmits any missing packets for which it receives a request in a HACK and for which it is currently holding the requested packets. The retransmissions are multicast on the local control channel. When a designated receiver fulfills a retransmission request, it deletes that request from the request bit field in the HACK to prevent any of its parents from duplicating the retransmission.

When a designated receiver retransmits a packet, it calculates a minimum period of time, Tmin_retransmit, before another retransmission request will be honored for this packet. The value of Tmin_retransmit is stored at the designated receiver along with the packet's sequence number and a count of the number of times the packet has been retransmitted. A request for retransmission which

arrives at a designated receiver during the period of time Tmin_retransmit is ignored.

Tmin_retransmit is calculated as the local RTT of the control channel multiplied by 2 to the power of the number of previous retransmissions of that packet. Tmin_retransmit can never exceed Tmax_retransmit, a fixed system constant.

If the number of retransmissions reaches RXmax, the sender or a designated receiver will ignore further requests for retransmission. The designated receiver will mark the packet as stable. The sender will terminate the stream. The receiver will eventually get a Data Fail Indication.

5.8 NACK Generation

A negative acknowledgment, or NACK, is an explicit request by a receiver or a designated receiver to its parent node for a retransmission of missed data. Networks that operate under low loss conditions can use NACKs, combined with reduced HACK traffic, to get faster recovery of lost Data packets with less control traffic overhead.

A NACK is a HACK packet with the NACK bit, N, set to 1. NACKs are allowed only for NACK enabled data streams.

After detecting a missing packet, a receiver waits for a random time Z before generating a NACK. The time, Z, is random variable having a truncated exponential distribution on the interval [0, T]. Z is obtained from a uniformly distributed random variable, X, on the [0, 1] by the formula:

Z = T/b1*Ln(b2*X - 1)

where
N = 3, This is the expected number of NACKs, if all the Children miss the packet.
T = (0.6)*RTT*b1/Ln(N)
b1 = Ln(B) + 1
b2 = Exp(b1) -1
RTT is the local round trip time to the parent

This backoff policy is discussed in [NB98], which also shows that this policy has a better response time than a uniformly distributed random variable and quantifies how the average latency varies as a function of the average worst case number of responses, N.

If a receiver receives the missing Data packet before the NACK timer

expires, it cancels its NACK. This reduces the chance that multiple receivers generate a NACK for the same packet and suppresses NACKs.

5.9 NACK Response

When a designated receiver receives a NACK, it immediately multicasts the retransmission on the control channel, unless it suppresses retransmission due to a recent HACK or NACK. Retransmissions due to a NACK are suppressed the same way as retransmissions due to a HACK.

If a designated receiver gets a NACK requesting some packets that it has and some packets that it does not have, it clears the retransmission request bits for the packets it has, forwards the NACK to its parent. It retransmits the packets that is has available. If some packets are unavailable, the designated receiver sends a Heartbeat with the NACK Suppression Option to its children. The NACK Suppression option suppresses NACKs from the designated receiver's children. A single NACK Suppression packet can cover multiple packets.

If an aggregator node receives a Heartbeat packet from its parent with the NACK Suppression option, it sends a Heartbeat packet to its children with bits set for all requested Data packets.

When a designated receiver receives retransmissions from its parent, it forwards these retransmissions to its children.

A top node or aggregator that receives a NACK acts just like a designated receiver that does not have the packets for retransmission. The sender for that stream is treated as the parent of the top node.

5.10 Congestion Control

The aggregators and designated receivers at the lowest level of the tree calculate the receiver loss averages based on the HACKs received. Using the HACK bitmap, the control node calculates the loss rate. Assuming the HACK bitmap covers 50 packets and 5 of the packets are missing, the loss rate for that receiver will be 10%. The control node reports the maximum of its children's loss rates to its parent. Finally, the top node passes the loss rate information to the sender.

Congestion control is a required component of RMTP-II. See Appendix - Congestion Control for more detail.

5.11 Failure Detection and Recovery

The Heartbeat, HeartbeatResponse and NullData packets provide the

primary failure detection mechanism in RMTP-II.

5.11.1 Fault tolerant top node

RMTP-II provides a fault-tolerant top node capability by allowing the configuration of a backup top node. The primary function of a backup top node is to monitor the top node's Heartbeat packets. If the top node fails, the backup top node takes over the responsibilities of the top node.

A backup top node is created administratively when configuring the top node and the backup top node. The backup top node is configured as a top node. It joins with StreamID set to zero. The backup top node receives the top node's Heartbeat packets. The backup top node sends HeartbeatResponse packets to the top node to enable the top node to detect the failure of the backup top node.

The top node sends the backup top node's address in the Heartbeat packets to it child nodes. An address of zero for the backup top node indicates that there is no backup top node available.

5.11.2 Top Node Failure Detection

The top node sends Heartbeat packets on its multicast control channel at time intervals Thb. If a child node does not receive a Heartbeat packet from the top node for a period of 2*F*Thb, it declares the top node failed.

If the backup top node does not receive a Heartbeat for an interval F*Thb, then it declares the top node failed.

If a child of the top node does not receive any Heartbeat from the top node for an interval 2*F*Thb, it detects the failure of the top node and the failure of the backup top node. The child node declares the top node failed and, because no backup exists, the child reports the parent unreachable and fails.

5.11.3 Top Node Failure Recovery

The backup top node detects the top node's failure within an interval that is half the interval required by the other children of top node. The backup top node takes over the top node's responsibility and sends Heartbeat packets on the top node's multicast control channel.

When the top node's children start to receive the backup top node's Heartbeat packets, they validate the backup top node's identity and send a JoinStream request with rejoin flag set to 1 to reconnect to the backup top node. The backup top node acquires information about

the children from their JoinStream requests.

5.11.4 Parent Node Failure Detection by a Child

Each control node sends Heartbeat packets on its multicast control channel every interval Thb. All the children of the control node receive the Heartbeat packets. If the child does not receive any Heartbeat from the parent in an interval F*Thb, it declare the parent failed.

5.11.5 Node Failure Detection by a Parent

A parent node receives its child node's HACKs and HeartbeatResponse packets. A child node sends either a HACK or a HeartbeatResponse packet every interval F*Thb. If a parent fails to receive a packet from a receiver child node within an interval 3*F*Thb, it detects the receiver's failure.

If a parent fails to receive a packet from a control node child within an interval 6*F*Thb, it detects the failure of the control node child.

If a the top node fails to receive a HeartbeatResponse from the backup top node for a period of 6*F*thb, it detects the failure of the sender node.

The failure detection interval for a control node or sender node is longer than that of a receiver node because these nodes reside in controlled environments and the probability of their failure is much less than that of a receiver node.

5.11.6 Child Node Failure Recovery

If a control node detects the failure of a child, it sends an Eject packet with the Reason Code set to 1 to indicate that there was no response from the child.

A control node may fail and restart before its children can detect the failure. If a control node receives a HACK or HeartbeatResponse packet from a child node that is not in its child list, it sends an Eject packet with Reason Code 2 to indicate that the parent restarted.

If a child node has information about other control nodes, it can try to reconnect to an alternate parent node. The child node makes a JoinStream request with the rejoin flag set to 1 to join an alternate parent node. The child node picks a random parent node from the list of parents.

If a child node receives an Eject packet from its parent, it must send a JoinStream request to establish its state at the parent. There is no guarantee that the JoinStream will succeed. If the parent rejects the JoinStream request, the child node can try to reconnect to an alternate parent node.

5.11.6.1 Example

Thb = 1 sec.
F = 3
Control node Heartbeat interval = 1 sec.
Control node's failure detection interval = 3 sec.
Receiver's HeartbeatResponse interval = 3 sec.
Receiver child node's failure detection interval = 9 sec.
Control child node's failure detection interval = 18 sec.
If a control node fails, its child receiver nodes detect the parent's failure in 3 sec. The control node's parent detects the control node's failure in 18 sec. The receivers have 15 sec. to connect to an alternate parent node before losing their connection to the RMTP-II tree.

5.12 Control tree discovery

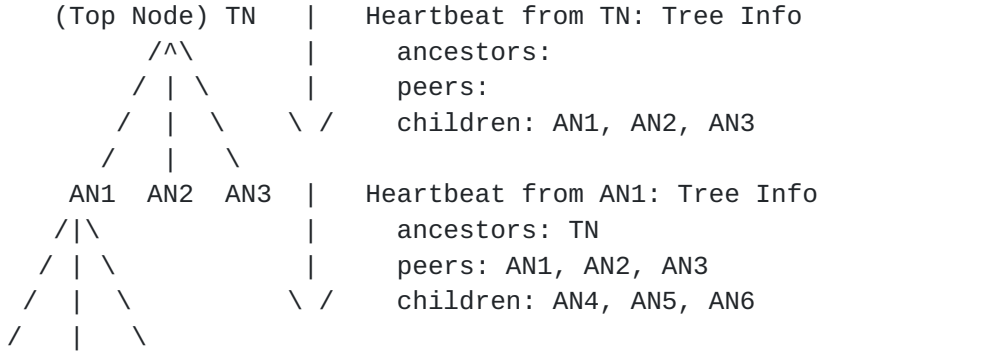
A Heartbeat packet from a control node may optionally include an ancestor list, a peer list, and a child list.

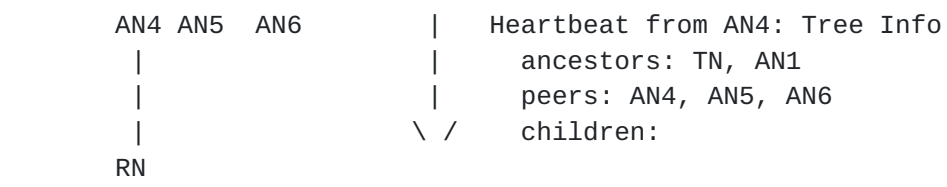
An ancestor list contains the address of the control node's parent and the intervening parent nodes up to the top node. The list is ordered from the top node to the issuing node's parent.

A peer list contains the addresses of the control node and its siblings under the parent node.

A children list contains the addresses of the child nodes of the current node.

A control-node only provides the addresses of the control child nodes from its children list.





The top node, TN, is at the root of the tree and has no ancestor list and no peer list. The top node's children list contains the addresses of AN1, AN2, and AN3.

The node AN1 receives the Heartbeat from the top node, TN, and receives: - an ancestor list with no entries, indicating its parent is
the top node

- a peer list with no entries, confirming that its parent is the top node

- a Children list which contains the addresses of AN1 and its peers.

The node AN1 sends the following lists in the Heartbeat packet to its child nodes:

- an ancestor list containing the address of its parent, TN

- a peer list containing the addresses of it and its siblings, AN

- a children list containing the address of AN4, AN5, and AN6

The node AN4 receives the following lists in the Heartbeat packet from AN1:

- an ancestor list containing the address of TN

- a peer list containing the addresses of AN1 and its peers. If AN1 fails then AN4 will try to join one of its parent's peers AN2 and AN3.
If the join is not successful, AN4 will try to connect to TN.

- a children list containing the addresses AN4 and its siblings

The node AN4 sends the following lists in the Heartbeat packet to its child nodes:

- an ancestor list containing the addresses of TN, AN1, and AN4

- a peer list containing the addresses of AN4 and its siblings, AN5

and AN6

- a children list with no entries; only control children are revealed

The receiver node, RN, receives the following lists in the Heartbeat packet from AN4:

- an ancestor list containing the addresses of TN, AN1, and AN4

- a peer list containing the addresses of AN4 and its peers, AN5 and AN6

- a children list with no entries

If AN4 fails, RN can try to join its peers, AN5 or AN6. If AN4's peers don't respond, RN can attempt to the join AN4's parent.

5.13 Control Tree Round Trip Times

RMTP provides an optional mechanism to get the round trip time (HopRTT), between a node and its children, the round trip time from a node to its bottom-most descendent (TotalRTT-down), and the round trip time from a node to the top node (TotalRTT-up). The RTT Option is used in Heartbeat and HeartbeatResponse packets to calculate and propagate these RTTs.

A child node gets information about HopRTT and TotalRTT-up from the Heartbeats of its parent. A parent node gets TotalRTT-down from the HeartbeatResponse of its children.

5.13.1 Algorithm for RTT

Each node uses the following values:

ResponseDelay: This is the time difference, in milliseconds, between the time that a Heartbeat packet was received from a parent and the time the HeartbeatResponse packet sent to the parent. The child node sends this information in HeartbeatResponse to its parent.

HopRTT: This is the maximum round trip time from a parent node to its children. The parent node calculates this from the TimeStamp and reception time of HeartbeatResponse. The children receive this from the parent in a Heartbeat packet.

TotalRTT-down: This is the node's RTT to its most distance descendent.

TotalRTT-up: This is the node's RTT to the top node.

A parent node sends a first Heartbeat packet with the RTT option to its child nodes to calculate the RTT. It sets the value of TimeStamp in the RTT option to the sending time. Other fields of the RTT option are set to 0.

The parent node receives HeartbeatResponse packets from its children with the following fields:

TimeStamp: This is the original Heartbeat TimeStamp value.

ResponseDelay: This is the time difference, in milliseconds, between the time the child received the Heartbeat packet and the time it sent the HeartbeatResponse packet.

HopRTT: This field is 0 in HeartbeatResponse packets.

TotalRTT-down: This is the total round trip time from the child to its most distant descendent. TotalRTT equals 0 for a node with no children.

The parent node calculates the round trip time for each of its children:

round trip time = reception time - TimeStamp - ResponseDelay

The parent node sets the value of HopRTT:

HopRTT = the maximum of child round trip times. HopRTT = 0 for a node with no children.

The parent node calculates the distance to its most distance descendent:

TotalRTT-down = maximum of the RTT to each child plus the TotalRTT-down values to that child. TotalRTT-down equals 0 for nodes with no children.

The node is also the child of its parent and receives Heartbeat packets from it parent. If the Heartbeat has the RTT option and the TimeStamp value is non-zero, then the parent is requesting information to calculate the RTT. The node returns a HeartbeatResponse with the following values:

TimeStamp: This is the parent Heartbeat TimeStamp value.

ResponseDelay: This is the difference between the time the node

received the Heartbeat and the time it sends the HeartbeatResponse.

HopRTT: This field is 0 in HeartbeatResponse packets.

TotalRTT-down: This is the total round trip time from the node to its most distant descendent, described above.

If the Heartbeat has the RTT option and the TimeStamp value is zero, then the parent is providing the RTT information. The HopRTT and the TotalRTT-up fields have the relevant information. The node calculates its own TotalRTT-up value:

TotalRTT-up = HopRTT plus parent's TotalRTT-up

The node sends the new information in the next Heartbeat packet it sends. The following information is set in the RTT option of the Heartbeat packet.

TimeStamp: This is set to 0.

ResponseDelay: This is set to 0.

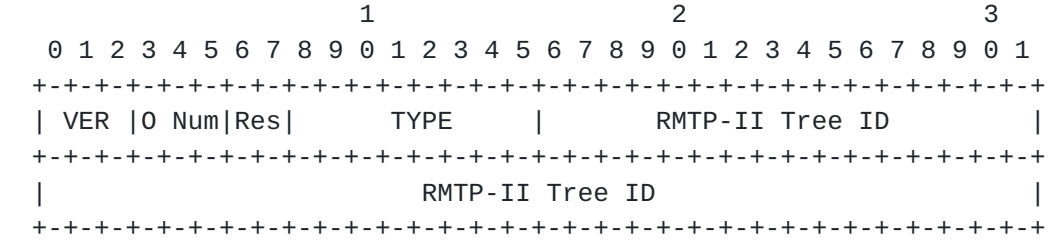
HopRTT: This is the maximum of the round trip times from the parent to its children.

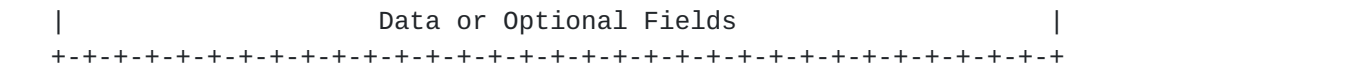
TotalRTT-up: This is the parent's RTT to the top node, determined from Heartbeats from its parent, described above. TotalRTT-up equal 0 for the top node

6 RMTP-II Packet Formats

This section contains the packet formats for the RMTP-II packet types. Each field of the packet is shown and briefly discussed. The packet formats assume a 32-bit addressing scheme.

6.1 RMTP-II Fixed Header





VER: Specifies the version number of the protocol, the first field to be checked by a receiver

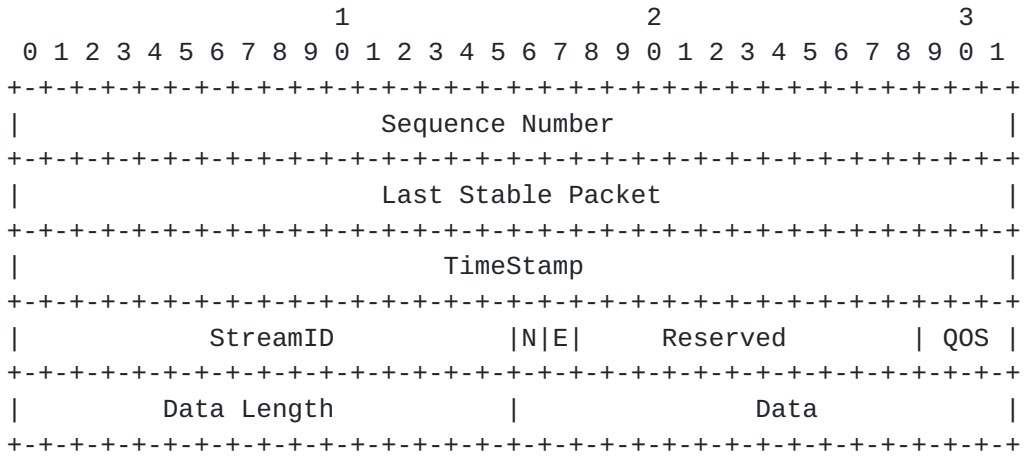
O: a number that specifies the number of options present in this packet. If no options are present, then this field is set to 0.

TYPE: Packet type.

Value	Meaning
0	Reserved
1	Data
2	Retransmission
3	HACK
4	JoinStream
5	JoinAck
6	JoinConfirm
7	LeaveStream
8	Heartbeat
9	NullData
10	Eject
11	EOS
12	HeartbeatResponse
13	LeaveConfirm
14-255	Reserved for Future Use

Tree ID: Tree ID is a six-byte identifier of the RMTP-II tree. Currently Tree ID consists of the four byte IP address and a two byte UDP port of the top node.

6.2 Data



Sequence Number: A sender stamps each Data packet that it sends with a sequence number. Sequence numbers are 32 bits in length and have a valid range of 1 to 2^32-1. RMTP-II does not use sequence number 0 and it is always ignored. Because this range is finite, all arithmetic and comparison with these numbers is Modulo 2^32. RMTP-II requires that no more than 2^31 packets can be created over a period equal to the maximum lifetime for a datagram packet in the network. This condition holds true when IP is used as the datagram service.

Last Stable Packet: This field contains the sequence number of highest numbered stable packet at the sender.

TimeStamp: This is a 32 bit time stamp with one second granularity used to indicate the time when the stream was started.

StreamID: This is a 16 bit ID which uniquely identifies the stream.

N: NACK enabled. This is a flag that is set to 1 if NACKs are enabled for this stream.

E: This is a flag that is set to 1 to indicate the last packet of the stream.

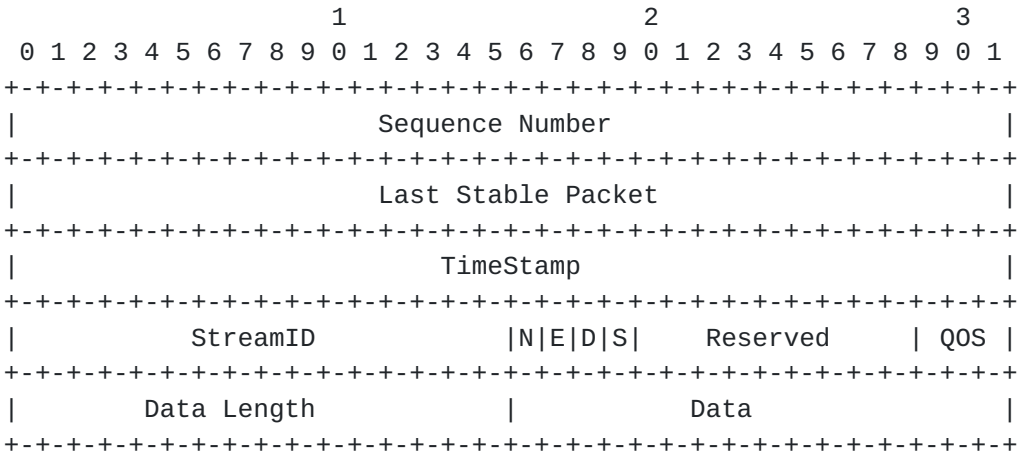
QOS: This specifies the desired quality of service for the Data packet. The valid values for the QoS field are:

2	Unordered
3	Source Ordered
4	Time bound Reliability

Data Length: This is the length of the data in bytes.

Data: This holds the data to be delivered.

6.3 Retransmission Packet



Sequence Number: This is the sequence number of the original data packet.

Last Stable Packet: This field contains the sequence number of the last stable packet at the sender.

TimeStamp: This is a 32 bit time stamp with one second granularity used to indicate the time when the stream was started.

StreamID: This is a 16 bit ID which uniquely identifies the stream.

N: This is a flag that is set to 1 if NACKs are enabled for this stream

E: This is a flag that is set to 1 to indicate the last packet of a stream.

D: This a flag that is set to 1 for a retransmission packet from a designated receiver.

S: This flag is set to 1 to enable subtree multicast for retransmission. If the flag is set to 1, an intermediate control node will not forward the packet to its child nodes.

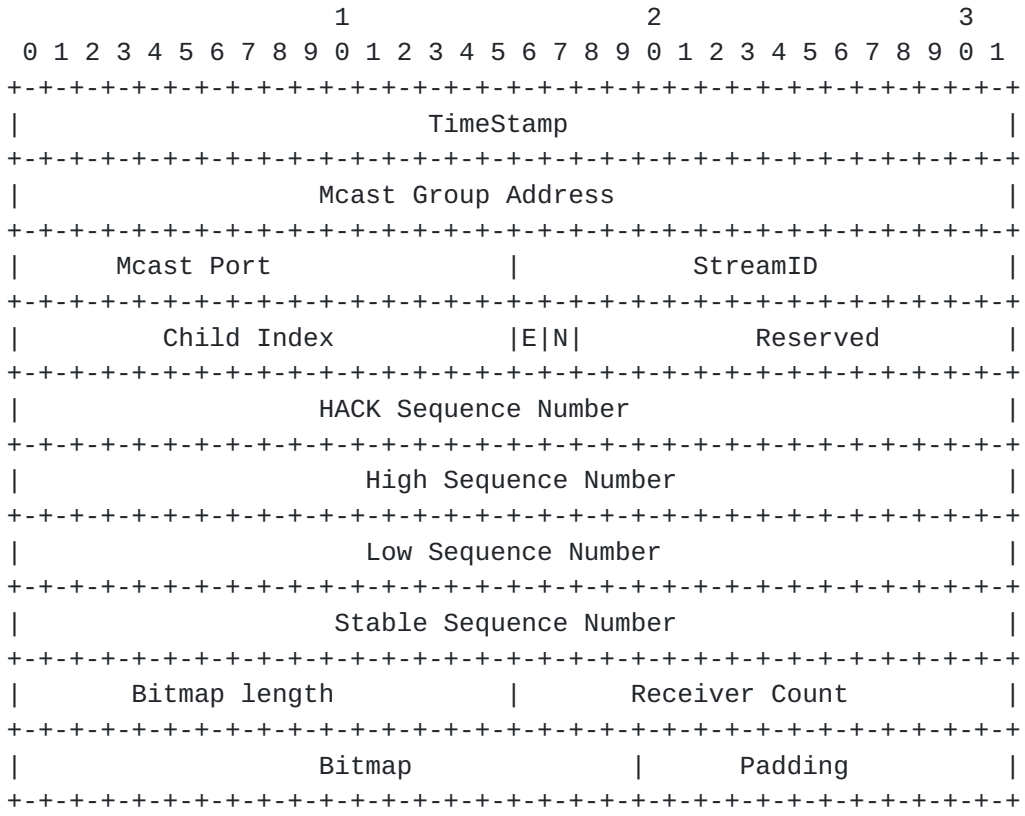
QOS: This is a number that specifies the desired QoS for the Data packet. The valid values for the QoS field are:

2	Unordered
3	Source Ordered
4	Time bound Reliability

Data Length: This is the length of the data in bytes.

Data: Data to be delivered.

6.4 HACK



TimeStamp: This is a 32 bit time stamp with one second granularity used to indicate the time when the stream was started.

McastGroup Address: This is the Multicast group address for this stream.

Mcast Port: This is the UDP port for this stream.

StreamID: This is a 16 bit ID which uniquely identifies the stream.

Child Index: This is the index number of the child node in its parent's child list.

E: This flag is set to 1 for the last HACK, to indicate all packets have been received.

N: This flag is set to 1 to indicate that this is a NACK packet.

HACK Sequence Number: This is the sequence number of the HACK packet.

High Sequence Number: This is the sequence number of the highest

numbered packet received.

Low Sequence Number: This is the sequence number of the lowest numbered missing packet.

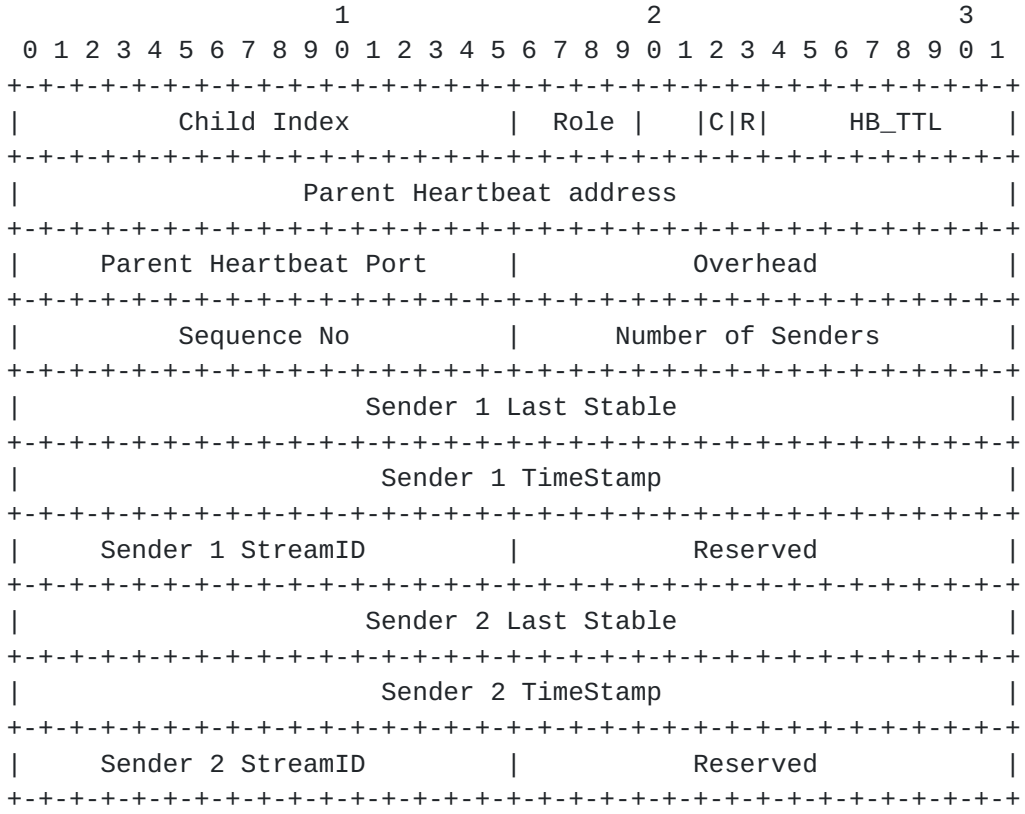
Stable Sequence Number: The stable sequence number is the sequence number of the highest contiguous stable packet known to the node.

Bitmap Length: This is the length of the bitmap in 32 bit words.

Receiver Count: This is the number of receivers connected to this node.

Bitmap: The bitmap of the missing packets between Low Sequence Number and High Sequence Number; starts at bit position Low Sequence Number modulo 32; Leading and trailing bits are ignored.

6.5 JoinConfirm



Child Index: This is the index of the child node in its parent's child list.

Role: This identifies or acknowledges the role of the parent.

C: This flag is set to 1, if the parent accepts the child, or to 0 if the parent rejects the child.

R: This flag is set to 1 to indicate confirmation for a Rejoin packet.

HB_TTL: This is the time-to-live for the Heartbeat packet.

Parent Heartbeat Address: This is the multicast address for the parent's local control channel where Heartbeats are sent.

Parent Heartbeat Port: This is the UDP port for the parent's local control channel where Heartbeats are sent

Overhead: This is the response constant that specifies the number of HACKs that the answering parent expects for each Data packet.

Sequence No.: This is the sequence number of the JoinStream packet.

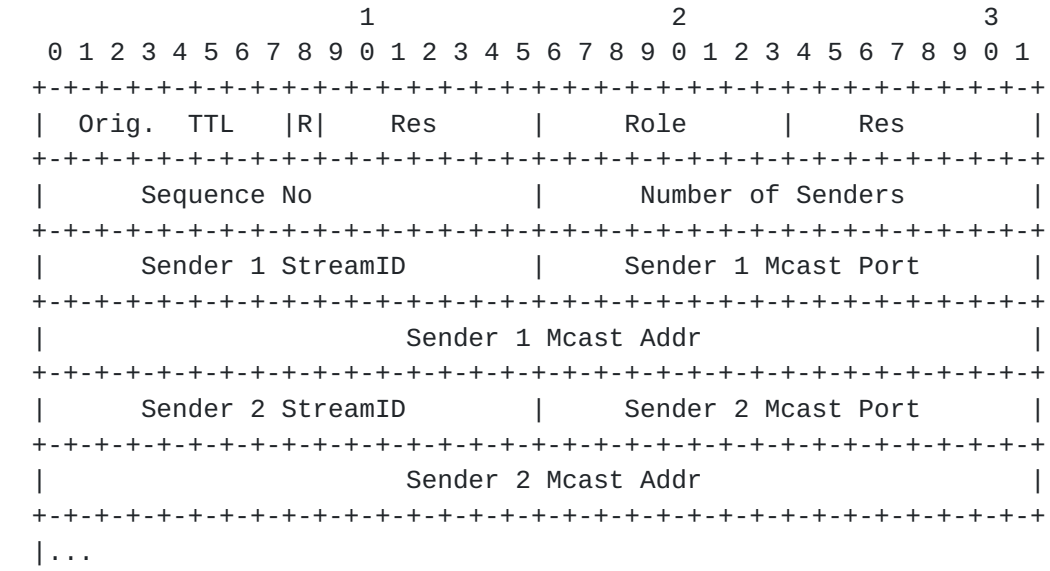
Number of Senders: This is the number of known senders.

Sender Last Stable: This is the number of the last stable packet for the data stream.

Sender TimeStamp: This is a 32 bit time stamp with one second granularity used to indicate the time when the stream was started.

StreamID: This is the identifier of the stream.

6.6 JoinStream



Orig. TTL: This is the original TTL sent from the originator. This field allows receiving node to assess roughly the distance from the source.

R: This flag is set to 1 to indicate a rejoin request.

Role: This specifies the role of the joining node in the RMTP tree.

Sequence No: This is the number of times the same request has been sent.

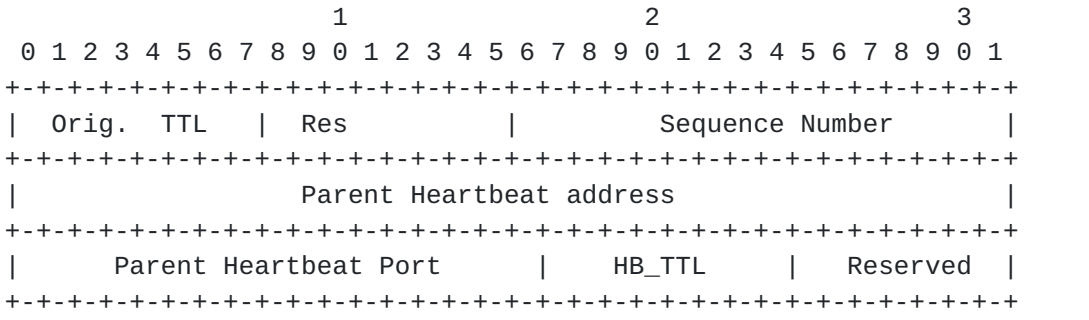
Number of Senders: This is the number of known senders.

Sender StreamID: This is the stream identifier for the sender's data stream.

Sender Mcast Port: This is the multicast port for the sender's data channel.

Sender Mcast Addr: This is the multicast address for the sender's data channel.

6.7 JoinAck



Orig. TTL: This is the original TTL sent from the originator. This field allows a receiving node to roughly assess the distance from the source.

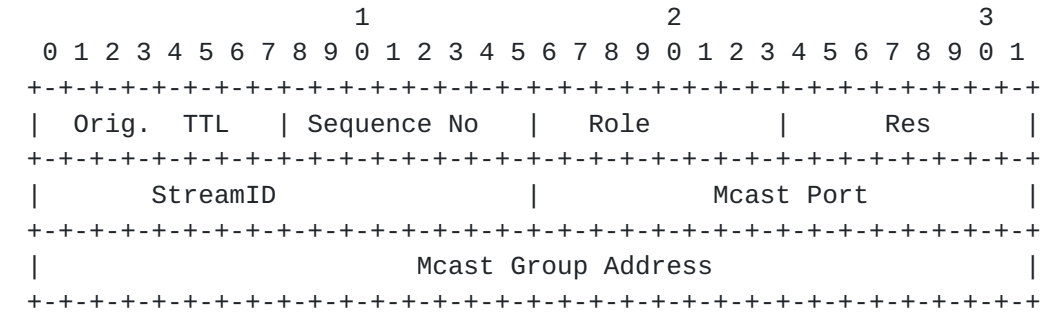
Sequence No.: This indicates how many times the same packet has been sent.

Parent Heartbeat Address: This is the Heartbeat and the parent control multicast address.

Parent Heartbeat Port: This is the Heartbeat and the parent control UDP port.

HB_TTL: This is the time-to-live for the Heartbeat packet.

6.8 LeaveStream



Orig. TTL: This is the original time-to-live sent from the originator.

Sequence no.: This indicates how many times the same packet has been sent.

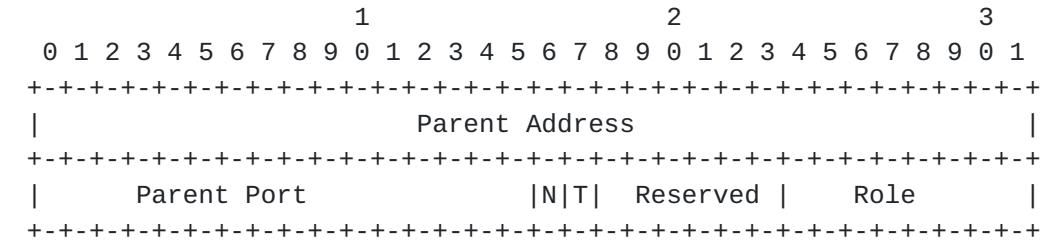
Role: This is the role of the node sending the leave packet.

StreamID: This is the ID of the stream.

Mcast UDP Port: This is the multicast port of this stream.

McastGroup Address: This is the multicast group address of this stream.

6.9 Heartbeat



Parent Address: This is the IP address of the parent.

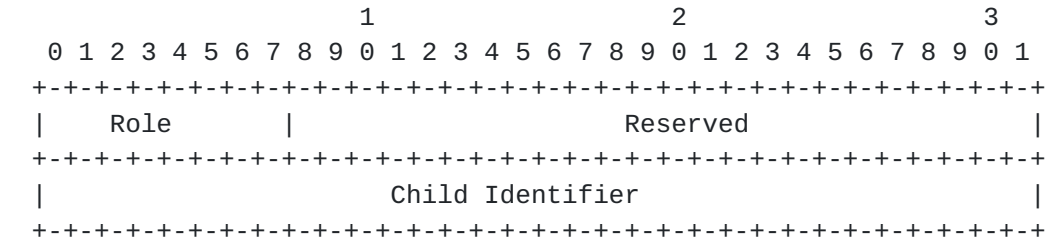
Parent Port: This is the UDP port number of the parent.

N: This flag is set to 1 to indicate that the node must respond immediately.

T: This flag is set to 1 to indicate that the backup top node is taking over as the top node for the tree.

Role: This is the role of the parent in the RMTP tree.

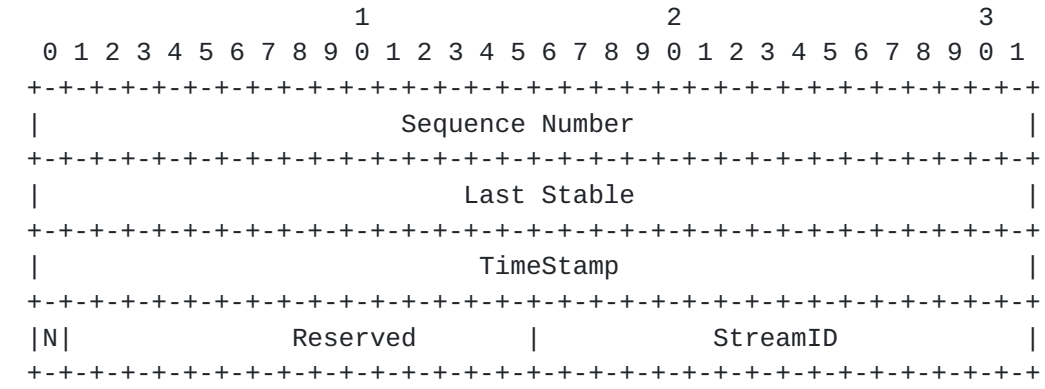
6.10 HeartbeatResponse



Role: This is the role of the node in the RMTP tree.

Child Identifier: This is a unique identifier for the child, the StreamID for a sender node, the Child Index for other nodes

6.11 NullData



Sequence Number: This is the sequence number of the last Data packet sent.

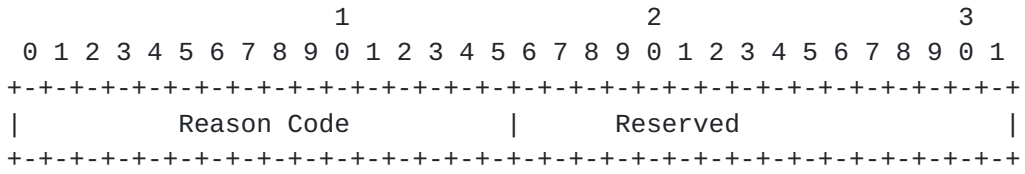
Last Stable: This is the sequence number of last stable Data packet at the Sender.

TimeStamp: This is a 32-bit time stamp with one second granularity used to indicate reincarnation of the sender.

N: This flag is set to 1 to enable NACKs for this stream.

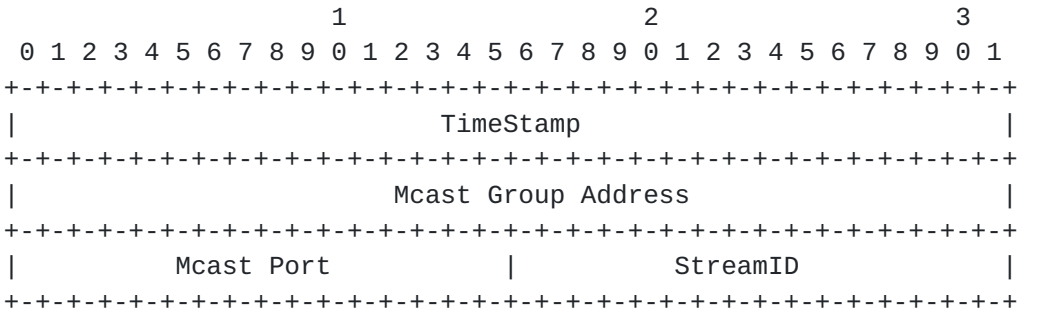
StreamID : This is the 32-bit stream identifier.

6.12 Eject



Reason Code: This code indicates the reason for sending the eject packet

6.13 EOS



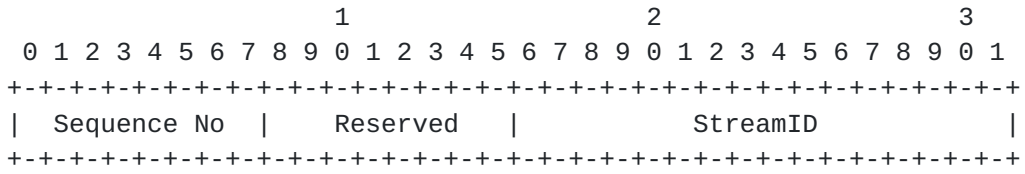
TimeStamp: This is a 32-bit time stamp with one second granularity used to indicate reincarnation of the sender.

McastGroup Address: This is the multicast group address for this stream.

Mcast Port: This is the UDP port number for this stream.

StreamID: This is the unique 32-bit stream identifier.

6.14 LeaveConfirm

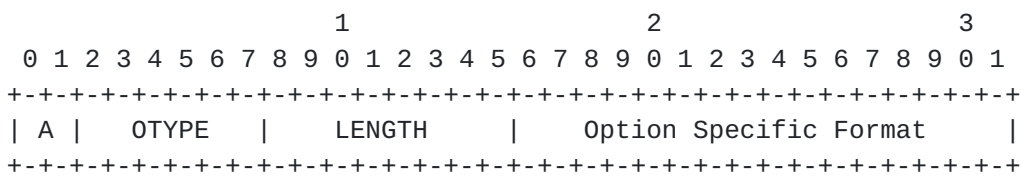


Sequence No.: This is the sequence number of LeaveConfirm packet.

StreamID: This is the unique 32-bit stream identifier.

6.15 RMTP-II Options Header

RMTP-II Options take a general form for their first 32 bits. That form is:



A: a 2-bit flag that indicates how the packet is processed if the option is not supported by the implementation.

00: ignore the option, but process the packet
01: ignore the packet
10: leave the RMTP tree
11: reserved

The flag A is set to 00 to indicate that the option is to be skipped and the packet processed. A is set to 01 to indicate that the whole packet is to be discarded. A is set to 10 to indicate that this is an important option which modifies the functional characteristics of the RMTP protocol and so the node must exit the RMTP tree.

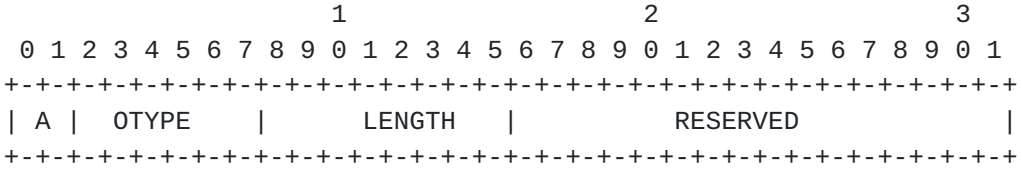
OTYPE: A number that indicates the option type. The predefined values of OTYPE are:

1	Sender Forward
2	FEC
3	Control tree discovery
4	Global parameter
5	fault tolerant backup top node information
6	congestion control information
7	Round-trip Time for control tree
8	Time bounded reliability
9	Sender status Update Option
10	Update Sender Parameters Option
11	NACK Suppression Option
12-63	reserved for future use

LENGTH: Specifies the length of this option block in 32 bit words.

6.15.1 Sender Forward Option

This option allows a sender which does not have multicast capability, to forward data to the top node which will multicast the data to the tree.



A: set to 00 ignore option, process packet

OTYPE: set to 1

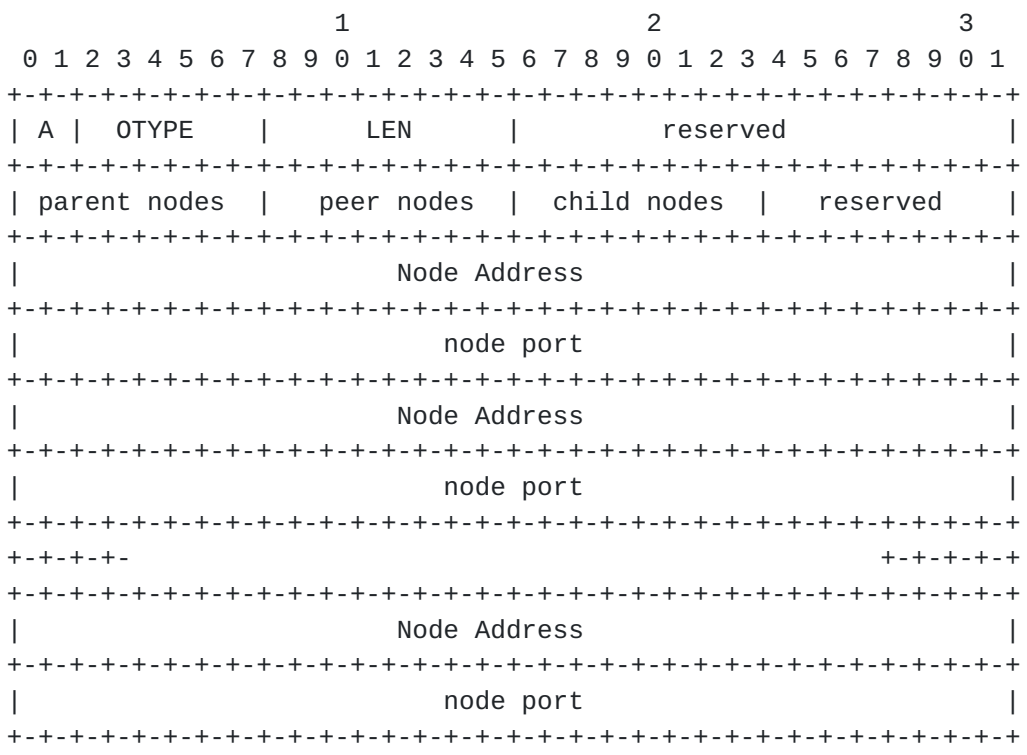
LENGTH: set to 1

6.15.2 FEC Option

See Appendix D for a description of the Forward Error Correction option.

6.15.3 RMTP tree control node Info Option

A control node may optionally include lists of control node addresses in its Heartbeat packet.



A: set to 00 ignore option, process packet

OTYPE: 3

LEN: This is the number of 32-bit words in the option packet.

Ancestor Nodes: This is the number of address-port pairs in ancestor's address list. The addresses are ordered from the top node down to the current node's parent. If the number is zero then this node does not have any parents and must be the top node.

Peer Nodes: This is the number of addresses-port pairs in the peer address list. The peers list contains the addresses of the control node children of this node's parent. The top node has zero peer nodes.

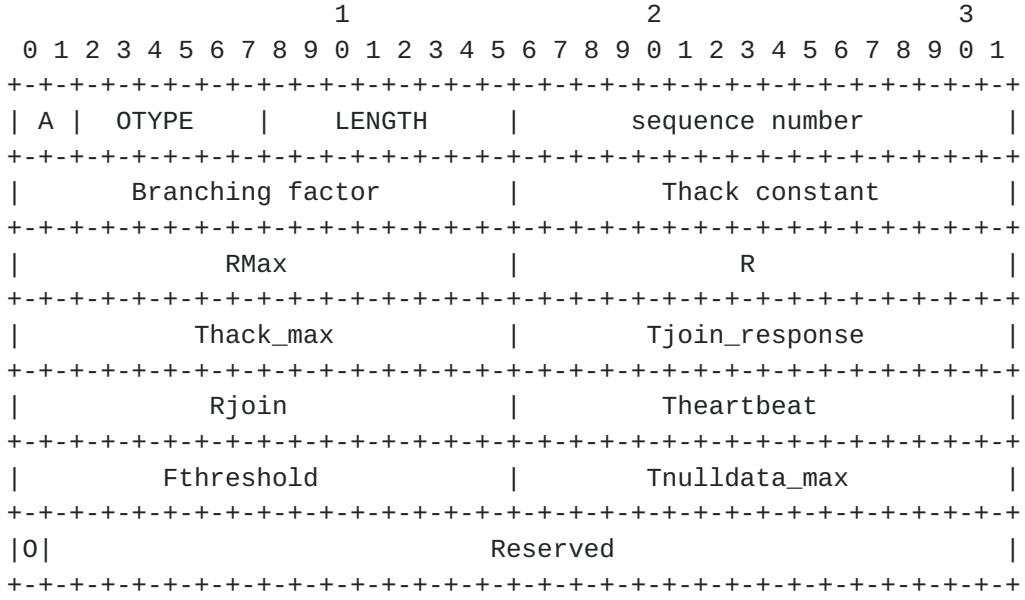
Child Nodes: This is the number of address-port pairs in the child control node list of this node.

Node Address: These entries contain the IP address of the listed nodes.

Node Port: The port number of the listed nodes

6.15.4 Global Parameter Modifications Option

This option enables modifications to the global parameters of the RMTP tree. This option is available for JoinConfirm, Heartbeat, and HeartbeatResponse packets.



A: set to 10 Exit the RMTP tree, this option modified the behavior of the timings.

OTYPE: 4

LENGTH: 6 (6 32-bit words)

Sequence Number: This is the sequence number that identifies this update. This number is set by the top node. All the nodes of the RMTP tree use this sequence number to determine if this is a duplicate packet. The Sequence Number is the only field when the option is use in a HeartbeatResponse packet.

Branching factor: The branching factor B denotes the maximum number of children for any node of the tree.

Thack constant: The Thack constant is a scaling coefficient for Thack.

R: This is a real number which specifies the number of HACKs which should be received, on average, for each Data packet sent, at any node.

RxMax: This is the maximum number of retransmissions allowed for a

data packet.

Thack_max: This number specifies the maximum time allowed between HACK transmissions for each receiver.

Tjoin_response: This number specifies the maximum time to wait for a response to a JoinStream request from the parent node. The response should be either a JoinAck or a JoinConfirm.

Rjoin: This number specifies the number of retries of the JoinStream request before declaring the parent unreachable.

Theartbeat: This number specifies the time interval Thb at which control nodes multicast Heartbeat packets.

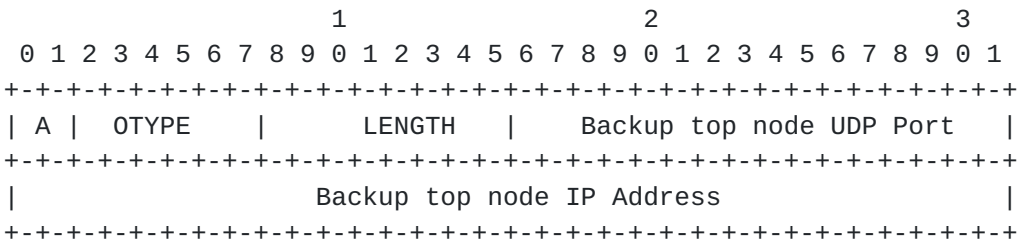
Fthreshold: This number specifies the threshold time for failure detection.

Tnulldata_max: This number specifies the maximum time interval for sending NullData packets. If the receiver does not receive any data packet from the sender within 2*Tnulldata_max, it detects a sender failure.

O: This is a bit flag that is set to 1 to indicate that all designated receivers should use the optimistic HACK mechanism. The flag is set to 0 to indicate the pessimistic HACK mechanism.

6.15.5 Fault Tolerant Top Node Option

This option enables a backup top node.



A: set to 00 skip option, process packet
set to 01 discard packet

OTYPE: set to 5 fault tolerant backup top node information

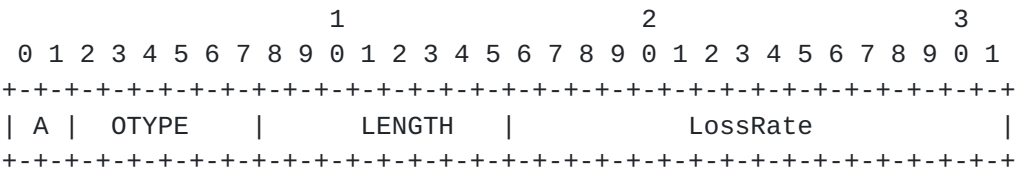
Lenth: set to 2 two 32-bit words in the packet

Backup top node UDP Port:
Set to 0 no backup top node
Set to UDP port of backup top node, if the backup top node exists

Backup top node IP Address:
Set to 0 no backup top node
Set to IP Address of backup top node, if the backup top node exists

6.15.6 LTRC Congestion Control Option

This option enables transmission of loss rate for congestion control.



A: set to 00 skip option, process packet
set to 01 discard packet

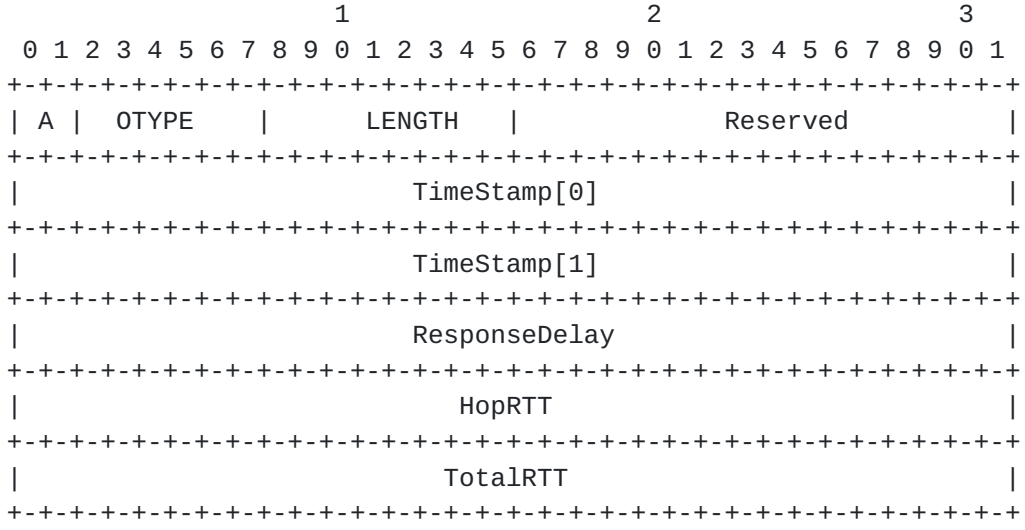
OTYPE: set to 6 LTRC Congestion Control

LENGTH: set to 1 one 32-bit field

LossRate: Set to loss rate value * 100. If the loss rate is , say, 10.34%, then LossRate will be 1034 (10.34*100).

6.15.7 Control Tree Round Trip Time (RTT)

This option is used in heartbeat and HeartbeatResponse packets to calculate the roundtrip time and to propagate the RTT to all the nodes in the RMTP tree.



A: set to 00 skip option, process packet

OTYPE: set to 7: Round Trip Time (RTT)

LENGTH: set to 6

TimeStamp: Set to the current time when a control node wants to calculate the RTT. When a child node receives a Heartbeat with this field set to a non-zero value, it sets the RTT option in the next HeartbeatResponse it sends. If this field is set to zero, then the control node is providing HopRTT-up and TotalRTT-up to its child nodes.

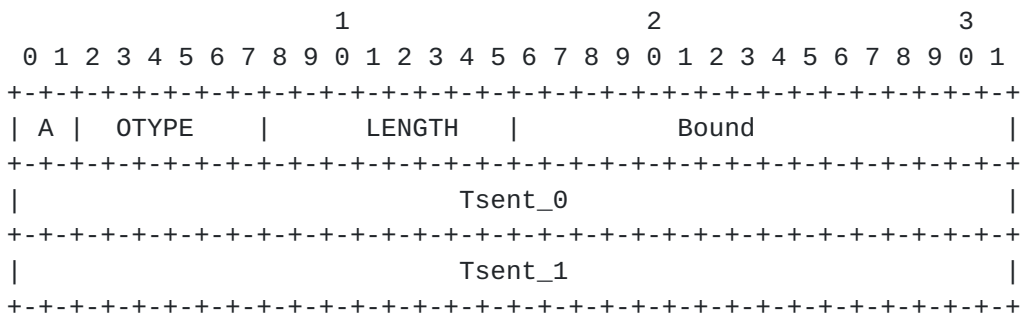
ResponseDelay: This field is set only in HeartbeatResponse packets. ResponseDelay is difference, in milliseconds, between the time the heartbeat packet was received and the time the HeartbeatResponse is sent.

HopRTT: The maximum round trip time from a parent node to child nodes. This is set only in heartbeat packets.

TotalRTT: If the packet is a heartbeat, then this field is TotalRTT-up, the round trip time from top node to the child node. If this packet is HeartbeatResponse packet, then this field is TotalRTT-down, the maximum round trip time from the node sending the HeartbeatResponse to its most distance descendent.

6.15.8 Time Bounded Reliability Option

This option provides support for time bounded reliability and can be used by a sender in the Data packets.



A: set to 10 leave RMTP tree

OTYPE: set to 8 Time bounded reliability

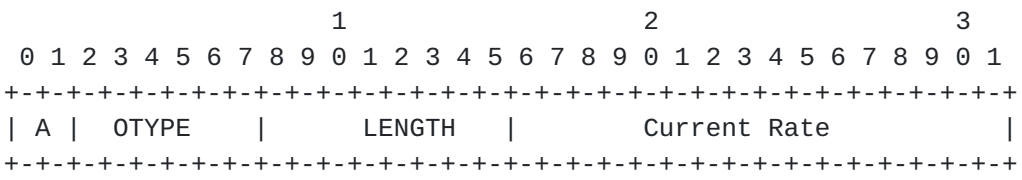
LENGTH: set to 3 three 32-bit words in the packet

Bound: This is the maximum time in seconds allowed for recovery of the packet

Tsent: This is the time at the sender when the packet was generated. The timestamp is expressed in elapsed seconds and microseconds since 00:00 Universal Coordinated Time, January 1, 1970. Tsent_0 contains the seconds part of the timestamp and Tsent_1 contains the microseconds part of the timestamp.

6.15.9 Sender Status Update Option

The sender uses this option in the HeartbeatResponse packet to update top node about current status at the sender.



A:
set to 00 skip option, process packet

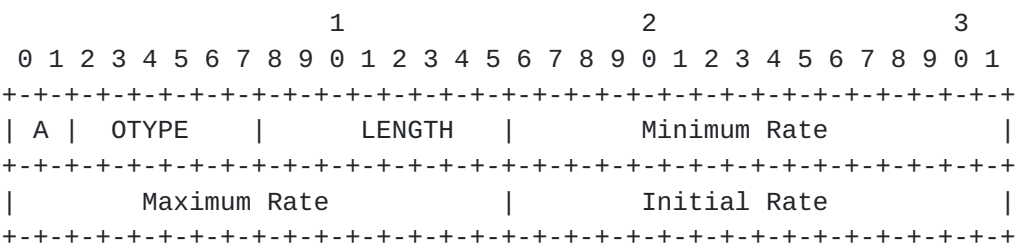
OTYPE: set to 9 This is the Sender Status Update Option

LENGTH: set to 1 There is one 32-bit word in the packet

Current Rate: This is the current admit rate at the sender.

6.15.10 Sender Parameters Update Option

The top node uses this option in heartbeat, HeartbeatResponse, Join, and JoinConfirm packets to pass updated parameters to the sender.



A:
set to 10 leave the RMTP tree

OTYPE: set to 10 This is an Update Sender Parameters Option.

LENGTH: set to 2 There are two 32-bit words in the packet.

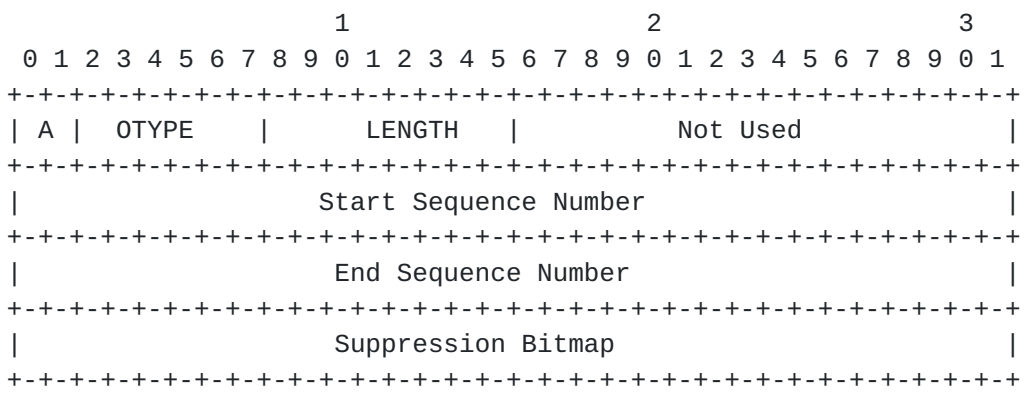
Minimum Rate: This is the minimum admit rate for the sender.

Maximum Rate: This is the maximum admit rate sender for the sender.

Initial Rate: This is the starting rate. If currently active then use Initial Rate as the current rate.

6.15.11 NACK Suppression Option

This option is sent in Retransmission or Heartbeat packets by control nodes to support NACK suppression.



A:
  set to 00 skip option, process the packet
  set to 01 discard packet

OTYPE: set to 11 NACK Suppression Option

LENGTH: set to 3 + Bitmap length

Start Sequence Number: This specifies the lowest sequence number of Data packet, requested for retransmission but not available at the control node.

End Sequence Number: This specifies the highest sequence number of Data packet, requested for retransmission but not available at the control node.

Suppression Bitmap: This is a bitmap of missing Data packets at the control node, requested by at least one child node for retransmission.

7 ACKNOWLEDGEMENTS

RMTP-II draws heavily on the rich work done by the reliable multicast research community over the last decade. The area of tree based reliable multicast started with RMTP [PSK94, PSLB97] and TMTP [YGS95]. RMTP was the first protocol to propose organizing receivers into a tree formation for local recovery and avoidance of ACK implosion. TMTP introduced a number of new concepts, most notably the efficiencies gained by combining local NACKs with global ACKs. The MTP family of protocols [AFK92, BOGKS94] were the first to introduce the notion of different types of nodes having different roles in the communication group. The SRM protocol [FVLMZ96] showed how to use multicast NACKs for efficient local recovery and NACK suppression. PGM [SFLT98] showed how to do NACK suppression without relying on many-many multicast.

The optional forward error correction is taken from [NB96, NBT97, Rizzo97]. The NACK timer backoff algorithms are taken directly from [NB98].

LORAX [LLG96] showed how to have multiple senders share a common acknowledgement tree. Studies have been done comparing different classes of reliable multicast protocols, showing that local recovery is a key piece of a scaleable reliable multicast protocol [NLJBC98], and that the combination of NACKs and ACKs proposed originally in TMTP has the highest scalability of any class of reliable multicast protocols [LG96]. MFTP [MRTW97] was the first protocol to emphasize the importance of support for satellite and asymmetrical networks.

Recent work [PPV98, LG97, LC98, SFLT98] has shown the benefits changing routers to better support reliable multicast. RMTP-II has drawn on all of these ideas, and owes much of its design to these works and many others.

CONTRIBUTORS AND REVIEWERS
Special thanks goes to the following individuals, who have greatly contributed to the design and review of RMTP-II.

Ernst Biersack, Institut EURECOM
Carsten Bormann, University of Bremen, TZI
Rick Buskens, Lucent
Jon Crowcroft, UC London
Christophe Diot, INRIA
Jamal Golestani, Lucent
Brian Levine, UC Santa Cruz
John Lin, Lucent
Don Newell, Intel
Jorg Nonnenmacher, Institut EURECOM

Joerg Ott, University of Bremen, TZI
Christos Papadopoulos, University of Washington, St. Louis
Krishan Sabnani, Lucent
Nils Seifert, Telligue
Muhammad Siddiqui, Lucent
Tony Speakman, Cisco
Paul Stirpe, Reuters
Gursel Taskale, Reuters
Wei Wu, Reuters
Qingming Wang, Lucent
Rajendra Yavatkar, Intel

8 REFERENCES

[AFK92] S. Armstrong, A. Freier, and K. Marzullo, "Multicast Transport Protocol", DARPA RFC 1301, February 1992.

[BOGKS94] C. Bormann, J. Ott, H.-C. Gehrcke, T. Kerschat and N. Seifert, "MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport'', International Conference on Computer Communications and Networks (ICCCN-94), 1994.

[FVLMZ96] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", ACM Transactions on Networking, November 1996.

[Jacobson88] V. Jacobson, "Congestion Avoidance and Control", Computer Communications Review, vol. 18, no. 4, pp.314-s29.

[Jacobson90] V. Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno", Proceedings of the Eighteenth internet Engineering Task Force, p 365, (Sept), University of British Columbia, Vancouver, B.C.

[LC98] D. Li and D. Cheriton. "OTERS: Exploiting the Routing Topology for High-Performance Reliable Multicast", work in progress.

[LG96] B. Levine and J.J. Garcia-Luna-Aceves, "A Comparison of Known Classes of Reliable Multicast Protocols", Proc. International Conference on Network Protocols (ICNP-96), October 1996.

[LG97] B. Levine and J.J. Garcia-Luna-Aceves, "Improving Internet Multicast with Routing Labels," IEEE International Conference on Network Protocols (ICNP-97), October 28 - 31, 1997. p. 241-250.

[LLG96] B. Levine, D. Lavo, and J.J. Garcia-Luna-Aceves. "The Case for Concurrent Reliable Multicasting Using Shared ACK Trees", Proc. The ACM Multimedia '96 Conference, November 1996.

[MRTW97] K. Miller, K. Robertson, A. Tweedly, and M. White. "StarBurst Multicast File Transfer Protocol (MFTP) Specification", <draft-miller-mftp-spec-02.txt>, January 1997, work in progress.

[NB96] J. Nonnenmacher and E.W. Biersack, "Reliable Multicast: Where to use Forward Error Correction", Proc. 5th. Workshop on Protocols for High Speed Networks, Sophia Antipolis, France, Oct. 1996.

[NB98] J. Nonnenmacher and E. W. Biersack, "Optimal Multicast Feedback", Proc. IEEE INFOCOM 1998, March 1998.

[NBT97] J. Nonnenmacher, E. W. Biersack, and Don Towsley. "Parity-Based Loss Recovery for Reliable Multicast Transmission", In Proc. of ACM SIGCOMM '97, Cannes, France, September 1997.

[NLJBC98] J. Nonnenmacher, M. Lacher, M. Jung, E. W. Biersack and Georg Carle, "How bad is reliable multicast without local recovery?", In Proc. of IEEE INFOCOM'98, San Francisco, CA, USA, March 1998.

[PPV98] C. Papadopoulos, G. Parulkar, and G. Varghese, "An Error Control Scheme for Large Scale Multicast Applications", INFOCOM '98, March 1998.

[PSK94] S. Paul, K. Sabnani, and D. Kristol, "Multicast Transport Protocols for High Speed Networks", Proceedings of International Conference on Network Protocols (ICNP-94), 1994.

[PSLB97] "Reliable Multicast Transport Protocol (RMTP)", S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, IEEE Journal on Selected Areas in Communications, Vol. 15, No. 3, April 1997.

[Rizzo97] L. Rizzo, "Effective erasure codes for reliable computer communications protocols", DEIT Technical Report LR-970115.

[SFLT98] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly, "Pragmatic Group Multicast (PGM) Transport Protocol Specification", Internet Draft <draft-speakman-pgm-spec-01.txt>, January 1998, work in progress.

[WMK94] B. Whetten, T. Montgomery, S. Kaplan, "A High Performance Totally Ordered Multicast Protocol", in "Theory and Practice in Distributed Systems", Springer Verlag LCNS938, 1994.

[YGS95] R. Yavatkar, J. Griffioen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications," Proceedings of the ACM Multimedia '95 Conference, November 1995.

AUTHORS' ADDRESSES

Brian Whetten
whetten@gcast.com

Murali Basavaiah
murali@gcast.com

Sanjoy Paul
sanjoy@dnrc.bell-labs.com

Todd Montgomery
tmont@cs.wvu.edu

Naveen Rastogi
naveen@gcast.com

Jim Conlan
jim@gcast.com

Thomas Yeh
yeh@gcast.com