INTERNET DRAFT

draft-kadansky-tram-02.txt

January 2000

Expires: July 2000

M. Kadansky, D. Chiu,
J. Wesley, J.Provino
Sun Microsystems Laboratories

Tree-based Reliable Multicast (TRAM)

Status of this Memo

This document is an updated (and renamed) version of an Internet-Draft titled draft-kadansky-tram-01.txt.

Abstract

This paper describes TRAM, a scalable reliable multicast transport protocol. TRAM is designed to support bulk data transfer from a single sender to many receivers. A dynamically formed repair tree provides local error recovery allowing the multicast group to support a large number of receivers. TRAM provides flow control, congestion control, and other adaptive techniques necessary to operate efficiently with other protocols. Several bulk data applications have been implemented with TRAM. TRAM has been tested and simulated in a number of network environments.

This update contains a new flow and congestion control section, an updated and expanded security section, updated packet formats to

accommodate IPV6 addressing, and several other minor updates.

Table of Contents

1 Introduction

Distributing significant amounts of identical data from a single sender to multiple receivers can take considerable time and bandwidth if the sender must send a separate copy to each receiver. IP multicasting allows a sender to distribute data to all interested parties while minimizing the use of network resources. Many applications, however, require reliable data delivery which can be supported by a reliable multicast transport protocol.

TRAM is designed to provide multicast reliability that scales to a large receiver population. TRAM ensures reliability by using a selective acknowledgment mechanism, and scalability by adopting a hierarchical tree-based repair mechanism. The hierarchical tree avoids acknowledgement implosion and inefficient global repairs by localized repairs.

The receivers and the sender of a multicast session dynamically form repair groups. These repair groups are linked together hierarchically to form a tree with the sender at the root of the tree. The use of a hierarchical tree has been shown to be the most scalable way of supporting reliable multicast transmissions [SURVEY], and is adopted by many other reliable multicast protocols, for example RMTP-II [RMTP].

Every repair group has a receiver that functions as a group head; the rest function as group members. These members are said to be affiliated with their head. Except for the sender, every repair group head in the system is a member of some other repair group. All members receive data multicast by the sender. The group members report lost and successfully received messages to the group head using a selective acknowledgment mechanism similar to TCP's [SACK]. The repair heads cache every data message received and retransmit them at a child's request. A group member may re-affiliate with a different head to improve repair effectiveness and efficiency. This dynamic nature of the tree allows it to react to changes in the underlying network infrastructure without sacrificing reliability.

TRAM has intentionally been kept as lightweight as possible. TRAM has been developed as part of a larger project, the Java(tm) Reliable Multicast(tm) Service [JRMS]. The JRM Service includes support for a wide range of features desirable for reliable multicast: group management, security, receiver customization of data, session advertisement, address allocation, etc. The JRM Service also includes a protocol-independent API, designed to support multiple transport protocols.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. TRAM Components

2.1 Sender

The Sender is the root of the multicast repair tree in TRAM. It transmits the data on the multicast address, initiates and controls the formation of the multicast repair tree, and receives and processes congestion reports from its immediate members.

2.2 Receivers

All members other than the sender are receivers in TRAM. Some of the receivers will retransmit lost packets for other receivers - they are called repair heads.

2.3 Repair Heads

Each repair head has a set of members for which it provides retransmission service. These members are referred to as the children of the repair head. The repair head keeps track of the packets its children have received and those that they missed. The repair head caches a packet until all of its children have acknowledged it. If a child reports that a packet is missing, the repair head retransmits the packet to all of its children by multicasting with appropriate TTL scope.

2.3.1 Eager Repair Heads

Eager heads are members that have been specifically configured to be repair heads. An eager head is expected to have sufficient system resources to cache data packets and service retransmission requests effectively.

The Sender is always an eager head.

2.3.2 Reluctant Repair Heads

Reluctant heads are repair heads that only accept members and perform repairs if an eager head is not available in the area. Reluctant heads solicit members to join their repair group just like eager heads. However, members select reluctant heads only if they do not hear from any nearby eager heads.

The default member role (memberRole) is RELUCTANT_HEAD. Members must be explicitly configured to be EAGER_HEAD or RECEIVER_ONLY members.

3. Key Protocol Elements

3.1 Transport Parameters

TRAM is started at each member with a number of transport parameters. A complete list of these parameters and their default values is included in the Appendix. The following descriptions will refer to these parameters by name.

Some transport parameters are common to all group members. For example, the multicast address, port number, minAckWindow, maxAckWindow, minDataRate, and maxDataRate. These group-wide parameters are typically created once and distributed to all members of the group, for example using SAP (Session Announcement Protocol [SAP]).

Some transport parameters are local, and their values can vary from member to member. An example is transportMode. TransportMode can be set to SEND_ONLY, RECEIVE_ONLY, SEND_RECEIVE, or REPAIR_NODE, depending on whether the member is a sender, receiver, both a sender and receiver, or a repair node only.

3.2 Session Id

The sender generates a sessionId to uniquely identify each session. This id is used to detect multicast address collisions, as well as sender restarts.

3.3 Data Message

A Data Message contains a payload and a TRAM protocol header. The protocol header contains information such as sessionId.

The sender transmits Data Messages using a rate between a minimum and maximum rate (minDataRate and maxDataRate) as specified in the transport parameters.

The sender's current rate is included in the Date Message header. It is used to compute a number of timers, for example, the ackInterval (see section 4.4) and the repair head's retransmission suppression timer (see section 4.6.5).

3.4 Sequence Number

Each data packet sent contains a sequence number. The first data

packet sent contains sequence number 1. This is incremented for each subsequent data packet. Members detect missing packets based on the packet sequence numbers received. Sequence numbers allow the receivers to pass the data packets up to the application in the same order they were sent. Setting the transport parameter ordered to TRUE selects ordered delivery of data packets to the application.

3.4.1 Subtree Sequence Number

While sequence number is a token to identify a data packet, subtree sequence number is used to define a subset of the sequence number space (from 1 to the subtree sequence number) that represent the packets received by all receivers in a subtree. This information is passed from repair head to repair head so that the sender (the root of the repair tree) knows which packets have been received by all the receivers.

3.5 Acknowledgment

Receivers send unicast Acknowledgment Messages to their repair head. The Acknowledgment Message contains a sequence number that indicates all data packets up to (but not including) this number have been received. The Acknowledgment Message can optionally contain a bit mask to indicate missing packets.

The ackWindow is the number of packets a member receives before sending a new Acknowledgment Message to its repair head. The parameter ackWindow is configured as part of the transport profile. A higher value of ackWindow tends to increase the efficiency of the transport protocol since fewer control (non-data) packets will be exchanged. The value of ackWindow is also used as a lower bound of the congestion window, hence it plays an important role in the congestion control algorithm as described in the flow control section (4.6).

A related transport parameter is the ackInterval, which is dynamically computed based on ackWindow, the current data rate, and maximum packet size, to be the current expected time to receive an ackWindow of packets (plus a fudge factor, see section 4.4). The Acknowledgment Message may be triggered by the expiration of ackInterval timers. The value of ackInterval is also used to set timers for tree maintenance control messages, as described in section 4.3.

3.6 Beacon

The sender uses Beacon Messages to signal the start and end of a multicast session. The sender also transmits Beacon Messages after

data transmission has started if the application stops sending data for a period of time. These Beacon Messages act as filler to notify members that the session is still active. Flag bits are used to indicate the purpose of the Beacon Message.

Like Data Messages, Beacon Messages are always multicast to the entire group.

3.7 TTL

All multicast packets, including Beacon Messages, Data Messages and their retransmissions, and other control packets, are transmitted with specifically chosen Time To Live (TTL) values. TTL determines the distance into the network a packet will travel.

Beacon and Data Messages have a TTL large enough to reach all members. This TTL is referred to as the sessionTTL. Only those receivers that receive the Beacon Messages should join the repair tree.

Repair heads set the TTL small enough to only reach their children. This TTL is referred to as the repair TTL.

4.0 TRAM Operation

4.1 Starting a TRAM Session

The Sender transmits Beacon Messages to initiate the session. The Beacon Message is sent to the entire multicast group at regular intervals (beaconInterval). Members begin the tree formation process when they receive a Beacon or Data Message.

After data transmission begins, the sender transmits Beacon Messages only when there is a gap in the application's data stream (see description in Section 4.7).

4.2 Tree Formation

The repair tree in TRAM provides the structure for local repair groups. The repair groups localize repair and control messages, and provide a feedback path from members to the sender. A repair head manages its repair group. Repair group management includes accepting new members, keeping track of children, retransmission of requested data packets, and aggregation of feedback messages from members.

4.2.1 Selecting the Best Repair Head

Each member selects the best repair head it can find. The best

repair head is the closest available head with the most children already attached. The multicast TTL value required to reach the member from the repair head defines the distance between them. A closer head requires a smaller TTL value. Eager heads are selected over reluctant heads if everything else is equal. Selecting a close repair head limits the distance multicast repair packets will travel into nearby networks. It also localizes control traffic between members and their repair heads.

Selecting a repair head with the most children minimizes the number of repair heads. Reducing the number of repair heads minimizes the number of control messages.

Other criteria used to break ties are: greatest maxChildren, and lowest IP address.

4.2.2 Repair Head Capacity (maxChildren)

Repair heads limit the number of children they support to maxChildren. The default is 32 children per repair head. Once the repair head has accepted its maximum number of children, it stops accepting new members until a change in membership causes the member count to go below this limit.

Since the repair tree is critical to the operation of TRAM, each repair head MUST reserve several slots for other repair heads. This guarantees the growth of the repair tree.

4.2.3 Repair Head Discovery

Receivers discover repair heads by using multicast solicitation and advertisement control messages. Some networks such as satellite based networks support multicast capability only in one direction. Such networks typically have slow back-channels that may not support multicast. This is referred to as a uni-directional multicast network, as opposed to a bi-directional multicast network.

There are two basic mechanisms for repair head discovery:

o member-solicited head advertisement
o unsolicited head advertisement

The member-solicited approach is used for bi-directional multicast networks. The unsolicited approach is more suitable for uni-directional multicast networks.

4.2.3.1 Bi-directional Multicast Networks

All members in bi-directional multicast networks can communicate with every other member via multicast. For such environments, TRAM supports a member-solicited repair head discovery algorithm to dynamically build the repair tree.

Receivers join the multicast group and remain idle until the multicast session is detected to be active. Reception of a Beacon Message or a Data Message from the sender signifies an active session. When the session becomes active, the members look for repair heads using a multicast Member Solicit Message. A repair head that is already attached to the repair tree and is able to handle additional members SHOULD respond to a Member Solicit Message by multicasting a Head Advertisement Message. The TTL used in this response is the same used in the Member Solicit Message. If the TTL value required to reach the member is greater than the TTL used to reach the repair head, the Head Advertisements with the TTL from the first Member Solicit Message will not reach the member. Future Member Solicit Messages will have increased TTL values. Eventually the TTL will be large enough for the Head Advertisement Message to reach the member. Repair heads that have not joined the repair tree MUST ignore Member Solicit Messages.

The receiver listens for Head Advertisements after sending the Member Solicit Message. If one or more Head Advertisements are received during a solicitInterval, the best repair head among them is selected. If no Head Advertisements are received, the receiver sends another Member Solicit Message with a larger TTL (incremented by the transport parameter solicitTTLInc). The process of sending the message with an increasing TTL value continues until a response is received. This process is known as Expanding Ring Search [TMTP].

4.2.3.2 Uni-directional Multicast Networks

Uni-directional multicast networks have links that support multicast in one direction. For such networks, TRAM uses an unsolicited head advertisement algorithm for head discovery. This method only requires multicast capability from the repair head to the children.

In a uni-directional multicast network, repair heads multicast Head Advertisement Messages announcing their existence. These messages are sent at regular intervals with an increasing TTL value (advertiseTTLInc). This is repeated until the value of TTL reaches advertiseLimit.

The sender computes this interval, known as the Head Advertisement Interval, as follows:

```
   HAI = max( .5 second, (Heads * HASize) / maxAdvertiseBW1 )
```

Heads: Number of currently advertising heads - this information is aggregated and propagated to the sender by every repair head (via Acknowledgment Messages).

HASize: Head Advertisement packet size

maxAdvertiseBW1: Head Advertisement bandwidth (bytes/second) - configured

The computed HAI is included in every Beacon and Data Message. This gives the sender control over the bandwidth used for head discovery. This is critical because there is no congestion control for tree formation messages. The sender reduces the rate at which each head advertises itself as the number of advertising heads increase.

This formula limits the amount of Head Advertisement traffic to a sender-specified bandwidth based on the number of advertising heads. Another transport parameter, maxAdvertiseBW2, is used to compute the HAI suitable for the time after data transmission has started.

Receivers join the multicast group and remain idle until the multicast session becomes active. Then each receiver listens for Head Advertisements for an Advertisement Listen Interval, computed as 60 seconds or 3 times the Head Advertisement Interval (HAI), whichever is smaller. The receivers MUST ignore the HAI value reported in a retransmitted data message.

If any Head Advertisements are received during this interval, the best repair head is selected. If no head advertisements are received, the receiver continues listening.

4.2.3.3 Discovery Mechanism Configuration

First, each member is configured with the transport parameter memberRole. If the memberRole is not RECEIVER_ONLY, then this member is a potential head.

Another parameter, treeScheme, controls which algorithm to use for discovering repair heads. It can have the following values:

- HEAD_ADVERTISE - This means all heads voluntarily advertise as described in 4.2.3.2.
- MEMBER_SOLICIT - This means heads only advertise upon receiving Member Solicitation Messages, as described in 4.2.3.1.
- COMBINED - This means using HEAD_ADVERTISE before data transmission starts and MEMBER_SOLICIT after data transmission has started.

The above parameters control which tree-formation algorithm is used

by the whole multicast group. Each receiver or repair head may also discover local configuration parameters (for example from a local configuration file). The local configuration information may tell a receiver whether to solicit, which potential heads to use (via given IP addresses and reserved port numbers); it may also tell a receiver whether to be a repair head, whether to advertise and how many members to take.

The combination of the global and local configuration parameters selects automatic tree-formation, static tree formation, or a combination of automatic and static tree formation.

4.2.4 Binding

After selecting the best repair head using one of the above head discovery schemes, the receiver proposes to be a child of the selected repair head with a unicast Head Bind Message.

If the repair head has not reached its capacity, it responds to the Head Bind Message with a unicast Accept Member Message; otherwise, it responds with a unicast Reject Member Message. Accepting a child requires the repair head to cache the received Data Messages until the child acknowledges them. Depending on the lateJoinPref transport parameter (detailed in Section 4.8), the Accept Member Message sent by the repair head MUST indicate the starting sequence number of the message from which data reliability is assured. The Accept Member Message also contains an optional Bit Mask field for the head to guarantee repair of additional non-contiguous packets.

4.2.5 LAN Tree Formation

When several members reside on the same LAN, TRAM attempts to create a repair group on the LAN. This confines the control traffic to the LAN and minimizes the number of heads on the LAN. Members elect a single repair head called the root LAN head. The root LAN head joins the rest of the repair tree as described above.

The root LAN head is elected as follows: potential heads on the LAN send out Head Advertisement Messages with a TTL of 1 and LANState set to Volunteering. An eager advertising head with the greatest capacity (maxChildren) is elected root LAN head. If there are two or more advertising heads with the same capacity, the one with the lowest IP address is elected. If there are no eager heads advertising, a reluctant head is elected. This method is compatible with the method for selecting the best head described in Section 4.2.1. After a period of one Head Advertisement Interval (HAI), the elected root LAN head changes its LANState to LAN_HEAD.

Potential root LAN heads listen for half of the HAI before sending out an advertisement. If a better volunteer or an elected root LAN head is heard from, the potential root LAN head suppresses its advertisement.

If the number of members on the LAN equals or exceeds the capacity of the root LAN head, additional heads, called LAN heads, are elected from the members affiliated with the root LAN head. The root LAN head or current LAN head announces the election using the Elect LAN Head flag in the Head Advertisement Message. This ensures that new members on the LAN will be able to affiliate with a LAN head if one is available.

Like all heads, LAN heads reserve slots for children that are also potential heads. In addition, LAN heads must reserve slots for potential heads that are also LAN members, in order to be able to grow the LAN tree.

Once LAN heads are elected, only the single LAN head that has room for more children continues to send Head Advertisement Messages. Two types of these messages are sent.

The first type has a TTL of 1 and LAN State set to LAN HEAD. These are intended to inform LAN members about the availability of a LAN head.

The other type are Head Advertisements sent to inform off-LAN members of the availability of this head. As described in the above sections, depending on the value of treeScheme, these Head Advertisement Messages may be triggered by the receipt of Member Solicitation Messages, or may be unsolicited. These allow off-LAN members to affiliate with the LAN head while suppressing excess Head Advertisement Messages from other LAN members.

This LAN Tree formation method is used when the allowLANTrees transport parameter is set to TRUE. The default value is FALSE.

4.3 Tree Maintenance

TRAM continuously adapts the repair tree to accommodate members joining and leaving. TRAM also adjusts the tree to changing conditions within the network. Repair heads and their children must continuously monitor each other's performance. A repair head SHOULD remove a child that is unresponsive or cannot keep up with the sender's minDataRate. A child can select a new repair head if its current repair head is not responding, or a better one is available. This continuous maintenance allows the tree to dynamically adapt to

changing membership and network conditions.

4.3.1 Tracking Repair Heads

Each head multicasts a Hello Message to its repair group once per helloInterval, as a form of keep-alive. After data transmission starts, a repair head multicasts a Hello Message before the expiration of a helloInterval when it has received an ackWindow of new data packets. Whenever a repair head performs a retransmission, however, it is counted as if it has sent a Hello Message, since the retransmission serves to assure its children their head is still active.

To make the tree maintenance overhead commensurate with the rate of data transmission, the helloInterval is set to the value of ackInterval (see description in 3.5 and 4.4), with a minimum of 3 seconds:

```
helloInterval = max(3, ackInterval)
```

The Hello Message is sent to the same multicast address and port as the multicast session. The TTL of the Hello Messages, however, is set to the TTL of the repair group, which is the TTL needed to reach the farthest child in the group.

If a child does not receive a retransmission or Hello Message from its repair head during a helloInterval, it sets the Hello Not Received flag in the next Acknowledgment Message it sends. If no Hello or Retransmission Message is received in (maxHelloMisses * helloInterval), the child attempts to locate a new repair head.

When the repair head receives an Acknowledgment Message with the Hello Not Received flag set, it MUST immediately respond to the child with a Unicast Hello Message.

Changes in network conditions can cause the members to lose Hello or Retransmission messages. This can happen when the changes in the network require the repair head to use a TTL that is larger than the previously used value. To adapt to such changes, the repair head increases its repair TTL by repairTTLInc in response to an Acknowledgment Message with the Hello Not Received flag set.

The repair TTL can also become larger than necessary. To fine tune the repair TTL, every child computes its actual TTL distance from the head. To enable this computation, the repair head includes the current repair TTL value in every multicast control message sent to the group. While the repair TTL value assigned in the IP header gets decremented on a hop by hop basis, the TTL in the TRAM header remains

unchanged. The difference between the TRAM header value and the IP header value gives the actual TTL distance. Each child then reports the actual TTL distance via the Actual TTL field in the Acknowledgment Message. The repair heads update each child's TTL distance based on this value. When necessary, the repair heads MUST update the repair TTL in addition to updating a child's TTL distance.

4.3.2 Tracking Children

Repair heads must identify children that become inactive. A repair head knows that a child is alive and well if it receives Acknowledgment Messages from it for every ackWindow of packets. If a child's last acknowledged sequence number is more than two ackWindows behind the sequence number of the latest packet received at the head, it includes that child in the Member Address List of its next Multicast Hello Message. This indicates to those in the Member Address List that their head has not heard from them recently. The children listed MUST respond immediately with an Acknowledgment Message. The repair head repeats this process two more times. If it has still not heard from the child, it SHOULD remove this child from the repair group.

4.3.3 Removing a Child

To remove a child from a repair group, the repair head sends the child a Unicast Hello Message with the Member Disowned flag set. The child must rejoin the repair tree in order to get retransmissions.

4.3.4 Leaving the Repair Group

Any member that is not a repair head can leave the group at any time. The member sends an Acknowledgment Message to its repair head with the Terminate Membership flag set. The repair head removes this child from its member list.

If the member trying to leave the group is a repair head, it SHOULD first send its children a Hello Message with the HState field set to Resigning. This signals the members to locate a new repair head. Members find new repair heads with the methods described in the following subsection. Once all of the repair head's children have terminated their membership, the repair head can leave the group.

4.3.5 Switching Repair Heads

A member can switch to a new repair head if a better repair head is found. If the current repair head is unresponsive, a new repair head is chosen as quickly as possible. A member SHOULD switch to a new repair head if a closer one is found or if the current head is

resigning. In this case, care must be taken to switch to the new one only after all outstanding repairs are received from the old repair head. The new repair head may not be able to provide repairs for packets received prior to the member affiliating.

Hello and Head Advertisement Messages aid in the detection of alternative repair heads in a region. Members SHOULD listen to Hello Messages of other heads in the region not only to learn about better heads but also to maintain a backup repair head list. This backup repair head list enables quicker switching when the current repair head becomes unresponsive. The HState reported in the Hello Message enables members to cache only those repairs heads that are currently accepting members.

A repair head who has lost its own head MUST not accept new members until it has re-affiliated to a new head.

Switching repair heads without checking their level in the tree can result in forming loops that are detached from the rest of the repair tree. To prevent loops from occurring, TRAM specifies a RxLevel parameter that indicates the tree level at which a member or a repair head is operating. The sender is at RxLevel 1, its members at RxLevel 2 and so on. When a repair head attempts to switch its own repair head, it MUST choose a repair head whose RxLevel is lower than or equal to its own. If the reason for the switch is loss of its parent, then the repair head tries to locate a new head for 30 seconds before transitioning to the resigning state.

A head reports its RxLevel periodically via the Hello Message. A member always tracks the head's RxLevel and assigns its RxLevel to be one more than the RxLevel reported by its head. When re-affiliating, if a head sees the RxLevel in the Accept Member Message is higher than its own RxLevel, it MUST proceed to terminate the membership. A member with no children does not need to perform the RxLevel checks when re-affiliating, as it is a leaf node in the tree hierarchy.

The process for affiliating with a new repair head is the same as the initial bind procedure with the following exception. If the member's current repair head is unresponsive and it has one or more missing packets, the member MAY send a Head Bind request to all of the repair heads that it knows about. The member checks each Accept Member Message it receives for a repair head that still has the missing packets available. The member can request retransmission of the missing packets from this repair node. It MUST then select the best repair head from those that accepted it and send an Acknowledgment Message with the Terminate Membership flag set to all of the others.

4.3.6 Pruning

The repair heads must keep track of all members they serve. If one of its children goes off-line, a repair head MUST detect this in time to prune the child from the repair tree before its repair cache fills up.

The repair heads MUST also detect the condition when one (or more) of their members cannot keep up with the sender's minDataRate. This is done by computing a session wide slowness measure described in section 4.6.4.2.

The sender adjusts its data transmission rate in reaction to receivers' feedback on congestion (described in Section 4.6). When the sender starts to operate at minDataRate, the members are told of this condition via the following two pieces of information in the Beacon or Data Messages.

- o Prune Members flag
- o global slowness measure

The global slowness measure is the aggregated value of the slowness measure of all the receivers (see section 4.6.4).

Upon seeing the Prune Members signal, a repair head proceeds to prune a member only if this member's slowness measure corresponds to that indicated with the Prune signal, and this member's slowness measure corresponds to its own rather than one of its descendents.

4.4 Packet Loss Recovery

The job of packet loss recovery is distributed among the repair heads. Each repair head receives Acknowledgment Messages from all its children. The repair heads use this information to retransmit lost packets to their children, and flush their caches.

Members send Acknowledgment Messages to their repair heads on ackWindow boundaries. The first Acknowledgment Message is sent on a random packet within the window. This distributes the Acknowledgment Messages sent from all children of a repair head across the entire window. For example:

If ackWindow is 32 packets, a receiver chooses a random initial packet between 1 and 32 to start sending Acknowledgment Messages to its repair head. If the first Acknowledgment Message is sent when packet 3 arrives, the next Acknowledgment Message is sent when packet 35 arrives, when packet 67 arrives, etc.

Each receiver computes ackInterval after sending an Acknowledgement Message:

```
ackInterval = 1.5 * ackWindow * maxPacketSize / rate
```

The current data rate is derived from the Data Message header. The newly computed ackInterval value is used to set a timer. At the expiration of the timer, an Acknowledgment Message is sent if there are missing packets at this receiver.

As noted in section 4.3.2, a complementary mechanism that ensures each receiver sends Acknowledgment Messages in the presence of packet losses is for the repair head to probe the receivers with its Hello Message by including the member who is behind in the Member Address List. This mechanism covers the case when the member lost many packets and does not realize it is behind.

4.5 Rate Based Transmission

TRAM transmits data packets using a rate-based traffic shaper, which computes the amount of time to delay each packet in order to achieve the current data rate. Basically, the delay is computed as:

```
packet size / current rate
```

The overhead in processing the packet is subtracted from this delay. TRAM then sleeps for the calculated period, sends the packet, and the cycle continues.

The implementation of this traffic shaper must also take into consideration a possibly inaccurate sleep function. For a detailed discussion of this subject, see [Experiences].

4.6 Flow and Congestion Control

The term flow and congestion control are used interchangeably, as we have a single unified algorithm to regulate the speed of transmission to meet network congestion as well as receiver limitations.

There are two flow control parameters both of which are adjusted dynamically: data rate and congestion window. The values of these parameters are constrained by the following configurable parameters:

o minDataRate
o maxDataRate
o ackWindow

The congestion window is not allowed to shrink below ackWindow, and grow above maxCongWindow times ackWindow. The value of maxCongWindow may be set to 4 or 5 in the implementation (and does not need to be configurable).

The data rate will not stray outside of the minDataRate and maxDataRate range. An implementation may choose to enforce a hardwired lower limit to data rate, for example 1 Kbytes/second, or 1 packet/second.

The congestion window is kept and adjusted at every receiver. Each receiver reports back its congestion window to the sender (via repair heads) in the form of a highest allowed sequence number to send, which is part of the Acknowledgement Message. This value is aggregated (via repair heads) at the sender as a single highestAllowedSeqno value.

Before each new data packet is sent, the sender checks for the following condition:

```
      new sequence number > highestAllowedSeqno
```

If this is true, the sender temporarily sends data at the minDataRate. When new acks cause this condition to become false, the sender reverts back to sending data at the current data rate.

The current data rate is maintained at the sender, and adjusted once per ackWindow depending on whether there is (aggregated) congestion reported from the receivers. The data rate adjustment is more aggressive initially (during the slow start phase), and returns to a TCP-compatible (additive increase and multiplicative decrease) algorithm in the steady state phase.

The details of these algorithms are described in the following subsections.

4.6.1 Data Rate Adjustments

The sender's algorithm for adjusting the data rate is different during the slow start and the subsequent steady state phases.

4.6.1.1 Slow Start

The sender initially starts sending data using the following parameters:

```
      rate = (maxDataRate + minDataRate) / 2
      rateIncr = (maxDataRate - minDataRate) / 10
      highestAllowedSeqno = ackWindow
```

For each ackWindow of acknowledged packets, if there is no congestion report, recalculate the following:

```
      rate = rate + rateIncr
```

At the first report of congestion, the slow start phase is over. The sender recalculates the following:

```
      oldrate = rate
      rate = max( 0.5*rate, minDataRate )
      rateIncr = ( oldrate - rate ) / 4
```

4.6.1.2 Steady State

When the sender receives a Congestion Message for an ackWindow, it reduces the data rate by a fraction (rateDecr), or to the minDataRate, whichever is greater. Future Congestion Messages for this ackWindow or previous ackWindows SHOULD be ignored.

```
      rate = max( rateDecr*rate, minDataRate )
```

A reasonable value for rateDecr is 0.75.

In the absence of congestion reports, the sender increases its rate by rateIncr (as determined in the slow start phase). This allows the sender to quickly increase its rate back up to where it had operated prior to the congestion. The new rate is capped by the maxDataRate value.

```
      rate = min( rate + rateIncr, maxDataRate )
```

Successive rate increases SHOULD be separated by at least ackWindow packet transmissions, without receiving congestion reports.

4.6.2 Congestion Detection and Feedback

In TRAM, receivers detect and signal congestion when the number of outstanding missing packets increases from one ackWindow to the next. When this occurs, a Congestion Message for the most recent ackWindow is sent to the member's repair head. The repair head MUST immediately forward a new Congestion Message up the repair tree unless a congestion report for that ackWindow or a later ackWindow has already been forwarded. This reduces the number of Congestion Messages arriving at the sender.

Repair heads can also generate congestion reports based on their cache occupancy. The cache has a Threshold, an implementation-specific parameter typically set to half of maxCache; the cache also has a High Water Mark which is initially set to equal to the Threshold. When the cache occupancy reaches High Water Mark, a Congestion Message for the current ackWindow is generated, and the

"High Water Mark" is incremented by ackWindow. When the cache occupancy falls below Threshold again, the value of High Water Mark is adjusted back down to Threshold. The Congestion Messages triggered by high cache occupancy are treated the same way as those generated by missing packets.

4.6.3 Congestion window Adjustment at the Receivers

The receivers all adjust the congestion window once per ackWindow of packets without congestion. Initially,

```
congestion window = ackWindow
```

Receivers also have a slow start phase during which the congestion window is adjusted more aggressively (once per ackWindow of packets):

```
congestion window = 2 * congestion window
```

As soon as the first congestion condition is detected, they enter the steady state phase. In this phase, the congestion window adjustment is more gradual. When there is no congestion in an ackWindow:

```
congestion window = congestion window + congWindowIncr
```

where congWindowIncr is an implementation constant. The recommended value for this parameter is 1 or 2.

Each time there is congestion in a particular ackWindow of packets, the receivers reduce the congestion window by an among equal to the number of lost packets, still maintaining the congestion window within the bounds of (ackWindow, maxCongWindow*ackWindow).

The window adjustment is used to cut down the amount of future packet losses. Maintaining fairness with other flows is the job of rate adjustments. Therefore it is not necessary to reduce the congWindow size by a multiplicative factor as in TCP.

4.6.4 Flow Control Information in Acknowledgment Messages

In addition to sending Congestion Reports when there is congestion, the receivers embed the following flow control related information in the Acknowledgment Messages:

o highestAllowedSeqno
o slownessMeasure

4.6.4.1 Highest Allowed Seqno

The value of highestAllowedSeqno is directly derived from the congestion window maintained at the receiver:

```
      highestAllowedSeqno = highestReceivedSeqno + congestion window
```

Each repair head aggregates the value of highestAllowedSeqno as follows:

```
      highestAllowedSeqno = min (highestAllowedSeqno(i), i=ith member)
```

and include the value in the Acknowledgment Message to its parent.

4.6.4.2 Slowness Measure

Each receiver also participates in the calculation to determine the slowest receiver in the whole multicast session. This is done by each receiver calculating its own slowness measure. In general, there are a number of ways to characterize slowness of a receiver. One such measure is the percentage of packets lost (PPL) which is chosen for TRAM.

At each receiver, a local percentage of packets lost (PPL) count is computed once per ackWindow packets using an exponential moving average:

```
        PPL(k+1) = 0.75 * PPL(k) + 0.25 * PPL(of current ackWindow)
```

where k is the kth ackWindow. This allows the algorithm to discover the slowest member that has been slow recently rather than based on a long term basis.

At each repair head, the highest value of PPL of all its members (including the head itself) is computed, and forwarded up the repair tree. The aggregation is:

```
      PPL = max (PPL(i), i=ith member)
```

Included in the Acknowledgment Message are the value of the aggregated slowness measure, as well as a flag indicating whether the value is the slowness of the reporting receiver or one of its descendants.

4.6.5 Retransmission Data Rate

The sender retransmits packets to its repair group at the current data rate. Retransmissions are sent before new data.

Repair heads send retransmissions at the current data rate which the

sender includes in the header of data packets.

The repair heads also try to avoid sending duplicate retransmissions. This is done by keeping a log of recently sent retransmissions (remembering the sequence number and time of retransmission). When a new retransmission is about to be sent, it is checked against the recent retransmission log. If the same sequence numbered packet has been sent within

```
      suppression timer = 1.5 (packetsize * ackWindow) / rate
```

the retransmission is suppressed.

4.7 Session Keep-alive

In some sessions, application data may arrive in bursts, rather than all be available at once. In this case, the sender sends Beacon Messages as a form of session keep-alive.

The sender uses an implementation-specific way to determine the beginning of an idle period. For example, one way is to wait for 3 times the the inter-packet-departure time (as described in Section 4.5) before sending the first filler Beacon Message. The wait, however, should not exceed a beaconInterval.

Once the sender determines an idle period has begun, it sends a Beacon Message with the F flag set. The sequence number included in this message is the sequence number of the latest Data Message sent. Additional filler Beacon Messages are sent every beaconInterval.

When a member receives a filler Beacon Message, it SHOULD check to see if it has any missing packets up to the sequence number in the Beacon Message. If so, it should send an Acknowledgment Message requesting repair.

A random delay SHOULD be observed before sending this Acknowledgment Message so as not to congest the repair heads.

When a receiver does not receive any Data Messages or filler Beacon Messages for more than 5 beaconIntervals, it MAY consider that the sender has aborted.

4.8 Late Join

A member joining after the sender has started transmitting data may select the following options for recovering data previously sent:

o NO_RECOVERY - Don't recover anything sent before the receiver

joined the repair tree. The start of the data stream for this receiver is the first Data Message that the receiver received after joining the repair tree.

- o LIMITED_RECOVERY - Recover as much data as possible. This option allows the receiver to request retransmission of all the data packets that the repair head has cached.
- o FULL_RECOVERY - Recover all data sent so far. This is normally not supported. If a member must receiver all or nothing, this option should be selected.

The option is selected using the transport parameter lateJoinPref. The default is NO_RECOVERY. All these options require that the receiver join the multicast repair tree before any data is forwarded to the application. This insures that all subsequent data can be received reliably.

4.9 End Of Transmission

Receivers must be able to determine when the session has completed to ensure they have received all the data before exiting.

When the sender application completes, end of transmission is signaled throughout the multicast group. The sender notifies all members of session completion with a Beacon Message that has the Transmission Done flag set. This packet also includes the sequence number of the last data packet sent. The sender transmits this packet once per beaconInterval until all of its children acknowledge the receipt of all packets sent.

A member sends an Acknowledgment Message immediately after receiving a Beacon Message with the Transmission Done flag set. If there are no missing packets, the member sends an Acknowledgment Message with the Terminate Membership flag set; otherwise, retransmissions of missing packets are requested.

If the member is a repair head, it MUST wait for all of its children to acknowledge and terminate their membership. During the time a head waits for its children to acknowledge, every Hello Message sent MUST contain the final sequence number and the Transmission Done flag set. Those children that failed to receive the Beacon Message react to the Hello Message in the same manner as the Beacon Message. The Hello Messages are sent once every helloInterval after Transmission Done has been signaled by the sender or head. Continuation of Transmission Done signaling via the Hello Message eliminates the need for the sender to send Beacon Messages until every receiver of the session completes.

The members that require retransmissions of data MUST send

retransmission requests in response to every Beacon Message or Hello Message with a Transmission Done flag set.

If a repair head does not receive an Acknowledgment Message from a child within a helloInterval, it includes that child's address in its next Multicast Hello Message. Children that find their addresses listed in the Hello Message MUST respond with an Acknowledgment Message. A repair head SHOULD disown a child that has not responded for 3 helloIntervals.

A repair head completes its head responsibilities when each child has either acknowledged all the packets or been disowned.

5. Security

The fundamental security issues that are to be addressed in an end to end solution that uses a transport protocol such as TRAM, UDP/IP, TCP/IP are -

1. Data Confidentiality - Prevent unauthorized parties from viewing the data.
2. Data Integrity - Ensure messages are not altered during transit.
3. Authentication - Ensure data originated from the expected sender.
4. Access Control - Ensure only authorized parties have access to the data.
5. Denial of Service(DoS)- Prevent disruption of a session by people with malicious intent.
6. Non-Repudiation - Ability to prove that a transaction, or an operation etc., took place when the party initiating the transaction/operation denies ever having initiated the operation.

Addressing all of the above issues at the transport layer can overly complicate the transport protocol. For instance, support for Denial of Service in multicast cannot be completely addressed by the transport layer alone even if complexity was not an issue. Support for Non-repudiation requires collecting details of every transaction and archiving the same for retrieval at a later time. Even though the transport can assist in gathering data, the issues of data management (archiving, retrieval and presentation) is outside the scope of a transport protocol.

In an attempt to reduce the burden on the transport protocol, TRAM addresses the above security issues as a multilayer solution. That is, TRAM handles those issues that are best addressed at the transport layer and leaves other issues to be addressed by the layers above it. The issues that are supported by the transport in TRAM are data integrity, sender authentication and some aspects of DoS.

A reliable transport layer has to keep track of data that is received so as to detect data loss and seek retransmissions. A robust tracking mechanism is required to prevent the transport from malfunctioning. Mechanisms to detect and discard masquerading/spoofing packets has to be supported at the transport layer to make the tracking system robust. Data Integrity and sender authentication aid in providing the required robustness to the tracking system. Data integrity and sender authentication checks are together performed by one module. This module is referred as the signature module in the rest of the document.

Preventing DoS attacks is extremely difficult in the current IP multicast infrastructure. While DoS cannot be prevented, TRAM supports some mechanisms to detect and discard DoS packets.

Support for data confidentiality may require mechanisms to handle encryption key changes during an ongoing session. The area of encryption key distribution and management of the same in multicast is very challenging and is an area of active research. The area of key management and support for data privacy is offloaded to be performed by a module above the transport. This approach enables the layer performing the encryption key management to use the services of the underlying reliable transport layer for distributing key updates efficiently. In TRAM's model, the encryption key update messages and the session data messages can be interleaved and sent on the same multicast address (and thereby share the sequence number space). Sharing the same sequence number space synchronizes the key distribution with key usage, especially when the message delivered is ordered. In other words, when a data message encrypted with a new key arrives, the new key should have also been delivered.

TRAM considers access control operation to be outside its scope. Multicast applications using TRAM will have to address this issue in their space.

TRAM does not specify the details of the algorithms used in the signature module. From TRAM's perspective, the signature module is a separate module (adopting a bump-on-the-side stack approach) whose services are utilized as the packets are sent to the network and received from the network via a well-defined interface (API). The details of distributing the authentication keys and other details required to initialize the signature module are considered to be done via out-of-band means. Further, TRAM recommends the use of asymmetric keys to perform sender authentication. This is to prevent payload substitutions and masquerading packet problems that occur as a result of using symmetric keys in a multicast environment.

TRAM identifies two modes of sender authentication. In the first mode, all the multicast messages sent by the sender/source of the multicast session to the entire multicast group (session scope) carry sender authentication information. In the second mode, every TRAM message exchanged by every node in the system carries sender authentication information (example - exchange of control messages between a head and a member). The first mode is simpler and quicker but is open to DoS attacks. DoS attacks such as feeding false congestion control messages, seeking unnecessary retransmissions, using up member slots in a repair node, etc., can be easily launched in this mode. The second mode is more secure since every message is authenticated but it suffers from added complexity (every participant has to know many other participants' authentication keys to successfully build a tree and exchange messages) and slowness as a result of additional processing. TRAM currently supports the first mode. The mechanics of supporting the second mode is currently being researched and the details of the adopted mechanism will be detailed in a future revision of this draft.

Support for the first mode of sender authentication requires the Beacon and Data message to carry the authentication information. The authentication information is attached at the end. The message SubType field in the multicast data message is considered to be a mutable field. The SubType field is modified in transit by a repair node performing repairs. The mutable fields are required to be zero'd before computing and verifying the authentication information.

Sender authentication operation/steps:

The following describes the authentication process at a TRAM sender:

- TRAM builds a TRAM data packet using the data provided by the application (that is, adds a TRAM header).
- Clears the mutable fields in the header of the built packet.
- Uses the signature module services to compute the authentication information for the packet.
- replaces the mutable fields with actual values.
- adds the authentication information to the end of the data packet
- schedules the packet for transmission.

The following describes the authentication process at a TRAM receiver:

- TRAM strips the authentication information part of the incoming message.
- The mutable fields are set to 0 after making copies of the original values.
- The received data part (without the authentication information), the authentication information and id of the sender that generated the message are passed to the signature module.

- The signature module uses the id to access the appropriate key to verify the authentication information.
- The signature module provides "discard" or "forward" signals to the transport layer depending on the results of the authentication verification tests.

6. Packet Formats

For all the packet formats defined in the following subsections, the Version Number field is set to 2.

6.1 Beacon Message (multicast)

```
       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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Version Number| Message Type  |   Sub Type    |V|   0   |F|D|P|
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          Session Id                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           Length              |  Head Advertisement Interval  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                        Sequence Number                        |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         Source IP Address (Variable)          |
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      +             Authentication Information (optional)             +
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 1

Message SubType: 1

Flags:

P: Set when slow members are to be pruned.
D: Set when transmission is done.
F: Set when used as a filler message.
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

Head Advertisement Interval: The number of seconds between transmission of Head Advertisement Messages. A value of zero disables unsolicited head advertisements.

Sequence Number: The packet sequence number of the last packet sent. If the Transmission Done flag is set, this field indicates the last sequence number; if data

```
                       transmission has not started, this field is zero.

      Source IP Address: IP address of the multicast source (4 bytes for
                         IPV4 and 16 bytes for IPV6).

      Authentication
      Information    : Authentication Information of the message.
```

6.2 Head Advertisement Message (multicast)

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version Number| Message Type  |   Sub Type    |V|     0     |L|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Session Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              |      TTL      | HState| MRole |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  RxLevel      |  LAN State    |        Unicast Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Direct Member Count        |          Capacity             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Address (Variable)                  |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 1

Message SubType: 3

Flags:

L: When set elect Lan Head.
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

TTL: TTL value this packet was sent with. A receiver subtracts the TTL value in the IP header from this TTL to determine the distance to its repair head.

HState:

Accepting Members:	1
Accepting Potential Heads:	2
Not Accepting Members:	3

Member Role:

Receiver Only:	1
Eager Head:	2
Reluctant Head:	3

RxLevel: The level of this member in the repair tree hierarchy.

LAN State:

Disabled:	1
Volunteering:	2
LAN Head:	3
LAN Member:	4

Unicast Port: Unicast port number to communicate with this member.

Direct Member Count: Total number of children for this repair head.

Capacity: The maximum number of children this head is configured to serve.

Source Address: IP address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

6.3 Member Solicit Message (multicast)

```
                         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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Version Number| Message Type  |   Sub Type    |V|     0       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          Session Id                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           Length              |      TTL      | MRole | Rsvd  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | RxLevel       | Reserved      |        Unicast Port           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                  Source Address (Variable)                    |
    ~                                                               ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
    Message Type: 1

    Message SubType: 4

    Flags:
           V: Set when the Source IP address is a IPV6 address.

    Session Id:          The identifier for the current session.

    Length:              The packet's length.

    TTL:                 The original Time to Live used to send this
                         message.

    Member Role:

         Receiver Only:                 1
         Eager Head:                    2
         Reluctant Head:                3

    RxLevel:             The level of this member in the repair tree
                         hierarchy.

    Unicast Port:        Port number that this member is using for
                         unicast communications.

    Source Address:      IP address of the multicast source (4 bytes
                         for IPV4 and 16 bytes for IPV6).
```

6.4 Head Bind Message (unicast)

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version Number| Message Type  |   Sub Type    |V|     0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Session Id                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              |    TTL        | MRole | Rsvd  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Direct Members                |  Indirect Members             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Source Address (Variable)                    |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 3

Message SubType: 6

Flags:
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

TTL: Member computed TTL distance from the head.

Member Role:

Receiver Only:	1
Eager Head:	2
Reluctant Head:	3

Direct Members: Number of children directly reporting to this member.

Indirect Members: Number of members indirectly reporting to this member. This includes all members below this point in the tree.

Source Address: IP Address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

6.5 Accept Member Message (unicast)

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version Number| Message Type  |   Sub Type    |V|     0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Session Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              | BitMask Length|   RxLevel     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Starting Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Source Address (Variable)                    |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    BitMask                                    |
~                    ...                                        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 3

Message SubType: 1

Flags:
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

BitMask Length: Number of valid bits in the BitMask field.

RxLevel: The level of this member in the repair tree hierarchy.

Starting Sequence Number: The base sequence number from which this repair head provides retransmission if requested.

Multicast Address: The multicast address this repair head is supporting.

Source Address: IP address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

BitMask: A bit mask indicating selected data packets earlier than the Starting Sequence Number available for repair. The first bit corresponds

```
                         to (Starting Sequence Number - BitMask Length).
```

6.6 Reject Member Message (unicast)

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version Number| Message Type  |   Sub Type    |V|      0      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Session Id                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              | Reason Code   | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Address (Variable)                     |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 3

Message SubType: 2

Flags:
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

Reason Code:

Accepting Potential Heads:	1
Membership Full:	2
TTL Out Of Limit:	3
Resigning:	4

Source Address: IP address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

6.7 Multicast Hello Message (multicast)

```
                       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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  | Version Number| Message Type  |   Sub Type    |V|    0    |D|A|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                         Session Id                            |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |           Length              |      TTL      | HState| Rsvd  |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |        Reserved               | Ack Member Cnt|    RxLevel    |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  | Unicast Port Number           | Member Count  | Reserved      |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |             Lowest Sequence Number in Cache                   |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |             Highest Sequence Number seen/reported             |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                  Source Address (Variable)                    |
  ~                                                               ~
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                        Member Address List                    |
  ~                           ...                                 ~
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 1

Message SubType: 2

Flags:
A: Immediate Acknowledgment requested from members listed in Member Address List field
D: Set to indicate Transmission Done, in which case the Sequence Number field contains the sequence number for the last packet of the session.
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

TTL: The repair TTL used by this head.

HState: (Head State)

Accepting Members: 1

Accepting Potential Heads:	2
Not Accepting Members:	3
Resigning:	4

Ack Member Count: Number of members listed in the Member
Address Field. This field is valid if the
Acknowledgment flag is set.

RxLevel: The level of this member in the repair tree
hierarchy.

Unicast Port: Unicast port number used in communicating
with this member.

Member Count: Total number of members under this member in
the tree.

Lowest Sequence Lowest sequence number data packet in the cache.
Number: When the data transmission has not started,
this field will be set to zero.

Highest Sequence Highest sequence number data packet received(in
cache)
Number: or detected. When the D flag is set, the sequence
number is the last packet sent by the sender.
When the data transmission has not started,
this field will be set to zero.

Source Address: IP address of the multicast source (4 bytes
for IPV4 and 16 bytes for IPV6).

Member Address List: List of IP Addresses of members that must
respond to the head. This field is set if
the Acknowledgment flag is set.

6.8 Unicast Hello Message (unicast)

```
                           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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Version Number| Message Type  |   Sub Type    |V|       0   |D|
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         Session Id                            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           Length              |    RxLevel    |   Reserved    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                   Source Address (Variable)                   |
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 3

Message SubType: 3

Flags:

D: set when the Member is Disowned
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

RxLevel: The level of this member in the repair tree hierarchy.

Source Address: IP address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

6.9 Data Message (multicast)

```
                           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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Version Number| Message Type  |   Sub Type    |V|   0   |R|D|P|
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         Session Id                            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           Length              |  Head Advertisement Interval  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Sequence Number                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |       Data Length             |  Current ACK Window           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       RESERVED                                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                Flow Control Information                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Data Rate                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                Source Address (Variable)                      |
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                            Data                               |
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                                                               |
      +              Authentication Information (optional)            +
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 2

Message SubType:

Original	1
Retransmission	2

Flags:

P: Set when slow members are to be pruned.
D: Set when the complete data transmission is done.
R: Set when members should reset their flow control information.
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length, comprising of packet header, Data and Signature(if any).

Head Advertisement Interval: The number of seconds between transmissions of Head Advertisement Messages. A value of zero disables unsolicited Head Advertisements.

Sequence Number: Packet sequence number, starting from 1.

Data Length: Length of the Application data.

Current ACK Window: Sender requests each receiver to send an ACK once every Current ACK Window

Flow Control Information: Flow control information for the slowest member of the group. The flow control information is used to help determine which slow member to prune. This information is only meaningful when the P flag is also set.

Data Rate: The data transmission rate at the time this packet was sent.

Source Address: IP address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

Data: Application data.

Authentication Information : Authentication Information of the message.

6.10 Acknowledgment Message (unicast)

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Version Number| Message Type  |   Sub Type    |V|   0   |S|T|H|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Session Id                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length              |        BitMask Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Actual TTL  | Reserved      |        Reserved               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Starting Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Direct Member Count           |  Indirect Member Count        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Direct Heads Advertising      |  Indirect Heads Advertising   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Highest Sequence Number Allowed             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Flow Control Information                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Source Address (Variable)                   |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     BitMask ...                               |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 3

Message SubType: 4

Flags:
H: Hello Message not received
T: Terminate Membership
S: Flow Control Information pertains to a sub-tree member rather than the system which sent the ACK.
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

BitMask Length: Length in bits of valid bits in the

BitMask field.

Actual TTL: The TTL distance from this member to its head, computed as the difference between the original TTL and the residue TTL of the head's Multicast Hello Message.

Starting Sequence Number: Base sequence number for the BitMask. If the data transmission has not started, this field will be set to zero.

Direct Member Count: Number of children

Indirect Member Count: Number of indirect members.

Direct Heads Advertising: Number of children that are currently advertising that they are a head.

Indirect Heads Advertising: Number of indirect members currently advertising that they are a head.

Highest Sequence Allowed: Highest sequence number for the sender is allowed to send at the current data rate.

Flow Control Information: Number reresenting how bad data reception is for this member.

Source Address: IP address of the multicast source (4 bytes for IPV4 and 16 bytes for IPV6).

BitMask: Bitmask representing missing and received packets.

6.11 Congestion Message (unicast)

```
                           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
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Version Number| Message Type  |   Sub Type    |V|     0       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         Session Id                            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           Length              |       Reserved                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Sequence Number                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                   Source Address (Variable)                   |
      ~                                                               ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Type: 3

Message SubType: 5

Flags:
V: Set when the Source IP address is a IPV6 address.

Session Id: The identifier for the current session.

Length: The packet's length.

Sequence Number: Last received sequence number. This
identifies the ackWindow that
congestion is being reported for.

Source Address: IP address of the multicast source (4 bytes
for IPV4 and 16 bytes for IPV6).

7. Discussion Regarding RFC2357

RFC2357 suggests a number of technical criteria for evaluating a reliable multicast transport protocol. In this section, we discuss some of these issues relating to TRAM.

7.1 Performance Analysis and Discussion

The design of TRAM was supported by simulation studies. A description of these simulation studies can be found in [TRAM1].

We developed a simple simulation and visualization model for tree building in Java which can directly interface to the tree building part of the implementation.

We also developed a separate model for studying flow and congestion control algorithms using the Network Simulator (NS) - a public domain tool. Initial simulation results show that TRAM shares network resources with TCP in a fair way [TRAM2]. We intend to participate in developing a suite of reference simulation scenarios for reliable multicast and demonstrate how well TRAM behaves in those contexts.

We believe, however, simulations only characterize protocol behaviors for specific network topologies and dynamics. While it is very difficult to conclusively describe a protocol's scalability, stability and fairness properties, below are some additional observations:

a) scalability

TRAM can scale to potentially very large numbers of receivers if all the receivers have adequate bandwidth (greater than the minimum data rate) between themselves and the sender at all times during the session. Our simulation studies showed that in a 200 node network with some network dynamics, TRAM behaved robustly. Simulation for larger networks and other reference scenarios [SCENARIOS] are underway.

b) Fairness with TCP

The following parameters of TRAM can be externally configured:

- o minDataRate (1000 bytes/sec)
- o maxDataRate (64K bytes/sec)
- o ackWindow (32)

The values in parentheses are default settings. Given a particular setting for these parameters, TRAM tries to be as fair with TCP as possible. As discussed earlier, TRAM uses algorithms similar to TCP, and simulation results are reported in [TRAM2].

In an intranet or a network with differentiated services, these external parameters allow the service operator to do resource allocation out-of-band, for example, by setting minDataRate and maxDataRate to the same predetermined value.

When the minDataRate and maxDataRate are set to be equal, a constant data rate is used. It is most desirable to set this constant rate to be the lowest rate that satisfies the application's need. Setting the rate higher than necessary only increases the chance of some receivers being forced to be pruned from the reliability service.

The ackWindow allows the user to select a particular level of efficiency, as the ratio of control traffic to data traffic is roughly proportional to the ackWindow size. While a large ackWindow allows the transport to be more efficient, it also makes it less responsive to congestion.

The ackWindow parameter also bounds the congestion window to be within the range [ackWindow, maxCongWindow*ackWindow]. The value of maxCongWindow is an implementation-specific constant (suitably set to 4 or 5).

When used in the public Internet, we encourage users to use parameter settings that let the internal algorithms adjust the rate and window to operate at a TCP-friendly level.

c) limiting factors

The following factors have been observed to affect TRAM's scalability

- o Different and varying bandwidth
  TRAM adapts the transmission rate to satisfy the slowest link (or minDataRate). When the capacity of receivers and other network resources vary wildly and/or have wide fluctuations over time, TRAM could be obliged to operate at minDataRate and potentially prune many members.
- o Sub-optimal repair tree
  When the repair tree is sub-optimal, the efficiency of the repair mechanism and the feedback mechanism diminishes. The whole system could have very low efficiency when losses occur.
- o Long feedback delay
  The congestion control algorithm depends on feedback from all receivers. When the number of receivers grows and spreads sparsely in the network, the feedback latency increases quickly. This slows down the sender's ability to quickly react to congestion, or increases the likelihood of rate oscillations.

7.2 Security Discussion

The authors are actively working on this problem, as well as participating in the IRTF Secure Multicast Working Group (SMuG). An updated version of this Draft will include the security specifications.

8 Limitations and Future Work

The design of TRAM was based on a number of choices that make it more suitable for certain applications and not others. Some limitations include:

- o Single Sender - Many data distribution applications (e.g. Pay-per-view and stock information distribution) require only a single sender, whereas many collaborative applications (e.g. shared whiteboard) would require multiple senders. Going from single-sender to multiple-sender increases the complexity of the design and the overhead of the protocol. While currently limited to single sender, TRAM is part of a framework [JRMS] that supports multiple-protocol selection and a common API.
- o Reliance on TTL - To minimize the need for manual configuration, TRAM comes with automatic repair-tree formation and maintenance. Many of the automated algorithms are based on using TTL as a measure of distance. In networks where TTL is not a good measure of distance, some of TRAM's algorithm may operate in non-optimal conditions. In such scenarios, it would be necessary to fall back to using manual configurations to define the repair tree.
- o Security - Secure multicast is still very much a research problem. While parts of the security mechanisms are intertwined with transport (e.g. authentication), other aspects can be decoupled and shared by different transports (e.g. key management). As noted before, TRAM will be integrated with open security mechanisms as standards emerge.

Finally, multicast congestion control is also expected to be updated as more research is done on this hard problem.

9 References

[EXPERIENCES] Chiu D. M, M. Kadansky, J. Provino, J. Wesley, "Experiences in Programming a Traffic Shaper", Sun Microsystems Laboratories Technical Report TR-99-77, http://www.sun.com/research/techrep/1999/abstract-77.html.

[JRMS] Kadansky M., S. Hanna, and P. Rosenzweig, "The Java Reliable Multicast Service: A Reliable Multicast Library", Sun Microsystems Laboratories Technical Report SMLI TR-98-68, September 1998.

[RMTP] Whetten B., M. Basavaiah, S. Paul, T.Montgomery, N.Rastogi, J.Conlan and T. Yeh, "The RMTP-II Protocol", draft-whetten-rmtp-ii-00.txt, Internet Draft, IETF, April 1998, http://www.talarian.com/rmtp-ii

[SACK] Mathis M., J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[SAP] Handley M., "SAP - Session Announcement Protocol", work in progress.

[SCENARIOS] Handley M., "Reference Simulations for Reliable Multicast Congestion Control Schemes", talk at Reliable Multicast IRTF meeting in London, July 1998.

[SURVEY] Levine B. and J. Garcia-Lune-Aceves, "A Comparison of Known Classes of Reliable Multicast Protocols", University of California, Santa Cruz, 1996.

[TMTP] Yavatkar R., J.Griffioen and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", University of Kentucky, 1995.

[TRAM1] Chiu D. M., S. Hurst, M. Kadansky and J. Wesley, "TRAM: A Tree-based Reliable Multicast Protocol", Sun Microsystems Laboratories Technical Report SMLI TR-98-66, September 1998.

[TRAM2] Chiu D. M, M. Kadansky, J. Provino, J. Wesley, "A Flow Control Algorithm for ACK-based Reliable Multicast", paper under preparation.

Acknowledgments

The authors gratefully acknowledge the many contributions of Germano Caronni, Steve Hanna, Phil Rosenzweig, and Radia Perlman.

Appendix: A Table of Transport Parameters

Parameter Name	Description	Value Range & Default setting
ackWindow	the number of packets received before sending an ACK.	[1, 2^16]
advertiseTTLInc	An increment to the TTL value when using expanding ring to send Head Advertisement Messages.	[1, 255] Default: 2
advertiseLimit	The maximum value to be used in the TTL field for sending Head Advertisement Messages.	[1, sessionTTL] Default: sessionTTL
allowLANTrees	A switch to enable or disable LAN tree formation.	TRUE, FALSE Default: FALSE
beaconInterval	The interval between successive Beacon Messages.	[1, 2^32] msec Default: 1000
helloTTLInc	An increment to the TTL value for sending Multicast Hello Messages, when re-adjusting the repair TTL.	[1, 255] Default: 2
lateJoinPref	The preference for data recovery when a receiver joins after data transmission has started.	LIMITED_RECOVERY NO_RECOVERY FULL_RECOVERY Default: NO_RECOVERY
maxAdvertiseBW1	The maximum bandwidth to be used for tree forming before data transmission begins.	[1, maxDataRate] bytes/sec Default: maxDataRate
maxAdvertiseBW2	The maximum bandwidth to be used for tree forming after data transmission begins.	[1,maxDataRate] bytes/sec Default: maxDataRate/20
maxCache	The maximum cache size. A value of 0 means no explicit limit.	[0, 2^32]packets Default: 1200
maxChildren	The maximum number of children supported by this head.	[1, 2^32] Default: 32

maxDataRate	The maximum data rate that the sender can transmit, and heads can retransmit repairs.	[1, 2^32] bytes/sec Default: 64000
maxHelloMisses	The threshold of Hello Messages missed by a child before it considers a parent unreachable (or inoperable).	[1, 2^32] Default: 5
memberRole	A member's role in tree forming: an EAGER_HEAD SHOULD actively seek members; a RELUCTANT_HEAD SHOULD act as a head when no other suitable head is available; a RECEIVER_ONLY member MUST never act as a head.	RECEIVER_ONLY RELUCTANT_HEAD EAGER_HEAD Default: RELUCTANT_HEAD
minDataRate	The minimum data rate for sender to transmit data.	[1, 2^32] bytes/sec Default: 1000
multicastAddr	The multicast address used for the session.	224.*.*.*
ordered	A switch to select ordered delivery or not.	TRUE, FALSE Default: TRUE
port	The multicast port number. The default (4567) is reserved with IANA for TRAM use	Default: 4567
sessionId	An Id used to uniquely identify a multicast session.	[0, 2^32] Default: 0
sessionTTL	The TTL used by sender to send Data and Beacon Packets.	[1, 255] Default: 1
solicitInterval	The interval between Member Solicit Messages.	[1, 2^32]msec Default: 500
solicitTTLInc	An increment to the TTL value in Member Solicit Messages.	[1, 255] Default: 2
sourceAddr	The sender's IP address.	
transportMode	The role of the local transport agent.	SEND_ONLY RECEIVE_ONLY

		SEND_RECEIVE REPAIR_NODE Default: RECEIVE_ONLY
treeScheme	Selection of the method used to form tree; HEAD_ADVERTISE is suitable for asymmetric networks; MEMBER_SOLICIT lets member trigger head advertisements; the COMBINED method starts with HEAD_ADVERTISE and switches to MEMBER_SOLICIT after data transmission begins.	HEAD_ADVERTISE MEMBER_SOLICIT COMBINED Default: MEMBER_SOLICIT
useAthentication	enable source authentication	Default: false

Author's Address

Miriam Kadansky
miriam.kadansky@sun.com

Dah Ming Chiu
dahming.chiu@sun.com

Joe Wesley
joseph.wesley@sun.com

Joe Provino
joe.provino@sun.com

Sun Microsystems Laboratories
1 Network Drive
Burlington, MA 01803