The Java™ Reliable Multicast™ Service: A Reliable Multicast Library

Phil Rosenzweig, Miriam Kadansky, and Steve Hanna

The Java™ Reliable Multicast™ Service: A Reliable Multicast Library

Phil Rosenzweig, Miriam Kadansky, and Steve Hanna

SMLI TR-98-68 September 1998

Abstract:

The Java™ Reliable Multicast Service™ (JRMS) is a set of libraries and services for building multicast-aware applications. It enables building applications that distribute data from “senders” to “receivers” over “channels” with distributed control over content mix. It includes a dynamic filtering mechanism that uses Java software that is pushed into the network for interpreting the data of the receiver. JRMS supports multiple reliable multicast transport protocols via a common programming interface, which provides isolation to applications. Supported transport protocols are selectable by applications based on reliability and type of service needs.

Emerging multimedia or “push” applications can use JRMS as a better platform for reliable delivery of content to very large constituencies. As compared to unicast (point-to-point) protocols), reliable multicast enables broadcasting to groups of receivers, ensuring bandwidth conservation and timely delivery. The JRMS reliable multicast transport protocol (TRAM) is designed for high scalability. JRMS also includes related services for multicast address allocation, channel advertisement and subscription, authentication and encryption, and receiver group management. JRMS is designed to be a flexible toolkit to the application developer for authoring new types of network-centric multimedia applications.

M/S MTV29-01
901 San Antonio Road
Palo Alto, CA 94303-4900

email address:
phil.rosenzweig@east.sun.com
miriam.kadansky@east.sun.com
steve.hanna@east.sun.com

© 1998 Sun Microsystems, Inc. All rights reserved. The SML Technical Report Series is published by Sun Microsystems Laboratories, of Sun Microsystems, Inc. Printed in U.S.A.

Unlimited copying without fee is permitted provided that the copies are not made nor distributed for direct commercial advantage, and credit to the source is given. Otherwise, no part of this work covered by copyright hereon may be reproduced in any form or by any means graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

TRADEMARKS
Sun, Sun Microsystems, and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

For information regarding the SML Technical Report Series, contact Jeanie Treichel, Editor-in-Chief <jeanie.treichel@eng.sun.com>.

The Java™ Reliable Multicast™ Service:

A Reliable Multicast Library

Phil Rosenzweig, Miriam Kadansky, and Steve Hanna

Sun Microsystems Laboratories
Boston Center For Networking
Chelmsford, Massachusetts

1 Introduction

1.1 What is IP Multicast?

IP Multicast is an internet protocol that enables transmission of data packets to a group of receivers. It is described in RFC 1112 [MULTICAST]. Datagrams are sent to a shared class D IP address to which all group members (or hosts) are listening. Hosts may attach to the group by joining or detach from the group by leaving. These actions associate or disassociate the host from the class D multicast IP address. In order for multicast to work, IP routers need to recognize hosts that have joined the multicast group and forward datagrams to each of their downstream links that have hosts on the multicast group. The IGMP protocol [IGMP] provides a mechanism for hosts to communicate with their associated routers to join and leave multicast groups. See [RTNG] for a complete description of IP Multicast routing.

1.2 Why IP Multicast?

Basic TCP/IP network protocols address the need for communicating between one sender and one receiver. This type of communication is known as “unicast” or point-to-point. When a Web browser is pointed at a Web site, TCP/IP provides a unicast connection to enable communications.

There is an emerging class of applications designed for collaborative computing, streaming multimedia, and real-time data delivery. These require multi-point communications between one or more senders and several receivers in order to work well. In this mode of operation, instead of a receiver explicitly requesting data from a sender, data is sent from a sender to the receivers asynchronously as needed. IP Multicast is designed to solve this problem.

For example, when groups of people share data that changes often, it may be necessary to notify everyone when the data changes. Unless it is possible to broadcast changes to all group members at the same time, notifying each member in turn is inefficient. This would require individual point-to-point connections in order for members to receive updates from other members. Obviously, this won’t scale beyond a handful of members.

1.3 IP Multicast Issues

Several challenging issues related to IP Multicast have emerged. The following subsections discuss these issues.

1.3.1 Multicast Routing Protocols

DVMRP [DVMRP], Sparse-mode PIM [PIM-SM], Dense-mode PIM [PIM-DM], MOSPF [MOSPF], and CBT [CBT] are the most common multicast routing protocols in use today. On the MBone, the Internet's experimental multicast backbone [MBONE], DVMRP is most widely deployed. Each of these protocols is designed to deliver packets from senders to receivers on a class D IP Multicast group address.

Shortest path protocols are designed to find the shortest path through the routing infrastructure between each sender and each of its receivers. Shared tree protocols attempt to determine a common path which is shared by all senders on a multicast group. Each approach requires some amount of network traffic and elapsed time in order to build the initial routes (or tree) between

the sender(s) and receivers, and optimize for best path depending on the approach. If group membership is fluid because of frequent or continuous joins and leaves, or additional new senders join the group, or the group membership is very large, routing instability may result.

For example, because DVMRP is a distance vector protocol, it has shown scalability problems in environments such as the MBone which contain large numbers (8000+) of subnets. This limitation is one of the key driving forces behind the transition from DVMRP to a more scalable multicast routing protocol [BGMP] on the MBone.

1.3.2 Product Issues

Although multicast routing protocols are reasonably well-defined, product issues remain in various categories of network equipment. This is because there are very few applications exercising the current infrastructure. These issues will get resolved as more multicast-aware applications become deployed. Some example product issues are:

NICs: Network interface cards (NICs) are the hardware and software devices which connect network nodes to multi-access media networks such as ethernet. NICs usually provide a level of support in hardware for receiving packets on the node's IP address, broadcast address, and multicast address(es). Multicast address support can vary from none to some upward limit. For example, SBUS cards for Sun workstations and servers typically support up to 20 multicast addresses. Beyond this limit requires filtering incoming packets by address in software, which is expensive from a performance perspective.

Switches: LAN switches sometimes treat multicast packets in the same manner as broadcast packets, which results in flooding all ports. This is usually considered a bug.

Routers: Significant bugs exist in many of the multicast routing implementations. In particular, experience with DVMRP on the MBone indicates some unpredictability and instability in its operation, causing some network managers to be reluctant to enable multicast routing in their own networks.

TCP/IP Stacks: Most current TCP/IP stacks support multicast. The key capability is support for IGMP, which is the protocol used for joining and leaving multicast groups. Older implementations often do not support IGMP.

1.3.3 Scalability

Router Tables: As multicast is increasingly deployed, more router table space will be consumed. Internal data structures that occupy router memory are created to indicate where to forward multicast packets. Routers on the tree for a particular multicast group must record state for that multicast group. In addition, depending on the multicast routing protocol used, a router may keep state for multicast groups even if it is *not* involved in transmitting data for the group. For backbone routers, if a large number of concurrent multicast groups are in use, there is the potential for exhaustion of table memory space or available processor bandwidth, which can limit scalability. The MBone routers on the Internet best illustrate this condition.

Congestion Control: The lack of congestion control mechanisms in multicast transports can cause inefficiencies in the bandwidth utilization of the network. This is especially dangerous since a single multicast group can affect large portions of a network. Other traffic, such as well-behaved TCP sessions, may be endangered by poorly behaved multicast flows.

1.3.4 Address Allocation

IP Multicast [IP] identifies a set of IP Multicast addresses (224.0.0.0 to 239.255.255.255) that are used to identify a group of recipients (a multicast group). An IP packet sent to one of these addresses is delivered to all members of the multicast group. These groups may span several IP networks.

Certain multicast addresses, 224.0.0.0 to 224.255.255.255, are reserved by the Internet Assigned Numbers Authority (IANA[1]) for special purposes. Other addresses, known as administratively-scoped addresses [ADMIN] **(**239.0.0.0 to 239.255.255.255), are reserved for use within a predefined administrative scope, such as site-local or organization-local. The rest, known as globally-scoped addresses, operate with a global scope.

At the moment, there are several techniques for allocating multicast addresses, but they all have serious disadvantages and they are not always

[1] See http://www.iana.org

used. This can lead to collisions where two multicast groups are established independently with the same multicast address assigned to both, possibly causing data intended for one group to be received by members of another group. The ideal address allocation scheme should scale up to the size of the Internet and guarantee unique addresses across all networks for a particular session. Although IPv4 allows 2^{28} multicast addresses, reuse is required in order not to run out of addresses over time.

1.3.5 Session Directories

Currently, end-users find out about multicast sessions by consulting a session directory [SDR]. Directory entries are multicast over a well-known multicast address using the Session Announcement Protocol [SAP] and the Session Description Protocol [SDP]. Information associated with particular multicast sessions, such as group address, start time or public key, may be made available by the session directory. These directories that are responsible for advertisement of available multicast sessions must scale or otherwise become an inhibitor to multicast deployment. Unless this information is highly available, potential receivers for individual multicast sessions may not receive timely notification in which case they would not be able to participate. Handley [SCALE] describes the scalability considerations with session directories.

1.3.6 TTL Scoping

Another issue is scoping of multicast groups. It is often useful to contain transmission of multicast traffic to a subset of a network. The Time To Live (TTL) field in the header of IP packets is commonly used to limit the number of router hops a packet is to be forwarded. This field is decremented each time a packet is forwarded by a router. At zero, the packet is discarded. If the TTL is too small, packets may not reach every desired part of the network. If the TTL is too large, multicast packets may be forwarded farther than required. In general, the TTL should be large enough to reach all members of a multicast group.

Several reliable multicast transport protocols adjust the TTL field of repair packets to limit their scope. However, using TTL to contain retransmission of packets to a subset of a multicast group does not necessarily confine the data to the targeted receivers.

1.3.7 Security

In a multicast environment, any node can join a multicast group, then send or receive packets. This is a major concern for applications that require security, either to keep data from being seen by unauthorized observers, to ensure that data is sent only by authorized senders, or just to prevent denial-of-service attacks caused by senders flooding the group with data.

For unicast applications, data can be kept confidential and senders can be authenticated using various encryption and signature techniques. While these techniques can also be used for multicast data, it is more difficult to distribute keying information in a multicast environment. Key distribution and revocation algorithms are complex as they need to consider an arbitrary number of senders and receivers, not just two.

1.4 Reliable Multicast

1.4.1 Introduction

Basic IP Multicast is unreliable. This does not mean that it fails often! It means that if any data is lost during transmission, there is no mechanism for repairing this loss. For video, this just causes a momentary glitch. For some other data (like a spreadsheet), it renders the data useless.

Reliable multicast adds a way to repair lost or damaged multicast data. When a receiver misses one or more data packets, it needs these packets to be retransmitted. This sounds easy, but it's not. In fact, it is an area of active research, with new ideas and protocols coming out all the time. The challenges of supporting reliable multicast include:

- Performing repairs for those receivers that need them
- Not overloading the sender with messages from receivers
- Managing receivers with different abilities (loss rate, bandwidth, latency, etc.)
- Congestion detection, avoidance and control
- Insuring scalability to large receiver groups
- Maximizing throughput
- Handling different types of applications
- Providing data security

1.4.2 JRM Service Objectives

One of the basic assumptions of the JRM Service is that the problem of applying reliability to multicast is best addressed as an end-to-end problem. This means that it is best done from one end system to another, rather than within the routing infrastructure. The need to save state for the purpose of repairing packets on a per packet / per group basis puts an inordinate resource load on routing nodes which typically have limited memory and processor resources. Packet repair requires establishing packet caches at appropriate points throughout the network. Scalability requirements require these caches to be within reasonable proximity of the receivers. Since routers are best-effort packet forwarders and will drop packets if resources become exhausted, higher level protocols must deal with lost packet detection and retransmission when necessary. This would suggest that reliable multicast is best done in higher level protocols and in devices which have sufficient computing resources to provide adequate services.

As seen in the next section, there are a variety of types of applications for reliable multicast. Each of these imposes different requirements on the network. This implies a corresponding variety of reliable multicast protocols in order to address the needs of applications. For example, an application which transfers bulk data or large files from one sender to many receivers (such as software distribution), is very different from a collaborative application, which has multiple senders and receivers (such as sharing a virtual whiteboard). The JRM Service addresses the needs of different application types by providing a framework for including multiple reliable multicast protocols. These protocols are accessible by applications via a common API. Since these protocols are implemented end-to-end, it is straightforward to provide them as they become available and define interfaces for others to plug into. It would be much harder to implement and deploy this mix of protocols in router devices — not only because of the reasons previously discussed, but also because routers are homogeneous devices which are not extensible by others. Router code bases come from a single vendor and tend to be released and installed by customers on long time cycles in a conservative manner. This slow-moving release methodology is analogous to operating systems release, except that operating systems allow users to add drivers and other extensions to provide new functions, within reasonably well-defined bounds. So, again, this argues that end systems are better than routers for timely deployment of reliable multicast protocol technology.

The objectives for the JRM Service are:

- Provide a platform for building multicast-aware applications organized as a set of libraries and services implemented on end stations
- Provide applications with access to multiple reliable multicast protocols through a common API which isolates applications from protocol changes, yet allows selecting appropriate protocols based on application need
- Design a highly scalable bulk data transfer protocol for large numbers of receivers which includes support for congestion detection, avoidance, and control
- Provide services for multicast address allocation, security (authentication and encryption), and administrative policy control
- Flexibility
  - Implement in Java software for multi-platform support
  - Handle simple or complex applications with scalable overhead for supporting services
  - Add new reliable multicast protocols with minimal impact to applications
  - Provide a mechanism for applying administrative policy to data at the receivers (dynamic filters)

2 Reliable Multicast Applications

There are several distinct types of multicast-aware applications. These are characterized by factors relating to such things as how the application behaves, characteristics of the data transmitted, demand placed on the network, connectivity characteristics such as stationary or mobile, and available bandwidth. This section describes multicast-aware application types and corresponding requirements that characterize them.

2.1 Application Characteristics

The requirements listed below are the key requirements of multicast-aware applications for

the reliable multicast protocols used to support them. These substantially affect the nature and design of the reliable multicast protocols used to address the applications. These requirements are:

Number of Senders: Protocols that allow more than one sender (known as "many senders") are substantially more complicated than those that allow only one sender (known as "single sender"), as they have no centralized point of control.

Late Joins: Different applications have different requirements for receiver readiness. Receivers that join the group after some data has been sent are known as *late joins*. If all receivers must have a single start time, protocol design is simplified considerably; late joins are not allowed. If receivers may start at any time and must receive copies of all data sent, a good deal of effort must go into preserving all data. This is known as late join with full data recovery. If receivers may start at any time but do not receive copies of all data sent earlier, less effort is required. This is known as late join with limited data recovery.

Real-time Performance: Some applications require that data be delivered within a certain period of time. Others simply require that it be delivered eventually.

Consistency: Requirements in this area differ widely. Some applications require that all data be delivered to all receivers at exactly the same time (such as stock price delivery). Others require simply that all receivers eventually get the data. Some provide application-level checkpoints at which consistency must be guaranteed.

Ordering: Most applications require data to be delivered in order, but some (such as news feeds) do not.

Reliability: Most applications require full reliability, but some (such as streaming video) can trade off a certain amount of data loss in order to achieve high throughput and timely delivery.

2.2 Types of Reliable Multicast Applications

2.2.1 Bulk Data

Bulk data applications are concerned with transferring files or other large blocks of data. Examples include distribution of corporate data, updating software, loading Web caches, or updating data warehouses. Such applications typically require: one sender per channel, no late joins, no real-time requirement, file level consistency, ordering, and full reliability.

2.2.2 Live Data

Live data applications are concerned with distributing an ongoing flow of small pieces of data. Examples include stock price distribution, news feeds, and event logging. Such applications typically require: one sender per channel, no start time (late joins with limited data recovery), varied real-time requirements (strict for stock prices, loose for some others), varying levels of consistency (extremely strict for stock prices, loose for most others), varying levels of ordering (strict for stock prices, none for news feeds), and varying levels of reliability (strict for stock prices, none for event logging).

2.2.3 Resilient Streams

Resilient streams applications (also known as semi-reliable applications) are concerned with distributing a stream of real-time multimedia data that can handle some losses. Examples include video or audio feeds. While such applications may use unreliable multicast, reliable multicast can provide limited data repair and (more important) congestion control. Such applications typically require: one sender per channel, various start times, moderate real-time requirements, no consistency requirements, no ordering requirements, and loose reliability requirements.

2.2.4 Collaborative

Collaborative applications are concerned with sharing data within a group. Examples include a shared whiteboard, collaborative editing, or controlling a multimedia conference. Such applications typically require: many senders per channel, late join with full data recovery, moderate real-time requirements, various consistency requirements, various ordering requirements, and full reliability.

2.2.5 Hybrid

Hybrid applications are some combination of the above. Examples include Distributed Interactive Simulation (DIS) and other shared virtual reality environments. These applications often have many different requirements for different data streams or at different times. Some of them may be implemented as a combination of the above application types. Others may have special requirements that require special solutions (such as rapid joining and leaving of channels).

2.3 Suitability of the JRM Service for Various Applications

The JRM Service is designed to support many of the application types listed above. As can be seen, the architecture of the JRM Service accommodates multiple reliable multicast protocol implementa-tions. This enables the JRM Service to address varying and perhaps conflicting requirements of multicast-aware applications. The JRM Service includes a transport protocol (Tree-based Reliable Multicast Protocol) [TRAM] that satisfies many of the needs of applications such as bulk data. Conversely, collaborative applications such as a shared whiteboard require a protocol that supports many senders. The JRM Service includes one such protocol, Lightweight Reliable Multicast Protocol [LRMP], and a simple *chat* application that supports multiple senders. In all cases, applications use a common API irrespective of the reliable multicast protocol in use.

The TRAM protocol is most useful with bulk data applications. These are typically single-sender applications with many receivers. TRAM supports ordering and reliability. Moderate real-time requirements are addressed in TRAM with support for maintaining minimum (and maximum) data rates. This is a best effort mechanism to sustain throughput within a range of transmission speeds. More details of the TRAM protocol can be found in section 4.

3 The JRM Service Architecture

3.1 Abstract Model

The abstract model for the JRM Service is that one or more *senders* transmit data on a given *channel* and one or more *receivers* receive this data. In Figure 1, A is a sender on channel E; B, C, and D are receivers. It is possible to have many senders and for a single node to be both a sender and a receiver.

A channel is a multicast transport session that conveys data from one or more senders to multiple receivers. This is conceptually similar to an IP Multicast group, but reliable delivery, security, and other features may be added on.

Each channel has one or more *channel managers* that are responsible for creating and destroying the channel, configuring it, and managing channel membership (that is, controlling who can send and receive on it). In Figure 1, M is the channel manager.

These are the basic components in the JRM Service abstract model: channels, senders, re-ceivers, and channel managers.

3.2 Systems

Figure 2 illustrates the major systems that make up the JRM Service .

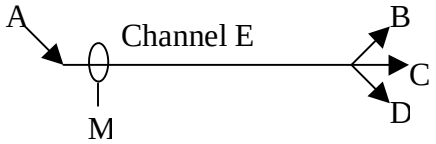

Figure 1: JRM Service Abstract Model

3.2.1 Transport

The transport system is responsible for providing a reliable multicast transport. It may be used directly by the application or indirectly through the channel management system. The transport system is composed of two parts, a set of protocol-independent transport APIs and one or more transport protocols.

3.2.1.1 Transport APIs

Many different reliable multicast transport protocols have been developed and many more are on the way. Currently, application vendors that want to use reliable multicast must develop code specifically to support one protocol. If another protocol comes out that meets their needs better, they must reintegrate their code. The JRM Service provides a interface that allows application vendors to write their code once and have it work with any reliable multicast protocol.

The JRM Service is provided with three protocols implemented: Tree-Based Reliable Multicast [TRAM], Lightweight Reliable Multicast [LRMP], and an unreliable multicast transport. Other protocols can easily be added using stream or packet interfaces.

3.2.1.2 The JRM Service Transport Protocol

The JRM Service Transport Protocol [TRAM] is a reliable multicast protocol designed to support bulk data transfer with a single sender and multiple receivers. TRAM uses dynamic trees to implement local error recovery that scales to a very large number of receivers without imposing a serious burden on the sender. It also includes congestion control and other techniques necessary to operate efficiently and fairly with other protocols across the wide variety of link and client characteristics that make up the Internet.

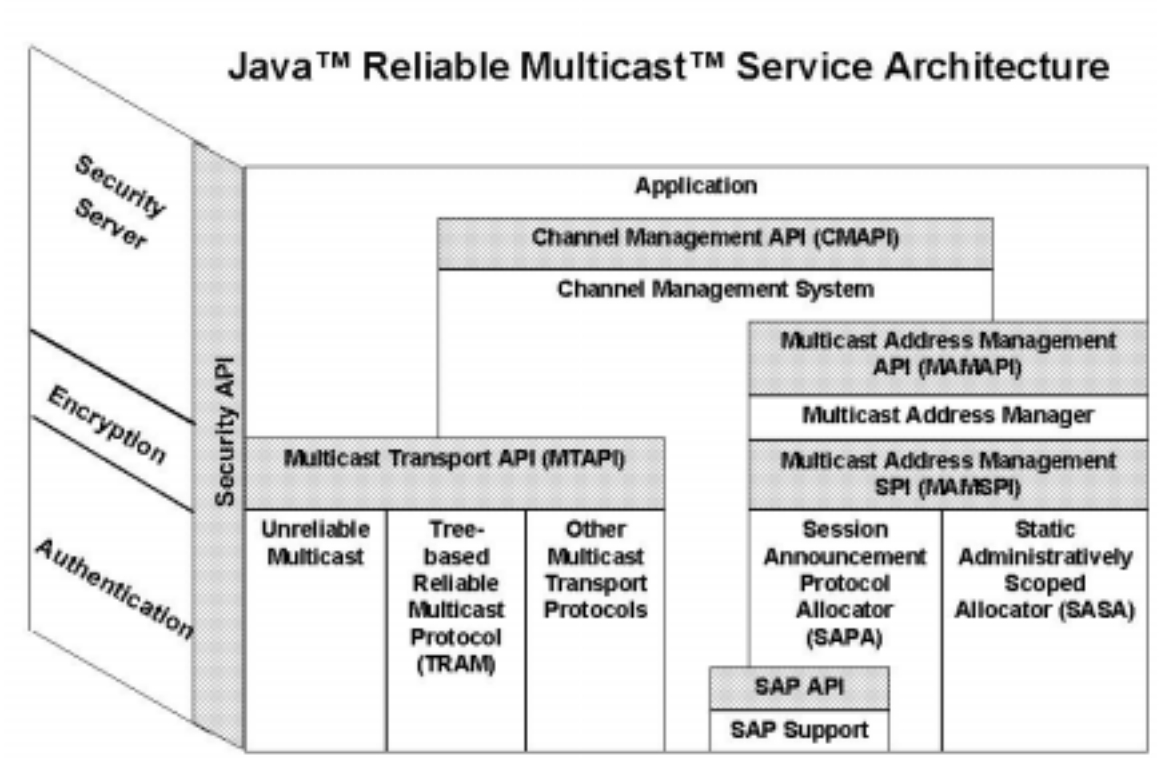

Figure 2: The JRM Service

3.2.2 Channel Management

The channel management system is responsible for creating, configuring, and destroying channels upon request. It handles end-to-end management of the multicast session, including session advertisement, discovery and joining, receiver authentication, event notification and distribution of dynamic filters. Operations at the channel layer are generic to the transport protocol in use.

A channel may be configured with one or more dynamic filters. A *dynamic filter* is a piece of Java software that runs on a receiver, which may be an end-station or intermediate node, and processes the data received on a channel, transforming it in some way. For instance, a dynamic filter can block access to entertainment channels during working hours. It can also highlight stories that match a user's interest profile.

Dynamic filters are delivered to receivers when they register for a channel. Different receivers may receive different filters, based on access control or preferences.

3.2.3 Address Allocation

The address allocation system is responsible for managing multicast address allocation. It may be used directly by the application or indirectly through the channel management system.

One of the first steps in establishing a new channel is choosing a multicast address. Because multicast addresses are shared across multiple networks, this is a tricky problem. Several techniques are in use today, but no one technique addresses every issue. The address allocation system provides a standard API that may be used to allocate and deallocate addresses. Different address allocators may be installed to implement different allocation techniques.

3.2.4 SAP Support

The SAP Support system provides an implementation of the Session Announcement Protocol [SAP] and Session Description Protocol [SDP]. These protocols are useful both for allocating multicast addresses and for advertising multicast sessions. The SAP Support classes are used by both the Address Allocation and Channel Management systems.

3.2.5 Security

Security solutions for reliable multicast are much more complex than those for unicast [SECURITY]. The JRM Service supports three aspects of security:

- data confidentiality
- data integrity and message authentication
- sender and receiver authentication and access control

Data confidentiality is achieved by encrypting data sent on the channel. Data integrity is achieved by including a message digest. In order to support a wide variety of implementations of data confidentiality and integrity, the JRM Service provides generic interfaces for confidentiality and authentication. Confidentiality is provided by a layer just above the transport; message and sender authentication is supported within the transport layer. The JRM Service provides implementations of several solutions using these open interfaces. New implementations can be plugged into the system via the supported open interfaces.

User authentication, access control, and key distribution are implemented as part of channel management. Senders and receivers identify themselves to the channel manager, and, after authentication, securely receive appropriate keying information. For instance, receivers are sent decryption and signature verification keys, while senders obtain encryption and signing keys.

4 TRAM

4.1 Overview

TRAM is designed to support bulk data transfer with a single sender and multiple receivers. It uses dynamic trees to implement local error recovery and to scale to a large number of receivers without impacting the sender. It also includes flow control, congestion control, and other adaptive techniques necessary to operate efficiently and fairly with other protocols across the wide variety of link and client characteristics that make up the Internet as well as intranets.

TRAM ensures reliability by using a selective acknowledgement mechanism, and scalability by

adopting a hierarchical tree-based repair mechanism. The hierarchical tree not only overcomes implosion-related problems but also enables localized multicast repairs and efficiently funnels feedback to the sender.

The receivers and the data source of a multicast session in TRAM interact with each other to dynamically form repair groups. These repair groups are linked together in a hierarchical manner to form a tree with the sender at the root of the tree. Figure 3 shows a typical TRAM repair tree. These types of trees have been shown to be the most scalable way of supporting reliable multicast transmissions [SURVEY]. 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 acknowledgement mechanism similar to TCP's [SACK]. The repair heads cache every message sent by the sender and provide repair service for messages that are reported as lost by the members. The dynamic nature of the tree allows it to react to changes in the underlying network infrastructure without sacrificing reliability.

The TRAM tree formation algorithm works in both bi-directional multicast environments and uni-directional multicast environments (such as satellite links). Repair nodes are elected based on a wide spectrum of criteria, and the tree is continuously optimized based on the receiver population and network topology. The acknowledgement-reporting mechanism is window-based with optimizations to reduce burstiness and processing overhead. The flow control mechanism is rate-based and adapts to network congestion. The sender senses and adjusts to the rate at which the receivers can accept the data. Receivers that cannot keep up with the minimum data rate can be dropped from the repair tree.

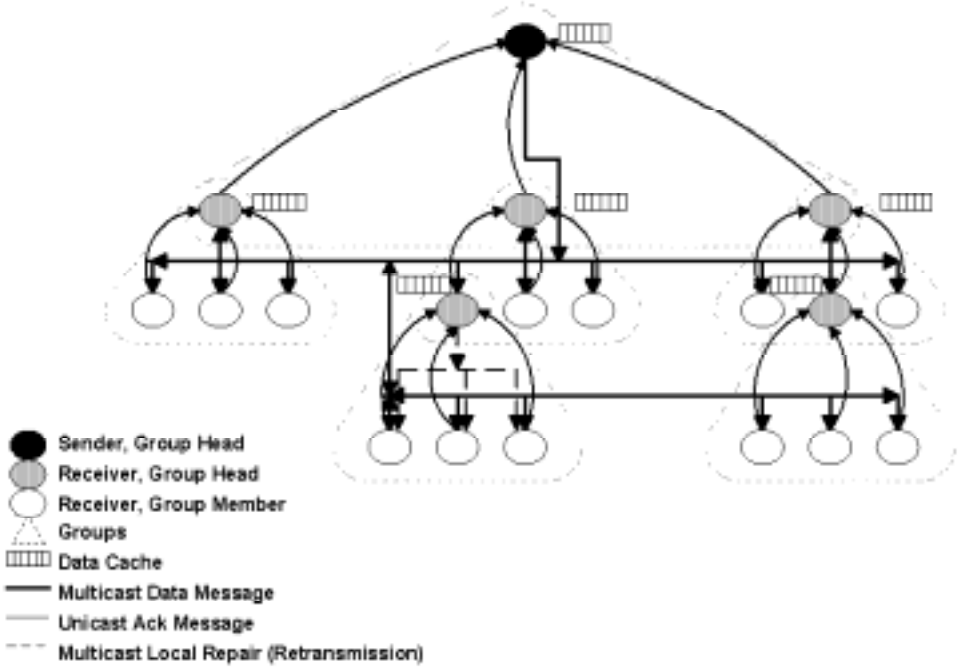

Figure 3: Typical Hierarchical Repair Tree in TRAM

Some of the major features of TRAM are:

Reliability: TRAM guarantees delivery of data to any receiver that joins the tree and is able to keep up with the minimum transmission speed specified by the sender. While this level of guarantee cannot protect applications from delivery failure, features of the JRM Service can be used to closely keep track of individual member's status.

Avoiding Acknowledgement Implosion: Point-to-point transports achieve reliability by acknowledging receipt of data directly to the sender. Unfortunately, this does not scale over multicast. A sender trying to field acknowledgements from many receivers will quickly become overloaded. TRAM avoids this implosion by dynamically designating a receiver to be a repair head for a group of members. The repair head fields acknowledgements from each of its group members and supplies them with repair packets. To avoid overload, each repair head is responsible for only a limited number of members.

Local Repair: TRAM builds the tree so that repair heads are close to their members. This enables repair heads to perform repairs using small time-to-live (TTL) values, which not only minimizes network bandwidth consumption but also avoids unnecessary processing at receivers not requiring repair.

Automatic Configuration: Different applications, networks, or user groups may perform better using different types of trees. TRAM provides for building different types of trees optimized for these different circumstances without adding complexity to the network infrastructure itself. As a receiver group changes, TRAM also provides for ongoing optimization of an existing tree, for instance, enabling a member to find a better repair head. A particular tree's formation method may also be changed at any time during the transmission at the discretion of the sender. TRAM's tree management works in networks providing only unidirectional multicast, as well as those supporting bi-directional multicast.

Rate-based Flow Control and Congestion Avoidance: TRAM schedules packet transmission according to a data rate. This data rate is dynamically adjusted based on congestion feedback from the receivers. Congestion feedback is aggregated by repair heads through the tree. The algorithm used to adjust the rate works in different network topologies. The rate is bounded by a maximum and minimum rate configured at the sender.

Feedback to the Sender: Each member of the tree periodically reports statistics to its repair head. This includes statistics that assist in building the tree (for instance, the number of available repair heads on the tree) as well as reports on congestion conditions, which allow the sender to adapt its data rate to network conditions. This information is aggregated at each level of the tree in order to reduce control traffic to the sender.

Controlling Memory Use: Each repair head is responsible for ensuring that the data is received by every member. This means that a repair head must cache data until it is sure that all of its members have received it. TRAM requires positive acknowledgements from members when data is received. This enables repair heads to reclaim cache buffers containing data that has been received by all members.

Repair Group Management: Both members and repair heads monitor each other to detect unreachability. Non-responsive members can be dropped from the repair group and corresponding cache buffers can be reclaimed. Non-responsive repair heads can be abandoned by their members in favor of an active repair head. Repair heads are also responsible for detecting receivers which cannot keep up with the minimum transmission rate specified by the sender. While such members cannot be dropped from the multicast group, they can be denied repair head support and receive no repairs.

Scalability: TRAM has been designed to be scalable in many situations, such as large numbers of receivers and sparsely or densely populated receiver groups. TRAM also accommodates wide ranges of receiver capabilities. Control message traffic is designed to be limited in all of these cases.

4.2 Data Transmission

The TRAM sender transmits data at a rate that adjusts automatically between a specified minimum and maximum. Sequence numbers in the data packets allow receivers to identify

missing packets. Each member is bound to a repair head that retransmits lost packets when requested. Acknowledgements sent by receivers contain a bitmap indicating received and missing packets. Missing packets are repaired by the repair head; packets received by all members are removed from the repair head's cache.

4.3 Limitations

Realistic TTL Values: Tree formation in TRAM relies on TTLs. Current multicast routing protocols may not provide accurate TTL values for this purpose.

Heterogeneous Receiver Population: Although TRAM adjusts itself to the capabilities of the receivers as much as possible, a particular slow receiver may end up without repair service if it cannot keep up with the minimum speed specified by the sender according to administrative policy.

5 Channel Management

The JRM Service provides an abstraction layer known as a "channel" for applications to deliver data to multiple receivers. In the protocol stack, it is layered above the transport. This allows channel to provide a common set of functions to applications irrespective of which transport protocols are in use. For example, one of channel's functions is authentication of receivers. This would be available to collaborative applications using a many-to-many transport protocol, such as LRMP, and to news distribution applications using a one-to-many transport protocol, such as TRAM. The choice of transport protocol can differ without affecting the functions provided by the channel layer.

Channels provide the following functions:

- Channel discovery
- Channel registration and authentication of channel users
- Event notification
- Application of policy through dynamic filtering
- Access to security services

A key motivation behind providing the channel layer comes from the requirements of applications moving from unicast to multicast. In unicast, there is a point-to-point connection between a single pair of nodes in a network. If a banking application distributes updated currency information to each of its remote offices over a network at the end of each day, it can make a unicast connection to each office in turn, transfer the data and log the fact that it did so. In multicast, senders and receivers can be anonymous to each other, given the way nodes join and leave multicast groups. If one were to improve the banking application by moving it to multicast, how would it be able to log that it transferred data to each of the remote offices? One way would be for each of the remote offices to tell the sender by establishing a unicast connection with the sender when transmission is complete. This, of course, would somewhat defeat the purpose of using multicast. A better way would be to provide the ability for receivers to be identified and remembered as part of the initialization process of the session so that the sender would know each of the receivers and be able to log the information. This somewhat recreates, for reliable multicast, the implicit knowledge that a pair of nodes has of each other's identities in a unicast environment. Channel has discovery and registration functions to provide this capability for applications.

5.1 Channel Discovery, Registration and Authentication

A useful way to think about the concept of a channel is to visualize it as a shared medium to which all users of the channel are attached. Data sent on a channel is visible to all that have subscribed to it. Users (applications running on network nodes) can be senders, receivers or both. The channel itself is an entity that has associated characteristics. Applications discover the existence of channels, then find out the characteristics of the channels in order to use them. An application can create a channel, use it for the duration of the application session and then destroy it. Channels can also persist indefinitely and be reused by more than one application.

5.1.1 Discovery

Potential users of a channel can discover it in several ways: by listening for session announcements, by querying a channel manager or by static configuration. Using a well-known multicast address, sessions can be advertised via the Session Announcement Protocol [SAP].

Session announcements contain information necessary for an application to become a user of the channel. This information is packaged into a structure called a *channel profile*. Since applications in a multicast environment are distributed, this mechanism provides for synchronization of users. This is often useful for bulk data transfer applications that require all receivers to be ready and waiting prior to starting data transmission. Among other things, channel advertisements contain a channel name and a multicast address and port for the session. They may also contain session start time information, security information, and other optional fields. The JRM Service uses the Session Description Protocol [SDP] to format session advertisement messages. The use of SAP and SDP allows leveraging existing mechanisms used on the MBone for announcing multicast sessions.

Another way to discover channels is for a potential channel user to contact a channel manager. Channel managers are processes in the network that provide functions for maintaining channels. Session details are cached by channel managers and are subsequently available to requestors once properly authenticated. A potential user of a channel could query a channel manager for information about a channel whose channel name is known. The channel manager then provides the information necessary for the application to attach to and become a user of the channel.

Finally, channel information can be statically configured within applications themselves. This would alleviate the need for SAP announcements or channel manager queries but it is the least flexible. However, it may be appropriate for certain applications.

5.1.2 Registration and Authentication of Users

Once a potential user of a channel discovers the channel and its information, the next step is registration. To do this, it makes a registration call to a channel manager. The channel manager performs authentication to be sure the user has appropriate permissions to access the channel, records the event that this user has attached and, if configured, downloads any dynamic filters to the user application which may be necessary for administrative control over the channel data itself (see section 5.3). The channel manager may also send security information, such as encryption and authentication keys, to the registered receiver (see section 7). Registration may also trigger events for which others can be waiting. For instance, the registration mechanism allows senders to wait for all receivers, or certain receivers, to register before transmitting data.

During the session, it may be required that receivers re-register. Re-registration causes users to contact their channel manager using the same registration calls performed the first time. This provides a way to apply new or changed configuration data during a session. For example, one way to change data encryption keys during a session is to require each receiver to periodically re-register and securely download new keys.

5.2 Event Notification

Channel provides interfaces that may be used to inform applications of various events. For instance, events are available to enable an application to monitor a particular user; the application can be informed when the user registers for a session, or when it completes reception of some data. Events can also be used to monitor the session as a whole, for instance, keeping track of how many users are registered at any particular time.

5.3 Dynamic Filters

Dynamic filters are designed to provide the flexibility to apply administrative control over the content of a channel. One or more dynamic filters may be installed on a receiver during the process of registering for a channel. Dynamic filters are pre-configured Java software classes which are associated with one or more receivers. Typically, they are downloaded to receivers from a service running on a server in the network. This service can be co-located with the channel manager or not but must be accessible by the channel manager for retrieving dynamic filters at registration time. The JRM Service stores dynamic filters in a database. Caching dynamic filters or otherwise making them persistent at the receiver is a possible enhancement.

Once dynamic filters are installed, they have access to data packets received from the network but not yet presented to the application. In the current implementation, dynamic filters are called on a per-packet basis which allow them to filter, transform or augment the data and return the result. Multiple dynamic filters may be

installed at a receiver in which case each is called in turn to process the received data packets. When the last dynamic filter completes, the data is presented to the application.

Some dynamic filter operations may require caching packets until a logical block of data is present in receiver memory. If, for example, a dynamic filter is performing language translation, it may require more than one packet at a time. Other examples may include encryption/decryption operations, forward error correction or distributed guaranteed delivery algorithms for time-sensitive data.

Application designers may choose to use dynamic filters as a way for end-users to customize preferences for their data. Additionally, network management personnel may apply centralized control for groups of end-users using dynamic filters to limit access to certain content based on local policies. The dynamic filtering feature is designed to be a flexible mechanism for accommodating many diverse requirements of multicast-aware applications.

6 Address Allocation

Allocating and managing multicast IP addresses is a tricky problem. Unlike a unicast IP address, data sent to a multicast address may be sent to receivers on many different networks. In fact, the set of receivers may change over time. If two senders use the same multicast address accidentally, the data they send may be mingled, causing application confusion. Therefore, it is important to manage multicast addresses carefully so that this doesn't happen.

Several multicast address allocation techniques are currently in use. Administratively scoped multicast addresses are often allocated with manual systems such as a list maintained by a network administrator. Non-administratively scoped addresses are allocated via multicast advertising with the Session Announcement Protocol (SAP). This allows for global dynamic allocation, but it will only scale to about 10,000 addresses [HANDLEY].

The Multicast Address Allocation Working Group of the IETF is working on solutions that will scale to millions of addresses. Their current architecture calls for a three-tiered model with interdomain allocation handled by MASC [MASC], intradomain allocation handled by AAP [AAP], and a host request protocol named MDHCP [MDHCP]. The JRM Service team is playing an active role in development of the new IETF protocols, and implementations of these protocols are being considered.

The JRM Service addresses the multicast address allocation problem in three ways. First, it provides a standard API that may be used to request and manage addresses. This insulates application vendors from any changes in address allocation techniques. Second, it provides support for allocating addresses from a predefined pool of administratively scoped addresses, set aside exclusively for the use of a single JRM Service channel manager. Third, it provides an implementation of SAP allocation.

The multicast address allocation APIs provided by the JRM Service are the Multicast Address Management API (MAMAPI), which applications use to allocate multicast addresses, and the Multicast Address Management Service Provider Interface (MAMSPI), which allows any multicast address allocator to be plugged in.

7 Security

The JRM Service security interfaces support both data confidentiality and data integrity. These interfaces expand on existing unicast security methods, enabling their use with multicast. Keys are managed by security servers (see section 7.5), which create, distribute, and update keys used by the channel. Senders and receivers use an encryption layer above the transport for confidentiality, and authentication services at the transport layer for message and sender authentication. Although the JRM Service provides a number of security methods, not all may be available for export. Figure 4 shows an overview of JRM Service security.

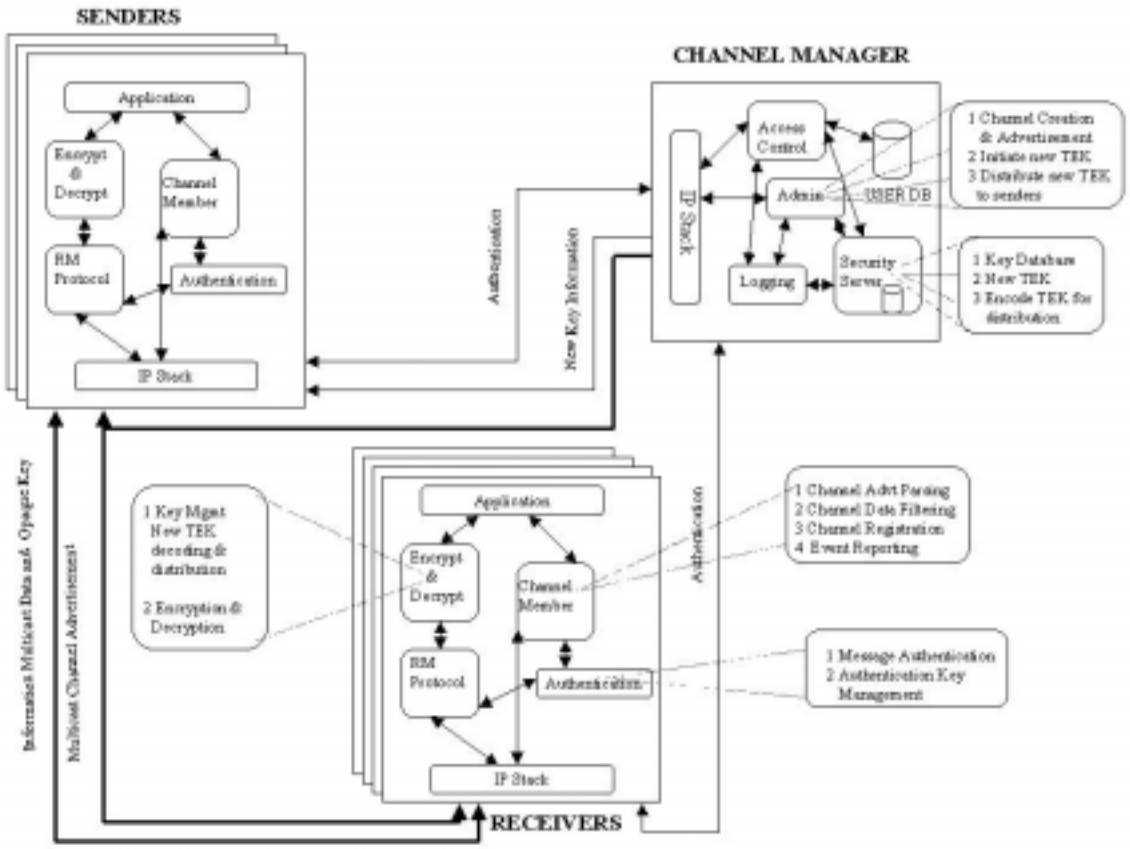

Figure 4: JRM Service Security Architecture

7.1 Data Confidentiality

Data confidentiality ensures that channel data can only be viewed by subscribers of the channel. At channel registration time, each registered sender or receiver is given the current text encryption keys. These are usually secret keys shared by the entire multicast group. Senders use these keys to encrypt data, and receivers use them to decrypt data.

7.2 Data Integrity

Data integrity guarantees the receiver that a message was not modified in transit. To do this, the sender computes a digest of the message, encrypts the computed digest and includes the encrypted value in the message that is to be sent. Receivers compute the digest of the message and compare the computed digest against the decrypted digest extracted from the received message. Messages that fail the comparison test are considered altered in transit and are discarded. At channel registration time, each registered receiver is given keys to verify digests from each authorized sender. The channel may require a separate key for each sender, or it may allow all of the senders to use the same key.

These keys can be the public key of each sender, or a new public key generated for the channel's use. Senders use their private key to generate a signature of the data, and receivers use the corresponding public key to verify the signature.

Alternatively, shared secret keys may be used, either a separate one for each sender, or one key used by all senders. In this case, the same key is used to both encrypt and decrypt. However, public/private key pairs are often used, since, as described in the next section, they also provide sender authentication.

7.3 Sender Authentication

Sender authentication guarantees the receiver that a message came from the indicated sender. Usually a public/private key pair is used. At channel registration time, each registered receiver is given the public key of each authorized sender. The channel may require a separate key pair for each sender, or it may allow all of the senders to use the same key pair. These keys can be the public key of each sender, or a new public key generated for the channel's use. Senders use the private key to generate a signature of the data, and receivers use the corresponding public key to verify the signature. In practice, such signature generation can take a considerable amount of processing, so, instead of signing the entire message, the sender first generates a digest of the message (a much quicker operation), then encrypts the digest. Receivers decrypt the digest, then validate the digest by regenerating it from the data.

7.4 Rekeying

The authentication and encryption mechanisms described above are very similar to those in use for unicast connections. The main issue for these mechanisms in a multicast environment is rekeying. If a unicast session suspects that a key has been compromised, it is not difficult for the two parties to confidentially generate a new key. In a multicast environment, it is not so easy. In addition, a multicast application may have several additional reasons to redistribute keying information. For instance, the application may be paid for on a daily basis, requiring that any encryption keys be changed on a daily basis. Or, an application may require that a new member not be able to see any data sent before it joined. In this case, since the new member may request that old data be repaired, the encryption key must be changed any time a new member joins the group.

If the group has a small number of members, each one could be made to re-register and then receive new keying information, as if it had just joined the group. However, this may not scale well enough to larger groups, and could take some time. Another option is to use one of several innovative proposed algorithms [KEYING] that allow key managers to multicast information to the receivers that allows them to recalculate new keys. This information is cleverly encrypted to prevent excluded users from decrypting it. JRM Service applications using these types of algorithms can use the channel to reliably send out the rekeying information.

Other algorithms may be used for data confidentiality and integrity; the JRM Service interfaces make the actual method used transparent to the JRM Service code itself. The implementor need only conform to general interfaces for encryption/decryption, signature generation/verification, and key management.

7.5 Security Servers

Security servers are used by channel managers to produce and distribute keys. Standard Java APIs accommodate implementations supporting most security strategies. Security servers within channel managers support distribution of keys to new users, as well as rekeying.

7.6 Authentication Services

The JRM Service provides authentication services solely within the transport layer. For reliable multicast, it is essential that authentication be done in the transport itself to avoid feeding invalid packets to the upper layers. By integrating authentication into the transport, the transport's reliability features can be used to recover the legitimate version of an altered packet. If an altered packet is accepted and passed to an upper layer for authentication, there is no possibility of replacing it with a legitimate one.

8 Conclusions

This paper describes the architecture and major features of a reliable multicast system designed to support the needs of new types of information delivery applications. We believe that a multicast-based system is inherently better than an HTTP-based system for pushing event-based data over a computer network. The results show better scalability through more efficient bandwidth utilization of the network and timely delivery of the data. When reliability is applied, this technology enables dissemination of mission-critical business data that expands the use of multicast beyond audio and video applications.

The JRM Service can be thought of as interesting "middleware". Since the JRM Service approaches the problem of adding reliability to multicast as an end-to-end problem, it layers software between the IP stack of the end-system and the application. The design center of the JRM Service is to isolate the application as much as possible from the details of the network through a cleanly layered set of APIs. Alternative approaches require that applications be intertwined with the network software. This makes them more difficult to write, given all of the considerations about the network that must be taken into account. However, such intertwined solutions have the potential to be more intelligent about such things as repairing lost data in collaborative applications, for example.

The JRM Service uses IP multicast. This raises two major problems. The first concerns the multicast routing protocols themselves: how well they work and how well they scale to large networks. The second is the availability of multicast. Many corporate network administrators have yet to enable it in their network routers, and many ISPs are evaluating the business case to offer multicast service and the costs of doing so. These issues will be resolved over time and multicast will be ubiquitous on corporate networks in the next couple of years and in widespread use on the Internet someday. Ubiquity on the Internet is unclear — perhaps within the realms of ISPs is most realistic.

One of the key objectives of the JRM Service was to develop a reliable multicast protocol [TRAM] that enables large-scale distribution of data. Through design, implementation, analysis and experimentation, we have learned how hard this problem is. In real networks there is always loss occurring somewhere in the network. Throughput bandwidth is variable when there is some level of network load. This required consideration of complex algorithms for congestion detection, avoidance and control as well as protocol overhead to construct, maintain and optimize the repair tree. As the number of receivers grows, the protocol overhead must be minimized so as not to consume any significant bandwidth on the network. A mechanism to deal with slow receivers was needed to allow disconnection of receivers from the repair tree which were too slow to keep up, thereby causing the whole transmission to all receivers to lock-step with the slowest receiver. This mechanism enables applications to decide how best to tradeoff completion of the overall transmission and error recovery. In addition, the TRAM protocol provides for transmission between a minimum and maximum speed. This is somewhat novel because a majority of IP network applications have been traditionally built to use as much bandwidth as they need and the network will provide without any regard for other applications or network traffic. Since TCP/IP networks are "best effort delivery," applications such as ftp use as much bandwidth as the network will provide. The JRM Service goes one step further and tries to hold itself between two bounds, perhaps behaving as a better network citizen but also attempting to give streaming applications a minimum quality of service. Other applications may be built similarly in the future.

9 Future Work

9.1 TRAM

Support for Live and Resilient Data: With a few adjustments, TRAM could support live data transmissions. The primary difference between bulk data and live data is the requirement to view live data in real time. Bulk data can afford to delay repair of a single packet; live data cannot..

Full Data Recovery for Late Joins: Currently TRAM has no guaranteed support for late joiners requiring full data recovery. If a member comes late to a session, it may not find a repair head that has all of the existing session data in its cache.

Congestion Control: Once TRAM is more fully deployed and there are more results concerning the performance of TRAM's congestion control, we expect to make more improvements. In particular, interaction with other network traffic, for instance, TCP, needs further study.

Repair Tree Management: Some situations call for more controlled tree formation. Some possibilities include

- Static assignments of members to repair heads
- Static assignment of standby repair heads

9.2 Channel Management

Caching Dynamic Filters: The current design calls for dynamic filters to be automatically updated by requiring receivers to re-register from time to time. However, it is likely that dynamic filters will not change very often. Caching dynamic filters or otherwise making them persistent at the receiver is a possible enhancement.

9.3 Address Allocation

Emerging Protocols: Address Allocation Protocol [AAP] and Multicast Dynamic Host Configuration Protocol [MDHCP] are currently Internet Drafts. As they move along the standards process, they may replace or supplement The JRM Service's current address allocation techniques.

9.4 SAP Support

JRMS leverages SAP as a mechanism to advertise the existence of multicast sessions. Alternatives or enhancements to SAP must be examined due to the inherent limitations in its ability to scale well once the session count becomes extremely large.

10 Acknowledgements

The authors wish to thank Dah Ming Chiu, Steve Hurst, Radia Perlman, Seth Proctor and Joe Wesley for their contributions to the content of this paper.

11 References

[AAP] MALLOC Working Group, M. Handley, *Multicast Address Allocation Protocol*, draft-ietf-malloc-aap-01.txt, work in progress, July 1998

[ADMIN] D. Meyer, *Administratively Scoped IP Multicast*, RFC 2365, July 1998

[AIM] B. Levine, J. Garcia-Luna-Aceves, *Improving Internet Multicast with Routing Labels*, University of California, Santa Cruz, 1997

[BGMP] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, M. Handley, *The MASC/BGMP Architecture for Inter-domain Multicast Routing*, SIGGCOMM '98

[CBT] T. Ballardie, *Core Based Trees (CBT) Multicast Routing Architecture*, RFC 2201, September 1997

[DVMRP] T. Pusateri, *Distance Vector Multicast Routing Protocol*, draft-ietf-idmr-dvmrp-v3-06, work in progress, March 1998

[HANDLEY] M. Handley, *On Scalable Multimedia Conferencing Systems*, University of London, November, 1997

[IGMP] W. Fenner, *Internet Group Management Protocol, Version 2*, RFC 2236, November 1997

[IP] D. Comer, *Internetworking with TCP/IP, Volume I*, Prentice-Hall, ISBN 0-13-468505-9, 1991

[KEYING] G. Caronni, M. Waldvogel, D. Sun, B. Plattner, *Efficient Security for Large and Dynamic Multicast Groups*, Proceedings of the Seventh Workshop on Enabling Technologies (WET ICE '98), IEEE Computer Society Press, 1998

[LMS] C. Papadopoulos, G. Parulkar, G. Varghese, *An Error Control Scheme for Large-Scale Multicast Applications*, Washington University St. Louis, 1997

[LRMP] T. Liao, *Light-weight Reliable Multicast Protocol as an Extension to RTP*, http://monet.inria.fr/lrmp/lrmp-rtp.html

[MASC] D. Estrin, R. Govindan, M. Handley, S. Kumar, P. Radoslavov, D. Thaler, *The Multicast Address-Set Claim (MASC) Protocol*, draft-ietf-idmr-masc-01, work in progress, August, 1998

[MBONE] *The MBone Information Web*, http://www.mbone.com

[MDHCP] S. Hanna, B. Patel, M. Shah, *Multicast Address Allocation based on the Dynamic Host Configuration Protocol*, draft-ietf-malloc-mdhcp-00.txt, work in progress, August, 1998

[MOSPF] J. Moy, *Multicast Extensions to OSPF*, RFC 1584, March 1994

[MULTICAST] S. Deering, *Host Extensions for IP Multicasting*, RFC 1112, August 1989

[PIM-DM] D. Estrin, V. Jacobson, D. Farinacci, D. Meyer, L. Wei, S. Deering, A. Helmy, *Protocol Independent Multicast Version 2 Dense Mode*, draft-ietf-pim-v2-dm-00, work in progress, August 1998

[PIM-SM] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, *Protocol Independent Multicast - Sparse Mode (PIM-SM); Protocol Specification*, RFC 2117, June 1997

[RANDOM] A. Costello, *Search Party: An Approach to Reliable Multicast with Local Recovery*, University of California, Berkeley, 1997

[RTNG] T. Maufer, C. Semeria, *Introduction to IP Multicast Routing*, draft-ietf-mboned-intro-multicast-03.txt, work in progress, July 1997

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

[SAP] M. Handley, *SAP: Session Announcement Protocol*, draft-ietf-mmusic-sap-03.txt, work in progress, March 1997

[SCALE] M. Handley, *Session Directories and Internet Multicast Address Allocation*, Computer Communication Review, volume 28, number 4, October 1998

[SDP] M. Handley, V. Jacobson, *SDP: Session Description Protocol*, RFC 2327, April 1998

[SDR] M. Handley, *The sdr Session Directory: An Mbone Conference Scheduling and Booking System,* http://mice.ed.ac.uk/mice/archive/sdr.html, April 1996

[SECURITY] C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World*, Prentice Hall, 1995

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

[TRAM] D. Chiu, S. Hurst, M. Kadansky, J. Wesley, *TRAM: a Tree-based Reliable Multicast Protocol*, Sun Microsystems Laboratories, SMLI TR-98-66, July 1998

About the Authors

Phil Rosenzweig is the Director of the Boston Center for Networking at Sun Microsystems Laboratories, Chelmsford, MA. He leads an advanced development group that focuses on new network protocols and services. Current projects include developing enabling technology for reliable IP Multicast and a new public key security infrastructure. For the previous six years, he led Sun's PC Networking Group, where he was responsible for the PC-NFS product family that provides connectivity between PCs and UNIX systems. Prior to Sun, Rosenzweig was Director of Software at Xyplex, where he helped develop one of the first multi-protocol communications servers. Rosenzweig received a Bachelor's degree in Computer Science from Rochester Institute of Technology in 1977.

Miriam Kadansky is a Senior Staff Engineer at Sun Microsystems Laboratories, Chelmsford, MA. She is the Technical Lead of the JRM Service team. Kadansky has a Bachelor's degree in Mathematics from Harvard College, and worked for several networking vendors before joining Sun. Her current interests include secure reliable multicasting and Java software.

Steve Hanna is a Staff Engineer at Sun Microsystems Laboratories, Chelmsford, MA. He is the Technical Lead of the Internet Security research team and former Technical Lead of the JRM Service team. Before arriving at Sun, Hanna worked at several software firms including ON Technology and Lotus. His current technical interests include Java software, multicast, security, internationalization, and human interface issues. Hanna is the co-chair of the Multicast Address Allocation (malloc) working group of the Internet Engineering Task Force. He has a Bachelor's degree in Computer Science from Harvard College.