SAM Research Group	J.F. Buford
Internet-Draft	Avaya Labs Research
Intended status: Informational	M. Kolberg, Ed.
Expires: January 29, 2012	University of Stirling
	July 28, 2011

Application Layer Multicast Extensions to RELOAD
draft-samrg-sam-baseline-protocol-00

Abstract

We define a RELOAD Usage for Application Layer Multicast as well as extensions to RELOAD message layer to support ALM. The ALM Usage is intended to support a variety of ALM control algorithms in an overlay-independent way. Scribe is defined as an example algorithm.

Status of this Memo

Copyright Notice

Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

The concept of scalable adaptive multicast includes both scaling properties and adaptability properties. Scalability is intended to cover:

*large group size

*large numbers of small groups

*rate of group membership change

*admission control for QoS

*use with network layer QoS mechanisms

*varying degrees of reliability

*trees connect nodes over global internet

Adaptability includes

*use of different control mechanisms for different multicast trees depending on initial application parameters or application class

*changing multicast tree structure depending on changes in application requirements, network conditions, and membership

Application Layer Multicast (ALM) has been demonstrated to be a viable multicast technology where native multicast isn't available. Many ALM designs have been proposed. This ALM Usage focuses on: [I-D.ietf-p2psip-base] has an application extension mechanism in which a new type of application defines a Usage. A RELOAD Usage defines a set of data types and rules for their use. In addition, this document describes additional message types and a new ALM algorithm plugin architectural component.

*ALM implemented in RELOAD-based overlays

*Support for a variety of ALM control algorithms

*Providing a basis for defining a separate hybrid-ALM RELOAD Usage

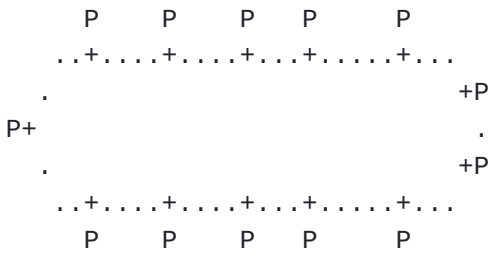
1.1. Requirements Language

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 *[RFC2119]*.

2. Definitions

We adopt the terminology defined in section 2 of [I-D.ietf-p2psip-base], specifically the distinction between Node, Peer, and Client.

2.1. Overlay Network



Overlay network - An application layer virtual or logical network in which end points are addressable and that provides connectivity, routing, and messaging between end points. Overlay networks are frequently used as a substrate for deploying new network services, or for providing a routing topology not available from the underlying physical network. Many peer-to-peer systems are overlay networks that run on top of the Internet. In the above figure, "P" indicates overlay peers, and peers are connected in a logical address space. The links shown in the figure represent predecessor/successor links. Depending on the overlay routing model, additional or different links may be present.

2.2. Overlay Multicast

Overlay Multicast (OM): Hosts participating in a multicast session form an overlay network and utilize unicast connections among pairs of hosts for data dissemination. The hosts in overlay multicast exclusively handle group management, routing, and tree construction, without any support from Internet routers. This is also commonly known as Application Layer Multicast (ALM) or End System Multicast (ESM). We call systems which use proxies connected in an overlay multicast backbone "proxied overlay multicast" or POM.

2.3. Peer

Peer: an autonomous end system that is connected to the physical network and participates in and contributes resources to overlay construction, routing and maintenance. Some peers may also perform additional roles such as connection relays, super nodes, NAT traversal, and data storage.

3. Assumptions

3.1. Overlay

Peers connect in a large-scale overlay, which may be used for a variety of peer-to-peer applications in addition to multicast sessions. Peers may assume additional roles in the overlay beyond participation in the overlay and in multicast trees. We assume a single structured overlay routing algorithm is used. Any of a variety of multi-hop, one-hop, or variable-hop overlay algorithms could be used.
Castro et al. [CASTRO2003]compared multi-hop overlays and found that tree-based construction in a single overlay out-performed using separate overlays for each multicast session. We use a single overlay rather than separate overlays per multicast sessions.
An overlay multicast algorithm may leverage the overlay's mechanism for maintaining overlay state in the face of churn. For example, a peer may store a number of DHT (Distributed Hash Table) entries. When the peer gracefully leaves the overlay, it transfers those entries to the nearest peer. When another peer joins which is closer to some of the entries than the current peer which holds those entries, than those entries are migrated. Overlay churn affects multicast trees as well; remedies include automatic migration of the tree state and automatic re-join operations for dislocated children nodes.

3.2. Overlay Multicast

The overlay supports concurrent multiple multicast trees. The limit on number of concurrent trees depends on peer and network resources and is not an intrinsic property of the overlay.

3.3. RELOAD

We use RELOAD [I-D.ietf-p2psip-base] as the distibuted hash table (DHT) for data storage and overlay by which the peers interconnect and route messages. RELOAD is a generic P2P overlay, and application support is defined by profiles called Usages.

3.4. NAT

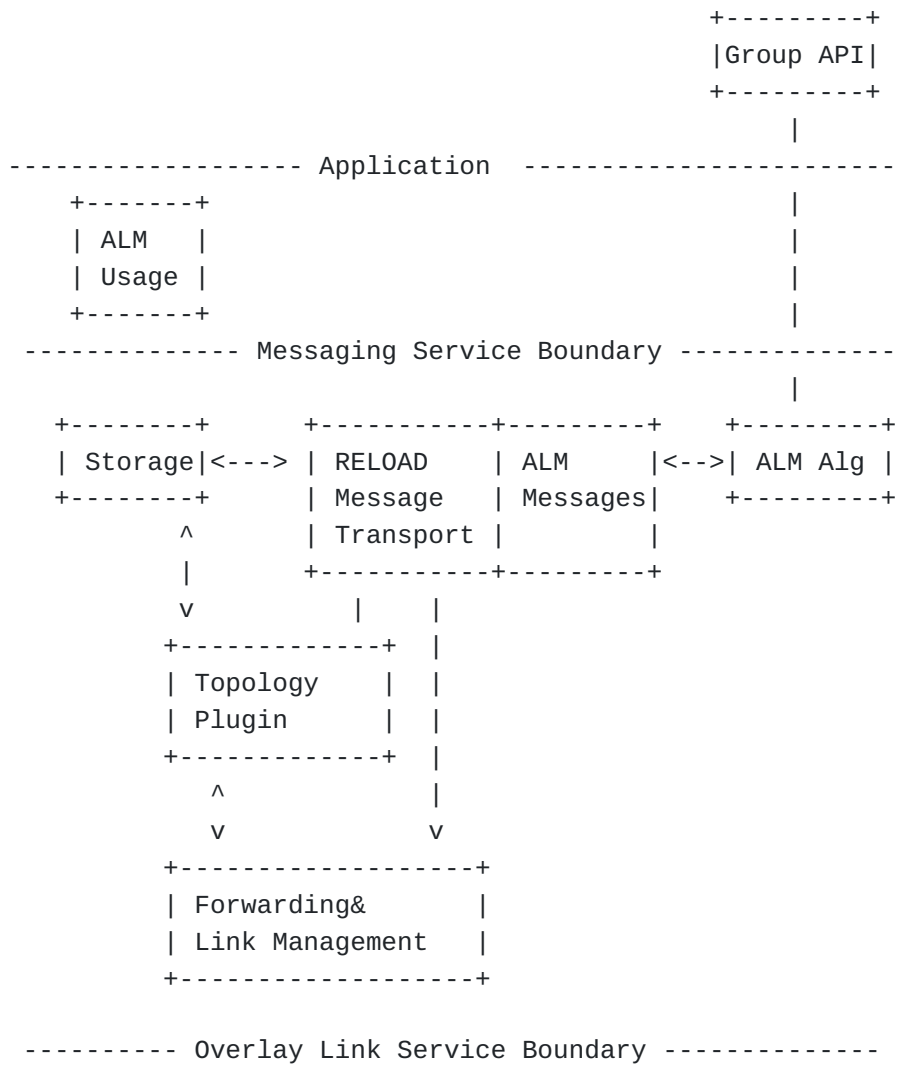
Some nodes in the overlay may be in a private address space and behind firewalls. We use the RELOAD mechanisms for NAT traversal. We permit clients to be leaf nodes in an ALM tree.

3.5. Tree Topology

All tree control messages are routed in the overlay. Two types of data or media topologies are envisioned: 1) tree edges are paths in the overlay, 2) tree edges are direct connections between a parent and child peer in the tree, formed using the RELOAD AppAttach method.

4. Architecture Extensions to RELOAD

There are two changes, shown in the figure below. New ALM messages are added to RELOAD Message Transport. A plug-in for ALM algorithms handles the ALM state and control. The ALM Algorithm is under control of the application via the Group API [I-D.irtf-samrg-common-api].



The ALM components interact with RELOAD as follows:

- ALM uses the RELOAD data storage functionality to store a ALMTree instance when a new ALM tree is created in the overlay, and to retrieve ALMTree instance(s) for existing ALM trees.

- ALM applications and management tools may use the RELOAD data storage functionality to store diagnostic information about the operation of tree, including average number of tree, delay from source to leaf nodes, bandwidth use, lost packet rate. In addition, diagnostic information may include statistics specific to the tree root, or to any node in the tree.

5. RELOAD ALM Usage

Applications of RELOAD are restricted in the data types that be can stored in the DHT. The profile of accepted data types for an application is referred to as a Usage. RELOAD is designed so that new applications can easily define new Usages. New RELOAD Usages are needed for multicast applications since the data types in base RELOAD and existing usages are not sufficient.
We define an ALM Usage in RELOAD. This ALM Usage is sufficient for applications which require ALM functionality in the overlay. The figure below shows the internal structure of the ALM Usage. This contains the Group API ([I-D.irtf-samrg-common-api]) an ALM algorithm plugin (e.g. Scribe) and the ALM messages which are then sent out to the RELOAD network.
A RELOAD Usage is required [I-D.ietf-p2psip-base] to define the following:

- Register Kind-Id points

- Define data structures for each kind

- Defines access control rules for each kind

- Defines the Resource Name used to hash to the Resource ID where the kind is stored

- Addresses restoration of values after recovery from a network partition

- Defines the types of connections that can be initiated using AppConnect

A ALM GroupID is a RELOAD Node-ID. The owner of a ALM group creates a RELOAD Node-ID as specified in [I-D.ietf-p2psip-base]. This means that a GroupID is used as a RELOAD Destination for overlay routing purposes.

6. ALM Tree Control Signaling

Peers use the overlay to support ALM operations such as:

- *Create tree
- *Join
- *Leave
- *Re-Form or optimize tree

There are a variety of algorithms for peers to form multicast trees in the overlay. We permit multiple such algorithms to be supported in the overlay, since different algorithms may be more suitable for certain application requirements, and since we wish to support experimentation. Therefore, overlay messaging corresponding to the set of overlay multicast operations must carry algorithm identification information. For example, for small groups, the join point might be directly assigned by the rendezvous point, while for large trees the join request might be propagated down the tree with candidate parents forwarding their position directly to the new node.
Here is a simplistic algorithm for forming a multicast tree in the overlay. Its main advantage is use of the overlay routing mechanism for routing both control and data messages. The group creator doesn't have to be the root of the tree or even in the tree. It doesn't consider per node load, admission control, or alternative paths.

```
groupID = create();  // allocate a unique groupId
                     // the root is the nearest
                     // peer in the overlay
                     // out of band advertisement or
                     // distribution of groupID,
                     // perhaps by publishing in DHT

// out of band discovery of groupID, perhaps by lookup in DHT
joinTree(groupID); // sends "join groupID" message
```

As stated earlier, multiple algorithms will co-exist in the overlay.

1. Peer which initiates multicast group:

2. Any joining peer:

   The overlay routes the join request using the overlay routing mechanism toward the peer with the nearest id to the groupID.

This peer is the root. Peers on the path to the root join the tree as forwarding points.

3. Leave Tree:

   leaveTree(groupID) // removes this node from the tree

   Propagates a leave message to each child node and to the parent node. If the parent node is a forwarding node and this is its last child, then it propagates a leave message to its parent. A child node receiving a leave message from a parent sends a join message to the groupID.

4. Message forwarding:

   multicastMsg(groupID, msg);

5. For the message forwarding there are two approaches:

   *SSM tree: The creator of the tree is the source. It sends data messages to the tree root which are forwarded down the tree.

   *ASM tree: A node sending a data message sends the message to its parent and its children. Each node receiving a data message from one edge forwards it to remaining tree edges it is connected to.

7. ALM Messages Added to RELOAD Protocol

7.1. Introduction

In this document we define messages for overlay multicast tree creation, using an existing proposal (RELOAD) in the P2P-SIP WG [I-D.ietf-p2psip-base] for a universal structured peer-to-peer overlay protocol. RELOAD provides the mechanism to support a number of overlay topologies. Hence the overlay multicast framework [I-D.irtf-sam-hybrid-overlay-framework] (hereafter SAM framework) can be used with P2P-SIP, and that the SAM framework is overlay agnostic.
As discussed in the SAM requirements draft, there are a variety of ALM tree formation and tree maintenance algorithms. The intent of this specification is to be algorithm agnostic, similar to how RELOAD is overlay algorithm agnostic. We assume that all control messages are propagated using overlay routed messages.

7.2. Tree Lifecycle Messages

Peers use the overlay to transmit ALM (application layer multicast) operations defined in this section.

7.2.1. Create Tree

A new ALM tree is created in the overlay with the identity specified by GroupId. The usual interpretation of GroupId is that the peer with peer id closest to and less than the GroupId is the root of the tree. The tree has no children at the time it is created.
The GroupId is generated from a well-known session key to be used by other Peers to address the multicast tree in the overlay. The generation of the GroupId from the SessionKey MUST be done using the overlay's id generation mechanism.
A successful Create Tree causes an ALMTree structure to be stored in the overlay at the node responsible for NodeID equal to the GroupId.

```
struct {
  NodeID PeerId;
  opaque SessionKey<0..2^32-1>;
  NodeID GroupId;
  Dictionary Options;
} ALMTree;
```

PeerId: the overlay address of the peer that creates the multicast tree.
SessionKey: a well-known string when hashed using the overlay's id generation algorithm produces the GroupId.
GroupId: the overlay address of the root of the tree
Options: name-value list of properties to be associated with the tree, such as the maximum size of the tree, restrictions on peers joining the tree, latency constraints, preference for distributed or centralized tree formation and maintenance, heartbeat interval.
Tree creation is subject to access control since it involves an Store operation. Before the Store of an ALMTree structure is permitted, the storing peer MUST check that:

*The certificate contains a SessionKey

*The certificate contains a Node-ID that is the same as GroupID that it is being stored at Node-ID (this is the NODE-MATCH access policy)

7.2.2. Join

Causes the distributed algorithm for peer join of a specific ALM group to be invoked. If successful, the PeerId is notified of one or more candidate parent peers in one or more JoinAccept messages. The particular ALM join algorithm is not specified in this protocol.

```
struct {
  NodeID PeerId;
  NodeID GroupId;
  Dictionary Options;
} Join;
```

PeerId: overlay address of joining/leaving peer
GroupId: the overlay address of the root of the tree
Options: name-value list of options proposed by joining peer

7.2.3. Join Accept

Tells the requesting joining peer that the indicated peer is available to act as its parent in the ALM tree specified by GroupId, with the corresponding Options specified. A peer MAY receive more than one JoinAccept from diffent candidate parent peers in the GroupId tree. The peer accepts a peer as parent using a JoinConfirm message. A JoinAccept which receives neither a JoinConfirm or JoinDecline response MUST expire.

```
struct {
  NodeID ParentPeerId;
  NodeID ChildPeerId;
  NodeID GroupId;
  Dictionary Options;
} JoinAccept;
```

ParentPeerId: overlay address of a peer which accepts the joining peer
ChildPeerId: overlay address of joining peer
GroupId: the overlay address of the root of the tree
Options: name-value list of options accepted by parent peer

7.2.4. Join Confirm

A peer receiving a JoinAccept message which it wishes to accept MUST explicitly accept it before the expiration of the JoinAccept using a JoinConfirm message. The joining peer MUST include only those options from the JoinAccept which it also accepts, completing the negotiation of options between the two peers.

```
struct {
  NodeID ChildPeerId;
  NodeID ParentPeerId;
  NodeID GroupId;
  Dictionary Options;
} JoinConfirm;
```

ChildPeerId: overlay address of joining peer which is a child of the parent peer

ParentPeerId: overlay address of the peer which is the parent of the joining peer
GroupId: the overlay address of the root of the tree
Options: name-value list of options accepted by both peers

7.2.5. Join Decline

A peer receiving a JoinAccept message which does not wish to accept it MAY explicitly decline it using a JoinDecline message.

```
struct {
  NodeID PeerId;
  NodeID ParentPeerId;
  NodeID GroupId;
} JoinDecline;
```

PeerId: overlay address of joining peer which declines the JoinAccept
ParentPeerId: overlay address of the peer which issued a JoinAccept to this peer
GroupId: the overlay address of the root of the tree

7.2.6. Leave

A peer which is part of an ALM tree idenfied by GroupId which intends to detach from either a child or parent peer SHOULD send a Leave message to the peer it wishes to detach from. A peer receiving a Leave message from a peer which is neither in its parent or child lists SHOULD ignore the message.

```
struct {
  NodeID PeerId;
  NodeID GroupId;
  Dictionary Options;
} Leave;
```

PeerId: overlay address of leaving peer
GroupId: the overlay address of the root of the tree
Options: name-value list of options

7.2.7. Re-Form or Optimize Tree

This triggers a reorganization of either the entire tree or only a sub-tree. It MAY include hints to specific peers of recommended parent or child peers to reconnect to. A peer receiving this message MAY ignore it, MAY propagate it to other peers in its subtree, and MAY invoke local algorithms for selecting preferred parent and/or child peers.

```
struct {
  NodeID GroupId;
  NodeID PeerId;
  Dictionary Options;
} Reform;
```

GroupId: the overlay address of the root of the tree
PeerId: if omitted, then the tree is reorganized starting from the root, otherwise it is reorganized only at the sub-tree identified by PeerId.
Options: name-value list of options

7.2.8. Heartbeat

A node signals to its adjacent nodes in the tree that it is alive. If a peer does not receive a Heartbeat message within N heartbeat time intervals, it MUST treat this as an explicit Leave message from the unresponsive peer. N is configurable.

```
struct {
  NodeID PeerId1;
  NodeID PeerId2;
  NodeID GroupId;
} Heartbeat;
```

PeerId1: source of heartbeat
PeerId2: destination of heartbeat
GroupId: overlay address of the root of the tree

8. Scribe Algorithm

8.1. Overview

The following table shows a mapping between RELOAD ALM messages (as defined in Section 5 of this draft) and Scribe messages as defined in [CASTRO2002].

Section in Draft	RELOAD ALM Message	Scribe Message
5.2.1	CreateALMTree	Create
5.2.2	Join	Join
5.2.3	JoinAccept	
5.2.4	JoinConfirm	
5.2.5	JoinDecline	
5.2.8	Leave	Leave
5.2.10	Reform	
5.2.11	Heartbeat	
new	Push/Deliver/Send	Multicast
	Note 1	deliver
	Note 1	forward
	Note 1	route
	Note 1	send

Note 1: These Scribe messages are handled by RELOAD messages.
The following sections describe the Scribe algorithm in more detail.

8.2. Create

This message will create a group with GroupId. This message will be delivered to the node whose NodeId is closest to the GroupId. This node becomes the rendevous point and root for the new multicast tree. Groups may have multiple sources of multicast messages.

CREATE : groups.add(msg.GroupId)

GroupId: the overlay address of the root of the tree

8.3. Join

To join a multicast tree a node sends a JOIN request with the GroupId as the key. This message gets routed by the overlay to the rendevous point of the tree. If an intermediate node is already a forwarder for

this tree, it will add the joining node as a child. Otherwise the node will create a child table for the group and adds the joining node. It will then send the JOIN request towards the rendevous point terminating the JOIN message from the child.
To adapt the Scribe algorithm into the ALM Usage proposed here, after a JOIN request is accepted, a JOINAccept message is returned to the joining node.

```
JOIN : if(checkAccept(msg)) {
                    recvJoins.add(msg.source, msgGroupId)
                    SEND(JOINAccept(nodeID, msg.source, msg.GroupId))
                }
```

8.4. Leave

When leaving a multicast group a node will change its local state to indicate that it left the group. If the node has no children in its table it will send a LEAVE request to its parent, which will travel up the multicast tree and will stop at a node which has still children remaining after removing the leaving node.

```
LEAVE : groups[msg.GroupId].children.remove(msg.source)
           if (groups[msg.group].children = 0)
              SEND(msg,groups[msg.GroupId].parent)
```

8.5. JoinConfirm

This message is not part of the Scribe protocol, but required by the basic protocol proposed in this draft. Thus the usage will send this message to conirm a joining node accepting its parent node.

```
JOINConfirm: if(recvJoins.contains(msg.source,msg.GroupId)){
                 if !(groups.contains(msg.GroupId)) {
                   groups.add(msg.GroupId)
                   SEND(msg,msg.GroupId)
                 }
                groups[msg.GroupId].children.add(msg.source)
                         recvJoins.del(msg.source, msgGroupId)
              }
```

8.6. JoinDecline

```
JOINDecline: if(recvJoins.contains(msg.source,msg.GroupId))
                         recvJoins.del(msg.source, msgGroupId)
```

8.7. Multicast

A message to be multicast to a group is sent to the rendevous node from where it is forwarded down the tree. If a node is a member of the tree

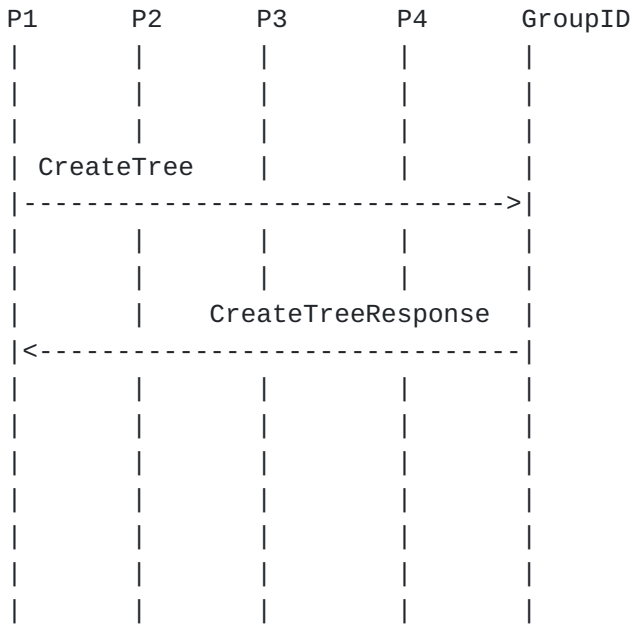
rather than just a forwarder it will pass the multicast data up to the application.

```
MULTICAST : foreach(groups[msg.GroupId].children as NodeId)
                SEND(msg,NodeId)
            if memberOf(msg.GroupId)
                invokeMessageHandler(msg.GroupId, msg)
```

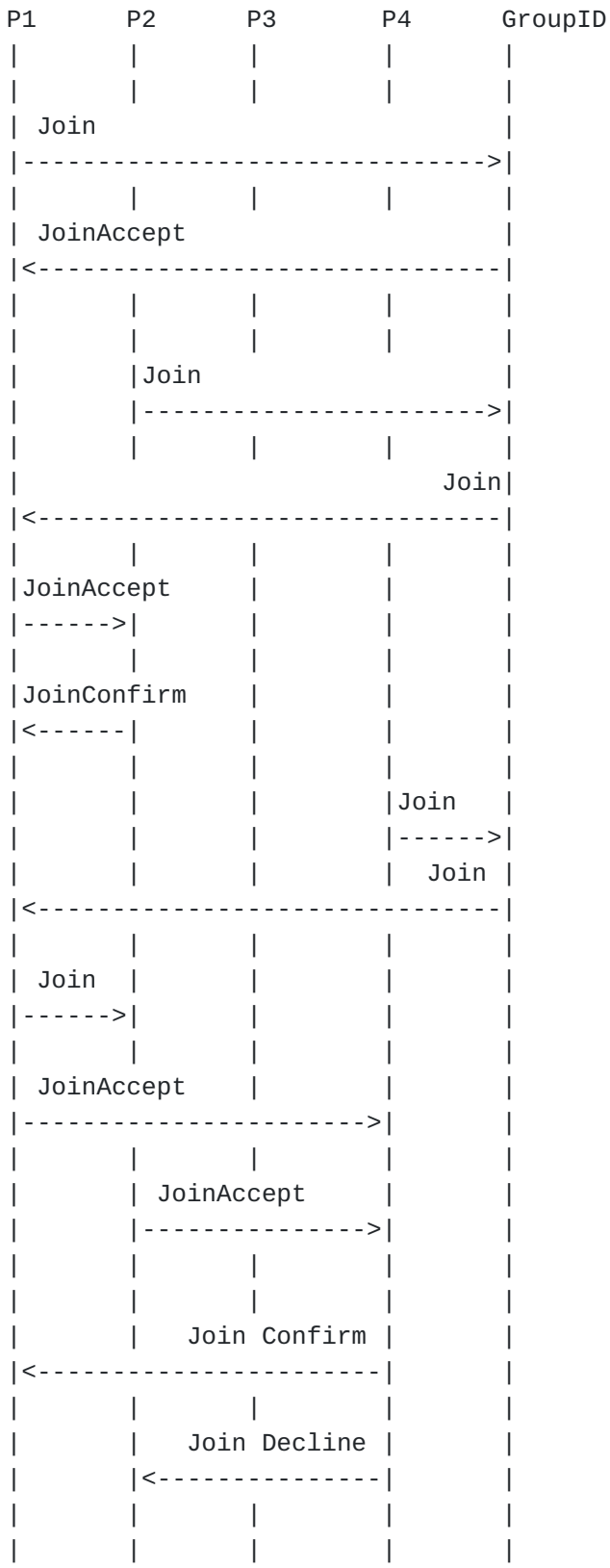
9. Examples

All peers in the examples are assumed to have completed bootstrapping. "Pn" refers to peer N. "GroupID" refers to a peer responsible for storing the ALMTree instance with GroupID.

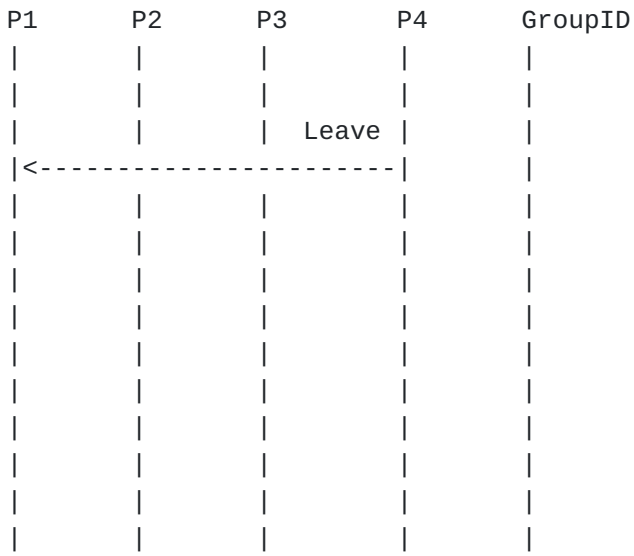
9.1. Create Tree



9.2. Join Tree



9.3. Leave Tree



9.4. Add Direct Application Edge

9.5. Adjust Tree to Churn

9.6. Push Data

10. Kind Definitions

10.1. ALMTree Kind Definition

This section defines the ALMTree kind.
Kind IDs The Resource Name for the ALMTree Kind-ID is the SessionKey used to identify the ALM tree
Data Model The data model is the ALMTree structure.
Access Control NODE-MATCH

11. Configuration File Extensions

In RELOAD, peers receive a configuration document at bootstrap time. ALM parameter definitions for the configuration file will be defined in a later version.

12. Change History

*Version 02: Remove Hybrid ALM material. Define ALMTree kind. Define new RELOAD messages. Define RELOAD architecture extensions. Add Scribe as base algorithm for ALM usage. Define code points. Define preliminary ALM-specific security issues.

13. Open Issues

*The specific capabilities of clients in terms of tree creation and being parents of other nodes will be described in subsequent versions.

*ALM parameter definitions for the RELOAD configuration file will be defined in a later version.

*Should any other ALM algorithms be mapped

14. IANA Considerations

This memo includes no request to IANA.
Code points for the kinds defined in this document MUST not conflict with any defined code points for RELOAD. For Data Kind-IDs, the RELOAD specification states: "Code points in the range 0xf0000001 to 0xfffffffe are reserved for private use". ALM Usage Kind-IDs will be defined in the private use range.
Code points for new message types defined in this document must not conflict with any defined code points for RELOAD. Unlike Data Kind-IDs which permit private code points, RELOAD does not define private or experimental code points for Message Codes. For experimental purposes we recommend using message code points in the range 0x7000 to 0x70FF for the new message types defined in this specification:
All ALM Usage messages support the RELOAD Message Extension mechanism.

Message	Code Point
CreateALMTree	0x7000
CreateALMTreeResponse	0x7001
Join	0x7002
JoinAccept	0x7003
JoinConfirm	0x7004
JoinDecline	0x7005
Leave	0x7006
LeaveResponse	0x7007
Reform	0x7008
ReformResponse	0x7009
Heartbeat	0x700A
Push	0x700B
PushResponse	0x700C

Message Code Points

No new Error Codes are defined.

Application-ID: The ALM Usage Application-IDs must not conflict with other applications of reload. Additionally if AppAttach is used, the port number must be selected to avoid conflicts.
Access Control Policies: No new policies.
ALM Algorithm Types: There is currently one type: SCRIBE-RELOAD.

15. Security Considerations

Overlays are vulnerable to DOS and collusion attacks. We are not solving overlay security issues. We assume the node authentication model as defined in [I-D.ietf-p2psip-base].
ALM Usage specific security issues:

*Right to create GroupID at some NodeId

*Right to store Tree info at some Location in the DHT

*Limit on # messages / sec and bandwidth use

*Right to join an ALM tree

16. References

16.1. Normative References

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC0792]	Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.
[RFC3376]	Cain, B., Deering, S., Kouvelas, I., Fenner, B. and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
[RFC3810]	Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
[RFC4605]	Fenner, B., He, H., Haberman, B. and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
[RFC4607]	Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
[RFC5058]	Boivie, R., Feldman, N., Imai, Y., Livens, W. and D. Ooms, "Explicit Multicast (Xcast) Concepts and Options", RFC 5058, November 2007.

16.2. Informative References

[RFC1930]	Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an

	Autonomous System (AS)", BCP 6, RFC 1930, March 1996.
[RFC3552]	Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
[RFC4286]	Haberman, B. and J. Martin, "Multicast Router Discovery", RFC 4286, December 2005.
[RFC1112]	Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
[I-D.ietf-mboned-auto-multicast]	Thaler, D, Talwar, M, Aggarwal, A, Vicisano, L, Pusateri, T and T Morin, "Automatic IP Multicast Tunneling", Internet-Draft draft-ietf-mboned-auto-multicast-11, July 2011.
[I-D.ietf-p2psip-base]	Jennings, C, Lowekamp, B, Rescorla, E, Baset, S and H Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", Internet-Draft draft-ietf-p2psip-base-19, October 2011.
[I-D.ietf-p2psip-sip]	Jennings, C, Lowekamp, B, Rescorla, E, Baset, S and H Schulzrinne, "A SIP Usage for RELOAD", Internet-Draft draft-ietf-p2psip-sip-06, July 2011.
[I-D.matuszewski-p2psip-security-overview]	Yongchao, S, Matuszewski, M and D York, "P2PSIP Security Overview and Risk Analysis", Internet-Draft draft-matuszewski-p2psip-security-overview-01, October 2009.
[I-D.irtf-p2prg-rtc-security]	Schulzrinne, H, Marocco, E and E Ivov, "Security Issues and Solutions in Peer-to-peer Systems for Realtime Communications", Internet-Draft draft-irtf-p2prg-rtc-security-05, September 2009.
[I-D.irtf-samrg-common-api]	Waehlisch, M, Schmidt, T and S Venaas, "A Common API for Transparent Hybrid Multicast", Internet-Draft draft-irtf-samrg-common-api-03, July 2011.
[I-D.irtf-sam-hybrid-overlay-framework]	Buford, J, "Hybrid Overlay Multicast Framework", Internet-Draft draft-irtf-sam-hybrid-overlay-framework-02, February 2008.
[AGU1984]	Aguilar, L., "Datagram Routing for Internet Multicasting", ACM Sigcomm 84 1984, March 1984.
[CASTRO2002]	Castro, M., Druschel, P., Kermarrec, A.-M. and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure", IEEE Journal on Selected Areas in Communications vol.20, No.8, October 2002.
[CASTRO2003]	Castro, M., Jones, M., Kermarrec, A.-M., Rowstron, A., Theimer, M., Wang, H. and A.

	Wolman, "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays", Proceedings of IEEE INFOCOM 2003, April 2003.
[HE2005]	He, Q. and M. Ammar, "Dynamic Host-Group/ Multi-Destination Routing for Multicast Sessions", J. Telecommunication Systems vol. 28, pp. 409-433, 2005.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

John Buford Buford Avaya Labs Research 233 Mt. Airy Rd Basking Ridge, New Jersey 07920 USA Phone: +1 908 848 5675 EMail: buford@avaya.com

Mario Kolberg editor Kolberg University of Stirling Dept. Computing Science and Mathematics Stirling, FK9 4LA UK Phone: +44 1786 46 7440 EMail: mkolberg@ieee.org URI: http://www.cs.stir.ac.uk/~mko