Support for Multicast over UNI 3.1 based ATM Networks.
<draft-ietf-ipatm-ipmc-05.txt>

Status of this Memo

This document was submitted to the IETF IP over ATM WG. Publication of this document does not imply acceptance by the IP over ATM WG of any ideas expressed within. Comments should be submitted to the ip-atm@matmos.hpl.hp.com mailing list.

Distribution of this memo is unlimited.

This memo is an internet draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress".

Please check the lid-abstracts.txt listing contained in the internet-drafts shadow directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim) to learn the current status of any Internet Draft.

Abstract

Mapping the connectionless IP multicast service over the connection oriented ATM services provided by UNI 3.1 is a non-trivial task. This memo describes a mechanism to support the multicast needs of Layer 3 protocols in general, and describes its application to IP multicasting in particular.

ATM based IP hosts and routers use a Multicast Address Resolution Server (MARS) to support RFC 1112 style Level 2 IP multicast over the ATM Forum’s UNI 3.1 point to multipoint connection service. A single endpoint interface behaviour is described, along with two levels of MARS - Class I and Class II. The Class I MARS service supports layer
3 multicasting using meshes of VCs. The Class II MARS adds the ability to use ATM level multicast servers to support distribution of layer 3 packets.

[Editorial note: This version has been substantially restructured from ipmc-04 in an attempt to group related topics together in a more logical fashion. Additions and modifications to the actual protocol are generally in accordance with the set of proposed changes published and updated during the March to May time period. Section 5.4 is a notable exception to this, and to a lesser extent so is section 5.3. Other tweaks were added as inspiration took me during the rewrite session.]
Contents.

1. Introduction.
   1.1 The Multicast Address Resolution Server (MARS).
   1.2 The ATM level multicast Cluster.
   1.3 Document overview.
2. The IP multicast service model.
3. UNI 3.1 support for intra-cluster multicasting.
   3.1 VC meshes.
   3.2 Multicast Servers.
   3.3 Tradeoffs.
   3.4 Interaction with local UNI 3.1 signalling entity.
4. Overview of the MARS.
5. Endpoint (MARS client) interface behaviour.
   5.1 Transmit side behaviour.
      5.1.1 Retrieving Group Membership from the MARS.
      5.1.2 MARS_REQUEST, MARS_MULTI, and MARS_NAK messages.
      5.1.3 Establishing the outgoing multipoint VC.
      5.1.4 Monitoring updates on ClusterControlVC.
         5.1.4.1 Updating the active VCs.
         5.1.4.2 Tracking the Cluster Sequence Number.
      5.1.5 Revalidating a VC’s leaf nodes.
         5.1.5.1 When leaf node drops itself.
         5.1.5.2 When a jump is detected in the CSN.
   5.2. Receive side behaviour.
      5.2.1 Format of the MARS_JOIN and MARS_LEAVE Messages.
         5.2.1.1 Important IPv4 default values.
      5.2.2 Retransmission of MARS_JOIN and MARS_LEAVE messages.
      5.2.3 Registering with the MARS.
   5.3 Support for Layer 3 group management.
   5.4 Support for redundant/backup MARS entities.
      5.4.1 First response to MARS problems.
      5.4.2 Connecting to a backup MARS.
      5.4.3 Dynamic backup lists, and soft redirects.
   5.5 LLC/SNAP encapsulations for transmit and receive.
6. The MARS in greater detail.
   6.1 Class I MARS requirements.
   6.2 Class II MARS requirements.
      6.2.1 Class II MARS response to a MARS_REQUEST.
      6.2.2 MARS_MSERV and MARS_UNSERV messages.
      6.2.3 Registering a Multicast Server (MCS).
      6.2.4 Class II response to MARS_JOIN and MARS_LEAVE.
      6.2.5 Sequence numbers for ServerControlVC traffic.
   6.3 Why global sequence numbers?
   6.4 Redundant/Backup MARS Architectures.
7. How an MCS utilises a Class II MARS.
8. Support for IP multicast routers.
   8.1 Forwarding into a Cluster.
8.2 Joining in ‘promiscuous’ mode.
8.3 Forwarding across the cluster.
8.4 Joining in ‘semi-promiscuous’ mode.
8.5 An alternative to IGMP Queries.
9. Multiprotocol applications of the MARS and MARS clients.
10. Key Decisions and open issues.
Acknowledgments
Appendix A. Hole punching algorithms for Class II MARS messages.
Appendix B. Minimising the impact of IGMP in IPv4 environments.
Appendix C. Further comments on ‘Clusters’.
1. Introduction.

Multicasting is the process whereby a source host or protocol entity sends a packet to multiple destinations simultaneously using a single, local ‘transmit’ operation. The more familiar cases of Unicasting and Broadcasting may be considered to be special cases of Multicasting (with the packet delivered to one destination, or ‘all’ destinations, respectively).

Most network layer models, like the one described in RFC 1112 [1] for IP multicasting, assume sources may send their packets to an abstract ‘multicast group addresses’. Link layer support for such an abstraction is assumed to exist, and is provided by technologies such as Ethernet.

ATM is being utilized as a new link layer technology to support a variety of protocols, including IP. With RFC 1483 [2] the IETF defined a multiprotocol mechanism for encapsulating and transmitting packets using AAL5 over ATM Virtual Channels (VCs). However, the ATM Forum’s currently published signalling specification (UNI 3.0 [4], with additions for UNI 3.1 released in late 1994) does not provide the multicast address abstraction. Unicast connections are supported by point to point, bidirectional VCs. Multicasting is supported through point to multipoint VCs. The key limitation is that the sender must have prior knowledge of each intended recipient, and explicitly establish a VC with itself as the root node and the recipients as the leaf nodes.

This document has two broad goals:

- Define a group address registration and membership distribution mechanism that allows UNI 3.1 based networks to support the multicast service of protocols such as IP.
- Define specific endpoint behaviour for managing point to multipoint VCs to achieve efficient multicasting of layer 3 packets.

As the IETF is currently in the forefront of using wide area multicasting this document’s descriptions will often focus on IP service model of RFC 1112. A final chapter will note the multiprotocol application of the architecture.

This document avoids discussion of one highly non-trivial aspect of using ATM – the specification of QoS for VCs being established in response to higher layer needs. Research in this area is still very formative, and so it is assumed that future documents will further clarify the mapping of QoS requirements to VC establishment. The
default at this time is that VCs SHOULD be established with a request for Unspecified Bit Rate (UBR) service (as typified by the IETF’s use of VCs for unicast IP, described in RFC 1755 [6]).

1.1 The Multicast Address Resolution Server (MARS).

The Multicast Address Resolution Server (MARS) is a superset of the ATM ARP Server introduced in RFC 1577 [3]. It acts as a registry, associating layer 3 multicast group identifiers with the ATM interfaces representing the group’s members. MARS messages, based on the ATM ARP format, support the distribution of multicast group membership information between MARS and endpoints (hosts or routers). Endpoint address resolution entities query the MARS when a layer 3 address needs to be resolved to the set of ATM endpoints making up the group at any one time. Endpoints keep the MARS informed when they need to join or leave particular layer 3 groups. To provide for asynchronous notification of group membership changes the MARS manages a point to multipoint VC out to all endpoints desiring multicast support.

Valid arguments can be made for two different approaches to ATM level multicasting of layer 3 packets – through meshes of point to multipoint VCs, or ATM level multicast servers (MCS). Two classes of MARS are described – Class I (allowing VC meshes to support layer 3 traffic), and Class II (which allows either VC meshes or MCSs to be assigned for use on a per-group basis).

1.2 The ATM level multicast Cluster.

Each MARS manages a ‘cluster’ of ATM-attached endpoints. A Cluster is defined as

The set of ATM interfaces chosen to participate in direct ATM connections to achieve multicasting of AAL_SDUs between themselves.

In practice, a Cluster is the set of endpoints that choose to use the same MARS to register their memberships and receive their updates from.

By implication of this definition, traffic between interfaces belonging to different Clusters passes through an inter-cluster device. (In the IP world an inter-cluster device would be an IP multicast router with logical interfaces into each Cluster.) This document explicitly avoids specifying the nature of inter-cluster (layer 3) routing protocols.

The mapping of clusters to other constrained sets of endpoints (such
as unicast Logical IP Subnets) is left to each network administrator. A simple approach in overlaid IP environments would be for each LIS to be served by a separate MARS, with the cluster being built from the LIS members. IP multicast routers would interconnect each LIS as they do with conventional subnets. However, there is no requirement that a cluster be limited to a single LIS.

1.3 Document overview.

This document assumes an understanding of concepts explained in greater detail in RFC 1112, RFC 1577, UNI 3.1, and RFC 1755 [6].

Section 2 provides an overview of IP multicast and what RFC 1112 required from Ethernet.

Section 3 describes in more detail the multicast support services offered by UNI 3.1, and outlines the differences between VC meshes and multicast servers (MCSs) as mechanisms for distributing packets to multiple destinations.

Section 4 provides an overview of the MARS and its relationship to ATM endpoints. This section also discusses the encapsulation of MARS control messages, and some encapsulation issues for data traffic.

Section 5 substantially defines the entire cluster member endpoint behaviour, on both receive and transmit sides. This includes both normal operation and error recovery.

Section 6 summarises the requirements of a Class I MARS, and provides a detailed description of the Class II MARS.

Section 7 looks at how a multicast server (MCS) interacts with a Class II MARS.

Section 8 discusses how IP multicast routers may make novel use of promiscuous and semi-promiscuous group joins. Also discussed is a mechanism designed to reduce the amount of IGMP traffic issued by routers.

Section 9 discusses how this document applies in the more general (non-IP) case.

Section 10 summarises the key proposals, and identifies areas for future research that are generated by this MARS architecture.

The appendices provide discussion on issues that arise out the implementation of this memo. Appendix A discusses MARS and endpoint algorithms for parsing MARS messages. Appendix B describes the
particular problems introduced by the current IGMP paradigms, and possible interim work-arounds. Finally, Appendix C discusses the use of 'clusters' in further detail.

2. Summary of the IP multicast service model.

Under IP version 4 (IPv4), addresses in the range of 224.0.0.0 and 239.255.255.255 are termed 'Class D' or 'multicast group' addresses. These abstractly represent all the IP hosts in the Internet (or some constrained subset of the Internet) who have decided to 'join' the specified group.

RFC1112 requires that a multicast-capable IP interface must support the transmission of IP packets to an IP multicast group address, whether or not the node considers itself a 'member' of that group. Consequently, group membership is effectively irrelevant to the transmit side of the link layer interfaces. When Ethernet is used as the link layer (the example used in RFC1112), no address resolution is required to transmit packets. An algorithmic mapping from IP multicast address to Ethernet multicast address is performed locally before the packet is sent out the local interface in the same 'send and forget' manner as a unicast IP packet.

Joining and Leaving an IP multicast group is more explicit on the receive side - with the primitives JoinLocalGroup and LeaveLocalGroup affecting what groups the local link layer interface should accept packets from. When the IP layer wants to receive packets from a group, it issues JoinLocalGroup. When it no longer wants to receive packets, it issues LeaveLocalGroup. A key point to note is that changing state is a local issue, it has no affect on other hosts attached to the Ethernet.

IGMP is defined in RFC 1112 to support IP multicast routers attached to a given subnet. Hosts issue IGMP Report messages when they perform a JoinLocalGroup, or in response to an IP multicast router sending an IGMP Query. By periodically transmitting queries IP multicast routers are able to identify what IP multicast groups have non-zero membership on a given subnet.

A specific IP multicast address, 224.0.0.1, is allocated for the transmission of IGMP Query messages. All IP multicast hosts must issue JoinLocalGroup for 224.0.0.1 during their initialisation. Each host keeps a list of IP multicast groups it has been JoinLocalGroup’d to. When a router issues an IGMP Query on 224.0.0.1 each host begins to send IGMP Reports for each group it is a member of. IGMP Reports are sent to the group address, not 224.0.0.1, "so that other members of the same group on the same network can overhear the Report" and
not bother sending one of their own. IP multicast routers conclude
that a group has no members on the subnet when IGMP Queries no longer
elicit associated replies.

3. UNI 3.1 support for intra-cluster multicasting.

This document will describe its operation in terms of ‘generic’
functions that should be available to clients of a UNI 3.1 signalling
entity in a given ATM endpoint. The ATM model broadly describes an
‘AAL User’ as any entity that establishes and manages VCs and
underlying AAL services to exchange data. An IP over ATM interface is
a form of ‘AAL User’ (although the default LLC/SNAP encapsulation
mode specified in RFC1755 really requires that an ‘LLC entity’ is the
AAL User, which in turn supports the IP/ATM interface).

The most fundamental limitations of UNI 3.1’s multicast support are:

- Only point to multipoint, unidirectional VCs may be established.

- Only the root (source) node of a given VC may add or remove leaf
nodes.

Leaf nodes are identified by their unicast ATM addresses. UNI 3.1
defines two ATM address formats - native E.164 and NSAP (although it
must be stressed that the NSAP address is so called because it uses
the NSAP format - an ATM endpoint is NOT a Network layer termination
point). In UNI 3.1 an ‘ATM Number’ is the primary identification of
an ATM endpoint, and it may use either format. Under some
circumstances an ATM endpoint must be identified by both a native
E.164 address (identifying the attachment point of a private network
to a public network), and an NSAP address (‘ATM Subaddress’) identifying the final endpoint within the private network. For the
rest of this document the term will be used to mean either a single
‘ATM Number’ or an ‘ATM Number’ combined with an ‘ATM Subaddress’.

3.1 VC meshes.

The most fundamental approach to intra-cluster multicasting is the
multicast VC mesh. Each source establishes its own independent point
to multipoint VC (a single multicast tree) to the set of leaf nodes
(destinations) that it has been told are members of the group it
wishes to send packets to.

Interfaces that are both senders and group members (leaf nodes) to a
given group will originate one point to multipoint VC, and terminate
one VC for every other active sender to the group. This criss-
crossing of VCs across the ATM network gives rise to the name ‘VC
mesh’.
3.2 Multicast Servers.

An alternative model has each source establish a VC to an intermediate node - the multicast server (MCS). The multicast server itself establishes and manages a point to multipoint VC out to the actual desired destinations.

The MCS reassembles AAL_SDUs arriving on all the incoming VCs, and then queues them for transmission on its single outgoing point to multipoint VC. (Reassembly of incoming AAL_SDUs is required at the multicast server as AAL5 does not support cell level multiplexing of different AAL_SDUs on a single outgoing VC.)

The leaf nodes of the multicast server’s point to multipoint VC must be established prior to packet transmission, and the multicast server requires an external mechanism to identify them. A side-effect of this method is that ATM interfaces that are both sources and group members will receive copies of their own packets back from the MCS. (An alternative method is for the multicast server to explicitly retransmit packets on individual VCs between itself and group members. A benefit of this second approach is that the multicast server can ensure that sources do not receive copies of their own packets.)

An MCS does NOT pay any attention to the contents of each AAL_SDU. It is purely an AAL/ATM level device.

3.3 Tradeoffs.

Arguments over the relative merits of VC meshes and multicast servers have raged for some time. Ultimately the choice depends on the relative trade-offs a system administrator must make between throughput, latency, congestion, and resource consumption. Even criteria such as latency can mean different things to different people - is it end to end packet time, or the time it takes for a group to settle after a membership change? The final choice depends on the characteristics of the applications generating the multicast traffic.

If we focussed on the data path we might prefer the VC mesh because it lacks the obvious single congestion point of an MCS. Throughput is likely to be higher, and end to end latency lower, because the mesh lacks the intermediate AAL_SDU reassembly that must occur in MCSs. The underlying ATM signalling system also has greater opportunity to ensure optimal branching points at ATM switches along the multicast trees originating on each source.

However, resource consumption will be higher. Every group member’s
ATM interface must terminate a VC per sender (consuming on-board memory for state information, instance of an AAL service, and buffering in accordance with the vendors particular architecture). On the contrary, with a multicast server only 2 VCs (one out, one in) are required, independent of the number of senders. The allocation of VC related resources is also lower within the ATM cloud when using a multicast server. These points may be considered to have merit in environments where VCs across the UNI or within the ATM cloud are valuable (e.g. the ATM provider charges on a per VC basis), or AAL contexts are limited in the ATM interfaces of endpoints (many current implementations allow only 2k, 1k, or less).

If we focus on the signalling load then MCSs have the advantage when faced with dynamic sets of receivers. Every time the membership of a multicast group changes (a leaf node needs to be added or dropped), only a single point to multipoint VC needs to be modified when using an MCS. This generates a single signalling event across the MCS’s UNI. However, when membership change occurs in a VC mesh, signalling events occur at the UNIs of every traffic source - the transient signalling load scales with the number of sources. This has obvious ramifications if you define latency as the time for a group’s connectivity to stabilise after change (especially as the number of senders increases).

Finally, as noted above, MCSs introduce a ‘reflected packet’ problem, which requires additional per-AAL_SDU information to be carried in order for layer 3 sources to detect their own AAL_SDU coming back.

The Class II MARS allows system administrators to utilize either approach on a group by group basis.

### 3.4 Interaction with local UNI 3.1 signalling entity.

The following generic signalling functions are presumed to be available to local AAL Users:

- `L_CALL_RQ` - Establish a unicast VC to a specific endpoint.
- `L_MULTI_RQ` - Establish multicast VC to a specific endpoint.
- `L_MULTI_ADD` - Add new leaf node to previously established VC.
- `L_MULTI_DROP` - Remove specific leaf node from established VC.
- `L_RELEASE` - Release unicast VC, or all Leaves of a multicast VC.

The signalling exchanges and local information passed between AAL User and UNI 3.1 signalling entity with these functions are outside the scope of this document.

The following indications are assumed to be available to AAL Users, generated by the local UNI 3.1 signalling entity:
L_ACK - Successful completion of a local request.
L_REMOTE_CALL - A new VC has been established to the AAL User.
ERR_L_RQFAILED - A remote ATM endpoint rejected an L_CALL_RQ, L_MULTI_RQ, or L_MULTI_ADD.
ERR_L_RELEASE - A remote ATM endpoint terminated an existing VC.

The signalling exchanges and local information passed between AAL User and UNI 3.1 signalling entity with these functions are outside the scope of this document.

4. Overview of the MARS.

The MARS may reside within any ATM endpoint that is directly addressable by the endpoints it is serving. Endpoints wishing to join a multicast cluster must be configured with the ATM address of the node on which the cluster’s MARS resides. (Section 5.4 describes how backup MARSs may be added to support the activities of a cluster. References to ‘the MARS’ in following sections will be assumed to mean the acting MARS for the cluster.)

Architecturally the MARS is an evolution of the RFC 1577 ARP Server. Whilst the ARP Server keeps a table of {IP,ATM} address pairs for all IP endpoints in an LIS, the MARS keeps extended tables of {layer 3 address, ATM.1, ATM.2, ...... ATM.n} mappings. It can either be configured with certain mappings, or dynamically ‘learn’ mappings. The format of the (layer 3 address) field is generally not interpreted by the MARS (except for a few special cases, described later).

A single MARS may not support more than one cluster (by definition). However, a single ATM node may support multiple logical MARSs, each of which support a separate cluster. The restriction is that each MARS has a unique ATM address (e.g. a different SEL field in the NSAP address of the node on which the multiple MARSs reside).

Two classes of MARS are defined in this memo - Class I (with the minimum support required to enable multicasting using VC meshes), and Class II (Class I + extensions to support the introduction of MCSs). Both Class I and Class II MARS distributes group membership information to cluster members over a point to multipoint VC known as the ClusterControlVC. A Class II MARS also establishes a separate point to multipoint VC out to registered MCSs, known as the ServerControlVC. All cluster members are leaf nodes of ClusterControlVC. All registered multicast servers are leaf nodes of ServerControlVC (described further in section 6).

The MARS message format is an extension of the ATM ARP message format. By default all MARS messages MUST be LLC/SNAP encapsulated.
in accordance with RFC 1483, using the same encapsulation as ATM ARP:

\[ [\text{0xAA-AA-03}] [\text{0x00-00-00}] [\text{0x08-06}] \text{[MARS message]} \]

(LLC) (OUI) (PID)

The choice of common encapsulation and message format means that MARS and ARP Server functionality may be implemented within a common entity if a network designer so chooses.

Finally, the MARS does NOT take part in the actual multicasting of layer 3 data packets.

5. Endpoint (MARS client) interface behaviour.

This section describes in detail the operation of what might best be thought of as a ‘shim layer’, sitting between your layer 3 protocol’s link layer interface and the underlying UNI 3.1 service. An endpoint in this context can be a host or a router - any entity that requires a generic ‘layer 3 over ATM’ interface to support layer 3 multicast. It is broken into two key subsections - one for the transmit side, and one for the receive side.

Multiple logical ATM interfaces may be supported by a single physical ATM interface (for example, using different SEL values in the NSAP formatted address assigned to the physical ATM interface). Therefore implementors MUST allow for multiple independent ‘layer 3 over ATM’ interfaces too, each with its own configured MARS (or table of MARSs, as discussed in section 5.4), and ability to be attached to the same or different clusters.

The primary signalling paths between a MARS client (managing an endpoint) and their associated MARS is a transient point to point, bidirectional VC. This VC is established by the MARS client, and is used to send queries to, and receive replies from, the MARS. It has an associated idle timer, and is dismantled if not used for a configurable period of time. The minimum suggested value for this time is 1 minute, and the RECOMMENDED default is 20 minutes. Where the MARS and ARP Server are co-resident, this VC may be used for both ATM ARP traffic and MARS traffic.

Most of this specification is concerned with managing and distributing information that allows the establishment of VCs for actually carrying layer 3 data packets. The actual format of the data carried on these VCs is almost completely outside the scope of this specification. However, when using MCSs (described in section 3) endpoints need to filter out the reflected packets that can occur. The solution to this problem in a general way requires the use of additional per-packet encapsulation. This is discussed in section 5.5.
MARS messages contain variable length address fields. In all cases null addresses MUST be encoded as zero length, and have no space allocated in the message. Addresses with non-zero length, but zero value can have specific meanings to the MARS, and MUST NOT be used in any other fashion.

5.1 Transmit side behaviour.

The following description will often be in terms of an IP/ATM interface that is capable of transmitting packets to a Class D address at any time, without prior warning. It should be trivial for an implementor to generalise this behaviour to the requirements of another layer 3 data protocol.

When a packet arrives for transmission, and there is no outgoing VC already marked as serving the packet’s multicast destination address, the MARS is queried for the set of ATM endpoints currently making up the multicast group.

The query is executed by issuing a MARS_REQUEST. The MARS_REQUEST message is formatted as an ATM ARP_REQUEST (RFC 1577) with operation type code (ar$op field) of 11 (decimal). The reply from the MARS may take one of two forms:

- **MARS_MULTI** - Sequence of MARS_MULTI messages returning the set of ATM endpoints that are to be leaf nodes of the outgoing VC.
- **MARS_NAK** - No mapping found, group is empty.

5.1.1 Retrieving Group Membership from the MARS.

If the MARS had no mapping for the desired Class D address a MARS_NAK will be returned. In this case the IP packet MUST be discarded silently. If a match is found in the MARS’s tables it proceeds to return addresses ATM.1 through ATM.n in a sequence of one or more MARS_MULTIs. A simple mechanism is used to detect and recover from loss of MARS_MULTI messages.

Each MARS_MULTI carries a boolean field x, and a 15 bit integer field y - expressed as MARS_MULTI(x,y). Field y acts as a sequence number, starting at 1 and incrementing for each MARS_MULTI sent. Field x acts as an ‘end of reply’ marker. When x == 1 the MARS response is considered complete.

In addition, each MARS_MULTI may carry multiple ATM addresses from the set {ATM.1, ATM.2, .... ATM.n}. A MARS MUST minimise the number of MARS_MULTIs transmitted by placing as many group member’s
addresses in a single MARS_MULTI as possible. The limit on the length of an individual MARS_MULTI message MUST be the MTU of the underlying VC.

Assume n ATM addresses must be returned, each MARS_MULTI is limited to only p ATM addresses, and p << n. This would require a sequence of k MARS_MULTI messages (where k = \( \frac{n}{p}+1 \), using integer arithmetic), transmitted as follows:

- MARS_MULTI(0,1) carries back \{ATM.1 ... ATM.p\}
- MARS_MULTI(0,2) carries back \{ATM.(p+1) ... ATM.(2p)\}
- \[........\]
- MARS_MULTI(1,k) carries back \{ ... ATM.n\}

If k == 1 then only MARS_MULTI(1,1) is sent.

Typical failure mode will be losing one or more of MARS_MULTI(0,1) through MARS_MULTI(0,k-1). This is detected when y jumps by more than one between consecutive MARS_MULTI’s. An alternative failure mode is losing MARS_MULTI(1,k). A timer MUST be implemented to flag the failure of the last MARS_MULTI to arrive. A default value of 10 seconds is suggested.

If a ‘sequence jump’ is detected, the host MUST wait for the MARS_MULTI(1,k), discard all results, and repeat the MARS_REQUEST.

If a timeout occurs, the host MUST discard all results, and repeat the MARS_REQUEST.

(Corruption of cell contents will lead to loss of a MARS_MULTI through AAL5 CPCS_PDU reassembly failure, which will be detected through the mechanisms described above.)

If the MARS is managing a cluster of endpoints spread across different but directly accessible ATM networks it will not be able to return all the group members in a single MARS_MULTI. The MARS_MULTI message format allows for either E.164, ISO NSAP, or (E.164 + NSAP) to be returned as ATM addresses. However, each MARS_MULTI message may only return ATM addresses of the same type and length. The returned addresses MUST be grouped according to type (E.164, ISO NSAP, or both) and returned in a sequence of separate MARS_MULTI parts.

5.1.2 MARS_REQUEST, MARS_MULTI, and MARS_NAK messages.

MARS_REQUEST is an RFC1577 ATM ARP_REQUEST, but with an ‘operation type value’ of 11 (decimal). The multicast address being resolved is placed into the the target protocol address field \( \text{ar$tpa} \). The target hardware address is set to null \( \text{ar$thtl} \) and \( \text{ar$tstl} \) both
zero). The hardware type (ar$hrd) is set to 19 (decimal), and in IP environments the protocol type is 2048 (decimal). Section 6.6 of RFC 1577 should be consulted for specific details and coding of the ar$shl and ar$sstl fields.

MARS_NAK is the MARS_REQUEST returned with operation type value of 16 (decimal). All other fields should be left unchanged from the MARS_REQUEST.

The MARS_MULTI message is identified by an ‘operation type value’ of 12 (decimal). The message format is:

Data:
- ar$hrd 16 bits  Hardware type (19 decimal, 0x13 hex)
- ar$pro 16 bits  Protocol type
- ar$shtl 8 bits  Type & length of source ATM number (q)
- ar$sstl 8 bits  Type & length of source ATM subaddress (r)
- ar$op 16 bits  Operation code (MARS_MULTI)
- ar$spln 8 bits  Length of source protocol address (s)
- ar$thtl 8 bits  Type & length of target ATM number (x)
- ar$sttl 8 bits  Type & length of target ATM subaddress (y)
- ar$tnum 16 bits  Number of target ATM addresses returned (N).
- ar$seqxy 16 bits  Boolean flag x and sequence number y.
- ar$msn 32 bits  MARS Sequence Number.
- ar$sha qoctets  source ATM number
- ar$ssa roctets  source ATM subaddress
- ar$spa soctets  source protocol address
- ar$tpa zoctets  target multicast group address
- ar$tha.1 xoctets  target ATM number 1
- ar$tsa.1 yoctets  target ATM subaddress 1
- ar$tha.2 xoctets  target ATM number 2
- ar$tsa.2 yoctets  target ATM subaddress 2
- [........]
- ar$tha.N xoctets  target ATM number N
- ar$tsa.N yoctets  target ATM subaddress N

ar$seqxy is coded with flag x in the leading bit, and sequence number y coded as an unsigned integer in the remaining 15 bits.

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

ar$tnum is an unsigned integer indicating how many pairs of (ar$tha,ar$tsa) (i.e. how many group member’s ATM addresses) are
present in the message. ar$msn is an unsigned 32 bit number filled in by the MARS before transmitting each MARS_MULTI. Its use is described further in section 5.1.4. Section 6.6 of RFC 1577 should be consulted for specific details and coding of all other fields.

As an example, assume we have a multicast cluster using 4 byte protocol addresses, 20 byte ATM numbers, and 0 byte ATM subaddresses. For n group members in a single MARS_MULTI we require a $(44 + 20n)$ byte message. If we assume the default MTU of 9180 bytes, we can return a maximum of 456 group member’s addresses in a single MARS_MULTI.

5.1.3 Establishing the outgoing multipoint VC.

Following the completion of the MARS_MULTI reply the endpoint may establish a new point to multipoint VC, or reuse an existing one.

If establishing a new VC, an L_MULTI_RQ is issued for ATM.1, followed by an L_MULTI_ADD for every member of the set {ATM.2, ..., ATM.n} (assuming the set is non-null). The packet is then transmitted over the newly created VC just as it would be for a unicast VC.

After transmitting the packet, the local interface holds the VC open and marks it as the active path out of the host for any subsequent IP packets being sent to that Class D address.

When establishing a new multicast VC it is possible that one or more L_MULTI_RQ or L_MULTI_ADD may fail. The UNI 3.1 failure cause must be returned in the ERR_L_RQFAILED signal from the local signalling entity to the AAL User. If the failure cause is not 49 (Quality of Service unavailable) or 51 (user cell rate not available), the endpoint’s ATM address is dropped from the set {ATM.1, ATM.2, ..., ATM.n} returned by the MARS. Otherwise, the L_MULTI_RQ or L_MULTI_ADD should be reissued after a delay of 10 to 20 seconds. If the request fails again, another request should be issued after twice the previous delay has elapsed. This process should be continued until the call succeeds or the multipoint VC gets released.

If the initial L_MULTI_RQ fails for ATM.1, and n is greater than 1 (i.e. the returned set of ATM addresses contains 2 or more addresses) a new L_MULTI_RQ should be immediately issued for the next ATM address in the set. This procedure is repeated until an L_MULTI_RQ succeeds, as no L_MULTI_ADDs may be issued until an initial outgoing VC is established.

Each ATM address for which an L_MULTI_RQ failed with cause 49 or 51 MUST be tagged rather than deleted. An L_MULTI_ADD is issued for these tagged addresses using the random delay procedure outlined
The VC MAY be considered ‘up’ before failed L_MULTI_ADDs have been successfully re-issued. An endpoint MAY implement a concurrent mechanism that allows data to start flowing out the new VC even while failed L_MULTI_ADDs are being re-tried. (The alternative of waiting for each leaf node to accept the connection could lead to significant delays in transmitting the first packet.)

Each VC MUST have a configurable inactivity timer associated with it. If the timer expires, an L_RELEASE is issued for that VC, and the Class D address is no longer considered to have an active path out of the local host. The timer SHOULD be no less than 1 minute, and a default of 20 minutes is RECOMMENDED. Choice of specific timer periods is beyond the scope of this document.

VC consumption may also be reduced by endpoints noting when a new group’s set of {ATM.1, ....ATM.n} matches that of a pre-existing VC out to another group. With careful local management, and assuming the QoS of the existing VC is sufficient for both groups, a new pt to mpt VC may not be necessary. Under certain circumstances endpoints may decide that it is sufficient to re-use an existing VC whose set of leaf nodes is a superset of the new group’s membership (in which case some endpoints will receive multicast traffic for a layer 3 group they haven’t joined, and must filter them above the ATM interface). Algorithms for performing this type of optimization are not discussed here, and are not required for conformance with this memo.

5.1.4 Monitoring updates on ClusterControlVC.

Once a new VC has been established, the transmit side of the cluster member’s interface needs to monitor subsequent group changes – adding or dropping leaf nodes as appropriate. This is achieved by watching for MARS_JOIN and MARS_LEAVE messages from the MARS itself. These messages are described in detail in section 5.2 – at this point it is sufficient to note that they carry:

- The ATM address of a node joining or leaving a group.
- The layer 3 address of the group(s) being joined or left.
- A Cluster Sequence Number (CSN) from the MARS.

MARS_JOIN and MARS_LEAVE messages arrive at each cluster member across ClusterControlVC. MARS_JOIN or MARS_LEAVE messages that simply confirm information already held by the cluster member are used to track the Cluster Sequence Number, but are otherwise ignored.
5.1.4.1  Updating the active VCs.

If a MARS_JOIN is seen that refers to (or encompasses) a group for which the transmit side already has a VC open, the new member’s ATM address is extracted and an L_MULTI_ADD issued locally. This ensures that endpoints already sending to a given group will immediately add the new member to their list of recipients.

If a MARS_LEAVE is seen that refers to (or encompasses) a group for which the transmit side already has a VC open, the old member’s ATM address is extracted and an L_MULTI_DROP issued locally. This ensures that endpoints already sending to a given group will immediately drop the old member from their list of recipients. When the last leaf of a VC is dropped, the VC is closed completely and the affected group no longer have a path out of the local endpoint (the next outbound packet to that group’s address will trigger the creation of a new VC, as described in sections 5.1.1 to 5.1.3).

In an IPv4 environment any endpoint leaving 224.0.0.1 is assumed to be ceasing support for IP multicast operation. If a MARS_LEAVE is seen that refers to group 224.0.0.1 then the ATM address of the endpoint specified in the message MUST be removed from every multipoint VC on which it is listed as a leaf node.

The transmit side of the interface MUST NOT shut down an active VC to a group for which the receive side has just executed a LeaveLocalGroup. This behaviour is consistent with the model of hosts transmitting to groups regardless of their own membership status.

If a MARS_JOIN or MARS_LEAVE arrives with ar$pnum == 0 it carries no <min,max> pairs, and is only used for tracking the CSN (and possibly for confirming the transmission of the local cluster member’s own MARS_JOIN or MARS_LEAVE, as described in section 5.2.2).

5.1.4.2  Tracking the Cluster Sequence Number.

It is important that endpoints do not miss group membership updates issued by the MARS over ClusterControlVC. However, this will happen from time to time. The Cluster Sequence Number is carried as an unsigned 32 bit value in the ar$msn field of many MARS messages (except for MARS_REQUEST and MARS_NAK). It increments once for every transmission the MARS makes on ClusterControlVC, regardless of whether the transmission represents a change in the MARS database or not. By tracking this counter, cluster members can determine whether they have missed a previous message on ClusterControlVC, and possibly a membership change. This is then used to trigger revalidation (described in section 5.1.5).
The current CSN is copied into the ar$msn field of MARS messages being sent to cluster members, whether out ClusterControlVC or on an point to point VC.

Calculations on the sequence numbers MUST be performed as unsigned 32 bit arithmetic, to ensure no glitches when the counters roll over.

Every cluster member keeps its own 32 bit Host Sequence Number (HSN) to track the MARS’s sequence number. Whenever a message is received that carries an ar$msn field the following processing is performed:

\[
\text{Seq.diff} = \text{ar$msn} - \text{HSN}
\]

\[
\text{ar$msn} \rightarrow \text{HSN}
\]

\[
\text{...process MARS message as appropriate...}
\]

\[
\text{if } ((\text{Seq.diff} \neq 1) \&\& (\text{Seq.diff} \neq 0))
\]

\[
\text{then } [...\text{revalidate group membership information...}]
\]

The basic result is that the cluster member attempts to keep locked in step with membership changes noted by the MARS. If it ever detects that a membership change occurred (in any group) without it noticing, it re-validates the membership of all groups it currently has multicast VCs open to.

The ar$msn value in an individual MARS_MULTI is not used to update the HSN until all parts of the MARS_MULTI (if more than 1) have arrived. However, the ar$msn field in consecutive messages of a multi-part MARS_MULTI MUST be constant. If the ar$msn field changes before the MARS_MULTI is completely received, then the entire MARS_MULTI MUST be discarded at the completion of the response, and the MARS_REQUEST re-issued.

The MARS is free to choose an initial value of CSN. When a new cluster member starts up it should initialise HSN to zero. When the cluster member sends the MARS_JOIN to register (described later), the HSN will be correctly updated to the current CSN value when the endpoint receives the copy of its MARS_JOIN back from the MARS.

5.1.5 Revalidating a VC’s leaf nodes.

Certain events may inform a cluster member that it has incorrect information about the sets of leaf nodes it should be sending to. If an error occurs on a VC associated with a particular group, the cluster member initiates revalidation procedures for that specific group. If a jump is detected in the Cluster Sequence Number, this initiates revalidation of all groups to which the cluster member currently has open point to multipoint VCs.
Each open and active multipoint VC has a flag associated with it called 'VC_revalidate'. This flag is checked everytime a packet is queued for transmission on that VC. If the flag is false, the packet is transmitted and no further action is required.

However, if the VC_revalidate flag is true then the packet is transmitted and a new sequence of events is started locally.

Revalidation begins with re-issuing a MARS_REQUEST for the group being revalidated. The returned set of members {NewATM.1, NewATM.2, .... NewATM.n} is compared with the set already held locally. L_MULTI_DROPS are issued on the group’s VC for each node that appears in the original set of members but not in the revalidated set of members. L_MULTI_ADDs are issued on the group’s VC for each node that appears in the revalidated set of members but not in the original set of members. The VC_revalidate flag is reset when revalidation concludes for the given group. Implementation specific mechanisms will be needed to flag the ‘revalidation in progress’ state.

The key difference between constructing a VC (section 5.1.3) and revalidating a VC is that packet transmission continues on the open VC while it is being revalidated. This minimises the disruption to existing traffic.

The general algorithm for initiating revalidation is:

- When a packet arrives for transmission on a given group, the groups membership is revalidated if VC_revalidate == TRUE. Revalidation resets VC_revalidate.
- When an event occurs that demands revalidation, every group has its VC_revalidate flag set TRUE at a random time between 1 and 10 seconds.

Benefit: Revalidation of active groups occurs quickly, and essentially idle groups are revalidated as needed. Randomly distributed setting of VC_revalidate flag improves chances of staggered revalidation requests from senders when a sequence number jump is detected.

5.1.5.1 When leaf node drops itself.

During the life of a multipoint VC an ERR_L_RELEASE may be received indicating that a leaf node has terminated its participation at the ATM level. The ATM endpoint associated with the ERR_L_RELEASE MUST be removed from the locally held set {ATM.1, ATM.2, .... ATM.n} associated with the VC.

After a random period of time between 1 and 10 seconds the
VC_revalidate flag associated with that VC MUST be set true.

5.1.5.2 When a jump is detected in the CSN.

Section 5.1.4.2 describes how a CSN jump is detected. If a CSN jump is detected upon receipt of a MARS_JOIN or a MARS_LEAVE then every outgoing multicast VC MUST have its VC_revalidate flag set true at some random interval between 1 and 10 seconds from when the CSN jump was detected.

The only exception to this rule is if a sequence number jump is detected during the establishment of a new group’s VC (i.e. a MARS_MULTI reply was correctly received, but its ar$msn indicated that some previous MARS traffic had been missed on ClusterControlVC). In this case every open VC, EXCEPT the one just established, MUST have its VC_revalidate flag set true at some random interval between 1 and 10 seconds from when the CSN jump was detected. (The VC being established at the time is considered already validated.)

5.2. Receive side behaviour.

A cluster member is a ‘group member’ (in the sense that it receives packets directed at a given multicast group) when its ATM address appears in the MARS’s table entry for the group’s multicast address. A key function within each cluster is the distribution of group membership information from the MARS to cluster members.

An endpoint may wish to ‘join a group’ in response to a local, higher level request for membership of a group, or because the endpoint supports a layer 3 multicast forwarding engine that requires the ability to ‘see’ intra-cluster traffic in order to forward it.

Two messages support these requirements - MARS_JOIN and MARS_LEAVE. These are sent to the MARS by endpoints when the local layer 3/ATM interface is requested to join or leave a multicast group. The MARS propagates these messages back out over ClusterControlVC, to ensure the knowledge of the group’s membership change is distributed in a timely fashion to other cluster members.

Certain models of layer 3 endpoints (e.g. IP multicast routers) expect to be able to receive packet traffic ‘promiscuously’ across all groups. This functionality may be emulated by allowing routers to request that the MARS returns them as ‘wild card’ members of all Class D addresses. However, a problem inherent in the current ATM model is that a completely promiscuous router may exhaust the local reassembly resources in its ATM interface. MARS_JOIN supports a generalisation to the notion of ‘wild card’ entries, enabling routers to limit themselves to ‘blocks’ of the Class D address space. Use of
this facility is described in greater detail in Section 8.

A block can be as small as 1 (a single group) or as large as the entire multicast address space (e.g. default IPv4 'promiscuous' behaviour). A block is defined as all addresses between, and inclusive of, a <min,max> address pair. A MARS_JOIN or MARS_LEAVE may carry multiple <min,max> pairs.

Cluster members MUST provide ONLY a single <min,max> pair in each JOIN/LEAVE message they issue. However, they MUST be able to process multiple <min,max> pairs in JOIN/LEAVE messages when performing VC management as described in section 5.1.4 (the interpretation being that the join/leave operation applies to all addresses in range from <min> to <max> inclusive, for every <min,max> pair).

In RFC1112 environments a MARS_JOIN for a single group is triggered by a JoinLocalGroup signal from the IP layer. A MARS_LEAVE for a single group is triggered by a LeaveLocalGroup signal from the IP layer.

Cluster members with special requirements (e.g. multicast routers) may issue MARS_JOINs and MARS_LEAVEs specifying a block of multicast group addresses.

An endpoint MUST register with a MARS in order to become a member of a cluster and be added as a leaf to ClusterControlVC. Registration is covered in section 5.2.3.

Finally, the endpoint MUST be capable of terminating unidirectional VCs (i.e. act as a leaf node of a UNI 3.1 point to multipoint VC). RFC 1755 describes the information required to terminate VCs carrying LLC/SNAP encapsulated traffic (discussed further in section 5.5).

5.2.1 Format of the MARS_JOIN and MARS_LEAVE Messages.

The MARS_JOIN message is indicated by an operation type value of 14 (decimal). MARS_LEAVE has the same format and operation type value of 15 (decimal). The message format is:

Data:
- ar$hrd 16 bits  Hardware type (19 decimal)
- ar$pro 16 bits  Protocol type
- ar$shtl 8 bits  Type & length of source ATM number (q)
- ar$sstl 8 bits  Type & length of source ATM subaddress (r)
- ar$op 16 bits  Operation code (MARS_JOIN or MARS_LEAVE)
- ar$spln 8 bits  Length of source protocol address (s)
- ar$tpln 8 bits  Length of multicast group address (z)
- ar$pnnum 16 bits  Number of multicast group address pairs (N)
ar$resv 16 bits ar$layer3grp flag, and 15 bits reserved.
ar$cmi 16 bits Cluster Member ID
ar$msn 32 bits MARS Sequence Number.
ar$sha qoctets source ATM number (E.164 or ATM Forum NSAPA).
ar$ssa roctets source ATM subaddress (ATM Forum NSAPA).
ar$spa soctets source protocol address
ar$min.1 zoctets Minimum multicast group address - pair.1
ar$max.1 zoctets Maximum multicast group address - pair.1
[........]
ar$min.N zoctets Minimum multicast group address - pair.N
ar$max.N zoctets Maximum multicast group address - pair.N

Refer to RFC 1577, section 6.6 for the coding of the ar$shtl and ar$sstl fields. ar$spln indicates the number of bytes in the source endpoint’s protocol address, and is interpreted in the context of the protocol indicated by the ar$pro field. (e.g. in IPv4 environments ar$pro will be 0x800, ar$spln is 4, and ar$tpln is 4.)

The ar$resv field contains a flag - ar$layer3grp - in its most significant bit, and 15 unused bits which MUST be zero. This flag is to allow the MARS to provide the ‘short cut’ group membership information described further in section 5.3. The rules for its use are:

ar$layer3grp MUST be set when the cluster member is issuing the MARS_JOIN a the result of a layer 3 multicast group being explicitly joined. (e.g. as a result of a JoinHostGroup operation in an RFC1112 compliant host).

The flag MUST be reset in each MARS_JOIN if the MARS_JOIN is simply the local ip/atm interface registering to receive traffic on that group for its own reasons.

The flag is ignored and MUST be treated as reset by the MARS for any MARS_JOIN that specifies a block covering more than a single group (e.g. a block join from a router ensuring their forwarding engines ‘see’ all traffic).

ar$pnum indicates how many <min,max> pairs are included in the message. This field must always be 1 when the message is sent from a cluster member. (It will be unchanged when returned by a Class I MARS. A Class II MARS may return a MARS_JOIN or MARS_LEAVE with any ar$pnum value, including zero. This will be explained further in section 6.2.4.)

The ar$cmi field SHOULD be zeroed by cluster members, and is used by the MARS during cluster member registration, described in section 5.2.3.
ar$msn MUST be zero when transmitted by an endpoint. It is set to the current value of the Cluster Sequence Number by the MARS when the MARS_JOIN or MARS_LEAVE is retransmitted. Its use has been described in section 5.1.4.

To simplify construction and parsing of MARS_JOIN and MARS_LEAVE messages, the following restrictions are imposed on the <min,max> pairs:

Assume max(N) is the <max> field from the Nth <min,max> pair.
Assume min(N) is the <min> field from the Nth <min,max> pair.
Assume a join/leave message arrives with K <min,max> pairs.
The following must hold:
\[
\begin{align*}
\text{max}(N) &< \text{min}(N+1) \quad \text{for } 1 \leq N < K \\
\text{max}(N) &\geq \text{min}(N) \quad \text{for } 1 \leq N \leq K
\end{align*}
\]

In plain english, the set must specify an ascending sequence of address blocks. The definition of "greater" or "less than" may be protocol specific. In IPv4 environments the addresses are treated as 32 bit, unsigned binary values (most significant byte first).

5.2.1.1 Important IPv4 default values.

The JoinLocalGroup and LeaveLocalGroup operations are only valid for a single group. For any arbitrary group address X the associated MARS_JOIN or MARS_LEAVE MUST specify a single pair <X, X>. In general the ar$layer3grp flag MUST be set under these circumstances.

A router choosing to behave strictly in accordance with RFC1112 MUST specify the entire Class D space. The associated MARS_JOIN or MARS_LEAVE MUST specify a single pair <224.0.0.0, 239.255.255.255>. Whenever a router issues a MARS_JOIN only in order to forward IP traffic it MUST reset the ar$layer3grp flag.

The use of alternative <min, max> values by multicast routers is discussed in Section 8.

5.2.2 Retransmission of MARS_JOIN and MARS_LEAVE messages.

Transient problems may result in the loss of messages between the MARS and cluster members.

A simple algorithm is used to solve this problem. Cluster members retransmit each MARS_JOIN and MARS_LEAVE message at regular intervals until they receive a copy back again, either on ClusterControlVC or the VC on which they are sending the message. At this point the local endpoint can be certain that the MARS received and processed it.
The interval should be no shorter than 5 seconds, and a default value of 10 seconds is recommended. After 5 retransmissions the attempt should be flagged locally as a failure. This MUST be considered as a MARS failure, and triggers the MARS reconnection described in section 5.4.

A ‘copy’ is defined as seeing a message of the same operation code containing the local host’s identity in the source address fields. The <min,max> pair set is not checked, and does not have to be the same (this is required to be compatible with the modification that a Class II MARS may effect on the retransmitted MARS_JOIN or MARS_LEAVE message).

This algorithm explicitly allows only ONE outstanding MARS_JOIN and MARS_LEAVE message at a time (although you may have one of both outstanding).

5.2.3 Registering with the MARS.

To become a cluster member an endpoint must register with the MARS. This achieves two things - the endpoint is added as a leaf node of ClusterControlVC, and the endpoint is assigned a 16 bit Cluster Member Identifier (CMI). The CMI uniquely identifies each endpoint that is attached to the cluster.

Registration with the MARS occurs when an endpoint issues a MARS_JOIN for a protocol specific multicast group address.

In IPv4 environments an endpoint (whether in a host or router) MUST explicitly issue a MARS_JOIN for the special address "0.0.0.0" in order to register with the MARS. In other words, a MARS_JOIN with ar$tpln of 4, and 8 bytes of zero starting at ar$min.1 (equivalent to the block of <0.0.0.0,0.0.0.0>). This function may be internal to the IP/ATM driver, and does not require the IP layer to believe it has 'joined' the all-zeros IP address.

The specific addresses signifying ‘registration’ for other layer 3 protocols will be defined in subsequent documents.

The cluster member retransmits this MARS_JOIN in accordance with section 5.2.2 until it confirms that the MARS has received it.

When the registration MARS_JOIN is returned it contains a non-zero value in ar$cmi. This value MUST be noted by the cluster member, and used whenever circumstances require the cluster member’s CMI.

An endpoint may also choose to de-register, using a MARS_LEAVE. In an IPv4 environment a MARS_LEAVE on the special address of "0.0.0.0"
would result in the MARS dropping the endpoint from ClusterControlVC and freeing up its CMI

5.3 Support for Layer 3 group management.

Whilst the intention of this specification is to be independent of layer 3 issues, an attempt is being made to assist the operation of layer 3 multicast routing protocols that need to ascertain if any groups have members within a cluster.

One example is IP, where IGMP is used (as described in section 2) simply to determine whether any other cluster members are listening to a group because they have higher layer applications that want to receive a group’s traffic.

Routers may choose to query the MARS for this information, rather than multicasting IGMP queries to 224.0.0.1 and incurring the associated cost of setting up a VC to all systems in the cluster.

The query is issued by sending a MARS_GROUPLIST_REQUEST to the MARS. MARS_GROUPLIST_REQUEST is built from a MARS_JOIN, but it has an operation code of 20 (ar$op = 20). A single <min,max> pair MUST be provided (ar$pnum = 1), and it specifies the range of groups in which the querying cluster member is interested.

The response from the MARS is a MARS_GROUPLIST_REPLY, carrying a list of the multicast groups within the specified <min,max> block that have Layer 3 members. A group is noted in this list if one or more of the MARS_JOINs that generated its mapping entry in the MARS contained a set ar$layer3grp flag.

MARS_GROUPLIST_REPLYs are transmitted back to the querying cluster member on the VC used to send the MARS_GROUPLIST_REQUEST.

MARS_GROUPLIST_REPLY is derived from the MARS_MULTI, it may have multiple parts if needed, and is received in a similar manner.

Data:

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar$hrd</td>
<td>16 bits</td>
<td>Hardware type (19 decimal, 0x13 hex)</td>
</tr>
<tr>
<td>ar$pro</td>
<td>16 bits</td>
<td>Protocol type</td>
</tr>
<tr>
<td>ar$shtl</td>
<td>8 bits</td>
<td>Type &amp; length of source ATM number (q)</td>
</tr>
<tr>
<td>ar$sstl</td>
<td>8 bits</td>
<td>Type &amp; length of source ATM subaddress (r)</td>
</tr>
<tr>
<td>ar$op</td>
<td>16 bits</td>
<td>Operation code (MARS_GROUPLIST_REPLY = 21 decimal)</td>
</tr>
<tr>
<td>ar$spln</td>
<td>8 bits</td>
<td>Length of source protocol address (s)</td>
</tr>
<tr>
<td>ar$tthl</td>
<td>8 bits</td>
<td>Unused - set to zero.</td>
</tr>
<tr>
<td>ar$tstl</td>
<td>8 bits</td>
<td>Unused - set to zero.</td>
</tr>
<tr>
<td>ar$tpn</td>
<td>8 bits</td>
<td>Length of target multicast group address (z)</td>
</tr>
</tbody>
</table>
ar$tnum 16 bits Number of group addresses returned (N).
ar$seqxy 16 bits Boolean flag x and sequence number y.
ar$msn 32 bits MARS Sequence Number.
ar$sha qoctets source ATM number (E.164 or ATM Forum NSAPA).
ar$ssa roctets source ATM subaddress (ATM Forum NSAPA).
ar$spa soctets source protocol address
ar$mgrp.1 zoctets Group address 1
[........]
ar$mgrp.N zoctets Group address N

ar$seqxy is coded as for the MARS_MULTI - multiple MARS_GROUPLIST_REPLY components are transmitted and received using the same algorithm as described in section 5.1.1 for MARS_MULTI. The only difference is that group address are being returned rather than ATM addresses.

As for MARS_MULTIs, if an error occurs in the reception of a multi part MARS_GROUPLIST_REPLY the whole thing MUST be discarded and the MARS_GROUPLIST_REQUEST re-issued. (This includes the ar$msn value being constant.)

Note that the ability to generate MARS_GROUPLIST_REQUEST messages, and receive MARS_GROUPLIST_REPLY messages, is not required for general host interface implementations. It is optional for interfaces being implemented to support layer 3 multicast forwarding engines. However, this functionality MUST be supported by both Class I and Class II MARS.

5.4 Support for redundant/backup MARS entities.

Endpoints are assumed to have been configured with the ATM address of at least one MARS. Endpoints MAY choose to maintain a table of ATM addresses, representing alternative MARSs that will be contacted in the event that normal operation with the original MARS is deemed to have failed. It is assumed that this table orders the ATM addresses in descending order of preference.

An endpoint will typically decide there are problems with the MARS when:

- It fails to establish a point to point VC to the MARS.
- MARS_REQUESTs fail (section 5.1.1).
- MARS_JOIN/MARS_LEAVEs fail (section 5.2.2).

(If it is able to discern which connection represents ClusterControlVC, it may also use connection failures on this VC to indicate problems with the MARS).
5.4.1 First response to MARS problems.

The first response is to assume a transient problem with the MARS being used at the time. The cluster member should wait a random period of time between 1 and 10 seconds before attempting to reconnect and re-register with the MARS. If the registration MARSJOIN is successful then:

The cluster member MUST then proceed to rejoin every group that its local higher layer protocol(s) have joined. It is recommended that a random delay between 1 and 10 seconds be inserted before attempting each MARSJOIN.

The cluster member MUST initiate the revalidation of every multicast group it was sending to (as though a sequence number jump had been detected, section 5.1.5).

The rejoin and revalidation procedure must not disrupt the cluster member’s use of multipoint VCs that were already open at the time of the MARS failure.

If re-registration with the current MARS fails, and there are no backup MARS addresses configured, the cluster member MUST wait for at least 1 minute before repeating the re-registration procedure. It is RECOMMENDED that the cluster member signals an error condition in some locally significant fashion.

This procedure may repeat until network administrators manually intervene or the current MARS returns to normal operation.

5.4.2 Connecting to a backup MARS.

If the re-registration with the current MARS fails, and other MARS addresses has been configured, the next MARS address on the list is chosen to be the current MARS, and the cluster member immediately restarts the re-registration procedure described in section 5.4.1. If it is successful the cluster member will resume normal operation using the new MARS. It is RECOMMENDED that the cluster member signals a warning of this condition in some locally significant fashion.

If the attempt at re-registration with the new MARS fails, the cluster member MUST wait for at least 1 minute before choosing the next MARS address in the table and repeating the procedure. If the end of the table has been reached, the cluster member starts again at the top of the table (which should be the original MARS that the cluster member started with).

In the worst case scenario this will result in cluster members
looping through their table of possible MARS addresses until network administrators manually intervene.

5.4.3 Dynamic backup lists, and soft redirects.

To support some level of autoconfiguration, a MARS message is defined that allows the current MARS to broadcast on ClusterControlVC a table of backup MARS addresses. When this message is received, cluster members that maintain a list of backup MARS addresses MUST insert this information at the top of their locally held list (i.e. the information provided by the MARS has a higher preference than addresses that may have been manually configured into the cluster member).

The message is MARS_REDIRECT_MAP. It is based on a single MARS_MULTI, but with an operation type code of 22 decimal. The source hardware address information MUST be that of the MARS, and the source protocol address field MUST be null (ar$hpln = 0, and no space allocated). The target protocol address MUST be null (ar$tpln = 0, and no space allocated). If a multi-part MARS_REDIRECT_MAP begins arriving it should be reassembled and accepted. If a part is lost, the entire message should simply be discarded.

This message is transmitted regularly by the MARS (it MUST be transmitted at least every 2 minutes, it is RECOMMENDED that it is transmitted every 1 minute).

In addition to keeping cluster members updated with the recommended list of backup MARSs, the MARS_REDIRECT_MAP is used to force cluster members to 'soft redirect' from one MARS to another. If the first ATM address contained in a MARS_REDIRECT_MAP is not the address of the MARS currently being used by a cluster member, the cluster member MUST initiate the following:

- open a point to point VC to the first ATM address.
- attempt a registration (e.g. MARSJOIN for "0.0.0.0").

If the registration succeeds, the cluster member shuts down its point to point VC to the current MARS (if it had one open), and then proceeds to use the newly opened point to point VC as its connection to the 'current MARS'. The cluster member does NOT attempt to rejoin the groups it is a member of, or revalidate groups it is currently sending to.

This is termed a 'soft redirect' because it avoids the extra rejoining and revalidation processing that occurs when a MARS failure is being recovered from. It assumes some external synchronisation mechanisms exist between the old and new MARS - mechanisms that are
outside the scope of this specification.

Some level of trust is required before initiating a soft redirect. A cluster member MUST check that the calling party at the other end of the VC on which the MARS_REDIRECT_MAP arrived (supposedly ClusterControlVC) is in fact the node it trusts as the current MARS.

Additional applications of this function are for further study.

5.5 LLC/SNAP encapsulations for transmit and receive.

Network administrators who require only VC mesh support for their multicasting would use a Class I MARS. In this case the default for data traffic carried on point to multipoint VCs is LLC/SNAP encapsulation with a header appropriate to the protocol being carried. For IP traffic this is defined in RFC 1483 as:

```
[0xAA-AA-03][0x00-00-00][0x08-00][IP packet]
```

Network administrators who require the ability to use MCSs on certain multicast groups will use a Class II MARS. They will also require endpoint interfaces that detect and filter out reflected packets. This is achieved by adding another field of information to the encapsulation that is already wrapped around layer 3 data packets. The information to be included is the Cluster Member Identifier (CMI), which is allocated during registration by both Class I and Class II MARSs (section 5.2.3).

When a packet is transmitted the CMI is inserted into the encapsulation. When a packet is received, if the CMI carried along with it matches the CMI of the local interface the packet is simply dropped.

The recommended encapsulation is:

```
[Editors note: This is a placeholder for the results of the WG discussion on the encapsulation options. Check draft-armitage-ipatm-encaps-01.txt or later version. The WG is expected to come up with some text that will simply be dropped into this section.]
```

Using a different LLC/SNAP value to identify packets containing the CMI allows endpoints to separate and simultaneously support both old and new encapsulated traffic.
6. The MARS in greater detail.

As noted in the overview of section 4, there are two types of MARS defined in this specification. The Class I MARS is a superset of the RFC1577 ARP Server, and is capable of managing clusters where only VC meshes are used to achieve intra-cluster multicasting.

The Class II MARS is a superset of the Class I MARS, with extensions that allow it to transparently introduce multicast servers into the data paths established by endpoints that comply with the specifications in section 5. (It is worth noting here that complete compliance with section 5 includes being able to use the new encapsulation carrying the Cluster Member ID. Networks built around a Class I MARS may choose to initially not fully comply with section 5 in this respect, although it is RECOMMENDED that they do.)

The MARS is intended to be a multiprotocol entity - all its mapping tables and control VCs MUST be managed within the context of the ar$pro field in incoming MARS messages. For example, a MARS supports completely separate ClusterControlVCs for each layer 3 protocol (ar$pro type) that it is registering members for. If a MARS receives messages with an ar$pro type that it does not support, the message is dropped.

6.1 Class I MARS requirements.

A Class I MARS must understand and/or generate the following MARS messages:

11  MARS_REQUEST
12  MARS_MULTI
14  MARS_JOIN
15  MARS_LEAVE
16  MARS_NAK
20  MARS_GROUPLIST_REQUEST
21  MARS_GROUPLIST_REPLY
22  MARS_REDIRECT_MAP

Section 5 covers how these messages are used or reacted to by endpoints within a cluster. This section provides a brief summary of how the Class I MARS uses or reacts to them.

When a registration MARS_JOIN arrives (e.g. for address "0.0.0.0" if ar$pro = 0x800 [IPv4]) the MARS performs the following:

- Adds the node to ClusterControlVC.
- Allocates a new Cluster Member ID (CMI).
- Inserts the new CMI into the ar$cmi field of the MARS_JOIN.
- Retransmits the MARS_JOIN back privately.

If the node is already a registered member of the cluster (given the ar$pro value in the MARS_JOIN) then its CMI is simply copied into the MARS_JOIN, and the MARS_JOIN retransmitted back to the node. A single node may register multiple times if it supports multiple layer 3 protocols. The retransmitted MARS_JOIN must NOT be sent on ClusterControlVC. (If a cluster member issues a MARS_LEAVE for the registration ‘special’ address it too is retransmitted privately.)

All other MARS_JOIN and MARS_LEAVE messages are retransmitted on ClusterControlVC (after successfully performing any required database updates) exactly as they arrived. The MARS retransmits MARS_JOIN and MARS_LEAVE messages even if they result in no change to the database. The ar$layer3grp flag (section 5.3) MUST be ignored (and treated as reset) for MARS_JOINS specifying more than a single group. If a MARS_JOIN is received that contains more than one <min,max> pair, the MARS MUST ignore the second and subsequent pairs.

An additional IPv4 specific behaviour exists - if a node issues a MARS_LEAVE for address "224.0.0.1" (the ‘all systems’ group) it is assumed to have ceased multicast support completely. All references to this node MUST be eliminated from any other IPv4 groups it is a member of in the database. Finally, the endpoint is released as a leaf node from ClusterControlVC.

If the MARS receives an ERR_L_RELEASE on ClusterControlVC indicating that a cluster member has died, that member’s ATM address MUST be removed from all groups for which it may have joined.

As mentioned in section 4, the MARS only needs to interpret the protocol address supplied in MARS messages on a few odd occasions. In general the MARS MUST treat protocol addresses as arbitrary byte strings. For example, the MARS MUST NOT apply IPv4 specific ‘class’ checks to addresses supplied under ar$pro = 0x800 to see if they really are Class D or not. It is sufficient for the MARS to simply assume that endpoints know how to interpret the protocol addresses that they are registering and deregistering mappings for.

A MARS_REDIRECT_MAP message (described in section 5.4.3) MUST be regularly transmitted on ClusterControlVC. It is RECOMMENDED that this occur every 1 minute, and it MUST occur at least every 2 minutes. If the MARS has no knowledge of other backup MARSs serving the cluster, it MUST include its own address as the only entry in the MARS_REDIRECT_MAP message. The design and use of backup MARS entities is beyond the scope of this specification, and will be covered in future work.
The Cluster Sequence Number (CSN) is described in section 5.1.4, and is carried in the ar$msn field of MARS messages being sent to cluster members (either out ClusterControlVC or on an individual VC). The MARS increments the CSN every time a message is sent on ClusterControlVC. The current CSN is copied into the ar$msn field of MARS messages being sent to cluster members, whether out ClusterControlVC or on a private VC.

A MARS should be carefully designed to minimise the possibility of the CSN jumping unnecessarily. Under normal operation only cluster members affected by transient link problems will miss CSN updates and be forced to revalidate. If the MARS itself glitches, it will be inundated with requests for a period as every cluster member attempts to revalidate.

Calculations on the CSN MUST be performed as unsigned 32 bit arithmetic, to ensure no glitches when the counters roll over.

(The regular transmission of MARS_REDIRECT_MAP serves a secondary purpose of allowing cluster members to track the CSN, even if they miss an earlier MARS_JOIN or MARS_LEAVE.)

One implication of this mechanism is that the MARS should serialize its processing of ‘simultaneous’ MARS_REQUEST, MARS_JOIN and MARS_LEAVE messages. Join and Leave operations should be queued within the MARS along with MARS_REQUESTS, and not processed until all the reply packets of a preceeding MARS_REQUEST have been transmitted. The transmission of MARS_REDIRECT_MAP should also be similarly queued.

6.2 Class II MARS requirements.

When using the services of a Class I MARS, the endpoint behaviour described in section 5 results in all groups being supported by meshes of point to multipoint VCs. Section 3 discusses some of the reasons why network administrators and designers may wish to utilise MCSs to achieve their intra-cluster multicasting instead. The Class II MARS includes all the functionality of the Class I, but modifies its use of various MARS messages to fool endpoints into using MCSs where needed.

The additional MARS messages supported by a Class II MARS are primarily associated with interaction between the MARS and the MCSs.

```
13 MARS_MSERV
17 MARS_UNSERV
18 MARS_SJOIN
19 MARS_SLEAVE
```
The following MARS messages are treated in a slightly different manner:

11   MARS_REQUEST  
14   MARS_JOIN  
15   MARS_LEAVE

A Class II MARS must keep two sets of mappings for each layer 3 group using MCS support. The original (layer 3 address, ATM.1, ATM.2, ... ATM.n) mapping (now termed the ‘host map’, although it includes routers) is augmented by a parallel (layer 3 address, server.1, server.2, .... server.K) mapping (the ‘server map’). It is assumed that no ATM addresses appear in both the server and host maps for the same multicast group. Typically K will be 1, but it will be larger if multiple MCSs are configured to support a given group.

The MARS also maintains a point to multipoint VC out to any MCSs registered with it, called ServerControlVC (section 6.2.3). This serves an analogous role to ClusterControlVC, allowing the MARS to update the MCSs with group membership changes as they occur. A Class II MARS MUST also send its regular MARS_REDIRECT_MAP transmissions on both ServerControlVC and ClusterControlVC.

6.2.1 Class II MARS response to a MARS_REQUEST.

When the MARS receives a MARS_REQUEST for an address that has both host and server maps it generates a response based on the identity of the request’s source. If the requestor is a member of the server map for the requested group then the MARS returns the contents of the host map in a sequence of one or more MARS_MULTIs. Otherwise the MARS returns the contents of the server map in a sequence of one or more MARS_MULTIs.

Servers use the host map to establish a basic distribution VC for the group. Cluster members will establish outgoing multipoint VCs to members of the group’s server map, without being aware that their packets will not be going directly to the multicast group’s members.

6.2.2 MARS_MSERV and MARS_UNSERV messages.

MARS_MSERV and MARS_UNSERV are identical to the MARS_JOIN message. An MCS uses a MARS_MSERV with a <min,max> pair of <X,X> to specify the multicast group X that it is willing to support. A single group MARS_UNSERV indicates the group that the MCS is no longer willing to support. The operation code for MARS_MSERV is 13 (decimal), and MARS_UNSERV is 17 (decimal).

When an MCS issues a MARS_MSERV the MARS adds the new ATM address to
the server map for the specified group, possibly constructing a new
server map if this is the first MCS for the group.

When an MCS issues a MARS_UNSERV the MARS removes its ATM address
from the server maps for each specified group, deleting any server
maps that end up being null after the operation.

Both of these messages are sent to the MARS over a point to point VC
(between MCS and MARS). After processing, they are retransmitted on
ServerControlVC to allow other MCSs to note the new node.

The operation code is then changed to MARS_JOIN or MARS_LEAVE
respectively, and another copy of the message is also transmitted on
ClusterControlVC. This fools the cluster members into thinking a new
leaf node as been added to (or dropped from) the group specified. The
ar$layer3grp flag MUST be reset for the retransmitted
MARS_JOIN/LEAVE.

The MARS retransmits but otherwise ignores redundant MARS_MSERV and
MARS_UNSERV messages.

It is assumed that at least one MCS will have MARS_MSERV’ed a group
before the first cluster member joins it. If a MARS_MSERV arrives for
a group that has a non-null host map but no server map the default
response of the MARS will be to silently drop the MARS_MSERV without
any further action. The MCS attempting to support the group will
eventually flag an error after repeated MARS_MSERVs fail.

The last or only MCS for a group MAY choose to issue a MARS_UNSERV
while the group still has members. When the MARS_UNSERV is processed
by the MARS the ‘server map’ will be deleted. When the associated
MARS_LEAVE is issued on ClusterControlVC, all cluster members with a
VC open to the MCS for that group will close down the VC (in
accordance with section 5.1.4, since the MCS was their only leaf
node). When cluster members subsequently find they need to transmit
packets to the group, they will begin again with the
MARS_REQUEST/MARS_MULTI sequence to establish a new VC. Since the
MARS will have deleted the server map, this will result in the host
map being return, and the group reverts to being supported by a VC
mesh.

A clean mechanism for the reverse process - transitioning a group
from a VC mesh to MCS supported while the group is active - is a
subject for further study.
6.2.3 Registering a Multicast Server (MCS).

Section 5.2.3 describes how endpoints register as cluster members, and hence get added as leaf nodes to ClusterControlVC. The same approach is used to register endpoints that intend to provide MCS support to a Class II MARS.

Registration with the MARS occurs when an endpoint issues a MARS_MSERV for a protocol specific multicast group address. Upon registration the endpoint is added as a leaf node to ServerControlVC.

In IPv4 environments an MCS endpoint MUST explicitly issue a MARS_MSERV for the special address "0.0.0.0" in order to register with the MARS. In other words, a MARS_MSERV with ar$tpin of 4, and 8 bytes of zero starting at ar$min.1 (equivalent to the block of <0.0.0.0,0.0.0.0>.

The specific addresses signifying ‘registration’ for other layer 3 protocols will defined in subsequent documents.

The MCS retransmits this MARS_MSERV until it confirms that the MARS has received it (by receiving a copy back, in an analogous way to the mechanism described in section 5.2.2 for reliably transmitting MARS_JOINs).

The ar$cmi field in MARS_MSERVs are set to zero by both MCS and MARS.

An MCS may also choose to de-register, using a MARS_UNSERV. In an IPv4 environment a MARS_UNSERV on the special address of "0.0.0.0" would result in the MARS dropping the MCS from ServerControlVC.

Note that multiple logical MCSs may share the same physical ATM interface, provided that each MCS uses a separate ATM address (e.g. a different SEL field in the NSAP format address). In fact, an MCS may share the ATM interface of a node that is also a cluster member (either host or router), provided each logical entity has a different ATM address.

6.2.4 Class II response to MARS_JOIN and MARS_LEAVE.

The existence of MCSs supporting some groups but not others requires the Class II MARS to modify its distribution of single and block join/leave updates to cluster members. The Class II MARS also adds two new messages - MARS_SJOIN and MARS_SLEAVE - for communicating group changes to MCSs over ServerControlVC.

The MARS_SJOIN and MARS_SLEAVE messages are identical to MARS_JOIN, with operation codes 18 and 19 (decimal) respectively.
When a cluster member issues MARS_JOIN or MARS_LEAVE for a single group, the MARS checks to see if the group has an associated server map. If the specified group does not have a server map the MARS provides a Class I service and simply retransmits the MARS_JOIN or MARS_LEAVE on ClusterControlVC.

However, if a server map exists for the group a new set of actions are taken.

A copy of the MARS_JOIN/LEAVE is made with type MARS_SJOIN or MARS_SLEAVE as appropriate, and transmitted on ServerControlVC. This allows the MCS(s) supporting the group to note the new member and update their data VCs.

The original message’s ar$pnum field is set to 0, and it is transmitted back using the VC it arrived on (rather than ClusterControlVC).

(Section 5.2.2 requires cluster members have a mechanism to confirm the reception of their message by the MARS. For mesh supported groups, using ClusterControlVC serves dual purpose of providing this confirmation and distributing group update information. When a group is MCS supported, there is no reason for all cluster members to process null join/leave messages on ClusterControlVC, so they are sent back on the private VC between cluster member and MARS.)

Receipt of a block MARS_JOIN (e.g. from a router coming on-line) or MARS_LEAVE requires a more complex response. The single <min,max> block may simultaneously cover VC mesh supported and MCS supported groups. However, cluster members only need to be informed of the VC mesh supported groups that the endpoint has joined. Only the MCSs need to know if the endpoint is joining any MCS supported groups.

The solution is to modify the MARS_JOIN or MARS_LEAVE that is retransmitted on ClusterControlVC. The following action is taken:

A copy of the MARS_JOIN/LEAVE is made with type MARS_SJOIN or MARS_SLEAVE as appropriate, and transmitted on ServerControlVC. This allows the MCS(s) supporting the group to note the membership change and update their outgoing point to multipoint VCs.

The <min,max> block supplied in the original MARS_JOIN/LEAVE is replaced with a ‘hole punched’ set of zero or more <min,max> pairs. The ‘hole punched’ set of <min,max> pairs covers the entire address range specified by the original <min,max> pair, but excludes those addresses/groups supported by MCSs.

If the hole-punched set contains 1 or more <min,max> pair, the
MARS_JOIN/LEAVE is transmitted on ClusterControlVC.

If the hole-punched set is empty, the ar$pnum field is set to zero, and the MARS_JOIN/LEAVE is transmitted back using the VC it arrived on (rather than ClusterControlVC).

(Appendix A discusses some algorithms for ‘hole punching’.)

It is assumed that MCSs use the MARS_SJOINs and MARS_SLEAVEs to update their own VCs out to the actual group’s members.

The ar$layer3grp flag is copied over into the messages transmitted by the MARS.

**6.2.5 Sequence numbers for ServerControlVC traffic.**

In an analogous fashion to the Cluster Sequence Number, a Class II MARS keeps a Server Sequence Number (SSN) that is incremented for every transmission on ServerControlVC. The current value of the SSN is inserted into the ar$msn field of every message the MARS issues that are being returned in response to a MARS_REQUEST from an MCS, and MARS_REDIRECT_MAP being sent on ServerControlVC. The MCS must check the MARS_REQUESTs source, and if it is a registered MCS the SSN is copied into the ar$msn field, otherwise the CSN is copied into the ar$msn field.

MCSs are expected to track and use the SSNs in an analogous manner to the way endpoints use the CSN in section 5.1 (to trigger revalidation of group membership information).

A Class II MARS should be carefully designed to minimise the possibility of the SSN jumping unnecessarily. Under normal operation only MCSs that are affected by transient link problems will miss ar$msn updates and be forced to revalidate. If the MARS itself glitches it will be innundated with requests for a period as every MCS attempts to revalidate.

**6.3 Why global sequence numbers?**

The CSN and SSN are global within the context of a given protocol (e.g. IP). They count ClusterControlVC and ServerControlVC activity without reference to the multicast group(s) involved. This may be perceived as a limitation, because there is no way for cluster members or multicast servers to isolate exactly which multicast group they may have missed an update for. An alternative was to try and provide a per-group sequence number.
Unfortunately per-group sequence numbers are not practical. The current mechanism allows sequence information to be piggy-backed onto MARS messages already in transit for other reasons. The ability to specify blocks of multicast addresses with a single MARS_JOIN or MARS_LEAVE means that a single message can refer to membership change for multiple groups simultaneously. A single ar$msn field cannot provide meaningful information about each group’s sequence. Multiple ar$msn fields would have been unwieldy.

Any MARS or cluster member that supports different protocols MUST keep separate mapping tables and sequence numbers for each protocol.

6.4 Redundant/Backup MARS Architectures.

If backup MARSs exist for a given cluster then mechanisms are needed to ensure consistency between their mapping tables and those of the active, current MARS.

(Cluster members will consider backup MARSs to exist if they have been configured with a table of MARS addresses, or the regular MARS_REDIRECT_MAP messages contain a list of 2 or more addresses.)

The definition of an MARS-synchronization protocol is beyond the current scope of this document, and is expected to be the subject of further research work. However, the following observations may be made:

The MARS_REDIRECT_MAP message exist enable one MARS to force endpoints to move to another MARS (e.g. in the aftermath of a MARS failure, the chosen backup MARS will eventually wish to hand control of the cluster over to the main MARS when it is functioning properly again).

Cluster members and MCSs do not need to start up with knowledge of more than one MARS, provided that MARS correctly issues MARS_REDIRECT_MAP messages with the full list of MARSs for that cluster.

Any mechanism for synchronising backup MARSs (and coping with the aftermath of MARS failures) should not require the endpoint behaviour to be modified from what is described in this specification.

7. How an MCS utilises a Class II MARS.

Along the data path the MCS is a protocol independent entity, in that its role is to accept AAL_SDUs from multiple sources and then transmit them sequentially out a single point to multipoint VC. It does not look inside the AAL_SDUs at all. However, when an MCS starts
up it must register with the MARS as described in section 6.2.3. This requires it to register for a particular protocol (specified in the ar$pro field of the MARS_MSERV).

Each MCS MUST terminate unidirectional VCs in the same manner as a cluster member would (e.g. terminate on an LLC entity when LLC/SNAP encapsulation is used, as described in RFC 1755 for unicast endpoints). This is because the MCS is acting as a surrogate cluster endpoint for the senders to the group.

The MCS manages its outgoing point to multipoint VC in an analogous way to a cluster member (as described in section 5.1). MARS_REQUEST is used by the MCS to establish the initial leaf nodes for the MCS’s outgoing point to multipoint VC. After the VC is established, the MCS reacts to MARS_SJOINS and MARS_SLEAVEs in the same way a cluster member reacts to MARS_JOINS and MARS_LEAVEs.

The MCS tracks the Server Sequence Number from the ar$msn fields of messages from the MARS, and revalidates its outgoing point to multipoint VC(s) when a sequence number jump occurs.

The MCS uses the same approach to backup MARSs as a cluster member, and tracks MARS_REDIRECT_MAP messages on ServerControlVC in an analogous manner to cluster members (as described in section 5.4).

An MCS MUST NOT share the same ATM address as a cluster member, although it may share the same physical ATM interface.

8. Support for IP multicast routers.

Multicast routers are required for the propagation of multicast traffic beyond the constraints of a single cluster (inter-cluster traffic). (There is a sense in which they are multicast servers acting at the next higher layer, with clusters, rather than individual endpoints, as their abstract sources and destinations.)

Multicast routers typically participate in higher layer multicast routing algorithms and policies that are beyond the scope of this memo (e.g. DVMRP [5] in the IPv4 environment).

It is assumed that the multicast routers will be implemented over the same sort of IP/ATM interface that a multicast host would use. Their IP/ATM interfaces will will register with the MARS as a cluster members, joining and leaving multicast groups as necessary. As noted in section 5, multiple logical ‘endpoints’ may be implemented over a single physical ATM interface. Routers use this approach to provide interfaces into each clusters they will be routing between.
The rest of this section will assume a simple IPv4 scenario where the scope of a cluster has been limited to a particular LIS that is part of an overlaid IP network. Not all members of the LIS are necessarily registered cluster members (you may have unicast-only hosts in the LIS).

8.1 Forwarding into a Cluster.

If the multicast router needs to transmit a packet to a group within the cluster its IP/ATM interface opens a VC in the same manner as a normal host would. Once a VC is open, the router watches for MARS_JOIN and MARS_LEAVE messages and responds to them as a normal host would.

The multicast router’s transmit side MUST implement inactivity timers to shut down idle outgoing VCs, as for normal hosts.

As with normal host, the multicast router does not need to be a member of a group it is sending to.

8.2 Joining in ‘promiscuous’ mode.

Once registered and initialised, the simplest model of IPv4 multicast router operation is for it to issue a MARS_JOIN encompassing the entire Class D address space. In effect it becomes ‘promiscuous’, as it will be a leaf node to all present and future multipoint VCs established to IPv4 groups on the cluster.

How a router chooses which groups to propagate outside the cluster is beyond the scope of this memo.

Consistent with RFC 1112, IP multicast routers may retain the use of IGMP Query and IGMP Report messages to ascertain group membership. However, certain optimisations are possible, and are described in section 8.5.

8.3 Forwarding across the cluster.

Under some circumstances the cluster may simply be another hop between IP subnets that have participants in a multicast group.


LAN.1 and LAN.2 are subnets (such as Ethernet) with attached hosts that are members of group X.

IPmcR.1 and IPmcR.2 are multicast routers with interfaces to the LIS.
A traditional solution would be to treat the LIS as a unicast subnet, and use tunneling routers. However, this would not allow hosts on the LIS to participate in the cross-LIS traffic.

Assume IPmcR.1 is receiving packets promiscuously on its LAN.1 interface. Assume further it is configured to propagate multicast traffic to all attached interfaces. In this case that means the LIS.

When a packet for group X arrives on its LAN.1 interface, IPmcR.1 simply sends the packet to group X on the LIS interface as a normal host would (Issuing MARS_REQUEST for group X, creating the VC, sending the packet).

Assuming IPmcR.2 initialised itself with the MARS as a member of the entire Class D space, it will have been returned as a member of X even if no other nodes on the LIS were members. All packets for group X received on IPmcR.2’s LIS interface may be retransmitted on LAN.2.

If IPmcR.1 is similarly initialised the reverse process will apply for multicast traffic from LAN.2 to LAN.1, for any multicast group. The benefit of this scenario is that cluster members within the LIS may also join and leave group X at anytime.

8.4 Joining in ‘semi-promiscous’ mode.

Both unicast and multicast IP routers have a common problem – limitations on the number of AAL contexts available at their ATM interfaces. Being ‘promiscuous’ in the RFC 1112 sense means that for every M hosts sending to N groups, a multicast router’s ATM interface will have M*N incoming reassembly engines tied up.

It is not hard to envisage situations where a number of multicast groups are active within the LIS but are not required to be propagated beyond the LIS itself. An example might be a distributed simulation system specifically designed to use the high speed IP/ATM environment. There may be no practical way its traffic could be utilised on ‘the other side’ of the multicast router, yet under the conventional scheme the router would have to be a leaf to each participating host anyway.

As this problem occurs at the link layer, it is worth noting that ‘scoping’ mechanisms at the IP multicast routing level do not provide a solution. An IP level scope would still result in the router’s ATM interface receiving traffic on the scoped groups, only to drop it.

In this situation the network administrator might configure their multicast routers to exclude sections of the Class D address space when issuing MARS_JOIN(s). Multicast groups that will never be
propagated beyond the cluster will not have the router listed as a member, and the router will never have to receive (and simply ignore) traffic from those groups.

Another scenario involves the product $M \times N$ exceeding the capacity of a single router’s interface (especially if the same interface must also support a unicast IP router service).

A network administrator may choose to add a second node, to function as a parallel IP multicast router. Each router would be configured to be ‘promiscuous’ over separate parts of the Class D address space, thus exposing themselves to only part of the VC load. This sharing would be completely transparent to IP hosts within the LIS.

Restricted promiscuous mode does not break RFC 1112’s use of IGMP Report messages. If the router is configured to serve a given block of Class D addresses, it will receive the IGMP Report. If the router is not configured to support a given block, then the existence of an IGMP Report for a group in that block is irrelevant to the router. All routers are able to track membership changes through the MARS_JOIN and MARS_LEAVE traffic anyway. (Section 8.5 discusses a better alternative to IGMP within a cluster.)

Mechanisms and reasons for establishing these modes of operation are beyond the scope of this memo.

8.5 An alternative to IGMP Queries.

An unfortunate aspect of IGMP is that it assumes multicasting of IP packets is a cheap and trivial event at the link layer. As a consequence, regular IGMP Queries are multicasted by routers to group 224.0.0.1. These queries are intended to trigger IGMP Replies by cluster members that have layer 3 members of particular groups.

However, the MARS_GROUPLIST_REQUEST and MARS_GROUPLIST_REPLY messages were designed to allow routers to avoid actually transmitting IGMP Queries out into a cluster.

Whenever the router’s forwarding engine wishes to transmit an IGMP query, a MARS_GROUPLIST_REQUEST can be sent to the MARS instead. The resulting MARS_GROUPLIST_REPLY(s) (described in section 5.3) from the MARS carry all the information that the router would have ascertained from IGMP replies.

It is RECOMMENDED that multicast routers utilise this MARS service to minimise IGMP traffic within the cluster.

By default a MARS_GROUPLIST_REQUEST SHOULD specify the entire address
space (e.g. <224.0.0.0, 239.255.255.255> in an IPv4 environment). However, routers serving part of the address space (as described in section 8.4) MAY choose to issue MARS_GROUPLIST_REQUESTs that specify only the subset of the address space they are serving.

(On the surface it would also seem useful for multicast routers to track MARS_JOINs and MARS_LEAVEs that arrive with the ar$layer3grp flag set. These might be used in lieu of IGMP Reports, to provide the router with timely indication that a new layer 3 group member exists within the cluster. However, this only works on VC mesh supported groups, and is therefore NOT recommended).

Appendix B discusses less elegant mechanisms for reducing the impact of IGMP traffic within a cluster, on the assumption that the IP/ATM interfaces to the cluster are being used by un-optimised IP multicasting code.

9. Multiprotocol applications of the MARS and MARS clients.

A deliberate attempt has been made to describe the MARS and associated mechanisms in a manner independent of a specific higher layer protocol being run over the ATM cloud. The immediate application of this document will be in an IPv4 environment, and this is reflected by the focus of key examples. However, the coding of each MARS message means that any higher layer protocol identifiable by a two byte Ethernet Type code can be supported by a MARS.

The 16 bit ‘Protocol type’ (ar$pro) at the start of each MARS message is taken from the set of Ethernet Type codes. Every MARS MUST implement entirely separate logical mapping tables and support. Every cluster member must interpret messages from the MARS in the context of the protocol type that the MARS message refers to.

The LLC/SNAP encapsulations described in section 5 similarly allow multiple protocols to be identified by the use of different values in appropriate encapsulation fields.

10. Key Decisions and open issues.

The key decisions this memo proposes:

A Multicast Address Resolution Server (MARS) is proposed to coordinate and distribute mappings of ATM endpoint addresses to arbitrary higher layer ‘multicast group addresses’. The specific case of IPv4 multicast is used as the example.

The concept of ‘clusters’ is introduced to define the scope of a MARS’s responsibility, and the set of ATM endpoints willing to
participate in link level multicasting.

A Class I MARS is described, with the necessary functionality to support intra-cluster multicasting using VC meshes. A Class II MARS is described as a superset of the Class I, with additional functionality required to support intra-cluster multicasting using either VC meshes or ATM level multicast servers.

MARS message formats and encapsulation allow co-resident MARS and ATM ARP Server implementations.

New message types: MARS_JOIN, MARS_LEAVE, MARS_REQUEST. Allow endpoints to join, leave, and request the current membership list of multicast groups.

New message type: MARS_MULTI. Allows multiple ATM addresses to be returned by the MARS in response to a MARS_REQUEST.

New message types: MARS_MSERV, MARS_UNSERV. Allow multicast servers to register and deregister themselves with the MARS.

New message types: MARS_SJOIN, MARS_SLEAVE. Allow MARS to pass on group membership changes to multicast servers.

New message types: MARS_GROUPLIST_REQUEST, MARS_GROUPLIST_REPLY. Allow MARS to indicate which groups have actual layer 3 members. May be used to support IGMP in IPv4 environments, and similar functions in other environments.

New message type: MARS_REDIRECT_MAP. Allow MARS to specify a set of backup MARS addresses.

‘wild card’ MARS mapping table entries possible, where a single ATM address is simultaneously associated with blocks of multicast group addresses.

The complete set of messages, and ar$op values, is:

11   MARS_REQUEST
12   MARS_MULTI
13   MARS_MSERV
14   MARS_JOIN
15   MARS_LEAVE
16   MARS_NAK
17   MARS_UNSERV
18   MARS_SJOIN
19   MARS_SLEAVE
20   MARS_GROUPLIST_REQUEST
A number of issues are left open at this stage, and are likely to be the subject of on-going research and additional documents that build upon this one.

The specified endpoint behaviour allows the use of redundant/backup MARSs within a cluster. However, no specifications yet exist on how these MARSs co-ordinate amongst themselves. (The default is to only have one MARS per cluster.)

The specified endpoint behaviour and Class II MARS service allows the use of multiple MCSs per group. However, no specifications yet exist on how this may be used, or how these MCSs co-ordinate amongst themselves. (The default is to only have one MCS per group.)

The MARS relies on the cluster member dropping off ClusterControlVC if the cluster member dies. It is not clear if additional mechanisms are needed to detect and delete ‘dead’ cluster members.

If a multicast server attempts to MARS_MSERV for an existing VC mesh supported group, it would be nice to have current senders to the group migrate their outgoing VCs from the actual cluster member leaf nodes to the newly registered multicast server(s). How this might be achieved, the load this would place on the MARS, and its scalability, have not yet been considered.

Supporting layer 3 ‘broadcast’ as a special case of multicasting (where the ‘group’ encompasses all cluster members) has not been explicitly discussed.

Supporting layer 3 ‘unicast’ as a special case of multicasting (where the ‘group’ is a single cluster member, identified by the cluster member’s unicast protocol address) has not been explicitly discussed.

The future development of ATM Group Addresses and Leaf Initiated Join to ATM Forum’s UNI specification has not been addressed. (However, the problems identified in this memo with respect to VC scarcity and impact on AAL contexts will not be fixed by such developments in the signalling protocol.)

Security Consideration
Security considerations are not addressed in this memo.

Acknowledgments

The discussions within the IP over ATM Working Group have helped clarify the ideas expressed in this document. John Moy (Cascade Communications Corp.) initially suggested the idea of wild-card entries in the ARP Server. Drew Perkins (Fore Systems) provided rigorous and useful critique of early proposed mechanisms for distributing and validating group membership information. Susan Symington (and co-workers at MITRE Corp., Don Chirieleison, Rich Verjinski, and Bill Barns) clearly articulated the need for multicast server support, proposed a solution, and challenged earlier block join/leave mechanisms. John Shirron (Fore Systems) provided useful improvements on my original revalidation procedures. Susan Symington and Bryan Gleeson (Adaptec) independently championed the need for the service provided by MARS_GROUPLIST_REQUEST/REPLY.

Author’s Address

Grenville Armitage
MRE 2P340, 445 South Street
Morristown, NJ, 07960
USA

Email: gja@thumper.bellcore.com

References


Appendix A.  Hole punching algorithms for Class II MARS messages.

Implementations are entirely free to comply with the body of this memo in any way they see fit. This appendix is purely for clarification.

A Class II MARS implementation might pre-construct a set of <min,max> pairs (P) that reflects the entire Class D space, excluding any addresses currently supported by multicast servers. The <min> field of the first pair MUST be 224.0.0.0, and the <max> field of the last pair must be 239.255.255.255. The first and last pair may be the same. This set is updated whenever a multicast server registers or deregisters.

When the MARS must perform 'hole punching' it might consider the following algorithm:

Assume the MARS_JOIN/LEAVE received by the MARS from the cluster member specified the block <Emin, Emax>.

Assume Pmin(N) and Pmax(N) are the <min> and <max> fields from the Nth pair in the MARS's current set P.

Assume set P has K pairs. Pmin(1) MUST equal 224.0.0.0, and Pmax(M) MUST equal 239.255.255.255. (If K == 1 then no hole punching is required).

Execute pseudo-code:

create copy of set P, call it set C.

index1 = 1;
while (Pmax(index1) <= Emin)
    index1++;

index2 = K;
while (Pmin(index2) >= Emax)
    index2--;

if (index1 > index2)
    Exit, as the hole-punched set is null.

if (Pmin(index1) < Emin)
    Cmin(index1) = Emin;

if (Pmax(index2) > Emax)
    Cmax(index2) = Emax;
Set C is the required ‘hole punched’ set of address blocks.

The resulting set C retains all the MARS’s pre-constructed ‘holes’ covering the multicast servers, but will have been pruned to cover the section of the Class D space specified by the originating host’s <Emin,Emax> values.

The host end should keep a table, H, of open VCs in ascending order of Class D address.

Assume H(x).addr is the Class address associated with VC.x.
Assume H(x).addr < H(x+1).addr.

The pseudo code for updating VCs based on an incoming JOIN/LEAVE might be:

\[
x = 1;
N = 1;
\]

while (x < no.of VCs open)
{
    while (H(x).addr > max(N))
    {
        N++;
        if (N > no. of pairs in JOIN/LEAVE)
            return(0);
    }

    if ((H(x).addr <= max(N) &&
        ((H(x).addr >= min(N))
            perform_VC_update();
        x++;
    )}
Appendix B. Minimising the impact of IGMP in IPv4 environments.

Implementing any part of this appendix is not required for conformance with this memo. It is provided solely to document issues that have been identified.

The intent of section 5.1 is for cluster members to only have outgoing point to multipoint VCs when they are actually sending data to a particular multicast group. However, in most IPv4 environments the multicast routers attached to a cluster will periodically issue IGMP Queries to ascertain if particular groups have members. The current IGMP specification attempts to avoid having every group member respond by insisting that each group member wait a random period, and responding if no other member has responded before them. The IGMP reply is sent to the multicast address of the group being queried.

Unfortunately, as it stands the IGMP algorithm will be a nuisance for cluster members that are essentially passive receivers within a given multicast group. It is just as likely that a passive member, with no outgoing VC already established to the group, will decide to send an IGMP reply - causing a VC to be established were there was no need for one. This is not a fatal problem for small clusters, but will seriously impact on the ability of a cluster to scale.

The most obvious solution is for routers to use the MARS_GROUPLIST_REQUEST and MARS_GROUPLIST_REPLY messages, as described in section 8.5. This would remove the regular IGMP Queries, resulting in cluster members only sending an IGMP Report when they first join a group.

Alternative solutions do exist. One would be to modify the IGMP reply algorithm, for example:

- If the group member has VC open to the group proceed as per RFC 1112 (picking a random reply delay between 0 and 10 seconds).
- If the group member does not have VC already open to the group, pick random reply delay between 10 and 20 seconds instead, and then proceed as per RFC 1112.

If even one group member is sending to the group at the time the IGMP Query is issued then all the passive receivers will find the IGMP Reply has been transmitted before their delay expires, so no new VC is required. If all group members are passive at the time of the IGMP Query then a response will eventually arrive, but 10 seconds later than under conventional circumstances.
The preceeding solution requires re-writing existing IGMP code, and implies the ability of the IGMP entity to ascertain the status of VCs on the underlying ATM interface. This is not likely to be available in the short term.

One short term solution is to provide something like the preceeding functionality with a ‘hack’ at the IP/ATM driver level within cluster members. Arrange for the IP/ATM driver to snoop inside IP packets looking for IGMP traffic. If an IGMP packet is accepted for transmission, the IP/ATM driver can buffer it locally if there is no VC already active to that group. A 10 second timer is started, and if an IGMP Reply for that group is received from elsewhere on the cluster the timer is reset. If the timer expires, the IP/ATM driver then establishes a VC to the group as it would for a normal IP multicast packet.

Some network implementors may find it advantageous to configure a multicast server to support the group 224.0.0.1, rather than rely on a mesh. Given that IP multicast routers regularly send IGMP queries to this address, a mesh will mean that each router will permanently consume an AAL context within each cluster member. In clusters served by multiple routers the VC load within switches in the underlying ATM network will become a scaling problem.

Finally, if a multicast server is used to support 224.0.0.1, another ATM driver level hack becomes a possible solution to IGMP Reply traffic. The ATM driver may choose to grab all outgoing IGMP packets and send them out on the VC established for sending to 224.0.0.1, regardless of the Class D address the IGMP message was actually for. Given that all hosts and routers must be members of 224.0.0.1, the intended recipients will still receive the IGMP Replies. The negative impact is that all cluster members will receive the IGMP Replies.
Appendix C. Further comments on ‘Clusters’.

The cluster concept was introduced in section 1 for two reasons. The more well known term of Logical IP Subnet is both very IP specific, and constrained to unicast routing boundaries. As the architecture described in this document may be re-used in non-IP environments a more neutral term was needed. As the needs of multicasting are not always bound by the same scopes as unicasting, it was not immediately obvious that apriori limiting ourselves to LISs was a win situation either.

It must be stressed that Clusters are purely an administrative being. You choose their size (i.e. the number of endpoints that register with the same MARS) based on your multicasting needs, and the resource consumption you are willing to put up with. The larger the number of ATM attached hosts you require multicast support for, the more individual clusters you may choose to establish (along with multicast routers to provide inter-cluster traffic paths).

Given that not all the hosts in any given LIS may require multicast support, it becomes conceivable that you might assign a single MARS to support hosts from across multiple LISs. In effect you have a cluster covering multiple LISs, and have achieved ‘cut through’ routing for multicast traffic. Under these circumstances increasing the geographical size of a cluster might be considered a good thing.

However, practical considerations limit the size of clusters. Having a cluster span multiple LISs may not always be a particular ‘win’ situation. As the number of multicast capable hosts in your LISs increases it becomes more likely that you’ll want to constrain a cluster’s size and force multicast traffic to aggregate at multicast routers scattered across your ATM cloud. (This is especially true for clusters based on Class I MARSs, as resource consumption of VC meshes increases rapidly with an increase in the number of senders/group members.)

Finally, multi-LIS clusters require a moderate amount of care when deploying IP multicast routers. Under the Classical IP model you need unicast routers on the edges of LISs. Under the MARS architecture you only need multicast routers at the edges of clusters. If your cluster spans multiple LISs, then the multicast routers will perceive themselves to have a single interface that is simultaneously attached to multiple unicast subnets. This situation can work, but may require some hand configuration of ‘default’ multicast router behaviour, depending on the inter-domain routing protocol you are using.