                    <draft-ietf-ion-marsmcs-01.txt>
         Multicast Server Architectures for MARS-based ATM multicasting.

Status of this Memo

                                Abstract

A mechanism to support the multicast needs of layer 3 protocols in general,
and IP in particular, over UNI 3.0/3.1 based ATM networks has been
described in RFC 2022. Two basic approaches exist for the intra-subnet
(intra-cluster) multicasting of IP packets. One makes use of a mesh of
point to multipoint VCs (the 'VC Mesh' approach), while the other uses a
shared point to multipoint tree rooted on a Multicast Server (MCS). This
memo provides details on the design and implementation of an MCS, building
on the core mechanisms defined in RFC 2022. It also provides a mechanism
for using multiple MCSs per group for providing fault tolerance. This
approach can be used with RFC 2022 based MARS server and clients, without
needing any change in their functionality.

[1](#) Introduction

A solution to the problem of mapping layer 3 multicast service over the connection-oriented ATM service provided by UNI 3.0/3.1, has been presented in [GA96]. A Multicast Address Resolution Server (MARS) is used to maintain a mapping of layer 3 group addresses to ATM addresses in that architecture. It can be considered to be an extended analog of the ATM ARP Server introduced in RFC 1577 ([ML93]). Hosts in the ATM network use the MARS to resolve layer 3 multicast addresses into corresponding lists of ATM addresses of group members. Hosts keep the MARS informed when they need to join or leave a particular layer 3 group.

The MARS manages a "cluster" of ATM-attached endpoints. A "cluster" is defined as

"The set of ATM interfaces choosing to participate in direct ATM connections to achieve multicasting of AALSDUs 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.

A sender in the cluster has two options for multicasting data to the group members. It can either get the list of ATM addresses constituting the group from the MARS, set up a point-to-multipoint virtual circuit (VC) with the group members as leaves, and then proceed to send data out on it. Alternatively, the source can make use of a proxy Multicast Server (MCS). The source transmits data to such an MCS, which in turn uses a point-to-multipoint VC to get the data to the group members.

The MCS approach has been briefly introduced in [GA96]. This memo presents a detailed description of MCS architecture and proposes a simple mechanism for supporting multiple MCSs for fault tolerance. We assume an understanding of the IP multicasting over UNI 3.0/3.1 ATM network concepts described in [GA96], and access to it. This document is organized as follows. Section 2 presents interactions with the local UNI 3.0/3.1 signaling entity that are used later in the document and have been originally described in [GA96]. Section 3 presents an MCS architecture, along with a description of its interactions with the MARS. Section 4 describes the working of an MCS. The possibility of using multiple MCSs for the same layer 3 group, and the mechanism needed to support such usage, is described in section 5. A comparison of the VC Mesh approach and the MCS

approach is presented in Appendix A.

2 Interaction with the local UNI 3.0/3.1 signaling entity

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

LCALL-RQ - Establish a unicast VC to a specific endpoint.
LMULTI-RQ - Establish multicast VC to a specific endpoint.
LMULTI-ADD - Add new leaf node to previously established VC.
LMULTI-DROP - Remove specific leaf node from established VC.
LRELEASE - Release unicast VC, or all Leaves of a multicast VC.

The following indications are assumed to be available to AAL Users, generated by by the local UNI 3.0/3.1 signaling entity:

LACK - Succesful completion of a local request.
LREMOTE-CALL - A new VC has been established to the AAL User.
ERRL-RQFAILED - A remote ATM endpoint rejected an LCALLRQ,
                    LMULTIRQ, or L-MULTIADD.
ERRL-DROP - A remote ATM endpoint dropped off an existing VC.
ERRL-RELEASE - An existing VC was terminated.

3 MCS Architecture

The MCS acts as a proxy server which multicasts data received from a source to the group members in the cluster. All multicast sources transmitting to an MCS-based group send the data to the specified MCS. The MCS then forwards the data over a point to multipoint VC that it maintains to group members in the cluster. Each multicast source thus maintains a single point-to-multipoint VC to the designated MCS for the group. The designated MCS terminates one point-to-multipoint VC from each cluster member that is multicasting to the layer 3 group. Each group member is the leaf of the point-to-multipoint VC originating from the MCS.

A brief introduction to possible MCS architectures has been presented in [GA96]. The main contribution of that document concerning the MCS approach is the specification of the MARS interaction with the MCS. The next section lists control messages exchanged by the MARS and MCS.

3.1 Control Messages exchanged by the MCS and the MARS

The following control messages are exchanged by the MARS and the MCS.

operation code   Control Message

     1   MARS_REQUEST
     2   MARS_MULTI
     3   MARS_MSERV
     6   MARS_NAK
     7   MARS_UNSERV
     8   MARS_SJOIN
     9   MARS_SLEAVE
    12   MARS_REDIRECT_MAP

MARSMSERV and MARS-UNSERV are identical in format to the MARSJOIN message.
MARSSJOIN and MARS-SLEAVE are also identical in format to MARSJOIN. As
such, their formats and those of MARSREQUEST, MARS-MULTI, MARSNAK and
MARSREDIRECT-MAP are described in [GA96]. Their usage is described in
section 4. All control messages are LLC/SNAP encapsulated as described in
section 4.2 of [GA96]. (The "mar$" notation used in this document is
borrowed from [GA96], and indicates a specific field in the control
message.) Data messages are reflected without any modification by the MCS.

3.2 Association with a layer 3 group

The simplest MCS architecture involves taking incoming AALSDUs from the
multicast sources and sending them out over the point-to-multipoint VC to
the group members. The MCS can service just one layer 3 group using this
design, as it has no way of distinguishing between traffic destined for
different groups. So each layer 3 MCS-supported group will have its own
designated MCS.

However it is desirable in the interests of saving resources to utilize the
same MCS to support multiple groups. This can be done by adding minimal
layer 3 specific processing into the MCS. The MCS can now look inside the
received AALSDUs and determine which layer 3 group they are destined for.
A single instance of such an MCS could register its ATM address with the
MARS for multiple layer 3 groups, and manage multiple point-to-multipoint
VCs, one for each group. This capability is included in the MCS
architecture, as is the capability of having multiple MCSs per group

(section 5).

4 Working of MCS

An MCS MUST NOT share its ATM address with any other cluster member (MARS or otherwise). However, it may share the same physical ATM interface (even with other MCSs or the MARS), provided that each logical entity has a different ATM address. This section describes the working of MCS and its interactions with the MARS and other cluster members.

4.1 Usage of MARSMSERV and MARS-UNSERV

4.1.1 Registration (and deregistration) with the MARS

The ATM address of the MARS MUST be known to the MCS by out-of-band means at startup. One possible approach for doing this is for the network administrator to specify the MARS address at command line while invoking the MCS. On startup, the MCS MUST open a point-to-point control VC (MARSVC) with the MARS. All traffic from the MCS to the MARS MUST be carried over the MARSVC. The MCS MUST register with the MARS using the MARS-MSERV message on startup. To register, a MARSMSERV MUST be sent by the MCS to the MARS over the MARSVC. On receiving this MARS-MSERV, the MARS adds the MCS to the ServerControlVC. The ServerControlVC is maintained by the MARS with all MCSs as leaves, and is used to disseminate general control messages to all the MCSs. The MCS MUST terminate this VC, and MUST expect a copy of the MCS registration MARSMSERV on the MARS-VC from the MARS.

An MCS can deregister by sending a MARSUNSERV to the MARS. A copy of this MARSUNSERV MUST be expected back from the MARS. The MCS will then be dropped from the ServerControlVC.

No protocol specific group addresses are included in MCS registration MARSMSERV and MARS-UNSERV. The mar$flags.register bit MUST be set, the mar$cmi field MUST be set to zero, the mar$flags.sequence field MUST be set to zero, the source ATM address MUST be included and a null source protocol address MAY be specified in these MARSMSERV and MARS-UNSERV. All other fields are set as described in section 5.2.1 of [GA96] (the MCS can be considered to be a cluster member while reading that section). It MUST keep retransmitting (section 4.1.3) the MARSMSERV/MARS-UNSERV over the MARSVC until it receives a copy back.

In case of failure to open the MARSVC, or error on it, the reconnection

procedure outlined in section 4.5.2 is to be followed.

4.1.2 Registration (and deregistration) of layer 3 groups

The MCS can register with the MARS to support particular group(s). To
register a group X, a MARSMSERV with a <min, max> pair of <X, X> MUST be
sent to the MARS. The MCS MUST expect a copy of the MARSMSERV back from the
MARS. The retransmission strategy outlined in section 4.1.3 is to be
followed if no copy is received. Multiple groups can be supported by
sending a separate MARSMSERV for each group.

The MCS MUST similarly use MARSUNSERV if it wants to withdraw support for a
specific layer 3 group. A copy of the group MARSUNSERV MUST be received,
failing which the retransmission strategy in section 4.1.3 is to be
followed.

The mar$flags.register bit MUST be reset and the mar$flags.sequence field
MUST be set to zero in the group MARSMSERV and MARS-UNSERV. All other
fields are set as described in section 5.2.1 of [GA96] (the MCS can be
considered to be a cluster member when reading that section).

4.1.3 Retransmission of MARSMSERV and MARS-UNSERV

Transient problems may cause loss of control messages. The MCS needs to
retransmit MARSMSERV/MARS-UNSERV at regular intervals when it does not
receive a copy back from the MARS. This interval should be no shorter than
5 seconds, and a default value of 10 seconds is recommended. A maximum of
5 retransmissions are permitted before a failure is logged. This MUST be
considered a MARS failure, which SHOULD result in the MARS reconnection
mechanism described in section 4.5.2.

A "copy" is defined as a received message with the following fields
matching the previously transmitted MARSMSERV/MARS-UNSERV:

-   mar$op
-   mar$flags.register
-   mar$pnum
-   Source ATM address
-   first <min, max> pair

In addition, a valid copy MUST have the following field values:

- mar$flags.punched = 0
- mar$flags.copy = 1

If either of the above is not true, the message MUST be dropped without
resetting of the MARSMSERV/MARS-UNSERV timer. There MUST be only one
MARSMSERV or MARS-UNSERV outstanding at a time.

4.1.4 Processing of MARSMSERV and MARS-UNSERV

The MARS transmits copies of group MARSMSERV and MARS-UNSERV on the
ServerControlVC. So they are also received by MCSs other than the
originating one. This section discusses the processing of these messages
by the other MCSs.

If a MARSMSERV is seen that refers to a layer 3 group not supported by the
MCS, it MUST be used to track the Server Sequence Number (section 4.5.1)
and then silently dropped.

If a MARSMSERV is seen that refers to a layer 3 group supported by the MCS,
the MCS learns of the existence of another MCS supporting the same group.
This possibility is incorporated (of multiple MCSs per group) in this
version of the MCS approach and is discussed in section 5.

4.2 Usage of MARSREQUEST and MARS-MULTI

As described in section 5.1, the MCS learns at startup whether it is an
active or inactive MCS. After successful registration with the MARS, an MCS
which has been designated as inactive for a particular group MUST NOT
register to support that group with the MARS. It instead proceeds as in
section 5.4. The active MCS for a group also has to do some special
processing, which we describe in that section. The rest of section 4
describes the working of a single active MCS, with section 5 describing the
active MCSs actions for supporting multiple MCSs.

After the active MCS registers to support a layer 3 group, it uses
MARSREQUEST and MARS-MULTI to obtain information about group membership
from the MARS. These messages are also used during the revalidation phase
(section 4.5) and when no outgoing VC exists for a received layer 3 packet
(section 4.3).

On registering to support a particular layer 3 group, the MCS MUST send a
MARSREQUEST to the MARS. The mechanism to retrieve group membership and the

format of MARSREQUEST and MARS-MULTI is described in section 5.1.1 and
5.1.2 of [GA96] respectively. The MCS MUST use this mechanism for sending
(and retransmitting) the MARSREQUEST and processing the returned
MARSMULTI(/s). The MARS-MULTI MUST be received correctly, and the MCS MUST
use it to initialize its knowledge of group membership.

On successful reception of a MARSMULTI, the MCS MUST attempt to open the
outgoing point-to-multipoint VC using the mechanism described in section
5.1.3 of [GA96], if any group members exist. The MCS however MUST start
transmitting data on this VC after it has opened it successfully with at
least one of the group members as a leaf, and after it has attempted to add
all the group members at least once.

4.3 Usage of outgoing point-to-multipoint VC

Cluster members which are sources for MCS-supported layer 3 groups send
(encapsulated) layer 3 packets to the designated MCSs. An MCS, on
receiving them from cluster members, has to send them out over the specific
point-to-multipoint VC for that layer 3 group. This VC is setup as
described in the previous section. However, it is possible that no group
members currently exist, thus causing no VC to be setup. So an MCS may
have no outgoing VC to forward received layer 3 packets on, in which case
it MUST initiate the MARSREQUEST and MARS-MULTI sequence described in the
previous section. This new MARSMULTI could contain new members, whose
MARSSJOINs may have been not received by the MCS (and the loss not detected
due to absence of traffic on the ServerControlVC).

If an MCS learns that there are no group members (MARSNAK received from
MARS), it MUST delay sending out a new MARSREQUEST for that group for a
period no less than 5 seconds and no more than 10 seconds.

Layer 3 packets received from cluster members, while no outgoing
point-to-multipoint VC exists for that group, MUST be silently dropped
after following the guidelines in the previous paragraphs. This might
result in some layer 3 packets being lost until the VC is setup.

Each outgoing point-to-multipoint VC has a revalidate flag associated with
it. This flag MUST be checked whenever a layer 3 packet is sent out on
that VC. No action is taken if it is not set. If it is set, the packet is
sent out, the revalidation procedure (section 4.5.3) MUST be initiated for
this group, and the flag MUST be reset.

In case of error on a point-to-multipoint VC, the MCS MUST initiate
revalidation procedures for that VC as described in section 4.5.3.

Once a point-to-multipoint VC has been setup for a particular layer 3
group, the MCS MUST hold the VC open and mark it as the outgoing path for
any subsequent layer 3 packets being sent for that group address. A
point-to-multipoint VC MUST NOT have an activity timer associated with it.
It is to remain up at all times, unless the MCS explicitly stops supporting
that layer 3 group, or no more leaves exist on the VC which causes it to be
shut down. The VC is kept up inspite of non-existent traffic to reduce the
delay suffered by MCS supported groups. If the VC were to be shut down on
absence of traffic, the VC reestablishment procedure (needed when new
traffic for the layer 3 group appears) would further increase the initial
delay, which can be potentially higher than the VC mesh approach anyway as
two VCs need to be setup in the MCS case (one from source to MCS, second
from MCS to group) as opposed to only one (from source to group) in the VC
Mesh approach. This approach of keeping the VC from the MCS open even in
the absense of traffic is experimental. A decision either way can only be
made after gaining experience (either through implementation or simulation)
about the implications of keeping the VC open.

If the MCS supports multiple layer 3 groups, it MUST follow the procedure
outlined in the four previous subsections for each group that it is an
active MCS. Each incoming data AALSDU MUST be examined for determining its
recipient group, before being forwarded onto the appropriate outgoing
point-to-multipoint VC.

4.3.1 Group member dropping off a point-to-multipoint VC

AN ERRL-DROP may be received during the lifetime of a point-to-multipoint
VC indicating that a leaf node has terminated its participation at the ATM
level. The ATM endpoint associated with the ERRL-DROP MUST be removed from
the locally held set associated with the VC. The revalidate flag on the VC
MUST be set after a random interval of 1 through 10 seconds.

If an ERRL-RELEASE is received for a VC, then the entire set is cleared and
the VC considered to be completely shutdown. A new VC for this layer 3
group will be established only on reception of new traffic for the group
(as described in section 4.3).

4.4 Processing of MARSSJOIN and MARS-SLEAVE

The MARS transmits equivalent MARSSJOIN/MARS-SLEAVE on the ServerControlVC
when it receives MARSJOIN/MARS-LEAVE from cluster members. The MCSs keep
track of group membership updates through these messages. The format of
these messages are identical to MARSJOIN and MARS-LEAVE, which are

described in section 5.2.1 of [GA96]. It is sufficient to note here that
these messages carry the ATM address of the node joining/leaving the
group(/s), the group(/s) being joined or left, and a Server Sequence Number
from MARS.

When a MARSSJOIN is seen which refers to (or encompasses) a layer 3 group
(or groups) supported by the MCS, the following action MUST be taken. The
new member's ATM address is extracted from the MARSSJOIN. An L-MULTIADD is
issued for the new member for each of those referred groups which have an
outgoing point-to-multipoint VC. An LMULTI-RQ is issued for the new member
for each of those refered groups which have no outgoing VCs.

When a MARSSLEAVE is seen that refers to (or encompasses) a layer 3 group
(or groups) supported by the MCS, the following action MUST be taken. The
leaving member's ATM address is extracted. An LMULTI-DROP is issued for
the member for each of the refered groups which have an outgoing
point-to-multipoint VC.

There is a possibility of the above requests (LMULTI-RQ or LMULTIADD or
LMULTI-DROP) failing. The UNI 3.0/3.1 failure cause must be returned in
the ERRL-RQFAILED signal from the local signaling entity to the AAL User.
If the failure cause is not 49 (Quality of Service unavailable), 51 (user
cell rate not available - UNI 3.0), 37 (user cell rate not available - UNI
3.1), or 41 (Temporary failure), the endpoint's ATM address is dropped from
the locally held view of the group by the MCS. Otherwise, the request MUST
be re-attempted with increasing delay (initial value between 5 to 10
seconds, with delay value doubling after each attempt) until it either
succeeds or the multipoint VC is released or a MARSSLEAVE is received for
that group member. If the VC is open, traffic on the VC MUST continue
during these attempts.

MARSSJOIN and MARS-SLEAVE are processed differently if multiple MCSs share
the members of the same layer 3 group (section 5.4). MARSSJOIN and
MARSSLEAVE that do not refer to (or encompass) supported groups MUST be
used to track the Server Sequence Number (section 4.5.1), but are otherwise
ignored.

4.5 Revalidation Procedures

The MCS has to initiate revalidation procedures in case of certain failures
or errors.

4.5.1 **Server Sequence Number**

The MCS needs to track the Server Sequence Number (SSN) in the messages
received on the ServerControlVC from the MARS. It is carried in the mar$msn
of all messages (except MARSNAK) sent by the MARS to MCSs. A jump in SSN
implies that the MCS missed the previous message(/s) sent by the MARS. The
MCS then sets the revalidate flag on all outgoing point-to-multipoint VCs
after a random delay of between 1 and 10 seconds, to avoid all MCSs
inundating the MARS simultaneously in case of a more general failure.

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

Each MCS keeps its own 32 bit MCS Sequence Number (MSN) to track the SSN.
Whenever a message is received that carries a mar$msn field, the following
processing is performed:

        Seq.diff = mar$msn - MSN

        mar$msn -> MSN

        (.... process MARS message ....)

        if ((Seq.diff != 1) && (Seq.diff != 0))
              then (.... revalidate group membership information ....)

The mar$msn value in an individual MARSMULTI is not used to update the MSN
until all parts of the MARSMULTI (if > 1) have arrived. (If the mar$msn
changes during reception of a MARSMULTI series, the MARS-MULTI is discarded
as described in section 5.1.1 of [GA96]).

The MCS sets its MSN to zero on startup. It gets the current value of SSN
when it receives the copy of the registration MARSMSERV back from the MARS.

4.5.2 Reconnecting to the MARS

The MCSs are assumed to have been configured with the ATM address of at
least one MARS at startup. MCSs MAY choose to maintain a table of ATM
addresses, each address representing alternative MARS which will be
contacted in case of failure of the previous one. This table is assumed to
be ordered in descending order of preference.

An MCS will decide that it has problems communicating with a MARS if:

   * It fails to establish a point-to-point VC with the MARS.

   * MARSREQUEST generates no response (no MARSMULTI or MARS-NAK returned).

   * ServerControlVC fails.

   * MARSMSERV or MARSUNSERV do not result in their respective copies being
     received.

(reconnection as in section 5.4 in [GA96], with MCS-specific actions used
where needed).

4.5.3 Revalidating a point-to-multipoint VC

The revalidation flag associated with a point-to-multipoint VC is checked
when a layer 3 packet is to be sent out on the VC. Revalidation procedures
MUST be initiated for a point-to-multipoint VC that has its revalidate flag
set when a layer 3 packet is being sent out on it. Thus more active groups
get revalidated faster than less active ones. The revalidation process
MUST NOT result in disruption of normal traffic on the VC being
revalidated.

The revalidation procedure is as follows. The MCS reissues a MARSREQUEST
for the VC being revalidated. The returned set of members is compared with
the locally held set; LMULTI-ADDs MUST be issued for new members, and
LMULTI-DROPs MUST be issued for non-existent ones. The revalidate flag
MUST be reset for the VC.

5 Multiple MCSs for a layer 3 group

Having a single MCS for a layer 3 group can cause it to become a single
point of failure and a bottleneck for groups with large numbers of active
senders. It is thus desirable to introduce a level of fault tolerance by
having multiple MCS per group. Support for load sharing is not introduced
in this version of the draft so as to reduce the complexity of the
protocol.

5.1 Outline

The protocol described in this draft offers fault tolerance by using
multiple MCSs for the same group. This is achieved by having a standby MCS
take over from a failed MCS which had been supporting the group. The MCS
currently supporting a group is refered to as the active MCS, while the one
or more standby MCSs are refered to as inactive MCSs. There is only one
active MCS existing at any given instant for an MCS-supported group. The
protocol makes use of the HELLO messages as described in [LA96].

To reduce the complexity of the protocol, the following operational
guidelines need to be followed. These guidelines need to be enforced by
out-of-band means which are not specified in this document and can be
implementation dependent.

  * The set of (one or more) MCSs (``mcslist'') that support a particular
    IP Multicast group is predetermined and fixed. This set MUST be known
    to each MCS in the set at startup, and the ordering of MCSs in the set
    is the same for all MCSs in the set. An implementation of this would
    be to maintain the set of ATM addresses of the MCSs in a file, an
    identical copy of which is kept at each MCS in the set.

  * All MCSs in ``mcslist'' have to be started up together, with the first
    MCS in ``mcslist'' being the last to be started.

  * A failed MCS cannot be started up again.

5.2 Discussion of Multiple MCSs in operation

An MCS on startup determines its position in the ``mcslist''. If the MCS
is not the first in ``mcslist'', it does not register for supporting the
group with the MARS. If the MCS is first in the set, it does register to

support the group.

The first MCS thus becomes the active MCS and supports the group as
described in section 4. The active MCS also opens a point-to-multipoint VC
(HelloVC) to the remaining MCSs in the set (the inactive MCSs). It starts
sending HELLO messages on this VC at a fixed interval (HelloInterval
seconds). The inactive MCSs maintain a timer to keep track of the last
received HELLO message. If an inactive MCS does not receive a message
within HelloInterval* DeadFactor seconds (values of HelloInterval and
DeadFactor are the same at all the MCSs), or if the HelloVC is closed, it
assumes failure of the active MCS and attempts to elect a new one. The
election process is described in section 5.5.

If an MCS is elected as the new active one, it registers to support the
group with the MARS. It also initiates the transmission of HELLO messages
to the remaining inactive MCSs.

5.3 Inter-MCS control messages

The protocol uses HELLO messages in the heartbeat mechanism, and also
during the election process. The format of the HELLO message is based on
that described in [LA96]. The Hello message type code is 5.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Sender Len    |   Recvr Len   | State | Type  |    unused     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         HelloInterval         |           DeadFactor          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     IP Multicast address                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                Sender ATM address (variable length)          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               Receiver ATM address (variable length)         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

    Sender Len
      This field holds the length in octets of the Sender ATM address.

Recvr Len
  This field holds the length in octets of the Receiver ATM
  address.

State
  Currently two states: No-Op (0x00) and Elected (0x01).
  It is used by a candidate MCS to indicate if it was successfully
  elected.

Type
  This is the code for the message type.

HelloInterval
  The hello interval advertises the time between sending of
  consecutive Hello Messages by an active MCS. If the time between
  Hello messages exceeds the HelloInterval then the Hello is to be
  considered late by the inactive MCS.

DeadFactor
  This is a multiplier to the HelloInterval. If an inactive MCS
  does not receive a Hello message within the interval
  HelloInterval*DeadFactor from an active MCS that advertised
  the HelloInterval then the inactive MCS MUST consider the active
  one to have failed.

IP Multicast address
  This field is used to indicate the group to associate the HELLO
  message with. It is useful if MCSs can support more than one
  group.

Sender ATM address
  This is the protocol address of the server which is sending the
  Hello.

Receiver ATM address
  This is the protocol address of the server which is to Reply to
  the Hello. If the sender does not know this address then the
  sender sets it to zero. (This happens in the HELLO messages sent
  from the active MCS to the inactive ones, as they are multicast
  and not sent to one specific receiver).

5.4 The Multiple MCS protocol

As is indicated in section 5.1, all the MCSs supporting the same IP
Multicast group MUST be started up together. The set of MCSs (``mcslist'')
MUST be specified to each MCS in the set at startup. After registering to
support the group with the MARS, the first MCS in the set MUST open a
point-to-multipoint VC (HelloVC) with the remaining MCSs in the ``mcslist''
as leaves, and thus assumes the role of active MCS. It MUST send HELLO
messages HelloInterval seconds apart on this VC. The Hello message sent by
the active MCS MUST have the Receiver Len set to zero, the State field set
to "Elected", with the other fields appropriately set. The Receiver ATM
address field does not exist in this HELLO message. The initial value of
HelloInterval and DeadFactor MUST be the same at all MCSs at startup. The
active MCS can choose to change these values by introducing the new value
in the HELLO messages that are sent out. The active MCS MUST support the
group as described in section 4.

The other MCSs in ``mcslist'' determine the identity of the first MCS from
the ``mcslist''. They MUST NOT register to support the group with the
MARS, and become inactive MCSs. On startup, an inactive MCS expects HELLO
messages from the active MCS. The inactive MCS MUST terminate the HelloVC.
A timer MUST be maintained, and if the inactive MCS does not receive HELLO
message from the active one within a period HelloInterval*DeadFactor
seconds, it assumes that the active MCS died, and initiates the election
process as described in section 5.5. If a HELLO message is received within
this period, the inactive MCS does not initiate any further action, other
than restarting the timer. The inactive MCSs MUST set their values of
HelloInterval and DeadFactor to those specified by the active MCS in the
HELLO messages.

On failure of the active MCS, a new MCS assumes its role as described in
section 5.5. In this case, the remaining inactive MCSs will expect HELLO
messages from this new active MCS as described in the previous paragraph.

5.5 Failure handling

5.5.1 Failure of active MCS

The failure of the active MCS is detected by the inactive MCSs if no HELLO
message is received within an interval of HelloInterval*DeadFactor seconds,
or if the HelloVC is closed. In this case the next MCS in ``mcslist''
becomes the candidate MCS. It MUST open a point-to-multipoint VC to the
remaining inactive MCSs (HelloVC) and send a HELLO message on it with the
State field set to No-Op. The rest of the message is formatted as

described earlier.

On receiving a HELLO message from a candidate MCS, an inactive MCS MUST
open a point-to-point VC to that candidate. It MUST send a HELLO message
back to it, with the Sender and Receiver fields appropriately set (not
zero), and the State field being No-Op. If a HELLO message is received by
an inactive MCS from a non-candidate MCS, it is ignored. If no HELLO
message is received from the candidate with the State field set to
"Elected" in HelloInterval seconds, the inactive MCS MUST retransmit the
HELLO. If no HELLO message with State field set to "Elected" is received by
the inactive MCSs within an interval of HelloInterval*DeadFactor seconds,
the next MCS in ``mcslist'' is considered as the candidate MCS. Note that
the values used for HelloInterval and DeadFactor in the election phase are
the default ones.

The candidate MCS MUST wait for a period of HelloInterval*DeadFactor
seconds for receiving HELLO messages from inactive MCSs. It MUST transmit
HELLO messages with State field set to No-Op at HelloInterval seconds
interval during this period. If it receives messages from atleast half of
the remaining inactive MCSs during this period, it considers itself elected
and assumes the active MCS role. It then registers to support the group
with the MARS, and starts sending HELLO messages at HelloInterval second
intervals with State field set to "Elected" on the already existing
HelloVC. The active MCS can then alter the HelloInterval and DeadFactor
values if desired, and communicate the same to the inactive MCSs in the
HELLO message.

5.5.2 Failure of inactive MCS

If an inactive MCS drops off the HelloVC, the active MCS MUST attempt to
add that MCS back to the VC for three attempts, spaced
HelloInterval*DeadFactor seconds apart. If even the third attempt fails,
the inactive MCS is considered dead.

An MCS, active or inactive, MUST NOT be started up once it has failed.
Failed MCSs can only be started up by manual intervention after shutting
down all the MCSs, and restarting them together.

5.6 Compatibility with future MARS and MCS versions

Future versions of MCSs can be expected to use an enhanced MARS for load
sharing and fault tolerance ([TA96]). The MCS architecture described in
this document is compatible with the enhanced MARS and the future MCS
versions. This is because the active MCS is the only one which

communicates with the MARS about the group. Hence the active MCS will only
be informed by the enhanced MARS about the subset of the group that it is
to support. Thus MCSs conforming to this document are compatible with
[GA96] based MARS, as well as enhanced MARS.

6 Summary

This draft describes the architecture of an MCS. It also provides a
mechanism for using multiple MCSs per group for providing fault tolerance.
This approach can be used with [GA96] based MARS server and clients,
without needing any change in their functionality. It uses the HELLO
packet format as described in [LA96] for the heartbeat messages.

7 Acknowledgements

We would like to acknowledge Grenville Armitage (Bellcore) for reviewing
the draft and suggesting improvements towards simplifying the multiple MCS
functionalities. Discussion with Joel Halpern (Newbridge) helped clarify
the multiple MCS problem. Anthony Gallo (IBM RTP) pointed out security
issues that are not adequately addressed in the current draft.

8 Authors' Address

Rajesh Talpade - taddy@cc.gatech.edu - (404)-894-6737
Mostafa H. Ammar - ammar@cc.gatech.edu - (404)-894-3292

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

References

[GA96]   Armitage, G.J., "Support for Multicast over UNI 3.0/3.1 based ATM
          networks", RFC 2022.

[BK95]   Birman, A., Kandlur, D., Rubas, J., "An extension to the MARS
          model", Internet Draft, draft-kandlur-ipatm-mars-directvc-00.txt,
          November 1995.
[LM93]   Laubach, M., "Classical IP and ARP over ATM", RFC1577,
          Hewlett-Packard Laboratories, December 1993.

[LA96]   Luciani, J., G. Armitage, and J. Halpern, "Server Cache
          Synchronization Protocol (SCSP) - NBMA", Internet Draft,
          draft-luciani-rolc-scsp-02.txt, April 1996.

[TA96]   Talpade, R., and Ammar, M.H., "Multiple MCS support using an
          enhanced version of the MARS server.", Internet Draft (work in
          progress), draft-talpade-ion-multmcs-00.txt, June 1996.