Internet Engineering Task Force
INTERNET-DRAFT

Talpade and Ammar
Georgia Institute of Technology

June 11, 1996
Expires: December, 1996

<draft-talpade-ion-multiplemcs-00.ps>

Multiple MCS support using an enhanced version of the MARS server.

Status of this Memo

Abstract

The basic Multicast Server architecture for layer 3 multicast over ATM has been described in draft-talpade-ion-marsmcs-00.txt. It includes a mechanism for fault tolerance using multiple MCSs. However no support for sharing senders/receivers of a group across multiple MCSs has been provided. This document aims to satisfy this need by involving an enhanced version of the MARS in the load sharing and fault tolerance mechanism. This approach is an improvement over the previous one as it subverts the need for any inter-MCS synchronization mechanisms.

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. Two possible approaches exist for a source to multicast data to group members (receivers). 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 receivers as leaves, and then proceed to send data out on it (the VC Mesh approach). 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).

The MCS approach has been briefly introduced in [GA96]. The basic MCS architecture, along with MARS-MCS interactions, has been described in [TA96]. An inter-MCS synchronization protocol based on HELLO messages ([LA96]) is used to support multiple MCSs for fault tolerance. However no support is provided for using the multiple MCSs for sharing the senders/receivers of a group. [TA96] thus allows atmost one MCS to be active per group, with one or more MCSs designated as backups.

The possibility of load sharing is an important advantage of using multiple MCSs. Experiments ([TA96a] have demonstrated the bottleneck effect that a single MCS can have on data traffic, which motivates the need for sharing the senders of a group. At the same time it is crucial to minimize the synchronization mechanisms that would be necessary to achieve this goal. One common feature of mechanisms that offer load sharing is the existence of an entity which handles allocation of senders/receivers to the individual MCSs that support a group. The MARS is a repository of all relevant information about the cluster (e.g. senders/receivers of all groups, MCSs in the cluster, groups supported by each MCS, etc). It has access to all the information that an allocation entity might require, and thus blends naturally into the role. Using the MARS in this way also removes the need for any inter-MCS synchronization. This is because the allocation and consequent load sharing can take place transparent to the MCSs supporting a group. This document provides a means for supporting the use of multiple MCSs per layer 3 group by involving the MARS in the allocation mechanism.

The document currently provides an outline of the proposed mechanism, and does not include a description of the details. Familiarity of the reader with the MARS approach as described in [GA96], and the MCS architecture described in [TA96] is expected. The main difference between the approach described in [TA96] and this document is that [TA96] uses an inter-MCS synchronization mechanism to offer fault tolerance, whereas this document offers fault tolerance as well as load sharing using the MARS as the allocation mechanism and does not need any additional inter-MCS protocol.

2 Nonoptimality of using the SCSP approach for multiple MCS support

The Server Cache Synchronization Protocol (SCSP) has recently been proposed ([LA96]) for supporting the use of multiple ATMARP, MARS, NHRP servers. SCSP essentially provides a means for synchronizing the caches (databases) of the involved servers. Thus any one of multiple servers can be used transparently by clients. The functionality provided by SCSP may lead to arguments being made in favour of its use for supporting multiple MCSs also, and not involve the MARS in the multiple MCS mechanism at all. However this is not an optimal approach, as shall be pointed out in the following discussion.

The primary function of the MCS is to be a forwarding entity of data traffic, and not a repository of control information that can be accessed by clients. The opposite is true for the ATMARP, MARS and NHRP servers. The MCS gets the information it needs from the MARS and not directly from the clients. This is another major difference between the MCS and the other servers. So an instance of the other servers can receive some information from client A, with another instance receiving different information from client B but not from client A. This can lead to inconsistent caches between the two server instances, which necessitates the use of SCSP to synchronize their caches. This is not true for MCSs. All of them use the MARS for obtaining requisite information, thus getting it from a consistent source(1). So there is no need to further synchronize the MCS caches, thus obviating the need for SCSP.

Even if the MCSs were synchronized using SCSP, an additional entity would be needed to allocate new senders/receivers. This allocation entity would probably be one of the multiple MCSs. An additional mechanism will thus be needed to ensure fault tolerance, as the allocation MCS is now a single point of failure (possibly the Designated Server mechanism described in [LA96]). As opposed to this, the MARS is a repository of all the information that may be needed by an allocation entity. So it can make the necessary decisions without needing any additional entity or mechanisms.

Also, using an inter-MCS synchronization protocol like SCSP would mean that all senders would continue to transmit data to all MCSs, even when the senders are being shared. The MCSs would forward data received from senders supported by it, dropping data received from other senders. This is highly inefficient in terms of bandwidth usage and processing. Using the MARS server avoids this problem, as the MARS can selectively inform each sender about the MCS supporting it, thus making the sender transmit data to one MCS only. Thus the MARS is more efficient

(1)this is true even for multiple MARS, as is discussed in section 3.3

that using SCSP for supporting multiple MCSs.

3 Overview of the multiple MCS approach

This section provides an overview of the proposed approach. We do not describe the details in this version of the draft, but provide a conceptual introduction to the approach.

3.1 Design goals

- An important consideration of the approach is to keep the clients and the MCSs ignorant of the existence of the allocation mechanism. This simplifies their design and implementation. It also facilitates interoperability between different versions of the clients and MCSs.
- The MCS should receive and forward data from senders supported by it only. It should not receive, and then have to drop, data from unsupported senders.
- Another design goal is to minimize the complexity and fallibility of the allocation mechanism. Both the above goals are achieved by using the MARS for the allocation mechanism.
- The decision to share senders or receivers (not both) shall be made offline, and can be made on a per group basis.

3.2 Architecture

3.2.1 Additional functionality needed in the MARS

The MARS server as defined in [GA96] does not have the capability for supporting multiple MCSs. Additional functionality is needed for it to perform as the allocation mechanism. This functionality is in terms of additional processing, and does not involve any new control messages.

- The MARS should be able to allocate existing senders/receivers to the MCSs supporting a group which is transitioning from being VC Mesh based to being MCS based. This allocation should be made such that load-sharing is achieved across the MCSs.

  Thus when sharing receivers, the MCSs should be informed of the subset of receivers they support in the MARS_MULTI messages that the MARS sends in response to MARS_REQUEST. So each MCS will only be made aware of a part of the receiver set, and shall add only those receivers to its outgoing point-to-multipoint data VC.

When sharing senders, the MARS should allocate the senders such that each MCS supports a distinct subset of the sender set. This can be done by having the MARS selectively indicate the MCS that a sender should use (possibly by round-robin) in the MARS_MULTI message that the sender gets in response to a MARS_REQUEST.

- The MARS should maintain information about the current allocation map, i.e., information about which sender/receiver has been allocated to which MCS. The MARS should ensure that each sender/receiver is allocated to one and only one MCS and that all the senders/receivers have been allocated.

- When sharing receivers, the MARS should allocate new receivers to one and only one MCS. This involves selectively forwarding MARS_SJOINs to an MCS, using the point-to-point control VC that exists between each MCS and the MARS instead of the ServerControlVC. Similarly, the MARS should deallocate a client that leaves the group by forwarding the MARS_SLEAVE on the relevant point-to-point VC only.

- When sharing senders, the MARS should inform a new sender about exactly one MCS. No specific action is needed if an existing sender stops transmitting to a group. An interesting point here is that there is no obvious way for the MARS to know about senders that have stopped transmitting to a group. This has implications for maintaining the accuracy of the allocation map. The MARS maintains load sharing across the MCSs by basing its allocation decisions on the map. Hence an incorrect map may lead to improper load sharing. This problem does not exist when sharing receivers as they are explicitly registered (using MARS_JOIN/MARS_LEAVE) with the MARS. We explain possible solutions in section 3.2.2.

3.2.2 Balancing the load when sharing senders

The MARS can learn about a new sender when it receives a MARS_REQUEST message from it. However, senders do not explicitly deregister from the MARS when they stop transmitting to the group. So the MARS cannot update its allocation map which may cause it to make incorrect allocation decisions. Thus load balancing may not be achieved.

A simplistic solution is to ignore the need for deregistration by the senders. This simplicity is ofcourse at the expense of the possibility of improper load balancing, as we saw above. However, assuming that all senders have a similar lifetime, the rate of addition and deletion of senders to an MCSs' set (set of senders currently supported by an MCS) will remain approximately the same for all MCSs.

Another solution to the problem is to have the senders explicitly register/deregister with the MARS. The MARS_REQUEST message that a sender transmits to the MARS can be used for registration, while a new message (MARS_SNDR_LEAVE) can be used for deregistration from the MARS. The MARS can thus be kept aware of the existing senders. This solution however involves changing the client behavior. So [GA96] based clients will not be considered by the MARS for load sharing decisions. Also, compatibility issues that arise due to existence of different versions of clients will have to be resolved.

One can avoid changing the client behavior by making the MCS responsible for deregistering senders. The MCS terminates a VC from each sender to the group (or atleast the senders being supported by it). It is aware of the state of each of these VCs, and so knows when such a VC is released. Release of such a VC can be viewed as the sender terminating transmission to the group. The MCS can then inform the MARS of this event using a new message (MARS_SNDR_LEAVE), causing the MARS to update its allocation map.

It remains to be seen as to which of the above is a better approach for solving the problem.

3.2.3 Interactions between the MARS and MCSs

As was indicated in section 3.1, an important goal of this approach is to not involve the MCSs in the allocation mechanism. An MCS must remain oblivious of the existence of other MCSs supporting the same group. Thus each MCS is made aware of only the receivers supported by it, when sharing receivers. So the MARS-MCS interactions do not change at all from those described in [TA96].

The only consideration needed in the MCS design is that an MCS should accept MARS_SJOIN and MARS_SLEAVE from the MARS on either the ServerControlVC or the point-to-point VC. This is because the MARS informs an MCS of new/leaving group member only if it allocates/has allocated that member to that MCS.

3.2.4 Interactions between the MARS and clients

The interactions between the MARS and the clients also remain the same. The only exception in this case is if the explicit deregistration mechanism explained in section 3.2.2 is adopted for senders. In that case the clients will have to send a new message (MARS_SNDR_LEAVE) when a group's outgoing point-to-multipoint data VC is closed. Closing the VC indicates that the client is no longer sending to that group, and hence the MARS can update its allocation map.

3.3 Using multiple MCSs with multiple MARS

The SCSP approach provides a mechanism for supporting multiple MARS in a cluster. The multiple MCS mechanism that we have outlined above will work with multiple MARS also. In this case, the Designated Server (DS) protocol described in [LA96] can be used to elect one of the MARS as the allocation mechanism. This DS will ensure consistency of the allocation map amongst the MARS, and allocates senders/receivers amongst the multiple MCSs. The DS mechanism also has a recovery mechanism that can be used to elect a new MARS to function as the DS, in case of failure of the existing one.

4 Summary

We propose a scheme for supporting multiple MCSs per group using the MARS as the allocation mechanism. This scheme subverts the need for an inter-MCS synchronization protocol. It requires enhancement to the current MARS server architecture as described in [GA96]. We aim to minimize the changes required to the client or MCS architecture in this scheme, thus maintaining interoperability between different versions of clients and MCSs.

Author's address

Rajesh Talpade - taddy@cc.gatech.edu - (404)-894-9910
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", Internet-Draft, draft-ietf-ipatm-ipmc-12.txt, Feb 1996.

[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., "Multicast Server Architectures for UNI 3.0/3.1 based ATM networks", Internet Draft, draft-talpade-ion-marsmcs-00.txt, March 1996.

[TA96a] Talpade, R., G. Armitage, M. Ammar, "Experience with Architectures for Supporting IP Multicast over ATM" , submitted to IEEE ATM '96 Workshop, San Francisco, August 1996.