Experience with Architectures for Supporting IP Multicast over ATM

(Extended Abstract) *

Rajesh R. Talpade (Georgia Tech)
Grenville J. Armitage (Bellcore)
Mostafa H. Ammar (Georgia Tech)

{taddy, ammar}@cc.gatech.edu
gja@thumper.bellcore.com

August 13, 1996

Abstract

Two basic techniques have been proposed for the intra-subnet multicasting of IP packets over ATM networks. Both the approaches are based on the use of a Multicast Address Resolution Server (MARS). One approach makes use of a mesh of point-to-multipoint Virtual Circuits (*VC Mesh*), while the other uses a shared point-to-multipoint tree rooted at a Multicast Server (*MCS*). In this paper we describe the design and implementation of a MARS, a MARS client and of an MCS. We compare the VC Mesh and MCS approaches and present results from performance experiments of the MCS approach.

*This work was supported in part by Bellcore.

1 Introduction

The ATM Forum's currently published signaling specifications (UNI 3.0 and UNI 3.1) do not provide a multicast address abstraction for ATM networking. A layer 3 multicast address has to be resolved into the corresponding set of ATM addresses of group members, before a point-to-multipoint unidirectional Virtual Circuit (VC) can be set up for multicasting layer 3 data. 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]. The Multicast Address Resolution Server (MARS) is used to maintain the mapping between layer 3 group addresses and the corresponding ATM addresses. Hosts in the ATM network use the MARS to resolve layer 3 multicast addresses into corresponding lists of group members. The source has two options for multicasting data to the group members. It can set up a point-to-multipoint VC from itself to each of the group members, and then proceed to send data out on it (the *VC mesh* approach). Or it 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 VC mesh approach has been specified in [GA96], while the MCS architecture is currently specified as an Internet-Draft ([TA96]). This article describes our experience with implementing the MARS architectures on SparcStations running SunOS 4.1.3, and ForeThought version 3.0.3 (1.5) software. The ATM hardware used was Fore SBA200e NICs, with Fore ASX200 switches (interconnected by OC/3 Multimode fiber) connected to the machines by OC/3 UTP cable. It presents a description of the MCS architecture, together with a comprehensive comparison of the VC Mesh and the MCS approaches. We also present preliminary results of performance experiments using our implementations.

The ATM Forum's forthcoming UNI 4.0 signaling specification does not have a true multicast address abstraction either, as it would need a pure multipoint-to-multipoint service in the ATM network, which is not supported. Instead, a Leaf Initiated Join (LIJ) service is proposed to allow leaf nodes to add themselves to an existing point to multipoint VC without the intervention of the VC's root. Each tree is identified by the root node's ATM address in combination with a 32 bit ID value unique to the root. Using LIJ to support IP multicast would require the MARS to propagate the identities of every sender to a group, so that receivers can add themselves to the appropriate VCs.

2 The MARS architecture

2.1 Overview of Operations

The MARS architecture is based on a client/server model. The MARS acts as a registry, associating layer 3 multicast group addresses with the ATM interfaces representing the group's members. It manages a *cluster* of ATM-attached endpoints, with a cluster being defined as the set of ATM interfaces that choose to use the same MARS to register their memberships and receive their updates from. The distribution of multicast group membership information between MARS and the clients is achieved by MARS messages. Clients query the MARS when a layer 3 multicast address needs to be resolved to the set of ATM level end-points making up the group at a given time. Clients keep the MARS informed when they need to join or leave specific layer 3 groups. The MARS maintains a point to multipoint VC out to all clients in the cluster so as to enable asynchronous notification of group membership changes.

2.2 The Implementation Design

The Internet Protocol (IP) is one layer 3 protocol which has to make use of the MARS architecture to support multicasting over ATM. Although the MARS implementation was general enough to support any layer 3 protocol, we implemented the client-end of the architecture to support IP Multicast in particular. A one-to-one mapping is assumed between a cluster and a Logical IP Subnet (LIS).

Figure 1 shows the structure of our MARS client implementation (*ipmcatmd*) which can be best thought of as a "shim", or "convergence", layer sitting between IP's link layer interface and the underlying UNI 3.1 service. *Ipmcatmd* has to establish a point-to-point bidirectional VC, which it uses to send queries to and receive replies from the MARS. It also terminates a point-to-multipoint VC, originating from the MARS and terminated by all clients, which is used to disseminate group change information.

Ipmcatmd has been implemented as a user-level daemon which executes at each of the hosts. Bellcore's experimental UNI 3.0/3.1 signaling stack (Q.Port^{TM} Portable ATM Signaling Software) is used as the signaling entity. *Ipmcatmd* communicates with the Q.Port^{TM} software, which also executes at user-level, through a TCP socket. In addition to the user-level code, *ipmcatmd* has an associated streams module. This is used by the IP link layer interface to communicate with *ipmcatmd*.

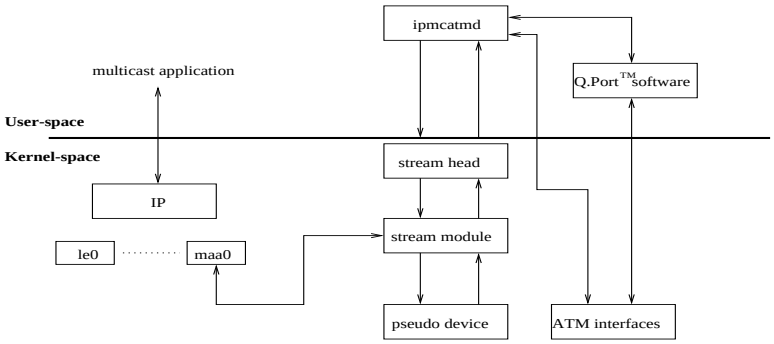

Figure 1: Structure of the MARS client.

The MARS was implemented entirely in user space under SunOS4.1.3 [MARS96]. As with the MARS client, the Q.Port^{TM} software was used to provide host side UNI signaling. Both the MARS and the Q.Port^{TM} software shared access to the FORE Systems ATM interface card. The Q.Port^{TM} software entity informed the MARS process of incoming point to point calls, and built outgoing point to multipoint VCs on behalf of the MARS. The MARS code provided most of the functionality described in [GA96].

3 The VC Mesh and MCS approaches

3.1 Summary of the VC Mesh approach

When an application at a client in the LIS joins a group, the client has to register this event with the MARS. The MARS thus maintains a mapping of IP Multicast group addresses and corresponding ATM addresses. Each of these mappings indicate the ATM addresses of clients which have joined that particular group. A source in the LIS which needs to transmit data to group members first sends a request to the MARS. The MARS responds with the ATM addresses of the clients that are members of the particular group. The source opens a point-to-multipoint VC to the clients and transmits data on this outgoing VC. The source can add/delete clients from the outgoing VC as it receives updates from the MARS whenever a client joins/leaves the group. The outgoing VC from the source is closed on absence of traffic for the group.

3.2 Summary of MCS Architecture and Operation

The MCS acts as a proxy server, retransmitting data received from senders to the group members. The simplest MCS architecture involves taking incoming AAL_SDUs from the multicast sources and sending them out over a 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 interest 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 AAL_SDUs and determine which layer 3 group they are destined for. A single instance of such an MCS could support several multicast groups, and have one outgoing point-to-multipoint VC for each of the groups. The MCS architecture, along with its interactions with the MARS, is described in detail in [TA96]. We now describe the working of an MCS.

The MCS opens a point-to-point bidirectional VC with the MARS (the identity of which must be known) on startup. This VC is used by the MARS for sending all control messages specific to this MCS, and by the MCS for sending control messages to the MARS. The MCS registers itself with the MARS on startup. The MARS then adds the MCS to a point-to-multipoint unidirectional VC (Server Control VC) that it maintains to each MCS in the cluster. This VC is used by the MARS to disseminate general cluster information to all the MCSs.

After registering itself, an MCS can register to support group(s) with the MARS. It then gets to know of the current group membership from the MARS, and opens a point-to-multipoint VC to the group members. The MCS thus has a separate outgoing point-to-multipoint VC for each group (with non-zero membership) that it has registered to support.

If the MARS knows of the existence of an MCS that supports a particular group, it makes the senders to the group aware of only the MCS as a group member, instead of the actual group members. As a result, the senders open their VCs and transmit data to the MCS. The MCS then retransmits the received data on the outgoing

point-to-multipoint VC. If the MCS is supporting multiple groups simultaneously, it looks inside the received AAL_SDUs (from the senders) before retransmitting it on the appropriate outgoing VC. The MARS transmits group membership change information to the MCSs on the Server Control VC, which enables them to keep their outgoing VCs updated.

It is possible that a group is already active (with senders and group members) before an MCS registers to support it. The MARS then makes the senders transition their traffic to the MCS by use of control messages. An MCS can also discontinue support of a group by unregistering for it with the MARS. The MARS has to convey this information to the senders, causing them to stop transmitting to that MCS. If there are no more MCSs supporting the group, the MARS has to convey group membership information to the senders so that they can open VCs and start transmitting to the actual group members.

3.3 A comparison of the VC Mesh and MCS approaches

Table 1 compares some quantitative parameters needed for supporting a single group while using the MCS and the VC mesh approaches (ignoring the control VCs maintained between the MARS and the cluster members).

3.3.1 Advantages of using MCSs

- As can be seen above, VC usage is much better in the MCS case. The increased VC usage in the VC mesh approach leads to greater consumption of resources like memory for maintaining state, buffer allocation per VC, and the VCs themselves, which may be a scarce and/or expensive resource. A clusters size is currently dictated by the LIS size, which in turn is dictated by the unicast subnetting that has been applied. An administrator might use knowledge of the VC mesh mode's VC consumption to pre-emptively subnet his network into small subnets, and thereby end up with more Inter-LIS devices (IP routers) between LISs. We shall compare the performance of IP routers (used in the VC Mesh case with more, smaller LISs) and MCS (used with larger LISs due to better VC consumption) in more detail in section 4.
- Group membership changes cause a decreased level of signaling load to be generated at the switch in the MCS approach. This is because only the MCS has to add/delete a cluster member from the point-to-multipoint VC. The other VCs are not affected. Thus signaling requests only occur at the UNI between the MCS and the switch, as opposed to occuring at the UNIs between all the sources and the switch in the VC mesh case. This is especially beneficial for large groups with several senders, or when the group is highly dynamic, or when the links between the switch and the cluster members are error-prone, which may cause group members to be temporarily dropped from the multicast group, thus making the group more dynamic than it actually is.
- The MCS approach provides a centralized control of multicast bandwidth usage over the ATM network. This is useful in cases where policy demands a limit on the share of bandwidth available for multicast purposes. In such a case, the administrator who sets up a cluster member as the designated MCS for a group, can control the rate at which the MCS multicasts data.

3.3.2 Disadvantages

There are some disadvantages of using the MCS approach, however solutions do exist which minimize them.

- Data throughput and end-to-end latency may be adversely affected due to the additional level of indirection introduced by the MCS.
- The MCS can potentially become a bottleneck and a central point of failure. Both this problem and the previous one can be solved by using multiple MCSs per group.

Number of multicast sources multicasting to group G = n
Number of group members in G = m

	MCS	VC Mesh
total VCs terminated at cluster members	$n + m$	$n * m$
point-to-multipoint VCs	$n + 1$	n
VCs terminated at each group member	1	n
signaling requests generated due to a single membership change	1	n

Table 1: Quantitative comparison of VC Mesh and MCS approaches

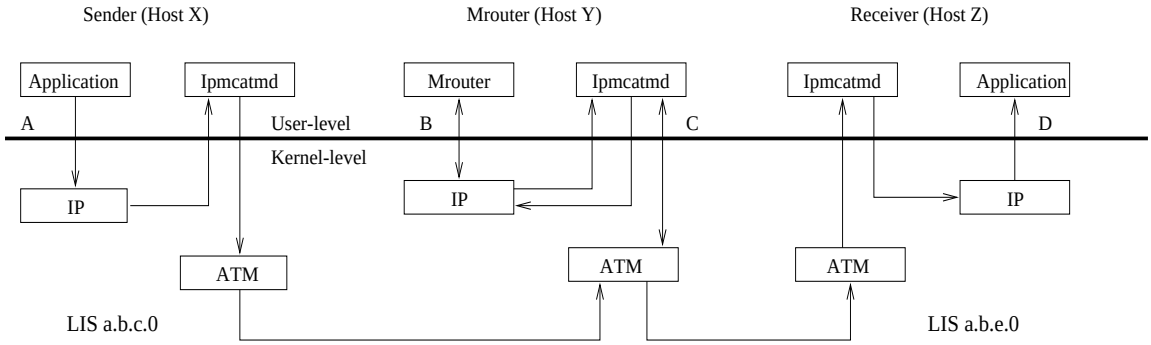

Figure 2: Use of an Mrouter with the MARS architecture.

- Each group member needs to terminate one point-to-multipoint VC originating from the MCS. So if a multicast source happens to be a member of the group it is transmitting to, it will receive a copy of the data back over the VC from the MCS. Thus additional header identification is needed for a source to discard such "bounced-back" data. This mechanism has been defined in [GA96] to be a 16 bit cluster member identifier (CMI).
- Each multicast source in the cluster may desire to use differing quality of service (QOS) parameters for outgoing traffic. Use of the MCS implies that all group members will receive data with QOS as determined by the MCS, irrespective of the QOS used to get them from the source to the MCS.

4 MCS Performance Experiments

We conducted several experiments to understand the performance implications of using the MCS approach. The testbed used for the experiments consisted Sparcstations (5 Sparc5s, 3 Sparc20s) running SunOS4.1.3, with Fore SBA200e NICs controlled by ForeThought version 3.0.3 (1.5). Two Fore ASX200 switches connected the machines through OC/3 UTP cable. The switches were interconnected through OC/3 Multimode fiber. The Sparc5s were never used as receivers of data due to the known problems that they have with receiving ATM cells when running SunOS 4.1.3 and Fore software.

The first experiment was aimed at comparing the delays involved in using a multicast IP router (an *mrouter*) between 2 smaller LISs, as opposed to using an MCS with a larger LIS. In the first case, a sender to group G existed on LIS a.b.c.0, and a receiver (member of group G) existed on LIS a.b.e.0. The mrouter, which had interfaces on both LISs, enabled data from the sender to be delivered to the receiver. The experiment measured the average application level delay for receiving 2 bytes at the receiver which were transmitted by an application at the sender (delay from A to D in figure 2). The second case involved using the sender and receiver on one LIS, but having an MCS forward traffic from the sender to the receiver. The delay was measured as before (from A to D in figure 3).

An average delay of 16 msec was observed in the mrouter case, while an average delay of 8.5 msec was observed in the MCS case. The mrouter introduced significant additional delay in the data path since it required data to

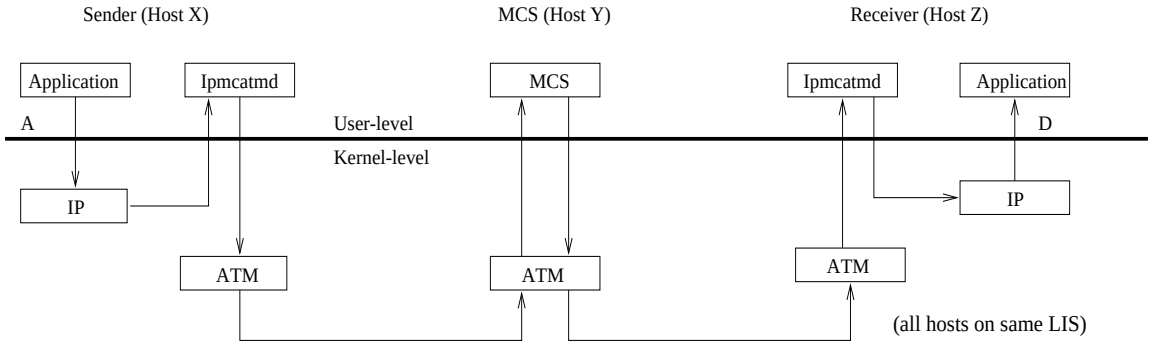

Figure 3: Use of an MCS.

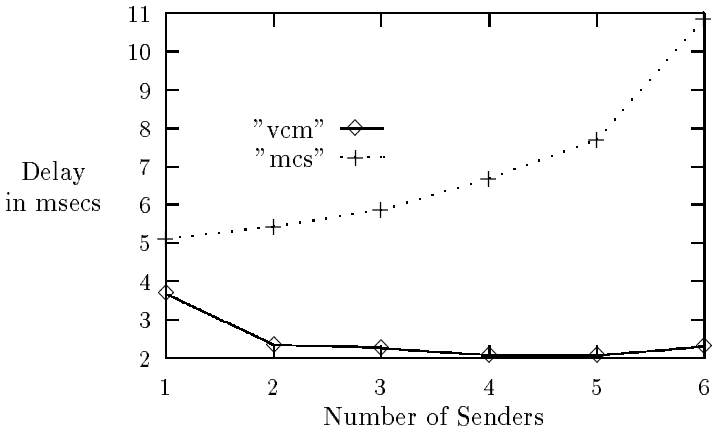

Figure 4: Impact of number of senders on VC Mesh and MCS approaches

be passed through an additional IP layer (at the intermediate mrouter). High performance hardware that could be used to make an mrouter run faster would also have an equivalent effect on a high performance MCS. Whilst the numbers (16ms vs 8.5ms) are not important in absolute terms, they are interesting as relative measures. Thus use of MCS, which enables us to use larger LISs due to lower VC usage, is definitely preferable to use of VC Mesh, which due to higher VC usage may lead to smaller LISs (as explained in section 3.3.2), and hence needs more LISs interconnected by mrouters to accommodate the same number of hosts.

The second experiment compared the VC Mesh and MCS approaches as the number of senders is increased, with the goal of demonstrating the effect of increased load on the two approaches. We used one receiver (group member) in the LIS, with one through six senders transmitting simultaneously. The delay at the receiver to receive 20MB of data was measured in all the cases. The amount of data received was kept constant so as to highlight the influence of the increasing number of senders, and not of the resulting increase in the amount of data received (which increases with number of senders also).

As can be seen in figure 4, the delay in receiving the same number of bytes for the VC Mesh case actually decreases with increase in the number of senders. This is because the receiver gets an increasing amount of data with the increase in the number of senders, which causes it to receive 20 MB faster. This continues until it cannot receive data any faster, and then the effect of saturation of the incoming link to the receiver combined with the increase in interrupt frequency at the receiver causes the delay to actually increase, as is seen for six senders.

In the MCS case, both the delay and the rate of increase of the delay is directly proportional to the number of senders. This can be attributed to the fact that the MCS needs to retransmit each AAL_SDU received from the senders. The interrupt frequency increases with the increase in number of senders. The MCS therefore takes a longer time to retransmit 20 MB. This in turn increases the delay for the receiver to get the 20 MB.

The last experiment highlights the fact that a single MCS can easily be overloaded with over-active senders. A possible solution to this problem is multiple MCSs, wherein senders are distributed across more than one MCS, with each sender being supported by exactly one MCS. We are currently working on mechanisms to support multiple MCSs [TA96].

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.

[MARS96] Online public release of MARS in part source, part object code form. URL: ftp://ftp.bellcore.com/pub/gja/mars

[TA96] Talpade, R.R., and Ammar, M.H., "Multicast Server Architectures for MARS-based ATM networks", Internet Draft, draft-talpade-ion-marsmcs-01.txt, July 1996.