# SEAM: Scalable and Efficient ATM Multicast

Matthias Grossglauser*

INRIA
Sophia Antipolis, France
Matthias.Grossglauser@inria.fr

K.K. Ramakrishnan

AT&T Labs-Research
Murray Hill, NJ
kkrama@research.att.com

**Abstract**

This paper proposes a multipoint-to-multipoint multicast architecture for ATM networks. The necessity for such an architecture stems from the scalability requirements, both in terms of state to be maintained in the network and in terms of the group population dynamics, of a wide range of networking applications.

We argue that approaches of using multicast servers or meshes of point-to-multipoint Virtual Circuits (VCs) may be inadequate solutions to this problem. We propose a true multipoint-to-multipoint architecture called SEAM, which uses a single VC for a multicast group consisting of multiple senders and receivers. We achieve this without changes to ATM's AAL5. SEAM relies on an additional switching feature we call cut-through forwarding, which enables the mapping of several incoming VCs into outgoing VCs.

We believe that SEAM is both an important and necessary step in the evolution of ATM. It will enable applications relying on group multicast to benefit directly from ATM's quality of service support and scalable bandwidth and the resulting performance advantages. Also, it considerably simplifies the problem of supporting IP multicast over large ATM networks.

## 1 Introduction

Support for multi-party communications is viewed as a critical building block for enabling transparent, scalable communication. In most of the existing work on IP and ATM multicasting, human collaborative applications such as audio/video-conferencing and shared interactive workspaces are cited as one of the primary applications to benefit from a multicast service. Since bandwidth has been and continues to be a scarce resource in the Internet, one of the main goals of these studies has been the gain in bandwidth efficiency that multicast offers over unicast. The obvious goal of minimization of resource usage to a single transmission of a message over a given link has been paramount. We believe that another fundamental attractiveness for multicasting is its ability to abstract the identity of individual members of the group to a single common group identity. This enables scaling of the communication mechanism to arbitrarily large numbers of participants and network sizes.

The emphasis on human collaborative applications and bandwidth efficiency has led to design decisions that can impede scalability for other more general uses of multicast. Multicast sessions for human collaboration, based on the nature of human behavior, can be expected to be relatively static. Also, the set of senders to the group is small: a person is often unable to look and listen to even a few speakers at the same time. Thus, scalability has been addressed only in terms of the size of the receiver population. This has led to designs based on per-source distribution trees, such as DVMRP [1], MOSPF [2], and ATM UNI 4.0 [3].

Multicast serves as a communication *abstraction* between a set of group members. In many circumstances, the group members are all potential senders; furthermore, the group population can be large and highly dynamic. The main motivation for using a multicast service then lies in the decoupling of the application and the group membership management. In other words, the application is relieved of the burden of tracking group membership; it can address the group as a single entity. This proves important in many distributed systems, where requiring the application to track membership and manage member arrivals and departures results in complexity and is an additional computational burden. Such applications include distributed interactive simulation (DIS), distributed databases and caches [4], and distributed

*This author was supported in part by a grant from France Telecom/CNET.

games.

In DIS, a multicast group represents a cell of simulated terrain, and group members are processes representing objects moving in the corresponding cell of the terrain. Objects "inform" other interested objects (i.e., objects in the same cell) of their actions (movement, state changes etc.) Thus, all objects are potential senders. Also, the objects' movements between cells can translate into frequent membership changes.

In summary, in order for a multicast service to efficiently support a wide range of applications, we believe it has to offer the following:

1. Membership management symmetry for senders and receivers.

2. Scalability in the total network size, the group size, and the frequency of membership changes.

These requirements impose certain design choices. First, the multicast group should be represented by a single tree shared by all group members (senders and receivers) [5]. We will elaborate on this in Section 2. Second, only network nodes (switches or routers) on the tree for a group should have to maintain state concerning that corresponding group. Otherwise, the cost of setting up and maintaining a large number of sparse groups will become excessive, as has been observed with DVMRP. Third, it must be possible to support member-initiated group membership changes. If all membership changes have to be processed centrally, the burden of communication and processing overhead, latency, and unreliability can become excessive.

It is important to note that these issues are fundamentally tied to the scalability requirements of multiparty communication, and do not depend on the particular network protocol, such as IP or ATM. Note that the clear distinction of the connection-less nature of IP and the connection-oriented nature of ATM vanishes for multicast: both protocols have to maintain per-connection state in network nodes. The only difference lies in how this state is set up and maintained. IP has traditionally preferred soft-state approaches based on periodic refreshes, while ATM has preferred hard-state approaches based on explicit state creation and deletion.

The only protocols proposed so far recognizing the need for a shared tree are CBT [5] and the sparse mode of PIM [6]. In ATM, member-initiated join and leave has been incorporated into ATM Forum UNI version 4.0 for point-to-multipoint communication. However, there has been no proposal for a "convenient" multipoint-to-multipoint communication mechanism. While MARS [7] uses a single tree spanning all receivers, rooted at a multicast server, senders have to create point-to-point (unicast) connections to the server, violating the network state scalability requirement above. Also, the server may be both a bottleneck (due to the necessity to perform reassembly-segmentation in the server) and a single point of failure. The number of Virtual Circuits (VCs) used on the link to the server may be a scarce resource as well.

In this paper, we propose a true multipoint-to-multipoint architecture for ATM, called SEAM. It is compatible with the requirements elaborated earlier, and is therefore based on a single, shared tree. A single VC per link is used to send cells from all the senders of the multicast group to the receivers in the group. Conceptually, SEAM is closest to CBT, while it maintains the cell-switched nature of ATM. Cell switching, i.e., forwarding small, fixed-sized cells, has been introduced in ATM to simplify the switch's hardware architecture, and providing for higher performance. However, manipulating sub-packet units means that we need to be careful when forwarding from more than one incoming VC into one (or several) outgoing VC, because cells belonging to one packet need to remain in sequence. We solve this problem through a small modification to cell forwarding, called *cut-through forwarding*. *Short-cutting* is a proposed signalling mechanism that sets up the cell forwarding tables so that packets span the shared tree, reducing delay and increasing bandwidth efficiency. Short-cutting is an emulation of reverse path forwarding on the shared tree through an appropriate mapping of the incoming VCs onto the outgoing VCs belonging to the same SEAM tree. SEAM achieves the multipoint-to-multipoint communication over AAL5, with no changes to the cell headers, and does not require switches to examine the cell-payload.

One of the ancillary goals in our proposing SEAM is a desire to support IP multicasting in ATM networks. An ATM backbone may have many IP routers at the periphery. A straightforward mapping of a large number of IP multicast addresses among these routers using point-to-point VCs results in excessive state to be maintained and managed in the ATM backbone. SEAM overcomes this by having a single virtual channel associated with each IP multicast address.

The remainder of this paper is structured as follows. Section 2 explores the necessity of having a single shared tree in more detail. Section 3 discusses signalling, and Section 4 discusses data forwarding. Section 5 concludes the paper.

## 2 Necessity of a Single Shared Tree

In this section, we show that the necessity of a single shared tree derives directly from our assumptions of sender/receiver symmetry and scalability. More specifically, we focus on three aspects of scalability: network state, dynamic membership management, and membership coherence.

### 2.1 Network State

Even when the group membership is static, group connectivity based on per-source trees or on a multicast server results in a number of VCs equal to the number of potential sources on the links towards the receivers or the multicast server. This may be reasonable in an environment of a few senders and a large number of receivers. However, a large number of applications rely on a model where all group members are receivers and potential senders.

### 2.2 Membership Management

We use simulation to illustrate the difference in join performance for multicast groups based on a shared tree (such as CBT and SEAM) and groups based on source-based trees (such as DVMRP or ATM UNI 4.0). We first generate random network topologies, using the algorithm proposed by Waxman [8]. The topologies we generate have 1000 nodes and an average outdegree of approximately 7[1]. We then randomly select nodes of this graph as group members. There are three types of possible group members: pure senders, pure receivers, or members that are both senders and receivers. We then create a multicast group connecting the members. There are two types of group connectivity: shared-tree groups (SEAM) and source-based tree groups. For shared-tree groups, we use the following heuristic to determine the core node. We select as the core the node of the random topology closest to the center of mass of the set of receivers. For source-based tree groups, we set up a shortest path tree for each sender (including members that are senders and receivers) to all of the receivers. We then randomly select a node among the non-member nodes to issue a join request. This new node can request to join as a receiver, as a sender, or as both.

We consider two metrics to evaluate the performance of the join request, namely latency and resource consumption. "Latency" measures the time it takes to create the state in the network to include the new member in the group. For shared-tree groups (SEAM groups), *join latency* is the time to reach the existing tree on the shortest path towards the core.[2] For source-based tree groups (meshes of point-to-multipoint VCs), join latency is the maximum of the join latencies to all of the source-based trees. In other words, we assume that the multiple joins can progress in parallel. Each hop traversed by the join contributes a unit of time to the latency. *Resource consumption* measures the total amount of processing a join requires in the network. We simply count how many hops are traversed by signalling messages related to this join. As in a SEAM group, there is only one signalling message, this is of course equal to the distance in hops between the new node and the tree.

For each random topology, we generate fifty multicast groups and one join of each type to this group (receiver join, sender join, sender-receiver join). As a stop criterion, we require that the 90 % confidence interval of both metrics for the three types of joins be within $\pm 10\%$ of the estimated mean.

We consider two types of group members. For the first type of group members, we assume only sending *and* receiving members. Fig. 1 and 2 depict the latency and resource consumption as a function of the group size. The join latency for the shared tree is considerably lower than for the collection of source-based trees. This is because it is very likely that at least one of the joins to the source-based trees has to travel quite far to reach the existing tree. Of course, the exact extent of the performance advantage of the shared tree over the collection of source-based trees depends on the network and the group topologies as well as on the core placement. The difference in terms of resource consumption is extremely large, and continues to grow with group size (cf. Fig. 2). This is because a new group member in the source-based tree case has to build up a new tree rooted at itself to all the existing receivers, and it has to join all of the existing source-based trees. Thus, the join cost grows approximately proportional to the group size, whereas for the shared tree, the cost actually gets lower as the group size increases, as on the average, the join hits the tree of a large group earlier.

For the second type of group members, we assume there are only pure senders and pure receivers in the group. The size of the receiver population is fixed at 100. We vary the number of senders. Fig. 3 and 4 depict the join latency and resource consumption for both new senders and new receivers. We see the excellent join performance, both in terms of join latency and of resource consumption, of the shared tree. For the group based on source-based trees, the join latency for senders is worst. This is because a new tree has to be built up, and therefore the latency is determined

[1] This outdegree is relatively large, but for random topologies of this size, it is computationally hard to get smaller outdegrees, as graphs tend to be disconnected with high probability.

[2] For senders, it is sufficient to reach any part of the shared tree, while receivers have to connect to a node on the receiver-spanning tree.

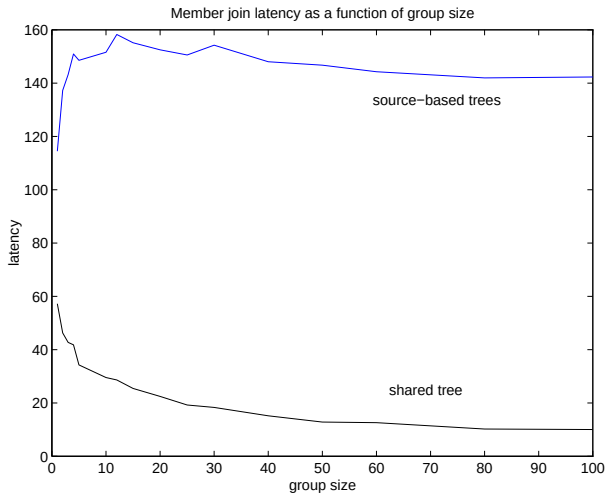


Figure 1: The join latency for members as a function of the number of members. A member is both a sender and a receiver.

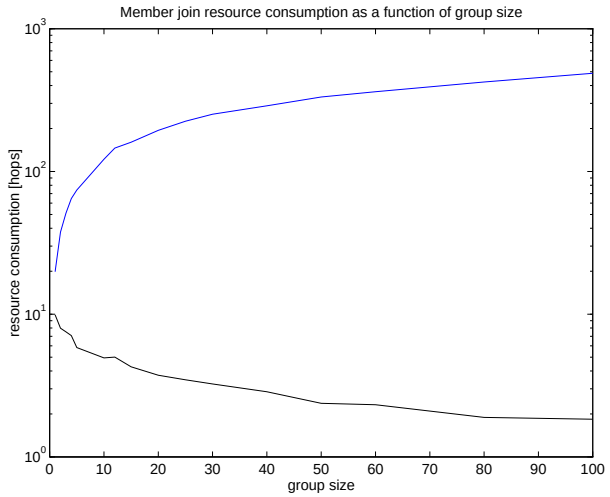


Figure 2: The join resource consumption for members as a function of the number of members. A member is both a sender and a receiver.

by the longest path between the new sender and any of the existing receivers. In terms of resource consumption, we again observe more than two orders of magnitude in difference between the shared-tree group and the source-based tree group. Only for small numbers of potential senders are receiver joins comparably cheap to the shared tree. Again, the join cost for receivers scales up approximately linearly with the number of sources.

Our simulation results suggest that a single shared tree is a good solution for large scale multicast groups, where the number of senders is not negligible.

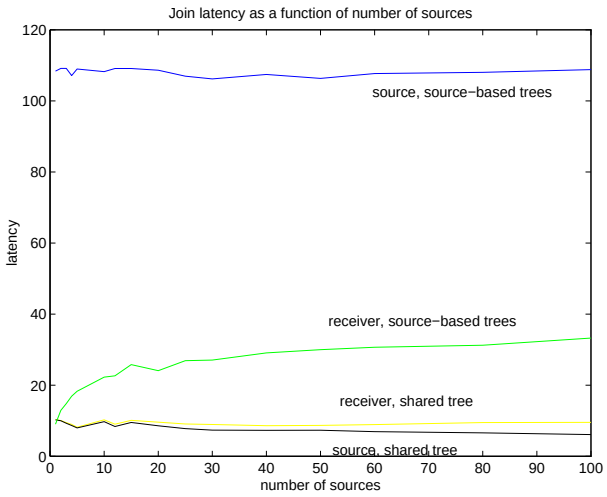


Figure 3: The join latency for senders and receivers as a function of the number of senders. The number of receivers is 100.

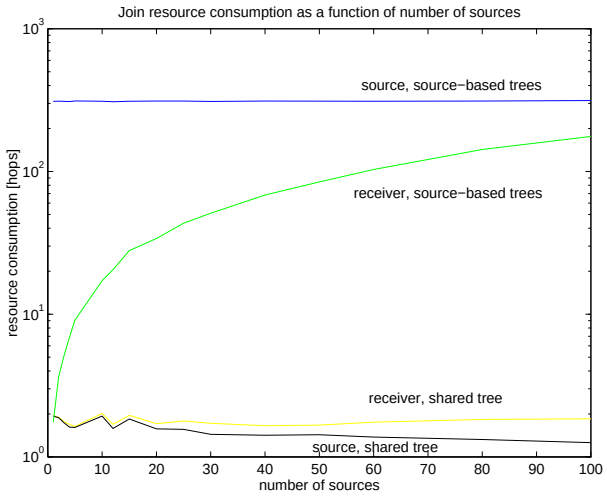


Figure 4: The join resource consumption for senders and receivers as a function of the number of senders. The number of receivers is 100.

## 2.3 Membership Coherence

In a per-source tree environment, providing the current membership information to a new sender/receiver is problematic as well. One way to provide information on the existing sources and receivers is to have a *repository* of such information somewhere in the network. The problems of keeping the repository up-to-date, ensuring that access to the repository is not a bottleneck in the presence of constant membership change and providing a coherent view to new participants have to be addressed. For example, it is difficult to deal with the situation of providing coherent information to new receivers while we are setting up a new source. If we

fail to provide the appropriate information, a receiver would have no way of knowing that it is not connected to all the current sources. A similar situation applies to a new source, as it may not be able to ensure all currently interested receivers have been "joined" into the group. The obvious dependence on a highly reliable repository exists, and recovery from the failure of a repository is not trivial.

With SEAM, the group abstraction provided by the shared tree avoids the need for such a repository. The core serves as a "beacon" towards which senders and receivers issue joins to the single shared tree. The join terminates at the point where a new branch has to be added to the existing tree. There is no danger of a receiver not "seeing" all the sources, and vice versa. Furthermore, failure of a switch (including the core) does not result in any failure of continued operation of the multicast group (other than the natural partitioning of the network). Recovery from a failure also is simple, since the failed switch does not have to maintain/recover any state of the current group membership.

### 2.4 Delay Issues

The major disadvantage of shared trees are delays to receive data that are potentially higher than in the case of shortest-path sender-based trees [9]. However, it should be kept in mind that most networks exhibit a certain degree of hierarchy. Even in a local area network, where the network may be physically connected in an arbitrary mesh, the routing layer typically organizes the network in a hierarchical fashion. For example, routing protocols such as OSPF or IS-IS use hierarchy to control the amount of routing information that is being distributed all over the network. Given this hierarchy, it can be expected that sender-based trees will not offer significantly different delays than shared trees, because the hierarchical structure reduces the number of alternative paths from a sender to a receiver. For example, a campus network is usually connected to the Internet over a single leased line. This may be the most likely bottleneck in a wide-area multicast session. Both shared trees and sender-based trees would have to choose this link to reach all members on the campus. We recognize the need to substantiate this claim further through simulations, which is the subject of our future work.

## 3 Signalling Support for SEAM

We first address the requirements on signalling and group management.

### 3.1 Group Creation

When senders or receivers want to join a multicast group, they send a join message towards the core. In other words, the choice of a core needs to be made prior to setting up any part of the multicast tree. The question is: who is responsible for setting up the core?

We propose to have an "initiator", who may or may not be a future member of the group, responsible for defining the core and disseminating the existence of the core to the potential members. This can happen, for example, through a name service, as proposed in [5], or through directly contacting the members, depending on the semantics of the group.

Note that it does not need to be the initiator's responsibility to choose what switch in the network is elected as core. In our view, the network would offer core selection as a service. The initiator could convey some information about the expected group membership (e.g., geographical information) in order for the network to optimize the choice of a core. The network answers a core selection request with a handle that the initiator may use to advertise the group. Since any switch may potentially act as a core, one may envisage some form of load distribution among several of them in the network to act as cores for different multicast groups.

### 3.2 Member Initiated Joins

It is obvious from the discussions in [5, 6] that member-initiated joins are a necessity for a scalable multicast service. The advantages of a member initiated join approach over a root initiated approach are twofold. First, the root of a multicast tree (in our case the core) does not need to know about or keep track of the membership of the group. This means saving processing resources and state space. Second, a join to a group that already has a tree set up can be terminated at the point where a new branch will be added to the existing tree. This means saving bandwidth (due to signalling messages traveling smaller distances, or hops), processing resources in the switches and reduced latency.

New members who wish to join the multicast group either as senders or receivers issue a join request. This join request travels towards the core on the shortest path, until it hits a switch that is already on the requested group's tree. A new branch is then created from that switch to the joining member. Basically, the procedure is similar to that of receivers joining a point-to-multipoint VC in UNI 4.0 [3], but we generalize it to both sending and receiving members. Similar options as proposed originally for UNI 4.0 (without sender participation, with sender notification, with sender permission) may be used.

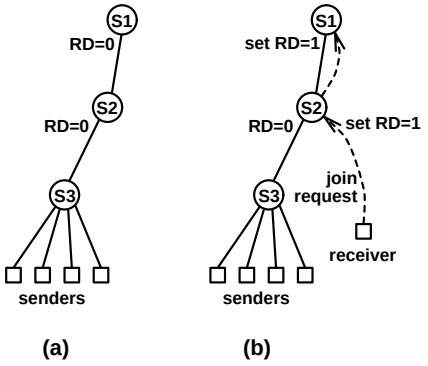


Figure 5: Updating of the RD bit upon receiver join.

### 3.3 The Receivers-Downstream (RD) Bit

The use of a single shared tree among all receivers and senders requires us to introduce a small amount of per-link state to avoid wasting resources by transmitting to sender-only end-systems. In order to avoid forwarding packets to members who are only senders, we associate a flag with the group at each "on-tree" switch. The flag, called the "Receivers Downstream" bit, indicates if there are any receivers downstream from this port. Consider situation (a) in Fig. 5: RD=0 at switches S1-S3 means that the respective ports only have senders downstream, and therefore no packets need to be forwarded on these ports.

If a new receiver connects to the existing tree at a port that has the RD bit cleared, then the forwarding table in some upstream switches have to be updated, so that packets will be forwarded down to the new receiver. The join request therefore has to travel towards the core on the tree and update the forwarding table in each switch. The join request stops when it hits the core or a switch with the RD bit set on at least one other port, which means that packets sent to the group already reach this switch. At each switch traversed on the tree, forwarding tables have to be updated such that packets will be forwarded towards the new receiver.

The core acts as any other SEAM switch. The only exception is that all data gets forwarded to the core, even if it does not have receivers on its other ports. This is because the RD bit is not helpful since receiver joins only go as far as the core. There is no clean way to have the signalling progress beyond the core to the set of switches downstream, until the point where we reach switches that all have their RD bit set. Therefore, by requiring the data to be forwarded to the core, irrespective of the RD bit, switches in the tree are led to believe that there are always receivers on the other side of the core.

### 3.4 Core Initiated Joins

Using a single tree means that several group members can be added in one step, initiated by the core. In a scheme using per-sender trees, this is not possible: either each sender has to set up a tree to all the receivers, which means that the set of receivers has to be communicated to the senders, or the receivers join the sender tree of each of the senders, which means that, in turn, the receivers need to know the set of senders. This can be an important performance consideration for applications that depend on rapid setup of centrally controlled groups. An example would be the setting up of a group teleconferencing session with the help of a centralized server. The function of the centralized server may be to provide security screening, the appropriate translation functions needed for the different participants and other coordinating functions prior to the start of the session. Unlike "simple teleconferencing" with member-initiated joins, we also envisage a need for a small set of multicasting applications that are centrally coordinated and managed. Here a core-initiated join is advantageous.

### 3.5 Short-Cutting

This is the second building block of SEAM, wherein we avoid having all the transmissions go to the core (which may be an end-system, or a switch with cut-through capability) before being forwarded to the receivers. Anytime a cell is received at a switch S on a VC H, and if the context for VC H has been established (i.e., the switch knows it is in the spanning tree for conversation H), then the switch forwards the arriving cell on all the links of the spanning tree other than the one it was received on. This is the concept of Reverse Path Forwarding (RPF) that protocols such as DVMRP exploit as well. The only constraint is that the forwarding is only on the links where the "Receivers Downstream" (RD) bit is set. We leave the issue of how the spanning tree is built to the routing protocol.

Short-cutting is enabled by modifications in the signalling path of the switch implementation, and does not require any changes in the data path. Figure 6 illustrates how forwarding tables have to be set to achieve short-cutting.

In principle, SEAM also allows reconfiguration of "segments" of the shared tree upon link failure, rather than reconfiguration of the entire shared tree. This may require appropriate signalling support to not 'clear' the entire multicast call upon a link failure: only those receivers/senders downstream of the failed link need to be notified and requested to re-issue joins towards the core (with an alternate path provided by the routing infrastructure). This enhances reliability

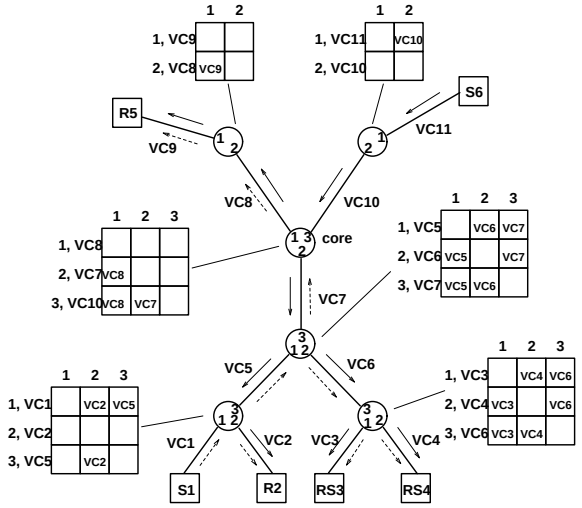


Figure 6: Forwarding tables for short-cutting. The paths followed by a packet from S1 and from S6 are shown.

and scalability.

## 4 Data Forwarding Issues

For our multicast scheme to work, we need to be able to map multiple incoming VCs into one or several outgoing VCs at switches. If this is done simply on a cell-by-cell basis, then cells belonging to different packets will interfere with each other (they will be interleaved, resulting in packet corruption).

One way to circumvent this problem is by reassembling the packets, perform packet-level scheduling, and re-segment one packet after the other onto an outbound point-to-multipoint VC (or mesh of point-to-point VCs). A multicast server typically used to do this function of reassembly and forwarding [7] makes for an obvious performance bottleneck and means that switches have to process packets. We show in this section that it is possible to achieve the same without reassembly and segmentation, by taking advantage of the fact that the AAL5 end-of-packet identifier is part of the ATM cell header.

A handle H for the group is used at the time of signalling to set up the SEAM VC. There is a one-to-one correspondence between the handle H and the VC on each individual link. Although the VC id may be different on each link, we will use the same term "H" for the SEAM VC, whenever this does not lead to confusion.

When multiple senders transmit to the same multicast group, identified by the handle, then their packets arrive on the same VC. The constraint imposed by ATM is that the data on a particular VC is ordered, and therefore, there is no need to identify cells as belonging to a particular packet. With AAL5, when a end-of-packet cell is received, all the previous cells received on that VC belong to that packet. When multiple senders send packets on the same VC, these need to be unambiguously ordered and forwarded so that there is no corruption of the data transmissions. We do this by having switches perform a function we call *cut-through forwarding*. Switches performing cut-through forwarding complete whole packets at a time, while buffering incoming packets from other input ports until an End-of-Packet (EOP) cell has been forwarded. Then, another port with an incoming VC for this group is selected for forwarding.

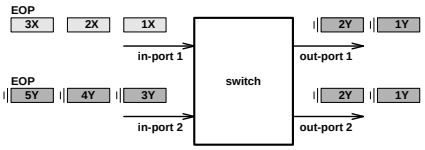


Figure 7: Cut through for VC H: Packet Y gets forwarded, while Packet X is buffered until cell 5Y with EOP for packet Y goes through switch.

With cut-through, the receivers do not have to distinguish cells of different packets arriving on the same VC (an impossible task). The specific actions for cut-through at a switch S are: the first cell of a packet arriving from any input port on VC H determines that this packet arriving on that input port gets unconditional priority to be forwarded on the outgoing VC H. Let this packet be Y from source B. Then, all of the cells of packet Y are forwarded first (e.g., Figure 7). Any other packet arriving on any other input port is queued at switch S for forwarding subsequent to the transmission of packet Y. When the last cell of packet Y (signified by the end-of-packet cell) is transmitted, then the cells queued for packet X are transmitted from switch S on the spanning tree. From that point onwards, packet X gets priority for being transmitted on VC H. A switch that is not a "merge point" would perform pure cell-switching for the multicast VC, even when following the above actions. Unicast VCs are unaffected by cut-through.

Thus, our requirement on switch S performing cut-through is to identify the first cell of an incoming packet on a given multicast VC H, and to transmit cells received on that input port only, until the last cell of that packet has been transmitted. The cells from other input ports that arrive in the meanwhile on VC H are queued for forwarding subsequent to sending the last cell of the packet currently being forwarded. Once the EOP cell is forwarded, another input port with cells awaiting transmission for the multicast VC is selected on a round-robin basis for transmission. Every switch (at least every *merge point* for multicast communication) on the tree is expected to be able to per-

form cut-through. We assume the support of per-VC queueing in the switches at least for multicast VCs.

Thus, a requirement we impose on switches is the need to recognize the EOP bit (for an AAL5 packet) to enable cut-through. AAL5 support of this nature is becoming more and more prevalent in ATM switches. This is because the notion of packet-discard is becoming a necessary aid to manage congestion, requiring switches to recognize the EOP flag. It is precisely this feature we exploit to achieve the ability to cut-through, in addition to the storage of cells of a packet, while a previous packet is being forwarded for that VC. A similar mechanism is briefly described in [10] in the context of multicast servers (called "hardware resequencers"). However, implementation issues have not been studied in detail.

The *fundamental* advantage of SEAM achieved with cut-through is that we do *not* have to look at the payload of the cell to do efficient multipoint-to-multipoint communication, and no changes are required for the cell header or AAL5 trailer.

Another observation we make is that LAN Emulation and IP Multicast over ATM (wherein a framework of point-to-multipoint VCs is used in conjunction with a multicast server to achieve the multipoint-to-multipoint service) use multicast servers. It is anticipated that switch vendors will begin to incorporate the server in the switch itself, so as to make ATM more attractive for deployment in traditional LAN environments. These switches will therefore incorporate the ability to switch frames. SEAM can exploit this functionality, and take advantage of switching frames only when essential (when there are multiple packets for the same SEAM VC contending for an output port of the switch), and allowing the switch to take full advantage of cell-switching whenever possible. Furthermore, we avoid reassembly and segmentation.

### 4.1 Recovery from Lost EOP Cells

Loss of an end-of-packet (EOP) cell causes cut-through to have somewhat more serious consequences than the unicast case. Just as in the unicast case, the loss of an EOP cell results in that packet not being successfully reassembled at the destination. That packet is lost. However, in multicast with SEAM, the loss of an EOP cell for a packet on a given input port also results in the continued queueing of the cells for that multicast VC on other input ports. Only the transmission of a subsequent packet on that same input port (and the forwarding of its EOP cell) would result in the queued cells on other ports being released. An approach for recovering from the loss of an EOP cell is to use a timer. If no cell (including an EOP cell) has been received for that multicast VC on an input port which is currently forwarding a packet for a period of time, then we may time-out that input port. The actions taken upon timeout would be:

1) To generate an EOP cell for that VC with a "null payload" so as to complete the packet loss event. This allows other switches to avoid the overhead associated with identifying the packet loss and having a timeout.

2) Once the EOP cell is forwarded, then allow the normal process of cut-through to continue. Another input port with cells awaiting transmission for the multicast VC is selected on a round-robin basis for transmission.

The setting of the timer for the multicast VC would have to be appropriate so that the loss of an EOP cell does not result in excessive delays. This is a function of the link speeds, and certainly based on the minimum rate that the VC may transmit at. One timer would be required per multicast VC at each independent forwarding point (one per SEAM VC for the switch, if it is an input buffered switch; or one timer per SEAM VC for each output port, if it is an output buffered switch).

### 4.2 When to Perform Cut-Through

When there is a slow input port on a switch with heterogeneous ports (of different speeds), a pure cut-through design may lead to unnecessary delay for cells arriving on the higher speed input ports for the same SEAM VC. Consider a switch with a fast output link and two input links, one that is fast (comparable to the output link) and another that is a slow input link. A packet arriving from the fast input link is able to saturate the output link. We can cut-through such a packet without potentially wasting downstream bandwidth by blocking packets for the same group arriving on other ports. On the other hand, a packet arriving on the slow incoming link cannot saturate the output port. During the time this packet uses the output port, link bandwidth proportional to the bandwidth difference between the fast output port and the slow input port is lost. Ideally, we would like this lost bandwidth to be exploited by packets concurrently arriving on the fast link. Therefore, we propose to store-and-forward packets arriving on incoming links that cannot saturate the output links. In other words, these input links would receive an entire packet before contending for access. More specifically, we suggest to perform cut-through on input port $i$ if

$$C_i \geq \min_{o \in O}\{C_o\} \quad (1)$$

and store-and-forward otherwise, where $C_j$ is the link bandwidth of port $j$, and $O$ is the set of output ports for this multicast group.

When there are multiple "active" input ports, then the switch follows a round-robin policy (motivated by

fairness issues) to determine which input port's cells should be forwarded. A "cut-through port" is active if it has at least one cell queued. A "store-and-forward port" is active if it has at least one packet queued.

## 5 Conclusions

SEAM is an efficient scheme for multipoint-to-multipoint communication in ATM networks, with the potential to scale up to large, dynamic groups. SEAM uses a single shared spanning tree for all senders and receivers.

We proposed the use of a unique handle (translates to a single VC on a link) to identify all packets associated with a given multicast group. Each multicast group has an associated "core", which is used as the focal point for routing signalling messages for the group. The handle is a tuple (core address, unique-id at core) that is unique across the network. We allow for member-initiated joins to the single core-based tree by senders and receivers as well as core-initiated joins. We exploit the proposed capabilities of leaf-initiated joins in ATM networks specified in UNI 4.0.

We compared through simulation the performance differences for member joins in a shared tree versus source-based trees. We observed that the SEAM join latency for a new member is significantly smaller than with source-based trees because the join travels far fewer hops. The consumption of resources by joins reduces (especially for senders) with a shared tree as the group size increases (for a given network topology). This is in contrast to a dramatic increase in resource consumption by joins (especially for receivers) in a source-based tree.

We introduced two new features in SEAM: "cut-through" forwarding and "short-cutting" to achieve efficient multicasting with low-latency for communication between senders and receivers. Cut-through forwarding in a switch enables the mapping of several incoming VCs into outgoing VCs at a switch. A switch capable of cut-through forwards a multicast packet from one input port at a time, taking advantage of the AAL5 end-of-packet cell to identify when to "switch" to forwarding a new packet. Incoming packets from other input ports are buffered until it is their turn, thus ensuring packets are transmitted on an outgoing VC atomically. Our preliminary simulation results indicate that the buffer requirements imposed by cut-through are modest. Short-cutting is the signalling mechanism allowing packets to follow the shortest path along the shared tree by emulating reverse-path forwarding. We avoid the packet having to go all the way to the core and then be forwarded back to the receivers, and therefore avoid it traversing links twice. No changes are needed to the cell header.

We believe no scheme is complete without addressing issues of migration. We have worked out the details of a SEAM-based environment working quite efficiently along with islands of non-SEAMable switches. The interoperability is such that only "boundary" SEAM switches need to be concerned with non-SEAMable islands, and within those islands, we fully exploit the point-to-multipoint capabilities of current ATM signalling.

Thus, we believe we have proposed a truly scalable and efficient ATM multicasting architecture. Issues of traffic management and reliable multicasting, both important topics, are the subject of our future work.

## References


[1] S. E. Deering, "Multicast Routing in Internetworks and Extended LANs," in *Proc. ACM SIGCOMM '88*, (Stanford, Calif.), August 1988.

[2] J. Moy, "Multicast Extensions to OSPF," in *Request For Comments 1584*, (Network Working Group, IETF), March 1994.

[3] P. Samudra, "UNI Signalling 4.0 (draft), ATM Forum/95-1434R9," Dec 1995.

[4] K. P. Birman, "The Process Group approach to reliable distributed computing," *Communications of the ACM*, December 1993.

[5] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT)," in *Proc. ACM SIGCOMM '93*, (San Francisco, Calif.), September 1993.

[6] S. Deering *et al.*, "An Architecture for Wide-Area Multicast Routing," in *Proc. ACM SIGCOMM '94*, (London), August 1994.

[7] G. J. Armitage, "Multicast and Multiprotocol support for ATM based Internets," *ACM Sigcomm Computer Communication Review*, vol. 25, April 1995.

[8] B. M. Waxman, "Routing of multipoint connections," *IEEE JSAC*, vol. 6, no. 9, 1988.

[9] L. Wei and D. Estrin, "The Trade-Offs of Multicast Trees and Algorithms," in *Proc. Int'l Conference on Computer Communications and Networks*, (San Francisco), September 1994.

[10] L. Wei, F. Liaw, D. Estrin, A. Romanow, and T. Lyon, "Analysis of a Resequencer Model for Multicast over ATM Networks," in *Proc. 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '92)*, (San Diego, Calif.), Nov 1992.