SEAM: An Architecture for Scalable and Efficient ATM Multipoint-to-Multipoint Communication

M. Grossglauser[a] * and K. K. Ramakrishnan[b]

[a]INRIA, BP 93, 06902 Sophia Antipolis Cedex, France

[b]AT&T Labs. Research, 600 Mountain Ave., Murray Hill NJ 07974, USA

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 including distributed interactive simulation (DIS), distributed databases and games, web caches and efficient support of IP multicasting. Our proposal, SEAM, uses a single VC for a multicast group consisting of multiple senders and receivers. SEAM relies on an additional switching feature called cut-through forwarding, which enables the mapping of several incoming VCs into one or several 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 from ATM's quality of service support and scalable bandwidth.

1. INTRODUCTION

Support for multi-party communications is viewed as a critical building block for enabling transparent, scalable and efficient communication. We believe the 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. 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 [5], MOSPF [8], and ATM UNI 4.0 [2].

Multicast also 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

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

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 problems, where requiring the application to track membership and manage member arrivals and departures results in complexity and an additional computational burden. Such applications include distributed interactive simulation (DIS), distributed databases and caches, and distributed games.

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 as a function of the total network size, the group size and composition, and the frequency of membership changes.

These requirements impose certain design choices. First, the multicast group has to be represented by a single tree shared by all group members (senders and receivers) [4]. Second, only network nodes (switches or routers) on the tree for a group should have to maintain state concerning that 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 can become excessive.

The only protocols proposed so far recognizing the need for a shared tree are CBT [4] and the sparse mode of PIM [6]. In ATM, member-initiated joins and leaves have been incorporated into ATM Forum UNI version 4.0 for point-to-multipoint communication [2]. However, there has been no proposal for a "convenient" multipoint-to-multipoint communication mechanism. Using a multicast server (e.g., an option in [3]), results in a single tree spanning all receivers, rooted at the server. Senders have to create point-to-point (unicast) connections to the server, violating the network state requirement above. Also, the server is both a bottleneck (due to the necessity to perform reassembly-segmentation in the server, and subsequent forwarding to all the receivers) 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.

One of the ancillary goals in our proposing SEAM was 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 being maintained and managed in the ATM backbone. SEAM overcomes this by having a single VC associated with each IP multicast address.

The remainder of this paper is structured as follows. Section 2 gives an overview of the SEAM proposal. Section 3 discusses signalling issues, and Section 4 discusses data

forwarding issues. Section 5 considers interoperability issues with the ATM Forum's UNI 4.0, and outlines a path for gradual migration towards SEAM. Section 6 concludes the paper.

2. THE SEAM PROPOSAL

The defining property of SEAM is a shared tree between all senders *and* receivers of a group. We use a "core" (as in [4]) as the root of the tree. Having a single shared tree per group has a number of important advantages. First, a group will allocate only one VC per link. Also, no per-sender state has to be maintained in switches.

Signalling is based on a *group handle*. A handle is a unique SEAM conversation identifier. The handle consists of the core address plus an identifier. The core address is necessary because it allows members and intermediate switches to know the core through the group handle. Note that to make the handle globally unique, it is sufficient to make the additional identifier locally (at the core) unique. Second, a simple signalling mechanism (termed *short-cutting*), allows cells to take short-cuts at each switch on the tree. In other words, each packet spans the shared tree from its sender to all the receivers, keeping delays low.

SEAM manages group members who are only senders, only receivers, or both, in the same way. All of these three types of members share one tree, rooted at the core. The tree's links are bidirectional channels. The core may be an ATM switch, not necessarily an end-system. Segmentation-reassembly is not required at the core and only occurs in the end-systems that are senders and receivers.

The shared tree is represented by a single VC. Thus, a switch at the "merging point" in the network (where transmissions from different senders on the multicast group arrive on different input links) has to be able to map multiple incoming VCs into one (or possibly more) outgoing VCs. The switch has to perform this without interleaving cells of different packets (thus not corrupting the packet), while not reassembling the packet and suffering the associated delays. Switches at "merging points" in the network transmit *all* the cells of a multicast packet, in order, downstream on the spanning tree before transmitting cells of the next packet for the same multicast VC. We call this function *cut-through*. When a packet on a given input port for a multicast VC is being forwarded, the switch is required to buffer other incoming packets for that multicast VC until the End-of-Packet (EOP) of the current packet has been forwarded. Thus, cells of a packet from an individual source are forwarded in order so that reassembly can be performed successfully at the destination. Cut-through affects only multicast VCs, and then only on a per-multicast VC basis. Cells transmitted on normal unicast VCs continue to be switched in the "true-ATM" mode, and see all the benefits of cell-switching.

An idea similar to cut-through was briefly described in [10], in the context of multicast servers (described as hardware resequencers), that may be distributed throughout the network. We describe the idea in greater detail in the context of a true multipoint-to-multipoint architecture that is proposed in this paper.

Quantitative analysis on the advantages of SEAM's shared tree in the context of ATM may be found in [7].

3. SIGNALLING ISSUES

We now define SEAM signalling support in more detail. The mechanisms used are quite similar to those defined in UNI 4.0 [2]. However, some extensions are necessary because group members may be senders or receivers, and because there can be several incoming and several outgoing multicast VCs at a switch.

3.1. Group Creation

When senders or receivers want to join a multicast group, they send a join message towards the core. 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 potential members. This can happen, for example, through a name service, as proposed in [4], 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.

The creation of the group must be preceded by the selection of an appropriate core. This may be achieved in two ways, namely (1) through a configuration server that resides on a well-known address, or (2) by taking advantage of routing information. We note that, in addition to disseminating reachability information, the routing system may also distribute core location information. Since every switch is a potential core, the core selection may be elegantly integrated into the distribution of reachability information. The core selection may be based on an a-priori known set of participants, or on a set of ATM addresses (that may or may not represent actual future group members) that are used as a "geographical hint" to assist core selection.

Assuming that core selection is integrated into the routing system, the initiator simply provides the first hop switch with a set of ATM addresses of potential or actual group members, and requests a core address.

Initiator -> [geog_hint=list<ATM address>] -> Next hop switch

The next hop switch returns the address of the core switch.

Next hop switch -> [core=ATM address] -> Initiator

The next hop switch may in fact return a list of cores, and the initiator may choose one based on additional local considerations.

Note that this first phase is not necessary if the initiator, for some reason, already knows the ATM address of a core, for example from a previous group. Now the initiator requests the group setup from the core. The core defines the unique group handle as (core-address, unique_id). The unique id may be similar to (if not the same as) the Call Identifier field used in the "Global Call Identifier" (GCID) in UNI 4.0 to uniquely identify a "Leaf-Initiated

Join (LIJ) Call". The initiator also provides the list of the initial set of participants (if any) and flags that define them as senders or receivers, so that the core can add them to the group at setup time (through core-initiated join).

Initiator -> [initial_members=list<ATM address, snd_flag, rcv_flag>] -> Core

Core -> [group_handle=(core_address, unique_id)] -> Initiator

The dissemination of group handle information to members is the initiator's responsibility and not part of SEAM. When SEAM is used to carry IP multicast, then the group identification may include the IP multicast address associated with the group as well.

3.2. Short-Cutting

Short-cutting is a signalling-level mechanism that sets up cell forwarding tables in such a way that reverse path forwarding (RPF) is emulated at each switch. This allows a packet sent to the group to span the tree, instead of going first through the core and then back down the receiver-spanning tree as in a multicast server approach. This reduces delay and ensures that packets traverse any link at most once.

Short-cutting is enabled by modifications in the signalling path of the switch implementation, and does not require any changes in the data path. We introduce the notion of an SD-bit, (for "Sender Downstream"), which is a per-port flag indicating if there are senders downstream of that port for the corresponding group. A sender join coming in sets the SD-bit on that port. When an SD-bit is set, the port is mapped on to all the other ports that have a "Receiver Downstream" (RD-bit), bit set. The RD-bit is a per-port flag indicating that there are receivers downstream of that port for the corresponding group.

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 and scalability.

3.3. Member Initiated Joins

It is obvious from the discussions in [4,6] 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 travelling smaller distances), processing resources in the switches, and reduced latency [7].

We discuss the mechanisms for member-initiated joins without root-notification. Joins with root notification are simple extensions, except that the root is the core in SEAM. A member join is initiated by a "Leaf Setup Request (LSR)" towards the core, using the group handle as part of the GCID field. The member uses its ATM address to identify itself, including two flags identifying it as a sender or a receiver. The leaf setup request progresses along the shortest path to the core until it reaches a switch at which the context

for the SEAM call exists (i.e., a switch that is already on the tree for this group). A receiver join request may have to progress further up if this switch is not on the receiver-spanning tree and is only on a branch of the tree with downstream senders.

A standard "Setup" procedure is initiated from the tree to extend it to the new member, with a "Setup-Connect" message exchange between the switch on the tree and the new member as the leaf. Cell forwarding state is set up at the switches for the group so that short-cutting is enabled everywhere in the tree. We now define the rules for manipulation of the cell forwarding state by the switches. First, some notation. We call RP (Request Port) the port on which the LSR has come in, i.e., the port that is on the shortest path towards the new member. We call DP (Designated Port) the port on the shortest path towards the core (analogous to the "root port" in bridges). The DP has its RD-bit and SD-bit set to 1 by default, to ensure that all the packets sent to the group are forwarded to the core, and that receivers get all the packets coming from the core. Finally, when we create forwarding state for a given SEAM group from the incoming VC on port A to an outgoing VC on port B, we will say that we map port A into port B.

We describe the actions at a local switch for sender and receiver where port 1 is the DP.

- *Sender-join*: Map RP to all the ports with the RD-bit set to one. For example, consider a sender-initiated join arriving at port 2 of a switch, which has the RD bit set on ports 3 and 4. The VC for port 2 will be mapped into ports 1 (the DP), 3, and 4.
- *Receiver-join*: Set the RD-bit of RP to one. Map all the other ports (with SD = 1) into RP. Let us consider an example where ports 4 and 5 of the switch have senders for the multicast VC. A receiver join at port 2 causes ports 1 (the DP), 4, and 5 to be mapped into port 2.
- *Sender&Receiver join*: apply both rules above.

We now discuss the forwarding of join requests. Forwarding of join requests is always on the shortest path towards the core, i.e., switches always forward the requests on the DP.

- *Sender join*: Forward a join until it reaches a switch that is already on the tree.
- *Receiver join and Sender&Receiver join*: Forward a join until it reaches a switch on the tree which has some port other than the RP and the DP with the RD-bit set.

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 for each of the senders, which means that 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. Here, a "core-initiated join" may prove advantageous.

The request messages for a core-initiated join always travel all the way from the core to the new member. At each switch, we simply check if a corresponding member-initiated join request would have traveled up to this switch or not. If it had, then the state setup rules defined above are applied in the same way, as if the corresponding request had been received on the port towards the member.

3.5. Call Teardown

Upon receiving a leaf-initiated release request, the appropriate cell forwarding state associated with the SEAM VC needs to be removed, and the request forwarded up the tree as needed. Deleting the state is simple: simply delete all mappings from other ports into the RP to release a receiver, and delete all mappings into other ports from the RP to release a sender.

In order to decide about forwarding the release request further upstream, we need to check if the above state deletion has caused "loose ends" on the DP, i.e., either a VC on the DP not being mapped anywhere, or a VC on the DP not having any other port mapped into it. If such loose ends exist, then the appropriate release request is forwarded upstream, i.e., a receiver release request in the former case, a sender release request in the latter.

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, resulting in packet corruption.

One way to circumvent this problem is by reassembling the packets at a multicast server, 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). We show in this section that it is possible to achieve the same without reassembly and segmentation, by taking advantage of the AAL5 End-of-Packet (EOP) identifier, which is part of the ATM cell header.

The packets of multiple senders transmitting to the same multicast group 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 EOP 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 to avoid corruption. We do this by having switches perform a function we call *cut-through forwarding*. Switches performing cut-through forwarding complete packets at a time, while buffering incoming packets from other input ports until an EOP cell has been forwarded. 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 X from a given source. Then, all of the cells of packet X are forwarded first. Any other packet arriving on any other input port is queued at switch S for forwarding subsequent to the transmission of packet X. When the last cell of packet X (the EOP cell) is transmitted, then cells queued for another packet, Y, are transmitted from switch S on the spanning tree. From that point onwards, packet Y 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, by following the above rules.

Once the EOP cell is forwarded, another input port with cells awaiting transmission for the multicast VC is selected for transmission on a round-robin basis. Every switch (at least every merge point for multicast communication) on the tree is expected to be able to perform cut-through.

Just as in the unicast case, the loss of an EOP cell results in that packet not being successfully reassembled at the destination, and hence being 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 "time-out". 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. More details may be found in [7].

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. In such a case, one may configure the switch to disable cut-through on slow input ports for the SEAM VC, and instead perform store-and-forward on a per-packet basis. When a packet has been received in its entirety on the SEAM VC on the slow input port, then the input port would contend for forwarding the cells of the packet just as a port performing cut-through. This allows faster input links to send a packet (or even more than one packet) using cut-through while the switch is receiving a packet on the slow link. When the "slow packet" is completely received at the switch, it becomes a candidate for transmission, and contends with the other input links. This reduces the impact on other input links, and potentially increases the output link utilization (when the flows are predominantly multicast). More details are in [7].

The *fundamental* advantage SEAM achieves with cut-through is that we do *not* have to look at the payload of the cell to do efficient multipoint-to-multipoint communication.

4.1. Performance Evaluation of Cut-Through

SEAM takes advantage of cut-through to perform cell-by-cell forwarding for a multicast VC. When only a single multicast packet is arriving at a switch, we can take full advantage of ATM switching and the associated latency improvements. When multiple packets for the multicast VC are being received at a switch simultaneously, we queue packets from other input flows while forwarding, on a cell-by-cell basis, one of the packets. As a result, we will observe latencies always better than packet-by-packet forwarding at the "merge

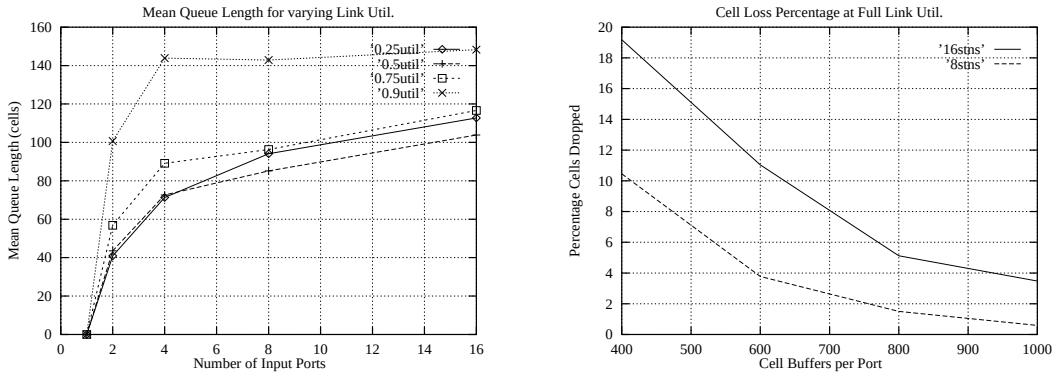

Figure 1. (a) Mean Queue Length per Port vs. No. of Input Ports for varying output link utilization. (b) Cell Loss Probability vs. Cell Buffers per Port for 8 and 16 port switches.

point" in the shared tree. From a packet forwarding perspective, cut-through gives the same latency advantages as cut-through bridging did for packet networks. The primary performance metric we examine here is the mean queue length at each input port.

The use of SEAM results in a slight increase in buffering at an input port compared to pure cell switching. We briefly examine the buffering requirements imposed on switches as a result of implementing cut-through through simulations. We model a single switch with varying number of input ports receiving packets addressed to the common multicast VC. The switch forwards all received data over a single output port at 155 Mbits/sec. The switch implements the data forwarding aspects of cut-through: i.e., forwards cells from a given input port until the EOP cell is forwarded; the next input port is selected on a round-robin basis after the current packet has been forwarded entirely.

If we were to assume that the packet sizes were fixed, the packet arrival instants were deterministically distributed, and the arrivals would not exceed the total capacity of the output link, then the buffer requirement per port would be minimal. At most one maximum-sized packet's worth of buffering would be needed per input port. With "perfect" traffic management of the entire multicast group, we could expect it to be the case, especially if even the short-term rate of cell arrivals across all input ports does not exceed the capacity of the output link. Some of the goals of explicit rate algorithms for ATM's Available Bit-Rate Service [1] make an attempt at it, albeit for unicast connections.

We also need to understand the buffer requirements when there is variability in both the packet inter-arrival time and the packet size. For our simulations, the inter-arrival time of packets is exponentially distributed. The size of the packet is uniformly distributed in the range of (2,200) cells. This is approximately the range of IP packet sizes anticipated over ATM links. We control the mean aggregate packet arrival rate to not exceed the capacity of the output link. However, both variance in the packet inter-arrival time (exponentially distributed) and the packet size (uniformly distributed) contribute to queueing at the switch. The cells of a packet arrive at the *full link rate*, so that we have the maximum burstiness within a packet. We examine the buffering requirements for varying levels of aggregate utilization of the output link, ranging from 25 % to 90 % of the link capacity. The number of sources (number of input ports on switch) is varied from 1 to 16.

Figure 1(a) shows the variation of the mean queue length for varying numbers of input

ports, for 4 different average utilization levels of the output link. The mean queue length increases as the number of ports increases. Even when the output link is 90 % utilized, the mean queue length is within 200 cells (one maximum size packet) for the 16 port case. Thus, the average buffer requirement appears reasonable. We also looked at the maximum queue length per port (realizing that this is not a good measure to examine, in a statistical sense): it was about 1550 cells for the 16 port case, when the utilization was 0.9. Given the variability in the packet inter-arrival time distribution and the cell-level burstiness, this requirement of no more than 8 packets for the total buffer is reasonable. Realize our simulations are as if the ATM network was operating with no congestion control (except that there is no long-term overload).

We also examine the cell loss probability at high average link utilizations, when the number of cell buffers is varied from 200 cells to 1000 cells (5 max. size packets), in Figure 1(b). For the 16 port case, the link was utilized 94.11 % and for the 8 port switch, 88.88 %. The cell loss probability is higher, about 19 %, for the larger switch (since the link is shared by more ports) but goes down fairly rapidly, to about 3 %, with increasing buffer sizes. One can anticipate that the actual *packet loss* probability could be reduced substantially by using mechanisms such as partial packet discard [9], since the switch is already keeping track of the EOP cell processing.

5. INTEROPERABILITY: MIGRATION FROM UNI 4.0 TO SEAM

As SEAM requires the cut-through capability of ATM switches, its deployment may be gradual. It is therefore important to provide for interoperability between SEAM switches and non-SEAMable switches. In this section, we propose an interoperability architecture between non-SEAMable *islands*, (i.e., one or several switches not having the cut-through capability) and SEAM environments.

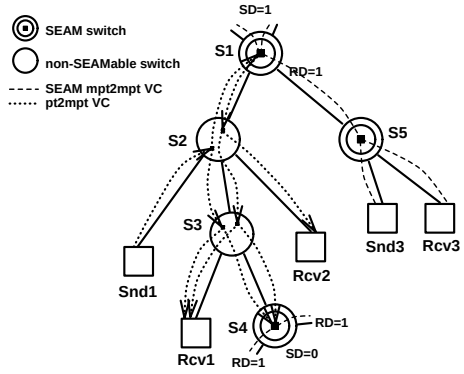

Figure 2. Interoperability between SEAM-able & non-SEAMable switches.

Consider the situation given in Fig. 2. There is an island of two non-SEAMable switches, S2 and S3. The questions we have to address are:
(a) How do we connect SEAMable switches having a SEAM multipoint-to-multipoint VC belonging to the same group through the island?
(b) How do we connect senders and receivers inside the island to the SEAM group? We

assume that the switches in the non-SEAMable island have the capability of setting up point-to-multipoint VCs.

To efficiently use signalling resources, we use the SD-bit (sender-downstream) at each SEAMable switch. It allows a SEAMable switch to set up connectivity to the non-SEAMable island for forwarding transmissions of the group from senders on the SEAM environment when such senders are present. The solution we propose can be summarized as follows.

1. Each sender (Snd1) in the non-SEAMable island sets up a point-to-multipoint VC reaching all island receivers (Rcv1, Rcv2) as well as all border SEAM switches having the RD bit set on at least one port, other than the one connected to the island (e.g., at switches S1, S4).

2. Each border SEAM switch having the SD bit set on at least one other port than the one connected to the island (e.g., S1) sets up a point-to-multipoint VC to all the non-SEAMable island receivers (Rcv1, Rcv2) as well as the other bordering SEAM switches having the RD bit set on at least one other port than the one connected to the island (e.g., S4). If no SD-bit is set, then the point-to-multipoint VC into the island rooted at that switch is not necessary. This avoids the signalling message making progress and a VC being set up unnecessarily.

3. Border SEAM switches (S1, S4) map all of the incoming point-to-multipoint VCs from the island into the SEAM multipoint-to-multipoint VC on the other ports. Reverse Path Forwarding is done *per port*, i.e., an incoming VC from the island is not forwarded into the point-to-multipoint VC to the island on that port.

4. Border SEAM switches map the point-to-multipoint VC into the island into the SEAM multipoint-to-multipoint VC as well.

This scheme has the following properties. First, SEAM switches not connected to a non-SEAMable island do not need to have any knowledge about the island (e.g., S5). Second, border SEAM switches (S1, S4) need to know about all of the island senders (e.g., Snd1) (in order to do a leaf-initiated join to the point-to-multipoint VC rooted at these senders) as well as about other border SEAM switches (for the same purpose). The same holds for island receivers (Rcv1, Rcv2). Basically, inside the island, the situation is as if there were no SEAM multipoint-to-multipoint VCs. Border SEAM switches are, in the general case, both senders and receivers for the non-SEAMable island. Third, as we "emulate" short-cutting inside the island with multiple point-to-multipoint VCs, we get the same delay as if there were only SEAM switches. Fourth, scalability in terms of sender population is obviously not achieved inside the island, as the number of point-to-multipoint VCs grows with the number of senders and bordering SEAM switches. However, the scheme is scalable in the number of islands; each island is isolated in the sense that it does not need to know about other islands.

We think that gradual migration from simple point-to-multipoint VCs with leaf-initiated joins, as proposed in UNI 4.0, to SEAM capable switches will be straightforward. Since deploying SEAM switches enhances scalability in sender population, a sensible strategy would be to place SEAM switches first at points where a large number of senders are expected.

6. CONCLUSIONS

We proposed an efficient scheme, called SEAM, for multipoint-to-multipoint communication in ATM networks. The scheme allows for scaling up to a large number of potential senders and receivers. SEAM uses a single shared spanning tree for all senders and receivers, thus using a single VC for the entire group. The key mechanism of SEAM is called *cut-through forwarding*. A SEAM switch forwards complete packets for the multicast VC, taking advantage of the AAL5 EOP bit to determine when to switch to forwarding a new packet. Non-SEAM VCs are unaffected by cut-through.

We have discussed the details of SEAM's signalling extensions. Joins and leaves for senders and receivers are fully symmetric from the user's point of view. We have argued that SEAM puts a minor additional burden on the switch design, while greatly enhancing the scalability of ATM multicast. We have also presented some simulation results indicating that the buffering requirements using SEAM are modest. Finally, we have shown that a gradual migration to SEAM is possible, by defining how SEAM should interoperate with non-SEAMable islands.

REFERENCES

1. ATM Forum Traffic Management Specification Version 4.0. ATM Forum Specification /af-tm-0056.000, ATM Forum, April 1996.
2. ATM User-Network Interface (UNI) Signalling Specification Version 4.0. ATM Forum Specification /af-sig-0061.000, ATM Forum, July 1996.
3. G. J. Armitage. Multicast and Multiprotocol support for ATM based Internets. *ACM Sigcomm Computer Communication Review*, 25(2), April 1995.
4. Tony Ballardie, Paul Francis, and Jon Crowcroft. Core Based Trees (CBT). In *Proc. ACM SIGCOMM '93*, San Francisco, Calif., September 1993.
5. S. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proc. ACM SIGCOMM '88*, Stanford, California, August 1988.
6. S. Deering et al. An Architecture for Wide-Area Multicast Routing. In *Proc. ACM SIGCOMM '94*, London, August 1994.
7. M. Grossglauser and K. K. Ramakrishnan. SEAM: Scalable and Efficient ATM Multicast. In *Proc. IEEE INFOCOM '97*, Kobe, Japan, April 1997.
8. J. Moy. Multicast Extensions to OSPF. In *Request For Comments 1584*, Network Working Group, IETF, March 1994.
9. A. Romanow and S. Floyd. Dynamics of TCP Traffic over ATM Networks. *IEEE JSAC*, 13(4):633–641, May 1995.
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, CA, Nov. 1992.