SEAM: Scalable and Efficient ATM Multipoint-to-Multipoint Multicasting [Extended Abstract]

M. Grossglauser

INRIA
BP 93
06902 Sophia Antipolis Cedex, France
Matthias.Grossglauser@inria.fr

K. K. Ramakrishnan

AT&T Research
600 Mountain Ave.
Murray Hill, NJ 07974-0636, USA
kkrama@research.att.com

Abstract

Human collaboration applications and distributed systems are expected to benefit from a group multicast service, both in terms of performance and design simplicity. We argue that such a service must be scalable both in the number of potential senders and receivers. The success of ATM will in part depend on the availability of an efficient group multicast service.

We provide a brief description of the requirements for a general multicast service, and then present SEAM, a proposal for scalable and efficient multicast in ATM. SEAM relies on an additional switching feature we call cut-through forwarding, which enables the mapping of several incoming Virtual Channels into one or several outgoing Virtual Channels. SEAM relies on a shared tree spanning all senders and *receivers of the group. It allows for centrally initiated group setup as well as dynamic group membership changes. An additional feature, "short-cutting", allows for the transmission of a packet to follow the shortest path along this shared tree.*

We believe that SEAM is both an important and necessary step in the evolution of ATM. It will enable a class of applications relying on group multicast to benefit directly from ATM's quality of service support and scalable bandwidth and the resulting performance advantages.

1 Introduction

Networking applications can benefit in terms of scalability, performance and design simplicity from a group multicast service, i.e., a service enabling multiple senders to communicate to multiple receivers. Examples of multicasting on a broader scale include human collaboration such as videoconferencing and shared workspaces. Many applications in Local Area Networks (LANs) have often taken multicasting for granted (e.g., for address resolution, resource discovery, etc.). We believe that the need for multicasting will become even more pressing with the wide-scale deployment of distributed systems (e.g., maintaining replicated databases). An important need for multicasting is to be efficient and have the ability to scale up, both in the number of senders and receivers.

The development of Asynchronous Transfer Mode (ATM) networks is fuelled by the need for efficient utilization of wide-area network resources, scalable bandwidth and support for quality of service. The underlying mechanism is the use of Virtual Connections (VC), where state for conversations in progress is maintained in the network's switches. The natural way VCs are set up is to associate state for a sender-receiver pair. While this is suitable for unicast communication, it becomes state-intensive to use the same method for multicast. Work in the ATM Forum to overcome this using point-to-multipoint VCs has been ongoing. However, further work is needed to be able to accomodate a large number of senders *and* receivers participating in a group. The use of multipoint-to-multipoint communication is highly desirable.

Cell switching, i.e., having forwarding units smaller than packets, has been introduced to simplify switching. Having small, fixed sized cells considerably simplifies the switch's hardware architecture, and provides for higher performance. However, breaking up packets into smaller units has its drawbacks, such as the much-cited cell versus packet loss probability problem. Another complication occurs in multicast: the fact that we manipulate sub-packet units means that we need to be careful when forwarding from more than one incoming VC into one (or several) outgoing VCs.

In this paper, we show that the cell-switched nature of ATM does not impede such group-style communication. We propose a flexible multicast architecture for ATM that is both scalable and efficient.

This extended abstract is structured as follows: In the next section, we briefly describe the needs of a general multicast service. We then outline the SEAM scheme in Section 3 and present its details in Section 4. Section 5 concludes the paper.

2 Requirements of a General Multicast Service

In this section, we derive the requirements of a desirable multicast service, based on the motivations that has encouraged the use of multicast in communication networks.

Let us briefly summarize some advantages of using a multicast service over using a collection of point-to-point (unicast) links. The most obvious reason is that of bandwidth usage: a packet sent to a number of receivers will traverse each link only once if multicast is used, because the packet is replicated only when the paths to the receivers diverge. However, in the unicast case, multiple copies of the same packet can traverse the same link multiple times. Another motivation to use multicast is the group abstraction that such a service can provide. Senders and receivers need not be aware of the group membership situation. For them, the group exists as a single object that they can address as a single entity. The availability of such a service often simplifies the design and implementation of distributed systems, as membership information is decoupled from the application. For example, the application does not need to be informed about dynamic membership changes. A multicast service provides an essential glue to assemble distributed systems.

We believe that to have a good chance of success even for future applications, we believe that a multicast service has to offer the following: (1) Group management symmetry for senders and receivers; (2) Scalability as a function of the total network size, the group size, and the frequency of membership changes; (3) Distributed management, for example joins and leaves initiated by members (senders *or* receivers *or* both) that are invisible to other members.

3 Overview of the SEAM Scheme

We first give a broad overview of our proposal for multipoint-to-multipoint multicast in ATM in this section.

The defining property of SEAM is a shared tree between all senders *and* receivers of a group. We use the concept of a "core" (as in [1]) as the root of the tree to be set up. 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.

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.

This approach has several advantages. First, we achieve the desired symmetry between senders and receivers. In SEAM, a large sender population is no more of a scalability concern than a large receiver population. Second, with a small additional switching feature (called "cut-through"), we are able to use a single virtual channel (VC) for the entire tree, thus conserving this potentially scarce resource. Third, a simple signalling mechanism (termed "short-cutting"), allows us to make a modification to the way cells are forwarded in the switches: instead of packets first being sent to the core and then multicast back to the receivers, as in MARS [2], we can 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.

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.

We believe that a crucial ingredient in a proposal for an extension of an existing architecture is a strategy for migration, i.e., defining a way elements of the existing and the new architecture can interoperate. We have worked through a proposal for gradual migration, but have not provided the details here due to space limitations.

4 The SEAM Proposal

This section discusses the proposed architecture in more detail. Where possible, we relate the mecha-

nisms to the existing ATM standards or drafts.

4.1 Use of a Single Shared Tree

The fundamental goal of SEAM is to have a single shared tree between all senders *and* receivers of a group. This results in allocating a single VC per link for the entire group. This results in "traffic concentration" [3], which may actually turn out to be an advantage in terms of manageability. For example, it would be considerably easier to enforce fairness between senders within the group (i.e., fairly share bandwidth allocated to the group between senders sending to this group) or to limit the bandwidth used by an entire group (per-group fairness instead of per-sender fairness) if all the traffic uses the same tree. Also, consistency is achieved more easily: if all senders send on a common tree, then a receiver joining this tree is sure to receive from all the senders. If the group exists as a collection of sender-based trees, then it is not easy for the receiver to ensure that it has joined all of these trees.

Second, 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 all 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.

The major disadvantage of shared trees are delays that are potentially higher than in the case of shortest-path sender-based trees [4]. 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.

4.2 Multicast Group Creation

When senders or receivers want to join a multicast group, they need to 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 then 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 [1], 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.

4.3 Signalling for Member Initiated Joins

It is obvious from the discussions in [1, 3] 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, 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 [5], but we generalize it to both sending and receiving members. The same options as was proposed for UNI 4.0 (without sender participation, with sender notification, with sender permission) may be used.

4.3.1 Use of a Receivers Downstream (RD) Bit

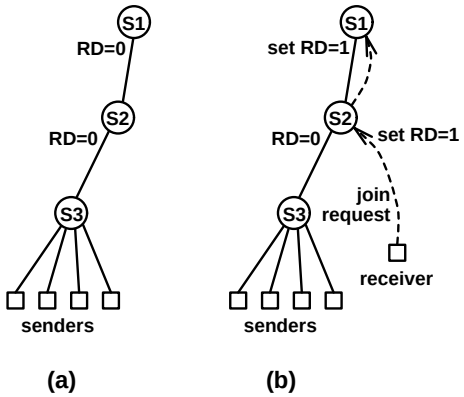

Figure 1: Updating of the RD bit upon receiver join.

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, such as 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. 1: 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 downstream of the core.

The remaining specific details of a leaf initiated join are very similar to the concepts specified in the UNI 4.0 draft on this topic.

4.4 Signalling for Core Initiated Joins

Member initiated joins are clearly necessary. However, we think that core initiated joins should be available as well. For example, if the initiator knows who the group members are going to be (e.g., because the application requires a well-defined set of members), then it would be much easier and more efficient for the initiator to be able to tell the core, upon setup, what hosts to connect to the group, instead of setting up the core, and then contact each member individually and invite the member to join.

4.5 Switch Support for Cut-Through

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 corruption of packets).

One way to circumvent this problem is by reassembling the packets, perform packet-level scheduling, and re-segment one packet after the other into the out-bound point-to-multipoint VC (or mesh of point-to-point VCs). A multicast server is typically used to do this function of reassembly and forwarding. This approach, suggested in [2], makes for an obvious performance bottleneck and means that switches (if implementing [2]) 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 dif-

ferent 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, these 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*. Switches performing cut-through forward complete packets at a time, while buffering incoming packets from other input ports until an End-of-Packet (EOP) cell has been forwarded.

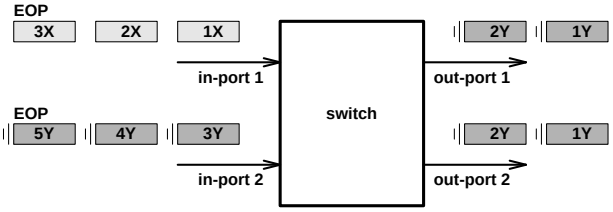

Figure 2: Cut through for VC H: Packet Y gets forwarded, while Packet X is buffered until cell 5Y with EOP for packet Y goes through switch.

Consider the case where two senders, A and B, are transmitting data packets X and Y to a set of receivers, as shown in Figure 2. These arrive at switch S, in the form of cells. If we just had packet-based networks, where packets were not being segmented into cells, the action of cut-through forwarding is simple: packet X would be transmitted and subsequently packet Y would be transmitted, both being identified by the handle H, which is the group handle. Whichever packet arrived first gets transmitted first. When we have ATM networks, senders A and B transmit packets X and Y, with the same VC, which for our purposes here can be considered to be H. The cells from packets X and Y, arriving at switch S, may be interleaved because the cell is the unit of transmission rather than a packet. Forwarding cells in the order received is undesirable. Rather, we mimic the behavior of packet networks. This way, the receivers do not have to distinguish cells of different packets arriving on the same VC (an impossible task). We do this by having the following actions at switch S: 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. Any other packet arriving on any other input port is queued at switch S for forwarding subsequent to the transmission of packet Y. For example, since all the cells of packet X are received after the first cell of packet Y is sent, these are queued. 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. The cells of packet Y are shown making progress on the output links of switch S in Fig. 2, because of this cut-through function.

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. We call this process cut-through, and every switch (at least every *branching point* for multicast communication, as defined in [6]) on the tree is expected to be able to perform this function.

There are several reasons we believe it is reasonable to expect ATM switches to be able to implement the cut-through feature that we propose here. First, switches that are capable to multicasting will likely follow a slightly different processing path for multicast packets in contrast to unicast packets. These points in the topology where replication has to be performed are the "branching points", defined in [6]. For example, for a cross-bar switch with input port buffering, there is inevitably a need for the input port to retain a cell until it can be replicated and transmitted across the cross-bar interconnection fabric to all the candidate output ports. In a switch using a batcher-banyan type interconnection network (Jon Turner's switch [7]), a concept of recirculation of a multicast cell is introduced so that the cell makes it to all the output ports when there is contention. In any case, the normal path may not be followed, and there is typically a need to store the cell until it makes its way to all the output ports.

The second requirement we impose on switches is

the need to recognize the end-of-packet (for an AAL5 packet) to enable cut-through, so that a subsequent sender's packet may then be forwarded. 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. The nature of ATM is such that, especially for AAL5 packets, if one cell is dropped, the rest of the packet is corrupted and the receiver is unable to reassemble the packet properly. As a result, ATM switches are encouraged to drop the rest of the AAL5 packet for a given VC, once it has dropped a cell from that VC. 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.

When there is a slow input port on a switch with heterogenous ports (of different speeds), a pure cut-through design may lead to unecessary delay for cells of this SEAM VC arriving on the higher speed input ports. To overcome this, one may configure the switch to disable cut-through on these slow input ports for the SEAM VC, and instead perform store-and-forward on a 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.

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. These servers may be workstations, which involves additional cost. It is expected 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 buffer packets, and forward in precisely the manner we describe, except that instead of a single switch-based server, we take advantage of every switch's ability to perform cut-through.

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.

4.6 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 in fact may be a switch with the capability to perform cut-through as described above) 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 etc. exploit as well. The only constraint is that the forwarding is only on the links where the "Receivers Downstream" (RD) bit is set (to accomodate the presence of asymmetry, where there are senders who are not receivers). We leave the issue of how the spanning tree is built to the routing protocol.

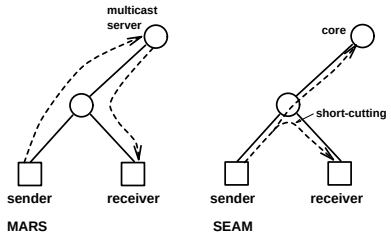

Figure 3: Multicast servers versus short-cutting in SEAM.

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 4 illustrates how forwarding tables have to be set to achieve short-cutting.

5 Conclusion

We proposed in this paper 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. There is 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 any packets associated with a given multicast group. The handle is a tuple (core address, unique-id at core) that is unique across the network. Each multicast group has an associated "core", which is used as the focal point for routing signalling messages for the group. We allow for leaf-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 introduced two new features in SEAM:

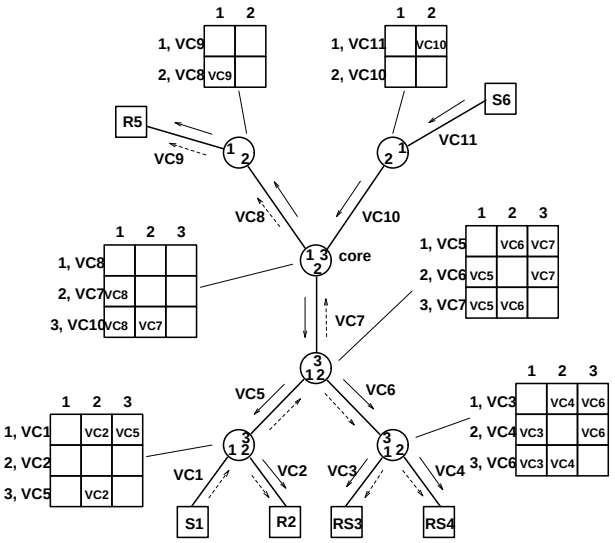

Figure 4: Forwarding tables for short-cutting. The paths followed by a packet from S1 and from S6 are shown.

"cut-through" forwarding and "short-cutting" to achieve efficient multicasting with low-latencies for communication between senders and receivers. Cut-through forwarding in a switch enables the mapping of several incoming VCs into one or more 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". A second feature, "short-cutting" is entirely in the signalling path in switches, and allows a packet to follow the shortest path along the shared tree spanning all senders and receivers of the group. We avoid the packet having to go all the way to the core and then be forwarded to the receivers, and therefore avoid it traversing many links twice.

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.

References

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

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

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

[4] 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.

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

[6] S. S. Sathaye, "ATM Forum Traffic Management Specification (Draft Specification of the ATM Forum), AF-TM 95-0013R8," Oct 1995.

[7] J. S. Turner, "An Optical Nonblocking Multicast Virtual Circuit Switch," in *Proc. IEEE INFOCOM '94*, pp. 298–305, June 1994.