Shared Access to Many-to-Many ATM Connections

Eric Gauthier, Jean-Yves Le Boudec, and Philippe Oechslin

Laboratoire de Réseaux de Communication (LRC),
Ecole Polytechnique Fédérale de Lausanne, Switzerland

Abstract— **We present a protocol for controlling a shared ATM multicast tree supporting many-to-many communication. The protocol supports one or several ATM VCCs of the many-to-many type. The number of VCCs is independent of the number of endpoints. The protocol guarantees that there is no interleaving on any VCC of the tree. The protocol also guarantees that the traffic contract associated with the VCCs is respected, thus making it possible to use ordinary VCCs of the CBR, VBR or UBR class. No re-sequencing server is required, and all cell forwarding is performed at the ATM layer.**

I. Introduction

A. Background

We consider the provision of a *many-to-many* multicast service directly with an ATM network. By "directly" we mean with a data path that is entirely switched at the ATM level, as opposed to going through servers. From a pure ATM switching viewpoint, many-to-many ATM connections can be easily supported [1]. Formally, a many-to-many ATM Virtual Channel Connection (VCC) has the following properties [2]: at one User-Network Interface (UNI), the connection is defined by one single VPI/VCI; all cells sent by one endpoint of the connection are delivered to all other endpoints (assuming there is no loss); conversely, one endpoint may receive cells with this single VPI/VCI value from any of the other connected endpoints. Similarly, many-to-many Virtual Path Connections (VPCs) are defined with the VPI field replacing the VPI/VCI field.

However, ATM multicast as defined in standard [3] supports one-to-many connections, but not directly many-to-many connections: in [3], one end-system, called "root", may send data cells that are multicast to a number of other systems, called "leaves". There are known difficulties in supporting, and using, many-to-many VCCs, which explains this situation. We concentrate here on the cell interleaving issue.

Over one single many-to-many VCC, data cells from different sources may arrive interleaved at one destination. If AAL5 is used, this makes the re-assembly of messages impossible. One solution is to use AAL3/4 and allocate multiplexing identifiers (MIDs) to sources, but this requires a coordinated distribution of MIDs and is usually made at the expense of generality. Various proposals exist to address this issue. One family of solutions [4], [5] relies on the combination of servers and one-to-many connections. Data is sent by any end-system to one server over a point-to-point connection; the server relays the data back to end-systems, and to other servers, over a one-to-many connection for which it is the root. Cell interleaving is prevented by having the servers re-assemble data frames that they receive before sending them back, using AAL5. Another family of proposals [6] proposes using one one-to-many connection per endpoint; with n endpoints, every system is root for one one-to-many connection, and leaf for $n-1$ connections. Yet another family of proposals [7] requires using VPCs instead of VCCs. With a many-to-many VPC, sources can use a VCI as source multiplexing identifier. The problem with such a solution is precisely that it uses VPCs, which may be rare in some environments, and, above all, cannot be multiplexed on VPCs.

B. Scope of This Proposal

We propose a new protocol, called Shared Many-to-Many ATM Reservations (SMART) [8], which supports many-to-many communication with demand sharing. SMART has the following features:

- it lies entirely in the ATM layer, and does not require any server;
- it does not require n multicast connections for n endpoints but in contrast works with only one or several VCCs; the number of VCCs is freely configured and is independent of the number of endpoints;
- it does not require any multiplexing identifier, nor distribution of MIDs or VCIs;
- it works with VCCs (and consequently also with VPCs).

SMART uses the concept of data blocks, as in ABT [9], [10], where a block is defined as a series of cells delineated by resource management (RM) cells. The protocol supports one or several VCCs. In its simplest form, there is exactly one VCC, for any number of end-points. In its most general form, there are v VCCs, where v is a fixed number, setup by signaling procedure. The cases where $v > 1$ may be interesting for dense mode multicast where a large number of sources may send small amount of data concurrently. In this paper, the set of these v VCCs is called the multicast tree. At a UNI, an end-system that participates in one multicast tree using SMART has to connect to v many-to-many VCCs. It may send data blocks on one or several of the corresponding VPI/VCIs, whenever the SMART protocol allows it, and it may receive at most one data block on each of the v VPI/VCIs. The protocol thus guarantees that at most v data blocks can be interleaved on one multicast tree.

SMART applies primarily to the case where the VCCs are of the reserved bandwidth type (constant bit rate, CBR, or variable bit rate, VBR, classes). In that case, the protocol guarantees that the traffic contract associated with each of the VCCs is respected. In particular, the various links of the VCCs can be handled by the network as links of ordinary, symmetrical point to point VCCs. SMART can thus be

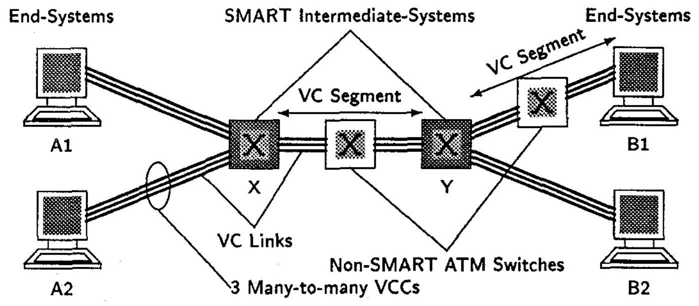


Fig. 1. A Multicast Tree Made of 3 Many-to-Many VCCs

viewed as a multiple access control protocol to the multicast tree [11], [12]. SMART also applies to UBR connections.

SMART prevents interleaving on each of the different VCCs, but supports concurrent data transfers as long as they do not result in interleaving. The protocol requires that switches and end-systems participate in the protocol; however, non-SMART switches may be used to interconnect SMART switches by means of ordinary VC links or VPCs.

II. Overview of SMART

During the setup of a many-to-many call, the end-system initiating SMART establishes a multicast tree and associates a maximum block size to it. The multicast tree interconnects a set of initial end-systems members of the many-to-many call. The tree is made of one or several associated many-to-many VCCs. Connections. The nodes of the multicast tree are called intermediate-systems. The initiating end-system has no other function than to setup the call at the beginning. The same service class (e.g. CBR, VBR or UBR) and traffic parameters are given to all the connections of the tree at call setup time. The resources for the given service are reserved in both directions on all the VC links of the tree until the connections are released. During the call, end-systems are free to join or leave the tree. The signaling procedures to establish, modify and release the multicast tree are beyond the scope of this paper.

The intermediate-systems described above are made of ATM switches implementing the SMART protocol. Intermediate and end-systems exchange control information with their *neighbors*[1] by transmitting Resource Management (RM) cells inside all VCCs of the multicast tree. RM cells are extracted and interpreted when received by a system implementing SMART. Cells carrying data, the non-RM cells, need no special treatment. To deal with possible RM cell losses, each system periodically sends RM cells to all its neighbors.

Two systems implementing SMART may be separated by non-SMART ATM switches. Thus two neighbors may be separated by several VC links in sequence. Hence an edge of the multicast tree is made of the concatanated sequence of one or several VC links per VCC, namely of one VC segment per VCC, as shown in Figure 1.

We now illustrate the major operations of the protocol by the simulation example shown in Figure 2.

A. How a Root Gets Elected

Figure 2(a) shows a multicast tree made of one VCC that interconnects end-systems A1, A2, B1 and B2. All VC segments have been established by a signaling procedure. The arrows at both ends of a VC segment indicate the state of the VC segment port in the following manner. An arrow pointing towards a VC segment port indicates that the port is ready to receive data cells and does not forward data cells to the VC segment. An arrow pointed outwards on a VC segment port indicates that the port can send data cells on the segment and discards data cells coming from the VC segment. The behavior of an arrow is similar to an electrical diode if one thinks of data cells as being the electrical current. Thus, the tree is initially idle since no data can flow on it.

Four control bits must be carried by each RM cell:

- GRANT bit to indicate if the sender can receive data;
- REQUEST bit to indicate if the sender has data to send;
- SN, a 2 bit sequence number to detect RM cell race conditions.

Initially, all end-systems send a grant to their neighbor since each of them can receive data. An RM cell in transit on a VC segment is shown by a solid arrow, while an RM cell that is received and accepted at a port is shown by a dashed arrow. As soon as a grant is received and accepted at a port at X or Y, the port arrow is reversed to indicate that data can be forwarded to the end-system; see Figure 2(b). X sends a grant to Y after having accepted the grants from A1 and A2. Similarly, Y sends a grant to X after having accepted the grants from B1 and B2. The grant from X and the one from Y cross each other on the VC segment.

In fact each VC segment is biased in the following way. A bias is negotiated between the two ports of each VC segment during the connection establishment procedure. Only one of the two VC segment ports has the bias. The bias is shown by a black dot inside the port arrow in Figure 2(c). The bias negotiation on a VC segment is independent of the bias negotiation on other VC segments. Thus the bias is a local initial condition. The signaling of the bias negotiation is outside the scope of this paper.

Y receives the grant from X but does not accept it since Y has not the bias on the VC segment to X. On the contrary, X accepts the grant from Y since X has the bias on the VC segment to Y.

The VCC is now oriented and X is the root. The tree is still idle since no data can flow on it.

B. How an End-System Starts Sending Data

Then a user at A1 invokes the UNITDATA.REQ primitive to request to send data to the multicast tree, see Figure 2(d). A1 sends a request to X. Upon accepting the request, X sends a grant back to A1, and reverses its port arrow to A1 since it can now forward data from A1, as shown in Figure 2(e). Then A1 accepts the grant and starts sending data. A1 is shown in white on black as long as it sends data and data flow is represented by a thick line, see Figure 2(f). As long as

1. Two systems implementing SMART are neighbors if no SMART switch lies between them along the multicast tree

A1 does not receive a request from X, A1 can send as many blocks of cells as it wants.

C. How Tree Access is Passed to Other Sender

Then a user at B1 invokes the UNITDATA.REQ primitive. B1 sends a request to Y. Y propagates the request of B1 to X. X forwards the request in direction of A1, as illustrated in Figure 2(g).

Upon receiving the request, A1 waits for its current data block to end, then stops transmitting and sends a grant to X. X accepts the grant from A1, changes the VC segment orientation to forward data to A1. Then X sends the grant in direction of Y and reverses the VC port arrow to forward data cells from X. Similarly Y accepts the grant from X, reverses the VC port arrow to X, sends a grant in direction of B1 and reverses the VC port arrow to B1. Finally B1 accepts the grant from Y, reverses the VC port arrow for sending data. B1 is the root of the VCC and starts sending a block of data as shown in Figure 2(g).

III. Description of SMART

This section describes the mechanisms of SMART for a multicast tree made of only one VCC.

A. How it Works

Each system keeps and updates a view of all its neighbors. It achieves this by maintaining the following state variables at each VC segment port:

ag, *ar* :	accepted grant and request
sg, *sr*, *ssn* :	grant, request and sequence number to send or last sent;
rg, *rr*, *rsn* :	last received grant, request and sequence number;
bias :	indicates if the first grant sent can be canceled or not;
status :	indicates the state of the VC segment as seen from that port.

The variables *ag*, *ar*, *sg*, *sr*, *rg* and *rr* take the values 0 and 1; all variables initially take the value 0. For example, *ar* equal to 1 means that a request has been accepted at that port, and *rr* equal to 0 means that no request is received at that port. The variables *ssn* and *rsn* take the values 0, 1 and 2; both variables initially take the value 0. The variable *bias* takes the value cancel or nocancel which is fixed during the connection establishment procedure. Finally the variable *status* takes the value active or inactive and initially takes the value active.

A VC segment of bad quality can be declared inactive by one of its ports before the link management declares it failed. A port uses a timer T1 to detect a bad quality segment in the following manner. If, after T1, a port has not received the RM cell it expected, variable *status* is changed from active to inactive. When the link management declares a VC segment as failed, the state variables are initialized, the segment is released by a signaling procedure and must be re-established to be considered active again.

A port starts to forward data cells to its VC segment if and only if *ag* is equal to 1. Data cells arriving at a port where *status* is not equal to active are not forwarded to other ports. The *status* of a port can change from inactive to active if and only if the variables *ag* of all other active ports to the same VCC of the system are equal to 1.

RM cells are sent periodically by an active port and carry the current value of *sg*, *sr* and *ssn* at that port. Moreover an RM cell is sent immediately by an active port in one of the following cases:

- if the value of *sr* changes to 1 and *ag* is equal to 0;
- if the value of *sg* changes to 1;
- if the value of *status* changes from inactive to active and *sg* is equal to 1.

Thus a request in the direction of the root and a grant propagates always as fast as possible.

The reception of an RM cell at an active or inactive port causes an immediate update of the values of *rg*, *rr* and *rsn* of that port.

B. Trees Merging

We now describe the case where two trees in operation are being merged. The state variables of a port are represented in the format of (1).

a:	*ag*	*ar*	*rsn*
s:	*sg*	*sr*	*ssn*

(1)

The value 1 of *ag* or *sg* is represented by the letter G. The value 1 of *ar* or *sr* is represented by the letter R. The value 0 of *ag*, *sg*, *ar* or *sr* is represented by a minus sign, -. The values of *rsn* and *ssn* are not modified. For example, the state variables of the port in (2) indicate that a grant was accepted but none are sent, a request is sent but none was accepted, the last received sequence number is 2 and the last sent sequence number is 0.

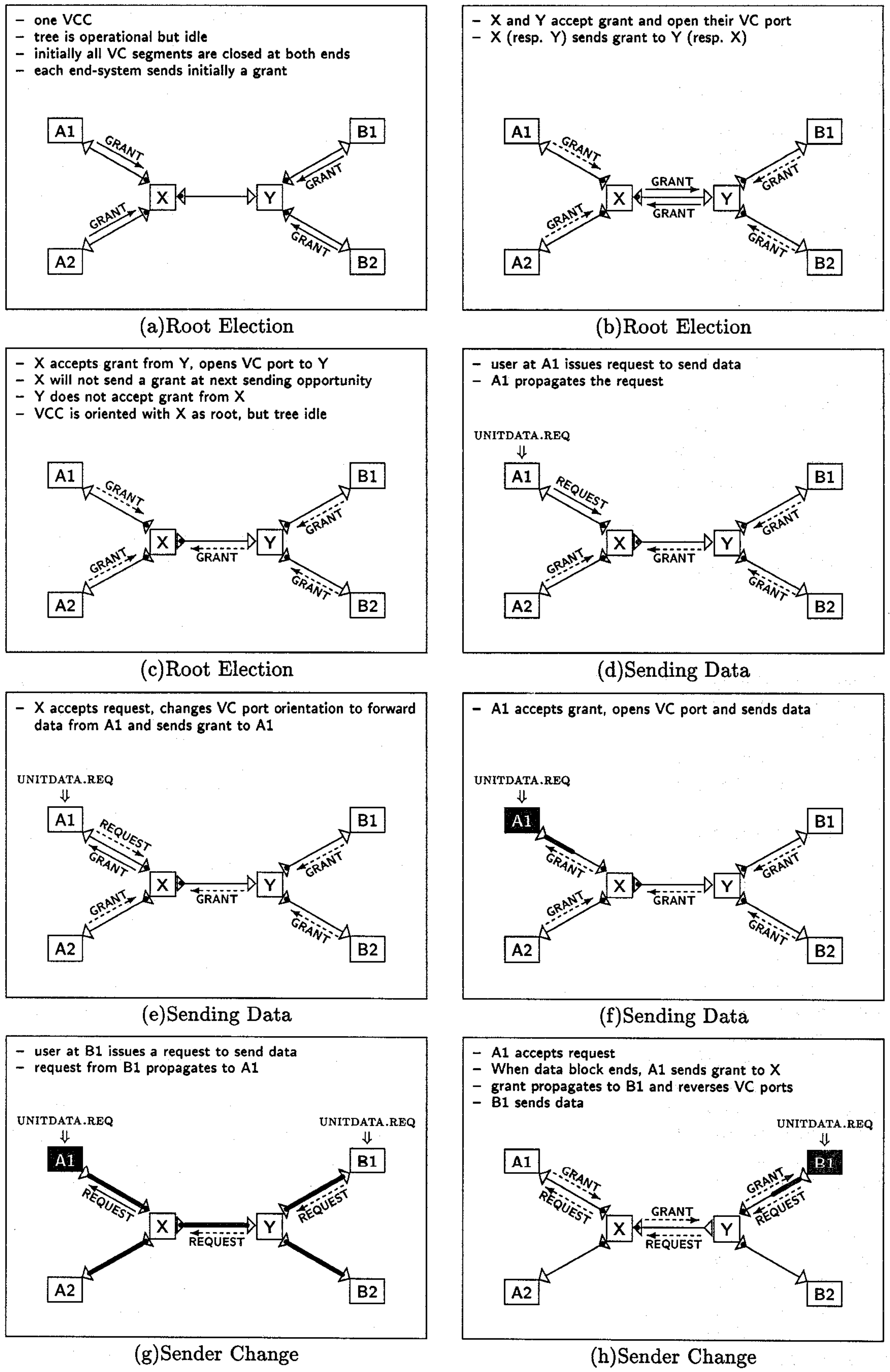
```
a:G-2
s:-R0
```

(2)

We assume that the trees X-A1-A2 and Y-B1-B2 were established and that A1 and B1 are the roots of the former tree and the later one, respectively, as shown in Figure 3(a). A1 and B1 send data on X-A1-A2 and Y-B1-B2 respectively. Meanwhile, segment X-Y is established and X wins the bias. However no data cells are forwarded on the X-Y segment since *ag* is equal to 0 at both ends of the segment. Nevertheless, the requests propagate from one tree to the other.

A1 and B1 accept their request, see Figure 3(b). A1 increments its *ssn* and sends a grant to X after it finishes sending the current block. We assume that the data block of A1 ends before the one of B1. X accepts the grant from A1 as shown in Figure 3(c). X sends a grant to Y since it has received a grant from all its other neighbors.

Y accepts the grant from X and can start forwarding data cells to X. The tree has now only one sender. Note that A1 and A2 will receive only the end part of the block sent by B1, see Figure 3(d). Note that SMART makes sure there is no cell interleaving but does not guarantee a reliable data transport.



– one VCC
– tree is operational but idle
– initially all VC segments are closed at both ends
– each end-system sends initially a grant

(a)Root Election

– X and Y accept grant and open their VC port
– X (resp. Y) sends grant to Y (resp. X)

(b)Root Election

– X accepts grant from Y, opens VC port to Y
– X will not send a grant at next sending opportunity
– Y does not accept grant from X
– VCC is oriented with X as root, but tree idle

(c)Root Election

– user at A1 issues request to send data
– A1 propagates the request

(d)Sending Data

– X accepts request, changes VC port orientation to forward data from A1 and sends grant to A1

(e)Sending Data

– A1 accepts grant, opens VC port and sends data

(f)Sending Data

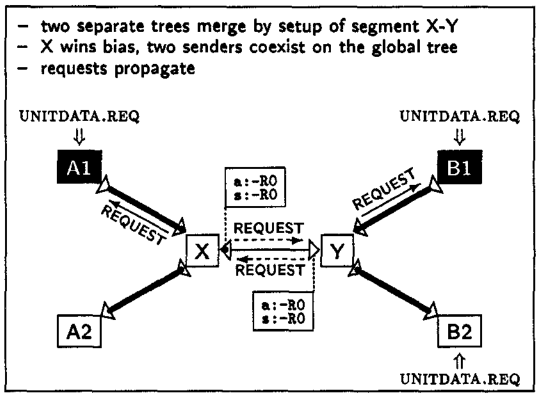
– user at B1 issues a request to send data
– request from B1 propagates to A1

(g)Sender Change

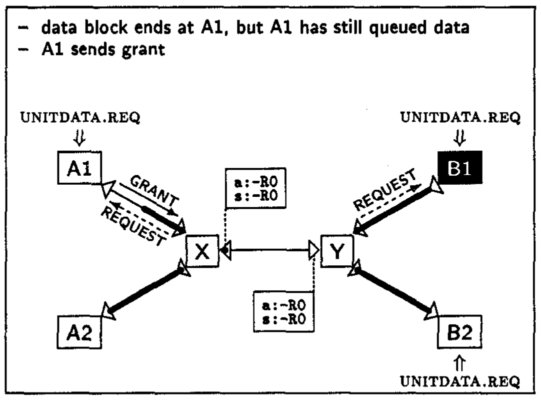
– A1 accepts request
– When data block ends, A1 sends grant to X
– grant propagates to B1 and reverses VC ports
– B1 sends data

(h)Sender Change

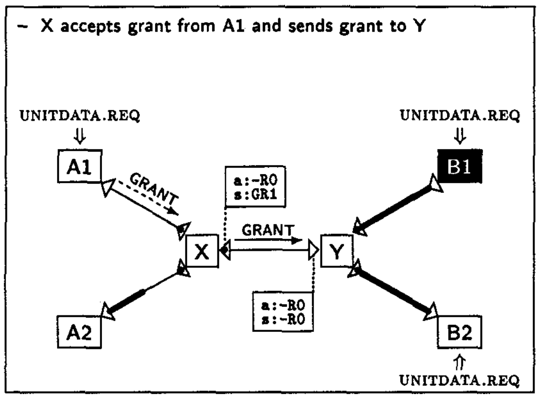
Fig. 2. Basic Simulation Example



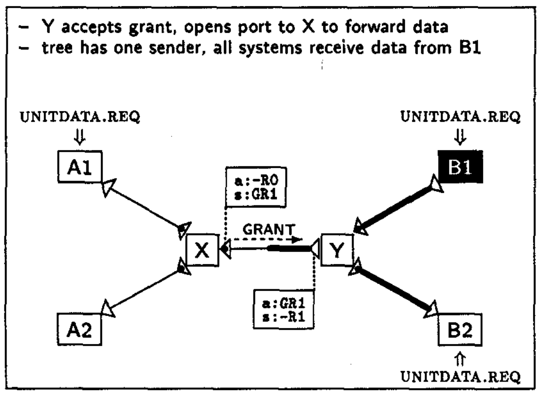
(a)



(b)



(c)


– Y accepts grant, opens port to X to forward data
– tree has one sender, all systems receive data from B1
UNITDATA.REQ
UNITDATA.REQ
A1
B1
a:-R0
s:GR1
GRANT
X
Y
a:GR1
s:-R1
A2
B2
UNITDATA.REQ

(d)

Fig. 3. Example of Two Trees Merging

IV. Conclusion

We have presented a protocol for controlling a shared ATM multicast tree supporting many-to-many communication. The protocol supports one or several ATM VCCs of the many-to-many type. The number of VCCs is independent of the number of endpoints. The protocol guarantees that there is no interleaving on any VCC of the tree. The protocol also guarantees that the traffic contract associated with the VCCs is respected, thus making it possible to use ordinary VCCs of the CBR, VBR or UBR class. No re-sequencing server is required, and all cell forwarding is performed at the ATM layer. The protocol is implemented in PROMELA [13] and the absence of cell interleaving in any VCC of the multicast tree was validated [14]. In an extension of SMART [15], called SMART/DR (Dynamic Reservations), only a minimum cell rate can be associated to the multicast tree at connection setup. The current cell rate is then determined dynamically by the protocol, based on requests by users, and decisions taken by all intermediate-systems, which may accept, reject, or degrade the requests.

References

[1] W. D. Zhong and K. Yukimatsu, "Design Requirements and Architectures for Multicast ATM Switching," *IEICE Trans. Commun.*, vol. E77-B, Nov. 1994.
[2] ITU-TS, "ITU Recommendation I.150, B-ISDN ATM Functional Characteristics." 1993.
[3] A. Forum, "ATM User-Network Interface version 3.1 Specification." 1994.
[4] G. J. Armitage, "Multicast and Multiprotocol Support for ATM based Internets," *ACM Sigcomm Computer Communication Review*, Apr. 1995.
[5] H. L. Truong, W. W. Ellington, Jr., J.-Y. Le Boudec, A. X. Meier, and J. W. Pace, "LAN Emulation on an ATM Network," *IEEE Comm. Mag.*, 1995.
[6] M. Kusayanagi *et al.*, "ATM-Based Access Network with Multicast Function for Multimedia Services," in *Proc. SPIE*, pp. 45–56, 1995.
[7] H. Tode, M. Yamamoto, and H. Okada, "Multicast Routing Scheme Adequate for Virtual Path Environment," *Electronics and Communications in Japan*, vol. 78, no. 1, 1995.
[8] E. Gauthier, J.-Y. Le Boudec, and Ph. Oechslin, "SMART: A Many-to-Many Multicast Protocol for ATM," tech. rep., Ecole Polytechnique Fédérale de Lausanne, Département d'Informatique, 1996. http://lrcwww.epfl.ch/PS_files/SMARTpaper.ps.gz.
[9] ITU-TS, "Recommendation I.371." draft, Dec. 1995.
[10] P. E. Boyer and D. P.Tranchier, "A Reservation Principle with Applications to the ATM Traffic Control," *Computer Networks and ISDN Systems*, vol. 24, pp. 321–334, 1992.
[11] H. R. van As, W. W. Lemppenau, P. Zafiropulo, and E. A. Zurfluh, "CRMA-II: A Gbits/s MAC Protocol for Ring and Bus Networks with Immediate Access Capability," in *Ninth EFOC LAN Conference*, (London UK), June 1991.
[12] P. Heinzmann, H. R. Mueller, D. A. Pitt, and H. R. van As, "Buffer Insertion Cell Synchronized Multiple Access (BCMA)," in *Second International Conference on Local Communications Systems*, (Palma Spain), June 1991.
[13] G. J. Holzmann, *Design and Validation of Computer Protocols.* Prentice Hall, 1991.
[14] E. Gauthier, "The SMART Protocol." http://lrcwww.epfl.ch/pub/smart/gauthier-smart.html.
[15] E. Gauthier, J.-Y. Le Boudec, and Ph. Oechslin, "Shared Many-to-Many ATM Reservations," Tech. Rep. 96/168, Ecole Polytechnique Fédérale de Lausanne, Département d'Informatique, 1996.