

# Analysis of a Resequencer Model for Multicast over ATM Networks

Liming Wei<sup>1</sup> \*, FongChing Liaw<sup>2</sup>, Deborah Estrin<sup>1</sup>, Allyn Romanow<sup>2</sup>,  
Tom Lyon<sup>2</sup>

<sup>1</sup> Computer Science Department, University of Southern California,  
Los Angeles, CA 90089-0782

<sup>2</sup> Sun Microsystems Computer Corporation, Mountain View, CA 94043-3156

**Abstract.** Multicast delivery saves bandwidth and offers logical addressing capabilities to the applications. The receivers of a multicast group need to differentiate cells sent by different sources. This demultiplexing requirement can be satisfied in an ATM environment using multiple dedicated point-to-multipoint virtual channel connections (VCs), but with certain shortcomings. This paper discusses an alternative resequencing model to solve this problem. It scales well in large networks. Three resequencing methods are developed and simulation results reported. The strategy is useful for applications spanning large regions where it is desirable to mix streams of cells from different bursty sources onto the same virtual channel. <sup>3</sup>

---

Appeared in the Proceedings of the 3rd International Workshop on Network and OS Support for Digital Audio and Video, San Diego, Nov 1992}}

## 1 Introduction

It is important for future ATM networks to have multicast capability, as they will support such applications as teleconferencing and information distribution services. Applications based on multicast have two major advantages over unicast: logical addressing and bandwidth savings [Deering]. With logical addressing, an application uses a single multicast group address to send and receive data. It is not necessary for senders and receivers to know the number or location of group members. Multicast provides significant savings in bandwidth because the sender transfers only a single copy of the data. Data cells are not replicated until they

---

\* Work supported by SUN summer internship 92 at SMCC, Mountain View, California.

<sup>3</sup> We would like to thank Steve Deering, Bryan Lyles, Lixia Zhang and the referees for helpful discussions and comments.

reach a branching point in the multicast delivery tree, at a location closer to the destinations.

The results of research and experimental implementations of multicast in IP networks have demonstrated the benefits of supporting multicast in public data networks [RFC1112] [MOSPF]. Although it would be desirable to adopt the implementation approaches successfully used in IP networks, the differences between an ATM network and a conventional packet switched network make it difficult to directly map multicast models for IP into ATM networks.

In IP networks, for a multicast group having N senders, N multicast delivery trees are established, each tree delivering packets from a source to all destinations in one-to-many fashion. Each IP packet carries a source address and a destination (i.e. multicast group) address in its header [RFC1112]. To route multicast packets both destination address and source addresses are used for lookup in the routing table. A receiver uses the source address field of the IP header to distinguish packets from different senders. Thus multicast packets can be arbitrarily intermixed with each other on the forwarding path.

ATM, however, is a connection oriented technology, in which connections must be explicitly setup before data can be transferred. An ATM connection can be viewed as a cached route [LLR], in the sense that each cell carries only a small routing tag — the virtual path identifier and virtual channel identifier (VPI/VCI). Unfortunately, if cells from different sources are multiplexed onto the same virtual channel, they will carry the same routing tag, or VPI/VCI, upon arrival at a receiver. There is not a straightforward way to distinguish cells sent from different sources. This is often referred to as the *cell demultiplexing* problem. Although several solutions have been proposed, they have shortcomings and, most importantly, they do not scale well.

This work proposes a solution that both solves the cell demultiplexing problem and scales well. The following section describes currently proposed solutions to ATM multicasting and discusses their shortcomings. The third section develops the proposed resequencer model. In section four, simulation results for the performance of several different methods of resequencing are presented.

## 2 Strategies for Implementing Multicast in ATM

The cell demultiplexing problem can be avoided if one VPI/VCI is used for each source. In this case, each multicast VPI/VCI is a one-to-many connection representing one multicast tree and is independent from other VPI/VCI's. Each tree can be optimized according to some criterion such as least delay or least cost.

The VPI is an 8-bit field in the cell header. If it is used to identify a multicast group, it restricts the number of multicast connections to even less than the number available with VCI's. In addition, VP switching may be used to bundle large number of VC connections, or to separate VCs of differing quality of service classes. Therefore, we assume that the VP will not be available for use as a multicast tree identifier. For the remainder of this discussion we refer to the use

**Fig. 1.** One-vc-per-source multicast model

The one-vc-per-source strategy is compatible with the current IP multicast models [RFC1112], provided that the switch signaling mechanisms have access to the installed IP multicast routing tables [LLR]. The scheme is useful for (a) applications with small numbers of sources; (b) local multicast groups where there is not a shortage of VCs relative to the demand; and (c) applications where all sources continuously transmit (e.g., video).

However this scheme uses a large number of VCIs and therefore does not scale well to networks with many nodes and large numbers of long-lived multicast groups.

In particular, the scheme has a number of limitations. The number of virtual channels is too limited to accommodate a sizable number of large, long-lived groups. Furthermore, if bandwidth is reserved or allocated on a per VC basis, the large number of VCs resulting from a multicast group will quickly use up the available bandwidth. Similarly, in the public network, if the cost of cell switching depends on the number of virtual channels, the additional VCs used in per-source-VC will be a costly disadvantage. In addition, since each sender defines a tree, when a new receiver joins a group, it must be added to all existing trees.

It is important for ATM multicast to scale to large size. Applications involving hundreds, or even thousands, of end users globally are not too far distant. Thus it is desirable to explore alternatives in which the number of available VCs in a switch or host interface is not an upper bound on the number of sources an application can support.

Recently several models have been proposed for ATM multicast, aimed at solving the VC depletion problem, and achieving high throughput and scalability [BGD] [GRELYL] [SBO] [KPP]. One proposal [GRELYL] is for all sources in

a multicast group to share one virtual channel, and to use collision detection techniques, such as Aloha, to avoid intermixing cells from different sources. The CRC will detect when a cell is out of order, i.e., not from the same sender, and the sender retransmits. In this scheme, the common virtual channel is treated as a shared media, similar to an ethernet cable, with some of the same advantages and disadvantages ethernet has. The scheme is useable for local, low bandwidth and non real-time applications.

In another scheme [BGD], an identifier in the ATM payload field is used to distinguish cells from different sources. It is not clear how simple or complex the procedures for negotiating unique identifiers among sources would be. This scheme was developed for AAL4 in which there is a MID (multiplexing ID) field in the cell information header. However, currently many developers of ATM plan to use AAL5 [SEAL] which does not have such a demultiplexing field in the header. In any case, the width of the MID field, which is 10 bits, puts a limit on the number of sources allowed in a group. Another cost is that the host interface is required to have additional demultiplexing hardware.

### 3 The Resequencer Solution to Multicast

The scheme we introduce solves the VCI depletion and bandwidth allocation problems without requiring specialized bandwidth allocation schemes for certain common cases of multicast. The traffic classes for which this approach is well suited are: (a) transmission per source is not continuous, (b) small to modest size PDUs, (c) large multicast groups. First we describe the basic resequencing strategy, followed by several extensions and refinements to the basic idea. Next we present a group tree model which extends the scheme to a larger network.

#### 3.1 Resequencers

To achieve better bandwidth sharing without complicating the VC-based resource management mechanisms, and to avoid the VC availability dilemma, cells from different sources are multiplexed onto a single VC. A designated source in the group is elected as a *resequencer*. All other sources send their multicast ATM cells to the designated resequencer, which buffers incoming cells from each source before all cells of a PDU are received. After the last cell of a PDU has been received, the resequencer forwards all cells of that PDU onto a single outgoing VC, without interleaving it with cells from other sources.

Note that the outgoing VC of the resequencer is a point-to-multipoint connection. Cells are duplicated at the branching point(s) and the order of cells is preserved across all links in this one-to-many connection. If a sender is also a receiver, there is a branch of the one-to-many VC leading back to the sender.

Although this single resequencer scheme provides the benefits of multicast, it has several problems. The processor and link speed of the source designated as a resequencer could become a bottleneck. Choosing the resequencer is complex,

and there may be situations where no ideal selection of designated resequencer exists. This scheme is best when all senders are close to the resequencer.

To extend the scheme to accommodate the case where members are spread over different regions, and each region has many members, members in each region elect a source to act as a resequencer for local senders. A one-to-many VC is created for each resequencer which delivers cells to all other resequencers of the group, and cells are therefore delivered to all receivers. The number of VCs required is now reduced to the number of participating regions, instead of the number of sources (N one-to-many VCs for N resequencers).

To make the scheme more efficient, resequencing can be done in switches instead of in hosts. Normally switches sustain higher aggregate cell rates. To participate in multicast, the host only needs to know the address of its nearest multicast resequencer. In a local area ATM, the location of a multicast resequencer can be either statically assigned, or dynamically discovered, so that the native nodes are kept updated about the available multicast servers. To avoid long call setup latency, the resequencer information can be cached in the host. This approach eliminates the necessity of going through a decision process to choose a host resequencer.

### 3.2 Multicast Group Trees

Our final refinement of this approach is to further reduce the number of VCs maintained for a group. We do this by building a group tree instead of per-source (or per-source-region) rooted trees. Connections among the resequencers are bidirectional. A resequencer serializes all cells destined for a common group and duplicates them onto the appropriate outgoing ports with correct VPI/VCI's — some lead to other resequencers, some go to local members. Figure 2 depicts a group tree of resequencers and receivers. The group tree is a delivery path in common for cells from all sources. As Figure 2 shows, each source establishes a point-to-point connection to its resequencer and sends cells towards a multicast group along that connection, as if it were a unicast destination.

Either point-to-point or point-to-multipoint connections can be used to distribute cells from a resequencer to its local receiving members, depending on support from the local network. If a member is both a sender and a receiver, a bidirectional point-to-point connection is used. The connections between s2, s3 and Resequencer M2 in fig 2 are such examples. To reduce the bandwidth used in local multicast cell delivery, local switches must support point-to-multipoint connections. The connection between resequencer M3 and r0, r1 and r2 in fig 2 is such a point-to-multipoint connection, where r0, r1 and r2 are all receivers<sup>4</sup>. Note also that there may be many “pure” ATM switches between any two resequencers, and between any resequencer and its local members. The paths drawn in figure 2 are logical channels.

---

<sup>4</sup> Note that it is not suggested to replace the reverse channels of s2 s3 and s4 with a single point-to-multipoint connection, unless the senders don't care if they receive duplicated cells looped back by resequencer M2.

**Fig. 2.** The *resequencer and group multicast tree* model

### 3.3 Discussion

The resequencer approach requires algorithms different from those used for IP multicast to compute the group multicast tree. The tree will not be optimal in terms of shortest path or least delay. However, suboptimal solutions should suffice for a number of applications [KPP] [CBT].

Since this resequencer and group tree model can be built upon one-to-one and one-to-many virtual channel connections, it can coexist with the one-vc-per-source model. An application can choose one of the two models. For example, video applications where sources transmit continuously may choose to use the one-vc-per-source model, whereas video conferencing with compressed streams and multiple changing sources may choose the resequencer approach. The resequencer and group tree model is also flexible in that it works both with and without point-to-multipoint virtual channels. The difference is in the bandwidth savings of the local cell delivery paths. This approach turns out to be especially well suited for voice conference applications, where one bandwidth allocation for the whole multicast group may be sufficient and is independent of the number of sources. In an audio conference, it is unusual to have multiple simultaneous transmitters. Usually only one or two persons speak at a time, as is consistent with higher level human protocols.

In order to setup and maintain the group tree, a higher layer protocol — either a network layer protocol or a call control level protocol, is needed. Differ-

ent protocols may be appropriate for this purpose. One possibility is to adapt Deering's host membership protocol and multicast tree setup protocols [Deering].

The resequencer and group tree multicast model has a nice scaling property. Only one virtual channel is needed for each receiver to receive cells from all sources. When a new member (be it a sender or a receiver or both) joins, it only needs to do the setup necessary to reach a resequencer already in the group. The operations of join and leave are inherently local, growing a branch or chopping off a branch does not have to involve a distant part of the tree.

Two performance parameters, maximum throughput and delay, will be used to assess the resequencer and group tree model, and will determine the range of applications of this model. Different implementation methods result in different maximum throughputs ranging from tens of megabits per second to that approaching the maximum unicast rate. We will show in the next section that as the needs for higher speed multicast service appear, there are ways to increase the maximum throughput without changing the end system.

## 4 Three Approaches to Resequencing

In the resequencer and group tree model, resequencers are the crucial components that determine performance. We describe three different schemes for resequencing cell streams from different sources: in software at the network layer; in hardware; and an optimized hardware scheme. The performance of these approaches is evaluated with respect to throughput, delay and delay jitter.

**Software Resequencing** Multicast cells from the same source can be reassembled at the resequencer in the same way that signaling messages are reassembled. An ATM switch with signaling capability has associated CPU and memory to process and forward signaling messages [Q.93B]. Similarly for resequencing of multicast cells, the reassembled PDUs are passed to the control processor, which makes routing decisions by inspecting the address field in the header of the PDU. The forwarding process is shown in figure 4(a). The route lookup and PDU forwarding are done in software, similar to a datagram router.

To examine performance, as shown in figure 4(b), the cell forwarding delay can be decomposed into three parts: (1) the delay to collect cells of the PDU; (2) the delay due to the route lookup algorithm; and (3) transmission delay.<sup>5</sup> We observe that the cell collection time and transmission time of consecutive PDUs can be overlapped. Therefore, route lookup is the dominant factor affecting throughput.

With certain optimizations in the implementation, the software forwarding scheme can achieve high throughput. Consider a switch equipped with a 50 MIPS processor. Assume its multicast routing table has 1000 entries and it

---

<sup>5</sup> PDU queuing delay needs not be considered here, if we assume no congestion.

**Fig. 3.** Software PDU forwarding

takes about 1000 instructions to forward one PDU<sup>6</sup>. The per PDU route lookup and forwarding time will be  $\sim 20 \mu s$ . For 256-byte PDU, this corresponds to a maximum throughput of 100 Mbps, if the cell collection times and transmission times of consecutive PDUs are completely overlapped.

The delay experienced by each individual PDU depends on a number of factors. In the one-vc-per-source model, a PDU is not reassembled until reaching an end host. Using the software resequencing method, a PDU has to be reassembled in every resequencer it passes. The additional delay is the sum of cell collection delays and routing delays the PDU experienced in all resequencers along its path. The PDU cell collection delay depends on the PDU size and link speed.

Figure 4 shows the PDU cell collection delay under different link speeds and PDU sizes. Multimedia applications with delay jitter or other real-time requirements tend to use small packet sizes, between 128 to 256 bytes. Even with larger PDU sizes (1000 Bytes) and slower link speed (155Mbps), the worst case aggregate reassembly and routing delay in the above example can still be less than

---

<sup>6</sup> This is a rough estimation. If the operating system in ATM switches are better designed for communication protocol processing, the number of instructions would be smaller.

~7.7ms, if less than 100 resequencers are involved in the longest path.

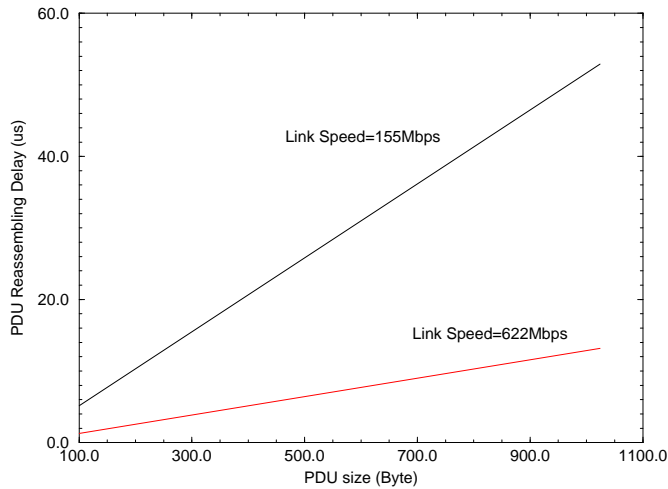


Fig. 4. PDU reassembly delay

Thus the performance of the software resequencing method, in terms of speed and delay, is good for applications requiring less than 100Mbps and involving less than thousands of regions — provided the longest path of the multicast group tree has less than a few hundred resequencers.

**Hardware Resequencing** When higher throughput is required, we propose a method using hardware support in the switch. Instead of a software route lookup, multicast cells can be forwarded directly by hardware switch fabrics using mechanisms similar to that used for forwarding point-to-point connection cells. Incoming VCs carrying multicast cells are treated separately from those for normal point-to-point connections; they are queued in buffers, neither forwarded nor reassembled. The switch monitors the End of Packet (EoP) cell. When the EoP cell arrives, the hardware dumps all cells in that PDU back-to-back onto outgoing links.

Hardware resequencing requires extra buffers and buffer management mechanisms in the resequencing switch hardware. The amount of buffer required depends on a variety of factors, including the PDU size, traffic characteristics, congestion control algorithms, etc. We can estimate the number of buffers required by a resequencer in the restricted case assuming no congestion and that sources obey negotiated resource allocations. In this case, a switch has sufficient bandwidth to carry the incoming traffic, and the queue size for each incoming VC should not grow beyond 2 PDUs. More buffer space is needed only when there is congestion. If the maximum number of active multicast groups is  $N$ , the

maximum number of active source streams coming in to a resequencer for each active group is  $S$ , and the maximum PDU size used by all multicast sources is  $P$ , the maximum amount of buffer required in a resequencer is  $2 \times N \times S \times P$ .

**Optimized Hardware Cell Forwarding** The third method is an optimization of the second. When the resequencer receives the first cell of a PDU from a source, if there are no cells from other sources queued or being forwarded, it directly forwards all cells on the outgoing links without queuing them first, until it forwards the End of Packet cell. During the forwarding action, if cells from other sources arrive (i.e., on other incoming links), they will be queued and processed according to either the software or hardware resequencing method.

To reduce delay variance, after an EOP cell has been sent, the switch should try to process queued cells before processing newly-arriving cells. These latter cells are queued.

This optimized scheme is particularly suited for certain applications, such as voice conferences, where normally only one or very few participants are actively transmitting to the group at the same time. Both queuing time and route lookup time for the preferred groups are eliminated. This scheme achieves almost the same cell delivery throughput and delay as one-vc-per-source method, however it uses only a single VC for the whole group. This method is less suitable for groups with many members simultaneously transmitting data at a constant rate to the group.

## 5 Simulation Studies

Since this model is targeted towards applications with real-time constraints, we have built an event driven simulator to study the cell forwarding delays and delay jitters. End-to-end delay and variance are dependant on per switch delays and other parameters. We simulate and measure the switches under different traffic conditions and with each of the resequencing methods described.

The simulator uses a queuing model to characterize the behaviors of different buffers in a switch. The input queues buffer incoming cells; the PDU queue is used by the software resequencer to save reassembled PDU's before route-lookup; the output queues store outgoing cells when the cell arrival rate is larger than the link speed. For simplicity, a FIFO algorithm is used to schedule the resources whenever competition occurs. The link speed, link delay, network connectivity, software PDU forwarding speed, end-to-end route, and multicast group information etc. are entered by the user and stored statically.

A simple traffic model has been used to approximate Audio/Video style traffic sources in the simulations. The traffic model assumes that each source generates data at a constant rate, and every  $t$  seconds, a PDU is ready to send. Then the PDU is chopped into ATM cells and sent onto the network at the link speed, while at the same time the application (slower than the network) is generating data for the next PDU. The source stops generating data after sending several

PDU's and stays silent for a randomly selected period of time. Then it starts sending again.

A link during the interval between delivery of two consecutive PDUs is like a vacuum. This partially explains why in the following simulation, even when the total active sending time of all the relatively slow members of the group exceeds 100%, the delay caused by resequencing different PDU's in collision is not very significant. Of course, there is a chance that the number of collisions can be larger if the slow sources together with the link delay make different PDU's frequently arrive at a resequencer at the same time.

A fully connected 17-node network (including 11 hosts and 6 switches) was constructed. A total of 10 multicast groups were created. The resequencer switch located at the network bottleneck location was setup for measurements. Figure 5 shows the distributions of the cell delays across that switch. The same traffic sources are used for the three runs with different resequencing methods. The measured path has four sources sending to the same group and whose percentage of time actively sending are 80%, 50%, 50%, 30%. Although the source rates are below the saturation point, they do collide inside the resequencer and get queued and reordered on exits. The tails of the curves are caused by the queuing delays. The horizontal distance between the Hardware resequencing curve and the software resequencing curve reflects the route look up delay (20  $\mu$ s). The horizontal distance between the optimized and the hardware resequencing curves reflects the cell collection time (related to link speed and PDU size). The differences in heights of the three curves depict the fact that the more time a PDU spends in a resequencer, the more chance it may collide with another PDU. The widths of the delay distribution curves shows the jitter the cell experienced.

Most of the sources simulated use 20-cell PDUs, except for one that uses smaller 9-cell PDU, and one that uses larger 30-cell PDUs. With larger PDU sizes, the curves in the picture will move horizontally. Curves 1 and 2 will move towards the right, indicating larger cell collection delay, and more chances of collision would occur. The lines at the measured resequencer are heavily utilized at 50% to 90% of available bandwidth.

## 6 Conclusion

We have introduced a resequencer and group tree multicast model for multicast over ATM networks and the traffic classes for which this approach is well suited.

## References

- [GRELYL] D. H. Greene and J. Bryan Lyles. *Reliability of Adaptation Layers*, 3rd IFIP WG6.1/6.4 workshop on protocols for high-speed networks, Stockholm, May 13-15, 1992
- [RFC1112] S. Deering. *Host Extensions for IP multicasting*, RFC1112, Aug 1989
- [Deering] S. Deering. *Multicast Routing in a Datagram Internetwork*, Stanford University PhD Thesis, Dec 1991

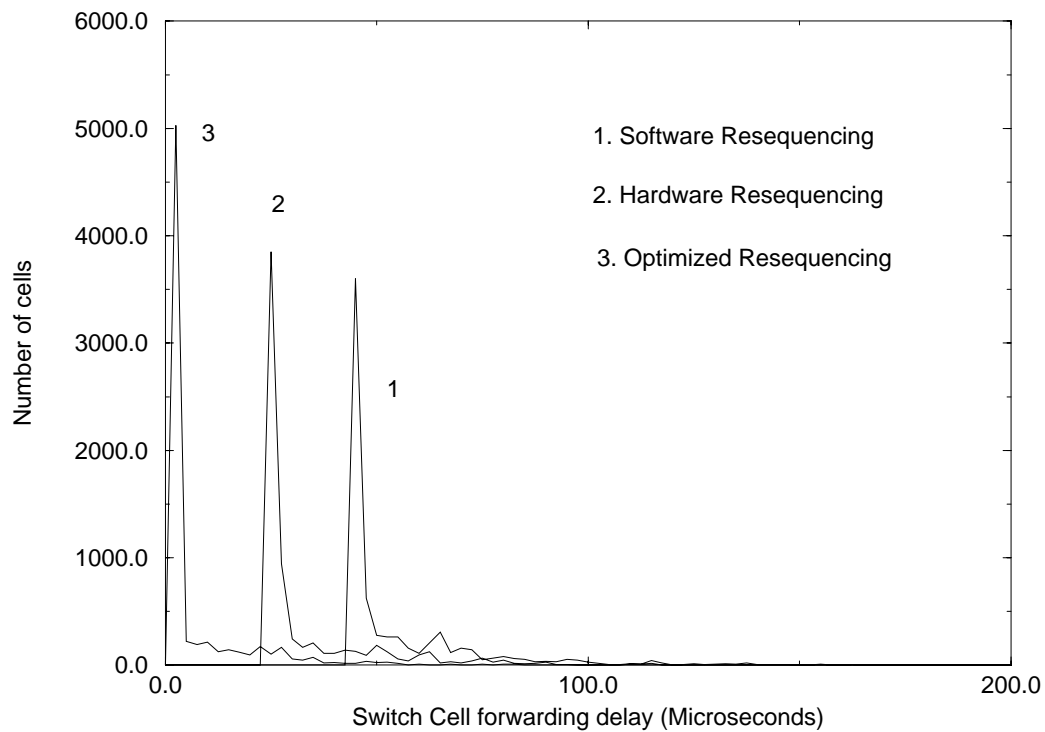


Fig. 5. Distribution of cell forwarding delays

- [BGD] R. Bubenik, M. Gaddis, J. DeHart. *Communicating with Virtual Paths and Virtual Channels*, Proceedings of Infocomm 1992
- [KPP] V. Kompella, J. Pasquale, G. Polyzos. *Multicasting for Multimedia Applications*, Proceedings of Infocomm 92
- [MOSPF] J. Moy. *Multicast Extensions to OSPF*, Internet Draft, November 1991
- [SEAL] T. Lyon, *Simple and Efficient Adaptation Layer(SEAL)*. Proposal T1S1.5/91-292 to ANSI working group T1S1.5, Aug 1991
- [CSZ] D. Clark, S. Shenker, L. Zhang. *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*, Proceedings of SigComm 92.
- [LLR] T. Lyon, F. Liaw, A. Romanow. *Network Layer Architecture for ATM Networks*, Internet Draft, June 1992
- [Lyles] B. Lyles. *Private Conversation*, June 1992
- [Q.93B] CCITT, draft text, *Q.93B specification*, March, 1992
- [SBO] A Segall, T. Barzilai and Y. Ofek, *Reliable Multi-user Tree Setup with Local Identifiers*, INFOCOMM 92
- [CBT] J. Crowcroft, A. Ballardie, P. Tsuchiya. *Core Based Trees*, Internet Draft, August, 1992