Multicast Transport Protocols for High Speed Networks

Sanjoy Paul Krishan K. Sabnani David M. Kristol

AT&T Bell Laboratories AT&T Bell Laboratories AT&T Bell Laboratories
Holmdel, NJ 07733 Holmdel, NJ 07733 Murray Hill, NJ 07974

Abstract

This paper presents the design and analysis of three reliable multicast transport protocols for high speed networks. The novelty of these protocols lies in the technique used in combining the acknowledgments of individual destinations along the underlying multicast tree to prevent acknowledgement implosion and in the technique used in preventing unnecessary retransmission by performing local multicasts. These protocols use the periodic exchange of complete state information [NRS90] between the source and the destinations and a block-based Selective Repeat retransmission scheme to improve the overall performance in a high speed networking environment. Performance of each protocol is analyzed in terms of throughput, end-to-end delay, buffer requirement, acknowledgment traffic and retransmission traffic. Based on this analysis and the complexity of implementation, one of the three protocols is recommended for reliable multicasting in high speed networks.

1 Introduction

Multicasting is an important issue which needs to be addressed in the high speed networks. Several papers have addressed the issue of designing multicast packet-switching networks [TL90],[E88] and a lot of work has also been done in multicast routing [A84][D88], but the design of a reliable multicast transport protocol in broadband high speed networks has not received as much attention until recently [AFM92]. [APR93] discusses the design of a reliable multicast protocol in a distributed environment. They do not focus on the transport of packets on wide area networks and hence the questions of round trip delay, acknowledgment traffic, remulticast traffic, or remulticast timeout do not arise in their design, while they do in ours. Chang and Maxemchuk [CM84] designed a reliable broadcast protocol in unreliable broadcast networks. Reliable multicast protocols for satellite broadcast channels [SS85] have also been designed.

Multicasting is a very broad term and different multicasting applications have, in general, different requirements. For example, a nationwide video conferencing, which is a real-time multipoint-to-multipoint multimedia multicasting application, has very different requirements from an electronic distribution of books by a publisher to bookstores across the nation, which is a point-to-multipoint reliable data transfer multicasting application. In this paper, we address multicasting applications of the second type which require reliable data transport with high throughput and low end-to-end delay. However, many of the ideas presented in this paper can be adapted for real-time, multimedia, multipoint-to-multipoint applications as well. In order to achieve high throughput and low end-to-end delay, we avoid *unnecessary retransmission* by the source using a technique called *local multicast*. In addition to that, we adopt a novel technique of *merging acknowledgments* from individual receivers along the multicast tree to avoid the *acknowledgment implosion* problem inherent in any multicasting scheme. We use the principle of *periodic and frequent exchange of complete state* between the transmitter and the receivers to avoid complex error recovery procedures [NRS90]. Finally we use a block-based selective repeat retransmission scheme to achieve high throughput [NRS90].

In this paper, we present the design and analysis of three multicast transport protocols for high speed networks for point-to-multipoint multicasting applications which require high throughput, low end-to-end delay and reliable transport of non-real-time data. The protocols are very general in the sense that they can be built on top of either virtual circuit networks or datagram networks. The only service expected by the protocols from the underlying network is the establishment of a multicast tree from the source to the destinations. It does not really care if the multicast tree is set up (including resource reservation) using ST-2 [PP92], RSVP [ZDESZ93] or any connection oriented

0-8186-6685-4/94 $4.00

protocol. As long as the tree exists, the protocols will work. The function of the protocols is to deliver packets from the source to the destinations in sequence along the multicast tree, independent of how the tree is created and resources are allocated. This makes our protocol deployable in various networks and thus they can take advantage of any clever resource allocation technique in datagram network or any efficient technique for setting up virtual circuits in connection oriented networks. In the context of broadband ATM networks, these protocols can be used in the Adaptation Layer for multicasting applications.

The three protocols for multicasting are the Designated Status Protocol (DSP), the Consolidated Status Protocol (CSP) and the Combined Protocol (CP). CSP introduces the concept of combining acknowledgments from the destinations along the multicast tree, DSP introduces the concept of local multicast (multicasting to the destinations in a local region rather than to all destinations in the network) and CP combines the ideas from both protocols. CP performs the best among all the three protocols. However, the cost paid in terms of communication and computation overhead does not justify its use. DSP, which is a close second in terms of performance, is our recommendation.

This paper is organized as follows. Section 2 discusses the assumptions we make in the design of the protocols. Designated Status Protocol (DSP) is described in Section 3, followed by the descriptions of Consolidated Status Protocol (CSP) and the Combined Protocol (CP) in Sections 4 and 5 respectively. The performance of the protocol is analyzed in section 6, followed by the simulation results in Section 7. The three protocols are compared in Section 8, followed by the conclusion.

2 Assumptions

In this section, we list the assumptions common to all three protocols described in the paper. Later we state the assumptions specific to each protocol. The common assumptions are as follows:

1. The endpoints/users are connected to a local exchange (LE) either directly or through access nodes. The local exchanges are interconnected using a backbone network (Fig.1).

2. A hierarchical addressing scheme like E.164 (which is very similar to the current telephone numbering system) is assumed. That is, given the address of an endpoint, it is possible to infer the area it belongs to. For example, if the address of an endpoint is 908-317-9128, it can be inferred that the endpoint is served by area code 908 and that it is an address in New Jersey. Similarly, if the endpoint has an address 415-394-1127, it is possible to conclude that it is served by area code 415 and that it is an address in California.

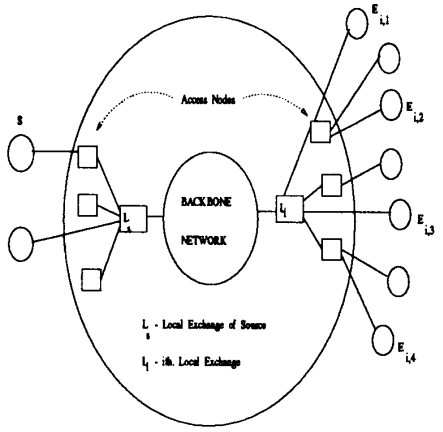

Figure 1: Model of the Network

In fact, none of the above assumptions are necessary in general. However, the ideas presented in this paper are easy to comprehend with these assumptions in mind. All that is necessary here is some sort of *localization* information which needs to be used by the protocols. Conceptually the destinations in a *local* region need to be treated as a group for the purpose of retransmission of packets. This effect of localizing requests for retransmission can be achieved, for example, in IP by using the TTL field of an IP packet. The details of how the TTL field can be used for this purpose is the subject of a different study.

3. We assume that a multicast tree is set up with the allocation of resources at the network layer (ATM layer in the context of ATM networks), rooted at the source S and spanning all the destinations (or endpoints). This is referred to as the *global multicast tree* in many occasions to distinguish it from the *local multicast tree* which serves a subset of the destinations and is a small portion of the global multicast tree. The global multicast tree identifies a multicast virtual circuit which we call an MVC. The MVC is shown by bold lines in Fig.2. The endpoints in the local exchange L_i are denoted by $E_{i,j}$. L_i is *not* an endpoint.

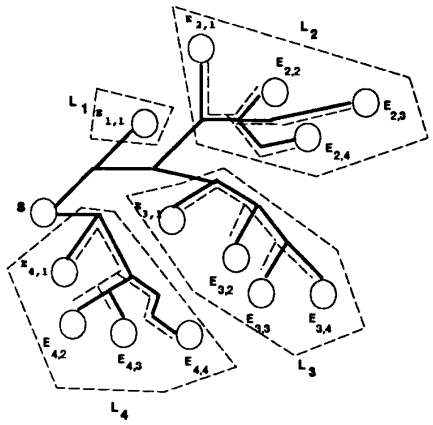

Figure 2: Global Multicast Tree rooted at S and Local Multicast Trees rooted at $E_{i,1}$'s

Finally, each protocol in this paper is described in terms of a point-to-multipoint multicast, although multipoint-to-multipoint multicast is also possible as long as the multicast trees are set up for each source.

3 Designated Status Protocol (DSP)

DSP is a protocol which chooses a particular destination in each local region as a representative of the region and that designated destination is responsible for sending the status messages on behalf of the region to the source and is also responsible for locally multicasting a packet if every destination in that region does not receive the packet.

More formally, the source picks an endpoint $E_{i,1}$, which can be thought of as the *representative* of the group of $E_{i,j}$'s, in each local exchange L_i (Fig.2). $E_{i,1}$ is the endpoint within L_i which is *closest* to S with respect to the global multicast tree among the $E_{i,j}$'s belonging to L_i. This $E_{i,1}$ is responsible for sending its *own* status to the source. Thus, if there are m LE's with final destinations (or endpoints), there will be m designated endpoints which are supposed to send their status to the source. In fact the status sent by $E_{i,1}$ will show which blocks[1] have been received by $E_{i,1}$. However, it will not say which blocks have been received by the $E_{i,j}$'s for $j \neq 1$. Each $E_{i,1}$ will be referred to as a *local source* (in contrast to the global source S) because it will multicast packets to the destinations in its local region (as opposed to all the destinations).

We assume that some estimate of round trip delay is available between S and each $E_{i,1}$ after the multicast tree is set up. We will refer to the round trip delay corresponding to $E_{i,1}$ by RTD_i. Peak bandwidth, packet size, block size and re-ordering buffer size required at each destination are also negotiated at the connection-establishment time.

We also assume that several *local multicast trees* are formed based on the global multicast tree. A local multicast tree rooted at $E_{i,1}$ is the portion of the global multicast tree spanning the $E_{i,j}$'s in the L_i. Local multicast trees are shown by dashed lines in Fig.2. Such a local multicast tree will be identified by a local multicast virtual circuit identifier, which we call an LMVCI.

[1] A block is a collection of packets and it is the unit chosen for selective repeat retransmission as described later in the paper. In the context of ATM network, there will be three levels of hierarchy: block, packet and cell.

3.1 Description of the protocol

Before going to the details of the packet formats and the detailed description of the protocol, we describe the main idea of the protocol in this sub-section. The protocol works as follows:

1. S multicasts data packets to *all* the destinations ($E_{i,j}$'s $\forall\, i,j$) using the global multicast tree. This multicast will be called a *global multicast.*

2. Each $E_{i,1}$ sends its *own* status to S in the form of control (status) packets at regular intervals. The status packet contains the information about which blocks have been received by $E_{i,1}$. S globally multicasts a block again if there exists an $E_{i,1}$ which has not received the block within the expected time. This time duration is, in general, a multiple of the round-trip-delay between S and the $E_{i,1}$ in question.

3. Each $E_{i,j}$ ($j \neq 1$) sends its status to the corresponding $E_{i,1}$ at regular intervals. $E_{i,1}$ locally multicasts a block if there exists an $E_{i,j}$ ($j \neq 1$) which has not received the block during global multicast. Note that an $E_{i,j}$ ($j \neq 1$) depends on the corresponding $E_{i,1}$ for the retransmission of a block if it did not receive the block either during the global multicast or during local multicasts.

4. S will multicast a *new* block provided each $E_{i,1}$ has available buffer space for new blocks. Note that an $E_{i,1}$ has two buffers: (1) a *reassembly* buffer which is used to assemble the packets it

receives from S and (2) a *retransmission* buffer which is used to retain the packets (blocks) that have not been received by all the $E_{i,j}$'s in its jurisdiction[2]. A block is transferred from the reassembly buffer to the retransmission buffer when there is space in the retransmission buffer. This transfer creates space in the reassembly buffer of $E_{i,1}$. If, however, both buffers are full, there is no space for new blocks. The status of space available in the reassembly buffer is sent to S as a part of the status message from $E_{i,1}$ to S so that S knows if it can multicast a new block.

A slightly modified version of the protocol in which the source multicasts to all the representatives $E_{i,1}$'s (instead of to all the $E_{i,j}$'s for all i, j) and the $E_{i,1}$'s multicast to the $E_{i,j}$'s ($j \neq 1$), thereby dividing the connection along each source destination pair into two segments: one from the source to $E_{i,1}$ and the other from the $E_{i,1}$ to $E_{i,j}$ ($j \neq 1$) can also considered. However, we do not consider it here.

The protocol involves several types of control packets and data packets in addition to the data structures which are modified on the reception of the control/data packets. In this paper, the formats of control and data packets are given. The details of data structures can be found in [PSL94].

3.2 Format of control and data packets

In this section, we give the detailed description of the packets involved in the protocol. The following packets are used in the protocol:

1. A control (status) packet from an $E_{i,1}$ to the source.
2. A control (status) packet from an $E_{i,j}$ ($j \neq 1$) to $E_{i,1}$.
3. A multicast data packet from source S to $E_{i,j}$'s $\forall i$, j.
4. A multicast data packet from an $E_{i,1}$ to $E_{i,j}$ $\forall j$ ($j \neq 1$).

The notation given in Table 1 is used to represent the fields of the control and data packets. Note that MVCI (Multicast Virtual Circuit Identifier) is reminiscent of VCI (Virtual Circuit Identifier) used in connection oriented networks. However, MVCI, as used in this paper, is more general and it simply identifies a specific multicast connection which may use a static allocation of resources at the connection establishment time or a dynamic allocation of resources as done by RSVP [ZDESZ93].

The packet formats are shown in Fig.3.

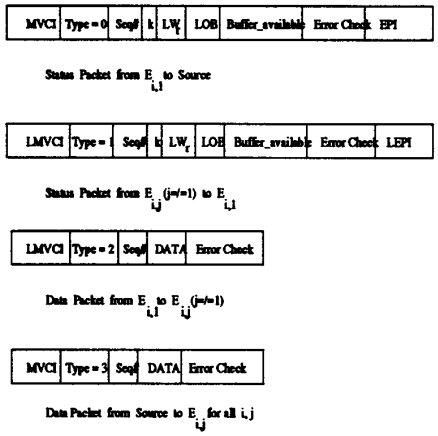

Figure 3: Packet Formats

4 Consolidated Status Protocol (CSP)

In DSP, an $E_{i,1}$ is chosen at each L_i and that designated endpoint sends its *own* status message to the source S. In CSP, a status message is sent to the source by each local exchange L_i and this status message is a *consolidated* status message of all the endpoints/destinations in its jurisdiction. Thus, CSP uses the local exchange while DSP uses a representative destination in a local region to send status messages to the source S. Also the designated destination in DSP sends its own status while the local exchange in CSP sends the status of all the destinations in the local region in a compact (consolidated) form.

The protocol works as follows:

1. S multicasts a block to all the destinations $E_{i,j}$'s $\forall\, i,j$ using the multicast tree which is set up at the connection establishment phase.
2. The $E_{i,j}$'s send their status at regular intervals to the corresponding local exchanges L_i's. The format of the status message is the same as that of the status message sent by $E_{i,j}$'s (for $j \neq 1$) to $E_{i,1}$ in DSP.
3. L_i consolidates the status messages from the $E_{i,j}$'s in its jurisdiction. Periodically, L_i sends

[2] $E_{i,j}$'s served by an L_i are said to be in the same jurisdiction.

Notation	Explanation
MVCI	a multicast virtual circuit identifier.
LMVCI	a local multicast virtual circuit identifier.
Type	type of a packet. type = 0 indicates a status packet from $E_{i,1}$ to the source. type = 1 indicates a status packet from $E_{i,j}$ $(j \neq 1)$ to $E_{i,1}$. type = 2 indicates a data packet from $E_{i,1}$ to $E_{i,j}$ $(j \neq 1)$. type = 3 indicates a data packet from S to $E_{i,j}$ $\forall\, i,j$.
Seq#	sequence number of a packet. It is incremented by sender each time a new packet is transmitted.
k	interval between two successive status packets in units of T_{IN} [NRS90]. $T_{IN} = max(\ RTD/kou, ipt)$. kou is a constant between 2 and 32. ipt is the interval between two successive packet transmissions.
LW_r	lower end of window in receiver, that is, the maximum sequence number of the block below which every packet in every block has been correctly received as known at the receiver. In the context of the global multicast tree, $E_{i,1}$'s are treated as receivers. For local multicast trees, $E_{i,j}$'s $(j \neq 1)$ are treated as receivers.
L	the largest allowed number of outstanding blocks chosen at the connection establishment time. An outstanding block is a block that has been multicast at least once but not yet been acknowledged by all the $E_{i,1}$'s.
LOB	list of outstanding blocks. A bit map representing the outstanding blocks between LW_r and $(LW_r\ + L\ - 1)$.
Buffer_ available	an integer representing the space available in the *reassembly* buffer of an $E_{i,1}$ in terms of the number of blocks.
EPI	an endpoint identifier which is used to identify an $E_{i,1}$.
LEPI	a local endpoint identifier which is used to identify an $E_{i,j}$ $(j \neq 1)$.

Table 1: Explanation of Notation used in Packet Formats

its consolidated status to S.

We will describe next how L_i prepares a consolidated LW_r, a consolidated LOB and a consolidated $buffer\_available$ from the corresponding fields of individual status messages of $E_{i,j}$'s. First we describe how the consolidated LW_r is formed.

$LW_{r(consolidated)} = \min\ (LW_r^{(j)})$ where $LW_r^{(j)}$ is the LW_r field of the status message of $E_{i,j}$ to L_i. That is, L_i chooses the lowest value of LW_r among the LW_r's of the $E_{i,j}$'s status messages as the $LW_{r(consolidated)}$. The $LOB_{consolidated}$ is a bitwise AND of the *properly aligned* LOB's of the $E_{i,j}$'s.

The $buffer\_available$ field in the consolidated status of L_i is the minimum of the

$buffer\_available$ values at the $E_{i,j}$'s in its jurisdiction.

L_i sends the consolidated status of the destinations in its jurisdiction to S. The format of this status message is exactly the same as that of the status message sent to S by the $E_{i,1}$'s in DSP.

4. S re-multicasts a block to all the destinations if there exists an $E_{i,j}$ that has not received the block within the expected time. However, S could retransmit the block only to the specific $E_{i,j}$, who has not received the block, provided point-to-point connections were available in addition to the point-to-multipoint connections.

5. S multicasts a *new* block provided buffer space is available in each L_i for receiving a new block. Here, if every endpoint $E_{i,j}$ in the local exchange L_i has buffer available for a new block, we say that the L_i has buffer space available for a new block.

   Note that we are trying to avoid using the L_i's as *store* and *forward* nodes because the assembly of packets at the L_i's will introduce unnecessary delay and unnecessary complexity at the switches.

5 Combined Protocol (CP)

CP combines the concept of *local multicasting* from DSP with the notion of sending a *consolidated status message* from CSP. Thus CP uses the local exchange (like CSP) to combine the status messages of the destinations in the local region and also uses a destination in each local region (like DSP) to locally multicast

packets. However, the destination used for local multicasting (which is also called the local source) may change dynamically unlike DSP in which such a destination is predetermined. The reason for dynamically altering the local source is given below.

The local multicast trees in DSP are set up at the connection establishment time and hence the $E_{i,1}$'s remain unchanged during the entire multicast from S. This may lead to a situation in which an $E_{i,1}$ has not received a block but some other $E_{i,j}$ in the same jurisdiction (that is, within the same L_i) has received that block. In such a situation, S will globally remulticast the block in DSP, leading to a redundant global retransmission. Such a redundant retransmission is prevented in CP by a local multicast by the destination $E_{i,j}$ which has received the block.

CP prevents redundant global retransmissions by dynamically choosing the local source and the local multicast tree. The main ideas are as follows:

1. S multicasts a block to all the destinations using the global multicast tree that is chosen at the connection establishment time.

2. The endpoints $E_{i,j}$'s send their status to the corresponding local exchange L_i exactly as in CSP. L_i sends a *consolidated* status to the source S. However, the consolidation of status is done differently in CP than in CSP.

   $LW_{r(consolidated)} = \max(LW_r^{(j)})$ where $LW_r^{(j)}$ is the LW_r field of the status message of $E_{i,j}$ to L_i. That is, L_i chooses the highest value of LW_r among the LW_r's of the $E_{i,j}$'s status messages as the $LW_{r(consolidated)}$. $LOB_{consolidated}$ is a bitwise OR of the *properly aligned* LOB's of the $E_{i,j}$'s. In some sense, CP is an *optimistic* protocol in contrast to CSP which is *pessimistic*.

   The $buffer\_available$ field in the consolidated status of L_i is the minimum of the

   $buffer\_available$ at the $E_{i,j}$'s in its jurisdiction.

   L_i sends the consolidated status of the destinations in its jurisdiction to S. The format of this status message is exactly the same as that of the status message sent to S by the $E_{i,1}$'s in DSP.

   Since the L_i has the global status of the destinations in its jurisdiction, it knows which blocks have been received by which endpoints and which blocks have not been received by all the endpoints in its jurisdiction. Based on this information, the L_i creates a local multicast tree rooted at the $E_{i,j}$ which has received a given block not received by every destination in the L_i. Let us refer to that particular $E_{i,j}$ as $E_{i,j_{source}}$.

   L_i informs the $E_{i,j_{source}}$ about the block(s) it needs to multicast locally to the destinations in L_i. $E_{i,j_{source}}$ multicasts repeatedly to the $E_{i,j}$'s until told to stop by L_i which has the aggregate status of the $E_{i,j}$'s in its jurisdiction.

   In fact, the destination $E_{i,j}$'s keep sending their status to L_i and *not* to the $E_{i,j_{source}}$. L_i collects the status of the destinations in its jurisdiction and creates a local multicast tree dynamically.

3. S re-multicasts a block if some L_i has not received the block. Here, if no $E_{i,j}$ within an L_i has received a given block, we say that the L_i has not received the block.

4. S multicasts a *new* block if each L_i has buffer space available for new blocks. In this context, if $buffer\_available > 0$ for each $E_{i,j}$ in an L_i, we say that buffer space is available for the L_i.

Term	Explanation
n	number of destinations
b	block error rate
b_n	aggregate block error rate for n destinations
W	window size
$\bar{t}_s$	average time to receive a status message indicating a successful reception by all the destinations
$\bar{w}_s$	number of blocks that fit in $\bar{t}_s$
r_f	retransmission timeout factor

Table 2: Explanation of Terms used in Performance Analysis

6 Analysis of Performance

In this section, we present the result of a simple analysis for computing the throughput of our multicast transport protocols. Our objective is to show the dependence of throughput on window size (buffer size), error rate, transmission speed, retransmission timeout and number of destinations. This analysis is built on the analysis of the SNR protocol by [DJNS93]. We summarize the notation used in this paper in Table 2.

Without going into the details (which appears in [PKL94]), the normalized throughput ρ can be expressed as:

$$\rho = \frac{W + b_n^{-1}}{w_x + b_n^{-1}} \quad (1)$$

where

$$b_n \approx b * n \quad (2)$$

and

$$w_x = (1 - \frac{ln(n)}{ln(b)}) * (r_f * \bar{w}_s + 1) + w_{s_{max}} \quad (3)$$

Combining (1), (2) and (3), we have:

$$\rho = \frac{(b * n)^{-1} + W}{(b * n)^{-1} + (1 - \frac{ln(n)}{ln(b)}) * (r_f * \bar{w}_s + 1) + w_{s_{max}}} \quad (4)$$

6.1 Observations

1. If $n = 1$, $\bar{w}_s = w_{s_{max}}$ and (4) reduces to:

$$\rho = \frac{b^{-1} + W}{b^{-1} + (r_f * w_s^{(1)} + 1) + w_s^{(1)}} \quad (5)$$

   which is the same as the expression for the throughput of SNR protocol for single source and single destination in [DJNS93].

2. If $W \leq w_x$ and $(b_n)^{-1} \gg w_x$, then from (1), $\rho \approx 1$.

   That is, if the aggregate error probability for n destinations is very low, then the normalized throughput is independent of other factors and is close to 1. Simulation results agree with this (Fig.11).

3. As W increases, ρ increases as well until $\rho = 1$ for $W = w_x$. Fig.4 shows the agreement of simulation results with this observation. This also means that wasted bandwidth decreases as the window size increases.

4. As the retransmission timeout duration increases, that is, as r_f increases, w_x increases (refer to (3)) while W remains unchanged and hence ρ decreases (refer to (1)). Simulation studies show this effect as well (Fig.9).

5. Since n (number of destinations) occurs as n^{-1} in both the numerator and the denominator and it occurs as $ln(n)$ in the denominator (refer to (4)), its effect is not significant on ρ, although ρ decreases slightly as n increases. This decrease in ρ can be attributed to the $ln(n)$ term in the denominator. Simulation results agree with this (Fig.12).

6. As transmission speed increases, $\bar{w}_s$ as well as $w_{s_{max}}$ increases and hence ρ decreases. Simulation results show this effect in Fig.10.

7 Simulation Results

We did a simulation for each of the three protocols assuming six local exchanges with fixed round trip delays of 10 ms, 25 ms, 30 ms, 40 ms, 50 ms and 60 ms which approximately correspond to the round trip delays between New Jersey and each of Massachusetts, Florida, Illinois, Texas, Colorado and California, respectively. In each of the areas, we have 3 to 5 local destinations. In total, we have 25 destinations. The parameters in the simulation are window size, transmission speed, retransmission timeout, error rate and number of destinations. These parameters are varied to study their influence on the throughput, delay, acknowledgment traffic, re-multicast traffic and the fraction of bandwidth wasted. In Figures 4 through 8, the parameters used in the simulation are: Transmission Speed = 150 Mbps, Block Size = 1024 bytes, Retransmission Timeout Factor r_f = 1.2, Block Error Rate = 10^{-3} and State Exchange Rate = once every 4 to 6 ms. In Figures 9 through 12, the window size is set to 2 times the maximum round trip delay (in number of blocks). The results are summarized below:

1. As the window size varies from 0.5 to 3.0, the normalized throughput (ρ) increases for each of the three protocols (Fig.4). This happens because the source can multicast for a longer time before the window closes and hence can utilize the bandwidth better. However, CP achieves the maximal throughput at a much lower window size of 1.0 as compared to DSP and CSP which attain maximal throughput for window sizes of 2.5. We also observe that,

   $\text{Throughput}_{CSP} \leq \text{Throughput}_{DSP} \leq \text{Throughput}_{CP}$

2. The end-to-end delay is constant with the variation of window size (Fig.5). However,

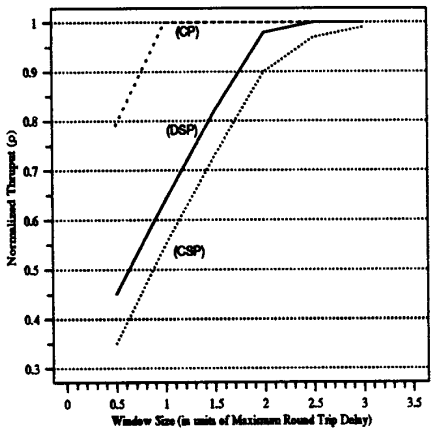

Figure 4: Normalized Throughput vs. Window Size

$$\text{Delay}_{CP} = \text{Delay}_{DSP} \leq \text{Delay}_{CSP}$$

Delay_{CSP} is expected to be higher because the source does all the re-multicasting and once a block is not received, the destination has to wait for at least another round trip delay. However, in DSP and CP, once a block is not received, the destination only has to wait for a local round trip delay.

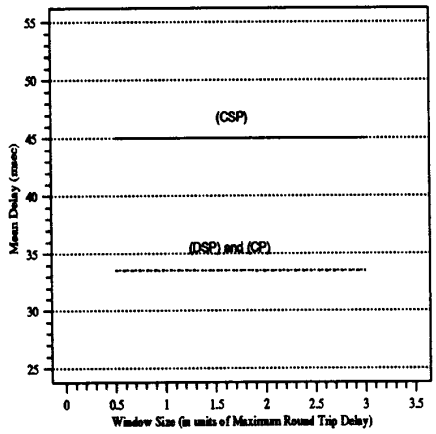

Figure 5: End-to-end Delay vs. Window Size

3. Global re-multicast traffic for CSP is higher than that of DSP and CP (Fig.6). This happens because the source multicasts if any destination does not receive a block in CSP, while in the other two protocols, some of the lost blocks are re-multicast locally. Global re-multicast traffic for CP is zero in this simulation because a globally multicast block was *always* received, during the simulation, by at least one destination in every local exchange. This is highly likely if the number of destinations in a local exchange is high and the error rate is low.

   Local re-multicast traffic for CP is higher than that for DSP because the re-multicast in DSP is done locally only when the local source $E_{i,1}$ receives the block while in CP the local re-multicast is done when any one of the destinations in a local exchange receives the block (Fig. 7).

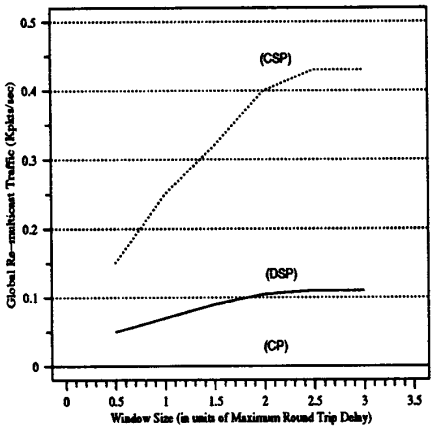

Figure 6: Global Re-multicast Traffic vs. Window Size

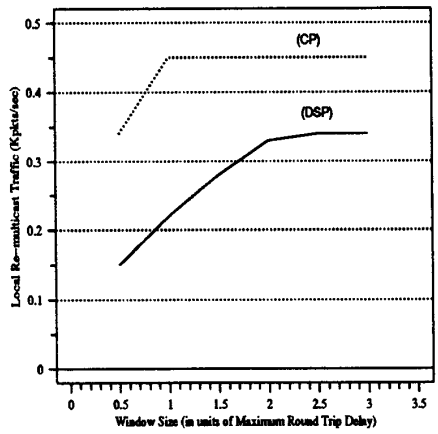

Figure 7: Local Re-multicast Traffic vs. Window Size

4. Status message traffic (or Acknowledgment traffic) remains *constant* in each of the three protocols

and moreover, they are the same (Fig.8). This happens because in each case, status messages are exchanged at exactly the same regular intervals (like every 5ms) between the source S and the local exchanges (or $E_{i,1}$'s). In contrast to the actual throughput ($\approx$ 20000 pkts/sec in a typical case), the status message traffic ($\approx$ 1500 pkts/sec in a typical case) is quite small.

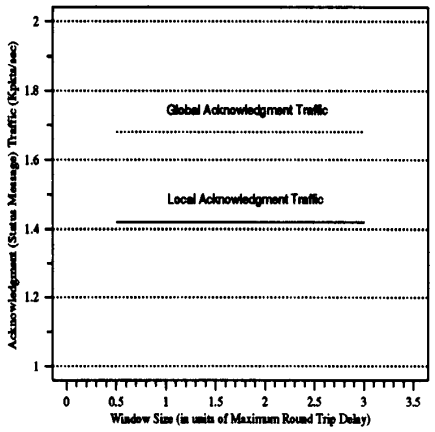

Figure 8: Acknowledgment Traffic vs. Window Size

5. Throughput decreases with the increase of retransmission timeout interval (Fig.9). This happens because the source has to wait for a longer time before it knows of an unsuccessful multicast. However, as Fig. 9 shows,

   $\text{Throughput}_{CSP} \leq \text{Throughput}_{DSP} \leq \text{Throughput}_{CP}$

   Throughput_{CP} does not fall with the increase in retransmission timeout because the source almost never needs to re-multicast a block, since the probability of at least one local destination in an area receiving the block is very high.

6. As the transmission speed increases (that is, the delay bandwidth product increases), the throughput falls for a given window size because the source cannot multicast as many blocks as it is capable of because of the fixed window size. The results are given in Fig.10.

7. Throughput performance of all three protocols suffers badly as the error rate is increased from 10^{-4} to 10^{-2} (Fig.11). DSP and CP give maximal throughput for error rates $\leq 10^{-3}$, while CSP gives maximal throughput for error rates $\leq 10^{-4}$. However, even in this case,

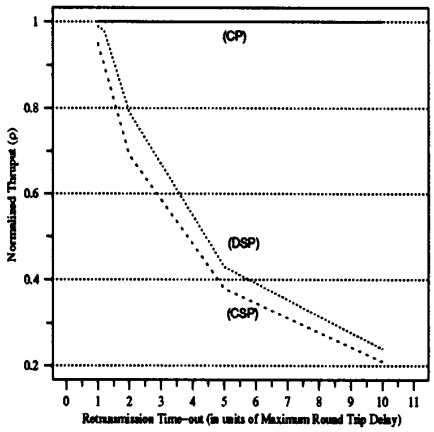

Figure 9: Normalized Throughput vs. Retransmission Timeout

Parameters	
Throughput	CSP $\leq$ DSP $\leq$ CP
End-to-end Delay	DSP = CP $\leq$ CSP
Buffer Requirement	CSP $\leq$ DSP $\leq$ CP
Global Re-multicast Traffic	CP $\leq$ DSP $\leq$ CSP
Local Re-multicast Traffic	DSP $\leq$ CP; Does not apply to CSP
Acknowledgment Traffic	DSP = CSP = CP
Processing Complexity	CSP $\leq$ DSP $\leq$ CP
ATM Switch Modification	Not required for DSP; required for CSP and CP

Table 3: Comparison Chart for the Protocols

   $\text{Throughput}_{CSP} \leq \text{Throughput}_{DSP} \leq \text{Throughput}_{CP}$

8. The performance of the protocols is not very sensitive to the number of destinations although the throughput falls slightly as the number of destinations increases (Fig.12).

8 Comparison Chart for DSP, CSP and CP

In this section, we present a chart (Table 3) comparing the performance of the three protocols and based on the comparison, draw some conclusions.

Based on the performance study of the protocols, we observe that CP has the best performance, followed by DSP and CSP. However, the improvement

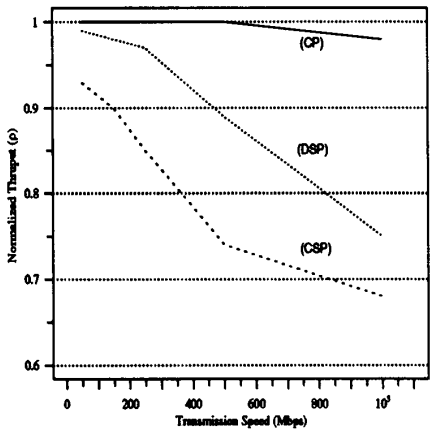

Figure 10: Normalized Throughput vs. Transmission Speed

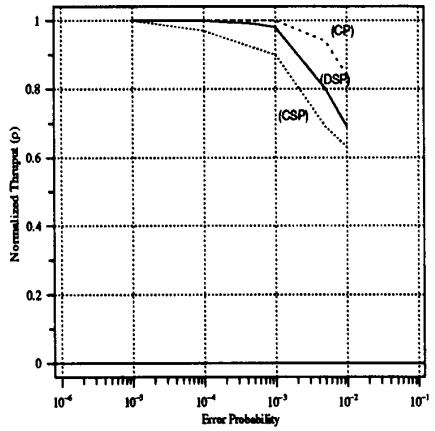

Figure 11: Normalized Throughput vs. Error Rate

in performance is not without price. CP has substantially more overhead than the other protocols in terms of computation as well as communication, which did not show up in the simulation results. In addition to that, CP requires a modification of the ATM switch, as does CSP, so that the switch at each local exchange can combine the *acks* from the local destinations.

DSP performs better than CSP because of the distribution of the re-multicasting traffic between the global source S and the local sources $E_{i,1}$'s. Also, it will perform better than CP in real implementations unless the computation and communication delay associated with the multicast of every packet is under a certain threshold, which is very unlikely at present time. DSP also has the inherent simplicity of choosing

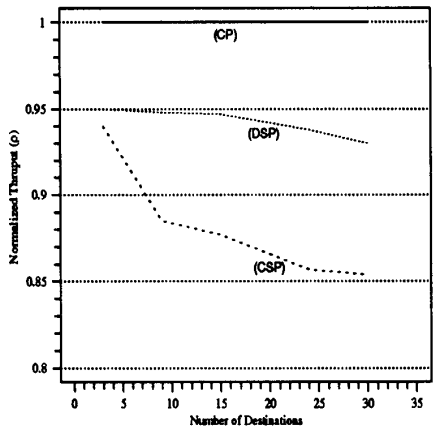

Figure 12: Normalized Throughput vs. Number of Destinations

the global and local multicast trees at the connection set up time and using them as long as the connection is active. Furthermore, it is the only protocol among the three that does not require any modification of the ATM switch to support multicast. In addition, DSP can satisfy throughput and delay requirements for most of the real applications. Considering all these factors, DSP becomes the protocol of choice.

To obtain high normalized throughput, the window size (and hence the buffer size) needs to be 2.5 to 3 times the maximum round trip delay (in number of blocks) which may be impractical in many situations. In that case, a proper buffer size may be chosen to obtain lower than maximum normalized throughput (say, 0.8) but retain the desired absolute throughput (say, 10000 pkts/sec) using either the simulation plot (Fig. 4) or the expression for approximate normalized throughput ((4)).

9 Conclusion

In this paper, we presented the design of three reliable multicast transport protocols, performed a simulation study for each of them, derived an expression for approximate throughput of the protocols, and compared the performance of the protocols based on various parameters like throughput, end-to-end delay, re-multicasting traffic, acknowledgment traffic and processing complexity. Various practical considerations along with the performance figures led us to conclude that DSP is the best reliable multicast transport pro-

tocol among the three alternatives.

The main contributions of the designs are in reducing the acknowledgment traffic by combining the *acks* along a multicast tree, in reducing unnecessary retransmissions by performing local multicasts, in reducing complex error handling mecahnisms by using the periodic exchange of complete state information between the source and the destinations as in SNR protocol and in obtaining high throughput by a combination of block-based selective repeat retransmission and localized retransmissions.

Acknowledgment

We are extremely grateful to Bharat Doshi, Praveen Johri, Henning Schulzrinne and Tom LaPorta for their valuable comments on this piece of work.

References

[A84] Lorenzo Aguilar, "Datagram Routing for Internet Multicasing", ACM Computer Communications Review, Vol. 14, No. 2, 1984, Pages 58-63.

[AFM92] S. Armstrong, A. Freier and K. Marzullo, "Multicast Transport Protocol," Network Working Group RFC 1302.

[APR93] R. Aiello, E. Pagani and G. P. Rossi, "Design of a Reliable Multicast Protocol," Proceedings of IEEE INFOCOM '93.

[CM84] Jo-Mei Chang and N.F. Maxemchuk, "Reliable Broadcast protocols," ACM Transactions on Computer Systems, Vol. 2, No. 3, August 1984, Pages 251-173.

[D88] Stephen E. Deering, "Multicast Routing in Inter Networks and Extended LANs," ACM Computer Communications Review, Vol. 19, No. 4, 1988, pages 55-64.

[DJNS93] B. T. Doshi, P. K. Johri, A. N. Netravali and K. K. Sabnani, "Error and Flow Control Performance of a High Speed Protocol," IEEE Transactions on Communications, May 1993.

[EY88] K.Y. Eng and Y.S. Yeh, "Multicast and Broadcast Services in a Knockout Packet Switch," Proceedings of IEEE INFOCOM '88.

[LMS93] A. M. Lapone, N. F. Maxemchuk and H. G. Schulzrinne, "The Bell Laboratories Network Emulator," Bell Labs Technical Memorandum 11382-930913-64TM.

[NRS90] A. N. Netravali, W. D. Roome and K. Sabnani, "Design and Implementation of a High Speed Transport Protocol," IEEE Transactions on Communications, Vol.38. No.11, November 1990.

[PP92] C. Patridge and S. Pink, "An Implementation of the Revised Internet Stream Protocol (ST-2)," Journal of Internetworking: Research and Experience, Vol.4, No.1, March 1992.

[PSL94] S. Paul, K. Sabnani and J. Lin, "Multicast Transport Protocols for High Speed Networks - from design to implementation," submitted for publication.

[SS85] K. Sabnani and M. Schwartz, "Multidestination Protocols for Satellite Broadcast Channels," IEEE Transactions on Communications, Vol. COM-33, No. 3, March 1985.

[TL90] S-C. Tu and W-H.F. Leung, "Multicast Connection-Oriented Packet Switching Networks ," Proceedings of SUPERCOMM International Conference on Communications (ICC '90).

[ZDESZ93] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zapppala, "RSVP: A New Resource ReSerVation Protocol," IEEE Networks Magazine, September 1993.