Effect of Topology on Performance of Reliable Multicast Communication*

Pravin Bhagwat† Partho P. Mishra‡ Satish K. Tripathi†

Abstract

In this paper we examine the performance implications of providing reliability in conjunction with multicast transport over a high speed wide area network. We use a block based acknowledgement and selective retransmission protocol to evaluate the impact of the loss rate and the multicast tree topology on the achievable throughput. Our results show that even when the buffer overflow probability at switches and receivers is low, the cumulative loss probability seen by a source may be quite high. We also demonstrate that the average throughput increases significantly if the transport protocol delivers packets to the application layer out-of-sequence. We investigate the scaling properties of the error control mechanism and show that the multicast tree topology that results in minimum transfer time is not necessarily the same as the one constructed using minimal bandwidth or shortest path algorithms.

1 Introduction

Applications involving multi-user interaction are expected to consume a significant portion of the bandwidth of future high speed networks. Examples of such applications include scientific visualization, publishing, cooperative work etc. These applications involve *one-to-many* and *many-to-many* communication models as opposed to the traditional *one-to-one* communication model that is commonly used today. To provide an efficient way to support such applications, network layer protocols usually provide *multicast* support. This involves constructing a special multicast routing tree with data packets being replicated at a node of the multicast routing tree only if there are multiple outgoing edges from that node. The use of such a multicast paradigm requires the reevaluation of certain design choices originally made in the context of applications involving using a one-to-one communication model. One of these concerns the choice of the *error control* [4] mechanism.

Much of the earlier work on error-control for multipoint communication has focussed on designing protocols in the context of communication over broadcast channels [2, 12, 14, 16]. As a result, the design choices and insights do not always apply to a wide area packet switched network. For example, most of the analytical models typically assume the occurence of only two kinds of losses either in the broadcast medium or at the receiver. However, in a wide area packet switched network a packet loss can occur at any node of the multicast tree and result in a subset of receivers not receiving a copy of the packet. Also, the common assumption that each receiver in the multicast group has buffering greater than the delay bandwidth product does not always hold in wide-area networks. Our work differs from above mentioned work in several ways. First, we explicitly consider the topology of the multicast tree in computing the cumulative loss probability and the net effect on the throughput. Second, we investigate the operating regime where transfer sizes are large and the receivers have limited buffering. Third, we examine the impact of delivering packets out of sequence on the achievable throughput.

The rest of this paper is organized as follows. In section 2 we outline the problem that we study and describe our protocol. In Section 3 we describe the analytic and simulation techniques we use to study the performance of the proposed protocol. Section 4 presents the results and the implications of the results on the choice of error control strategies for multicast image transfers in high bandwidth delay product networks. Section 5 concludes the paper with a brief discussion.

2 Problem description

We consider the problem of reliable information transfer for text and image applications involving one-to-many or many-to-many communication. The communication is assumed to proceed via a high-speed wide area packet switched network. We assume that the network layer protocols provide multicast support (as in IP Multicast [8]) by constructing a multicast tree for routing. We assume that errors due to transmission or switching errors are very rare and that the main cause of data loss is buffer overflow at the routers or hosts. The main metric of performance that we use is the transfer time which depends on the average throughput seen by a session. The average throughput seen by a session depends on both the available bandwidth and the packet loss characteristics. In this paper, we study the latter effect assuming that a rate based congestion control mechanism is used to ensure that the available bandwidth is not exceeded. We assume that a retransmission based scheme is used to ensure reliable delivery.

*This work was supported in part by TASC.

†Department of Computer Science, University of Maryland, College Park, MD 20742.

‡AT&T Bell Laboratories, Murray Hill, New Jersey

In retransmission based schemes the receiver(s) sends information to the source about the status of individual packets. The source identifies lost packets based on this information and then initiates the error recovery procedure. Two types of schemes may be used to communicate information from the receiver to the source - we distinguish between them as acknowledgement (ACK) and negative acknowledgements (NACK) based schemes. In an ACK based scheme, the receiver sends information about which packets it has received to the sender. This information is either sent periodically or on the receipt of a certain amount of data. In cumulative ACK schemes the receiver informs the sender about the highest sequence number below which all packets have been successfully received. In selective ACK schemes, the receiver informs the source about the sequence number of the packet(s) that it has received since the last ack packet was sent. We use the term block based ACK schemes to describe those schemes in which a block of packets rather than an individual packet is acknowledged by the receiver. In NACK based schemes, the receiver sends a message to the sender only if it realizes that a packet has been lost. When a sender realizes that packets have been lost it can adopt a variety of retransmission strategies - two of the more commonly used ones are *go-back-n* and *selective retransmission*. In a go-back-n scheme, the source retransmits all packets starting from the first packet lost. In a selective retransmission scheme, only the lost packets are retransmitted.

For multicast communication we assume that each receiver sends acknowledgements to the sender. As a result there is an implosion effect at the sender as the size of the receiver set increases. While strategies have been proposed to aggregate these acks [15] or for a single receiver to multicast acks [1], the feasibility of these ideas in a WAN environment is unclear. We believe that it is appropriate to reduce the amount of reverse traffic by using either a block based ACK scheme [9] or a NACK scheme [11]. While NACK based schemes are likely to generate a lesser amount of reverse traffic than block based ACK schemes, a purely NACK based scheme cannot guarantee complete reliability[1]. This is because there may be situations where the sender is unaware that a receiver has not received a packet[2]. Hence a block based ACK scheme seems a more appropriate choice. For the purpose of retransmission a selective retransmission strategy seems appropriate. Although a go-back-n mechanism requires very simple functionality on the part of receivers, namely no packet resequencing ability, it wastes bandwidth and increases the total transfer time in a WAN environment [17].

For multicast sessions a packet lost at a router may not be received by a subset of receivers. Hence the packet has to to be retransmitted to these receivers. There are three ways in which a lost packet may be retransmitted. The first possibility is to set up a new multicast group whose members are the receivers which did not receive the packet. The latency involved in performing such an operation over a wide-area-network makes it unfeasible. A second possibility is to unicast the lost packet to each of the receivers that lost the packet. This is an attractive option if a small number of receivers in the multicast group lose most of the packets. However, when packet losses occurs at routers, several hosts may not receive a packet necessitating transmission of multiple copies of the same packet. This results in a reduction of throughput and wastage of bandwidth. A third possibility, which is most often used [1, 3, 14], is to multicast the lost packet to the entire group of receivers. This is the simplest strategy and will perform best if more than a certain number of receivers do not receive a packet [7].

[1] Hybrid schemes are possible where a NACK based scheme is augmented with a polling mechanism used to force ACKS.

[2] for example if a negative ack packet is lost.

2.1 Protocol description

The protocol that we have chosen to study reflects the observations made above. We next describe a generic protocol which will be used as a platform for the analysis in the rest of the paper. We would like to emphasize that this is not meant to be a *new* protocol but rather to capture the general characteristics of block based selective retransmission protocols that have earlier been proposed for point to point connection over high speed networks [6, 9].

The protocol that we model uses a block based acknowledgement scheme, selective retransmissions and multicasts lost packets. The sender is assumed to *know* all the receivers and provide reliable delivery of data to all of them. Blocks may either be delivered in-sequence or out-of-sequence, but packets within a block are delivered in-sequence[3]. Each packet is identified by a block number and an offset number defining its position within the block. The sender follows a variant of a stop-and-wait protocol: it transmits all the packets within a block, sends a special control packet to force the generation of ack packets from the receivers and then waits to receive acks from all the receivers. It is assumed that the control packets are sent at a higher level of priority or over a separate channel (out of band signaling). Alternatively, the sender could set a flag in the last packet of a block to force receivers to send acks. If the control packet or the specially marked last packet is lost, the sender would time out and resend the packet. This would continue until the sender received acks from all the receivers.

Each receiver in the multicast group maintains a β bit-vector, where β is the number of packets in a block. When a packet is received the corresponding bit is set in the bit-vector. The ack packet sent by a receiver contains the number of the block being acknowledged and the bit vector for that block. By examining the acknowledgements from each receiver, the sender identifies lost packets and schedules them for retransmission. If no packets are lost, then the transmission of

[3] Out-of-sequence delivery of blocks is quite useful for certain categories of applications, such as image transfers [5].

a block is complete and the sender can send the next block. The same technique is used to force acknowledgements while retransmitting packets. When processing the acknowledgements for retransmitted packets the sender only checks the acks from receivers that did not receive the packet originally. We assume that a fixed number of blocks can be in transmission simultaneously - this is analogous to the usual concept of a window. The number of blocks that can be in transmission simultaneously and the number of packets in each block depends on the available buffering at the receiver - these values are negotiated at connection set-up time.

3 Analysis

Next, we describe the analytical model that we use for studying the effect of the system parameters on the observed performance. The primary metric of performance that we use is the average throughput seen by a connection. The parameters of the system are the block size, the receiver buffer size, the topology of the multicast tree and the packet loss probabilities at the routers and hosts. We model the effect of contention with other sessions indirectly by limiting the maximum sending rate of a session (as measured by the inter packet gap) and assuming certain loss probability characteristics.

In our analytical model we assume that packets are of a fixed size. We do not explicitly model retransmission timeouts or the effects of ack packet losses; such an event would lead to a timeout driven retransmission of the control packet used to force an ack from receivers and is hard to model. We assume that a rate based flow control mechanism is used and that the rate is not adjusted during the duration of the connection - while this may not always be true in an actual network it allows us to isolate the effects of error control and congestion control.

3.1 Analytical model

To compute the average throughput, we first compute the number of transmission attempts until a copy of a single packet is received by all receivers in the multicast group. Assume that source S is the root of a multicast tree $\mathcal{M}$ which interconnects the set of multicast group members. In such a system, packet loss events at various nodes of the tree are not independent since a packet lost at a router prevents all the receivers within its subtree from receiving a copy of that packet. However, it turns out that there is a simple way to compute the probability distribution function (pdf) of the number of *packet* transmission attempts. We use it to compute the average number of rounds required to reliably deliver a *block* consisting of β packets. Finally, we derive an expression for the total time required to multicast N packets from S to the multicast group as a function of the block size. We introduce the following notation:

$\mathcal{M}$	Multicast tree rooted at source S
N	Packets to be multicast from S
β	Number of packets in a block
t_r	Inter-packet spacing.
$T_\beta^{\mathcal{M}}(N)$	Time to reliably multicast N packets over $\mathcal{M}$ when block size is β
$R(n)$	Number of transmission of a packet to node $n \in \mathcal{M}$ until accepted by all nodes in the sub-tree rooted at n
F_n	Probability distribution function of $R(n)$, i.e., $F_n(j) = P[R(n) \leq j]$
$child^{\mathcal{M}}(n)$	Set of children of node $n \in \mathcal{M}$.
p_n	Packet loss probability at node $n \in \mathcal{M}$
p_S^{eff}	Effective packet loss probability as seen from S (i.e., root of $\mathcal{M}$)
$rounds(\beta)$	Number of transmission of a block initially containing β packets until accepted by all nodes $\in \mathcal{M}$

3.2 Distribution of $R(S)$

For a node n, let F_n denote the probability distribution function of $R(n)$, i.e.,

$$F_n(i) = P[R(n) \leq i]$$

$F_n(i)$ can be computed recursively in terms of the distribution functions of its children. There are two cases to consider :

Case 1: n is a leaf node.

p_n^i is the probability that all i attempts to deliver a packet to node n will fail. Therefore, the probability that fewer than $i+1$ rounds will be needed to deliver a packet to node n is given by:

$$F_n(i) = P[R(n) \leq i] = 1 - p_n^i \tag{1}$$

Case 2: n is not a leaf node.

In this case, $F_n(i)$ can be computed in terms of p_n and $F_k(i)$ where $k \in child^{\mathcal{M}}(n)$. Suppose i attempts are made to deliver a packet to node n. The probability that out of i delivery attempts the packet would get dropped at node n exactly u times is $p_n^u(1-p_n)^{i-u}$. In the remaining $i-u$ attempts, node n would forward a copy of this packet to its children. The conditional probability that the packet would be received successfully by all receivers in the subtree rooted at n is[4] $\prod_{k \in child^{\mathcal{M}}(n)} F_k(i-u)$. The exact expression for $F_n(i)$ may now be derived by taking summation over all possible values of u :

[4] since packet loses in subtrees rooted at $k \in child^{\mathcal{M}}(n)$ are mutually independent.

$$F_n(i) = \sum_{u=0}^{i-1} \left(\begin{array}{c} i \\ u \end{array} \right) p_n^u (1-p_n)^{i-u} \prod_{k \in child^{\mathcal{M}}(n)} F_k(i-u) \quad (2)$$

Given any arbitrary multicast tree $\mathcal{M}$ and packet loss probabilities(p_n) at each node n, the above recursive equation 2 along with the terminating condition 1 allows us to compute the distribution function for a node in terms of the distribution function of its children. Thus, using a bottom-up evaluation technique we can compute $F_S(i)$ - the distribution function as seen from the root of the multicast tree. $F_S(i)$ can be interpreted as the probability that $\leq i$ trials would be required to deliver a copy of a packet from source S to all receivers in the multicast group. Therefore, the average number of times a packet has to be retransmitted by the sender S before it is received at least once by all receivers in the multicast group is given by:

$$E[R(S)] = \sum_{i=0}^{\infty}(1 - F_S(i))$$

The effective packet loss probability as seen from the source S can now be estimated from the above expression:

$$p_S^{eff} = 1 - \frac{1}{E[R(s)]}$$

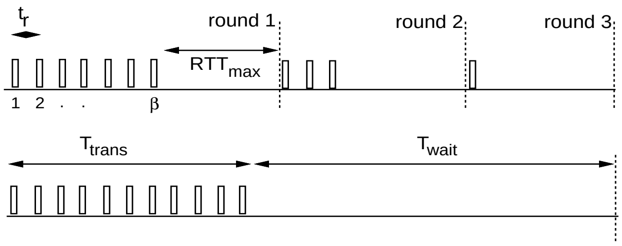

Figure 1: Block transmission cycle

3.3 Transfer time computation

As mentioned earlier, the protocol that we consider is a variant of stop-and-wait protocol and consists of two phases: the transmission phase and the wait phase (See Figure 1). The sender first transmits β packets in a block and then waits for an ACK from all receivers. The length of the wait period is determined by the receiver which is located furthest from the source (let it be denoted by RTT_{max}). Since the inter-packet gap is t_r, the time taken to transmit β packets is $\beta * t_r$. The time to transmit the control packet is assumed to be negligible and is not explicitly modeled in the analysis. Out of β packets transmitted during the first round, a few may be lost and may require retransmission. Let $rounds(\beta)$ denote the number of transmission of a block initially containing β packets until accepted by all receivers. The random variable $rounds(\beta)$ is the β^{th}-order statistic of $R(S)$. Therefore :

$$P[rounds(\beta) \leq j] = F_S(j)^{\beta}$$

The expected number of rounds can easily be computed as:

$$E[rounds(\beta)] = \sum_{j=0}^{\infty} P[rounds(\beta) > j]$$

$$= \sum_{j=0}^{\infty}(1 - F_S(j)^{\beta})$$

The block transmission cycle time can be broken up into two components (see Figure 1): T_{trans}, the time spent transmitting packets and T_{wait}, the time spent waiting for acks. On an average, each packet requires E[R(S)] transmissions for successful delivery. Therefore, the average transmission time for a block of size β is

$$E[T_{trans}] = \beta t_r E[R(s)]$$

The average time spent waiting for ack packets is proportional to the expected number of transmission rounds and is given by

$$E[T_{wait}] = RTT_{max} E[rounds(\beta)]$$

$\frac{N}{\beta}$ block transmission cycles are required to complete the multicast of N packets. Therefore, the average time to multicast N packets is :

$$T_{\beta}^{\mathcal{M}}(N) = (E[T_{trans}] + E[T_{wait}])\frac{N}{\beta}$$

or

$$T_{\beta}^{\mathcal{M}}(N) = N t_r E[R(s)] + RTT_{max} E[rounds(\beta)]\frac{N}{\beta} \quad (3)$$

If the chosen value of block size is small and the multicast session is running over a high speed wide area network, it is very likely that $E[T_{wait}] > E[T_{trans}]$. Under such conditions, throughput can be improved by overlapping the block transmission cycles of W blocks, where $W = \lfloor \frac{E[T_{wait}]}{E[T_{trans}]} \rfloor$. Of course, the buffer space required at each receiver would also increase by a factor of W. There are two possibilities depending on how the transport layer delivers these blocks to the application. If the blocks are delivered in-sequence, it is hard to analytically capture the dynamics of multiple parallel sessions. We use simulations to study the performance in this case. However, if the blocks are allowed to be delivered out-of-sequence, each receiver buffer can be divided into W independent portions, each serving one block transmission session. The transfer time in this case is effectively reduced by a factor of W. Although this is an approximation, our simulation results, which are reported in the next section, indicate that it does not have a significant impact on the results in the operating regime.

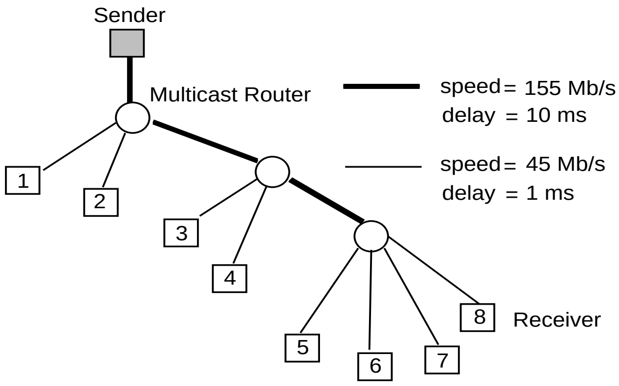

Figure 2: Multicast tree configuration

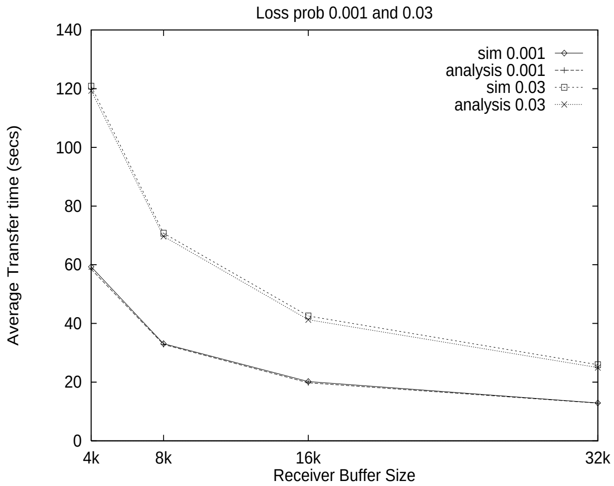

Figure 3: Effect of buffer sizes on transfer time

4 Results

In this section, we use analysis and simulations to study the performance of the error control scheme. The principal metric of performance used is the transfer time, which is determined by the average throughput of the session. We first investigate the effects of the buffer and block size on the achievable performance for a representative multicast tree. We next

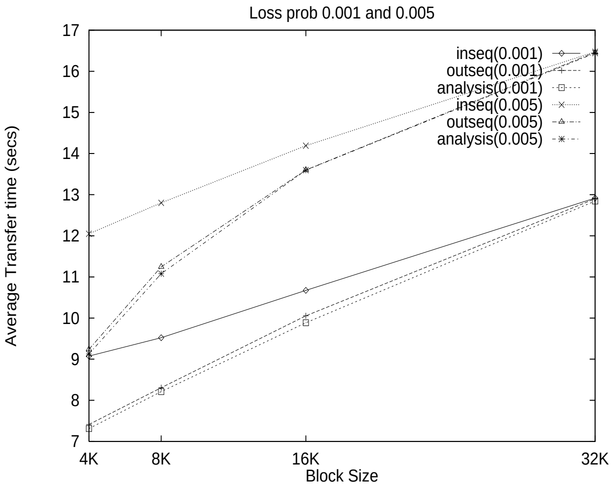

Figure 4: Effect of out of sequence delivery on transfer time

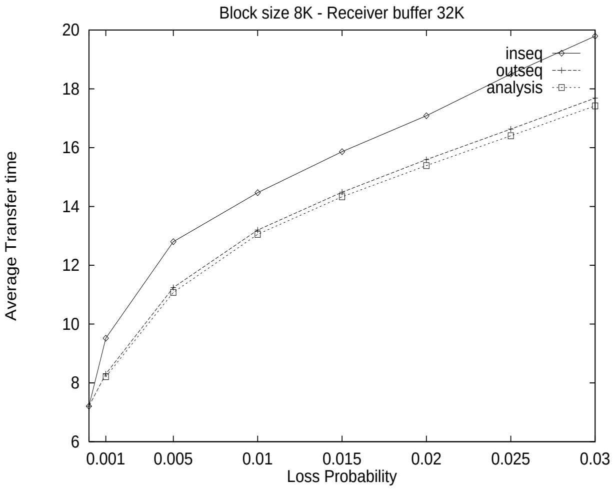

Figure 5: Effect of loss Probability on transfer time

examine the scaling properties of the system by studying how the performance deteriorates as either the loss probability or the number of receivers in the multicast group increases. We briefly discuss the implication of our results on the process of multicast tree construction.

Figure 2 shows the multicast tree topology that we use to study the effect of the buffer sizes and loss probabilities on the achievable performance. There are 8 receivers in the multicast group. The link speeds and propagation delay values are chosen to be representative of a wide area network with the average round trip delay being approximately 63ms (this includes the link delays and processing time at various switches). We choose a fixed packet size of 500 bytes. The effect of contention with other sessions is indirectly modeled by limiting the maximum sending rate of the multicast session and assuming a certain likelihood of losing packets at each switch and receiver in the multicast tree. The sender is assumed to inject packets with an inter-packet gap of 500 μsecs, corresponding to a maximum sending rate of 8 Mb/s. In all the simulations, the transfer size is assumed to be 3.2 Mbytes. We replicate simulation runs for each experiment so that the width of the 95% confidence interval for the transfer times is less than 5% of the mean value. The simulator used was a version of the MIT network simulator [13], suitably augmented to allow multicast sessions.

The performance of any retransmission mechanism is strongly dependent on the amount of receiver buffer space [10]. In the first set of experiments we examine the effects of the buffer size on the achievable throughput. The buffer sizes chosen for the experiments, 4, 8, 16 and 32 Kbytes, are representative of the values used by transport layer protocols on UNIX workstations. We initially set the block size equal to the size of the receiver buffers. Figure 3 shows the average transfer time values as obtained by analysis and simulation.

The total transfer time consists of two components: (see Equation 3) the average time taken to successfully *transmit* all the packets (which is independent of the block size) and the time spent waiting for acknowledgements. Thus it is the second component that needs to be minimized to obtain minimum transfer time. Equation 3 indicates that the wait time is proportional to the product of the number of transmission cycles($\frac{N}{\beta}$) and the expected number of rounds for the chosen block size (β). When the receiver buffer size is large[5], fewer transmission cycles are needed to multicast all the packets resulting in a smaller wait time. However, for nonzero loss probabilities the reduction in the transfer time is not linear. In other words, doubling the receiver buffer size does not reduce the wait time by half. This is because the expected number of rounds is also a function of the block size. For a bigger block size, more rounds are needed on the average for reliable multicast. The net effect, however, is that the product of the number of transmission cycles and the expected number of rounds decreases with the increasing buffer size.

For a given value of the buffer size, the choice of block size has a significant effect on performance. Figure 4 shows the average transfer time for four values of the block size 4K, 8K, 16K and 32K with a receiver buffer size of 32K. Figure 5 shows the average transfer time for different values of the loss probability. It is clear from Figures 4 and 5 that performance improves as the block size is reduced. This is because when the block size is less than the receiver buffer size, multiple blocks can be transmitted by the sender to better utilize the receiver buffer space. This is true both when blocks are delivered in-sequence or out-of-sequence. The improvement in performance with in-sequence delivery is due to two factors. First, the sender requires fewer rounds on the average to reliably multicast a small block[6] as compared to a large block. Second, the length of the transmission cycle is shorter when the block size is small.

When out-of-sequence delivery of blocks is permitted, in addition to the above two factors, a third factor results in a further improvement in throughput. When the block size is smaller than the receiver buffer size, multiple blocks can be transmitted by the sender. If there are no losses at all, the blocks will be received in the same order in which they were transmitted from the sender. However, in the presence of random losses a block may be received successfully before an earlier block. If blocks are passed up to the application out-of-sequence, the buffers held up by blocks whose transmission is complete can be released allowing the next block transmission to commence. Thus, out-of-sequence delivery results in better utilization of receiver buffer space resulting in improved performance. In general, this improvement is greater as the loss probability increases and the block size decreases.

The relationship between the loss probability and the transfer time, as shown in Figure 5 is similar to what would be observed for a point to point connection. The main difference is that the exact relation between the average transfer time and the loss probability also depends on the topology of the multicast tree and the number of receivers. These curves show that even when the individual loss probability values are very low there is a significant impact on the average transfer time. For example, the transfer time for a 3.2 Mbytes file is 7.2 secs in the absence of losses for the topology shown in Figure 2. For the same topology with a nodal loss probability as low as .001, the transfer time increases by 15% for out-of-sequence delivery and 32% for in-sequence delivery.

4.1 Scaling

Figure 5 shows that even low loss probabilities at individual nodes get magnified because of the multicast nature of the communication. Clearly it is important to study how performance is affected as the number of receivers increases. The transfer time depends on the cumulative loss probability which depends on the topology of the tree. Thus the impact of increasing the number of receivers varies depending on where they are located. To study this effect we use three topologies - a linear chain, a binary tree and a two level tree. The first topology results in the greatest amount of overlap between the different paths, while the last one has no overlap. The binary tree has an intermediate amount of overlap.

Figure 6 shows the cumulative loss probability as seen by the source for the different loss probabilities. The results show that even for moderate size groups

[5] consequently the block size is also large since we have set block size equal to the receiver buffer size.

[6] The number of rounds needed to deliver a block consisting of k packets is actually the k^{th}-order statistic of the number of trials needed to deliver a single packet. This means that a larger block requires more rounds on the average compared to a smaller block.

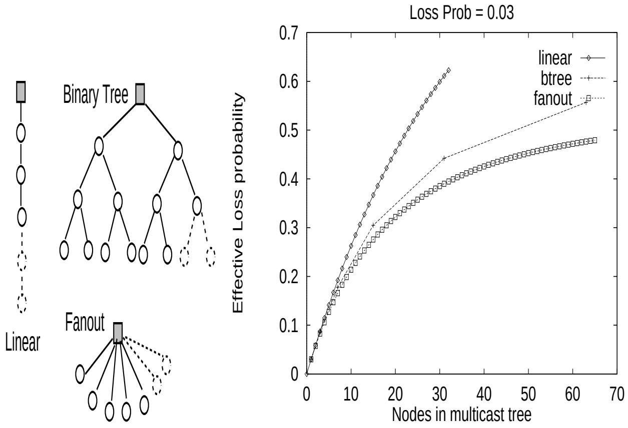

Figure 6: Effective loss probability for different topologies

of say 40-50 receivers the cumulative loss probabilities range from 30-70% for an individual loss probability value of 3%. As might be expected, the linear topology which has the greatest amount of overlap results in the highest cumulative loss probability.

In general, multicast trees are built so that they use a minimal amount of bandwidth. When receivers are added dynamically, a shortest path computation is used to attach the receiver to avoid recomputing the entire multicast tree. However, to minimize the cumulative loss probability and hence the transfer time, it is desirable to reduce the degree of overlap between paths as Figure 6 shows. In Figure 7, we show two possibilities for attaching a new receiver in an existing multicast tree when the loss probability value is 0.03. Alternative 2 represents the choice that uses minimal bandwidth. However, Alternative 1, which represents the shortest path, results in lower transfer time. In Figure 8 we show two other possibilities for attaching a new receiver in the same multicast tree. This time the choice based on minimum bandwidth does better than the one based on shortest path computation.

The dependence of the optimal transfer time on the topology is quite complex and can vary depending on the individual loss probability values. In Tables 1 and 2 we show the expected number of rounds vs the loss probability values for the choices shown in Figures 7 and 8. The tables show a very interesting phenomenon: neither alternative outperforms the other for all loss probability values. In the tables we have drawn a horizontal line to show the crossover point. As a general rule we have found that the alternative which increases the *fanout* performs better in the asymptotic case, i.e. as the loss probabilities keep increasing.

5 Conclusions

In this paper we have examined the performance implications of providing reliable multicast transport.

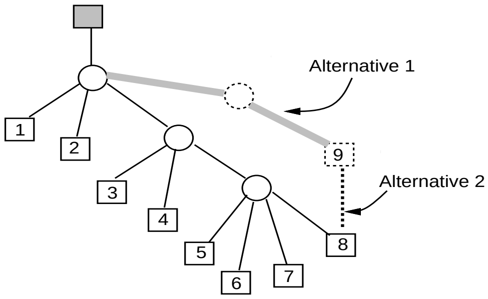

Figure 7: Alternative 1 is better

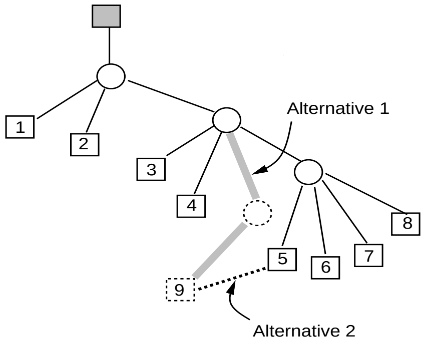

Figure 8: Alternative 2 is better

We have used a block based acknowledgement and selective retransmission protocol to evaluate the impact of the *loss rate* and the *multicast tree topology* on the average throughput seen by a session. We have developed an analytical technique to derive the cumulative loss probability seen by a source as a function of the loss rates at individual nodes of the multicast tree. This cumulative loss probability may be used to derive the average throughput seen by a session. We have shown that even for small receiver group sizes and low nodal loss probabilities, the cumulative loss probability seen by a source may be quite high. We have demonstrated that out-of-sequence delivery of packets by the transport protocol to the application layer can lead to a significant increase in the throughput. We have also investigated the scaling properties of the error control mechanism and shown that the multicast tree topology that results in minimum transfer time is not necessarily the same as the one constructed using minimal bandwidth or shortest path algorithms.

The results of this paper suggest that completely reliable transmission to a large group of receivers can only be achieved at the cost of a large increase in the time taken to transfer blocks of data vis-a-vis the transmission time in a loss free environment. The poor scaling properties of a sender controlled reliability mechanism suggest that it may be necessary to augment or modify this approach suitably. One possibility is to use a core based tree approach in which

p_n	$E[rounds(\beta)]$ for $\beta = 64$	
	Alternative 1	*Alternative 2*
0.010	2.16	2.15
0.015	2.35	2.36
0.020	2.56	2.57
0.025	2.77	2.79
0.030	2.96	2.99
0.035	3.15	3.19
0.040	3.32	3.37
0.045	3.50	3.56
0.050	3.67	3.74

Table 1: Average block rounds for Figure 7

p_n	$E[rounds(\beta)]$ for $\beta = 64$	
	Alternative1	*Alternative 2*
0.010	2.17	2.15
0.015	2.37	2.36
0.020	2.59	2.57
0.025	2.80	2.79
0.030	3.00	2.99
0.035	3.19	3.19
0.040	3.37	3.37
0.045	3.55	3.56
0.050	3.72	3.74

Table 2: Average block rounds for Figure 8

certain core nodes in the multicast tree undertake to distribute data reliably to all receivers in the sub-tree rooted at these nodes. This also reduces the magnitude of the ack-implosion problem. For image applications, another solution may be to segment informations into multiple levels of priority and provide reliable delivery for the more important information. We expect to investigate these options in future work.

The throughput that may be achieved by a session is also dependent on the congestion control mechanism that is used. Using a more aggressive congestion control mechanism may provide greater throughput but also result in greater losses. Clearly the interaction of the error control and congestion control mechanisms will determine the optimal operating point. We expect to study this interaction in future work. We would also like to evaluate the performance of NACK based protocols because of their superior scaling properties and study the degree of reliability in transmission that can be achieved with this class of protocols.

References

[1] XTP Protocol Definition Revision 3.6. Protocol Engines Incorporated, PEI 92-10, Mountain View, CA, January 1992.

[2] Mostafa H. Ammar and Li-Ran Wu. Improving the Throughput of Point-to-Multipoint ARQ Protocols Through Destination Set Splitting. In *Proceeding of IEEE INFOCOM 92*, 1992.

[3] S. Armstrong, A. Freier, and K. Marzullo. Multicast Transport Protocol. RFC 1301, Feb 1992.

[4] D. Bertsekas and R. Gallager. *Data Networks*, pages 297–333. Prentice-Hall, Inc., 1987.

[5] R. Braudes and S. Zabele. Requirements for Multicast Protocols. RFC 1458, May 1993.

[6] David R. Cheriton and Carey L. Williamson. VMTP as the Transport Layer for High-Performance Distributed Systems. *IEEE Communications Magazine*, 27(6):37–44, June 1989.

[7] J. Crowcroft and K. Paliwoda. A Multicast Transport Protocol. In *Proceeding of ACM SIGCOMM*, pages 247–256, 1988.

[8] Stephen E. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proceeding of ACM SIGCOMM*, pages 55–64, 1988.

[9] B.T. Doshi, P. K. Johri, A. N. Netravali, and K. K. Sabnani. *Retransmission Protocols and Flow Controls in High Speed Packet Networks.* Elsevier Science Publishers B.V. (North-Holland), 1991. Teletraffic and Datatraffic in a period of change, ITC-13.

[10] Jr. E. J. Weldon. An Improved Selective-Repeat ARQ Strategy . *IEEE Transactions on Communications*, 30(3):480–486, 1982.

[11] A. Erramilli and R. P. Singh. A Reliable and Efficient Mulitcast Protocol for Broadband Broadcast Networks . In *Proceeding of ACM SIGCOMM*, pages 343–352, 1988.

[12] Inder S. Gopal and Jeffrey Jaffe. Point-to-Multipoint Communication over Broadcast Links. *IEEE Transactions on Communications*, 32(9):1034–1044, 1984.

[13] Andrew Heybey. NETSIM: The Network Simulator, Sept 1990.

[14] Sheshadri Mohan, Jianguo Qian, and Nagendra L. Rao. Efficient Point-to-Point and Point-to-Multipoint Selective-Repeat ARQ Schemes with Multiple Retransmissions: A Throughput Analysis . In *Proceeding of ACM SIGCOMM*, pages 49–57, 1988.

[15] Bala Rajagopalan. Reliability and Scaling Issues in Multicast Communication. In *Proceeding of ACM SIGCOMM*, pages 188–198, August 1992.

[16] Don Towsley. An Analysis of a Point-to-Multipoint Channel Using a Go-Back-N Error Control Protocol. *IEEE Transactions on Communications*, 33(3):282–285, 1985.

[17] Philip S. Yu and Shu Lin. An Efficient Selective-Repeat ARQ Scheme for Satellite Channels and Its Throughput Analysis. *IEEE Transactions on Communications*, 29(3):353–263, 1981.