Scalable Reliable Multicast Using Multiple Multicast Groups

Sneha K. Kasera, Jim Kurose and Don Towsley

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{kasera,kurose,towsley}@cs.umass.edu

Abstract

We examine an approach for providing reliable, scalable multicast communication, involving the use of multiple multicast groups for reducing receiver processing costs in a multicast session. In this approach a single multicast group is used for the original transmission of packets. Retransmissions of packets are done to separate multicast groups, which receivers dynamically join or leave. We first show that protocols using an infinite number of multicast groups incur much less processing overhead at the receivers compared to protocols that use only a single multicast group. This is due to the fact that receivers do not receive retransmissions of packets they have already received correctly. Next, we derive the number of unwanted redundant packets at a receiver due to using only a finite number of multicast groups, for a specific negative acknowledgment (NAK)–based protocol. We then explore the minimum number of multicast groups required to keep the cost of processing unwanted packets to a sufficiently low value (i.e., to achieve most of the benefit of using an infinite number of multicast groups). For an application consisting of a single sender transmitting reliably to many receivers we find that only a small number of multicast groups are required for a wide range of system parameters. In the case of an application where all participants simultaneously act as both senders and receivers a moderate number of multicast groups is needed. Finally, we present a local filtering scheme for minimizing join/leave signaling when multiple multicast groups are used.

1 Introduction

Many applications such as shared whiteboards, distributed interactive simulation and distributed computing, require reliable multicast, where sender(s) transmit data to a group of receivers in a reliable manner. Designing a reliable multicast protocol for an application that can have several thousand receivers, (e.g., wb [8] which has been used with more than one thousand users) is a challenging task.

In this paper we examine an approach for providing reliable, scalable multicast communication. This approach (first discussed in [3] and [5]) involves the use of multiple multicast groups for reducing receiver processing costs in a multicast session. To illustrate the problem in using a single multicast group, consider an ARQ–based reliable multicast scenario with a single multicast group. Since all packet transmissions and retransmissions are done using the single multicast group, each receiver receives all the retransmissions of a packet even after correctly receiving the packet. This can impose unnecessary receiver processing overhead especially as the number of receivers increase. The approach we examine in this paper allows one to overcome this *problem of receiver processing unwanted redundant packets in a multicast scenario.* It can can be informally described as follows. Following the Internet model of IP multicast, we will use the terms multicast groups, channels and addresses interchangeably. Let us consider a system with a sender and R receivers such that data is transmitted reliably from the sender to the receivers using $G + 1$ channels. One of the channels is used for the original transmission of packets from the sender. The remaining G channels are mapped to ranges of sequence numbers such that a lost packet is retransmitted on the channel corresponding to the range to which it belongs. A receiver, after detecting a packet loss, "joins" the appropriate channel for that packet's retransmission. Once the lost packet is received, the receiver "leaves" the appropriate channel. Past work on reliable multicast assumes $G = 0$ and in such a situation all transmissions and retransmissions are sent on the same multicast channel.

In our work we first show that protocols using an infinite number of multicast groups incur much less processing overhead at the receivers compared to protocols that use only a single multicast group. Next, through simple analyses, we derive the number of unwanted packets at a receiver due to using only a finite number of multicast groups, for a specific negative acknowledgment (NAK)–based protocol. We then explore the minimum number of multicast groups required to keep the cost of processing unwanted packets to a sufficiently low value (i.e., to achieve most of the benefit of using an infinite number of multicast groups). For an application consisting of a single sender transmitting reliably to many receivers (referred to as a *one-many* application [17]) we found that only a small number of multicast groups are required for a wide range of system parameters. In the case of an application where all participants simultaneously act as both senders and receivers (referred to as a *many-many* application [17]) a moderate number of multicast groups is needed.

One potential concern with using multiple multicast groups is the processing of "joins" and "leaves," as receivers dynamically add and delete themselves from these groups. In order to avoid shifting the burden from the receivers to the underlying network and at the same

time maintaining the reduced processing benefits, we propose a simple receiver filtering mechanism that results in the same amount of network traffic as in the case of a single multicast group, but does not require joins and leaves to be processed inside the network once the multicast session begins.

The remainder of the paper is organized as follows. In the next section we examine the existing work on the subject. In section 3 we present several protocols that use multiple multicast groups. In section 4, we analyze the processing cost performance of these protocols. In section 5, we present numerical results to show that use of multiple multicast groups leads to reduction of processing overhead at the receivers. In section 6 we derive the number of multicast groups required to keep the processing overhead due to unwanted packets at a receiver to a sufficiently low value for a specific NAK based protocol. In section 7 we present our filtering scheme for handling multiple multicast groups without incurring any significant overhead in the underlying network. Our conclusions and suggestions for future work are contained in Section 8.

2 Related Work

Much of the existing work on reliable multicast concerns the design and implementation of ACK–based and/or NAK–based error recovery. In [16] and [17] the authors have provided a quantitative demonstration of the superiority of receiver-initiated NAK–based approaches over sender-initiated ACK–based approaches. Yavatkar [18] proposes error recovery that is driven by receivers who use a combination of restricted negative acknowledgment with NAK suppression and periodic positive ACKs. Grossglauser [10] has described a rigorous approach to the NAK implosion problem. Floyd *et al.* [8] have proposed a NAK based reliable multicast for a shared whiteboard application. They have also proposed the use of local recovery. All of the above work assumes that there is only a single multicast group, to which all the receivers belong.

Cheung *et al.* [4] and later McCanne [14] have proposed the use of multiple multicast groups for flow and congestion control, but not for error recovery. In [9] the possibility of using separate multicast groups for defining "local groups" for local recovery has been suggested. Cheriton [3] and Crowcroft [5] suggested the use of multiple multicast groups, for error recovery in reliable multicast, in a discussion on the end-end mailing list. Holbrook [11] proposes the use of separate retransmission channels for error recovery as future work.

Even earlier, Ammar and Wu [1] proposed the idea of destination set splitting for improving the throughput of some specific positive acknowledgment based point-to-multipoint protocols. They suggested that receivers could be divided into groups based on their capabilities and the sender carries out as many simultaneous independent conversations as the number of groups. Our work, inspired by Cheriton and Crowcroft's suggestion, differs from Ammar's work in several ways. We point out three significant differences. First, we do not group receivers based on their capabilities. Rather we group packets such that retransmissions of packets belonging to each group is done on a separate multicast channel. Second, we have considered generic NAK based protocols instead of specific ACK based

protocols. Third, in addition to point-to-multipoint scenario, we have also considered the multipoint-to-multipoint scenario.

3 Protocols and System Model

We now present three NAK–based protocols for using multiple multicast channels for reliable multicast from a sender to several receivers . Based on arguments presented in [8] and [16, 17] it is clear that for many applications receiver–based reliability is a better scheme (in terms of performance) for reliable multicast then sender–based reliability. Hence in our work we focus only on receiver–based recovery, or negative acknowledgment (NAK)–based schemes. The section ends with a description of the applications and the network model.

3.1 Protocol Description

The protocols we describe below are modified versions of the generic protocols, N1 and N2, proposed in [16, 17] . In [16, 17], the authors have considered only one multicast group for both transmissions and retransmissions. The reliable multicast protocols we will consider will be denoted P1, P2 and P3. We initially make the assumption that we have an infinite number of multicast groups at our disposal so that each packet can be recovered on a separate channel, i.e., $G = \infty$. Later, we will see that we can easily *reuse* a small number of multicast groups to achieve almost the same affect that can be obtained by using large number of multicast groups. The discussion below is based on an IP multicast like network scenario, where the sending of a packet to a multicast group address causes that packet to be sent (potentially with some loss) to the members of that group.

The protocol P1 exhibits the following behavior

- the sender sends all original transmissions on a multicast address A_{org}.

- when required, the sender retransmits a packet with sequence number i on multicast address A_i where $i = 0, 1, 2, ...$.

- whenever a receiver detects a lost packet i, it transmits a NAK to the sender over a point-to-point channel and subscribes to the multicast address A_i and starts a timer.

- the expiration of a timer without prior reception of the corresponding packet serves as the detection of a lost NAK packet, a NAK is retransmitted for the associated packet and a timer again started.

- on receiving packet i on A_i a receiver unsubscribes to A_i.

Protocol N1 in [16, 17] is equivalent to P1 when i takes only the value 0. The protocol P2 exhibits the following behavior

- the sender sends all the original transmissions on a multicast address A_{org},

- the sender retransmits packet i on multicast address A_i where $i = 0, 1, 2, ...$,

- whenever a receiver detects a lost packet(say i), it waits for a random period of time and then broadcasts a NAK on the address A_i, and starts a timer,

- upon receipt of a NAK for a packet for which a receiver has not received, but for which it initiated the random delay prior to sending a NAK, the receiver sets a timer and behaves as if it had sent that NAK,

- the expiration of a timer without prior reception of the corresponding packet serves as the detection of a lost NAK packet.

Protocol N2 in [16, 17] is equivalent to P2 when i takes only the value 0. Protocol P3 exhibits the same behavior as P2 except that a receiver sends a NAK for packet i on the original multicast address A_{org} instead of A_i as in P2. The important similarity between P2 and P3, which distinguishes them from P1, is that both have the capability to suppress NAKs [8] to the sender. They attempt to ensure that at most one NAK is sent out to the sender per packet by delaying the generation of the NAKs and multicasting them to all the participating receivers. The suppression of NAKs to the sender does not come for free, however. The price is paid in terms of extra NAK processing at the receivers as the NAKs are now multicast instead of being unicast to the sender. P2 reduces the NAK processing cost at the receivers by sending NAKs for packet i on A_i. Only those receivers, that have not received packet i, subscribe to A_i. Hence NAKs are processed a few receivers. In comparison, in P3, NAKs are retransmitted on A_{org} and are received by all the receivers that have subscribed to A_{org}. It should be noted that the sender has to be a member of all the retransmission groups in P2 and only a member of the group corresponding to A_{org} in P3. A sender does not have to be a member of any retransmission group in P1.

Before quantitatively evaluating the performance of P1, P2 and P3 , let us first qualitatively examine their behavior. Recall that in P2, NAKs are received *by only* those receivers that themselves have not received the packet i. This means that a lost packet can only be recovered from the sender and *not* from a *local* receiver. Additional mechanisms would have to be provided for local recovery. In contrast, in P3, NAKs are received by all the receivers participating in the multicast session. Some of these receivers might have received packet i correctly and can then retransmit the NAKed packet locally. Thus P3 has a provision for local recovery built into itself. Another shortcoming of P2 is that its performance is sensitive to the latency associated with detecting packet loss. If two receivers incur very different latencies in detecting the loss of the same packet, it is possible for one to return a NAK prior to the other joining the appropriate group. Consequently the second receiver will miss that NAK and may transmit its own redundant NAK. For P3, since all NAKs are sent to A_{org}, and since all receivers belong to A_{org}, such a situation does not arise.

3.1.1 System Model

When examining the performance of P1, P2 and P3, we will study two different system models, corresponding roughly to two broad classes of applications that use reliable multicast. In the first model, corresponding to the *one-many* application (e.g. telelecturing), we assume that one sender transmits a continuous stream of packets to R identical receivers. In the

second model, corresponding to the *many-many* application (e.g. distributed interactive simulations [12]), we assume that there are $R+1$ identical nodes in the system. All nodes can function as both a sender and receiver. In this model we assume that a node is a sender with a probability $1/(R+1)$ and a receiver with probability $R/(R+1)$ [17]. That is, for each packet there is a single sender and each node is equally likely to be the sender. For both models we assume that all loss events at all receivers for all transmissions are mutually independent and that the probability of packet loss, p, is independent of receiver. We further assume that NAKs are never lost.

As noted earlier, we will begin our analysis below assuming an infinite number of available multicast channels and then consider the more realistic case of a limited number of channels.

4 Processing Cost Analysis

For the purpose of understanding the improvement in processing costs of protocols P1, P2 and P3 over protocols N1 and N2, we now analyze processing costs for both the *one-many* and *many-many* models for protocols P1, P2 and P3. We start with the *one-many* model and later use these results to obtain processing costs for the *many-many* model.

The receive processing cost(time) is determined by computing the processing involved in *correctly* receiving a randomly chosen packet. This includes the time required to receive those copies of this packet (i.e., the original copy plus any retransmissions) that arrive at the receiver, the time required to send/receive any NAKs associated with this packet and the time needed to handle any timer interrupts associated with this packet. The send processing cost is determined by the processing involved in correctly transmitting a packet to all the receivers. This includes the cost required to process the original transmission of the packet, process any received NAKs, and process retransmissions that are sent out in response to these NAKs.

We now derive expressions for receiver processing requirements for the protocols P1, P2 and P3. Table 1 describes the notation used in the analysis. Most of the notation has been reintroduced from [17] for the sake of consistency. We assume that the processing times have general distributions, that they are independent of each other, and that $\{X_p\}$, $\{X_n\}$, $\{Y_p\}$, $\{Y_t\}$, $\{Y_n\}$, $\{Y'_n\}$, $\{Y_j\}$ and $\{Y_l\}$ are each identically distributed sequences of random variables.

Following an approach similar to the one in [17], the mean per packet processing time for a randomly chosen packet at a receiver for the P1 protocol can be expressed as,

$$E[Y^{P1}] = E[Y_p] + (E[Y_j] + E[Y_l])p + E[(M_r - 1)^+]E[Y_n] + E[(M_r - 2)^+]E[Y_t] \qquad (1)$$

where $(x)^+ = \max\{0, x\}$. The first term corresponds to the processing required to correctly receive a packet. A receiver will only receive one copy of the packet (either on A_{org}, or on A_i for packet i). The next term represents the processing required for joining and leaving a multicast group. Note that the join and leave processing times have been multiplied by the loss probability because this cost is incurred only when a packet is lost and a NAK is

6

X_p	–	the time to process the transmission of a packet
X_n	–	the time to process a NAK
Y_p	–	the time to process a newly received packet
Y_t	–	the time to process a timeout
Y_n	–	the time to process and transmit a NAK
Y_n'	–	the time to receive and process a NAK at a receiver
Y_j	–	the time to process a join
Y_l	–	the time to process a leave
p	–	the loss probability at a receiver
R	–	the total number of receivers
M_r	–	the number of transmissions necessary for receiver r to successfully receive a packet
M	–	the number of transmissions for all receivers to receive the packet correctly; $M = max_r\{M_r\}$
X^ω, Y^ω	–	the send and receive per packet processing time in protocol $\omega \; \epsilon \; \{$P1, P2, P3$\}$

Table 1: Notation

transmitted. Although several NAKs might be sent from a receiver to recover a lost packet, a receiver needs to join and leave the corresponding multicast group only at most once per packet. The third term represents the processing required to prepare and return NAKs. The last term corresponds to the processing of the timer when it expires. Since M_r, the number of times a message must be sent before a given receiver receives the packet is independent of the number of multicast groups (and indeed is independent of whether an ACK or NAK protocol is used), we can use the results in [17] directly:

$$E[M_r] \;=\; 1/(1-p), \tag{2}$$
$$E[(M_r - 1)^+] \;=\; p/(1-p), \tag{3}$$
$$E[(M_r - 2)^+] \;=\; p^2/(1-p). \tag{4}$$

Substitution of (3) and (4) into (1) yields

$$E[Y^{P1}] = E[Y_p] + (E[Y_j] + E[Y_l])p + pE[Y_n]/(1-p) + p^2E[Y_t]/(1-p). \tag{5}$$

For P2 the mean per packet processing time can be expressed as,

$$
\begin{aligned}
E[Y^{P2}] \;=\; & E[Y_p] + (E[Y_j] + E[Y_l])p \\
& + (E[(M_r] - 1)(E[Y_n]/R + (R-1)E[Y_{n'}]/R) \\
& + E[(M_r - 2)^+]E[Y_t]
\end{aligned} \tag{6}
$$

Here, the first two terms are the same as in case of P1. The third term is comprised of two terms. The first one results from generation of NAKs and the second results from the

7

reception of NAKs at the receiver. A receiver generates a NAK with a probability $1/R$ and receives a NAK with a probability of $(R-1)/R$. The last term is the same as that for P1. Substitution of (2) and (4) into (6) yields

$$E[Y^{P2}] = E[Y_p]+(E[Y_j]+E[Y_l])p+p/(1-p)(E[Y_n]/R+(R-1)E[Y_{n'}]/R)+p^2E[Y_t]/(1-p) \quad (7)$$

Using similar arguments, the mean per packet processing time at a receiver for the P3 protocol can be expressed as:

$$\begin{aligned} E[Y^{P3}] &= E[Y_p] + (E[Y_j] + E[Y_l])p + (E[M] - 1)(E[Y_n]/R + (R-1)E[Y_{n'}]/R) \\ &\quad + p^2 E[Y_t]/(1-p) \end{aligned} \quad (8)$$

The only difference between the expressions for P2 and P3 is the replacement of $E[M_r]$ in P2 by $E[M]$ in P3. Recall that in P2 a receiver sends a NAK for a packet i to address A_i. A receiver r, that is trying to recover packet i, will subscribe to A_i for only the time until it receives i. Since all NAKs are sent to address A_i, the number of NAKs it will receive during this time is M_r. On the other hand, in protocol P3, a NAK is sent to the original transmission address A_{org} and all the NAKs sent for packet i are received by all receivers. $E[M]$ can be expressed in terms of R and p as follows.

$$\begin{aligned} E[M] &= \sum_{m=1}^{\infty} mP[M=m] = \sum_{m=1}^{\infty} P(M \geq m), \\ &= 1 + \sum_{m=1}^{\infty}(1 - P(M \leq m)), \\ &= 1 + \sum_{m=1}^{\infty}(1 - (1 - p^m)^R) \end{aligned}$$

The sender processing costs of P1 and P2, P3 are same as that for N1 and N2 respectively. This is because the use of multiple multicast groups only reduces processing of redundant packets at the receivers. There is no change in the number of NAKs received by the sender and hence there is no change in the number of packets sent out by the sender. There is no per–packet join/leave processing cost at the sender. In P1 and P3, the sender does not have to be a member of any retransmission group because it does not receive anything on the retransmission groups. In P1, NAKs are received point-to-point and in P3, NAKs are received on the original transmission multicast address and there is no join/leave cost associated with a multicast group if one only sends to that group[6]. In P2 the sender has to be a member of all the retransmission groups before it starts transmitting packets because it receives NAKs on the retransmission groups. It remains a member until the multicast session ends. Hence it does not incur any join/leave processing costs during the session. We can thus use the expressions of sender processing costs from [17]. The mean *sender* processing time needed to successfully transmit a packet to all the receivers, for protocols P1, P2 and

P3 is given by the following expressions.

$$E[X^{P1}] = E[X^{N1}] = E[M]E[X_p] + RpE[X_n]/(1-p) \tag{9}$$
$$E[X^{P2}] = E[X^{P3}] = E[X^{N2}] = E[M]E[X_p] + (E[M]-1)E[X_n]. \tag{10}$$

The overall protocol throughput, for the *one-many* case, for the family of protocols P1, P2 and P3 is given by the following definition.

$$\Lambda_o^\omega = \min\{\Lambda_s^\omega, \Lambda_r^\omega\}$$

where $\omega\epsilon\{P1, P2, P3\}$, $\Lambda_s^\omega = 1/E[X^\omega]$ and $\Lambda_r^\omega = 1/E[Y^\omega]$. Λ_s^ω represents the rate at which a sender could successfully transmit packets to all receivers. Λ_r^ω is a measure of the maximum packet processing rate at a receiver. The overall protocol throughput is the minimum of the two rates because the overall performance of the protocol would be dictated by the slower of the send and receive processing rates.

Recall that under the *many-many* scenario, each of the $R+1$ end system nodes are equally likely to be the sender of the randomly chosen packet. Hence the mean packet processing time is $E[X^\omega]/(R+1) + R\,E[Y^\omega]/(R+1)$ and the overall protocol throughput, which is the inverse of the mean packet processing time, is expressed as

$$\Lambda_m^\omega = \frac{R+1}{E[X^\omega] + RE[Y^\omega]} \tag{11}$$

where $\omega\epsilon\{P1, P2, P3\}$.

5   Numerical Results

We now examine the relative performance of protocols P1, P2 and P3 and N1 and N2. In order to do so we need to know the processing times associated with sending/receiving a data packet and a NAK packet, as well as their interrupt processing times. In order to measure these values, we instrumented a Linux kernel version 1.2.13 on a 133 MHz Pentium PC. As we had complete control over the processes running on PC we ensured that the PC had the minimum possible load and performed all of our measurements inside the Linux kernel. We considered two packet sizes, 1024 bytes for data packets and 32 bytes for NAK packets. The results of our measurements are summarized in the Table 2. Each processing time, in microseconds, was measured 1000 times and averaged. The timer processing time includes time to set, execute and delete the timer. To determine the join and leave processing time, we sequentially performed 20 join operations of distinct IP multicast groups followed by 20 leaves operation. The join and leave operations were "local" meaning that no IGMP reports were sent out on these operations. The concept of "local" join and leave is discussed later in the section on local filtering.

It is worth noting that several measurement studies have been reported in the literature, notably [13] and [15]. However, neither of these included measurements of join and leave processing times thus necessitating the series of measurements described above.

9

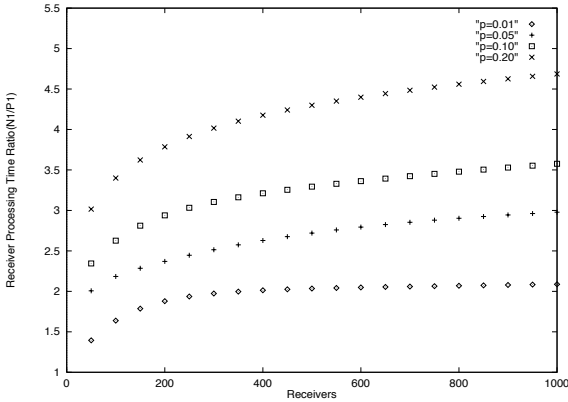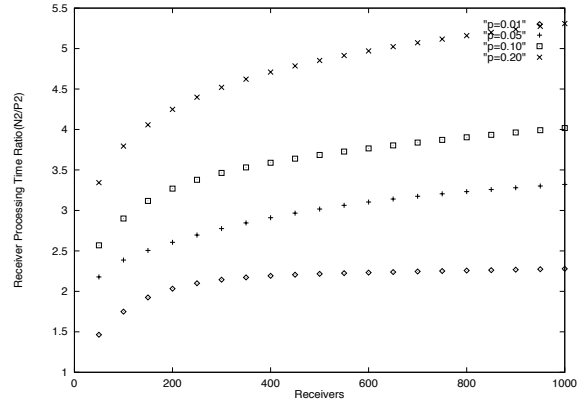| Operation	Mean Processing Time	Standard Deviation
Send-Data	502	13.8
Send-NAK	87	5.8
Recv-Data	487	12.6
Recv-NAK	86	5.1
Join	75	6.6
Leave	74	6.8
Timer	32	1.5

Table 2: Processing Time(in microseconds)

Figure 1: N1 vs P1

Figure 2: N2 vs P2

We can now compute the mean send and receive processing costs(times) for the protocols P1, P2 and P3 using equations (5)–(10), for several values of R and p. In the following, we will directly use the expressions derived in [17] to do the same for protocols N1 and N2.

Figures 1-3 show how the ratio of receiver processing costs obtained under the N and P protocols varies with R and p. It can be seen that the family of P protocols always perform better in all the three graphs. This is because the P protocols, by using multiple multicast groups, eliminate the reception of unwanted and redundant data packets at the receivers and hence receiver processing costs decrease. Although there is a slight increase in processing costs due to the processing of extra joins and leaves at the receivers, this increase is much less than the benefit obtained by reducing the processing of unwanted packets. Further, the benefit increases as the loss probability, p and number of receivers, R increases. There are two factors contributing to this behavior. The first factor is that the cost of processing a data packet is much higher than the cost of processing a join or a leave operation; Table 2 shows that the cost of processing a data packet at a receiver is almost 6.5 times the cost of processing a join or leave. The other factor is that only one join and one leave are required to recover a lost packet at a receiver. Whereas in the case of protocols N1 and N2 the number of unwanted packets per packet recovery, $(E[M] - 1)(1 - p)$, at a receiver is greater than one even for small loss probabilities and small number of receivers. This number increases with
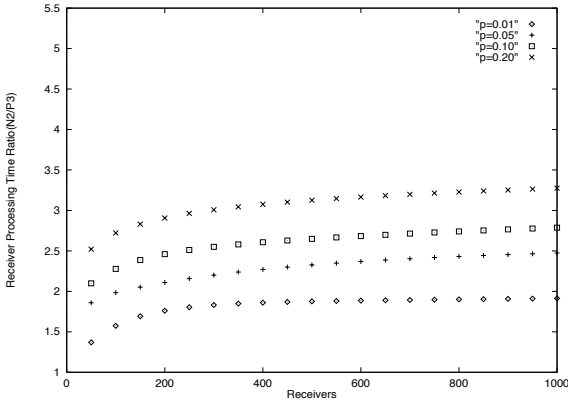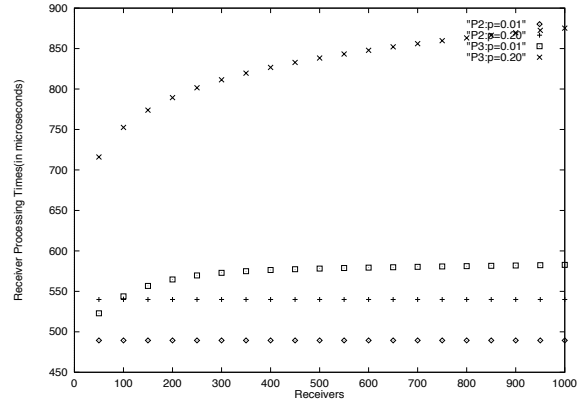
10

Figure 3: N2 vs P3

Figure 4: P2 vs P3

p and R.

Figures 1 and 2 show the receiver processing improvement of P1 over N1 and P2 over N2 respectively. Observe that the relative performance of P2 over N2 is better than that of P1 over N1. This is because the receiver processing time of N2 is higher than that of N1 [17]. On the other hand the receiver processing times of P1 and P2 are almost same. In fact, from equations (5) and (7) it can be seen that these are exactly same if $E[Y_n] = E[Y_n']$, i.e., if the processing time to send a NAK is the same as the processing time to receive a NAK. From Table 2 we see that $E[Y_n] = 87$ microseconds and $E[Y_n'] = 86$ microseconds. Hence the receiver processing times of P1 and P2 are same for any practical purposes.

Figure 3 shows the receiver processing performance of P3 over N2. Although P3 does not perform as well as P2 it still substantially outperforms N2. Figure 4 shows the receiver processing times of P2 and P3. We see that P3 incurs higher receiver processing costs than P2 (and therefore P1), as expected. Recall from section 3 that P3 has to perform extra NAK processing in comparison to P1 and P2.

So far we have concentrated on the receiver processing costs. We now examine the overall protocol throughput of the protocols P1, P2 and P3. As noted before, the use of multiple multicast groups does not have any impact on the sender processing costs. Hence the send processing rate of protocols P1 is the same as that of N1 and the send processing rate of P2 and P3 is the same as that of N2. If the send processing rate is the bottleneck then the overall protocol throughput, for the *one-many scenario*, defined as the minimum of the send and receive processing rates, does not change even with the improvement in receiver processing cost(rate). In [17] it has been shown that the send processing rate is indeed the bottleneck for protocols N1 and N2. This means that the send processing rate is also the bottleneck for protocols P1, P2 and P3. Thus, while we see a marked performance advantage in receiver processing rates, the overall protocol throughput for the *one-many* model does not improve with the use of multiple multicast groups. However, this should not minimize the importance of reducing receiver processing cost. Most receivers are likely to be systems with multi-tasking operating environments so that the reduction of unnecessary processing
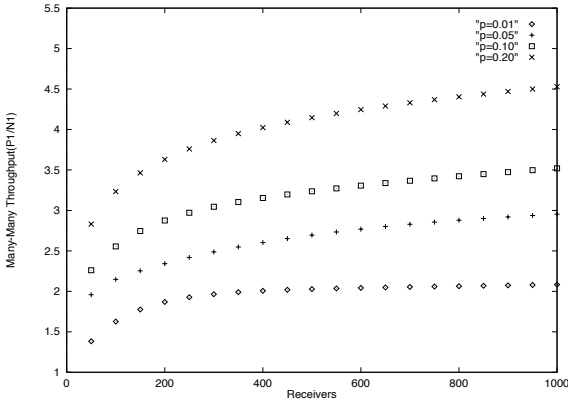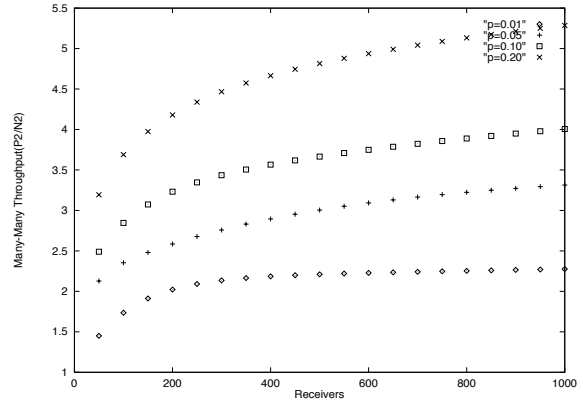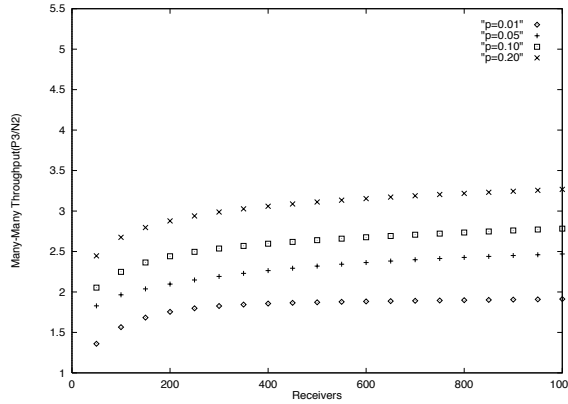
Figure 5: P1 vs N1

Figure 6: P2 vs N2

Figure 7: P3 vs N2

is highly desirable. Also, one should not let a receiver pay by processing redundant packets requested by other receivers.

We next examine the *many-many* model, using equation (11) to compute the overall protocol throughput for protocols P1, P2 and P3 and N1 and N2. Figures 5–7 show how the *many-many* throughput ratio of the P and the N protocols varies with the number of receivers for different loss probabilities. We see in Figure 5 that P1 outperforms N1 and in Figures 6 and 7 that P2 and P3 outperform N2. This follows from the fact that a participant in a *many-many* application is much more likely (with probability $R/(R+1)$) to perform receive processing on a packet than send processing. Consequently the throughput ratios exhibited in Figures 5, 6, and 7 exhibit behavior nearly identical to the mean per packet receive processing ratios for those protocols in Figures 1, 2, and 3. Among the P protocols, the overall protocol throughput performance of P2 is the best. This is because through suppression of NAKs, the sender processing cost of P2 is less than that of P1 and the same as that of P2. Also, as we have noted before, the receiver processing cost of P2 is also less than that of P1 and P3.

12

In summary, we see a significant reduction of receiver processing costs by using multiple multicast groups. For the *many-many* application, we also see a substantial improvement in overall protocol throughput.

6 Number of Retransmission Multicast Groups

In the previous section we made the assumption that the number of available multicast groups was infinite. This is unrealistic and even if a very large number of multicast groups was available, practical considerations such as the size of routing tables within the network would argue in favor of a smaller number of multicast groups. Recall that the main purpose for choosing a multicast group per packet was to avoid receiving any unwanted redundant packets. In this section we demonstrate through analysis that with only a finite and small number of multicast groups we can keep the overhead of processing redundant packets extremely low – approaching that achievable with an infinite number of groups.

Our system model is the same as the *one-many* model in the previous section and we consider protocol P1. The retransmission of packet i is now done on the multicast address $A_{i \bmod G}$ where G is the number of retransmission multicast groups. That is, instead of subscribing to A_i, as in the previous section, a receiver trying to recover packet i subscribes to $A_{i \bmod G}$. Unlike the case of $G = \infty$, retransmission of a lost packet on a multicast group can now be interspersed with the retransmissions of another packet corresponding to the same group. This interspersion depends upon the rate of retransmission of lost packets with respect to the rate of transmission of *new* (first time) packets. For this reason, we must model the manner in which retransmissions are sent in relation to the new packets. We consider that the sender multicasts new packets periodically with a fixed time interval Δ, and retransmits packets periodically with a fixed interval Δ' if there is a pending NAK for that packet (equivalently, that the receivers retransmit NAKs with a time interval of Δ'). We define $\Delta' = \max_r \tau_r$, $r = 1, 2, 3, ..., R$ where τ_r is a measure of the round trip time from the sender to the receiver r. We will observe that Δ/Δ' is a key parameter in our protocol performance.

6.1 Analysis

In this section our goal is to determine the number of unwanted redundant packets received by a receiver due to the use of only a finite number(G) of multicast groups into protocol P1.

In order to illustrate the considerations that will impact performance, let us consider an infinite stream of packets. Let us pick one packet randomly and label it 0. Suppose that receiver r_1 does not receive the original transmission of packet 0. On detecting that it has not received this packet, it joins retransmission group corresponding to address $A_{0 \bmod G} = A_0$. It sends a NAK to the sender and sets its NAK retransmission timer to Δ'. The sender retransmits packet 0 on A_0. If this packet is lost again then receiver r_1's NAK retransmission timer expires and it retransmits the NAK. During the time until packet 0 is correctly received at r_1 other receivers may using the retransmission group A_0 to recover
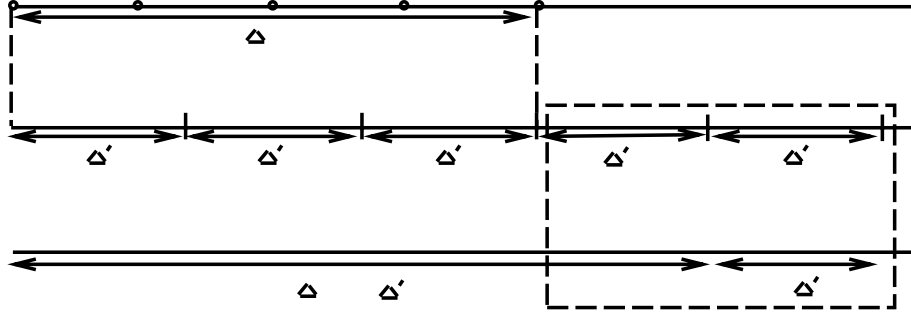
Figure 8: Sender timeline

packets $\ldots, -3G, -2G, -G, G, 2G, 3G, \ldots$. If r_1 has already received these packets, then any retransmissions of these packets are received, but are unwanted for r_1.

We wish to calculate the number of these unwanted packets at r_1. For this purpose we need to consider the overlap between the retransmissions of packets $\ldots, -3G, -2G, -G, G, 2G, 3G, \ldots$ and retransmissions of packet 0 during the period of time when r_1 is using A_0 to receive packet 0. Figure 8 shows the overlap between retransmissions of packets $-G$ and 0. For simplicity of Figure 8 we have shown $G\Delta$ to be an integral multiple of Δ', although this is not a requirement in our analysis. We introduce the random variable U to denote the number of unwanted packets at r_1 while it is attempting to receive packet 0, given that it requires at least one retransmission. Its expectation, $E[U]$, can be expressed as,

$$E[U] = (1-p)(\sum_{j=1}^{\infty}(E[Z(-jG)] + E[Z(jG)]) - \sum_{j=1}^{\infty}(E[Z'(-jG)] + E[Z'(jG)])) \qquad (12)$$

where random variables $Z(k)$, $k = \pm 1, \pm 2, \ldots$ denote the number of retransmissions of packet k due to all the R receivers that overlap with the retransmissions of packet 0 to r_1 and random variables $Z'(k)$, $k = \pm 1, \pm 2, \ldots$ denote the number of retransmissions of packet k due to r_1 that overlap with the retransmissions of packet 0 to r_1. In the remainder of this section we derive expressions for $E[Z(k)]$ and $E[Z'(k)]$, $k = \pm 1, \pm 2, \ldots$. The notation used in the analysis is described in Table 3.

For $k = -1, -2, -3, \ldots$, i.e., for packets transmitted before packet 0, $Z(k)$ is determined by considering the minimum of the number of retransmissions of packet 0 and any retransmissions of packet k, over all the receivers, after r_1 starts using A_0 for recovering packet 0. The quantity $\lfloor -k\Delta/\Delta' \rfloor$ determines the number of possible retransmissions of k before r_1 starts using A_0 to recover 0. Therefore,

$$Z(k) = \min(N, \max(0, \max_{1 \le r \le R}(N_r - \lfloor \frac{-k\Delta}{\Delta'} \rfloor))). \qquad (13)$$

For $k = 1, 2, 3, \ldots$, i.e., for packets transmitted after packet 0, $Z(k)$ is determined by considering the maximum number of retransmissions of packet k, over all receivers, until r_1 finishes

$Z(k)$	–	the number of retransmissions of packet k, due to R receivers, that overlap with the retransmissions of packet 0 to receiver r_1
$Z'(k)$	–	the number of retransmissions of packet k, due to r_1, that overlap with the retransmissions of packet 0 to r_1
N	–	number of NAKs generated by receiver r_1 for packet 0
N_r	–	number of NAKs generated by receiver r for any packet
U	–	number of unwanted packets received at a receiver per packet recovery
p	–	the loss probability at a receiver
G	–	the number of retransmission multicast groups
Δ	–	the time interval between successive transmissions from the sender
Δ'	–	the time interval between successive retransmissions from the sender
R	–	the total number of receivers

<div align="center">Table 3: Notation</div>

recovering packet 0.

$$Z(k) = \min(\max(0, N - \lfloor \frac{k\Delta}{\Delta'} \rfloor), \max_{1 \le r \le R} N_r). \tag{14}$$

In equations (13) and (14), $P(N = n) = (1 - p)p^{n-1}$ for $n \ge 1$ and $P(N_r = n) = (1 - p)p^n$ for $n \ge 0$. The difference in $P(N = n)$ and $P(N_r = n)$ arises from the fact that N can take only values greater than or equal to 1 because we start with the assumption that r_1 loses the original transmission of packet 0. On the other hand, N_r can be zero when receiver r receives the original transmission of packet k and does not need a retransmission of k. These two probabilities can be used to derive the following two relations.

$$\begin{aligned} P(N \le n) &= 1 - p^n, & n \ge 1 \\ P(N_r \le n) &= 1 - p^{(n+1)}, & n \ge 0 \end{aligned}$$

Let Z_1 and Z_2 be two independent random variables. We have the following useful relations.

$$\begin{aligned} P(\min(Z_1, Z_2) \le z) &= 1 - (1 - P(Z_1 \le z))(1 - P(Z_2 \le z)) \\ P(\max(Z_1, Z_2) \le z) &= P(Z_1 \le z)P(Z_2 \le z) \end{aligned}$$

Based on these equations it is easy to derive

$$\begin{aligned} P(Z(k) \le z) &= 1 - p^z(1 - (1 - p^{z + \lfloor -k\Delta/\Delta' \rfloor + 1})^R), k = -1, -2, \ldots \\ E[Z(k)] &= 1 - (1 - p^{\lfloor -k\Delta/\Delta' \rfloor + 1})^R \\ &\quad + \sum_{z=1}^{\infty} p^z(1 - (1 - p^{z + \lfloor -k\Delta/\Delta' \rfloor + 1})^R), \quad k = -1, -2, \ldots \tag{15} \end{aligned}$$

Using a similar approach we can obtain the following expression for $E[Z(k)]$ for $k = 1, 2, \ldots$.

$$E[Z(k)] = p^{\lfloor k\Delta/\Delta' \rfloor}(1 - (1 - p)^R) + \sum_{z=1}^{\infty} p^{z + \lfloor k\Delta/\Delta' \rfloor}(1 - (1 - p^{z+1})^R), \quad k = 1, 2, \ldots \tag{16}$$
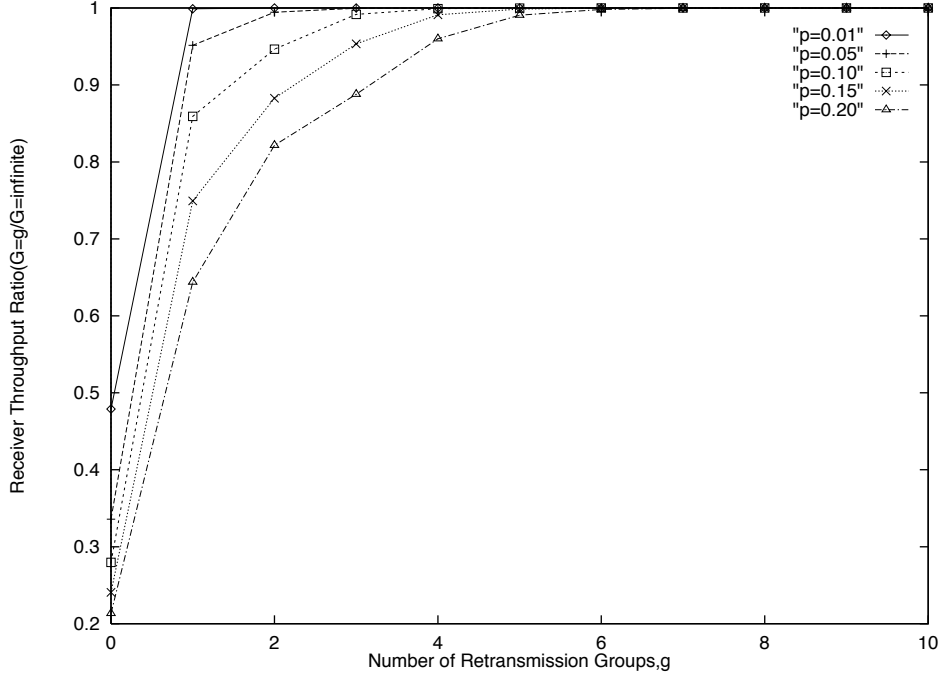
Figure 9: $\frac{\Delta}{\Delta'} = 1$ and $R = 1000$

By setting $R = 1$ in (15) and (16) we obtain

$$E[Z'(k)] = \frac{p^{\lfloor |k|\Delta/\Delta'\rfloor+1}}{1-p^2}, \qquad k = \pm 1, \pm 2, \dots \tag{17}$$

The expressions in (16)and (17) can be substituted into (12) to yield $E[U]$.

6.2 Numerical Results

We now examine the number of retransmission groups required to achieve a receiver through-put that is "close" to the receiver throughput obtained by using an infinite number of re-transmission groups. We also see how this number changes with loss probability p, the ratio Δ/Δ' and the number of receivers R. For this purpose, we define a performance metric γ, which is the ratio of receiver throughput with $G = g$ to receiver throughput with $G = \infty$, where $g \geq 0$. For P1, γ is defined as

$$\gamma = \frac{E[Y^{P1}]}{E[Y^{P1}] + pE[U]E[Y_p]}$$

Recall that $E[U]$ depends on the values of G, p, R and Δ/Δ' and thus γ depends on these variables too.

Figure 9 shows how γ varies with number of retransmission groups,g for several loss probabilities. In this figure $\Delta/\Delta' = 1$ and $R = 1000$. We found that as g increases the mean
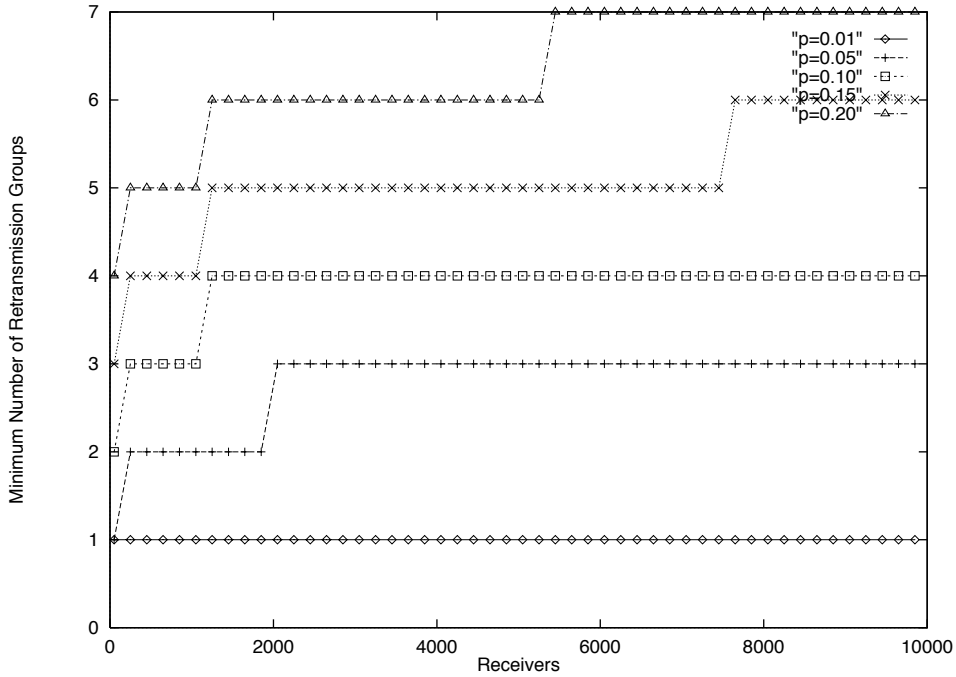
16

Figure 10: G^* vs R for $\frac{\Delta}{\Delta'} = 1$, $\gamma > 0.99$

number of unwanted packets, $E[U]$ falls very sharply to zero(not shown in the figure). This is because the larger the number of multicast groups, the larger will the separation (in time) between recovery of packets mapped to the same retransmission group. As a result γ in Figure 9 approaches 1 very fast. Let us define G^* be the minimum number of retransmission groups required to make $\gamma > 0.99$. We found that $G^* = 3$ for $p = 0.10$ and $G^* = 5$ for $p = 0.20$.

Figure 10 shows how G^* varies with the number of receivers for different loss probabilities, with $\Delta/\Delta' = 1$. We see that G^* increases very slowly with increasing number of receivers. For $p = 0.20$, $G^* = 5$ for $R = 1000$, $G^* = 6$ for $R = 5000$ and $G^* = 7$ for $R = 10000$. On the other hand, as the number of receivers decrease to a small value, the minimum number of retransmission groups does not drop considerably. For $p = 0.20$, even for 50 receivers we need at least 4 retransmission groups to keep $\gamma > 0.99$.

Thus we see that even for a very large number of receivers and relatively high loss probabilities the minimum number of retransmission groups required to achieve a receiver throughput that is close to the the receiver throughput with $G = \infty$ is very small. In the above numerical results we have considered the ratio Δ/Δ' to be 1. Next we see how this ratio affects G^*.

Figure 11 shows the behavior of G^* with Δ/Δ', for $R = 1000$, for several loss probabilities. We see that G^* is very sensitive to small values of Δ/Δ'. This is because the likelihood of overlap between retransmissions corresponding to the same retransmission group increases as Δ/Δ' decreases. This behavior diminishes as Δ/Δ' increases. In fact beyond a certain
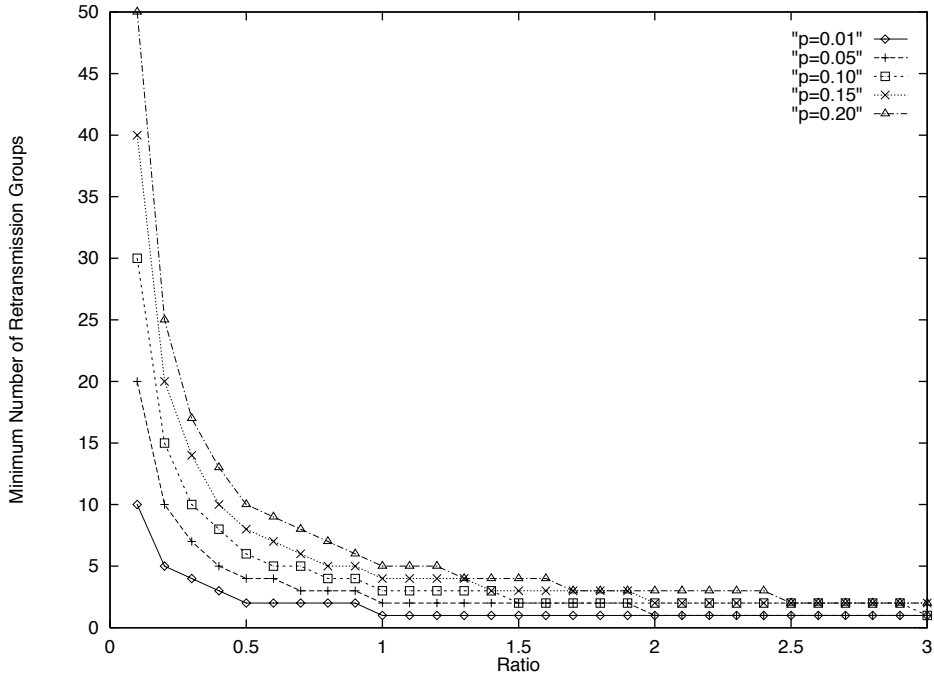
17

Figure 11: G^* vs $\frac{\Delta}{\Delta'}$ for $R = 1000$, $\gamma > 0.99$

value G^* becomes insensitive to Δ/Δ', as seen in Figure 11. Figure 11 also shows that for a given Δ/Δ', G^* is higher for higher loss probabilities. This increase can be attributed to the fact that there are more packets being recovered and hence an increasing chance of overlap between recovery of packets belonging to the same retransmission groups. Interestingly, with increase in Δ/Δ', G^* becomes insensitive to p. This is because the retransmissions of packets belonging to the same group are significantly separated. Hence, it is clear that Δ/Δ' plays an important role in determining the mean number of retransmission groups. The value of Δ/Δ' depends upon many factors, such as application requirements, network topology and network stability.

In summary, we see that for a wide range of system parameters we need a very small number of multicast groups to get a receiver throughput that is within 1% of the maximum achievable receiver throughput. If we choose a less stringent requirement and can tolerate unwanted processing slightly greater than 1% of the ideal case then we need even fewer multicast groups.

6.3 The Multiple Sender Case

In section 6.1 we analyzed the mean number of unwanted packets at a receiver due to a finite number of multicast groups for the *one-many* model. We now extend the analysis to the *many-many* scenario where each node is a receiver as well as a sender. Hence, instead of a single sender, we now have multiple senders. The protocol we consider is still P1. As
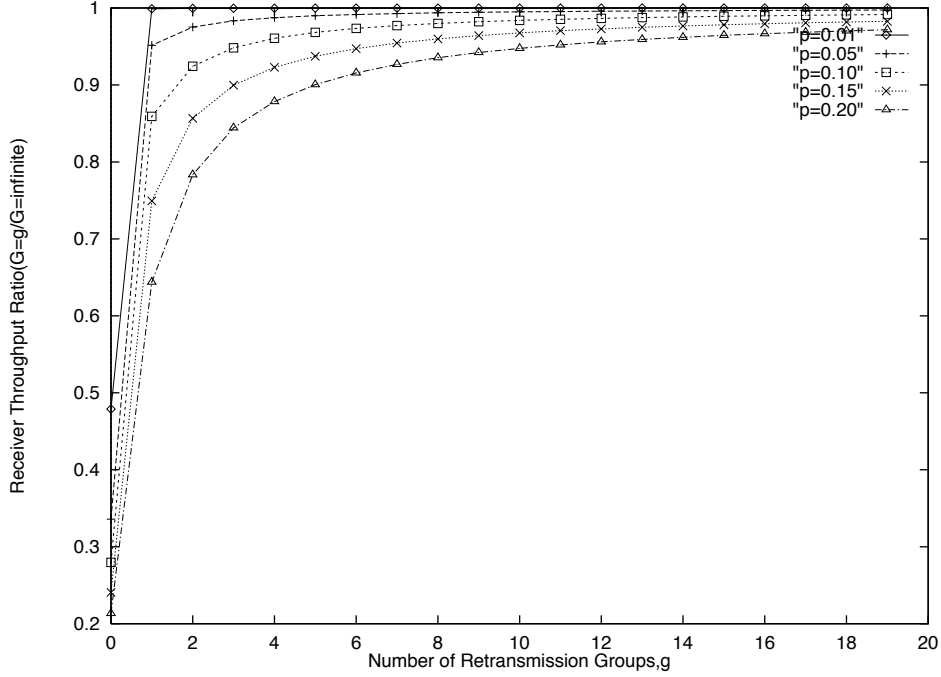
Figure 12: Many-Many Scenario, $\frac{\Delta}{\Delta'} = 1$ and $R = 1000$

before, there are G retransmission groups in addition to the group for original transmissions. A receiver on losing a packet i from sender j joins the multicast group $A_{i \bmod G}$ and sends a NAK to sender j. The sender j retransmits the packet on $A_{i \bmod G}$. After correctly recovering packet i from sender j the receiver leaves $A_{i \bmod G}$.

For the convenience of analysis we model the multiple senders by a single global sender. That is, we assume that all the transmissions and retransmissions are originating from the global sender. Let the time between original transmissions be Δ and let the time between retransmissions be Δ' (note that the Δ and Δ' for a global sender could be different from those for individual senders). A stream of original transmissions of packets from the global sender would look like $(s_1, n_1), (s_2, n_2), (s_3, n_3), \ldots$ where s_i is the source of the ith packet and n_i is the sequence number of that packet with respect to sender s_i.

Let us assume that traffic streams from senders are independent of each other. Hence a packet in the combined packet stream from the global sender is equally likely to belong to any of the retransmission groups. In other words, the probability of a packet belonging to a certain retransmission group(say g) is $1/G$. As before, we choose a packet from the global stream at random and call it 0. The expected number of unwanted packets received at a receiver r_1 when it is trying to recover packet 0 can be expressed as

$$E[U] = (1 - p)(\frac{1}{G}(\sum_1^\infty (E[Z(k)] + E[Z(-k)]) - \sum_1^\infty E[Z'(k)] + E[Z'(-k)]))) \qquad (18)$$

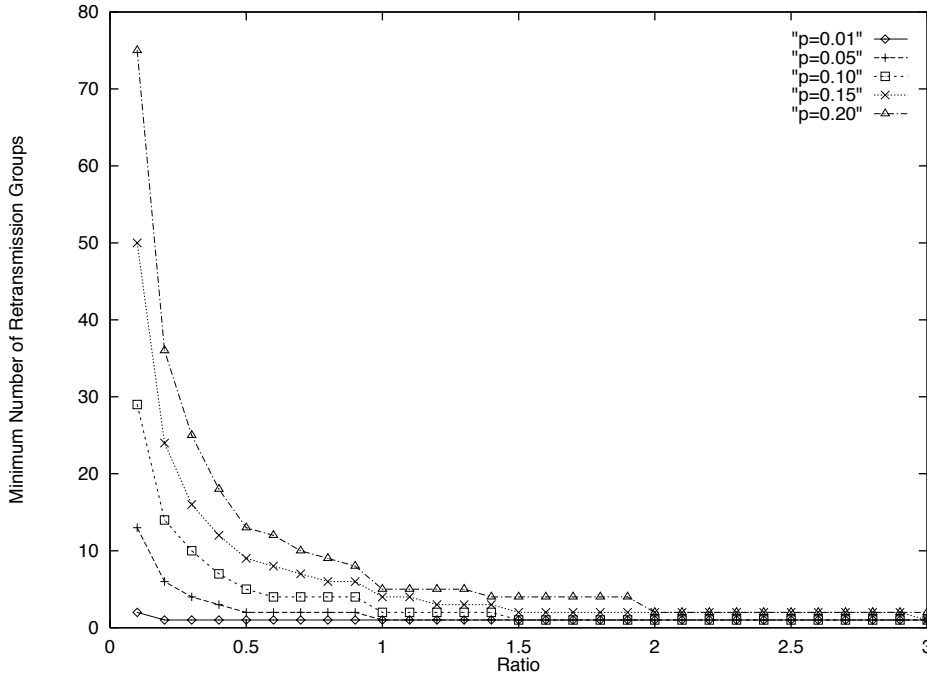Using equations (15)–(17) we can compute the value of $E[U]$ for different values of G, p, R

19

Figure 13: Many-Many Scenario, G^* vs $\frac{\Delta}{\Delta'}$ for $R = 1000$, $\gamma > 0.90$

and Δ/Δ'. We compute γ, as defined in section 6.2, for different values of g. Figure 12 shows how γ varies with g for $\Delta/\Delta' = 1$ and $R = 1000$. We see that γ increases very fast for small values of g, but then the rate of increase decreases as g increases. Still even for $p = 0.20$ we can get $\gamma > 0.90$ with just 5 retransmission groups. This means that with 5 retransmission groups we could get 90% of the benefit when using an infinite number of retransmission groups. We now define G^* to be the minimum number of retransmission groups required to keep $\gamma > 0.90$. Figure 13 show how G^* varies with Δ/Δ' for several loss probabilities for $R = 1000$. We can see from Figure 13 a wide range of values of Δ/Δ', G^* takes moderate to low values. Thus even for the multiple sender case we can get 90% of the benefit of infinite groups by actually using only a moderate or small number of groups. However, now the receivers must be prepared to tolerate a slightly larger number of unwanted packets than in the *one-many* case.

We end this section by making two observations. The first observation is that in a pathological case all the traffic streams from the senders might synchronize in transmitting packets such that packets with same sequence number get bunched together. If $A_{i \bmod G}$ is the retransmission group of packet i irrespective of the sender then a bunch of 1s(or 2s or 3s ...) would correspond to the same retransmission group and there is likely to be a great deal of overlap in their recovery. This would then result in a large number of unwanted packets at the receivers involved in recovering these packets. To reduce the likelihood of such a synchronization we could use certain deterministic rules. An example of one such rule is that each sender uses the retransmission groups in a different order, i.e. if there are

say 5 retransmission groups then sender 1 might use the retransmission groups in the order $[1, 4, 3, 0, 5, 2]$ and sender 2 might a different order like $[5, 3, 2, 0, 4, 1]$ and so on. This means that sender retransmits packet 0 on group 1, packet 2, on group 4 , packet 3 on group 3 and so on. The receivers know the order of retransmission group usage for each sender and hence join the appropriate group, depending on the the sender, for recovering packet i instead of joining $A_{i \bmod G}$ for all the senders.

Our second observation is that if the senders in the system send packets in batches then with the increase in batch size the separation between packets belonging to the same retransmission group would increase leading to a lesser degree of overlap between recovery of packets belonging to the same group. This would decrease the number of unwanted packets at the receivers and hence reduce the number of multicast groups needed.

7 A Scheme for Local Filtering

Even though the number of multicast groups required for achieving reliable multicast is not large, their use imposes demands on the network in the form of join and leave operations. In the current networks supporting IP multicast routing, a join or leave from a receiver is detected by the nearest multicast router, the one attached to the subnet of the receiver, through the IGMP protocol [7]. This information is then propagated to the nearest branching point of the multicast tree, rooted at the sender [6], or to a suitable core, in the case of core based trees [2]. Each join and leave message, results in processing overhead at all the intermediate routers thereby potentially reducing the router throughput. The significance of this processing burden is highly topology dependent. For example, if the network topology is such that there is always at least one receiver on the same subnet which is a member of the multicast group which another receiver in the same subnet wants to join, then the joining overhead on the network would be considerably reduced. More generally, the closer the branching point is to a receiver, the lower the join or leave overhead on the network. In order to minimize this signaling overhead we have designed a scheme that does *local filtering* at a receivers network interface. This introduces no additional signaling (i.e. join/leave packets to be sent to/from routers) in the network. Instead, all packets belonging to both the original transmission and all retransmission groups are always allowed to reach the local network to which the receiver is attached and then all unwanted packets are filtered out by the network interface hardware of the receiver.

In our scheme we distinguish between two kinds of join/leave operations. The first kind, which we call non-local join and non-local leave, is the regular join and leave as described above. The second kind, which we call local join and local leave, has to do only with the local network interface at a receiver. A local join or leave message travels only to the receiver's network interface hardware. No IGMP messages are sent to the nearest IP multicast router. A local join or leave is simply an indication to the hosts network interface to filter packets locally.

We now describe the scheme for local filtering. As a first step a receiver joins, non-locally, both the original transmission and all of the retransmission multicast groups. This results

21

in the transmission of IGMP messages to the nearest IP multicast router and transfer of subsequent graft messages towards the branching point. Subsequently, the receiver locally leaves all of the retransmission groups. From this point on, whenever the receiver needs to recover a packet it does a *local* join to the appropriate multicast group. Once the packet has been received correctly, the receiver does a local leave. Hence, as far as the network routing tables are concerned the receiver is always a member of all the groups and all packets to these groups are duly forwarded to the local interface of the receiver. It is left to the local interface to filter out the unwanted packets, based on the information provided by the local join and leave operations, to save the receiver from processing these packets. When a receiver wants to drop out of the multicast session it non-locally leaves all of the multicast groups.

Since the filtering is done locally at a receiver, this scheme does not reduce the data traffic in the network, and hence does not prevent wastage of network bandwidth. At the same time it does not add any extra traffic in comparison to the scenarios that use a single multicast group. The closer we move the filtering point to the sender the more we are able to save in terms of extra traffic towards a receiver, but then the cost of join and leave on the network would increase.

Note that our local filtering scheme does not require any changes at the network routers. There is, however, a need to modify the networking code at the receiver. This change appears not to be excessive. It is clear that for this scheme to work, the network interface hardware at a receiver must provide support for filtering. The Ethernet interface provides multicast filtering in the hardware. Finally, it should be noted that even with local filtering it is still necessary for the routers to include entries in the multicast routing table for each of the multicast groups.

8   Conclusions

In this paper we have examined an approach for providing reliable, scalable multicast communication by using multiple multicast groups for recovery of lost packets. In this approach, rather than having all receivers receive all retransmitted packets (regardless of whether a given receiver had already received a given packet correctly), the multiple multicast groups are used to allow only those receivers that actually *want* a particular packet to actually receive that packet.

We considered the idealized case of an infinite number of multicast channels as well as the more realistic scenario of using a small, fixed number of multicast channels. We also considered two different models of sender behavior: the one-many scenario and the many-to-many scenario. Our analytic models have demonstrated that significant performance gains (in terms of reduced receiver overhead, and an overall increased protocol throughput in the case of many-many communication) can be realized in such environments, and over a range of system parameters. We also presented a local filtering scheme for minimizing join/leave signaling when multiple multicast groups are used.

Our work can proceed in several directions. There is a need to study the affect of using finite number of groups in the context of protocols P2 and P3. We have considered a

round–robin approach in reusing the multiple retransmission groups; other approaches are also possible. Also, we have assumed that losses are spatially and temporally independent. In [19], Yajnik *et al.* have observed some long bursts of losses on the MBone. They have also observed some spatial correlation in loss. We have grouped packets in such a way that packets belonging to a group are recovered on a separate multicast channel. It would be interesting to combine our approach with the previously studied approach of destination set grouping [1] in which receivers are grouped based on their capabilities. Our local filtering reduces the processing overhead at the receivers but does not help in reducing the used network bandwidth. Hence there is also a need to study better mechanisms for filtering to be able to save on both receiver and router processing overheads and network bandwidth.

References

[1] M.H. Ammar and L. Wu, *Improving the Throughput of Point–to–Multipoint ARQ Protocols Through Destination Set Splitting.* Proceedings of IEEE INFOCOM, pages 262-269, June 1992.

[2] A.J. Ballardie, P.F. Francis and J. Crowcroft, *Core Based Trees.* Proceedings of ACM SIGCOMM. 1993

[3] D. Cheriton, End-to-End mailing list, September 1994.

[4] S.Y. Cheung, M.H. Ammar and X. Li, *On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution.* Tech Report: GIT-CC-95-25, Georgia Institute of Technology, Atlanta, July 1995.

[5] J. Crowcroft, End-to-End mailing list, September 1994.

[6] S. Deering, *Multicast Routing in Datagram Internetwork.* Ph.D. Dissertation, Stanford University. December, 1991.

[7] S. Deering *Host Extensions for IP Multicasting.* RFC 1112. August 1989.

[8] S. Floyd, V. Jacobson, S. McCanne, C. Liu and L. Zhang, *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.* Proceedings of ACM SIGCOMM, pages 342-356, August 1995.

[9] S. Floyd, V. Jacobson, S. McCanne, C. Liu and L. Zhang, *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.* A later version of the ACM SIGCOMM paper.November 1995.

[10] M. Grossglauser, *Optimal Deterministic Timeouts for Reliable Scalable Multicast.* Proceedings of IEEE INFOCOM, pages 1425-1432, April 1996.

[11] H.W. Holbrook, S.K. Singhal and D.R. Cheriton, *Log-Based Receiver Reliable Multicast for Distributed Interactive Simulation.* Proceedings of ACM SIGCOMM, pages 328-341, August 1995.

[12] Institute for Simulation and Training, *Standard for Distributed Interactive Simulation - Application Protocols.* Technical Report IST-CR-94-50, University of Central Florida, Orlando, Fla, 1994

[13] J. Kay and J. Pasquale, *The Importance of Non-Data Touching Processing Overheads in TCP/IP.* Proceedings of ACM SIGCOMM, 1993.

[14] S. McCanne, V. Jacobson and M. Vetterli, *Receiver-driven Layered Multicast.* Proceedings of ACM SIGCOMM, August 1996.

[15] C. Partridge and S. Pink, *A Faster UDP.* IEEE/ACM Transactions in Networking, Vol1., No.4, pages 429-439, August 1993.

[16] S. Pingali, D. Towsley and J. Kurose, *A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols.* Proceedings of ACM Sigmetrics. 1994.

[17] D. Towsley, J. Kurose and S. Pingali, *A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols.* Submitted for Publication.

[18] R. Yavatkar, J. Griffioen and M. Sudan, *A Reliable Dissemination Protocol for Interactive Collaborative Applications.* Proceeding of ACM Multimedia. November 1995.

[19] M. Yajnik, J. Kurose and D. Towsley, *Packet Loss Correlation in the MBone Multicast Network.* To appear in Global Internet Conference, November 1996.