Scalable Reliable Multicast using Multiple Multicast Channels

Sneha K. Kasera*
Univ of Massachusetts Amherst
{kasera@cs.umass.edu}

Gísli Hjálmtýsson
AT&T Labs Research
{gisli@research.att.com}

Don Towsley and Jim Kurose
Univ of Massachusetts Amherst
{towsley,kurose@cs.umass.edu}

Abstract

We examine an approach for providing reliable, scalable multicast communication, involving the use of multiple multicast channels for reducing receiver processing costs and reducing network bandwidth consumption in a multicast session. In this approach a single multicast channel is used for the original transmission of packets. Retransmissions of packets are done on separate multicast channels, which receivers dynamically join and leave. We first show that protocols using an infinite number of multicast channels incur much less processing overhead at the receivers compared to protocols that use only a single multicast channel. 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 channels, for a specific negative acknowledgment (NAK)–based protocol. We then explore the minimum number of multicast channels 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 channels). For an application consisting of a single sender transmitting reliably to many receivers we find that only a small number of multicast channels 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 channels is needed. Finally, we present two mechanisms for implementing multiple multicast channels, one using multiple IP multicast groups and the other using additional router support for selective packet forwarding. We discuss the impact of both mechanisms on performance in terms of end–host (sender, receiver) and network resources. The approach of implementing multiple multicast channels that uses additional router support reduces both end–host processing costs and network bandwidth usage.

1 Introduction

Many applications such as shared whiteboard, multicast file transfer, stock quote dissemination, web cache updates, distributed interactive simulation and distributed computing, require reliable multicast, where sender(s) transmit data to a group of receivers in a reliable manner. Using multicast, rather than sending to each receiver individually, has the potential of saving on sender

*Correspondence author.

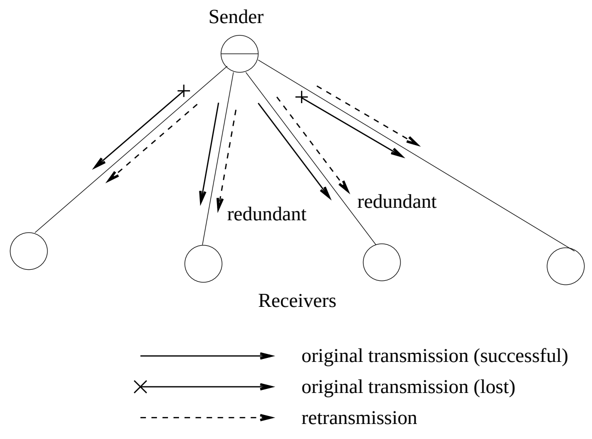
Sender
redundant
redundant
Receivers
original transmission (successful)
original transmission (lost)
retransmission

Figure 1: Retransmission Scoping Problem

and network resources, and reducing the time to completion by overlapping transmission to multiple receivers. The design of reliable multicast architectures and protocols that make efficient use of both the network and end–host resources, and scale to applications that can potentially have several thousand receivers, is a challenging task.

In this paper we examine an approach for providing reliable, scalable multicast communication, with end-to-end loss repairs, with the goal of reducing receiver processing overhead and reducing network bandwidth consumption. This approach (first suggested in [4] and [6]) involves the use of multiple multicast channels. To illustrate the problem, consider a reliable multicast scenario using a single multicast channel. All packet transmissions and retransmissions are done over the single multicast channel. Each receiver therefore receives all of the retransmissions of a packet, even after correctly receiving the packet (see Figure 1). This imposes unnecessary receiver processing overhead and wastes network bandwidth on links leading to receivers who have already received the packet, especially as the number of receivers increases. Hence a fundamental problem in reliable multicast is how to scope retransmissions so as to shield receivers and the links leading to them from loss recovery due to other receivers.

In this paper we examine an approach that allows one to overcome this *retransmission scoping problem* in a multicast scenario. The approach can be informally described as follows. 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 channel is used for the original transmission of packets from the sender. A lost packet is retransmitted on one of the remaining G channels. Packets are assigned in a deterministic manner to the channels. After detecting a packet loss, a receiver “joins” the appropriate channel for that packet’s retransmission. Once the lost packet is received, the receiver “leaves” the appropriate channel. Previous work on reliable multicast has focussed on the case $G = 0$, where all transmissions and retransmissions are sent on the same multicast channel.

We begin by showing that protocols using an infinite number of multicast channels incur much less processing overhead at the receivers compared to protocols that use only a single multicast channel.

Next, through analysis, we derive an expression for the the number of unwanted packets at a receiver due to using only a finite number of multicast channels, for a specific negative acknowledgment (NAK)–based protocol. We then explore the minimum number of multicast channels 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 channels). For an application consisting of a single sender transmitting reliably to many receivers (referred to as a *one–to–many* application [30]) we find that only a small number of multicast channels 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–to–many* application [30]) a larger but still moderate number of multicast channels is needed.

The use of multiple multicast channels is expected to reduce the network bandwidth consumption as well as receiver processing cost. This is because retransmissions are only forwarded along the links leading to subscribers of the retransmissions channels. However, in practice the realizable bandwidth savings is tied to the mechanism used to implement multiple multicast channels. In this paper, we explore two mechanisms, one using multiple IP multicast groups and the other using additional router support for selective packet forwarding (see [10]). We identify the advantages and disadvantages of both mechanisms and discuss their impact on performance in terms of end–host and network resources.

We find that due to practical limitations, including potentially high join and leave latency, achieving bandwidth savings in the network may be difficult using current IP multicast, even though it is possible to reduce receiver processing costs as described later. We also find that the approach of implementing multiple multicast channels that uses additional router support for selective packet forwarding, offers greater opportunities for saving both sender and receiver processing costs and network bandwidth. In this approach, additional resources are required at the routers in terms of processing for implementing the selective packet forwarding functionality, and, memory for storing state associated with the selective packet forwarding. In this paper, we do not study the processing overheads required to support selective packet forwarding at the routers but assume that router vendors will be willing to expend the necessary processing and memory resources at the routers to facilitate the savings in end–host processing and network bandwidth. Pragmatic General Multicast, PGM [28], from CISCO proposes to use router support for selective packet forwarding for reliable multicast. This corroborates our view that router assisted implementation of multicast channels is practical. The analysis presented in this paper, with minor modifications, is also applicable to PGM.

The remainder of the paper is organized as follows. In the next section we examine existing work on the subject. In Section 3 we present three protocols that use multiple multicast channels. 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 channels leads to reduction of processing overhead at the receivers. In Section 6 we derive the number of multicast channels 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 mechanisms for implementation of multiple multicast channels. Conclusions and suggestions for future work are contained in Section 8.

2 Related Work

Reliable multicast has been an active research area in the last few years. Several architectures and protocols have been proposed. In this section we present some of the existing and on-going research in reliable multicast.

One of the most popular existing reliable multicast protocols is scalable reliable multicast, SRM [9]. SRM is a NAK–based protocol that has been implemented for a shared whiteboard application. In its basic form, SRM suffers from the problem of unwanted redundant packets being sent to, and processed at, receivers. Local recovery enhancements in SRM [9] are likely to scale down this problem but not solve it. Local recovery helps to isolate the domains of loss and, thereby, reduce global retransmissions. For a multicast application with thousands of local neighborhoods, unless receivers are arranged in a hierarchy with a small bounded degree and retransmission of packets is properly scoped to remain within the neighborhood, the receivers will still receive unwanted retransmissions if only one channel is used for both transmissions and retransmissions. Log–based reliable multicast, LBRM [12], and reliable multicast transport protocol, RMTP [22], are two hierarchical approaches[1] in which designated receivers (or loggers) at a certain level supply repairs to lower level designated receivers or loggers. The problem of placing these designated receivers and determining their processing and storage requirements is still being studied.

Recently, there has been an increasing interest in using processing and storage inside the network for enhancing reliable multicast performance [10, 17, 18, 21, 26, 28]. Control-on-demand [10], pragmatic general multicast, PGM [28], and active reliable multicast, ARM [18], propose to maintain retransmission state for selective forwarding of retransmissions and for duplicate NAK suppression. In this paper, we propose to use the control–on–demand [10] architecture for implementing multiple multicast channels. The analysis presented in this paper, with minor modifications, is also applicable to PGM. ARM, control–on–demand and L1 [17] propose the use of buffering "current-data" at strategic locations inside the network for providing local retransmissions. Even though it has been shown in [17] that buffering data inside the network for the purpose of retransmissions, results in higher performance, the issues of where to place buffers and when and how to invoke "repair service" remain open questions. In this paper we focus on error recovery only from the sender.

Reliable multicast could benefit from the use of forward error correction (FEC) [24]. This is because the FEC techniques allow recovery of multiple lost packets with the help of a single FEC packet. However, FEC–based loss recovery, using end–to–end means only, will not perform well in the presence of heterogeneous loss. Even if only a few receivers experience very high loss, a large number of FEC packets will be generated at the sender and will be sent everywhere thereby wasting network bandwidth and causing unnecessary packet processing at all the other receivers.

[1]Several other hierarchical approaches such as [11, 20, 33] also exist.

FEC techniques could be combined (as proposed in [28]) with the mechanism of using additional router support for implementing multiple multicast channels to enhance our work.

In [9] an approach based on IP *Time–to–live* or TTL has been proposed for scoping retransmissions. There are two problems with TTL based scoping. First, TTL based scoping limits packets within a radius and is not suitable for tree structures as in the case of multicast. Second, it is hard to approximate a good TTL value.

We now look at existing work that uses multiple multicast groups or channels. Cheung *et al.* [5], McCanne [23] and Vicisano [32] have proposed to use 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 [4] and Crowcroft [6] first suggested the use of multiple multicast groups for error recovery in reliable multicast, in a discussion on the end-to-end mailing list. Holbrook [12] 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 that the sender would carry 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 three significant ways. 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.

Among the existing analytical work on reliable multicast, the work of Towsley, Kurose and Pingali presented in [27, 30] provides a simple analytical framework for studying the performance of reliable multicast protocols. They have used this framework for a quantitative demonstration of the superiority of receiver-initiated NAK–based approaches over sender-initiated ACK–based approaches. This work has subsequently been used in several performance analyses such as [17, 19, 20, 24]. It also forms the basis of our analyses.

3 Protocols and System Model

We now present three generic NAK–based protocols for using multiple multicast channels for reliable multicast from a sender to several receivers. Based on arguments presented in [9] and [27, 30], receiver–based reliability (i.e., NAK-based schemes) outperform schemes employing sender–based reliability in providing reliable multicast for many applications of interest. Hence we focus only on receiver–based recovery, or negative acknowledgment (NAK)–based schemes in our work. The section ends with a description of the applications and the network model.

3.1 Protocol Description

The protocols described below are modified versions of the generic protocols, N1 and N2, proposed in [27, 30]. In [27, 30], the authors have considered only one multicast channel for both transmissions and retransmissions. The reliable multicast protocols we will consider will be denoted P1, P2 and P3. These protocols are very generic in nature. The details of any specific implementations of these protocols are not discussed in this paper.

We initially make the assumption that we have an infinite number of multicast channels at our disposal so that each packet can be recovered on a separate channel, i.e., $G = \infty$. Later, we will observe that only a small number of multicast channels is required to achieve almost the same effect as can be obtained with an infinite number of channels. The discussion below is based on an IP multicast like network scenario, where the sending of a packet on a multicast channel causes that packet to be sent (potentially with some loss) to the members subscribing to that channel.

Protocol P1 exhibits the following behavior

- the sender sends all original transmissions on a multicast channel A_{org}.
- when required, the sender retransmits a packet with sequence number i on multicast channel A_i, where $i = 0, 1, 2, ...$
- whenever a receiver detects a lost packet i, it subscribes to the multicast channel A_i and transmits a NAK to the sender over a point-to-point channel and starts a timer.
- the expiration of a timer without prior reception of the corresponding packet serves as the detection of a lost NAK or retransmission, a NAK is retransmitted for the associated packet, and a timer is started again.
- on receiving packet i on A_i, a receiver unsubscribes from A_i.

Protocol P1 is equivalent to N1 in [27, 30] when only A_{org} is used for both transmissions and retransmissions.

Protocol P2 exhibits the following behavior

- the sender sends all original transmissions on a multicast channel A_{org},
- the sender retransmits packet i on multicast channel A_i, where $i = 0, 1, 2, ...,$
- whenever a receiver detects a lost packet (say i), it subscribes to multicast channel A_i, waits for a *random period of time* and then multicasts a NAK on the channel A_i, and starts a timer,
- upon receipt of a NAK for a packet that a receiver has not received, but for which it (the receiver) has 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 or retransmission.

- on receiving packet i on A_i, a receiver unsubscribes from A_i.

Protocol P2 is equivalent to N2 in [27, 30] when only A_{org} is used for both transmissions and retransmissions. Protocol P3 exhibits the same behavior as P2 except that a receiver sends a NAK for packet i on the original multicast channel A_{org} instead of on A_i. The important similarity between P2 and P3, which distinguishes them from P1, is that both suppress NAKs [9] 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 participating receivers. This 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 only at a few receivers. In comparison, in P3, NAKs are retransmitted on A_{org} and are received by all receivers that have subscribed to A_{org}. It should be noted that the sender has to subscribe to all retransmission channels in P2 and only subscribe to channel A_{org} in P3. A sender does not have to subscribe to any retransmission channel in P1.

Before quantitatively evaluating the performance of P1, P2 and P3, let us first qualitatively examine their behavior. In P2, NAKs are received *by only* those receivers that have lost packet i. Thus 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 receivers participating in the multicast session. Some of these receivers might have received packet i correctly and could then retransmit the NAKed packet. Thus P3 could be easily modified to include local recovery from other receivers. Another difference is that the performance of P2 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 channel. 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 subscribe to A_{org}, this situation would not occur.

3.2 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–to–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–to–many* application (e.g., distributed interactive simulations [8]), 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 for each packet there is a single sender and each node is equally likely to be the sender. That is, a node is a sender with a probability $1/(R+1)$ and is a receiver with probability $R/(R+1)$ [30]. 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. Our analysis can be generalized to the case where the probability of packet loss is distinct for different receivers. We further assume that NAKs are never lost. This assumption can be relaxed by following the analysis given in [31].

As noted earlier, we will begin our analysis below by assuming an infinite number of available multicast channels and later consider the case when there is 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 protocols P1, P2 and P3. The bandwidth improvement is tied to the implementation and hence will be studied in the section (Section 7) on implementation mechanisms. For analyzing the processing costs, we start with the *one–to–many* model and later use the results of the *one–to–many* model to obtain processing costs for the *many–to–many* model.

The receive processing cost 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 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 protocols P1, P2 and P3. Table 1 describes the notation used in the analysis. Most of the notation has been reintroduced from [30]. We assume that the processing times have general distributions and that they are independent of each other.

Following an approach similar to the one in [30], the mean per packet processing time for a randomly chosen packet at a receiver for P1 can be expressed as,

$$E[Y^{P1}] = E[Y_p] + pE[Y_j] + E[(M_r-1)^+]E[Y_n] + E[(M_r-2)^+]E[Y_t] \qquad (4.1)$$

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 channel. Note that the join and leave processing times have been multiplied by the loss probability, p, because this cost is incurred only when a packet is lost and a NAK is 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 channel at most once per packet recovery. 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

X_p	–	sender time to process the transmission of a packet
X_n	–	sender time to process a NAK
Y_p	–	receiver time to process a newly received packet
Y_t	–	receiver time to process a timeout
Y_n	–	receiver time to process and transmit a NAK
$Y_n^{'}$	–	receiver time to receive and process a NAK
Y_j	–	receiver time to process a join and the corresponding leave
X^{ω},Y^{ω}	–	the send and receive per packet processing time in protocol $\omega \in$ {P1, P2, P3, N1, N2}
Z^{ω}	–	the sum of send and receive per packet processing time at a node in protocol $\omega \in$ {P1, P2, P3, N1, N2} (used in the many–to–many case)
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; $M = max_r\{M_r\}$

Table 1: Notation

receives the packet, is independent of the number of multicast channels (and indeed is independent of whether an ACK or NAK protocol is used), we can use the results in [30] directly:

$$\begin{aligned} E[M_r] &= 1/(1-p), && (4.2)\\ E[(M_r-1)^+] &= p/(1-p), && (4.3)\\ E[(M_r-2)^+] &= p^2/(1-p). && (4.4) \end{aligned}$$

Substitution of (4.3) and (4.4) into (4.1) yields

$$E[Y^{P1}] = E[Y_p] + pE[Y_j] + pE[Y_n]/(1-p) + p^2E[Y_t]/(1-p). \qquad (4.5)$$

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

$$\begin{aligned} E[Y^{P2}] &= E[Y_p] + pE[Y_j] \\ &\quad + (E[(M_r-1)](E[Y_n]/R + (R-1)E[Y_{n'}]/R) \\ &\quad + E[(M_r-2)^+]E[Y_t] \qquad (4.6) \end{aligned}$$

Here, the first two terms are the same as in the case of P1. The third term consists of two parts. The first one results from NAK generation and the second results from NAK reception. A receiver generates a NAK with probability $1/R$ and receives a NAK with probability of $(R-1)/R$. The last term is the same as that for P1. Substitution of (4.2) and (4.4) into (4.6) yields

$$E[Y^{P2}] = E[Y_p] + pE[Y_j] + p/(1-p)(E[Y_n]/R + (R-1)E[Y_{n'}]/R) + p^2E[Y_t]/(1-p) \tag{4.7}$$

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

$$\begin{aligned} E[Y^{P3}] &= E[Y_p] + pE[Y_j] + (E[M]-1)(E[Y_n]/R + (R-1)E[Y_{n'}]/R) \\ &\quad + p^2E[Y_t]/(1-p) \end{aligned} \tag{4.8}$$

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 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 NAKs sent for packet i are received by all receivers. $E[M]$ can be expressed in terms of R and p as follows.

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

In Section 7.2 we will observe that when we use additional router support to implement multiple channels there are no explicit join/leave operations and hence the receivers do no incur any join/leave processing cost as above.

The mean per–packet processing times at a receiver, for the protocols N1 and N2 are given by the following expressions from [30],

$$E[Y^{N1}] = E[M](1-p)E[Y_p] + pE[Y_n]/(1-p) + p^2E[Y_t]/(1-p) \tag{4.9}$$

$$\begin{aligned} E[Y^{N2}] &= E[M](1-p)E[Y_p] + (E[M]-1)(E[Y_n]/R + (R-1)E[Y_n^{'}]/R) \\ &\quad + p^2E[Y_t]/(1-p) \end{aligned} \tag{4.10}$$

The sender processing costs of P1 and P2 are the same as that for N1, the sender processing cost of P3 is the same as that for N2. This is because the use of multiple multicast channels only reduces the 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 subscribe to any retransmission channel because it does not receive anything on the retransmission channels. In P1, NAKs are received point–to–point and in P3, NAKs are received on the original transmission multicast address. In P2 the sender must subscribe to all of the retransmission channels before it starts transmitting packets because it receives NAKs on these retransmission channels. It remains a member until the multicast session ends. Hence the sender does not incur any join/leave processing costs during the session. We can thus use the expressions of sender processing costs from [30]. The mean *sender* processing time needed to successfully transmit a packet to all receivers, for protocols P1, P2 and P3 is given by the following expressions.

$$E[X^{P1}] = E[X^{N1}] = E[M]E[X_p] + RpE[X_n]/(1-p) \tag{4.11}$$

$$E[X^{P2}] = E[X^{P3}] = E[X^{N2}] = E[M]E[X_p] + (E[M]-1)E[X_n] \tag{4.12}$$

Later in Section 7.2, we will see that when we use additional router support to implement multiple multicast channels, the number of NAKs received by the sender in P1 reduces to that received by the sender in P2.

Recall that under the *many–to–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 expressed as

$$E[Z^{\omega}] = E[X^{\omega}]/(R+1) + R\,E[Y^{\omega}]/(R+1) \tag{4.13}$$

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

Heterogeneous Receivers

The above analysis can be extended to include heterogeneous receivers. If receiver r loses a packet with probability p_r, then the mean receiver processing costs for receiver r, under protocols P1, P2 and P3, can be obtained by replacing p by p_r in equations 4.5–4.8. The mean receiver processing costs for receiver r, under protocols N1 and N2, can be obtained by replacing p by p_r in equations 4.9 and 4.10. Equations 4.2–4.4 also remain the same except that p is replaced by p_r. $E[M]$ is now given by the following equation.

$$E[M] = 1 + \sum_{m=1}^{\infty} (1 - \prod_{r=1}^{R} (1 - p_r^m))$$

The expressions for $E[X^{P2}]$, $E[X^{P3}]$ and $E[X^{N2}]$ remain the same as given by equations 4.11 and 4.12. The expression for $E[X^{P1}]$ and $E[X^{N1}]$ is now given by the following equation.

$$E[X^{P1}] = E[X^{N1}] = E[M]E[X_p] + \sum_{r=1}^{R} p_r E[X_n]/(1-p_r)$$

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 150 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,

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)

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 an average reported. The timer processing time includes the 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 leave operations. The join and leave operations were "local" meaning that no IGMP reports were sent out as part of these operations. Higher processing costs will incur if IGMP reports are also sent out. The concept of "local" join and leave is discussed later in the section on local filtering.

It is worth noting that several measurement studies of per–packet processing times have been reported in the literature, notably [15] and [25]. However, neither of these included measurements of join and leave processing times, thus necessitating the series of measurements described above.

5.1 One–to–many Model

Using the measured processing times, we can compute the mean send and receive processing costs(times) for the protocols P1, P2, P3, N1 and N2 using equations (4.5)–(4.12), for several values of R and p.

Figures 2-4 show how the ratios of receiver processing costs obtained under the P and N protocols vary with R and p. It can be seen in each of these graphs that the family of P protocols always perform better. This is because the P protocols, by using multiple multicast channels, eliminate the reception of unwanted and redundant data packets at the receivers and hence receiver processing

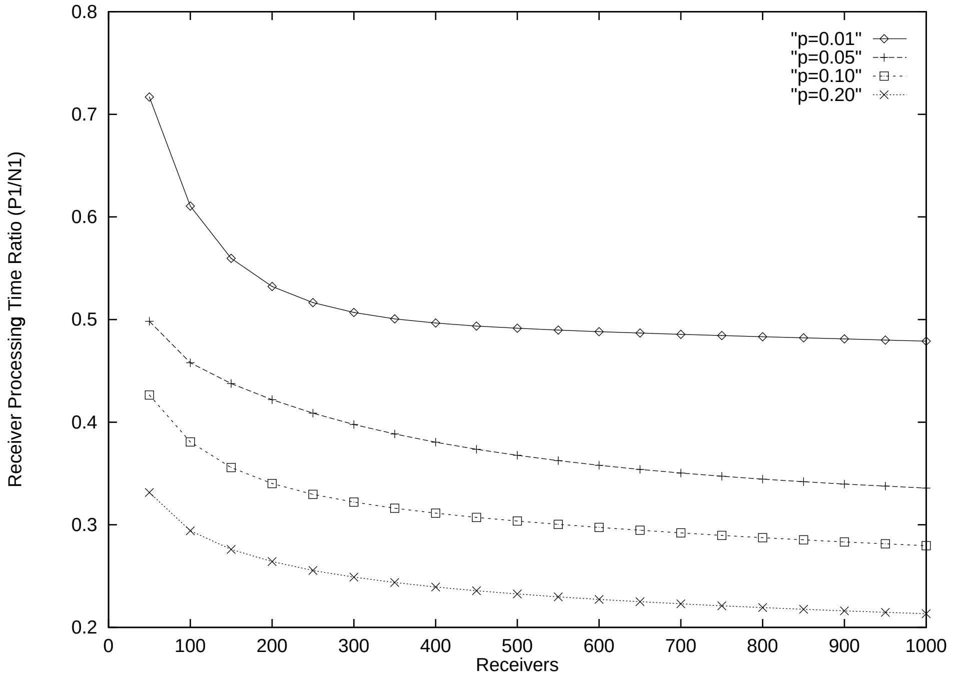

Figure 2: Receiver processing cost reduction of P1 over N1

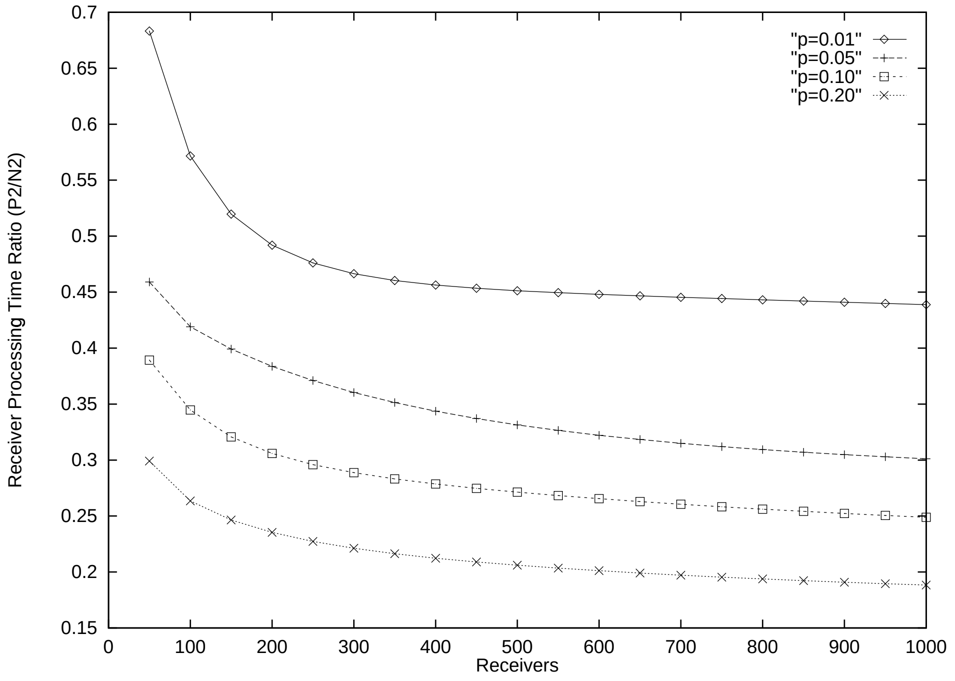

Figure 3: Receiver processing cost reduction of P2 over N2

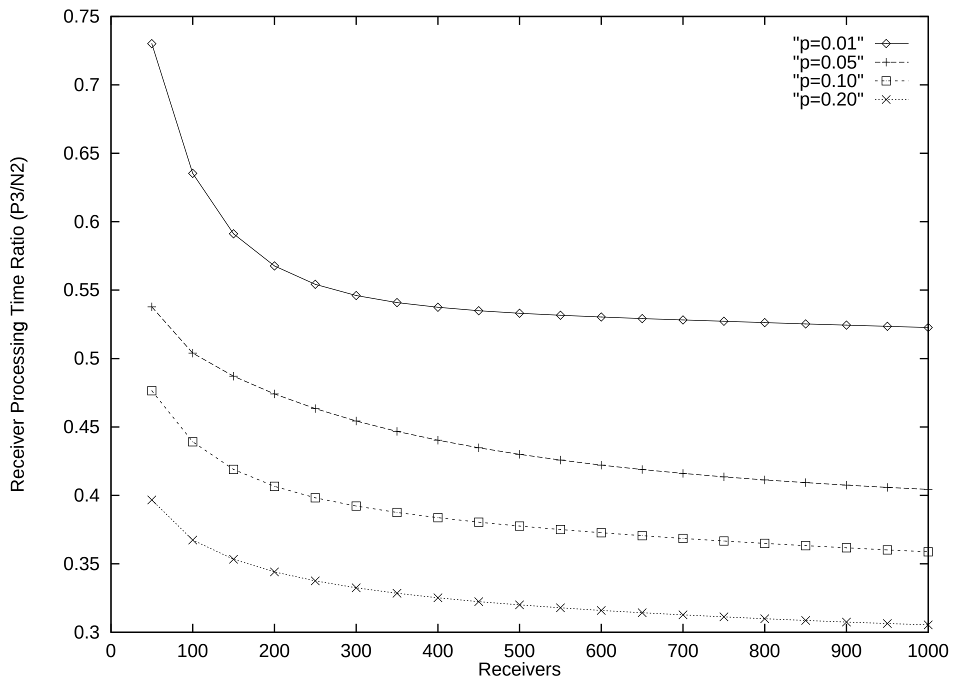

Figure 4: Receiver processing Cost Reduction of P3 over N2

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 over six times more costly then the processing a join or leave. The second 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 a small number of receivers. This number increases with p and R.

Figures 2 and 3 show the receiver processing reduction 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 [30]. On the other hand the receiver processing times of P1 and P2 are almost same. In fact, from equations (4.5) and (4.7) 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 all practical purposes.

Figure 4 shows the receiver processing performance of P3 over N2. Although P3 involves more

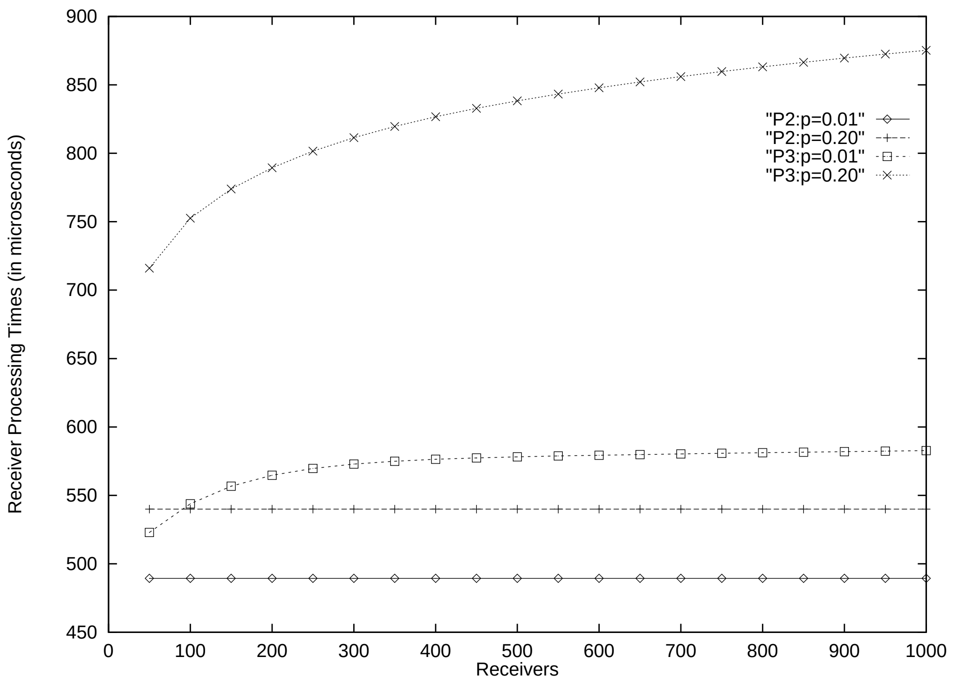

Figure 5: Receiver Processing Costs of P2 and P3

processing than P2, it still substantially outperforms N2. Note that in Figures 2–4, the reduction in receiver processing cost flattens out with increasing R. This is because the receiver processing cost of protocols N1 and N2 increases as $\log R$ [30]. The receiver processing cost of P1 and P2 is independent of R. The receiver processing cost of P3 also increases as $\log R$ but much more slowly than N2.

Figure 5 shows the receiver processing times of P2 and P3. As expected, we see that P3 incurs higher receiver processing costs than P2 (and therefore P1), as expected. Recall from Section 3 that P3 must perform extra NAK processing in comparison to P1 and P2.

5.2 Many–to–Many Model

In this section, we examine the *many–to–many* model, using equation (4.13) to compute the mean per–packet overall processing cost at a node for protocols P1, P2 and P3 and N1 and N2. Figures 6–8 show how the *many–to–many* overall processing cost ratio of the P and the N protocols varies with the number of receivers for different loss probabilities. We see in Figure 6 that P1 outperforms N1 and in Figures 7 and 8 that P2 and P3 outperform N2. This follows from the fact that a participant in a *many–to–many* application is much more likely (with probability $R/(R+1)$) to perform receive processing of a packet than send processing. Consequently the mean per packet overall processing cost ratios exhibited in Figures 6, 7, and 8 exhibit behavior nearly identical to the mean per packet receive processing ratios for those protocols that we saw previously in Figures 2, 3, and 4. Among

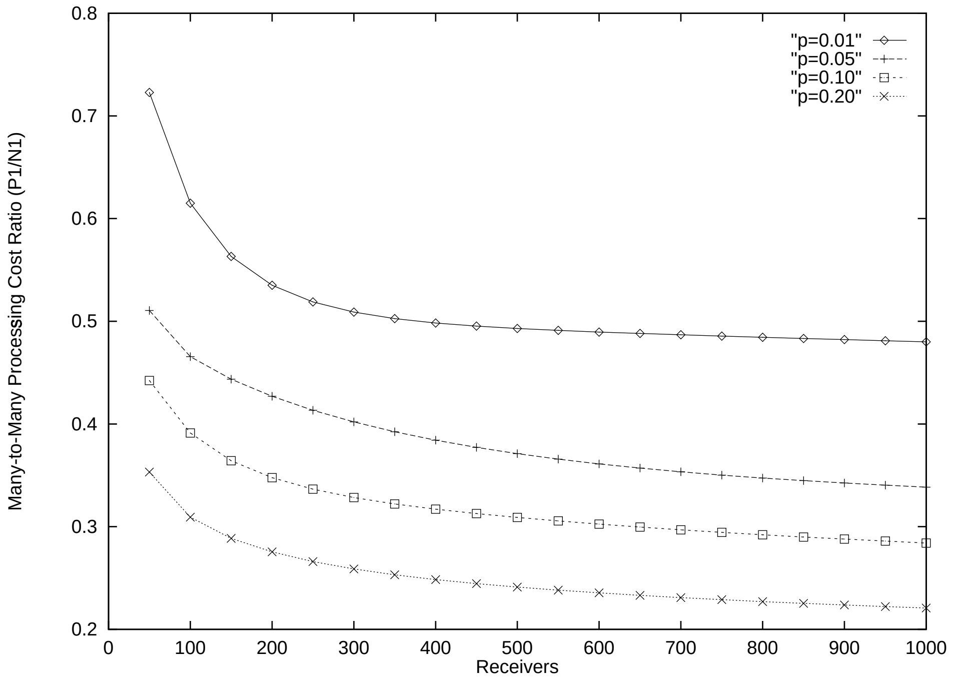

Figure 6: Many–to–Many: Overall processing cost reduction of P1 over N1

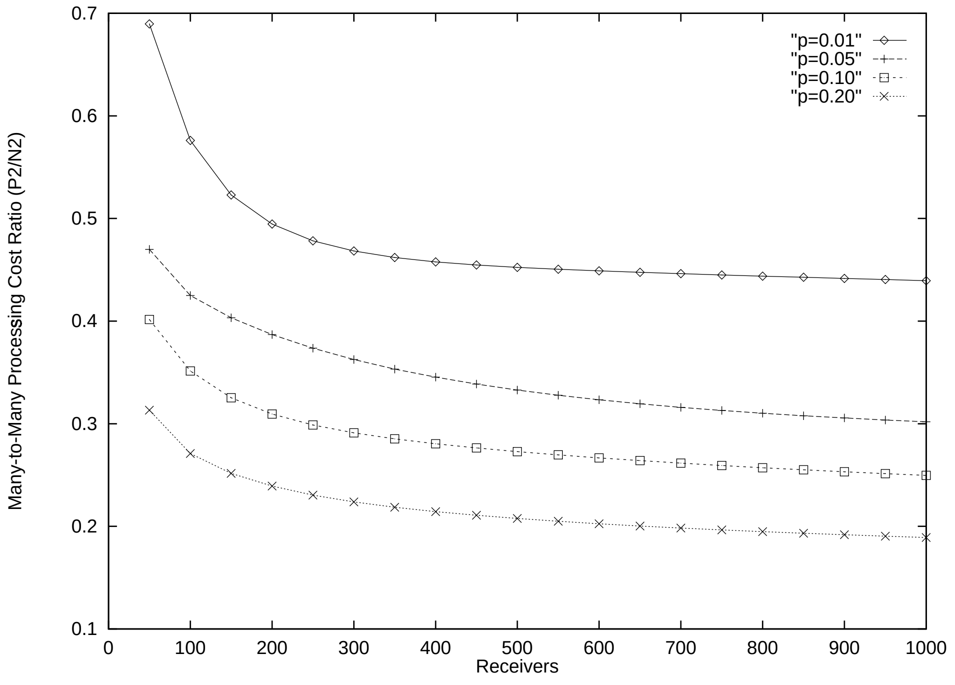

Figure 7: Many–to–Many: Overall processing cost reduction of P2 over N2

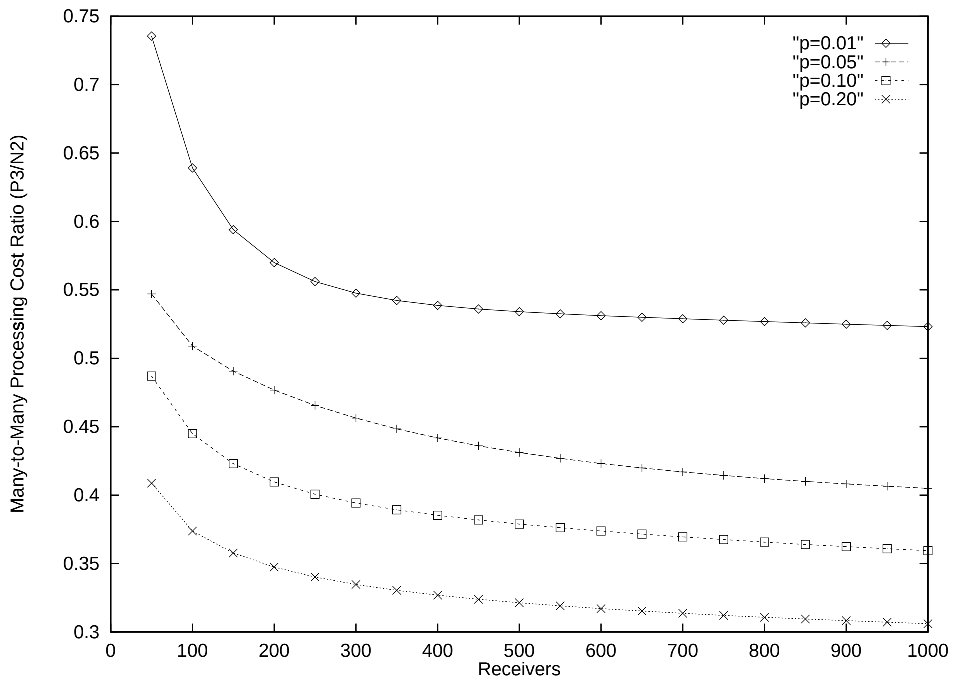

Figure 8: Many–to–Many: Overall processing cost reduction of P3 over N2

the P protocols, the mean per packet processing cost under P2 is only slightly lower than that of P1. This is because the receiver processing cost is the same for P1 and P2 and the sender processing cost, even though lower for P1 due to NAK suppression, influences the mean per packet processing cost only slightly as R increases. Both P1 and P2 have lower overall processing cost than P3, as they have lower receiver processing cost.

If end host processing were to determine the protocol throughput (i.e., other factors such as network bandwidth are not bottlenecks) then protocol throughput for the *one–to–many* model could be defined as $\min\{1/E[X^{\omega}], 1/E[Y^{\omega}]\}$, where $\omega \epsilon \{N1, N2, P1, P2, P3\}$ (see [30]). Similarly, the protocol throughput for the *many–to–many* model could be defined as $1/E[Z^{\omega}]$. As noted in Section 3, the use of multiple multicast channels does not change the sender processing costs. From [30] we know that the sender processing cost is greater than the receiver processing cost. Therefore, the protocol throughput in the *one–to–many* model, determined by the sender processing cost, does not change even when we use multiple multicast channels. However, in the case of *many–to–many* model, because each node acts like a receiver most of the time, the reduction in receiver processing cost obtained by using multiple multicast channels reduces overall processing cost at a node thereby increasing protocol throughput. Thus Figures 6, 7, and 8 also show that the P protocols achieve higher protocol throughput in comparison to the N protocols.

In summary, we observe a significant reduction of receiver processing costs by using multiple multicast channels. For the *many–to–many* application, we also see a substantial reduction in in overall processing costs at a node.

6 Finite Number of Retransmission Multicast Channels

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

We analyze the *one–to–many* model for protocol P1. The retransmission of packet i is now done on multicast address $A_{i \bmod G}$ where G is the number of retransmission multicast channels. That is, instead of subscribing to A_i, as in the previous section, a receiver needing to recover packet i subscribes to $A_{i \bmod G}$. Unlike the case of $G = \infty$, retransmissions of distinct packets may now be interleaved on the same multicast channel. This interleaving depends upon the retransmission rate of lost packets with respect to transmission rate 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 assume that the sender multicasts new packets periodically with a fixed time interval Δ, and retransmits a packet periodically with a fixed interval $\Delta^{'}$ as long as there is a pending NAK for that packet[2]. The value of Δ/Δ' depends upon many factors, such as application requirements, network topology, and network behavior. We will observe that Δ/Δ' is a key parameter in our protocol performance.

6.1 Analysis

Our goal in this section 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 channels by protocol P1.

In order to illustrate the factors that will impact performance, let us consider an infinite stream of packets. We pick one packet randomly and label it 0. Suppose that a receiver $r^{'}$ does not receive the original transmission of packet 0. On detecting that it has not received this packet, it joins a retransmission channel corresponding to address $A_{0 \bmod G} = A_0$. It sends a NAK to the sender and sets its NAK retransmission timer to $\Delta^{'}$. The sender retransmits packet 0 on A_0. If this packet is lost again then receiver $r^{'}$'s NAK retransmission timer expires and it retransmits the NAK. While $r^{'}$ is listening to channel A_0, other receivers may use the retransmission channel A_0 to recover packets $\ldots, -3G, -2G, -G, G, 2G, 3G, \ldots$. If $r^{'}$ has already received these other packets, then any retransmissions of these packets received by $r^{'}$ are unwanted.

We focus on the number of these unwanted packets received by $r^{'}$. 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 $r^{'}$ is using A_0 to receive packet 0.

[2]Realistically, both Δ and $\Delta^{'}$ are random variables.

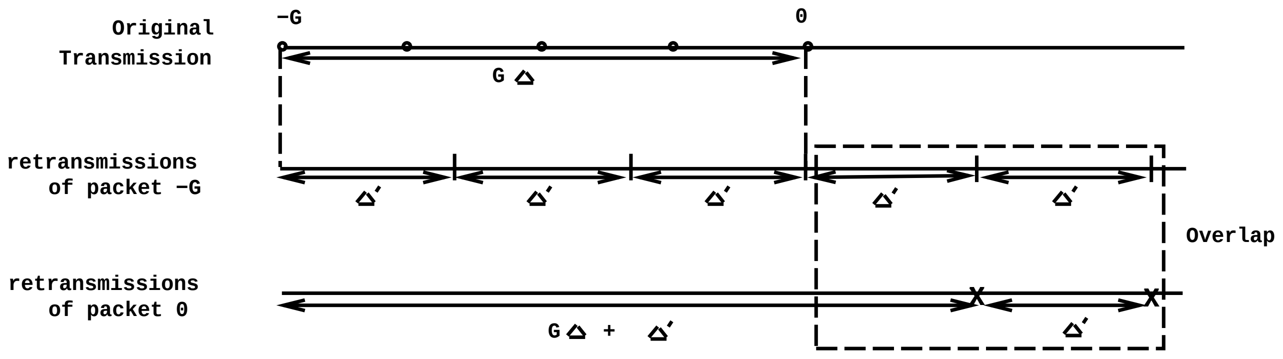

Figure 9: Sender timeline

Figure 9 shows the overlap between retransmissions of packets $-G$ and 0. For simplicity we have shown $G\Delta$ to be an integral multiple of $\Delta^{'}$ in Figure 9, although this is not a requirement for our analysis. We introduce the random variable U to denote the number of unwanted packets received by $r^{'}$ while it is attempting to receive packet 0, given that it requires at least one retransmission. Its expectation, $E[U]$, can be expressed as,

$$E[U] = (1-p)\left(\sum_{j=1}^{\infty}(E[Z(-jG)] + E[Z(jG)]) - \sum_{j=1}^{\infty}(E[Z^{'}(-jG)] + E[Z^{'}(jG))]\right) \tag{6.14}$$

where the random variable $Z(k)$, $k = \pm1, \pm2, \ldots$, denotes the number of retransmissions of packet k that overlap with the retransmissions of packet 0 to $r^{'}$ and the random variable $Z^{'}(k)$, $k = \pm1, \pm2, \ldots$, denotes the number of retransmissions of packet k that overlap with the retransmissions of packet 0 before $r^{'}$ leaves channel A_0. In the remainder of this section we derive expressions for $E[Z(k)]$ and $E[Z'(k)]$, $k = \pm1, \pm2, \ldots$. The notation used in the analysis is described in Table 3.

$Z(k)$	–	the number of retransmissions of packet k, due to R receivers, that overlap with the retransmissions of packet 0 to receiver $r^{'}$
$Z^{'}(k)$	–	the number of retransmissions of packet k, due to $r^{'}$, that overlap with the retransmissions of packet 0 to $r^{'}$
N	–	number of NAKs generated by receiver $r^{'}$ 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 channel
Δ	–	the time interval between successive transmissions from the sender
$\Delta^{'}$	–	the time interval between successive retransmissions from the sender
R	–	the total number of receivers

Table 3: Notation

For $k = -1, -2, -3, \ldots$, $Z(k)$ is determined by considering the minimum of the number of retransmissions of packet 0 and number of retransmissions of packet k, after r' 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' starts using A_0 to recover packet 0. Therefore,

$$Z(k) = \min(N, \max(0, \max_{1 \leq r \leq R}(N_r - \lfloor \frac{-k\Delta}{\Delta'} \rfloor))). \tag{6.15}$$

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' finishes recovering packet 0.

$$Z(k) = \min(\max(0, N - \lfloor \frac{k\Delta}{\Delta'} \rfloor), \max_{1 \leq r \leq R} N_r). \tag{6.16}$$

In equations (6.15) and (6.16), $P(N = n) = (1-p)p^{n-1}$ for $n \geq 1$ and $P(N_r = n) = (1-p)p^n$ for $n \geq 0$. The difference in $P(N = n)$ and $P(N_r = n)$ arises from the fact that N can only take values greater than or equal to 1 because we start from the assumption that r' 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 thus does not need a retransmission of k. These two probabilities can be used to derive the following two relations,

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

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

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

Based on these relations it is easy to derive

$$\begin{aligned} P(Z(k) \leq z) &= 1 - p^z \left(1 - (1 - p^{z + \lfloor -k\Delta/\Delta' \rfloor + 1})^R\right), \quad k = -1, -2, \ldots \\ E[Z(k)] &= 1 - (1 - p^{\lfloor -k\Delta/\Delta' \rfloor + 1})^R \\ &\quad + \sum_{z=1}^{\infty} p^z (1 - (1 - p^{z + \lfloor -k\Delta/\Delta' \rfloor + 1})^R), \quad k = -1, -2, \ldots \end{aligned} \tag{6.17}$$

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

$$E[Z(k)] = p^{\lfloor k\Delta/\Delta' \rfloor}(1 - (1-p)^R) + \sum_{z=1}^{\infty} p^{z + \lfloor k\Delta/\Delta' \rfloor}(1 - (1 - p^{z+1})^R), \quad k = 1, 2, \ldots \tag{6.18}$$

By setting $R = 1$ in (6.17) and (6.18) we obtain

$$E[Z'(k)] = \frac{p^{\lfloor |k|\Delta/\Delta' \rfloor + 1}}{1 - p^2}, \qquad k = \pm 1, \pm 2, \ldots \tag{6.19}$$

The expressions in (6.18) and (6.19) can be substituted into (6.14) to yield $E[U]$.

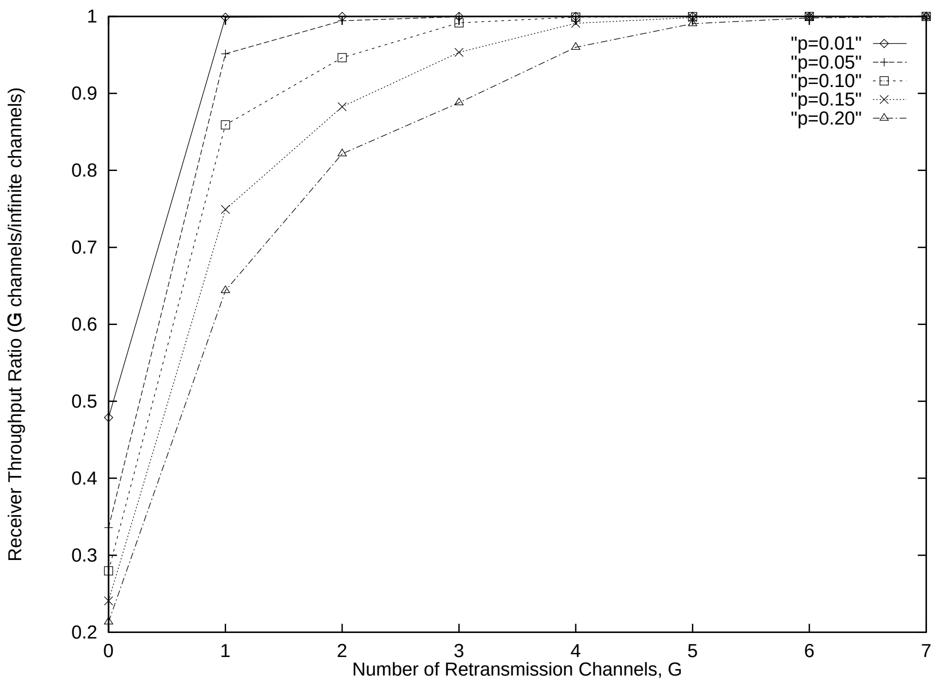

Figure 10: $\frac{\Delta}{\Delta'} = 1$ and $R = 1000$

6.2 Numerical Results

We now examine the number of retransmission channels required to achieve a receiver throughput "close" to the receiver throughput obtained by using an infinite number of retransmission channels. 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[3] for the case of G retransmission channels ($G \geq 0$) to that for the case of an infinite number of channels. 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 10 shows how γ varies with the number of retransmission channels, g, for several loss probabilities. In this figure $\Delta/\Delta' = 1$ and $R = 1000$. We find that as g increases, γ approaches 1 very fast. This is because with more multicast channels, the separation (in time) between recovery of packets mapped to the same retransmission channel becomes larger causing $E[U]$, the mean number of unwanted packets, to fall very sharply to zero. Let $G^*(\epsilon)$ be the minimum number of retransmission channels required to make $\gamma > \epsilon$. We find that $G^*(0.99) = 3$ when $p = 0.10$ and

[3]Receiver throughput is defined to be the inverse of receiver processing cost, $E[Y^{P1}]$.

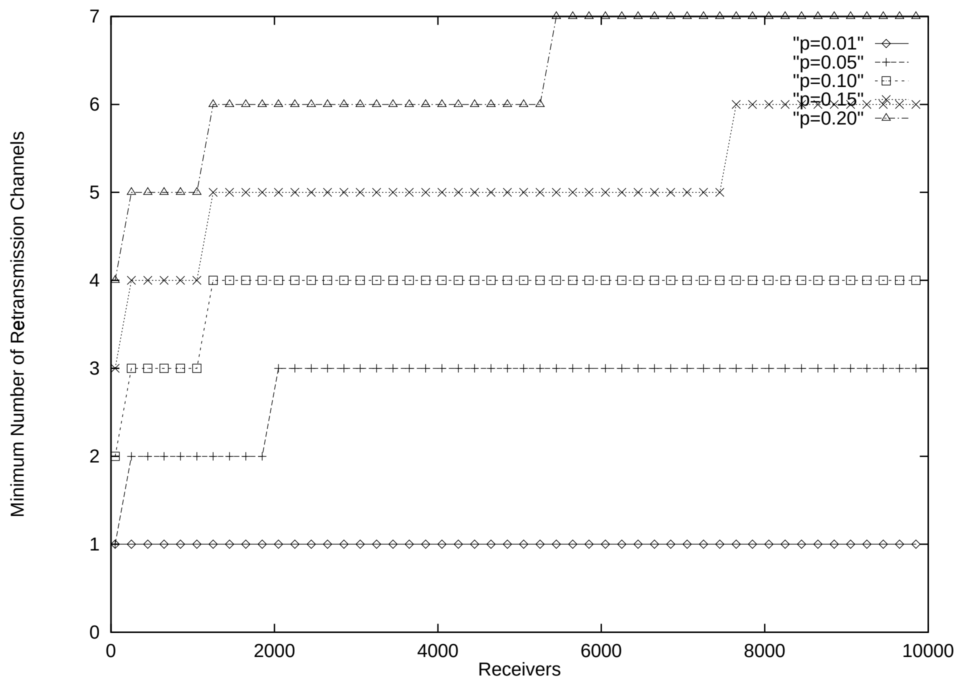

Figure 11: $G^*(0.99)$ vs R when $\frac{\Delta}{\Delta'} = 1$

$G^*(0.99) = 5$ when $p = 0.20$. Smaller values of epsilon reduce $G^*(\epsilon)$. We find that $G^*(0.90) = 2$ when $p = 0.10$ and $G^*(0.90) = 4$ when $p = 0.20$.

Figure 11 shows how $G^*(0.99)$ varies with the number of receivers for different loss probabilities, with $\Delta/\Delta^{'} = 1$. As expected, $G^*(0.99)$ is an increasing function of the number of receivers. We observe however, that this growth is very slow. For $p = 0.20$, $G^*(0.99) = 5$ when $R = 1000$, $G^*(0.99) = 6$ when $R = 5000$ and $G^*(0.99) = 7$ when $R = 10000$. On the other hand, as the number of receivers decreases, the minimum number of retransmission channels does not drop considerably. $G^*(0.99) \geq 4$ when $p = 0.20$ even for 50 receivers.

These results suggest that only a small number of channels are needed to achieve a receiver throughput that is close to the the receiver throughput with $G = \infty$ even when the loss probability is high and the number of receivers is large. These results were obtained when the ratio $\Delta/\Delta^{'}$ is one. Next we observe how this ratio affects G^*.

Figure 12 shows the behavior of $G^*(0.99)$ as a function of $\Delta/\Delta^{'}$, for several loss probabilities, when $R = 1000$. We see that G^* is very sensitive to small values of $\Delta/\Delta^{'}$. This is because the likelihood of overlap between retransmissions mapped to the same retransmission channel increases as $\Delta/\Delta^{'}$ decreases. This behavior diminishes as $\Delta/\Delta^{'}$ increases. In fact, beyond a certain value of $\Delta/\Delta^{'}$, G^* becomes insensitive to $\Delta/\Delta^{'}$, as seen in Figure 12. Figure 12 also shows that for a given $\Delta/\Delta^{'}$, G^* is higher for higher loss probabilities. This increase can be attributed to the fact that more retransmissions are required to recover lost packets, thereby increasing the chance of overlap between recovery of packets mapped to the same retransmission channel. With an increase

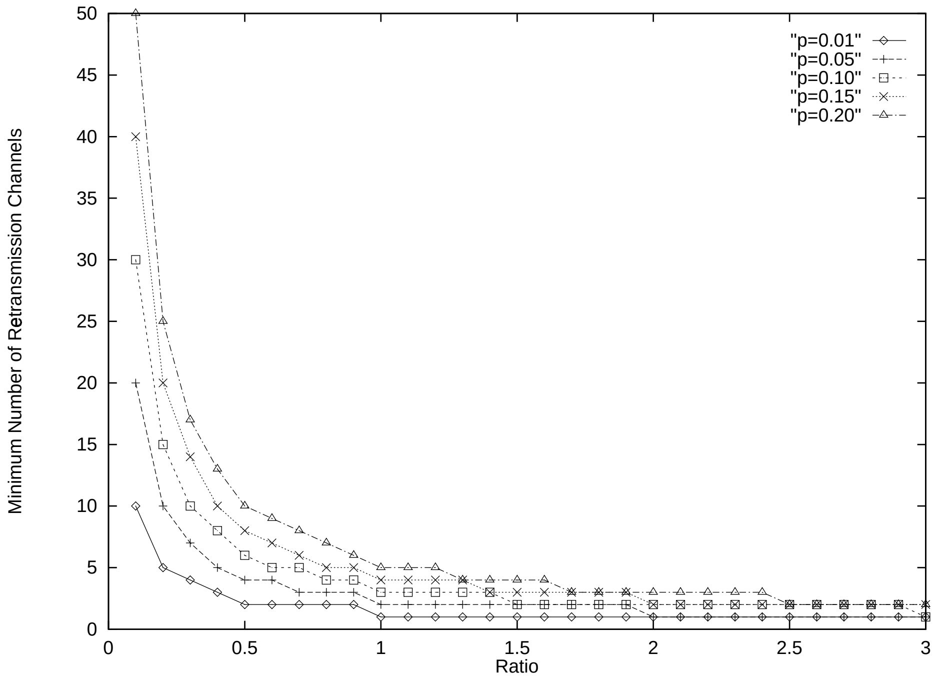

Figure 12: $G^*(0.99)$ vs $\frac{\Delta}{\Delta'}$ when $R = 1000$

in Δ/Δ', G^* becomes insensitive to p. This is because the retransmissions of different packets using the same channel are significantly separated.

In summary, we observe that one can achieve a throughput within 1% of the maximum achievable throughput using a very small number of multicast channels and that this holds for a wide range of system parameters. If we choose a less stringent requirement and can tolerate unwanted processing slightly greater than 1% of the ideal case, then even fewer channels are required.

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 channels for the *one–to–many* model. We now extend the analysis of P1 to the *many–to–many* scenario, where each node is a receiver as well as a sender. Hence, instead of a single sender, we now have multiple senders. As before, there are G retransmission channels in addition to the channel for original transmissions. On losing a packet i from sender j, a receiver joins the multicast channel $A_{i \mathrm{mod} G}$ and sends a NAK to sender j. Sender j retransmits the packet on $A_{i \mathrm{mod} G}$. After correctly recovering packet i from sender j the receiver leaves $A_{i \mathrm{mod} G}$.

For the convenience of analysis we model the multiple senders as a single global sender. That is, we assume that all of the transmissions and retransmissions originate from a fictitious global sender. We also assume that the global sender multicasts new packets periodically with a fixed time interval Δ, and retransmits a packet periodically with a fixed interval Δ' if there is a pending NAK for

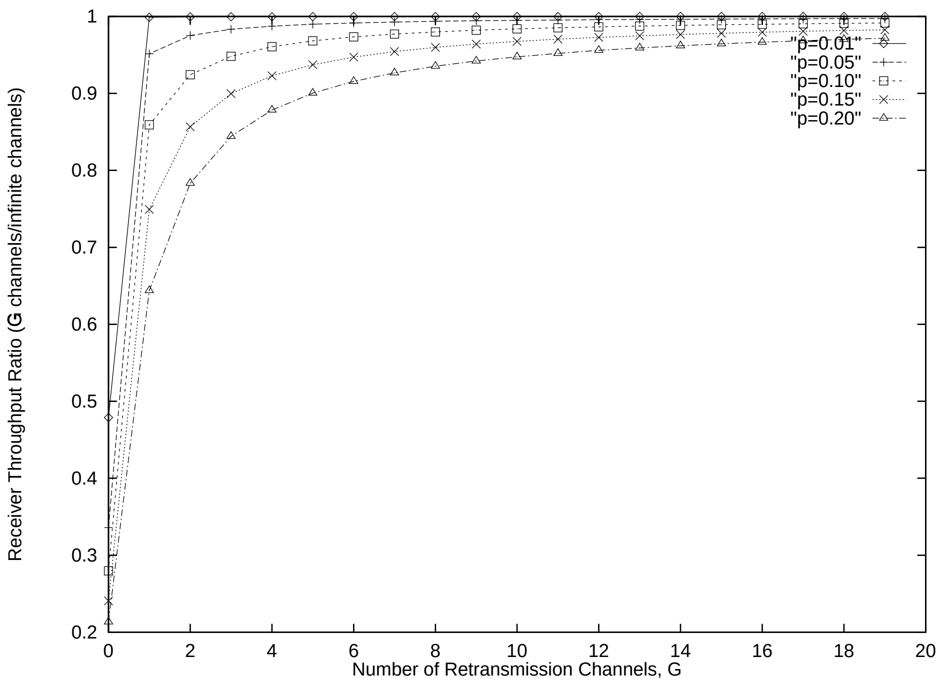

Figure 13: Many–to–Many Scenario, $\frac{\Delta}{\Delta'} = 1$ and $R = 1000$

that packet[4]. A stream of original packet transmissions 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 given to that packet by sender s_i.

Let us assume that the senders' traffic streams are mutually independent. Hence a packet in the combined packet stream from the global sender is equally likely to be mapped to any of the retransmission channels. In other words, the probability that a packet corresponding to a certain retransmission channel is $1/G$. As before, we randomly choose a packet from the global stream and call it 0. The expected number of unwanted packets received at a randomly chosen receiver, say r', when it tries to recover packet 0 can be expressed as

$$E[U] = (1-p)\left(\frac{1}{G}(\sum_{k=1}^{\infty}(E[Z(k)] + E[Z(-k)]) - \sum_{k=1}^{\infty}(E[Z'(k)] + E[Z'(-k)]))\right) \tag{6.20}$$

Using equations (6.17)–(6.20) we can compute the value of $E[U]$ for different values of G, p, R and Δ/Δ'. We compute γ, as defined in Section 6.2, for different values of G. Figure 13 shows how γ varies with G when $\Delta/\Delta' = 1$ and $R = 1000$. We observe that γ rapidly increases when G is small, but then approaches a horizontal asymptote of 1 as G increases. We define $G^*_m(\epsilon)$ to be the minimum number of retransmission channels required to keep $\gamma > \epsilon$. Due to the asymptotic behavior of γ,

[4]We have made this assumption for simplifying the analysis. Our goal is to obtain an estimate of the number of multicast channels required in the multiple sender case without complicating the analysis. The combined stream of packets from several senders is unlikely to have a constant inter–packet interval.

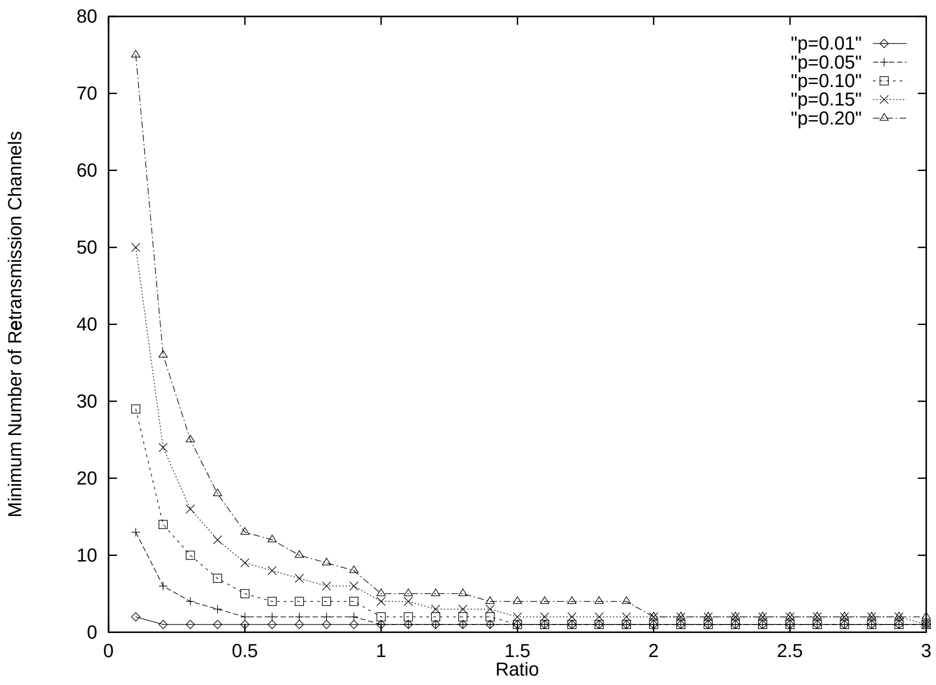

Figure 14: Many–to–Many Scenario, $G^*_m(0.90)$ vs $\frac{\Delta}{\Delta'}$ when $R = 1000$

choosing values of ϵ very close to 1 results in large $G^*_m(\epsilon)$. For example, $G^*_m(0.99) = 17$ when $R = 1000$ and $p = 0.10$. Choosing a somewhat smaller value of ϵ is more practical. For example, $G^*_m(0.90) = 2$ when $R = 1000$ and $p = 0.10$. This means that with 2 retransmission channels we could get 90% of the benefit that we would have obtained with an infinite number of retransmission channels.

We now study how the ratio Δ/Δ' affects $G^*_m(\epsilon)$. Figure 14 shows how $G^*_m(0.90)$ varies with Δ/Δ' for several loss probabilities when $R = 1000$. We observe from Figure 14 that $G^*_m(0.90)$ takes moderate to low values for a wide range of values of Δ/Δ'. Thus, even in the presence of multiple senders, we can obtain 90% of the benefit of infinite channels by using only a moderate or small number of channels. However, now the receivers must be prepared to tolerate a slightly larger number of unwanted packets than in the *one–to–many* case.

We end this section by observing that in a pathological case, all of the traffic streams from the senders might synchronize in transmitting packets such that packets with the same sequence number get bunched together. If $A_{i \bmod G}$ is the retransmission channel of packet i irrespective of the sender, then a bunch of $1s$ (or $2s$ or $3s$ ...), from different senders, would correspond to the same retransmission channel and there is likely to be a great deal of overlap in their recovery. This would 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 different rules for mapping packets to retransmission channels. An example of one such rule is that each sender uses the retransmission channels in a different order, i.e., if there are, say, 6 retransmission channels then sender 1 might use the retransmission channels in the order $[1, 4, 3, 0, 5, 2]$ and sender 2 might use a different order such as $[5, 3, 2, 0, 4, 1]$ and so on. This means that sender 1 retransmits packet 0 on channel 1, packet 2, on channel 4, packet 3 on channel 3 and so on. The receivers know the order of retransmission channel usage for each sender and hence join the appropriate channel, depending on the sender, for recovering packet i, rather than joining $A_{i \bmod G}$ for all the senders.

7 Implementation Mechanisms

We now look at mechanisms to implement scalable reliable multicast with end–to–end recovery using multiple multicast channels. The use of multiple retransmission channels reduces receiver processing costs. By allowing the forwarding of retransmissions to only those branches leading to receivers who need the retransmitted packets, network bandwidth consumption is also expected to reduce. The key implementation issue is how to efficiently manage the multiple channels. We consider two different implementation options, one using existing IP multicast mechanisms, the other using additional router support for selective packet forwarding [10].

7.1 Using IP Multicast Groups

One mechanism to implement the required multicast channels is to use multiple IP multicast groups. Here, each multicast channel is implemented as an IP multicast group, and joining and leaving a multicast channel corresponds to joining and leaving an IP multicast group (using the IGMP protocol [14]). The basic scheme is simple. Whenever a packet is lost, the receiver determines the retransmission group on which the lost packet will be retransmitted, sends a join request for that multicast group (unless it is already a member of that retransmission group), and then generates and sends a NAK to the sender. When all losses associated with a particular group have been successfully recovered, the receiver leaves the retransmission group (recall, due to the sharing of a finite number of groups to recover multiple packets the receiver may be waiting for multiple packets on the same group).

Even though the number of multicast groups is likely to be relatively small, there are potential concerns with using IP multicast groups. First, there are overheads for processing join and leave operations at routers, as receivers dynamically add and delete themselves from multicast groups (i.e., generation, forwarding and processing of join and leave signaling messages). Second, join and leave messages (reliable unicast transmissions between routers) use some additional bandwidth. In 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 (also called subnet router), through the IGMP protocol [14]. This information is then propagated to the nearest branching point of the multicast tree, rooted at the sender [7], or to a suitable core, in the case of core–based trees [2]. Each join and leave message results in processing overhead at all intermediate routers. The significance of this processing burden is highly topology dependent. In general, the closer the branching point is to a receiver, the lower the join or leave overhead on the network.

Another problem with using IP multicast is the high leave latency. When a receiver sends a leave request to its subnet router, the router usually waits for several seconds (3 second being a typical value) before it actually "leaves" the group and sends a leave message upstream. This means that, even after a receiver has left a multicast group, the subnet of the receiver continues to receive packets sent to that multicast group. Therefore, a leave operation at a receiver can only save receiver processing but not network bandwidth. When the join latency is high, it is also possible that a receiver will join the same multicast group for recovering a different packet even before the subnet multicast router leaves the group for an earlier packet. If this happens frequently, then, as far as the multicast router is concerned a receiver never leaves any retransmission multicast group and the network bandwidth consumption becomes the same as in the case of using a single multicast group for both transmissions and retransmissions.

High join latencies can also lead to inefficient performance. The path along which join messages traverse is slower than the path along which NAKs propagate towards the sender. If the join latency for a multicast group is high, the multicast route setup can take longer than the time required by the NAK to reach the sender and generate a retransmission from the sender. This could result in the loss of the retransmission.

In order to minimize the impact of signaling processing due to join and leave operations, and to be able to use IP multicast, 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 retransmission groups are always allowed to reach the local network to which the receiver is attached. All unwanted packets are then filtered out by the network interface hardware at 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, concerns only the receiver's local network interface. A local join or leave message, from the reliable multicast protocol layer, 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 non–locally joins both the original transmission and all of the retransmission multicast groups. This results in the transmission of IGMP messages to the nearest IP multicast router and, subsequently, the propagation of 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 performs a *local* join to the appropriate multicast group. Once the packet has been received correctly, the receiver performs a local leave. Hence, as far as the network routing tables are concerned the receiver is always a member of all retransmission 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 reduce network bandwidth consumption. At the same time it does not introduce any additional traffic in comparison to the scenarios that use a single multicast group.

Note that our local filtering scheme does not require any changes to 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[5]. Although it is best to perform the local filtering in the network interface hardware, benefits are also possible by implementing an efficient software packet filter. Finally, it should be noted that even with local filtering it is still necessary for the routers to set up routes and include entries in the multicast routing table for each of the multicast groups. We still need multiple IP multicast addresses to be allocated to each reliable multicast session.

7.2 Using Additional Router Support

Local filtering reduces the processing requirements at the receiver end–system by moving the filtering from the host's CPU to its network interface card. However as we have seen, it does not save on bandwidth. Moving filtering to a point inside the network does not adequately economize on network bandwidth either, because of the high leave latency associated with this move.

In order to reduce signaling overhead and speed up join/leave operations, we propose another approach that uses additional router support for selective packet forwarding. With this approach, all transmissions and retransmissions are sent to the same IP multicast group. Routers along the path from the receivers to the sender process NAKs sent from the receivers to the sender and maintain state for selective forwarding of subsequent data retransmissions from the sender[6] only on those links from which NAKs were received. Such a selective packet forwarding can be implemented by using the control–on–demand architecture proposed in [10]. The control–on–demand architecture supports router programmability but retains basic IP forwarding. While having the capability of acting in the data path (processing every data packet), control–on–demand also supports control plane programmability where the user installed program manages connectivity and router resources without interfering with data forwarding. Routers can be programmed to "snoop" retransmissions from the sender and, for attempting to stop the forwarding of the retransmissions on links that do not lead to receivers desiring the retransmission, based on the NAK state. The snooping mechanism is executed asynchronously of data forwarding and, hence, does not reduce data forwarding

[5]IP multicast addresses are mapped to Ethernet multicast addresses. Filtering is done based on Ethernet addresses and not IP multicast address. Such a filtering is desirable in the Ethernet hardware according to [13]. Unfortunately, many Ethernet cards do not provide appropriate multicast filtering.

[6]The reverse path for sending NAKs from the receivers to the sender through the routers that offer selective packet forwarding could be established by using Source path messages (SPMs) as proposed in [28].

performance. At the same time, snooping provides only a probabilistic filtering of undesired retransmissions. There is no guarantee that a retransmission could be halted from being forwarded even after it has been snooped.

In the ideal case, a retransmission is forwarded only on links leading to receivers that have requested the retransmission. This is equivalent to recovering every lost packet on a separate channel. Only one retransmission is sent on a branch for each NAK. After a retransmission corresponding to a NAK is forwarded downstream, the router removes all the state associated with that NAK so as to conserve network resources. This also avoids the need for an explicit receiver leave operation. An interesting consequence of the above data–driven management of NAK state is that it allows the network to exploit our mechanism "under the hood" transparently to the receivers. The receivers would see improved service quality as mostly only desired packets would be delivered.

Instead of using snooping, one could use *router alert* IP options in retransmissions (as suggested in PGM [28]) to get the attention of a router for selective forwarding of retransmissions. The problem in using IP options is that retransmissions with IP options move along a slower path instead of the fast data forwarding path, even though the filtering is ideal.

7.2.1 Performance Benefits

We have observed above that implementing multiple multicast channels using additional router support will improve performance by reducing bandwidth consumption. We now present some numerical examples to substantiate this observation. We will assume that routers perform ideal filtering.

Our bandwidth measure, termed *bandwidth consumption*, is defined to be the sum of the expected number of transmissions (including retransmissions) per successful transmission of a packet, along all of the links to all of the receivers. For simplicity, we assume that all receivers are identical and suffer losses with probability p. Bandwidth consumption is dependent upon the multicast tree topology. We consider three topologies as shown in Figure 15. In the first topology, a star, receivers have disjoint paths from the sender. In the second topology, all receivers share the same path from the sender. In the third topology, a modified star [34], a portion of the path is common to all receivers. The loss probability along the common path is denoted by p_S and the loss probability along each of the disjoint paths in the modified star is denoted by p_I, where $p = 1-(1-p_S)(1-p_I)$. The identical–path and the modified star topologies also model spatially correlated loss. In the identical–path topology the loss experienced by all the receivers is 100% correlated. The loss probability p_S represents the spatially correlated loss in the modified star topology. The modified star topology reduces to the identical–path topology when $p_S = p$, $p_I = 0$, and, to the star topology when $p_I = p$, $p_S = 0$.

We now determine the ratio of the bandwidth consumed by a router supported implementation of multiple multicast channels to the bandwidth consumed when only a single multicast channel is used for both transmissions and retransmissions. For the star topology the bandwidth consumption

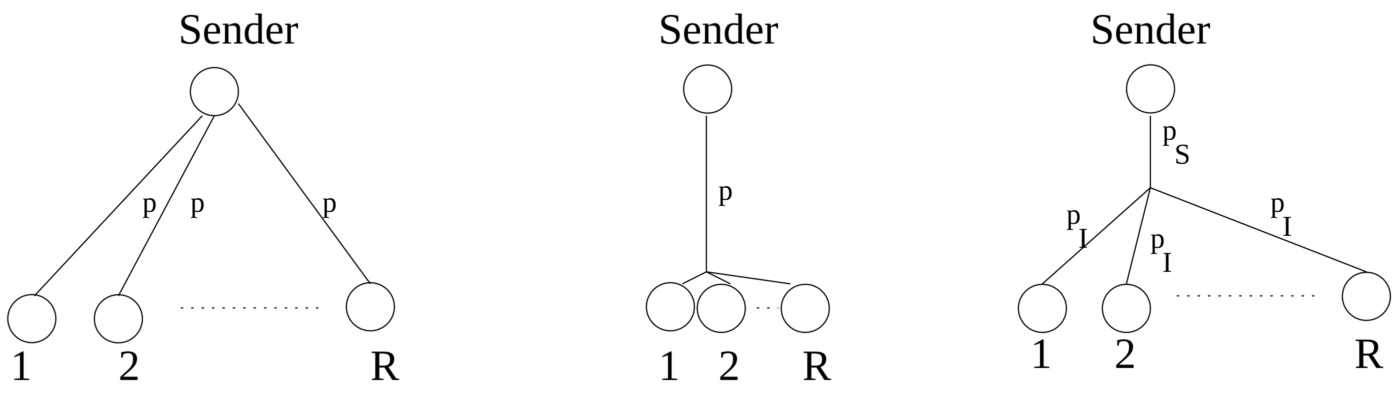

Figure 15: Topologies

ratio can be expressed as $E[M_r]/E[M]$, where $E[M_r]$ is the mean number of sender transmissions required for correctly transmitting a packet to one receiver and $E[M]$ is the mean number of sender transmissions required for all receivers to correctly receive a packet. From the expressions for $E[M_r]$ and $E[M]$ derived in Section 4, we have

$$E[M_r]/E[M] \quad = \quad 1/(1-p)(1+\sum_{m=1}^{\infty}(1-(1-p^m)^R))$$

For the identical–path topology, since all receivers have the same path from the sender, the bandwidth consumption ratio is $E[M_r]/E[M_r] = 1$. For the modified star, the bandwidth consumption ratio is expressed as

$$(E[M] + E[M_r^{'}]R)/(E[M] + E[M](1-p_I)R)$$

where $E[M_r^{'}] = 1/(1-p_I)$ is the mean number of transmissions required by a receiver to recover loss on one link (with loss probability p_I). Now, $E[M]$ can be expressed as follows:

$$E[M] = (1/(1-p_S))(1+\sum_{m=1}^{\infty}(1-(1-p_I^m)^R))$$

Figure 16 shows the reduction in bandwidth consumption due to the use of an infinite number of multicast channels over a single multicast channel as the number of receivers increases. In this figure, $p = 0.1$. The star topology and the identical–path topology are the two extreme cases. The reduction in bandwidth consumption for other topologies will lie in between these two. The reduction in bandwidth consumption of the modified star topology is shown as an example (here $p_S = p_I \approx 0.05$). We can analyze more complex topologies using the framework of [3]. We also note that the benefits of using multiple multicast channels reduces as the spatial correlation in loss increases. The same is also true for the savings in receiver processing costs. In the case when the loss is 100% spatially correlated no receivers will receive any unwanted packets even with a single multicast channel. It is possible to extend the analysis presented in Section 6 to show that fewer channels will be required to achieve the benefits of using infinite channels when some of the loss experienced at the receivers is spatially correlated.

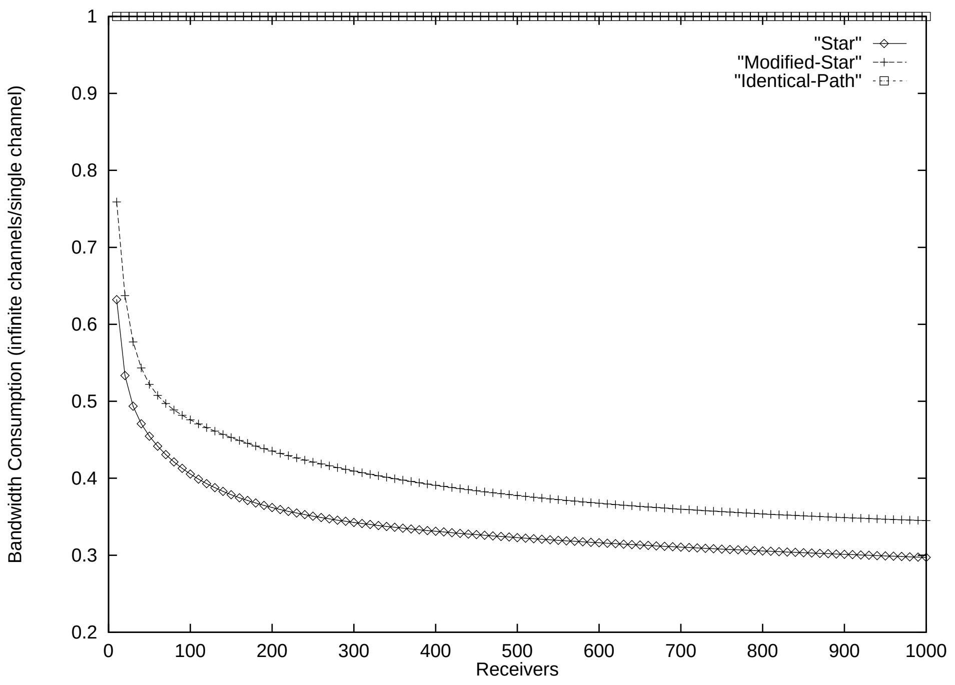

Figure 16: Bandwidth Consumption Reduction

In addition to reducing bandwidth consumption, the implementation of multiple multicast channels using additional router support further improves the end–host processing costs. The receiver processing cost is the one derived in Section 4 for infinite channels minus the join and leave processing costs, as no explicit join or leave operations are required. The NAK state at routers could also be used for eliminating duplicate NAKs for the same loss hence there is no need for explicit NAK suppression among receivers. This means that ideally, for each packet, the sender would receive only as many NAKs as the number of transmissions required by the sender to successfully transmit the packet to all receivers. Thus the sender processing costs under P1 reduces to that under P2.

7.2.2 Memory Requirement For Storing NAK State

We now estimate the memory required for maintaining NAK state at a router for a multicast session. We again assume that the sender multicasts new packets periodically with a fixed time interval Δ, and retransmits a packet periodically with a fixed interval Δ' as long as there is a pending NAK for that packet, as in Section 6. We focus on the memory requirements of a subnet router. Let there be B buffers at this subnet router, such that each buffer is capable of holding the state (sender address, multicast group address, packet sequence number, NAK sequence number required for NAK suppression [18], interface numbers on which retransmission needs to be forwarded, flags, etc.) associated with a single NAK. The probability of loss of a packet between the sender and the subnet router is denoted by p. Loss events are assumed to be independent. We assume that there are no losses between the subnet router and the receivers below it [34].

First–time–NAKs (not the retransmitted NAKS) arrive at intervals spaced apart by Δ/p at the subnet router[7]. The state associated with a NAK is kept from the time the NAK is first received by the subnet router until the time the packet corresponding to the NAK is forwarded downstream. Due to limited space, it is possible that no buffers will be available when a new NAK arrives. We use an *oldest first* buffer replacement policy in such a case, i.e., that state associated with a new NAK replaces replaces the state associated with the oldest NAK. Consider a random lost packet, i. Let q denote the probability of replacing the NAK state associated with i, even before i is received, by the NAK state associated with a new packet. The maximum number of retransmissions of i before the NAK state for i is overwritten is equal to $\lfloor B\Delta/\Delta' \rfloor$. Let the random variable N denote the number of retransmissions of i from the sender such that it is correctly received by the subnet router. Then q is equal to the probability of N exceeding $\lfloor B\Delta/\Delta' \rfloor$. Therefore,

$$\begin{aligned} q &= P(N > \lfloor B(\Delta/\Delta')/p \rfloor) \\ &= p^{\lfloor B(\Delta/\Delta')/p \rfloor} \end{aligned}$$

We now determine the minimum number of NAK state buffers required to ensure $q \leq 0.001$. Figure 17 shows how the minimum NAK state buffer requirement varies with the ratio Δ/Δ' for different values of p. We observe that the required number of NAK state buffers increases in p and decreases in Δ/Δ'. Fewer than 10 buffers are required to ensure $q \leq 0.001$ for a wide range of loss probabilities and ratios Δ/Δ'. Therefore, the subnet router needs atmost 10 buffers to maintain NAK state per session. Considering that the state associated with a NAK comprises of sender address (4 bytes), multicast group address (4bytes), packet sequence number (4 bytes), NAK sequence number (1 byte) required for NAK suppression (see [18]) and another 7 bytes for interface numbers (on which the retransmission needs to be forwarded), flags and any other fields, a buffer sufficient to store the state of 10 NAKs amounts to approximately 200 bytes per multicast session. For small loss probabilities or high values of Δ/Δ', this memory requirement is much less.

On the rare occasion that the NAK state for a packet at a subnet router is overwritten, a retransmission is not forwarded downstream. The downstream receiver will eventually timeout and retransmit the NAK which will create a new NAK state at the subnet router.

In this section we have considered NAK state buffer requirement at subnet routers. The NAK state required by routers inside the network per multicast session is likely to be of the same order, though the number of multicast sessions a router must support is larger.

8 Conclusions

In this paper we have examined an approach for providing reliable, scalable multicast communication by using multiple multicast channels for recovery of lost packets. In this approach, rather than having all receivers receive all retransmitted packets (regardless of whether a given receiver

[7]Realistically, NAKs would arrive at random time intervals.

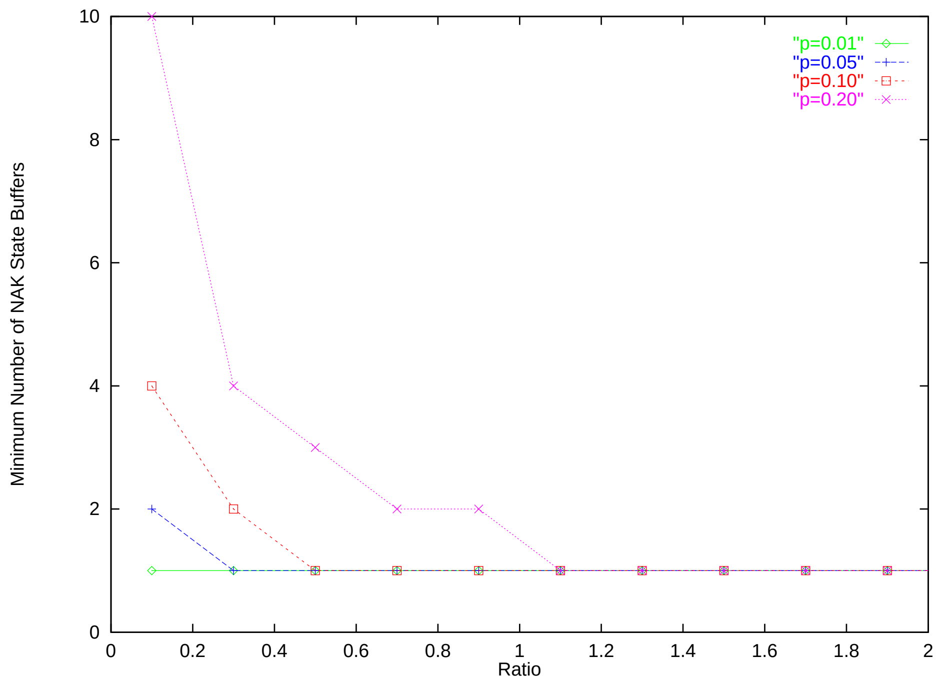

Figure 17: NAK State Buffer Requirement

had already received a given packet correctly), the multiple multicast channels 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–to–many* scenario and the *many–to–many* scenario. Our analytic models have demonstrated that significant performance gains (in terms of reduced receiver overhead, and a reduced overall protocol overhead in the case of *many–to–many* communication) can be realized in such environments, over a range of system parameters.

We discussed two mechanisms for implementing multiple multicast channels, one using multiple IP multicast groups and the other using additional router support for selective packet forwarding. With the current IP multicast model, we obtain savings in receiver processing through our local filtering scheme but cannot save on network bandwidth whether or not we move the filtering point inside the network. However, with the second approach, by using some processing at routers and a small buffer space (up to 200 bytes per multicast session), we could save on both sender and receiver processing costs as well as network bandwidth.

Our work can proceed in the following directions. Our analyses assume that loss events are temporally independent. In [34] and [35] Yajnik *et al.* have observed temporal correlation in losses on the Internet. In the future, we plan to study the impact of temporal correlation in loss, especially in the context of reusing a finite number of multicast channels. We have considered a *round–robin* approach in reusing the multiple retransmission channels; other approaches should also be investigated.

It is possible to combine our work with existing work on local recovery. Local recovery helps in isolating the domains of loss, thereby reducing global retransmissions and recovery latencies. If there are several local domains separated by a wide area network, and the domains see a sufficient amount of independent loss, then multiple multicast channels could be used to deliver packet reliably to the domains. If the domains themselves have a large number of receivers experiencing high intra–domain losses then multiple multicast channels could also be used for reliable multicast inside these domains.

In order to be able to use IP multicast for implementing multiple multicast channels we need to clearly understand the processing overheads of join and leave signaling on the routers, determine actual values of join and leave latencies (through actual measurements) and find approaches for reducing leave latency [29]. With regard to the approach that uses additional router support for selective packet forwarding, we need to model the processing costs at routers. We also need to study the impact of multiple multicast sessions, using a router at the same time, in terms of its processing and buffering resource requirements.

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 Conference, 1993.

[3] P. Bhagawat, P.P. Mishra and S.K. Tripathi, *Effect of Topology in Performance of Reliable Multicast Communication.* Proceedings of IEEE Infocom, June 1994.

[4] D. Cheriton, End–to–End mailing list, September 1994.

[5] 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.

[6] J. Crowcroft, End–to–End mailing list, September 1994.

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

[8] 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.

[9] S. Floyd, V. Jacobson, S. McCanne, C. Liu and L. Zhang, *A Reliable Multicast Framework for Light–weight Sessions and Application Level Framing.* IEEE/ACM Transactions on Networking, Vol.5, No.6, pages 784–803, December 1997.

[10] G. Hjálmtýsson and S. Bhattacharjee, *Control–on–Demand.* AT&T Technical Memorandum, 1997.

[11] M. Hofmann, *Enabling Group Communication in Global Networks.* Proceedings of Global Networking'97, Calgary, Alberta, Canada, November 1996.

[12] 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.

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

[14] W. Fenner *Internet Group Management Protocol, Version 2.* RFC 2236, November 1997.

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

[16] Sneha K. Kasera, J. Kurose and D. Towsley, *Scalable Reliable Multicast Using Multiple Multicast Groups.* Proceedings of ACM Sigmetrics Conference, June 1997.

[17] Sneha K. Kasera, J. Kurose and D. Towsley, *A Comparison of Server–Based and Receiver–Based Local Recovery Approaches for Scalable Reliable Multicast.* Proceedings of IEEE Infocom, March 1998.

[18] L. Lehman, S. Garland and D. Tennenhouse, *Active Reliable Multicast.* Proceedings of IEEE Infocom, March 1998.

[19] B.N. Levine and J. Garcia–Luna–Aceves, *A Comparison of Known Classes of Reliable Multicast Protocols.* Proceedings of IEEE ICCC, November 1996.

[20] B.N. Levine, D.B. Lavo and J. Garcia–Luna–Aceves, *The Case for Reliable Concurrent Multicasting Using Shared Ack Trees.* Proceedings of ACM Multimedia, November 1996.

[21] B.N. Levine and J. Garcia–Luna-Aceves, *Improving Internet Multicast with Routing Labels.* Proceedings of ICNP, October 1997.

[22] J. Lin and S. Paul, RMTP: A Reliable Multicast Transport Protocol. Proceedings of IEEE Infocom, 1995.

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

[24] J. Nonnenmacher, E. Biersack and D. Towsley, *Parity–Based Loss Recovery for Reliable Multicast Transmission.* IEEE/ACM Transactions on Networking, Vol.6, No.4, pages 349–361, August 1998.

[25] C. Partridge and S. Pink, *A Faster UDP.* IEEE/ACM Transactions in Networking, Vol.1, No.4, pages 429–439, August 1993.

[26] C. Papadopoulos, G. Parulkar and G. Varghese, An Error Control Scheme for Large–Scale Multicast Applications. Proceedings of IEEE Infocom, March 1998.

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

[28] T. Speakman, D. Farinacci, S. Lin and A. Tweedly, *Pragmatic General Multicast.* Internet Draft, August 1998.

[29] L. Rizzo, *Fast Group Management in IGMP.* In HIPPARCH98 Workshop, June 1998.

[30] D. Towsley, J. Kurose and S. Pingali, *A Comparison of Sender–Initiated and Receiver–Initiated Reliable Multicast Protocols.* IEEE Journal of Selected Areas in Communications, April 1997.

[31] D. Towsley, *An Analysis of a Point–to–Multipoint Channel Using a Go–Back–N Error Control Protocol.* IEEE Transactions on Communications, 33:282–285, March 1985.

[32] L. Vicisano, L. Rizzo and J. Crowcroft, *TCP–like Congestion Control for Layered Multicast Data Transfer.* Proceedings of IEEE Infocom, March 1998.

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

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

[35] M. Yajnik, Sue Moon, J. Kurose and D. Towsley, *Measurement and Modeling of the Temporal Dependence in Packet Loss.* To appear in IEEE Infocom 1999.