2003

Recovery Performance of PGM with Redundancy Estimation for Proactive FEC

Sonia Fahmy
Purdue University, fahmy@cs.purdue.edu

Renin M. Jegadeesan

Report Number:
03-019

RECOVERY PERFORMANCE OF PGM WITH REDUNDANCY ESTIMATION FOR PROACTIVE FEC

Sonia Fahmy
Renin M. Jegadeesan

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD TR #03-019
June 2003

Recovery Performance of PGM with Redundancy Estimation for Proactive FEC

Sonia Fahmy and Renin M. Jegadeesan

Abstract— **We study the recovery performance of the Pragmatic General Multicast (PGM) reliable multicast protocol using adaptive proactive forward error correction (FEC). We compare the performance of PGM without FEC, PGM with static proactive FEC, PGM with previously proposed redundancy estimation algorithms, and PGM with a new redundancy estimation technique which we call smooth increase/decrease (smooth-ID). Our technique periodically estimates whether the network is becoming more or less congested/error-prone based on loss rates, and determines the amount of redundancy information to include based upon this estimate. Temporary fluctuations of the loss rates do not cause drastic increases or decreases of the amount of redundancy. We evaluate the performance with different group memberships, different loss models and congested links, and different transmission group (TG) sizes. Our results show that PGM with our redundancy estimation technique performs best in terms of high recovery percentage and low overhead. We also find that the PGM transmit window size and advance frequency are key parameters, and affect the choice of transmission group size.**

Keywords— **Forward Error Correction (FEC), proactive FEC, Pragmatic General Multicast (PGM), reliable multicast, multicasting, error recovery**

I. Introduction

MULTICAST has become an important component of the Internet within the past decade. There is immense IETF standards activity in areas such as multicast routing, address allocation, naming and deployment, and multicast transport protocols. Fundamentally new multicast models are also being discussed to solve address allocation and access control issues.

With the expected growth of multicast group memberships, providing reliable multicast (RM) services becomes a challenge. One of the main problems is feedback implosion when many receivers try to request repairs, thus flooding the network elements and the sender. Many RM protocols today solve this problem using suppression strategies. Random delays are employed when sending negative acknowledgments (NAKs), which are locally multicast, so that other receivers do not send a separate NAK for the same packet.

The authors can be reached via e-mail: fahmy@cs.purdue.edu, Address: Purdue University, 250 N. University St., West Lafayette, IN 47907–2066

As many of the multicast applications are real-time, recovery latency is a significant factor in RM protocols. Real-time applications need packets and repairs within a certain time frame. Outside that time frame, the packets may be useless to the application. When the round trip time (RTT) is large, the time taken for the NAK to be sent to the sender, and the repair packet sent by the sender to arrive at the receiver application may be too long for the receiver application which discards the repair. In addition, the overhead imposed on the sender for the error recovery is significant. As the number of retransmissions the sender has to perform increases, the overhead on the sender increases, thus affecting its performance.

Forward Error Correction (FEC) can improve the repair performance of multicast sessions. FEC does not face Internet deployment barriers like router assist or server assist approaches. Error correction in RM protocols is typically achieved by Automatic Repair reQuest (ARQ), where the receivers send a NAK for a particular packet, and the sender retransmits the packet. But ARQ suffers from the implosion, latency and overhead problems previously discussed. Using FEC in addition to ARQ, we can address these issues quite effectively. The amount of FEC information to add to a transmission can be estimated based upon the state of the network and its links. We propose a smooth increase/decrease algorithm (smooth-ID) that performs well in our experiments. The key idea of the algorithm is to periodically estimate whether the loss rate is increasing or decreasing, and increase or decrease the redundancy estimate accordingly. The main advantage of the algorithm is that it adapts well to loss rate fluctuations.

We study the reliable multicast protocol, Pragmatic General Multicast (PGM), with proactive FEC with a number of redundancy estimation techniques. The paper is organized as follows. We first explain reliable multicast, the PGM RM protocol and FEC operation. Then, we discuss our proposed adaptive proactive FEC approach. We evaluate the performance of a number of FEC parameters using a number of simulation experiments and performance metrics. Finally, we analyze the significance of the results of the simulation, and conclude with a discussion of future work.

II. Reliable Multicast Transport Protocols

Reliable multicast protocol design has been an active research area since group management and multicast routing support (IP multicast) were incorporated [4]. Various protocols have been designed and implemented for Mbone applications. The scalable reliable multicast (SRM) protocol [6] was one of the earliest. Other popular protocols include the Reliable Multicast Transport Protocol (RMTP) with its recent extensions to RMTP-II [29]. Obraczka [17] provides an excellent summary and taxonomy of the various RM protocols.

The general model of reliable multicast transport protocols is to provide a combination of *temporal redundancy* and *spatial redundancy*. Temporal redundancy (or soft state) provides redundancy over time by sending retransmission or repair traffic based upon receipt reports from receivers. Spatial redundancy, on the other hand, provides redundancy within the original transmission, typically in the form of proactive forward error correction (FEC). In general, the two key problems in RM are:

[A] Implosion of control traffic on the reverse direction from multiple receivers to the sender. The proposed approaches to this problem include: (1) Scoped multicasting of reverse control traffic combined with probabilistic suppression techniques at the receivers [6]; (2) Use of negative acknowledgments (NAKs) or bitmaps instead of simple acknowledgments (ACKs) [25], [29], [6]; (3) Explicit aggregation, e.g., generic router-assist [25], [29]; (4) Scheduled selection of receivers which would send control traffic; and (5) Scaling the feedback frequency based upon group size.

[B] Optimization of repair traffic and its required bandwidth resources. The proposed approaches to this problem include: (1) Use of local retransmitters (e.g., Designated Local Repairers (DLRs) in PGM and Designated Receivers (DRs) in RMTP [25], [29]), or allowing receivers themselves to transmit repairs [6]. Such retransmission must necessarily be scoped to the subtree (e.g., TTL and administrative scoping); (2) Router-assist to form a separate subtree per-packet to be retransmitted, and constrain retransmission to flow on this subtree [25]; (3) Forward error correction (FEC) that generates redundant parity packets [19], [21]. The sender can reactively construct FEC knowing what *fraction* of packets reached the destination, without dependency on which particular packets have been lost. Proactive "spatial redundancy" is also attractive [15], [19]. We focus on proactive FEC in this paper.

III. Pragmatic General Multicast (PGM)

The Pragmatic General Multicast (PGM) is one of the most recently designed reliable multicast protocols. PGM can be used with applications requiring ordered or unordered, duplicate-free, multicast data delivery from multiple sources to multiple receivers [25]. PGM is specifically intended for applications with basic reliability requirements.

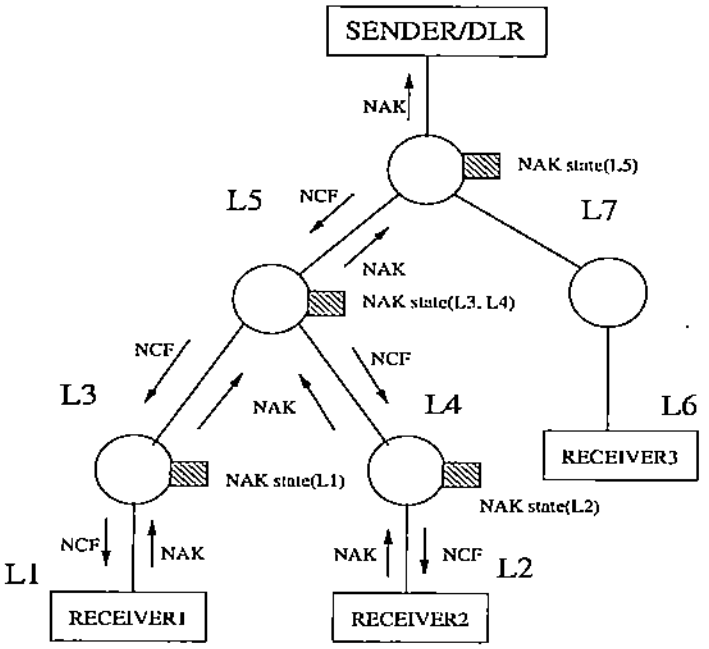


Fig. 1. Recovery in the PGM protocol

In PGM, a source multicasts sequenced original data packets (ODATA), and the receivers unicast selective NAKs for data packets detected to be missing from gaps in the sequence numbers. The NAKs are sent to the last hop PGM network element on the reverse data path established by Source Path Messages (SPM). The PGM network element multicasts a NAK confirmation (NCF) on the interface from which it received the NAK (on that local LAN), and forwards the NAK upstream. The nodes wait for a random delay before sending a NAK. When the receivers receive an NCF for a packet they have lost, they refrain from sending the NAK for that packet. Since the PGM network element sends only one copy of the NAK upstream, the implosion is effectively reduced (figure 1). The NAKs are sent to the sender or a designated local repairer (DLR), PGM-hop-by-PGM-hop, and the sender or DLR retransmits the lost packet if it is within its *transmit window*. The main aim of the *transmit window* is to limit the number of packets the sender or DLR needs to store and retransmit upon request. The transmit window is periodically advanced. At each PGM network element, a NAK state is stored when it receives a NAK, as shown in

figure 1. This state contains information about the interfaces the NAKs came on. When the retransmission comes from the sender, the packets are multicast *only* to those interfaces from which NAKs had been received. This significantly reduces the duplicate packets seen by the receivers.

In addition to the basic data transfer operation, PGM provides many options to address specific application requirements. These options include fragmentation to allow a transport layer entity at the source to break up application layer data into multiple PGM packets; late joining for the source to indicate whether a receiver can request all the available repairs when it initially joins the group; timestamps for the receivers to specify the interval in which the RDATA is relevant to them, redirection to allow a designated local repairer (DLR) to receive the NAKs instead of the sender; and FEC to be used under a layer of ARQ, as discussed next.

IV. Forward Error Correction (FEC)

FEC involves the transmission of additional redundant data which can be used to reconstruct the original data if some of it is lost [3], [10], [11], [8], [26]. The construction of the redundant data (or erasure correction codes) is based on the principles of linear algebra. A Reed-Solomon erasure (RSE) correcting code is a popular technique for generating the redundant data. The original data packets are divided into groups of packets, called the Transmission Groups (TGs). Suppose each TG contains k packets, each of which is P bits long. The RSE encoder takes these k data packets, and generates h redundant packets ($h \geq 0$) each P bits long as well. The receiver can decode the entire data contained in the k original packets if it receives any k of the $(k + h)$ packets, as shown in figure 2.

FEC offers many advantages in the reliable multicast scenario. First, it reduces the number of retransmissions the sender needs to perform. This is because one parity h packet can be used to replace any one of the k packets in the original TG. Thus if receiver A loses packet i, and receiver B loses packet j, and if both i and j are in the same TG, then the sender needs to retransmit only one parity packet, whereas without FEC, 2 retransmissions are required. Second, FEC improves scalability, as the sender need not remember all the data packets any more. It suffices for the sender to remember only the parity packets for each TG. Third, FEC reduces the number of NAKs sent by the receivers, since now, they can send NAKs for each TG (stating how many packets they lost in that TG), and not for each lost packet.

Several flavors of FEC are currently being used for error recovery. With *on-demand* FEC, the original k packets are transmitted first, and only if any repair requests are received for that TG, the parity packets are generated and transmitted to the receivers. This has no bandwidth overhead and does not introduce duplicate packets. However, if the multicast applications are real-time and the round-trip latency is too large, then the error recovery performance will degrade with this approach.

With *proactive* FEC, the redundant data packets are generated at the time of original transmission itself, and the k original data packets are followed by the h parity packets. This significantly reduces the recovery latency and repair requests and responses. However, h extra packets are transmitted for every k packets, which increases bandwidth requirements. In addition, if the loss rate is very low in the network, the receivers may receive many redundant packets. Our redundancy estimation technique attempts to remedy these problems.

V. Redundancy Estimation for Proactive FEC

The key idea we investigate is a redundancy estimation algorithm to determine the amount of proactive FEC h to send with the k data packets of a transmission group. The parameter h is varied depending on the congestion and error state of the network. We start off with a standard value for h (say $\frac{k}{2}$ or $\frac{k}{4}$), and we send the parity packets proactively along with the original data. If the receiver still cannot retrieve the required k packets in a TG, it sends a NAK, and the sender retransmits (ARQ).

We periodically compare the losses denoted by the NAKs received by the sender over a period of time to the previous loss rate to determine if we are entering or leaving a congestion or high error phase. The value of h is increased if the current congestion or error estimate is higher than the previous value, and reduced h if the current estimate is significantly lower. This reduces the network bandwidth requirements for FEC and the duplicate packets received by the receivers in case of low loss, while maintaining proactive FEC recovery benefits including low latency, when needed.

Note that the algorithm works within the congestion control framework applied to the RM protocol. Thus the total bandwidth used by the RM algorithm is reduced in periods of congestion or high error and increased during underload. An example congestion control protocols that have been tested with PGM are pgmcc [20] and SBMCC [14].

The redundancy estimation algorithm can be designed in several ways. The main research issues are:

1. Determining when to invoke the algorithm to update h, e.g., with the receipt of NAKs or at periodic intervals which may depend on the RTTs. This can be tightly coupled to the congestion control algorithm, where the h up-

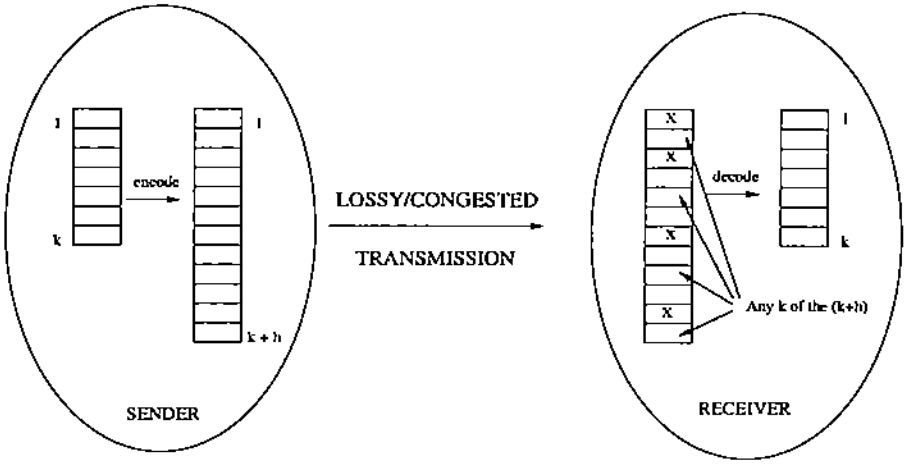


Fig. 2. Forward error correction (FEC) operation

date and rate/window update can be performed when the same event triggers;

2. Determining the start of a congestion/high error phase or the end of a congestion/high error phase, e.g., by comparing the congestion estimate denoted by losses in NAKs to its previous value/average or comparing it to thresholds;

3. Determining how to increase h, e.g., by the number of losses or by a fixed parameter;

4. Determining how to decrease/decay h, e.g., halve h or use EWMA filters or fixed parameters;

An alternative approach can use the loss ratio (as a percentage of packets sent) as a factor to scale h between upper and lower bounds.

The pseudo-code for the redundancy estimation algorithm we used for adaptive proactive FEC is given in table I. The algorithm uses smooth increase and decrease (hence we call it smooth-ID), and it is timer-triggered (as opposed to NAK-triggered). It determines whether to increase or decrease h based upon a comparison of the loss rate with the previous exponentially weighted average loss rate. We use a threshold of 50% to determine that the loss rate has significantly decreased, instead of waiting till the number of losses in a group is zero, as used in [8]. We give equal weights to the current and previous loss rate samples in our simulations (i.e., $\alpha = \frac{1}{2}$). The algorithm does not require loss rate thresholds, which need to be configured, but rather updates h based on the *direction of the slope* of the loss rate graph over time. The increase and decrease amounts, *Incr* and *Decr* in the pseudo-code, are computed as a function of the transmission group size k. They can also be a function the reported loss rate. *Decr* may be a function of the current h value, resulting in a multiplicative decrease. We used simple fractions of the TG size k in our simulations, and leave the evaluation of multiplicative decrease, and loss count-sensitive increments to future work.

Our algorithm differs from earlier schemes proposed by Kermode [7] and Li and Cheriton [8] in its invocation and congestion estimation technique, and its redundancy increase/decrease algorithm. Their approaches increased h by the number of losses reported in every NAK, and decreased it by half if no losses are reported for a transmission group. We believe the approach presented above is more stable since we limit the oscillations of the h value. Temporary fluctuations of the loss rates do not cause drastic increases or decreases of h. The longer term behavior and trends are given more importance. The main aim of our study is to investigate the error recovery performance of PGM with redundancy estimation in different configurations, parameters and metrics, since PGM was not addressed in the earlier work.

VI. Performance Metrics

This section summarizes the performance metrics we studied in our experiments. These include:

1. *Average number of packets recovered at the receivers (as a percentage of number of packets lost):* The main objective of a reliable multicast protocol is to make sure that the receivers receive most, if not all, of the packets transmitted by the sender. The efficiency of an error recovery protocol can thus be effectively measured as the ratio of the number of packets recovered to the number of packets lost averaged on all receivers. The closer this ratio gets to 1, the better the error recovery performance is.

2. *Number of NAKs processed by the sender (as a percentage of total packets sent):* An important error recovery concern is the overhead imposed on the sender or designated local repairers for taking part in the recovery. This overhead can be directly represented by the number of retransmissions that the sender has to perform in order to provide the receivers with recovery data. This is reflected by the number of NAKs the sender receives from the re-

// A timer is set to expire after a particular interval of time (call this interval t_{FEC}). The h value is updated every t_{FEC}.
// A counter ($count_{loss}$) counts the number of losses in NAKs received by the sender. This counter is reset to zero every time the timer triggers. Thus the counter indicates the number of packets NAKed during that interval. $count_{packets}$ is the number of packets transmitted during the interval for which the NAKs are received. The ratio of the 2 gives the loss rate.
// We maintain a variable called $lastRate_{loss}$, where we maintain the loss rate of the last t_{FEC}.
// We maintain a variable called $h_{current}$, where we store the current value of the FEC block size. We increment or decrement this value based on the comparisons that we perform.

// When the timer triggers:
if (($count_{loss}$ / $count_{packets}$) > $lastRate_{loss}$)
    $h_{current}+ =$ Incr;
else if (($count_{loss}$ / $count_{packets}$) << $lastRate_{loss}$)
    $h_{current}- =$ Decr;
if ($h_{current} < 0$)
    $h_{current} = 0$;
$lastRate_{loss} = \alpha \times lastRate_{loss} + (1 - \alpha) \times count_{loss}/count_{packets}$;
$count_{loss} = 0$;
Reschedule the timer to trigger after t_{FEC}

TABLE I
PSEUDO-CODE FOR ADAPTING THE h PARAMETER (SMOOTH-1D)

ceivers. Even with NAK suppression and random backoff, the number of NAKs received by the sender can be high, justifying the need to study and improve this metric.
3. *Average number of duplicate packets at the receivers (as a percentage of total packets sent):* If a receiver that did not lose packets receives recovery or parity packets, network bandwidth is wasted and receivers unnecessarily need to filter these duplicate packets. Thus the number of duplicate packets should be minimized.

The first of the three metrics evaluates the recovery performance of the algorithm, and is thus the most important. The second and third metrics evaluate recovery and FEC overhead respectively.

VII. Simulation Experiments

The network simulator ns-2.1b6 [27] was used to simulate a number of network topologies, and extract measurements of the above metrics for the different protocols under study. The network animator (nam) was also helpful in debugging and interpreting the simulation results. We ported a PGM implementation by the Washington University at St Louis group [24] to new ns release, and added FEC support to it as explained in the appendix of [25].

A. Simulation Setup

We first conducted 3 experiments, representing increasing loss and congestion levels and also an increased number of receivers. The first 2 experiments have low loss, so they allow us to see how the redundancy estimation decreases the FEC information and overhead, while the third experiment represents congestion, so it shows the increase of redundancy information to improve recovery.

The first experiment used a topology of 16 receivers representing a sparse membership scenario. There is no network congestion, and packet loss is only a small percentage on most of the links. The tree is a 6-level symmetric one: all receivers are 6 hops away from the sender (figure 3 but with only half the receivers at the lowest tree level). The link bandwidths are 1.5 Mbps and the link delays are 10 ms (thus the maximum RTT is 120 ms) to simulate a wide-area network scenario. The PGM rate is around 1 Mbps. Packet sizes of 1024 bytes were used in all simulations. Simulation time was around 6 seconds. The PGM transmit window was set to 2 to 3 times the maximum RTT and the advance interval was set to half the window. This places a severe constraint that packets cannot be recovered late.

The second experiment used a topology of 32 receivers representing a slightly more dense membership scenario.

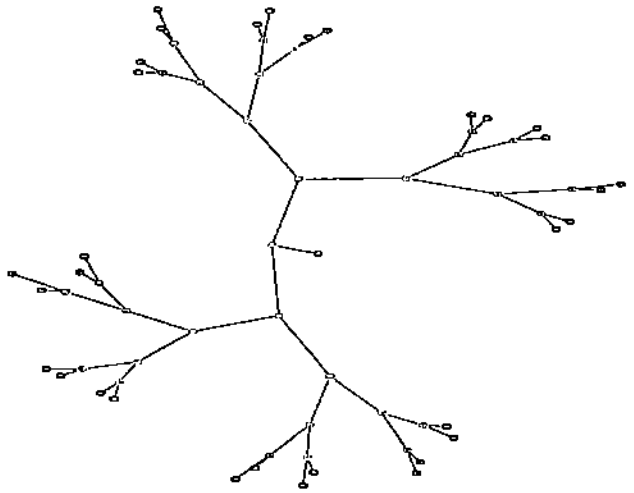
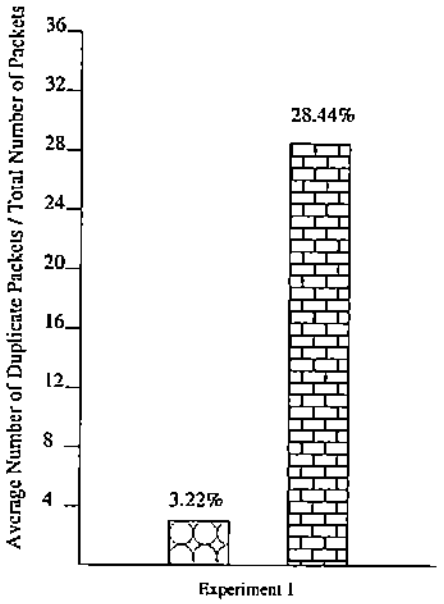


Fig. 3. Topology with 32 receivers and 6 tree levels



Graph 3: The overhead on receivers due to the duplicate packets, analysed for PGM with Proactive Adaptive FEC, and PGM with Proactive Static FEC both with k = 7 for the first experiment.

- PGM with Proactive Adaptive FEC with k = 7

- PGM with Proactive Static FEC with k = 7

The topology is shown in figure 3 (the maximum RTT is 120 ms like with the previous experiment). Again some of the links have low loss models.

The third experiment also used the same topology with 32 receivers as in figure 3, but with one link having much lower capacity (0.3 Mbps), leading to severe congestion in one of the subtrees. There is also some loss on some of the tree links. Thus, the experiment represents more extreme conditions in the network. We use a larger PGM transmit window size (many RTTs) and lower update frequency in this experiment to allow more packets to be recovered.

The above 3 experiments were conducted for PGM without FEC, then for PGM with our smooth-ID proactive FEC technique for transmission group sizes k=7, 20 or 100. The experiments were also conducted for PGM with static FEC (fixed h), to compare the network requirements and performance. The h value is initialized to 2 packets, and the increase and decrease parameters, Incr and Decr, are set to 1/7 of the transmission group size k. The interval for updating h in the first 2 experiments was set to around 210 ms, while it was larger for the third experiment as the transmission window was larger.

B. Performance Analysis

In this section, we analyze each of the experiments with respect to each of the performance metrics.

B.1 Error Recovery Efficiency (Graph 1)

Experiment 1 represents a situation with few receivers and little loss. Due to the small sender *transmit window*, the efficiency (28.5%) is very low for PGM without FEC. PGM with FEC performs more efficient recovery than the one without FEC. For smooth-ID FEC with $k = 7$, we experience higher efficiency (66.6%) than when $k = 20$ or $k = 100$. This is because we set the *PGM transmit window* to a small value and we advance it frequently. Thus some of packets cannot be recovered in time due to the NAK delay caused by the large TG size.

For experiment 2, the group membership is more dense and there is more loss. The problems due to the sender transmit window and recovery latency are more acute, and so the efficiency of PGM without FEC drops down to 19.2%. PGM with FEC performs much better, with a TG size of 7 performing best because of the small transmit window.

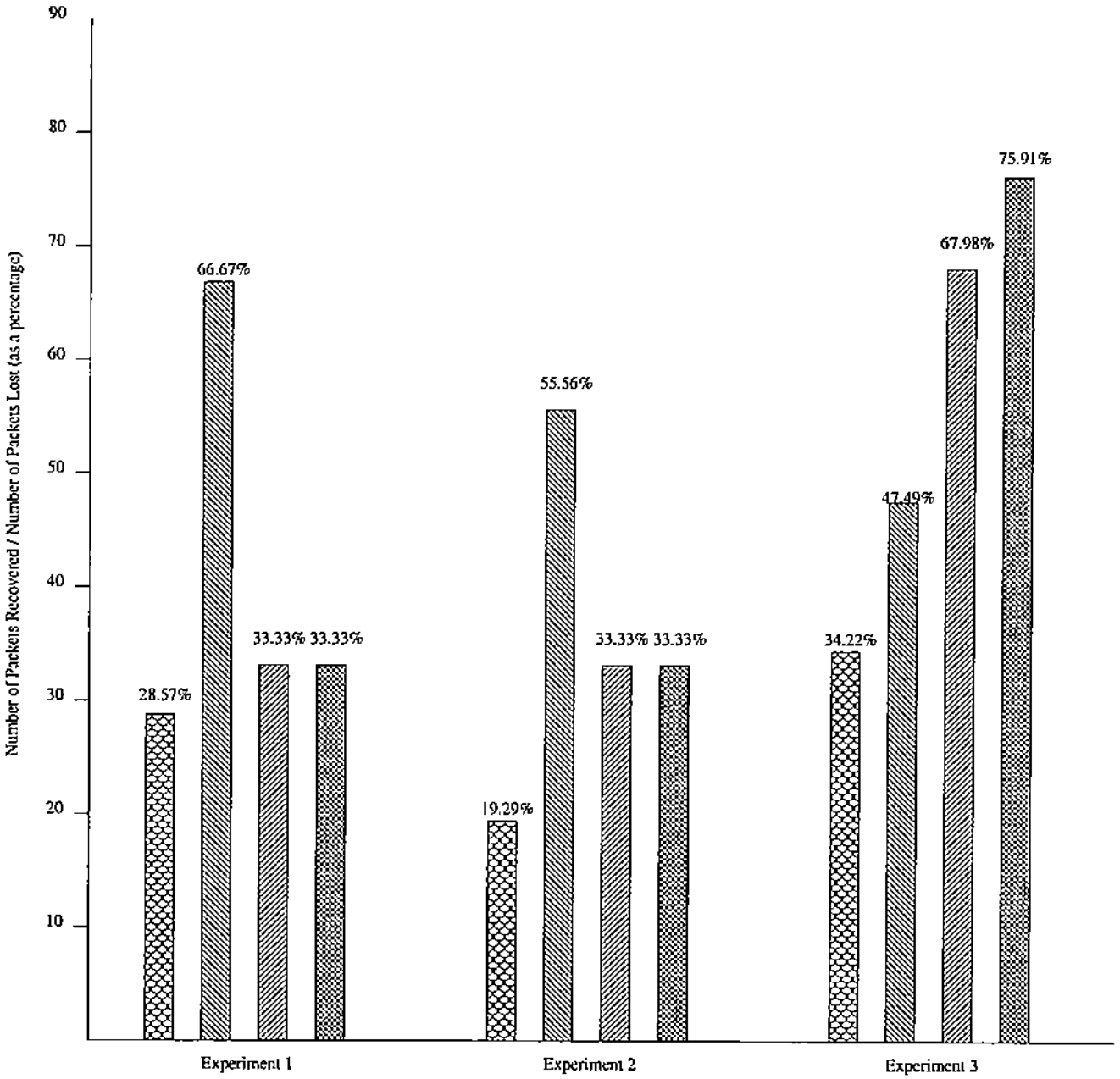
Experiment 3 contains a highly congested link (low available bandwidth), which causes many consecutive packets to be dropped. Even in this scenario, PGM with FEC has recovery efficiency as high as 75.9% when k is 100. All the values are higher because of the larger sender transmit window, especially for large k.

B.2 Retransmission Overhead (Graph 2)

In experiment 1, PGM without FEC experiences very low NAK count (1.7%) of the total packets sent. But the count drops significantly for PGM with FEC with different k values, as most of the lost packets are recovered by the proactive redundant packets.

In experiment 2, PGM without FEC experiences higher NAK count (4.4%), versus PGM with FEC.

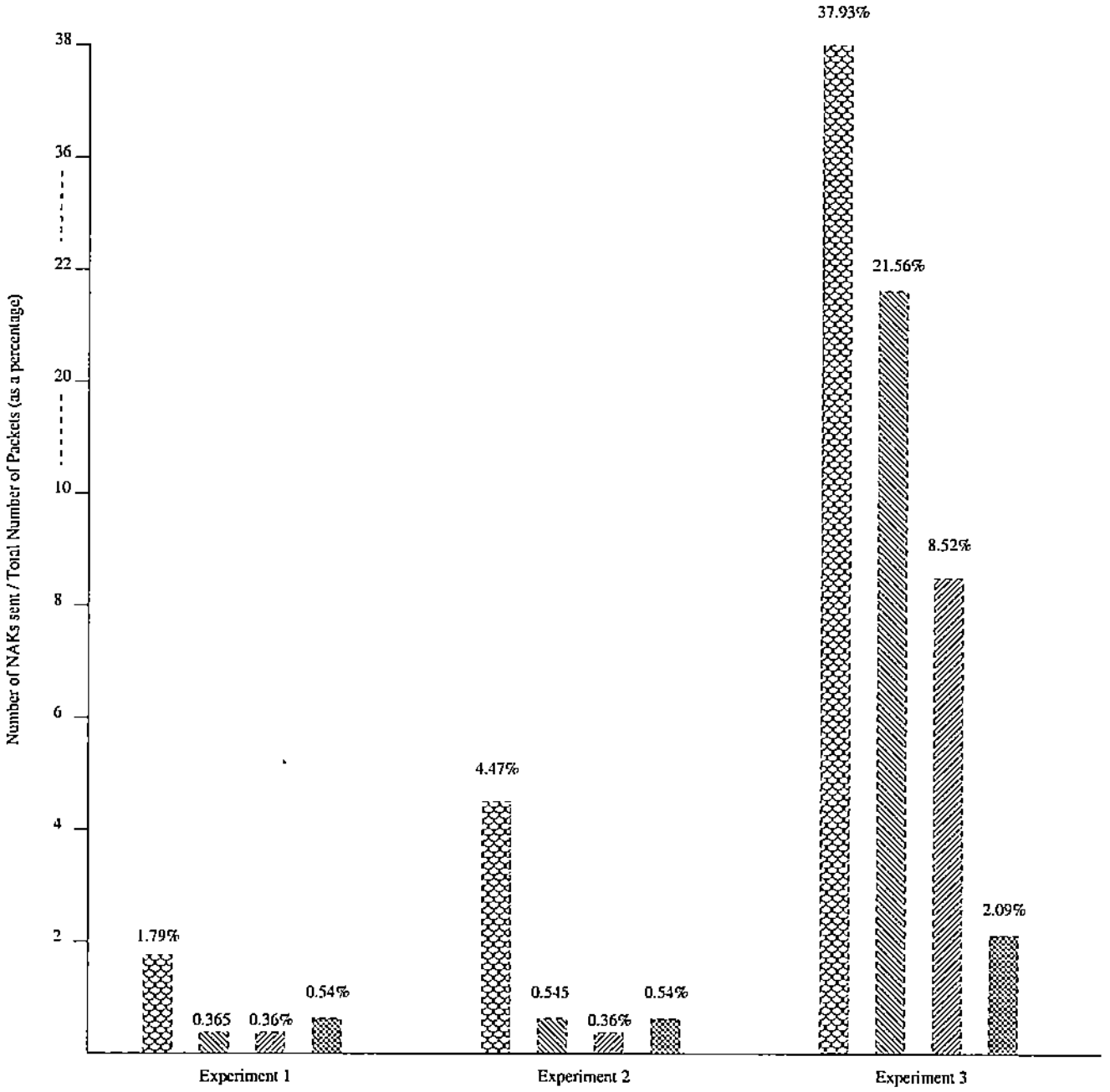
In experiment 3, PGM without FEC experiences very high NAK count (37.9%), as a NAK has to be sent for



Graph 1 : Error Recovery Efficiency analysis for PGM without FEC, and PGM with FEC with various combinations of k values (7, 20, and 100) for the three experiments

Legends :

- PGM without FEC
- PGM with Proactive Adaptive FEC with k = 7
- PGM with Proactive Adaptive FEC with k = 20
- PGM with Proactive Adaptive FEC with k = 100



Graph 2 : Average Number of NAK's experienced by the Sender, Analysed for PGM without FEC, and for PGM with Proactive Adaptive FEC (with k = 7, 20 or 100) for the three experiments

Legends :

- PGM without FEC
- PGM with FEC with k = 7
- PGM with FEC with k = 20
- PGM with FEC with k = 100

every lost packet. But PGM with FEC with k=7, 20 and 100, the NAKs are only about 21.5%, 8.5% and 2.09% respectively. This is because for PGM with FEC, we only need to send one NAK for every TG. So as the TG size increases, the NAK count decreases. Of course, higher TG sizes may increase recovery latency with ARQ.

B.3 Duplicate Packets (Graph 3)

Since PGM retransmits recovery data only on those interfaces from which the NAK came, the duplicate packets observed by the receivers are inherently reduced. With the use of proactive FEC in PGM, the duplicate packets that the receivers experience increase due to the parity packets. In graph 3, we compare the performance of PGM with smooth-ID proactive FEC, and PGM with static proactive FEC (with $h = 2$) with respect to the number of duplicate packets received. Since experiment 1 was a low loss no congestion situation, the number of duplicates drops significantly when we use the adaptive approach (it is 18 or 3.22%) compared to a very high 159 (28%) for the static approach. This is expected since the packet loss was very low (only a handful of packets lost in this experiment), and so the duplicate packet percentage is close to $\frac{h}{k} = frac27 = 28\%$ in this situation. The recovery performance and NAK overhead (not shown) are similar for both the static and adaptive cases since there is little loss in this experiment.

C. Comparison with Static FEC with Persistent Congestion

For experiment 3 where there is congestion and errors, we also compared adaptive FEC ($k = 7$) with static FEC with 2 h values: $h = 2, 7$ (28% and 100% overhead), and no FEC. We chose the smaller transmit window size of 1 second and advance interval of 0.5 seconds than those used in experiment 3 above to better illustrate the recovery differences. We use an h update interval of 0.2 seconds. We also ran the simulations for an extended 10 seconds. The results are shown in table II.

Clearly, the experiment with no FEC performs very poorly. Static FEC is highly sensitive to the choice of the redundancy parameter h. Even with this simulation when there is persistent congestion and conditions are not dynamically changing, static FEC with a small h has a lower recovery performance, while that with a large h has a high overhead. The smooth-ID redundancy estimation algorithm performs best.

D. Comparison with an AIMD Algorithm

We compared our algorithm with the additive increase multiplicative decrease algorithm mentioned in section V. The algorithm decays h by half when no NAKs for this transmission group are received, and increases it by the number of lost packets with every TG NAK (additive increase/multiplicative decrease (AIMD)). Our implementation of that algorithm is different from [8] in that it triggers the increase and decrease by a timer and not when NAKs are/are not received for a TG. However, the same increase and decrease factors are used, and the decrease takes place when no losses are reported as in [8].

We use the same setup as in the previous section for experiment 3 with the same PGM window size and advance time, same simulation time, h update time of 0.5 seconds, and transmission group sizes $k = 7, 20$. The results are shown in table III. For $k = 7$, our approach has higher recovery performance and lower NAK overhead. With $k = 20$, the recovery performance also improves, but the overhead slightly increases. The reason for the lower recovery performance for the larger k here is again the correlation between k and the transmit window size. We have noticed that the h values grow much higher with AIMD, and hence the bandwidth consumption of parity packets is high, even though we do not compute this metric here.

With the little loss no congestion situation (experiment 1), the results are shown in table IV. The 2 schemes perform similarly with little loss and $k = 7$, but the smooth increase/decrease scheme performs better for the larger k size. The multiplicative decrease, however, results in less duplicate packets, since it decreases quickly (by half). Since this experiment had isolated loss cases, quick reduction was better. In general, though, it is better to be cautious when decreasing FEC, as shown by our previous results in table III.

E. Cross TCP Traffic

The topology we used for this simulation is shown in figure 4. TCP and PGM share a common bottleneck link. Node 0 is the PGM source and there are 3 PGM receivers which join the multicast group at 0.1 seconds simulation time. We setup a TCP-Reno connection from node 6 to node 4, node 6 being the source and node 4 the sink. An infinite FTP application is used to generate data for the TCP connection. The link between nodes 1 and 2 has 0.5 Mbps bandwidth while all other links have 1.5 Mbps. RED queues are used with size 100 packets, minimum threshold 40, maximum threshold 70 and maximum drop probability 0.2. Simulation time is 15 seconds. The PGM transmit window is 4 seconds and advance interval is 2 seconds. The h update interval is 0.2 seconds. Table V shows the results. Again the smooth linear algorithm adapts best to the changing network conditions.

TABLE II
COMPARISON WITH STATIC FEC WITH CONGESTION

	No FEC	Static ($h = 2$)	Static ($h = 7$)	Smooth-ID
Percentage Recovery	0.48%	12%	14.4%	17.19%
Percentage NAKs	17.7%	9.55%	13.58%	12.1%%
Percentage Duplication	15%	36%	46%	28.72%

TABLE III
COMPARISON WITH AIMD (PERSISTENT CONGESTION)

	AIMD ($k = 7$)	Smooth-ID ($k = 7$)	AIMD ($k = 20$)	Smooth-ID ($k = 20$)
Percentage Recovery	11.63%	15.52%	8%	9.37%
Percentage NAKs	11.85%	9.79%	2.3%	3.9%
Percentage Duplication	40%	44%	39%	41%

TABLE IV
COMPARISON WITH AIMD (LITTLE LOSS)

	AIMD ($k = 7$)	Smooth-ID ($k = 7$)	AIMD ($k = 20$)	Smooth-ID ($k = 20$)
Percentage Recovery	66.67%	66.67%	33.33%	100%
Percentage NAKs	0.35%	0.35%	0.35%	0%
Percentage Duplication	5.18%	18.69%	2.1%	9.88%

TABLE V
COMPARISON IN A CONFIGURATION WITH TCP CROSS TRAFFIC

	No FEC	Static ($h = 2$)	Static ($h = 7$)	AIMD	Smooth-ID
Percentage Recovery	27.24%	30.1%	32.98%	27.81%	32%
Percentage Duplication	2.45%	3.9%	5.66%	1.86%	3.8%

VIII. CONCLUSIONS AND FUTURE WORK

We have studied the recovery performance of PGM without FEC, with static proactive FEC, and adaptive proactive FEC for different transmission group sizes. The results show that PGM with adaptive FEC has the best recovery performance. Redundancy estimation also reduces the number of duplicate packets received and the number of redundant packets sent, thereby reducing the network bandwidth requirements significantly compared to static FEC. Our redundancy estimation algorithm was superior for the network topologies, congestion and loss models studied. Larger transmission group sizes performed well only with larger PGM transmit window size. In general, the transmit window size and frequency of advancement are key parameters to the FEC scheme.

The redundancy estimation algorithm for adaptive FEC used simple increase and decrease, based upon periodic comparisons of the loss rate to its previous averaged value. Preliminary simulation results indicated this approach adapts better than the less smooth approaches that increase or decrease more drastically with loss rate fluctuations. The FEC estimation algorithm is extendible to RM protocols using bitmaps and hierarchical ACKs (HACKs) as well.

This work is clearly in a preliminary stage and a lot of experiments remain to be done. We plan to evaluate different increase and decrease increments that are dependent on the loss rate or current h value. We also plan to experiment with different percentage loss rate increases or decreases to decide when to increase or reduce h. We will conduct experiments in more realistic configurations, with receivers up to 50,000 or more, TCP and UDP background using various traffic models, other round trip times, loss models and link bandwidths, and different application traffic models for the PGM traffic. We plan to evaluate, in addition to the metrics above, the latencies of packets and bandwidth consumption in relation to other connections. We are also

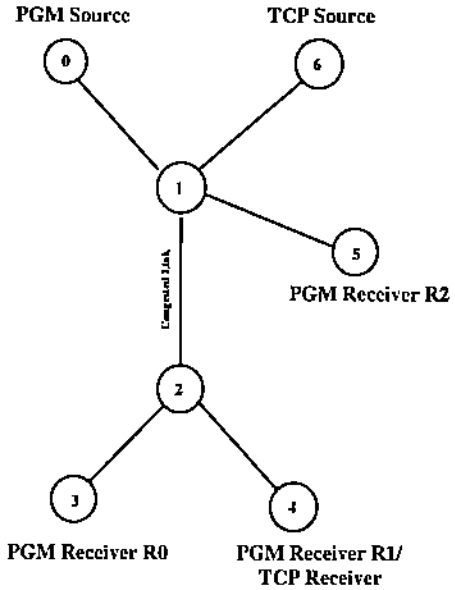


Fig. 4. Simple topology with cross TCP traffic

currently working on incorporating multicast congestion control into our PGM implementation and combining it with the proactive adaptive FEC scheme. It is important to see how well these can be integrated to achieve the best performance.

ACKNOWLEDGMENTS

The authors would like to thank S. Kalyanaraman at RPI for his helpful suggestions; S. Shi at Washington University, St Louis, for making the PGM implementation public; and D. Bhagat and D. Purohit for porting the implementation.

REFERENCES

[1] J-C. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-Based Error Control for Internet Telephony. In *Proceedings of IEEE INFOCOM*, March 1999.

[2] J-C. Bolot, A. Vega-Garcia. A Case for FEC-Based Error Control for Packet Audio in the Internet. ACM Multimedia Systems.

[3] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of the ACM SIGCOMM*, pages 56–67. September 1998.

[4] S. E. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. ACM Transactions on Computer Systems (TOCS), Vol. 8, No. 2, pp. 85–110, May 1990.

[5] S. Fahmy, R. Jain, R. Goyal, B. Vandalore, S. Kalyanaraman, S. Kota, and P. Samudra. Feedback consolidation algorithms for ABR point-to-multipoint connections. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1004–1013, March 1998. http://www.cis.ohio-state.edu/~jain/papers/cnsldt.htm.

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

[7] R. Kermode. Scoped hybrid automatic repeat request with forward error correction. In *Proceedings of the ACM SIGCOMM*, pages 278–289, September 1998.

[8] D. Li and D. Cheriton, "Evaluating the utility of FEC with reliable multicast," In Proceedings of ICNP, pp. 97–105, November 1999.

[9] S. Lin, D. J. Costello and M. J. Miller. Automatic-repeat-request error-control schemes. IEEE Communication Magazine, 1984.

[10] M. Luby et al. "Reliable Multicast Transport Building Block: Forward Error Correction Codes," Internet draft, March 2000.

[11] M. Luby et al. "Asynchronous Layered Coding: A scalable reliable multicast protocol," Internet draft, March 2000.

[12] A. J. McAuley. Reliable broadband communications using a burst erasure correcting code. In *Proceedings of ACM SIGCOMM*, September 1990.

[13] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proceedings of ACM SIGCOMM*, pages 117–130, August 1996.

[14] N. Natu, P. Rajagopal, and S. Kalyanaraman. SBMCC: A Generic Source-based Congestion Control Algorithm for Reliable Multcast. Journal of Computer Communications, to appear. http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers-rpi.html

[15] J. Nonnenmacher and E. W. Biersack. Parity-based loss recovery for reliable multicast transmission. IEEE/ACM Transactions on Networking, Vol. 6, no. 4, August 1998.

[16] J. Nonnenmacher and E. W. Biersack. Reliable multicast: Where to use FEC. In *Proceedings of 5th International Workshop on Protocols for High Speed Networks*, October 1996.

[17] K. Obraczka. Multicast transport protocols: A survey and taxonomy. IEEE Communications Magazine, 1997.

[18] M. Podolsky, C. Romer, S. McCanne. Simulation of FEC-based error control for packet audio on the Internet. In *Proceedings of IEEE INFOCOM*, San Francisco, CA, p. 505, April 1998.

[19] L. Rizzo. Effective erasure codes for reliable computer communication protocols. ACM Computer Communication Review, vol. 27, no. 2, pp. 24–36, Apr. 1997.

[20] L. Rizzo. pgmcc: a TCP-friendly single-rate multicast congestion control scheme. In *Proceedings of ACM SIGCOMM*, August 2000.

[21] L. Rizzo and L. Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques", In *Proc. of The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS)*, June 1997.

[22] J. Rosenberg, L. Qiu and H. Schulzrinne, "Integrating Packet FEC into Adaptive Voice Playout Buffer Algorithms on the Internet," in *Proceedings of IEEE INFOCOM*, March 2000.

[23] D. Rubenstein, J. Kurose, and D. Towsley. Real-Time Reliable Multicast Using Proactive Forward Error Correction. In *Proceedings of NOSDAV*, pp. 279–293, July 1998.

[24] S. Shi, Washington University at St Louis. PGM implementation. http://www.arl.wustl.edu/~sherlia/

[25] T. Speakman, et al. PGM Reliable Transport Protocol Specification. Work in progress, April 2000.

[26] T. Tuan and K. Park. Multiple time scale redundancy control for QoS-sensitive transport of real-time traffic. In *Proceedings of IEEE INFOCOM*, March 2000.

[27] UCB/LBNL/VINT Network Simulator. http://www-mash.cs.berkeley.edu/ns/

[28] L. Vicisano, J. Crowcroft, and L. Rizzo. TCP-like congestion control for layered multicast data transfer. In *Proceedings of IEEE INFOCOM*, volume 3, pages 996–1003, March 1998.

[29] B. Whetten and G. Taskale, "An Overview of Reliable Multicast Transport Protocol II," *IEEE Network Magazine*, January/February 2000.