*Proceedings of the 2001 IEEE Symposium on Applications and the Internet* (SAINT-2001), San Diego-Mission Valley, California, January 2001.

# An Experimental Study of Adaptive Forward Error Correction for Wireless Collaborative Computing

Philip K. McKinley [*]
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
mckinley@cse.msu.edu

Arun P. Mani [†]
Lucent Technologies - Bell Laboratories
101 Crawfords Corner Rd
Holmdel, NJ 07733
amani@lucent.com

## Abstract

*This paper describes an experimental study of a proxy service to support collaboration among mobile users. Specifically, the paper addresses the problem of reliably multicasting web resources across wireless local area networks, whose loss characteristics can be highly variable. The software architecture of the proxy service is described, followed by results of a performance study conducted on a mobile computing testbed. The main contribution of the paper is to show that an adaptive forward error correction mechanism, which adjusts the level of redundancy in response to packet loss behavior, can quickly accommodate worsening channel characteristics in order to reduce delay and increase throughput for reliable multicast channels.*

## 1. Introduction

The large-scale deployment of wireless communication services and advances in portable computers are quickly making “anytime, anywhere” computing into a reality. One class of applications that can benefit from this expanding and varied infrastructure is collaborative computing. Examples include computer-supported cooperative work, computer-based instruction, collaborative scientific experimentation, and crisis management systems. A diverse infrastructure enables individuals to collaborate via widely disparate technologies, some using workstations on high-speed local area networks (LANs), and others using wireless handheld/wearable devices.

Collaborative applications differ widely in their quality-of-service requirements and, given their synchronous nature, they are particularly sensitive to the heterogeneous characteristics of the computing devices and the network connections used by participants. One approach to accommodating heterogeneity is to introduce a layer of *middleware* between applications and underlying transport services. The appropriate middleware framework not only can help to hide differences among networks and computing devices, but can facilitate the development of new applications through software reuse and domain-specific extensibility.

Towards this end, we have developed Pavilion [11], an object-oriented middleware framework for collaborative web-based applications. Pavilion enables a developer to construct new applications by inheriting and extending its default functionality. In a follow-on project, we are creating RAPIDware, a design methodology that enables middleware components to dynamically respond to disparities among participating hosts and changing conditions on their network connections.

One technology that is likely to play an important role in this area is the wireless LAN, which can provide network access to mobile users with handheld, laptop, and wearable computing devices. Usually, such a network is installed as an extension to an existing wired LAN. We refer to such a configuration as a *heterogeneous LAN*. While many future collaborative applications will involve mobile users interconnected by cellular services, other applications will involve users (the entire group or a subset thereof) in a local environment, such as a school, office, factory, or hospital. In such settings, a multicast-capable local infrastructure, such as a heterogeneous LAN, may be less expensive and offer higher bandwidth than cellular services.

Extending collaborative applications to wireless hosts calls for redesign of communication services in order to accommodate the relatively high loss rates and generally lower bit rates of such environments. Many approaches to solving this problem involve the use of proxies [2,3], which represent wireless receivers to the rest of the wired network.

[*] This work was supported in part by the NSF grants CCR-9503838, CDA-9617310, NCR-9706285, and CCR-9912407.

[†] This work was conducted while the author was a graduate student at Michigan State University.

In this paper, we describe an experimental study in the use of proxy services to support collaboration across heterogeneous LANs. Specifically, we address the issue of reliably multicasting web resources to users in such environments.

The main contributions of this work are threefold. First, we show how object-oriented proxy services can be constructed from existing protocols and processing components. Second, we demonstrate the effect of proxy-based flow control on reliable multicast receivers in heterogeneous LANs, specifically, that the use of multiple instances of the protocol in the proxy hides the effects of slow wireless receivers on faster wired receivers. Third, and the primary focus of the paper, we describe an adaptive forward error correction (FEC) mechanism, which adjusts the level of redundancy in response to channel loss behavior. Our results complement those of Rizzo [15] and Towsley [13] by providing experimental results demonstrating that a proxy-based, adaptive FEC generator can quickly accommodate dynamic channel characteristics in order to reduce delay and increase throughput. We focus on block erasure codes [8, 15], due to their ability to correct uncorrelated packet losses among multiple mobile receivers.

The remainder of the paper is organized as follows. In Section 2, we provide background information on the Pavilion and RAPIDware projects. Section 3 discusses the relevant issues in reliable multicasting in wireless LANs and motivates the use of a proxy server. Section 4 describes how FEC is integrated into the proxy, and Section 5 presents the details of the adaptive FEC protocol and evaluates its performance. In Section 6 we review related research projects, and in Section 7 we summarize the results and discuss future directions. Due to space limitations, many details of the project are omitted in this paper, but may be found in a companion technical report [12].

## 2. Pavilion and RAPIDware Projects

Pavilion [11] is a framework that supports synchronous web-based collaboration. As with similar frameworks, Pavilion can be used in a default mode, in which it operates as a collaborative web browser [9]. Moreover, Pavilion enables a developer to construct new collaborative applications by reusing and extending existing components: interfaces to commercial web browsers, a suite of communication protocols, a leadership protocol for session floor control, and a variety of proxy servers that manipulate data streams enroute to participating applications. The Pavilion framework itself is written in Java, but supports components written in other languages, including off-the-shelf software such as Netscape Navigator, Internet Explorer, and virtually any helper application for displaying a particular media type.

Pavilion makes use of *local* proxies, which execute on the client's host, as well as *remote* proxies, which execute on other systems accessible to the client. Local proxies are typically used to enhance application functionality and usually require access to client system resources, such as the hard disk. Remote proxies are commonly used to enhance performance by tailoring data streams to match the capabilities of the client platform or network connection.

We have used both types of proxies in Pavilion-based applications. For example, Pocket Pavilion [10] enables multimedia collaboration among users of Windows CE handheld computers. A remote proxy, comprising existing Pavilion components and a set of plug-ins, is used to reduce the processing load on the handheld computers and accommodate the limitations of the Windows CE environment. In addition, we have developed VGuide [1, 11], a collaborative virtual reality application that enables a user to select a VRML (Virtual Reality Modeling Language) file from the Internet and lead a group of users through that "world." This task requires monitoring the position and orientation of the leader and multicasting it to the clients. VGuide uses plug-ins on both local and remote proxies to enable synchronous navigation and to avoid overwhelming slow receivers.

RAPIDware extends Pavilion by *automating* the instantiation and reconfiguration of middleware components, such as proxies, in order to accommodate resource-limited hosts and dynamic network conditions. Figure 1 depicts a simple example; adaptive components operate as plug-ins that extend the functionality of existing components, both on the local hosts and on remote proxies. Some plug-ins (observers) monitor the system for conditions that potentially affect the operation or performance of the application. Examples include changes in the quality of a network connection, disparities among participating devices, and changes in user/application preferences or policies. Other plug-ins (responders) are programmed to handle such events by instantiating new components or modifying the behavior of a communication protocol.

In this paper, we investigate proxy-based adaptability as applied to a key service needed in mobile collaboration, namely, reliable multicasting in wireless environments. The project uses and extends the Web-Based Reliable Multicast (WBRM) protocol [9], which is used by Pavilion to distribute web resources and control information to participating systems. WBRM is an application-level protocol that implements reliability atop IP multicast. Pavilion-based applications use the protocol to deliver web resources efficiently from the leader of a session to rest of the participants. Like many reliable multicast protocols [4, 16], the WBRM protocol is a receiver-initiated, or NAK-based, protocol. A receiver notifies the sender only when it misses a packet, with packets identified by sequence numbers. Both the sending and receiving components of the protocol comprise a set of Java threads and data structures. Flow control

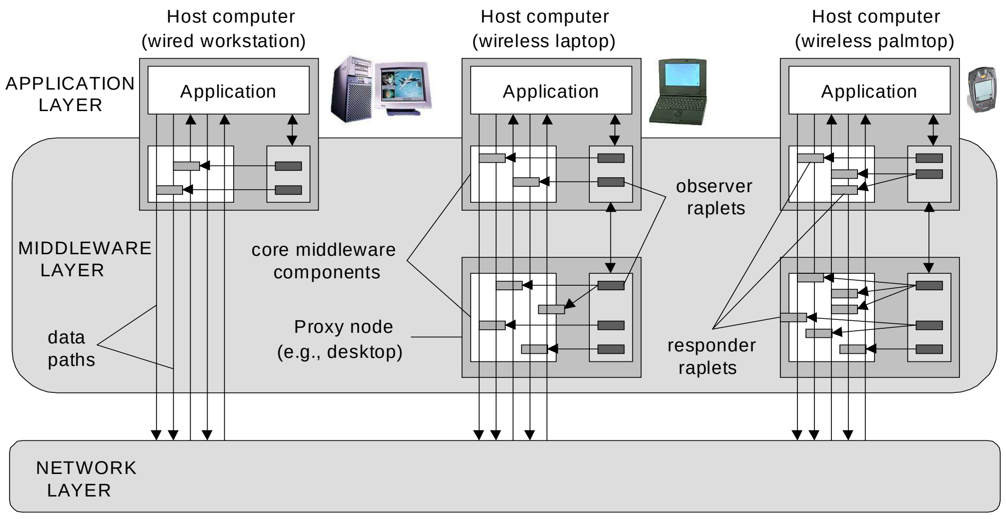

Host computer (wired workstation)
Host computer (wireless laptop)
Host computer (wireless palmtop)
APPLICATION LAYER
Application
Application
Application
MIDDLEWARE LAYER
core middleware components
observer raplets
data paths
Proxy node (e.g., desktop)
responder raplets
NETWORK LAYER

**Figure 1. Configuration of RAPIDware adaptive middleware components.**

in WBRM is extensible and user configurable. The default method is rate-based and is similar to that of the RAMP protocol [7]: the delay between packets is a function of the ratio of the total number of packets sent during an interval to the number of NAKs received during the same interval. Additional details of the operation and performance of the WBRM protocol can be found in [9, 11].

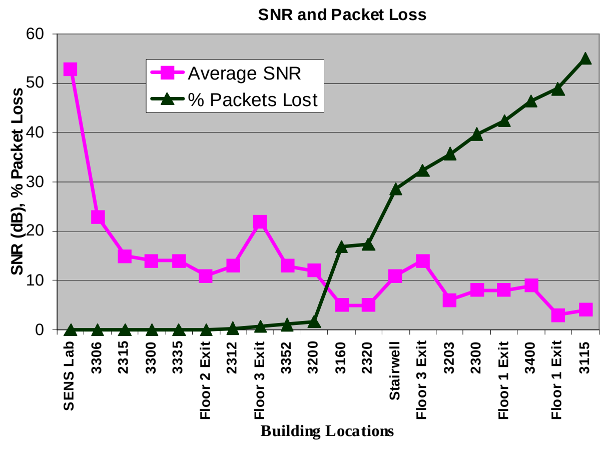
## 3. Need for Reliable Multicast Proxies

Our prior work with the WBRM protocol has concentrated primarily on wired network environments. Extending a reliable multicast service to accommodate wireless LANs must address three main characteristics of such networks: highly dynamic channel loss rates, the behavior of the 802.11 CSMA/CA MAC layer (unlike unicast frames, no link-level acknowledgements for multicast frames), and the effect of a shared channel on reverse traffic such as NAKs and flow control requests. Figure 2(a) plots sample results illustrating the relationship between signal-to-noise ratio (SNR) and packet loss rate on one of the wireless LANs in our laboratory testbed. The network is a Lucent WaveLAN network, which uses direct sequence spread spectrum signaling and has a raw bit rate of 2 Mbps. (We have recently added an 11 Mbps Aironet wireless LAN to the testbed, but our testing on that network is not yet complete.) We collected these data using a laptop computer equipped with a WaveLAN interface card; measurements for SNR and packet loss rate were recorded at various locations within range of the WaveLAN access point. The results demonstrate the highly variable loss rate that can occur in such environments, and the SNR values quickly drop below the level of 20 dB.

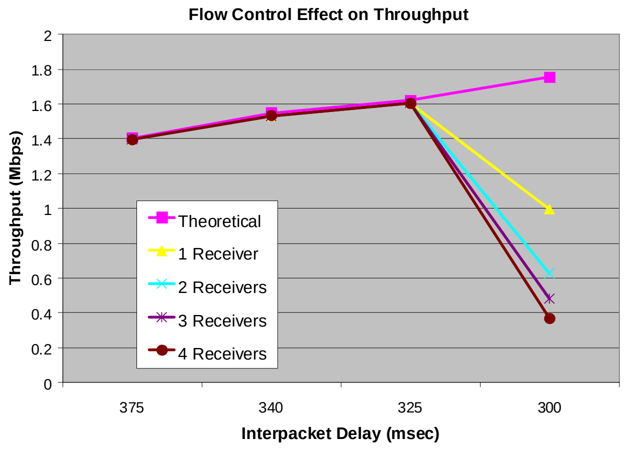
However, even near the access point, the performance of reliable multicasting can suffer due to other factors. For example, Figure 2(b) shows the results of a set of experiments using the WBRM protocol to reach multiple wireless receivers, all located inside our laboratory, where the SNR value is high. In these tests, we artificially fixed the inter-packet delay values used in WBRM's rate-based flow control method. While the protocol can achieve a throughput of approximately 1.6 Mbps on the 2 Mbps link, the value drops dramatically when the sending rate is too high, apparently due to buffer overflow at the wireless access point. Such behavior affects not only reliable multicasting, but other group operations, such as leader election and floor control.

The bandwidth differential problem can be addressed by inserting a proxy between the wireless nodes and the rest of the wired network, as shown in Figure 3. The proxy node plays the role of both a reliable multicast receiver, with respect to the original wired sender, and a reliable multicast sender, with respect to the wireless nodes. Executing two instances of the WBRM protocol, one for the wireless receivers and the other for the wired receivers and proxies, enables the flow control algorithm to tune itself independently to match the characteristics of each network segment.

By buffering data for delivery on the slower channels, the proxy enables the receivers with fast connections to make better use of that capacity. Moreover, using a proxy can improve (slightly) the performance for the wireless receivers, since NAKs and retransmissions are exchanged between the receiver and the proxy, instead of between the receiver and the original sender. Figure 4 shows the results of a set of experiments that we conducted in our laboratory. Figure 4(a) plots throughput results when the proxy service is turned off; the performance of wired nodes degrades to that



(a) SNR values and packet loss rate



(b) flow control effect on reliable multicast throughput

**Figure 2. Experimental measurements of wireless LAN performance**

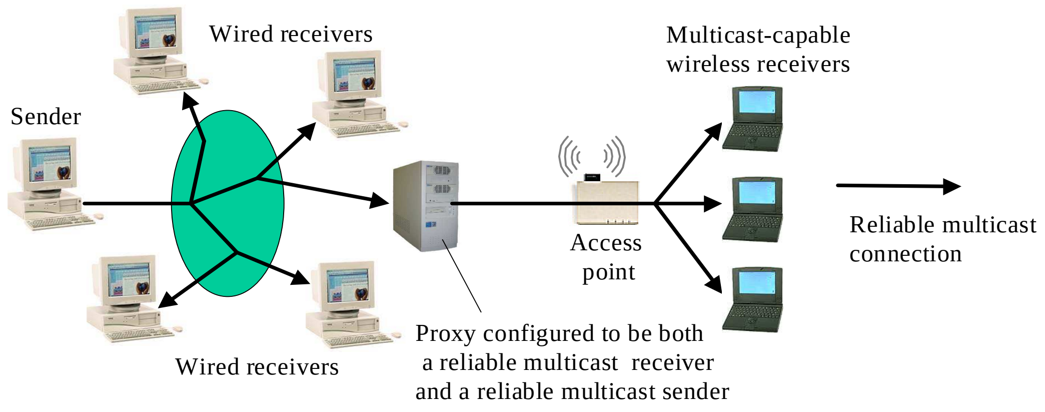


**Figure 3. Proxy configuration for nodes on a wireless LAN.**

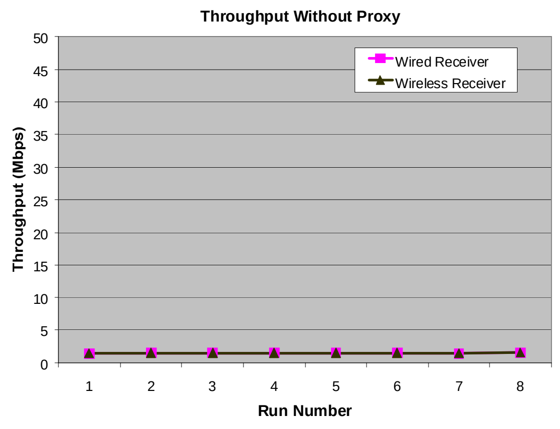
of the wireless nodes (1.5 Mbps). In contrast, Figure 4(b) plots throughput results when the proxy service is turned on; the wired nodes achieve approximately 40 Mbps, while the wireless nodes achieve 1.6 Mbps.

While buffering and flow control are necessary to accommodate the lower bit rates of wireless channels, they do not directly address the potentially higher packet loss rates. In order to do so, we experimented with forward error correction techniques to determine which specific methods and parameter settings might best serve collaborative applications when executed, partially or wholly, on heterogeneous LANs. Our approach is discussed in the next two sections.

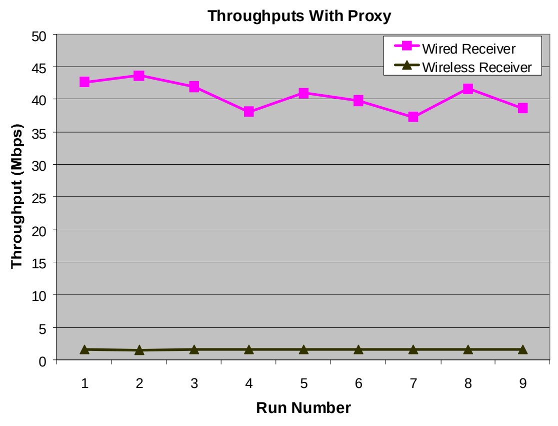
## 4. Proxy-Based Forward Error Correction

Forward error correction is the sending of redundant information with a data stream, enabling the receiver to correct errors/losses without contacting the sender. Since CRC-based error detection at the data link layer typically results in the removal of the corrupt packet(s) from the stream, many FEC-based protocols target *erasures* [15]. An $(n, k)$ *block erasure code* converts $k$ source packets into $n$ encoded packets, such that any $k$ of the $n$ encoded packets can be used to reconstruct the $k$ source packets [8]. A code is *systematic* if the first $k$ of the $n$ encoded packets are identical to the $k$ source packets. In this work, we use only systematic $(n, k)$ codes. We refer to the first $k$ packets as *data* packets, and the remaining $n - k$ packets as *parity* packets. Each set of $n$ encoded packets is referred to as a *group*.

The advantage of using block erasure codes for reliable multicast is that a single parity packet can be used to correct independent single-packet losses among different receivers. Hence, sending parity packets with data packets reduces the number of NAKs sent by receivers. Moreover, when receivers do require additional parity packets, the sender can respond to NAKs from different receivers with a sin-



(a) proxy not running



(b) proxy running

**Figure 4. Effect of a proxy server on reliable multicast throughput.**

gle set of parity packets. Recently, Rizzo [14] studied the feasibility of software encoding/decoding for packet-level FEC, using a particular block erasure code called the Vandermonde code. Depending on the values of $k$ and $n - k$, Rizzo showed that this code can be efficiently executed on many common microprocessors. Recently, several reliable multicast protocols have been designed to incorporate FEC, for example [5, 6, 13, 15]. Most use a combination of FEC and ARQ (Automatic Retransmission reQuest), in that some parity packets initially are sent with the data packets, and additional parity packets are sent as needed in response to NAKs from receivers; see Section 6.

Figure 5 shows the components that constitute the FEC proxy. Several components are reused directly from the original WBRM protocol; plug-ins and associated data structures are shaded. The *WBRM Receiver* is directly reused from the WBRM protocol. It receives multicast data packets over the wired network and delivers a reliable output stream of these packets to the *Packet Buffer*. The remaining components implement the sending half of the Wireless WBRM (W-WBRM) protocol. The *FEC Group Filter* collects the data packets into FEC data blocks of size $k$ and places them in the Dispatcher Queue. The *FEC Encoder* is a plug-in that monitors the Dispatcher Queue; when it detects that a group of $k$ packets is full, it invokes the encoding routines and produces the $n - k$ parity packets. The *Packet Dispatcher* is reused from WBRM and simply uses IP multicast to transmit packets in the Dispatcher Queue. If a receiver detects that it has lost more data packets than the number of parity packets it has received, then it informs the proxy by sending a NAK message. Based on received NAKs, the *NAK Processor* signals the dispatcher to send additional parity packets. Most of the proxy components are written in Java and comprise one or more threads. The lone exception is the FEC Encoder, for which we used Rizzo's public domain C code [15]. With minor modifications, we compiled this code to a native library and invoked it from our Java proxy code using the Java Native Interface.

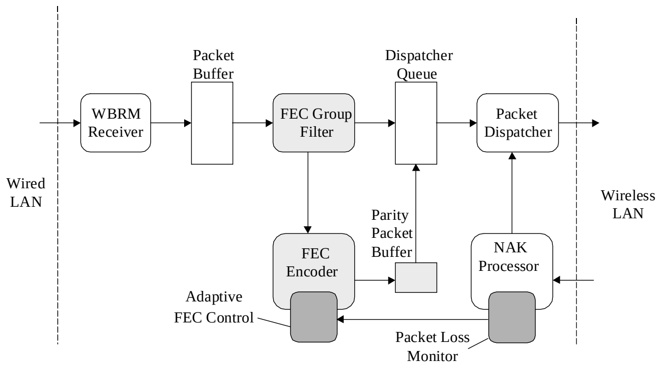


**Figure 5. Operation of FEC proxy.**

For a given group, after sending the $k$ data packets, the W-WBRM protocol could continue and send all $n - k$ parity packets. However, in most situations not all of these packets will be needed, and their transmission will consume bandwidth unnecessarily. While we could require each receiver to send back positive information when it has received enough parity packets for each group, this solution is inconsistent with the NAK-based nature of the protocol and would require the proxy to maintain per-receiver state information. Instead, we chose a compromise solution that uses forward error correction to reduce feedback from wireless receivers, while attempting to minimize transmission of unneeded parity packets. We introduce a *proactive* parameter, $\alpha$, which represents the level of redundancy in the transmissions of data and parity packets. For each group of $n$ encoded packets, the proxy immediately sends $k(1 + \alpha)$ packets (actually, it sends $\lceil k(1 + \alpha) \rceil$ packets; we ignore this technicality in the remaining discussion). Any receiver that loses fewer than $\alpha k$ of these packets can recover from the losses, while a receiver that loses more than $\alpha k$ packets will send a NAK to the proxy requesting additional parity packets. The NAK format includes fields to identify $G$, the packet group, and $L$, the number of packets required to reconstruct the original $k$ data packets in group $G$.

We emphasize that $\alpha$ is applied to *all* transmissions, including transmission of requested parity packets. Hence, in response to a NAK, the NAK Processor makes available to the dispatcher $L(1 + \alpha)$ additional parity packets for transmission. Our early experiments showed that multiple NAK rounds, resulting from the loss of parity packets sent in response to a NAK, can be very detrimental to performance. At present, the W-WBRM protocol does not implement global NAK suppression [4] among wireless receivers, but rather uses the proactive transmission of parity packets to "suppress" NAKs. However, the NAK Processor does implement *parity packet suppression* to avoid unnecessary transmission of parity packets, by comparing the $L$ values in NAKs, for a given packet group, that are received within a specified window of time.

The main challenge for the W-WBRM protocol is to reduce the amount of feedback traffic from receivers in the form of NAKs while maintaining reliability and bandwidth efficiency. The proactive parameter, $\alpha$, helps reduce NAK traffic by sending parity packets for anticipated losses. However, sending too many parity packets immediately may waste bandwidth in low-loss situations. Clearly, the value of $\alpha$ depends on the dynamic loss characteristics of the wireless channel at the receivers. To study the effect of different $\alpha$ values on the performance of the protocol for different loss rates, we conducted a set of tests; a sampling of the results is shown in Figure 6. The plots show the results for reliably multicasting to three wireless laptop receivers under loss rates of 5% and 10%. In order to control the error rate, we emulated random packet losses on the all hosts (we discuss experimental loss conditions later). An FEC packet size of 1400 bytes was used to transfer a 4 MB file to wireless receivers. The FEC parameters $(n, k)$ were $(60, 20)$, that is, 40 parity packets are computed for each group of 20 data packets.

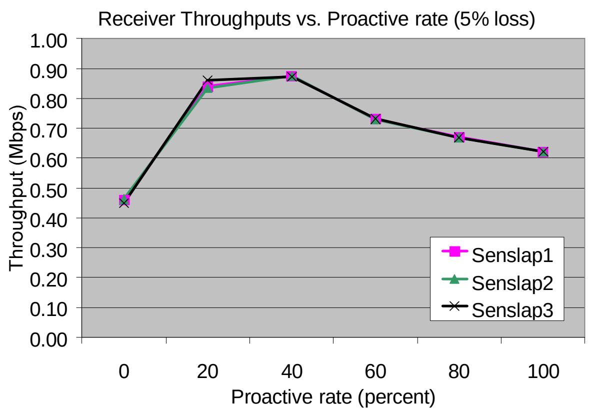
These plots demonstrate two important results about the throughput for the wireless receivers. First, as expected, the ideal value for $\alpha$ varies with network conditions. Specifically, the $\alpha$ value at which throughput is maximized increases with the packet loss rate. Therefore, the W-WBRM protocol should update the value of $\alpha$ in response to changing loss rates among receivers. Second, all of the curves follow a similar pattern: a relatively steep ascent just before the optimal value, and a gradual descent beyond this value. This behavior suggests that sending more parity packets than necessary is better than relying on feedback in the form of NAKs from receivers. However, sending too many unnecessary parity packets will eventually cause significant decrease in throughput.

## 5. Design and Evaluation of Adaptive FEC

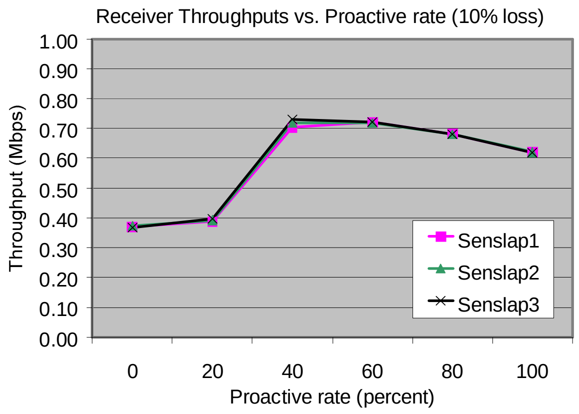
Given the observations of the previous section, we modified the W-WBRM protocol at the proxy so that the value of $\alpha$ is dynamically adjusted to accommodate loss conditions. As shown in Figure 5, a plug-in to the NAK Processor called the *Packet Loss Monitor* collects information on NAKs and forwards it to the *Adaptive FEC Control* (AFC) plug-in, which adjusts the value of the $\alpha$ by setting $\alpha = \alpha + \alpha_{\text{inc}}$. The value of $\alpha_{\text{inc}}$ depends on the actual error rate and cannot be determined a priori. We take a conservative approach and use the formula $(\alpha_{\text{inc}} = M * L/k)$, where $M$ is a small integer (specific values are discussed below), since our previous experimental results indicate that it is less costly to overestimate $\alpha$ than to underestimate it. To avoid having $\alpha$ remain unnecessarily high, we periodically reduce its value using a function, $\alpha_{\text{dec}}$, if no NAKs have arrived at the proxy in a window of W groups. In the experiments discussed below, we used both a linear and an exponential function for $\alpha_{\text{dec}}$. Eventually, $\alpha$ becomes low enough that the receivers produce one or more NAKs. At this point, $\alpha$ is again increased by an amount relative to the number of requested packets, and the process repeats. Considering a stream of web resources being multicasted to the wireless receivers, our goal is to keep the the number of packets transmitted "hovering" slightly above the number actually needed to decode the data packets.

We conducted a set of tests to evaluate the behavior of the resulting adaptive FEC protocol. We focused primarily on the parameters $\alpha_{\text{inc}}$ and $W$. Figure 7 shows a sampling of the results, in which we sent a 4MB file twice via the proxy to a single wireless receiver; the packet size is 1400 bytes and the packet loss rate in all cases is 20 percent. The FEC parameters are again $(60, 20)$ and as before, losses are emulated by dropping packets randomly. Each plot contains three curves that illustrate the behavior of the protocol as the experiment progresses. The *Required Parity* curve shows, for each group in the file, the total number of parity packets needed by the receiver to decode the source packets of the group. The *Proactive Packets* curve plots, for each group, the total number of parity packets proactively sent by the proxy. The *NAK Response* curve in the graph shows the total number of parity packets actually sent by the proxy in response to (all) NAKs from the receiver for each group. Whenever the number of proactive packets is less than the required parity packets for the group (indicated by the spikes in *NAK Response* curve), feedback is received in the form of a NAK and the proactive parameter $\alpha$ is bumped to a higher value.

Clearly, the values of parameters $M$ and $W$ directly af-

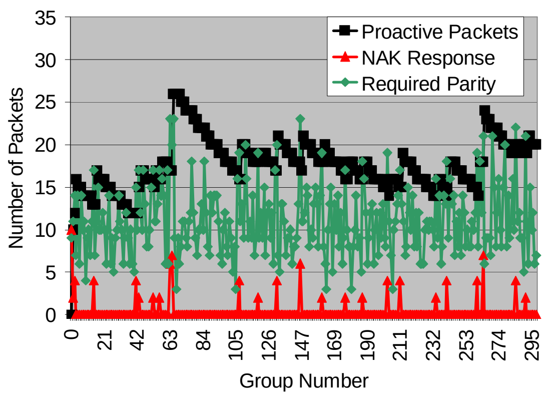


(a) 5% packet loss at all receivers

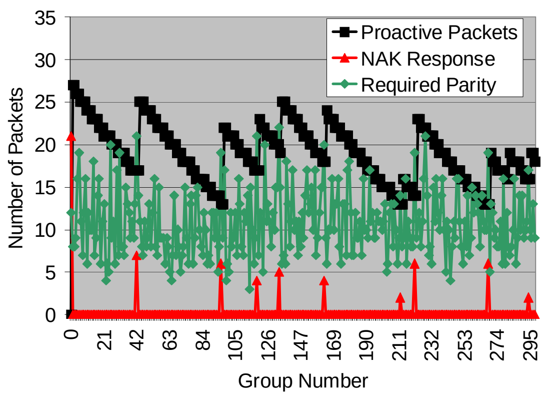


(b) 10% packet loss at all receivers

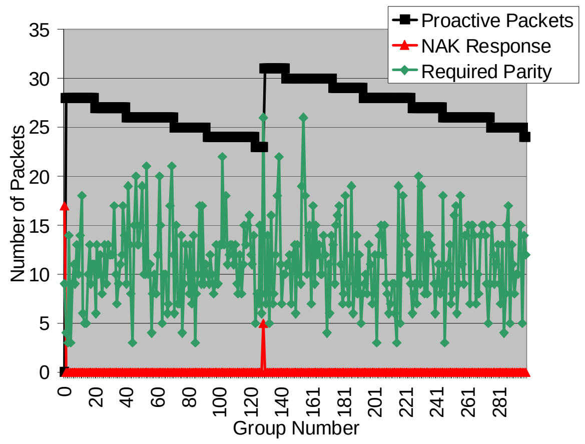
**Figure 6. Throughput results with static proactive rates.**



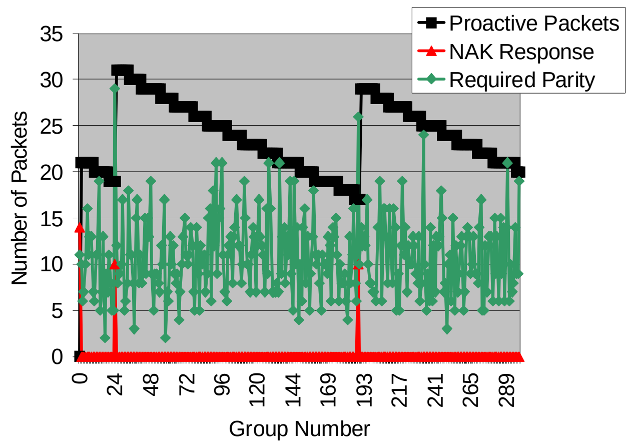
(a) $\alpha_{\text{inc}} = 2 * L/k, W = 3$ groups



(b) $\alpha_{\text{inc}} = 3 * L/k, W = 3$ groups



(c) $\alpha_{\text{inc}} = 4 * L/k, W = 10$ groups



(d) $\alpha_{\text{inc}} = 3 * L/k, W = 10$ groups

**Figure 7. Adaptive FEC behavior for single receiver, varying parameters $M$ and $W$.**

fect the NAK behavior of the protocol. When $M = 2$ (that is, $\alpha_{\text{inc}} = 2 * L/k$), as shown in Figure 7(a), the proactive rate $\alpha$ is increased in response to a NAK, but the receivers often require additional parity packets in subsequent groups, producing many NAKs. Increasing $M$ to 3 improves the situation, as shown in Figure 7(b), but with the window $W = 3$ groups, the $\alpha$ decreases too quickly, producing many situations in which additional parity packets must be sent in response to NAKs. By increasing the window size to 10, as shown in Figures 7(c) and 7(d), the protocol limits this behavior. As shown in Figure 7(c), however, if the value of $M$ is too large (4 in this case), the protocol risks transmitting too many unneeded parity packets. In the remainder of the experiments reported here, we set $M = 3$.

Figure 8 plots the resulting throughput when using adaptive FEC in the W-WBRM protocol, compared to the original WBRM protocol (no FEC) and W-WBRM with static FEC. The data stream is multicast to three wireless laptop computers under different (emulated) packet loss conditions: 5%, 10%, and 20%. In these tests, $M = 3$ and the FEC parameters are $(60, 20)$. The value of $\alpha_{\text{dec}}$ is set to 0.02 and $W = 10$, so $\alpha$ will decrease by 0.02 every 10 groups (in the absence of NAKs). The adaptive protocol improves throughput by a factor of 2 for all situations involving packet loss.

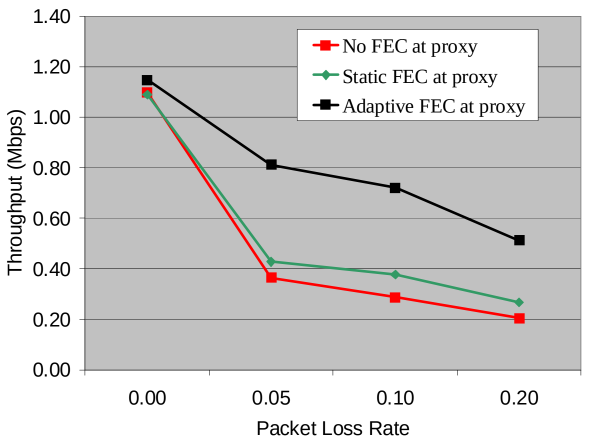


**Figure 8. Performance of adaptive FEC protocol (three laptop receivers).**

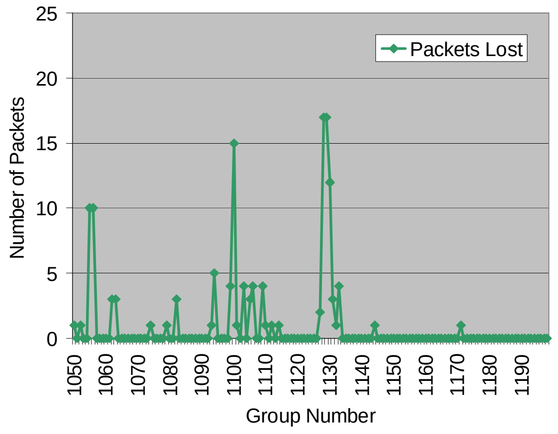
In addition to random packet losses, we also consider burst errors, which have been shown to be common in wireless LANs. Figure 9(a) shows an example of this behavior: we measured the loss rate on a laptop receiver during a short "excursion" outside our laboratory, in which we multicasted a 4MB file repeatedly. The packet loss behavior shown in Figure 9(a), as well as that in the other subfigures, exhibits a bifurcation between (1) relatively large burst errors and (2) random single-packet losses. For such environments, it might be desirable for the $\alpha_{\text{dec}}$ function to decrease faster than linear. Figures 9(b-d) show results for the W-WBRM protocol when executed in our testbed with real (not emulated) packet losses. Three laptop receivers were involved; two remained in our lab, and one was carried by a user who traversed a nearby hallway. In Figure 9(b) we used a linear $\alpha_{\text{dec}}$ function, as in earlier tests. While the proactive parity packets handle most of the losses, the overhead is quite large. In Figure 9(c) we changed $\alpha_{\text{dec}}$ so that $\alpha$ decreases exponentially. In this case, the protocol correctly predicts several large losses and handles them. However, the lower bound on $\alpha$ is 0, and many single-packet losses produce NAKs. In Figure 9(d) we use the same exponential function, but require that the protocol always send at least one proactive packet. In this case, the NAK feedback is significantly reduced.

In summary, we have conducted an experimental study (using both emulated packet losses and real packet losses) of proxy-based FEC for reliable multicasting. Our results show that different loss distributions (e.g., random losses vs. burst errors) require different logic, and hence different plug-in components for inserting FEC packets into the data stream. For both random and burst errors, the value of $\alpha$ must increase quickly in response to losses. However, for burst errors it is also important that the value of $\alpha$ decreases quickly, at least to some lower threshold. Our ongoing studies explore alternative functions for $\alpha_{\text{inc}}$ and $\alpha_{\text{dec}}$ and their effects on performance.

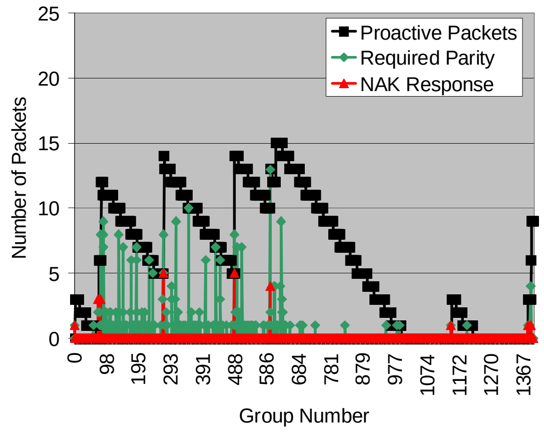
## 6. Related Work

The Pavilion/RAPIDware projects complement studies in several areas; comparisons with other groupware/middleware frameworks can be found in [11, 12]. In this section, we confine our discussion to the relationship between the W-WBRM protocol and other FEC-oriented reliable multicast protocols.

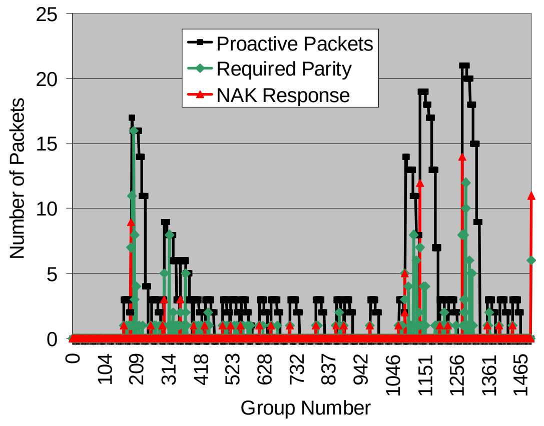
Rizzo's work on efficient implementations of erasure codes [14] is an important contribution that has affected the design of multicast protocols for lossy environments. The resulting public-domain source files have been used in many research projects, including this one. The RMDP protocol proposed by Rizzo and Vicisano [15] is an FEC-based reliable multicast protocol to be used over the MBone and wireless mobile networks with asymmetric communication channels. RMDP is a hybrid FEC+ARQ protocol that uses several operating parameters set according to the type of network. One such parameter, $D$, the *expansion factor*, is the rate at which parity packets are sent unconditionally with the data packets. The protocol uses global NAK suppression by multicasting NAKs and staggering their transmission randomly, as in SRM [4]. The $\alpha$ parameter in the W-WBRM protocol is similar to $D$ in RMDP, except that we apply $\alpha$ to all transmissions, including sets of of parity packets sent in response to NAKs. Rizzo and Vicisano provide a detailed analysis of the parameter $D$, pointing out that the appropriate value depends on the loss rate and showing that values of $D$ between 1.5 and 2.0 make the probability of NAKs very low. Apparently, $D$ is fixed for a given environment, but the authors do discuss adaptability in terms of changing the value of $n$ at the encoder. By allowing the parameter $\alpha$ to adapt to loss conditions, the W-



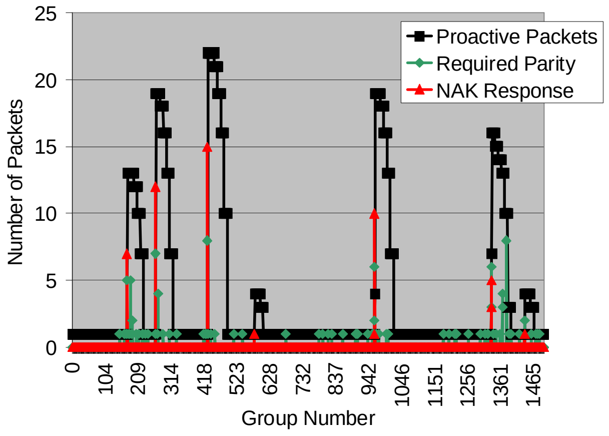
(a) sample trace of measured packet loss



(b) $\alpha_{\text{dec}} = 0.02$



(c) $\alpha_{\text{dec}} = 2^i(0.02)$, lower bound 0 parity packets



(d) $\alpha_{\text{dec}} = 2^i(0.02)$, lower bound 1 parity packet

**Figure 9. Experimental results for W-WBRM in mobile testbed, where errors are bursty.**

WBRM protocol quickly tunes the level of redundancy to match the needs of receivers.

Nonnenmacher et al. [13] present an extensive analysis of the relative merits of using an integrated FEC+ARQ protocol, compared to using a separate FEC layer beneath an ARQ-based protocol. The authors describe an integrated FEC-based multicast protocol, NP, for multicast data delivery in the Internet. The NP protocol tries to keep the number of packets transmitted to a minimum at the expense of latency, and does not send more parity packets than requested. However, the protocol uses pipelining of groups to improve throughput: the sender transmits data packets of group $i+1$ while waiting for NAK(s) for group $i$. NAKs are multicast, and SRM-like global NAK suppression is used to reduce feedback from receivers. The approach in W-WBRM protocol is more aggressive than that of NP, with a goal of achieving low latency for relatively small resources and good throughput for large ones. Hence, we are willing to error on the side of sending too many parity packets. Like NP, W-WBRM pipelines the transmission of groups, but uses proactive packets on both data and parity-only transmissions, instead of global NAK suppression.

Gemmell et al. [5] describe two FEC-based reliable multicast protocols to be used in one-to-many tele-presentations over the Internet. One of these, FCAST, is intended for bulk transfer of session-persistent data and does not involve sender feedback, but rather uses an FEC-based *carousel*. The other protocol, ECSRM, is closer in design to W-WBRM and is intended for multicasting dynamic data during a session. The ECSRM protocol uses both FEC and global NAK suppression. The parity packet suppression method of W-WBRM, used to avoid sending redundant parity packets in response to multiple NAKs, is similar to the method used in ECSRM. However, the ECSRM protocol does not use proactive transmission of parity packets, and does not adapt to changing loss conditions in the network.

Since ECSRM is designed for large groups on the Internet, with long round-trip delays, such adaptation may not be useful. W-WBRM, operating on a proxy in a local heterogeneous LAN environment, can make better use of a proactive adaptive mechanism.

## 7. Conclusions and Future Work

In this paper, we have described a study in the use of proxy services to support web-based collaboration when some of the participants are located on heterogeneous LANs. We demonstrated the integration of proxy services into the Pavilion middleware framework and described an FEC-based extension of the WBRM protocol. We showed that a proactive approach to sending parity packets can minimize feedback by dynamically adapting the rate of redundancy in response to changes in packet loss rate. Given the increasing presence of wireless LANs in homes and businesses, we envision immediate application of the proposed techniques to improve performance of collaborative applications involving users who roam within the range of a wireless access point. Topics of our ongoing and future work include: additional analysis and experimentation of various W-WBRM parameters, including $\alpha_{\rm inc}$ and $\alpha_{\rm dec}$; a simulation study to determine parameter settings for relatively large numbers of receivers; and a performance study of the W-WBRM protocol as used in Pocket Pavilion, our collaborative application for wireless handheld computers.

**Further Information.** A number of related papers and technical reports of the Communications Research Group at Michigan State University are available via the World-Wide Web at: `http://www.cse.msu.edu/~mckinley/crgweb`.

**Acknowledgements.** The authors would like to thank Robel Barrios, Aaron Malenfant, Jesus Arango, Ji Li, and Chiping Tang for their contributions to this work. This project was supported in part by the NSF grants CCR-9503838, CDA-9617310, and NCR-9706285, and CCR-9912407.

## References

[1] J. Arango and P. K. McKinley. VGuide: Design and performance evaluation of a synchronous collaborative virtual reality application. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, July 2000.

[2] B. R. Badrinath, A. Bakre, R. Marantz, and T. Imielinski. Handling mobile hosts: A case for indirect interaction. In *Proc. Fourth Workshop on Workstation Operating Systems*, Rosario, Washington, October 1993. IEEE.

[3] Y. Chawathe, S. Fink, S. McCanne, and E. Brewer. A proxy architecture for reliable multicast in heterogeneous environments. In *Proceedings of ACM Multimedia '98*, Bristol, UK, September 1998.

[4] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and applications level framing. In *Proceedings of SIGCOMM '95 (Cambridge, MA USA)*, pages 342–356, 1995.

[5] J. Gemmell, E. Schooler, and R. Kermode. A scalable multicast architecture for one-to-many telepresentations. In *Proceedings of IEEE International Conference on Multimedia Computing Systems*, pages 128–139, 1998.

[6] R. Kermode. Scoped Hybrid Automatic Repeat ReQuest with Forward Error Correction (SHARQFEC). In *Proceedings of ACM SIGCOMM*, September 1998. Vancouver, Canada.

[7] A. Koifman and S. Zabele. RAMP: A Reliable Adaptive Multicast Protocol. In *Proceedings of IEEE INFOCOM*, pages 1442–1451, March 1996.

[8] A. J. McAuley. Reliable broadband communications using burst erasure correcting code. In *Proceedings of ACM SIGCOMM*, pages 287–306, September 1990.

[9] P. K. McKinley, R. R. Barrios, and A. M. Malenfant. Design and performance evaluation of a Java-based multicast browser tool. In *Proceedings of the 19th International Conference on Distributed Computing Systems*, pages 314–322, Austin, Texas, 1999.

[10] P. K. McKinley and J. Li. Pocket Pavilion: Synchronous collaborative browsing for wireless handheld computers. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, July 2000.

[11] P. K. McKinley, A. M. Malenfant, and J. M. Arango. Pavilion: A distributed middleware framework for collaborative web-based applications. In *Proceedings of the ACM SIGGROUP Conference on Supporting Group Work*, pages 179–188, November 1999.

[12] P. K. McKinley and A. P. Mani. A study of proxy-based adaptive forward error correction for collaborative computing on wireless LANs. Technical Report MSU-CPS-00-01, Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan, February 2000.

[13] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. *IEEE/ACM Transactions on Networking*, 6(4):349–361, 1998.

[14] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, April 1997.

[15] L. Rizzo and L. Vicisano. RMDP: An FEC-based reliable multicast protocol for wireless environments. *ACM Mobile Computer and Communication Review*, 2(2), April 1998.

[16] D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas in Communications 15, 3*, pages 398–406, April 1997.