Improving Groupware Performance in Lossy Networks with Adaptive Forward Error Correction

Jeff Dyck and Carl Gutwin
HCI Lab, Computer Science Department, University of Saskatchewan
57 Campus Drive, Saskatoon, Saskatchewan S7N 5A9 Canada
+1 306 966 8646
jeff.dyck@usask.ca, carl.gutwin@usask.ca

ABSTRACT
Network delays can cause serious usability problems in synchronous groupware systems. Delays can often be reduced by switching to lightweight protocols, such as UDP, but this switch also sacrifices any guarantee of message reliability. To address this problem, we investigate the use of adaptive forward error correction (AFEC) for groupware. Adaptive FEC is a technique that can maintain a predefined level of reliability while avoiding the overhead of acknowledgements or retransmission. This paper describes an adaptive FEC technique designed to meet the needs of synchronous groupware. We ran an experiment that measured message reliability and latency using TCP, plain UDP, UDP with non-adaptive FEC, and UDP with adaptive FEC under several different simulated network conditions. Our results show that for message types that can tolerate some loss, such as awareness information, both FEC techniques keep latency at nearly the plain-UDP level while dramatically improving reliability. Adaptive FEC, however, is the only technique that can guarantee a level of reliability while minimizing delay as network conditions change.

Keywords
Synchronous groupware, network delay, latency, adaptive forward error correction, network protocols

INTRODUCTION
Synchronous distributed groupware systems often perform poorly and lag behind due to network delays. These delays can result in user errors, can inhibit interaction, and if the delay is large enough, can ultimately cause collaboration to break down [8]. Although it is impossible to eliminate lag completely, it can be reduced if networked applications can make more efficient use of available network resources.

Most groupware systems, however, are not particularly efficient users of the network. In particular, a main reason for delays in groupware is that most current systems send messages using heavyweight network protocols such as TCP. These protocols guarantee reliability, but increase delay in two ways. First, they increase network traffic by requiring that the receiver send an acknowledgement (ACK) every time a packet is received. If the ACK is not received by the sender within a certain amount of time, the sender resends the packet. Second, TCP guarantees in-order delivery, so if a packet is lost, all the packets that are currently in transmission must wait until the lost packet is resent and received. Thus, the reliability of protocols like TCP comes at the expense of additional network traffic and waiting for late packets, both of which increase delay.

One method for reducing network delays is to use lightweight transport protocols such as UDP that do not send acknowledgements and do not retransmit packets. The reduction in packet traffic (by half) and the avoidance of retransmit delays can substantially reduce latency. However, removing these mechanisms from the protocol also removes any reliability guarantees, and means that packet loss (and therefore lost messages between clients) will occur.

For certain types of messages in a groupware system, lack of reliability is not acceptable. For example, model updates, transactions, and control messages must arrive correctly for distributed systems to function properly. Other message types, however, are tolerant to some loss. In particular, awareness messages like telepointer updates are most often organized into streams, where the loss of one message in the stream does not prevent accurate interpretation of the information. Awareness messages are also much less tolerant of delay, since they encode the information that people are using to engage in moment-by-moment visual interactions like pointing, gesturing, and negotiating access to shared resources (e.g. [8]). Therefore, it seems reasonable to use faster, unreliable protocols for sending awareness messages.

The problem is to get the benefits of the faster protocol, but still manage to maintain control over message reliability. Even though awareness streams are loss-tolerant, they do have quality of service (QoS) requirements for loss that need to be maintained for the awareness techniques to be effective. For instance, a telepointer that is experiencing high amounts of loss makes it difficult to interpret the other person's activity, difficult to anticipate their actions, and

difficult to see and interpret their gestures [9]. Therefore, a minimum level of service is required to ensure that awareness information is being conveyed effectively.

Forward error correction (FEC) is a technique that improves reliability by duplicating recent messages in subsequent network packets. The inclusion of this seemingly-redundant information allows for loss recovery without requiring the overhead of protocols that guarantee delivery. Real-time streaming multimedia applications have shown that FEC is effective for increasing reliability while minimizing delays [11,1]. Since packet loss rates on real-world networks can be highly variable, researchers have recently proposed an adaptive extension to the FEC technique. Adaptive FEC (AFEC) monitors the current state of the network and dynamically modifies the amount of redundancy being added to packets, with the goal of meeting a predefined level of reliability. Several versions of AFEC have been proposed for delivering streaming multimedia and have been shown to be an optimal compromise between latency and reliability when network conditions are variable (e.g. [3,14]).

Real-time awareness information appears well suited to AFEC. Awareness information is similar to real-time streaming multimedia in that it is sensitive to delay, requires high throughput, and can tolerate some loss. In addition, awareness streams meet two of AFEC's other criteria: first, message frequency is high enough that recovered messages will still be useful to the receiver, and second, messages are compact enough to be added to subsequent packets without undue overhead.

This paper investigates the use of AFEC as a technique for reducing delay in groupware without sacrificing reliability. We first describe how FEC and AFEC work in general, and then introduce our version of AFEC that was designed specifically to meet the needs of groupware awareness messages. We then report on an experiment that compared the AFEC technique with non-adaptive FEC, simple UDP, and TCP in a simple telepointer application. The experiment showed that TCP is poorly suited for sending real-time awareness information, that both types of FEC are dramatically better than simple UDP, and that only AFEC can maintain a predefined level of reliability while minimizing delays under a variety of network conditions.

HOW FEC WORKS

FEC is a technique for improving reliability in unreliable protocols. It has been used extensively in real-time distributed systems where delays need to be minimized and data has some degree of loss tolerance, such as live streaming video and voice over IP (e.g. [3]). FEC is also commonly used in wireless and low bandwidth networks such as 802.11b, cellular packet systems, or satellite networks [1,4,7,12].

FEC works by adding redundant information to packets so that information from lost packets can be recovered (see Figure 1). In groupware systems, for example, we are interested in recovering lost awareness messages. When packet losses occur, the information in subsequent packets is used to rebuild the lost information without requiring retransmission (see Figure 2). However, recovery also implies that the latency of the recovered information will be higher, and so the receiver must have a means for making use of the recovered data for FEC to be a useful technique. Most multimedia applications do this using client-side buffering.

IP Header 20 bytes	**UDP Header** 8 bytes	**User ID** 4 bytes	**New message** variable length	**Redundant messages** variable length

Figure 1: An example message-based FEC packet.

Many varieties of FEC have been proposed and customized to suit the requirements of specific applications and network conditions, but all varieties have four main parts. The *coding scheme* determines how redundant information is added to packets. The *code set* is the portion of the packet containing the redundant information. The *code set size* is the amount of redundant information, which is predefined in non-adaptive FEC and varies in AFEC according to QoS requirements and network conditions. The *code selection algorithm* is used to determine which redundant information gets added to the code set [4].

Although FEC can recover from losses, fully guaranteed delivery is not possible using FEC alone. The amount of information that can be recovered depends on the coding scheme used and the type of data being sent. Since FEC relies on redundancy to recover packets, loss still occurs when multiple packets are lost in a row. When such a burst loss happens, the messages at the start of the burst cannot be recovered. Some systems combine FEC with retransmission and interleaving to recover from burst losses, but these techniques are expensive (e.g. [3]).

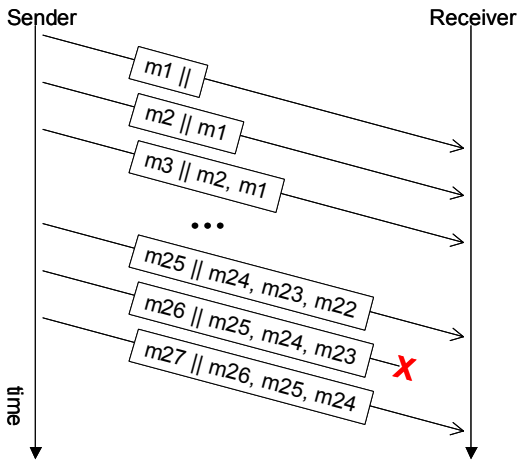

Figure 2: Diagram representation of a message-based FEC scheme with a code set size of three. The packet with message 26 is lost, but the message can be recovered from the next packet. Note that message loss only occurs when more than three packets are lost in a row.

Adaptive FEC involves only a small conceptual change from non-adaptive FEC. It works by using a coding scheme that dynamically changes the code set size based on network conditions. A message error rate (MER) range is specified by the user or by the application and the code set size is calculated to meet that MER. The receiver then monitors the MER and tells the sender when to increase or decrease the code set size. This feedback information is often sent through a separate reliable channel to ensure that the control signals arrive correctly.

Adaptive FEC must also consider the effects that its own actions have on network traffic. As the code set size increases, the size of each packet and thus the overall network traffic also increase, which can add to the packet loss problem. To compensate for the changes in bandwidth requirements, AFEC is often coupled with adaptive rate control that decreases the send rate as the code set size increases (e.g. [3,10]).

In cases where the sender cannot meet the QoS requirements because of rising loss rates or decreasing available bandwidth, the sender must notify the receiver and the receiver must lower the QoS specifications. QoS can be lowered either automatically by the application or by alerting the user and having them intervene to modify the requirements.

GROUPWARE CHARACTERISTICS THAT AFFECT FEC

Existing AFEC schemes are designed to meet the needs of multimedia applications where there is a single type of data being transmitted, where each packet contains only a portion of the data, where the total amount of data transmitted is large, and where packets are sent at a predictable rate over a period of time. Real-time groupware, however, differs from this scenario in four ways:

- most awareness messages are small (< 200 bytes)
- messages encode remote procedure calls rather than multimedia data
- a variety of different message types can be sent
- messages often occur in irregular clusters

The small size of groupware awareness messages is a tremendous advantage for the use of FEC. Small messages have low bandwidth requirements, allowing more redundancy to be added to each packet. This means that several previous messages can be sent with each packet, allowing recovery from small burst losses, which is expensive or impossible in multimedia applications.

Since groupware messages are usually remote procedure calls (RPCs), they are useless if not sent in their complete format. This means that two techniques that multimedia schemes use when adding redundancy – sending partial data or sending a lower resolution version of the data – will not work. Therefore, an encoding scheme for groupware should only include complete messages. This requirement can be met due to the small size of groupware messages.

Several different types of groupware messages can be sent within the same session, each with different QoS requirements for packet loss, update frequency, and delay. For example, telepointers and participant list information have different QoS requirements. Telepointers can tolerate a fairly high amount of loss, but require a high update frequency and minimum delay to support closely-coupled work [9]. In contrast, a participant list requires a much lower update frequency and is tolerant to higher amounts of delay, but is also less tolerant of loss. Different message types with different requirements means that the receiver will have to identify the types of messages that are lost, and track loss rates for each message type separately. The sender will have to use a code selection algorithm with multiple sets of QoS requirements rather than a single general set of service parameters.

The clustering of awareness messages in groupware reflects the way that people work: telepointer motion, for example, is most often a series of moves with stops in between, rather than consistent and continuous movement. Note that we assume that awareness messages are event-driven and so are only sent out when something is happening. For FEC, the variable frequency means that when losses occur at the end of a cluster (e.g. just before a mouse move stops), messages cannot be recovered until more messages are sent. To deal with this, senders must detect the end of a message cluster and send packets containing only redundant messages until the recovery requirements are met.

These four differences between groupware and multimedia applications are significant enough that we cannot apply an existing AFEC technique to groupware. Instead, we have devised a variation on the scheme that takes the characteristics of groupware messages into account. The new scheme is described below.

AFEC FOR GROUPWARE

We designed an AFEC technique that is tailored to meet the specific requirements of real-time groupware. This technique consists of a coding scheme, adaptive logic, and a code selection algorithm. Our technique is based on the assumptions that messages are small, that messages must be sent in their entirety, and that the receiver knows the minimum and maximum MER requirements for each message type.

The coding scheme for groupware AFEC is simple: redundant messages are included in their entirety and in their original form, rather than being distributed over several packets, compressed, or sent at lower resolution. This approach works because of the small size of awareness messages. Several redundant messages can be added to each packet, allowing full recovery of lost messages, even from small burst losses. The total number of redundant messages still varies, however, depending on the network bandwidth, the actual message size, and the message frequency.

Sending entire messages allows full recovery from a single packet. For example, adding just one redundant message allows full recovery from any single lost packet. If two packets are never lost in a row, this scheme would provide full reliability. In general, however, accommodating burst losses means that a scheme must include as many redundant messages in each packet as the maximum burst length, in order to prevent any message loss. In cases where this cannot be achieved, the number of messages lost during a burst is the number of packets lost during the burst minus the number of redundant messages in a packet.

Since it is very difficult to predict loss occurrence and burst length [2], it is not feasible to determine the number of redundant messages required to meet MER requirements ahead of time. Although MER can be reduced significantly using a fixed amount of redundancy, non-adaptive schemes cannot guarantee reliability. An adaptive approach is required to meet MER requirements while keeping the amount of redundancy to a minimum.

The need to accommodate MER requirements for a variety of message types also makes the adaptive logic for groupware more complex. In our technique, the receiver monitors each message type separately to ensure that MER requirements are being met, and tells the sender when to add or remove redundancy. The sender uses a code selection algorithm that considers each message type separately to meet the requested redundancy requirements. Since there can be many participants in a groupware system, MER is monitored separately for each user by the receiver and the redundancy requirements are tracked separately for each user by the sender.

In the next two sections we provide more details on the elements of groupware AFEC that are required in the receiver and the sender of a message (a schematic of these elements is shown in Figure 3).

Receiver. The receiver is responsible for monitoring MERs to ensure that they stay between the minimum and maximum values set out for each message type from each user[1]. Incoming packets consist of one current message and some number of redundant messages, all from a single user (see Figure 1). The messages in a packet are processed in reverse order from oldest to newest. This means that redundant messages are processed first. The receiver checks for message loss using message indexes (described below); if loss has occurred, the receiver increments the lost message counter for that message type. Any message that has not been seen before (including recovered and new messages) are then processed as required.

Message loss detection is performed using message indexes. The highest received index is stored for each message type, and if the index of a message exceeds this value by more than one, the number of lost messages is reported and the MER for that message type is updated. To be able to record MER correctly for each message type, we need to know what type of message was lost. Therefore, we specify type within the index using a two-character message type code that precedes the index number.

Current MER is calculated using the previous 1000 messages only, rather than all of the messages during the life of the session. This is necessary to allow the system to react to changes in network conditions and to have the effects of adjustments reflected quickly. Otherwise, the average would be spread out over the life of the session, resulting in reduced responsiveness as the number of messages in the session increases. The rolling average could also be accomplished using a time-based window; however, since the message frequency is irregular, a time-based approach could result in too small of a sample size to make accurate changes to the code set size.

The receiver is responsible for deciding when to increase or decrease the amount of redundant information being sent. If the MER exceeds the maximum MER, the code set size must be increased so that more messages can be recovered. Likewise, when the MER is below the minimum MER, the code set size must be decreased to avoid sending unnecessary redundancy (since it increases traffic without benefit). These control actions are accomplished with a module that periodically checks each message type's MER. If any of the MERs are above the maximum, a negative acknowledgement (NAK) is sent to the sender immediately, telling the sender to increase the amount of redundancy for the specified message type. If an MER is below its minimum value, the receiver sends an increase acknowledgement (INC) message, which tells the sender to reduce the redundancy for the specified message type[2].

Since groupware awareness messages are often clustered, code set size should only be updated when there is significant activity. Otherwise, if the MER is outside of its target range, it will not be changing fast enough, but the code set size would be updated repeatedly because it runs on a timed thread, resulting in overcompensation during low activity periods.

Sender. The sender is responsible for meeting the redundancy requirements as requested by the receiver. The

[1] One might ask why we would ever set a non-zero minimum for message error rate; the answer is that for information that can tolerate some loss without any problems for the users, the extra bandwidth needed to provide further reliability could be better allocated to another channel that can make better use of it (e.g. we may be able to send accompanying videoconference data at higher resolution).

[2] The term increase acknowledgement (INC) is from the typical coupling of rate control with FEC – decreasing the amount of redundant information increases the rate. We use this term to be consistent with prior usage [4].

code selection algorithm works to ensure that the correct number of redundant messages are included in each packet as requested by the receiver. The sender keeps a history buffer of each message and knows how many times each message has been sent. When loading messages into a packet, the encoding scheme starts from the most recent message and moves back through the history list. Only messages whose redundancy requirements have not yet been met are included.

The sender keeps a tally of the packet size and ensures that the amount of redundancy added does not exceed the path maximum transfer unit (path MTU) for the network route. The path MTU is the largest packet size that is forwarded by a router [13]. If the path MTU is exceeded, the packet will be divided into smaller packets, which adds delay. Therefore, the path MTU must be discovered by the system and should not be exceeded. In the case that adding redundancy will exceed the path MTU, the sender sends as many messages as it can without exceeding the path MTU and sends a notification to the receiver that the QoS requirements cannot be met and should be lowered. Adjustment of requirements can be handled automatically by the application or by the user through a dialog.

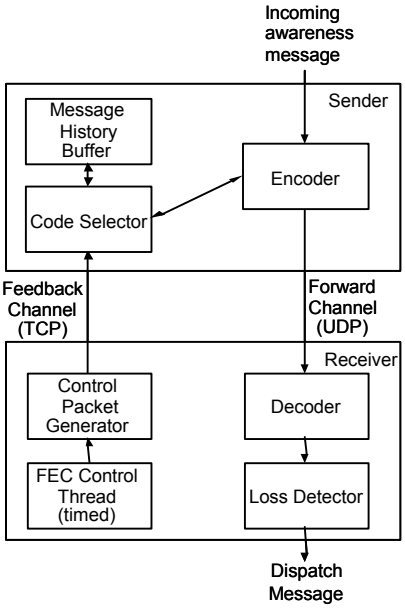

Figure 3: AFEC operation. Incoming messages are combined with redundant messages based on decisions made by the code selector. Encoded messages are sent via UDP and decoded at the receiver, which detects and reports any loss that occurs. A timed FEC control thread periodically verifies the MER of each message type and sends control packets to increase or decrease the amount of redundancy if the MER is outside of the specified range.

Finally, since groupware messages are often strongly clustered, times in which there are no messages to be sent are detected and a few extra messages are sent from the history buffer, in order to meet redundancy requirements for the last few messages of the cluster. Without adding messages after a cluster, MER would appear to be higher for the last few messages in each burst.

It is important to note that messages recovered using FEC or AFEC arrive at the same time as the newest message, which creates a similar effect as network jitter. User interface level solutions are required to deal appropriately with this late information. One method is to use client-side buffering, but this technique adds additional latency. Another method that is suitable for groupware is to use traces, a technique that visualizes recent awareness information, adding to accuracy without adding latency [9].

EXPERIMENTS

We ran a set of experiments to see how the proposed AFEC technique would compare with TCP, plain UDP, and non-adaptive fixed-length FEC schemes under a variety of network conditions. Network conditions were simulated on a LAN using The Cloud 3.0, a software-based network disabling emulator [15]. For each experiment, we measured MER and latency for each message. Our goals were to determine the following:

- How a guaranteed protocol like TCP compares with UDP-based schemes in lossy conditions;
- How FEC performance compares to UDP for a variety of code set sizes and bandwidth constraints;
- If and when non-adaptive FEC performs poorly;
- How AFEC compares with non-adaptive FEC;
- If and when AFEC performs poorly.

The test application was a simple telepointer widget with a telepointer trace [9]. We selected a common groupware awareness technique to ensure that the message patterns were similar to those found in real world groupware applications. A message trace was recorded by moving a mouse manually in a manner that simulated natural activity with a whiteboard. This simulated the clusters of messages that happen in a groupware application. The update frequency for the telepointer was fixed at a maximum rate of 30 updates/second for all tests.

Simulations using the trace were then run with two clients, both of which were on the same machine. All messages were sent through a server on a different machine on the LAN. The Cloud also ran on the client machine, simulating a variety of network conditions between the two clients. This enabled accurate message latency measurements using the system clock on the client machine.

Loss rates, loss patterns, and amount of bandwidth were specified using The Cloud to simulate different network conditions.

- *Loss rates*. We used three loss rates: 0%, 10%, and 20%. Zero loss simulates groupware use on a LAN or during low traffic times with a wired Internet connection. The 10% loss experiments simulate a dial-up, DSL, or cable connection during a high load period; loss rates in this range are common on the

Internet during peak times [2]. The 20% loss experiments simulated a lossy wireless connection where signal strength is low; loss rates over wireless connections are often in this range or even higher [12].

- *Loss patterns*. Two different loss patterns were used: random and burst. The random pattern loses packets randomly based on a percentage chance of loss. We ran random loss tests at 0%, 10%, and 20% loss. For burst loss, two parameters determined the loss rate: the number of packets that would be lost in a burst and the probability of a burst loss occurrence. Burst loss tests were run with 1-5 messages being lost with a 4% chance of starting on any packet (10% overall loss), 1-5 messages being lost with an 8% chance (20% overall), 1-10 messages being lost with a 2% chance (10% overall), and 1-10 messages being lost with a 4% chance (20% overall). Other loss models such as Gilbert-Elliot loss [6] will be added in future work.
- *Bandwidth*. Each experiment was also run using two bandwidth amounts: 56Kbps to simulate a dial-up line and 256Kbps to simulate a Uniform Cable connection. Broadband tests above 256Kbps performed similarly to the 256Kbps tests in all cases.

For non-adaptive FEC, we ran tests with a variety of code set sizes between 2 and 7. Since different code set sizes lead to different performance characteristics, we consider each as an independent technique, and will refer to them as 'FEC-n' where n is the code set size.

The experiments with AFEC were run with a target MER range of 5% minimum and 6% maximum. In a telepointer example, this means that we want to ensure that at least 95% of position messages arrive, but that more 96% is not needed for the type of interaction being supported.

For each experiment, we measured message latency and MER. A more in-depth review of all experiments and results is available at [5].

RESULTS

The results from our experiments are presented below, organized by the goals stated above – how different protocols perform in realistic conditions, the advantages and disadvantages of the basic FEC approach, and the behaviour and performance of groupware AFEC.

Comparing protocols in a realistic network situation

Our first investigation makes a basic comparison of all protocols in terms of message latency and message error rate. Figure 4 shows average latency in a 56Kbps channel with 10% random loss for TCP, UDP, FEC-3 (i.e. FEC with code set size of 3), and groupware AFEC. Figure 6 shows MER for the same set of conditions.

The most obvious result in Figure 4 is that when networks are experiencing loss, TCP's acknowledgment and retransmission policy lead to large latency. TCP latency is larger even in networks with no loss, but as loss increases, TCP quickly becomes unusable. In addition, the high variance in latency with TCP (see Figure 5) makes it very difficult for users to adapt to the delay.

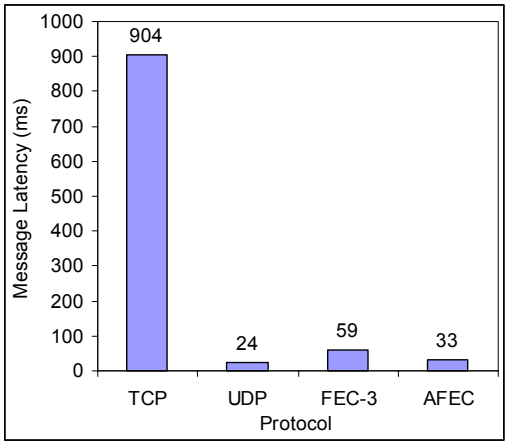

Figure 4: Average latency of different protocols in a 56Kbps channel with 10% random loss.

In contrast, all of the schemes based on UDP maintain a low average latency. Differences among these three schemes can be attributed to packet size and to the resultant traffic level, since higher traffic in a limited channel generally corresponds to higher latency. UDP has the smallest packet size (equivalent to FEC-1) and therefore generates the least traffic; FEC-3 has the largest packet in this scenario, and so has slightly higher latency.

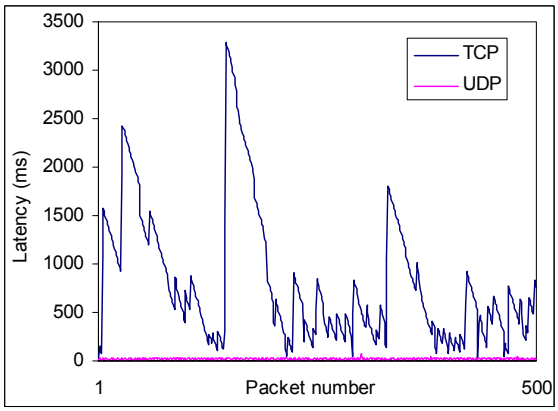

Figure 5: Typical variation in TCP latency (upper line) and UDP latency (lower line, near zero) in a 56Kbps channel with 10% random loss.

Results for message error rate (Figure 6) also show substantial differences between the protocols. Since TCP is a guaranteed reliable protocol, MER is always 0%. Since UDP provides no reliability control, its MER will always be approximately equal to the packet loss rate (here 10%). Groupware AFEC has an MER of 5%, which is within the bounds of the target MER of 5-6. FEC-3 in this case was highly reliable – in fact too reliable, in that it used bandwidth that could have been better allocated to a different information channel (such as a video stream).

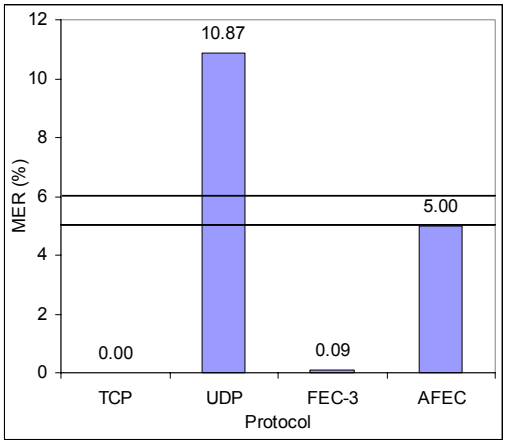

Figure 6: MER of different protocols in a 56Kbps channel with 10% random loss. The target bounds for MER (5%-6%) are marked with lines.

Performance of non-adaptive FEC

We tested non-adaptive FEC with several different code set sizes. In all tests, reliability was much higher than plain UDP, and in some experiments non-adaptive FEC showed the best performance for both MER and latency. However, these cases occurred only when the code set size happened to be an appropriate choice for the network conditions. When the code set size was not appropriate for conditions, then non-adaptive FEC showed either higher latency or an MER that was outside the target range.

One variable that greatly affected non-adaptive FEC was loss pattern. Random loss experiments resulted in very low MERs. Tests with burst loss, however, resulted in much higher MER values (e.g. MER of 8% for 20% packet loss). The poor performance occurred because messages are lost whenever the size of the burst exceeds the code set size.

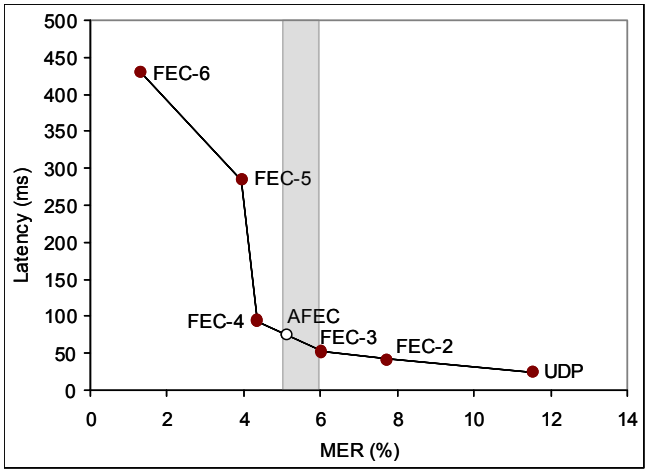

Figure 7. Average latency and MER for non-adaptive FEC with various code set sizes, and AFEC (white circle), on a 56Kbps channel with burst loss (burst size of 1-10 and 2% probability of occurrence).

In general, although non-adaptive FEC provides an enormous improvement over plain UDP, an inappropriate choice of code set size can lead to reduced performance. This can be seen in Figure 7, where, the latency and MER of FEC with various code set sizes is compared to AFEC. Although different code set sizes lead to different performance extremes, only a few values provide a balance between latency and error rate.

Performance and characteristics of AFEC

The performance of AFEC was always as good or better than that produced by non-adaptive FEC, as long as it was possible to meet the target MER range without exceeding the amount of bandwidth. AFEC shows optimal performance over time because it always moves towards the minimum code set size needed to meet the target MER range. In cases where non-adaptive FEC exceeded the reliability requirements (i.e. had a larger code set size), AFEC showed lower latency due to its smaller packet size. AFEC also selected a higher code set size to meet the target MER range in cases where FEC did not meet reliability requirements. The MER for AFEC varied between 4% and 7%, but was most often within the target range of 5% and 6%. In high burst loss conditions, AFEC was still able to meet the target MER range with low latency, as long as the amount of bandwidth was sufficient.

Figures 8 and 9 illustrate the differences and show how AFEC works. Figure 8 shows latency of various techniques as loss increases from zero to 20%. Because AFEC varies its code set size, it can perform as fast as UDP when loss is low. As loss increases, AFEC's increasing packet size leads to higher latency; however, this is the minimum latency possible when maintaining the QoS requirements for MER.

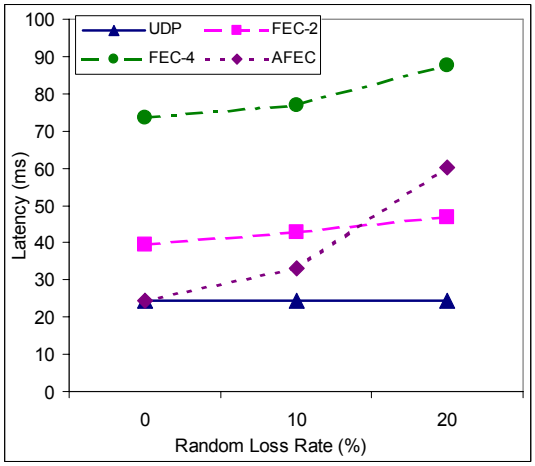

Figure 8. Average latency of different protocols at three loss rates. Note that AFEC latency increases as a result of increasing redundancy to maintain reliability.

The behaviour of AFEC is illustrated in the packet trace shown in Figure 9. At the start of the trace, AFEC has a code set size that is slightly too large, resulting in a gradual decline in MER past the specified minimum. Once below 5%, AFEC reduces code set size by one (line A in Figure 9); since this reduces packet size, latency decreases. The

reduction in code set size is not enough to prevent another decline past the minimum (line B), so AFEC reduces again. This results in low latency, but also in a code set that is slightly too small for network conditions, so MER climbs rapidly. When the MER is observed outside the maximum, code set size is increased (line C); this stops the increase in MER, but does not reduce it below the maximum, so another reduction is made (line D), also increasing latency.

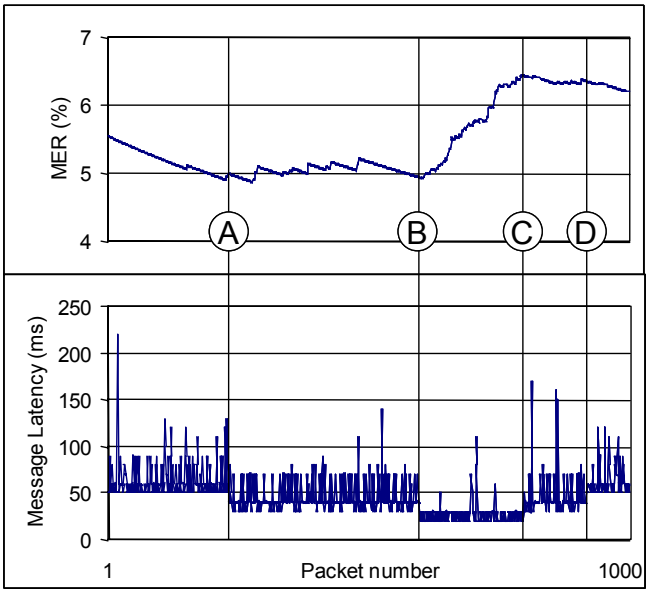

Figure 9: MER (upper) and per-message latency (lower) for 1000 AFEC packets, on a 56Kbps channel with 20% random loss. Vertical lines indicate points at which AFEC changed its code set size to stay within MER limits.

Note that the latency spikes in Figure 9 result from the latency of recovered messages, not from packet latency. Whenever messages are recovered, additional latency results from the time between when the message was first sent and when it is sent as redundant information. Thus message latency in AFEC is always slightly higher than packet latency.

Finally, our experiments showed certain conditions where the current AFEC technique cannot be used successfully. Since it is not coupled with adaptive rate control, certain high burst loss, low bandwidth conditions resulted in high amounts of latency. This situation arose when AFEC attempted to increase its code set size past the bandwidth limits in order to meet MER requirements. However, when used with rate control, this problem can be avoided.

DISCUSSION

These experiments compared the effectiveness of TCP, UDP, FEC, and AFEC for sending real-time awareness data under a variety of network conditions. The following conclusions can be drawn from our results:

- TCP is not suitable for sending awareness information over lossy networks due to very high latency;
- FEC produces substantial reliability increases over UDP alone without adding much latency in most cases;
- The optimal code set size is impossible to determine beforehand in most cases because of unpredictable network conditions;
- AFEC is able to meet a predefined MER range while minimizing the amount of latency, given a fixed message frequency;
- When it is not possible to meet the level of reliability without exceeding bandwidth requirements, AFEC should either decrease message frequency, decrease MER requirements, or accept higher latency.

Clearly, TCP is not suitable for sending real-time awareness data where guaranteed delivery is not required. The high amount of latency and jitter of TCP simply do not meet the requirements for real-time data under lossy conditions. Therefore, TCP should not be used for sending real-time awareness information.

Non-adaptive FEC is simple to implement and provides large reliability benefits over plain UDP. In general, a small code set size (e.g. 2) can be used to improve reliability considerably without adding substantial latency. However, a small code set size may not always provide the desired level of reliability, so under certain conditions, larger code set sizes may be better choices. Several factors affect the amount of latency added and the level of reliability attainable using FEC.

The amount of latency added by FEC depends on the available network bandwidth, message size, message frequency, number of users, and code set size. Under broadband conditions, large code set sizes add a trivial amount of latency. However, low bandwidth conditions can cause the problem described above, where code set size cannot be supported by the available bandwidth. Therefore, when clients have large amounts of bandwidth, a larger code set size is a good choice, but when bandwidth is low, the code set size must be set at a level that avoids exceeding the bandwidth capabilities.

The level of reliability provided by FEC depends on the loss type, the loss rate, and the code set size. A larger code set size always results in the same or better reliability because it allows more subsequently lost messages to be recovered. Higher loss rates always result in the same or worse reliability because the frequency of losses is higher. Random losses where few packets in a row are lost result in high reliability from FEC, while longer bursts result in lower levels of reliability. Therefore, when long, frequent burst losses are occurring, a larger code set size is desirable, but when only short bursts occur less frequently, a smaller code set size is sufficient.

The problem with FEC is that the application programmer must set the code set at a fixed size, but an optimal choice depends on several unpredictable parameters: loss rate, loss pattern, and available bandwidth. If the programmer knew the network conditions beforehand, an optimal code set size could be selected. Since this is not possible, the application programmer is left with the problem of

determining the pros and cons of each code set size. Therefore, the code set size must be selected conservatively and without certainty based on the cost-benefit analysis of reliability vs. latency under unknown conditions.

AFEC improves on non-adaptive FEC by dynamically adapting its code set size to meet the MER requirements for each technique. This removes the uncertainty that results from choosing a fixed code set size for non-adaptive FEC. AFEC adjusts the code set size to the minimum that still maintains the desired MER range. In cases where non-adaptive FEC exceeds the required level of reliability, AFEC will always produce lower latency. When non-adaptive FEC does not meet the MER requirements, AFEC will automatically increase its code set size to meet the reliability requirements, as long as this does not require exceeding the path MTU.

The only remaining problem with AFEC is that it is sometimes not possible to meet the desired level of reliability at a certain message frequency. In these cases, either rate can be decreased or MER requirements can be lowered. The AFEC algorithm presented here does not include adaptive rate control, although this can be added in future work.

Overall, AFEC is the most effective technique for sending real-time awareness information. It is not trivial to implement, so if timelines for implementation do not allow AFEC to be implemented, non-adaptive FEC should be used with a conservative amount of redundancy. AFEC should be considered by groupware toolkit designers, as it provides substantial benefits under a wide range of conditions and for a wide range of groupware application types, and because its complexity would be best abstracted away from application programmers.

FUTURE WORK

Several additional investigations will follow on from this research. Among these are: investigating the effects of trading off parameters such as latency, reliability, and frequency in real groupware systems; improving the performance of the adaptive control algorithm; and exploring other adaptive techniques that could be coupled with AFEC to create further efficiency gains.

Tradeoffs and groupware QoS. The adaptive algorithm presented here is forced to make tradeoffs between latency, reliability, and message frequency. Under some network conditions, it is not clear what choices will best preserve the usability of the groupware system for its users. QoS parameters for groupware have not been investigated and the effects on the user of modifying QoS parameters are generally unknown. For example, coupling AFEC with rate control is common in streaming video, but the effects on the user of decreasing rate at the expense of reliability are unknown in groupware. To provide information to guide an adaptive strategy in making these tradeoffs, we are developing a QoS model for groupware that will identify parameters that can be adapted and will determine how the parameters interact in different usage situations.

AFEC improvements. The adaptive algorithm presented here could be improved in several ways. One problem is that the code set size is currently an integer value, which can lead to the 'up and down' behaviour seen in Figure 9. We are currently building a finer-grained solution that sends extra redundant messages but not in every packet, resulting in fractional values for code set size that can be more finely tuned to the current network conditions.

Adding other techniques. Other adaptive techniques could be coupled with AFEC to produce further benefits. We are currently exploring adaptive rate control, adaptive concurrency policies, adaptive subscription to awareness information at a fine-grained level, and dynamic load balancing for groupware. The relationships between these techniques are complex and the effects that they have on each other and on the user are generally unknown.

CONCLUSION

Our experiments show that AFEC is an effective technique for sending real-time awareness information because it meets reliability requirements while minimizing latency under lossy conditions. AFEC works by adaptively adjusting the amount of redundancy in packets so that lost messages can be recovered without requiring retransmission. Groupware messages are generally small, which allows several redundant messages to be added to each packet. This allows AFEC to recover lost messages, even when burst losses occur.

AFEC for groupware is considerably different from AFEC for multimedia due to several key differences between the application types. Since groupware sends many different message types with different reliability requirements, multiple message type support was required for groupware AFEC. The irregular bursts of messages that occur in groupware were compensated for by detecting breaks between bursts and sending redundant information at the end of the bursts. Groupware's small remote procedure calls made it appropriate to send complete messages rather than compressed portions of historic messages.

Results of experiments comparing several protocols in realistic lossy network conditions show that non-adaptive FEC improves on the reliability of UDP, but code set size must be chosen carefully. In dynamic network conditions, however, this choice is impossible to make correctly, whereas AFEC is able to move towards the optimal value in most cases. Our experiments also showed that TCP is unsuitable for sending real-time awareness information, especially under lossy conditions.

The problem of heterogeneity in groupware is becoming more severe with the appearance of mobile wireless devices. As a result, an adaptive approach to groupware has become necessary to facilitate collaboration among clients with different amounts of available resources. AFEC is an

adaptive technique that can help groupware systems to cope with heterogeneous clients and dynamically changing environments. AFEC and other adaptive techniques can help to preserve interaction richness and groupware usability in dynamic, heterogeneous environments.

REFERENCES

1. Alagöz, F., Walters, D., Alrustamani, A., Vojcic, B., Pickholtz, R. On the effects of adaptive forward error correction mechanism in direct broadcast satellite networks. *Proc. 2nd ACM Workshop on modeling, analysis and simulation of wireless and mobile systems*, August 1999.
2. Bolot, J-C End-to-end packet delay and loss behavior in the Internet, *Computer Communication Review, ACM SIGCOMM '93*, 23(4):289–298, Sept. 1993.
3. Bolot, J-C, Fosse-Parisis, S., Towsley, D. Adaptive FEC-Based error control for Internet Telephony, *Proc. Infocom '99*, New York, NY, March 1999.
4. Cho, S., Goulart, A., Akyildiz, I. F., Jayant, N. An Adaptive FEC with QoS Provisioning for Real-Time Traffic in LEO Satellite Networks, in *Proc. of IEEE ICC 2001*, Helsinki, Finland, pp. 2938-2942, June 2001.
5. Dyck, J., Gutwin, C. A comparison of network protocols for message passing in groupware, University Of Saskatchewan HCI Lab Research Report 02-03. 2002.
6. Ebert, J.-P., Willig, A. A Gilbert-Elliot Bit Error Model and the Efficient Use in Packet Level Simulation. *Technical Report TKN-99-002*, Telecommunication Networks Group, Technische Universität Berlin, 1999.
7. Eckhardt, D., Steenkiste, P. A Trace-based Evaluation of Adaptive Error Correction for a Wireless Local Area Network. *Mobile Networks and Applications (MONET)* 4:4, ACM/Baltzer Science Publishers, 1999.
8. Gutwin, C. (2001) Effects of Network Delay on Group Work in Shared Workspaces. *Proc. ECSCW 2001*.
9. Gutwin, C. (2002) Traces: Visualizing the Immediate Past to Support Group Interaction. To appear, *Proc. Graphics Interface 2002*.
10. Hsu, C.-Y., Ortega, A., Kanshari, M. Rate control for robust video transmission over wireless channels. In *Visual Communications and Image Processing*, San Jose, February 1997.
11. Liu, H., Ma, H., El Zarki, M., Gupta, S. Error Control Schemes For Networks**.** *Mobile Networks and Applications*. Volume 2 Issue 2. October 1997.
12. McKinley, P., Gaurav, S. Experimental evaluation of forward error correction on multicast audio streams in wireless LANs. Proc. 8th ACM international conference on communications, October 2000.
13. Mogul, J.C., Deering, S.E. Path MTU Discovery. *Technical Report RFC 791*, September 1981.
14. Park, K. AFEC: an adaptive forward error-correction protocol and its analysis. *Technical Report CSD-TR97 - 038*, Computer Science Dept., Purdue University, 1997. http://citeseer.nj.nec.com/article/park97afec.html
15. Shunra Software Ltd. "The Cloud WAN Emulator: User Manual: Version 3.0." www.shunra.com/PDFs/ TheCloudManual.pdf. 2000.