Adaptive Use of Error-Correcting Codes for Real-time Communication in Wireless Networks *

Moncef Elaoud and Parameswaran Ramanathan
Department of Electrical and Computer Engineering
University of Wisconsin, Madison, WI 53706–1691
elaoud@ece.wisc.edu, parmesh@ece.wisc.edu

Abstract

The growing demand for mobility has increased the need to develop more efficient, reliable and cost effective services for data transmission over the air interface. The air interface, as compared to the wired domain, is characterized by a high bit-error-rate, limited bandwidth, and intermittent connectivity. In addition, power in the hand-held devices and laptops is limited by the battery technology.

In this paper, we present AFEC, an Adaptive Forward Error-Correction scheme, which makes effective use of the air interface by minimizing the number of data bits transmitted to convey message packets in a real time stream over an air interface. Through simulations, we show that AFEC outperforms the traditional single forward-error-correction scheme.

1 Introduction

Users of portable devices increasingly want mobility while continuing to access and exchange information on the global communications infrastructure. This demand is being met by rapid advances in the wireless technology. Unlike in the wired domain, where high speed, high reliability, and low bit-error rates are customary, the communication links based on the wireless technology have limited bandwidth, intermittent connectivity, and high bit-error rates. Considerable research effort is presently directed towards developing techniques which make efficient use of the limited wireless bandwidth. The technique proposed in this paper is also based on this objective.

More specifically, in this paper, we focus on the problem of real-time communication over a wireless link. Unlike in non-real-time applications, message packets in real-time applications like video-conferencing, video-phone, and flexible manufacturing system have deadline constraints by which they are expected to be delivered to the destination. Message packets which do not reach the destination on time contain stale information and must be discarded by the application. This results in a deterioration in the quality of service perceived by the application.

Techniques in the literature for real-time communication in the wired domain rely on a combination of admission control, resource reservation, and scheduling to guarantee the deadlines of message packets (see [12] for key references). Since the bit-error rates are extremely low in the wired domain, these techniques assume that all packets reach their destination error-free. This assumption, however, does not hold in the wireless domain because of the high-bit-error rates. In this paper, we consider the problem of real-time communication under high bit-error rates.

Existing solutions to deal with the high bit-error rates of a wireless link involve use of error-correcting codes and retransmissions. For example, in [6, 11], the solution relies on multiple retransmissions to convey the packet correctly across the wireless interface. The solution in [5], sends multiple copies of the packet at once in the hope that at least one of these copies will be correctly received. These schemes do not focus on reducing the bandwidth or the power consumed in the mobile host to reliably convey these packets.

To reduce the bandwidth required to transmit the packets reliably, Badrinath and Sudame introduced the "send or not to send" concept [1]. In this scheme, the sender monitors the status of the air interface between the sender and the receiver. If the air interface is judged to be noisy, the sender defers transmitting the packet to a later time. A different approach was proposed in [9]. Here, the sender monitors the

* The work reported here is supported in part by the National Science Foundation grant MIP-9526761.

status of the air interface to identify the best error-correcting code to maximize the channel goodput for a given bound on the probability of the packet having non-correctable errors.

In this paper, we adapt and combine the ideas in [1] and [9] to deal with packets with deadline constraints. However, unlike in [1], the decision to defer transmission of a packet is made based on both the deadline of the packet and the status of the air interface. Furthermore, depending on the deadline and the status of the air interface, our solution also selects the best error-correcting code from a given set of codes with an objective of balancing the need for delivering the packet on time with the need to conserve bandwidth. Here again, unlike in [9], the deadline of the packet is crucial in selecting the error-correcting code. The decision to defer and the selection of the best error-correcting code in our solution, is done using a simple table-driven approach. The entries in this table are optimally determined once during initialization of the application using dynamic programming. Simulation results show that the proposed strategy is very effective in reducing the bandwidth without jeopardizing the probability of packets meeting their deadlines.

The rest of this paper is organized as follows. A formal description of the assumptions and the model considered in this paper are presented in Section 2. The problem statement and the proposed solution along with some implementation issues are presented in Section 3. The simulation results are presented in Section 4. The paper concludes with Section 5.

2 System Model

Model of a real-time message stream. The scheme proposed in this paper does not depend on the direction of the communication. Therefore, we use the generic terms, *sender* and *receiver* in the rest of this paper. Due to the timing constraints within the application running on the mobile host, each packet in the stream has a deadline constraint by which it is expected to be delivered to the receiver. If a packet does not reach the receiver within the deadline, we assume that there is a deterioration in the quality of service perceived by the application. The amount of deterioration depends on the type of the packet. For example, in an MPEG video stream, a tardy I-type packet will cause more deterioration than a tardy P-type packet, which in turn will cause more deterioration than a tardy B-type packet [4]. In this paper, we model these differences between packet types by associating a different reward with each type of packet. The reward represents the positive impact on the quality of service perceived by the application as a result of the packet being delivered to the application on time.

Let $\mathcal{T}$ be the set of packet types in a real-time message stream. For each type $\tau \in \mathcal{T}$, let $\Re(\tau) \geq 0$ denote the associated reward.

Model of the error-recovery scheme. The two commonly used techniques for error-recovery are Automatic Repeat Request (ARQ) and Forward Error Correction (FEC). In ARQ, the sender transmits a packet and waits for an acknowledgment from the receiver. If an acknowledgment is not received within a pre-determined time period or if a negative acknowledgment is received, the packet is retransmitted by the sender. FEC schemes, on the other hand, incorporate redundancy into the packets so that the receiver can directly correct the errors which occur during transmission. The redundancy introduced into the packets is derived using error-correcting codes. In this paper, we assume that the sender uses a combination of ARQ and FEC scheme. However, since in a real-time message stream, there is no use in delivering packets past their deadlines, the ARQ scheme does not retransmit packets whose deadlines have expired. The FEC is done using Reed Solomon (RS) codes.

Also, the traditional assumption in literature is that the sender uses the same error-correcting code in each of its attempt to deliver a packet on time. In contrast, in this paper, we assume the sender can use any one of a given set of error-correcting codes, $\mathcal{C} = \{c_0, c_1, \ldots, c_q\}$, in each of its transmissions. We assume that code c_{i+1} can correct more errors than code c_i, $0 \leq i \leq q-1$. Correspondingly, for a given number of message bits, the number of check bits required for code c_{i+1} is more than that for code c_i. We let $\psi_L(c_i)$ be the total number of bits to be transmitted when a packet of L message bits is encoded using code c_i. By our earlier assumption, $\psi_L(c_{i+1}) \geq \psi_L(c_i)$ for $0 \leq i \leq q$.

In the proposed approach, the sender can also choose to defer the transmission of a packet. For simplicity of presentation, we let $c_0 \in \mathcal{C}$ represent the decision to defer, i.e., code c_0 is a special code with $\psi_L(c_0) = 0$, but with zero probability of correct delivery. Clearly, if the sender uses more than one error-correcting code in its attempts to correctly deliver a packet, the packet must be encoded using multiple codes. This adds to computation overhead, which in turn may increase the energy consumed. However, Rizzo et al. [8] show that software FEC is not exceedingly expensive. The decrease in energy consumed due to the bandwidth reduction of the proposed scheme will most likely compensate for the the increase in the energy consumed by the additional computation.

Model of the air interface.

We assume that the quality of the air interface between the base station and the mobile host is modeled as an S state Markov chain [3, 7, 10]. The states represent different levels of the "quality" of the signal between the host and the base station. The states in this chain are denoted by $0, 1, \ldots, S-1$. We define $t_{i,j}$ to denote the transition probability from state i to state j, $0 \leq i,j \leq S-1$. Furthermore, a packet transmission is said to be *correctable* if the associated error-correcting code can correct the errors which occurred during its transmission. Let $p_{i,c}$ be the probability of a correctable transmission when the air interface is in state i and the packet is encoded using the error-correcting code $c \in \mathcal{C}$.

Without loss of generality, we assume that state i is clearer than state $i+1$, $0 \leq i \leq S-2$, in the sense that, for any error-correcting code $c \in \mathcal{C}$, $p_{i,c} \geq p_{i+1,c}$.

3 Problem Statement & Solution

The problem addressed in this paper can be stated as follows. The sender has a packet of type $\tau \in \mathcal{T}$ for transmission. The sender knows the packet's length L, its deadline D, and the associated reward $\Re(\tau)$ for delivering the packet on time. The problem is to develop a strategy for relaying the packet to the receiver so as to maximize the expected net profit.

More formally, let TBT be the total number of bits transmitted by the sender in conveying this packet to the receiver. This includes one or more transmissions of the message bits and the associated check bits. Note that, if multiple transmissions occur, then the number of check bits in each transmission may be different depending on the code used. Then, the net profit

$$\eta = \begin{cases} \Re(\tau) - \text{TBT} & \text{if packet meets its deadline} \\ -\text{TBT} & \text{otherwise.} \end{cases}$$

The problem is to develop a transmission strategy for maximizing the expected value of η.

3.1 Solution approach

To provide quality of service guarantees, most present day networks use a connection-oriented approach in which an application must first establish a *connection* between the sender and the receiver before it can start sending/receiving packets [12]. The solution we propose is an Adaptive Forward Error Correction (AFEC) scheme. The first part of this scheme is performed at connection establishment, while the second part is used at run time when the application is sending a stream of packets.

3.1.1 Connection establishment phase

During the connection establishment phase, the sender computes a table of codes called the *Code Lookup Table* (CLT). The CLT is indexed using three inputs: (i) an estimate of the number of transmissions of the packet possible prior to its deadline, (ii) the state of the air interface, and (iii) the type of the packet. This table is used at run time to determine the optimal code for transmission. The method for computing the entries in this table is based on the following two observations.

Observation 1 *Consider an L-bit packet of type $\tau \in \mathcal{T}$. Suppose that at most one transmission is possible prior to the packet's deadline. Also suppose that the state of the air interface is s. Then, the expected net profit if the sender uses code $c \in \mathcal{C}$ is*

$$\phi_c(1, s, \tau) = \Re(\tau) \cdot p_{s,c} - \psi_L(c) \cdot b_c, \qquad (1)$$

where b_c is the cost of transmitting a bit over the air interface.

The justification for this observation is as follows. Since there is possibility for only one transmission, the delivery of the packet is successful only if the transmission resulted in correctable errors. When the air interface is in state s, code c will result in a correctable transmission with probability $p_{s,c}$. Therefore, there is probability $p_{s,c}$ that the transmission results in a reward of $\Re(\tau)$. Since the cost incurred by the sender in using code c is $\psi_L(c) \cdot b_c$, the expected net profit is as given by Equation (1). From this observation, the best possible code to use for this packet and the optimal expected net profit are respectively

$$\begin{aligned} opt\_code(1, s, \tau) &= \arg\max_{c \in \mathcal{C}} \phi_c(1, s, \tau). & (2) \\ \Phi(1, s, \tau) &= \max_{c \in \mathcal{C}} \phi_c(1, s, \tau). & (3) \end{aligned}$$

Observation 2 *Suppose $m \geq 2$ transmissions are possible prior to the packet's deadline and the state of the air interface is s. Then, if the sender uses code $c \in \mathcal{C}$, the expected net profit and the optimal code for transmission of a packet of type τ with length L are*

$$\phi_c(m, s, \tau) \quad = \quad \Re(\tau) \cdot p_{s,c} - \psi_L(c) \cdot b_c$$

$$+(1-p_{s,c}) \cdot \sum_{j=0}^{S-1} \Phi(m-1,j,\tau) \cdot t_{s,j}$$

$$opt\_code(m,s,\tau) = \arg\max_{c \in \mathcal{C}} \phi_c(m,s,\tau).$$

The optimal expected net profit can therefore be obtained by recursively solving

$$\Phi(m,s,\tau) = \max_{c \in \mathcal{C}} \{\Re(\tau) \cdot p_{s,c} - \psi_L(c) \cdot b_c + (1-p_{s,c}) \sum_{j=0}^{S-1} \Phi(m-1,j,\tau) t_{s,j}\},$$

for $m \geq 2$, with $\Phi(1,s,\tau)$ as given by Equation (3) for all s and τ.

Numerical example. Assume that the state of the air interface is characterized by a two-state Markov chain. Label the states as *noisy* and *clear*. Let the transition probabilities from *clear* to *noisy* and from *noisy* to *clear* states be 0.096 and 0.904, respectively.

Assume that the stream has only one type of packet, and all packets are of length 200 bytes. Further,let the reward associated with each packet be 250 and suppose that the sender has access to the codes $\mathcal{C} = \{c_0, c_1, c_2\}$ where c_0 corresponds to deferment, c_1 is (255, 247) RS code, and c_2 is (255, 239) RS code. Let the cost for using codes c_0, c_1, and c_2 be 0, 208, and 216, respectively. Also, let the probability of correctable transmissions for codes c_1 and c_2 in the *clear* state be 1.0, and in the *noisy* state be 0.59 and 0.96 respectively.

Now suppose the deadline of a packet only allows for a single transmission, and the current state of the air interface is *clear*. Then, the expected net profit $\phi_c(1, clear, 1)$ for delivering this packet using code c is the product of the reward and probability of correctable transmission minus the cost required for transmission. Hence, the net expected profit for c_0, c_1, and c_2, are 0, 52, and 44, respectively. Therefore, c_1 is the optimal code and the maximum expected net profit is 52. Hence, the entry in the CLT for state *clear* and $max\_tx = 1$ is c_1 (see Table 1). Similar, calculations can be used to compute the entry in the CLT for state *noisy* and $max\_tx = 1$.

Now suppose that, the packet deadline will allow for two transmissions, and the state of the air interface is *noisy*. Then for any code $c \in \mathcal{C}$ the total expected net profit is:

$$\phi_c(2,ns,1) = \Re(1)p_{c,ns} + (1-p_{c,ns})(\Phi(1,ns,1) \cdot t_{ns,ns} + t_{ns,cl}\Phi(1,cl,1)) \quad (4)$$

Air state	*max_tx* 1	2	3
noisy	c_2	c_0	c_0
clear	c_1	c_1	c_1

Table 1: Code Lookup Table for the example in Section 4.2.1.

where for simplicity of notation *ns* denotes *noisy* and *cl* denotes *clear*. Using Equation (4), the net expected profits for c_0, c_1, and c_2 are 50.23, −34, and 35.60 respectively. Therefore, the optimal code in this case is c_0 or deferring the transmission at the current time. The rationale behind the deferment is that there is a good possibility,0.909, that the next time around the state of the air interface will be *clear*. The rest of the entries of Table 1 are calculated in a similar fashion.

3.1.2 Run time phase

At run time, the sender periodically monitors the state of air interface. Now suppose a packet of type τ having a deadline of D is ready for transmission at time t. Then, the sender executes the following steps in order to convey the packet to the receiver.

1. Estimate the maximum number of possible retransmissions prior to the deadline of the packet as $max\_tx = \lfloor \frac{D-t}{T} \rfloor$. Where T is the retransmission timeout period. If $max\_tx \leq 0$ drop the packet.
2. Using the state of the air interface, type of the packet, and *max_tx* as an index into the CLT obtain the optimal code *opt_code* for the current transmission.
3. If *opt_code* is c_0, then defer transmission of the packet. Otherwise, transmit the packet after encoding it using *opt_code*. In either case, set the timeout timer to T.
4. If the timer expires prior to the receipt of an acknowledgment, go back to step 1 to consider retransmission of the packet.

3.1.3 Implementation Issues

Zero stuffing. As is customary in any FEC scheme, if the packet length is less than k blocks, the sender can append a sequence of zeroes to bring the total block size to k, then use an (n,k) RS encoder to determine the $(n-k)$ check symbols. Precious wireless bandwidth can be saved by not transmitting the appended zeroes over the air interface. This requires the receiver to be made aware of the number of zeroes to append prior to decoding (see discussion below on *air-link-packet*).

air-link-packet. We assume that the message packet and the check symbols are encapsulated in an *air-link-packet* containing a header specifying the error-correcting code used, length of the message packet, and a header checksum. The receiver uses the checksum to determine whether the header is corrupted. If it is corrupted, the entire *air-link-packet* is assumed to be not correctable. Otherwise, from the information in the length and the code fields of the *air-link-packet*, the receiver decodes the message packet and if possible corrects the errors that occurred.

Selection of model parameters. The proposed scheme relies on the model of the air interface. The parameters of interest are: (i) the transition probability between the states, and the (ii) the probability of a correctable transmission for each code in $\mathcal{C}$ for a packet of a given length. The transition probability can be estimated by periodically monitoring the bit-error rate over the air interface. Both the base station and the mobile host usually monitor the quality of air interface to make other important decisions like handover. If each state is characterized by a range of bit-error rates, then the transition probabilities can be estimated by measuring the relative frequency of changes from one state to the other.

The probability of a correctable transmission for a packet of length L using a code $c \in \mathcal{C}$ can be estimated experimentally. The sender can maintain the fraction of correctable transmissions for each state of the air interface. The sender can also analytically estimate the probabilities as follows. Consider a state s of the air interface. Let avg_ber be the average bit-error rate in state s. Let (n, k) be the RS code under consideration and let sym_len be the number of bits in each symbol of the (n, k) RS code. Also, let ℓ be the number of symbols in the message packet. Then, the probability of a correctable transmission, pct, is estimated as

$$\begin{aligned} \text{pse} &= 1-(1-\text{avg\_ber})^{\text{sym\_len}} \\ \text{pct} &= \sum_{i=0}^{(n-k)/2} \binom{\ell}{i} (\text{pse})^i (1-\text{pse})^{\ell-i}. \end{aligned} \quad (5)$$

In the above equation, pse, is the probability of symbol error, pct is the probability of fewer than $(n-k)/2$ symbol errors in a packet of ℓ symbols.

4 Empirical Evaluation

In this section, we evaluate the effectiveness of AFEC. The evaluation is carried out using a discrete-event simulator based on the OPNET™ network modeling tool. The bandwidth of the air interface is assumed to be 1 Mbps.

The inputs to the simulator are the model of the air interface, characteristics of the real-time-message stream and the associated error-correcting codes. In the simulator, the air interface is modeled at a level of detail greater than required for implementing the proposed adaptive scheme. This is done to mimic the situation that will occur in practice where the bit-error rate varies over time and the exact values of the stochastic parameters of this variation are not known to the sender or the receiver. However, the sender can monitor the past variation in the bit-error rate and estimate the stochastic parameter values necessary to make the code selection in the adaptive approach. Clearly, there will be in uncertainties in the parameter values estimated by the sender. Our evaluation shows that the performance of the proposed scheme is not very sensitive to the errors in the estimated values [2].

The study of this effectiveness is carried out for two different types of real-time message streams. The first is a stream of ATM cells, each with a deadline constraint. The second is a stream of deadline-constrained IP packets. We choose error-correcting codes appropriate to the type of the real-time message stream.

4.1 Simulation Method

Model of the air interface in the simulator. In the results presented here, the air interface between the mobile host and the base station is modeled using a two-state Markov chain. We refer to the states as *clear* and *noisy*. The time spent in the *clear* and the *noisy* states are taken from exponential distributions with rates λ_c and λ_n respectively. For the simulation results presented here, the average length of the *clear* state is 330 ms and the bit error rate at each time instant is taken from a uniform distribution in the range of 0 to 10^{-5}. When the air interface is in the *noisy* state, the bit-error rate follows a bell-shaped function. More specifically, the time spent in the *noisy* state, say μ, is first picked from an exponential distribution with rate $\lambda_n = \frac{1}{35ms}$. Then, the bit-error rate at time t, $t_0 \le t \le t_0 + \mu$ is $ber(t) = 0.02 \times e^{-6(\frac{t-t_0}{\mu}-1)^2}$, where

[0]OPNET is a trademark of MIL 3, Inc.

t_0 is the time at which the air interface switches to the *noisy* state. Note that, at time $t_0+\mu$, the air interface switches back to *clear* state.

Simulation process. We assume that the sender has an infinite stream of message packets to convey to the receiver. The message packet is either an ATM cell or an IP packet depending on the real-time message stream. As described in Section 3.1.3, the message packet is encapsulated in an *air-link-packet.*

When the receiver gets an *air-link-packet*, it must determine whether the errors in the *air-link-packet* are correctable. In the simulator, this is carried out as follows. From the bit-error function, the probability of a correctable transmission, pct, is first computed using Equation(5). The receiver then declares the packet to be correctable with probability pct.

Performance metrics. For each scheme simulated, the simulator considers the transmissions of a large number of message packets from the sender to the receiver. Let N be the total number of message packets considered in the simulation. For each of these packets, the simulator maintains the relevant statistics. From these statistics, the simulator computes the following three performance metrics: percent bandwidth overhead (B), miss rate (M), and the normalized net profit (P). These three metrics are defined as follows.

Consider the i^{th} message packet in the simulation. Define the TBT(i) to be the total number of bits transmitted by the sender to relay the packet to the receiver. This includes one or more transmissions of the message packet, the associated check bits and the *air-link-packet* headers. Note that, if multiple transmissions occur, the number of check bits in each transmission may be different in the AFEC depending on the error-correcting code used. Let Bits_C1(i) be the number of bits in the *air-link-packet* when the i^{th} message packet is encoded using code c_1. Note that, Bits_C1(i) is the minimum number of bits that must be transmitted by the sender in order to deliver the packet to the receiver. We define the bandwidth overhead for transmitting the i^{th} packet as $\beta_i = \dfrac{\mathrm{TBT}(i) - \mathrm{Bits\_C1}(i)}{\mathrm{Bits\_C1}(i)}$. Then, define the Percent Bandwidth Overhead (B) as

$$\mathrm{B} = \frac{\sum_{i=1}^{N} \beta_i}{N} \times 100.$$

A larger value of B means that more bandwidth was used to convey the same amount of information. It also means more energy consumption since energy is consumed in transmitting each bit. Therefore, schemes with smaller B are better with respect to this metric.

Miss rate (M) is the fraction of message packets which miss their deadlines. The simulator runs until there is 95% confidence that the estimated miss rate is within 25% of the actual miss rate. A scheme with smaller M is better with respect to this metric.

Normalized Net Profit (P) is defined as follows. Consider the relay of the i^{th} message packet by the sender during the course of the simulation. Let R_i be the reward associated with a message packet. Then, the net profit to the sender as a result of conveying the i^{th} message packet is

$$\eta_i = \begin{cases} \mathrm{R}_i - \mathrm{TBT}(i) & \text{if packet meets its deadline} \\ -\mathrm{TBT}(i) & \text{otherwise.} \end{cases}$$

Note that, $\eta_i^* = \mathrm{R}_i - \mathrm{Bits\_C1}(i)$ is the maximum net profit the sender can get as a result of conveying the i^{th} packet. Then, the normalized net profit is defined as

$$\mathrm{P} = \frac{\sum_{i=1}^{N} \eta_i}{\sum_{i=1}^{N} \eta_i^*}.$$

Therefore, P for a given FEC scheme, is the fraction of the maximum cumulative net profit achieved as a result of using that scheme. Higher values of P means a better scheme with respect to this metric. In theory, P can be negative indicating a loss.

4.2 AFEC vs. SFEC

We assume that the sender has exact knowledge of the parameter values of λ_c and λ_n used in the simulated air interface. From the values of λ_c and λ_n the sender derives the state transition probabilities to the *clear* and the *noisy* states using the method described in Section 4.1. For the results presented here, let the transition probabilities from the *noisy* to *clear* and from *clear* to *noisy* are 0.904 and 0.096 respectively. Results of sensitivity of the proposed scheme to the estimation errors in the values of λ_c and λ_n can be found in [2].

4.2.1 ATM stream

We assume that each 53-byte ATM cell is encapsulated in an *air-link-packet* before transmission over the air interface. The *air-link-packet* contains the 53-byte ATM cell, the check bytes, and an 11-byte header containing a byte specifying the error-correcting code, a length field specifying the number of information bytes in the packet, and a header checksum. We consider two Reed-Solomon codes, $c_1 \equiv (255, 247)$ and c_2

$\equiv (255, 239)$. The (255, 247) RS code can detect and correct 4 corrupted bytes while the (255, 239) RS code can detect and correct 8 corrupted bytes.

The deadlines of the ATM cells vary from 0.75 ms to 2.5 ms. For the assumed air interface bandwidth of 1 Mbps, the transmission time for an *air-link-packet* is approximately 0.7 ms. Since the sender must wait for an acknowledgment, the retransmission timeout is set at 0.75 ms. Therefore, depending on its deadline, each ATM cell can be transmitted at least once, but no more than three times prior to its deadline. For the results presented here, the reward for correctly delivering each ATM cell prior to its deadline is 1.25 times the cost of transmitting the ATM cell over the air interface using the $c1$ code, i.e., reward of each ATM cell is $1.25 \times 72 \times$ Byte_cost, where Byte_cost is the cost of transmitting one byte over the air interface. For simplicity, we assume Byte_cost is unity.

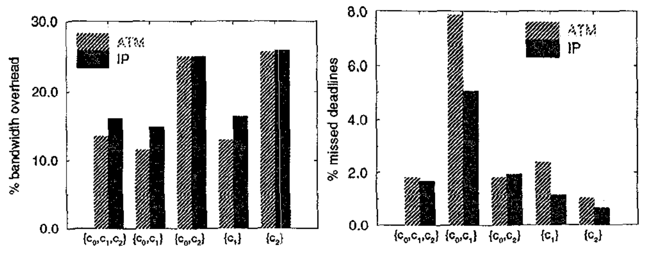
Using Equation (5) and the average bit-error rate of 0.007 for the *noisy* state and 0.5×10^{-5} for the *clear* state, the probability of correctable transmission of the *air-link-packet* in the *clear* state is found to be 1.0 for both $c1$ and $c2$, and in the *noisy* state to be 0.59 and 0.96 for codes $c1$ and $c2$ respectively.

4.2.2 IP stream

Most IP packets have length of fewer than 500 bytes. Therefore, we assume the length of the IP packet to be 512 Bytes. IP packets longer than 512 Bytes can be fragmented into smaller IP packets. As in the case of the ATM stream, each IP packet is encapsulated in an *air-link-packet* similar to the ATM case in Section 4.2.1. Further we assume that the two Reed-Solomon codes available to the sender and the receiver are $c_1 \equiv (1023, 943)$ and $c_2 \equiv (1023, 895)$. The (1023, 943) RS code can detect and correct 40 corrupted symbols[1] while the (1023, 895) RS code can detect and correct 64 corrupted symbols.

The deadlines of the IP packets are assumed to range from 6.5 ms to 20 ms. For the assumed air interface bandwidth of 1 Mbps, the transmission time for an *air-link-packet* is 6.25 ms. Since the sender must wait for an acknowledgment, the retransmission timeout is set at 6.5 ms. Therefore, depending on its deadline, each IP packet can be transmitted at least once, but no more than three times prior to its deadline. As in the case of the ATM stream, the reward for correctly delivering each IP packet prior to its deadline is assumed to be 1.25 times the cost of transmitting the packet over the air interface using c_1.

[1] A symbol is 10 bits long.



(a) Bandwidth overhead (b) Miss rate

Figure 1: Bandwidth overhead and miss rate.

The probabilities of correctable transmissions were computed, as shown in Section 3 using the average bit-error rate of 0.007 for the noisy state and 0.5×10^{-5} for the clear state, to be 1.0 for both codes $c1$ and $c2$ in the *clear* state and 0.36 and 0.99 in the *noisy* state for $c1$ and $c2$ respectively.

Bandwidth overhead and Miss rate. Figure 1 shows the percent bandwidth overhead (B) and the miss rate (M) for AFEC and SFEC schemes with the different code sets for the ATM and IP streams. AFEC with $\{c_0, c_1\}$ achieves the smallest bandwidth overhead, but has the highest miss rate. The miss rate in $\{c_0, c_1\}$ is high because it defers transmission of packets even when the deadlines are very close; this is due to the low probability of correctable transmission for code c_1 in the noisy state . AFEC with $\{c_0, c_2\}$ and the SFEC with $\{c_2\}$ incur a very large bandwidth overhead because of the large number of check bits introduced by code c_2. However, due to these additional check bits of c_2, the receiver can correct a larger number of symbol errors resulting in a lower miss rate. Although, AFEC with $\{c_0, c_1, c_2\}$ uses code c_2, the judicious use of deferment and the codes c_1 and c_2 results in a bandwidth overhead comparable to that of SFEC scheme with $\{c_1\}$. However, AFEC with $\{c_0, c_1, c_2\}$ has a lower miss rate because of the added protection of code c_2.

Normalized Net Profit. The normalized net profit is a measure of the balance between the bandwidth overhead of a scheme and its miss rate. The relative importance of these two factors is reflected in the reward associated with each type of packet. Figure 2 compares the normalized net profit (P) for the AFEC schemes with the different code sets. AFEC scheme with $\{c_0, c_1, c_2\}$ and with $\{c_0, c_1\}$has substantially higher normalized net profit than the SFEC scheme. AFEC with $\{c_0, c_2\}$ and SFEC with $\{c_2\}$ have

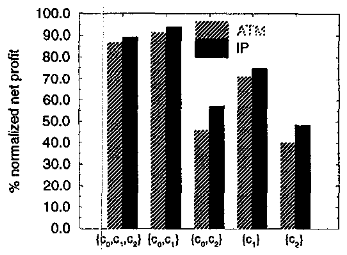


Figure 2: Normalized net profit.

lower normalized net profit because of the high bandwidth overhead. AFEC with $\{c_0, c_1\}$ achieves better profit than SFEC with $\{c_1\}$ because of the lower bandwidth achieved by AFEC. In contrast, AFEC with $\{c_0, c_1, c_2\}$ has much higher profit than SFEC with $\{c_1\}$ because of its lower miss rate. Between the two good adaptive schemes, the one with $\{c_0, c_1\}$ has slightly higher normalized net profit, but its miss rate of 8% for the ATM stream and 5% for the IP stream may not be acceptable to many applications.

5 Conclusion

In this paper, we proposed an adaptive forward error-correction scheme for real-time communication over a wireless interface. Its main objective is to reduce the bandwidth overhead without jeopardizing the timely delivery of message packets in a real-time message stream. This objective is achieved by carefully deferring transmission of a packet and by adaptively selecting the optimal error-correcting code from a set of available codes. Through simulations we compare the performance of the proposed scheme with that of a traditional single code scheme in which all the message packets are encoded using the same error-correcting code. The simulation results show that the proposed adaptive scheme is much better than the single code scheme in terms of the bandwidth utilized and in terms of a profit function which combines the bandwidth utilized and the deadline miss rate.

References

[1] B. R. Badrinath and P. Sudame, "To send or not to send: Implementing deferred transmissions in a mobile host," in *Proc. of ICDCS*, pp. 327–333, may 1996.

[2] M. Elaoud and P. Ramanathan, "Adaptive use of error-correcting codes for real-time communication in wireless networks," Technical Report ECE-97-7, Dept. of ECE, Univ. of Wisc, Madison, 1997.

[3] B. D. Fritchman, "A binary channel characterization using partitioned markov chains," *IEEE Trans. on Information Theory*, vol. IT-13, no. 2, pp. 221–227, April 1967.

[4] D. Gall, "MPEG: A video compression standard for multimedia applications," *Comm. of the ACM*, pp. 46–58, April 1991.

[5] H. M. D. Lima and O. C. M. B. Duarte, "Performance of repeated retransmission GB(N) schemes for point-to-multi-point noisy channels [satellite communications]," in *Proc. of Midwest Symp. on Circuits and Systems*, pp. 1–4, Aug 1995.

[6] R. Marasli, P. D. Amer, and P. T. Conard, "Retransmission-based partially reliable transport service: An analytic model," in *Proc. of INFOCOM*, pp. 621–629, March 1996.

[7] G. T. Nguyen and B. Nobel, "A trace-based approach for modeling wireless channel behavior," in *Proc. of Winter Simul. Conf.*, pp. 597–604, 1996.

[8] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *Computer Communication Review*, vol. 27, no. 2, pp. 24–36, April 1997.

[9] Y. Ryu, S. C. Kim, I. Song, S. I. Park, and H. M. Kim, "Adaptive change of code rate in ds-ssma communication systems," in *Proc. of Intl. Symp. on Signals, Systems, and Electronics*, pp. 199–202, October 1995.

[10] T. Sato, K. Tokuda, M. Kawabe, and T. Kato, "Simulation of burst error models and an adaptive error control scheme for high speed data transmission over analog cellular systems," *IEEE Trans. on Veh. Tech.*, no. 2, pp. 443–452, 1991.

[11] K. Yasui, T. Nakawaga, and H. Sandoh, "An automatic-repeat request policy for a data transmission system with three types of error probabilities," *Electronics and communications in Japan, part3 [Fundamental Electronic Science]*, vol. 79, no. 4, pp. 43–51, April 1996.

[12] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *IEEE Proc.*, vol. 83, no. 10, pp. 1374–1399, October 1995.