88 - 0017

1

AD-A229 690

SRI International

DYNAMIC SELECTION OF ERROR-CORRECTING CODES IN HYBRID ARQ PROTOCOLS

SRNTN 32

October 1985

By: Nachum Shacham

Telecommunications Sciences Center
Computer Science and Technology Division

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

Attention: Dr. Barry M. Leiner

SRI Project 5732

Effective Date: March 17, 1983
Expiration Date: May 15, 1984

Sponsored by
Defense Advanced Research Projects Agency (DoD)
DARPA Order No. 4570
Under Contract No. MDA903-83-C-0155 issued by
Department of Army, Defense Supply Service--Washington,
Washington, D.C. 20310

333 Ravenswood Ave. • Menlo Park, CA 94025
415 326-6200 • TWX: 910-373-2046 • Telex: 334 486

SRI International

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Dynamic Selection of Error-Correcting Codes in Hybrid ARQ Protocols	5. TYPE OF REPORT & PERIOD COVERED Survivable Radio Networks Temporary Note
	6. PERFORMING ORG. REPORT NUMBER 5732-SRNTN #32
7. AUTHOR(s) Nachum Shacham	8. CONTRACT OR GRANT NUMBER(s) MDA903-83-C-0155
9. PERFORMING ORGANIZATION NAME AND ADDRESS SRI International 333 Ravenswood Avenue Menlo Park, California 94025	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS P627083
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209	12. REPORT DATE June 1985
	13. NUMBER OF PAGES 42
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS *(of this report)* Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Unlimited distribution

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Packet radio networks, ARQ protocols, error correcting codes, code selection, performance evaluation, error rate, frequency hopping.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Packet transmission via automatic repeat request (ARQ) protocol over a channel with unknown, time-varying characteristics is considered. Extensions to more than two codes are discussed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ABSTRACT

Packet transmission via automatic repeat request ~~(ARQ)~~ protocol over a channel with unknown, time-varying characteristics is considered. The transmitter, with two error-correcting codes at its disposal, has to decide which to use at any given time. Two algorithms for adapting the error-correcting code to the channel conditions are presented; one of them assumes some knowledge of the distribution of errors in a packet, while the other makes no such assumption. Both algorithms are based on the observation that, when a packet's decoding is successful, the receiver knows the number of errors in that packet. Both algorithms build a measure for link quality and update it according to the decoding results. The first algorithm makes use of the number of errors in the packet to evaluate the probability that the channel is in a given state. The second algorithm updates its measure according to the highes-rate code that could have possibly corrected that packet. When this measure is above some predefined threshold, the first code is used; otherwise the second is employed. The throughput of both algorithms is ascertained and they are found to have excellent adaptivity, approaching that of a transmitter with perfect knowledge of channel conditions. Extensions to more than two codes are discussed.

Keywords: Adaptive communications networks, [illegible] Transp. (KR)

I. INTRODUCTION

Automatic repeat request (ARQ) is a technique for ensuring reliable data transfer, which is used in link-level protocols of computer communication networks [1]. Data are sent over the link by packets (frames) to which the transmitter adds parity bits to help the receiver decide whether or not the incoming data contain errors or not. If there are errors, the receiver sends the transmitter a negative acknowledgment that prompts retransmission. Hybrid ARQ (HARQ) is a modification of this scheme in which the data are encoded by an error-*correcting* code [2]. By correcting some of the errors, the use of this code results in fewer retransmissions. A packet's reception is unsuccessful only when its error pattern is beyond the correcting capability of the code, in which case we say that the decoding has failed.

An error-correcting code requires many more redundant bits in the packet than does the error-detecting code, so that there are fewer data bits in a packet of a given size. The number of redundant bits grows with the error correction capability. When the channel bit error rate (BER) is high, a packet under an error-detecting code will have to be retransmitted many times before it is received successfully. The redundancy of the error correction code is more than offset by the reduced number of retransmissions. However, when the BER is low few packets have to be retransmitted, even with an error-detecting code. In this case the full power of the error-correcting code is not utilized; its effect, then, is just a reduction in the rate at which data are transmitted over the channel. We illustrate this trade-off in Figure 1, which shows the performance of HARQ for a channel with random bit errors. The ordinate is the channel throughput, defined as the expected number of data bits successfully received per packet, normalized to the packet length, and the abscissa is the probability of bit error. The parameter is the code rate, defined as the number of data bits divided by the number of bits in the packet. What these curves mean is that, if the transmitter has several codes at its disposal and it knows the channel BER, it can then select the code that yields the highest throughput under the (H)ARQ protocol.

In many instances, however, channel error statistics are not only unknown in advance, but also vary with time. In such cases, which are common in ground radio links that are subject to fluctuations in signal strength and to interference,-- and in spread spectrum, multiple-access channels in which the traffic intensity varies with time -- it is not desirable to determine a priori the code rate at which the system will operate. Rather, we may want to adapt the error correction capability according to channel conditions. By doing this, we hope to obtain a performance curve that follows the "envelope" of the curves in Figure 1. Such an ARQ protocol that changes the code rate according to channel conditions is denoted here as an *adaptive hybrid ARQ* (AHARQ).

Several versions of AHARQ techniques can be found in the literature. The most common one is based on encoding the data with an error correcting code, but sending first only the data bits, protecting them by a high-rate error-detecting code [3,4,5]. If this transmission is unsuccessful, the error-correcting bits are transmitted. This technique avoids wasting channel capacity at a low BER yet has the power of an error-correcting code available when it is needed.

Another AHARQ that is aimed at channels with high BER [6] encodes the packet with an error-correcting code and sends it to the receiver. If the decoding is unsuccessful, the packet is retransmitted and the two versions of the packet are combined at the receiver (possibly with different weights) and effectively obtain a lower-rate code. The receiver stores all the copies of the received packet and attempts to decode them under all possible combinations. The receiver requests a retransmission only if all these decoding efforts fail.

The aforementioned AHARQ techniques achieve improved channel throughput by increasing the complexity of the decoding mechanism as compared with nonadaptive HARQ while using the same feedback, namely one bit ("ACK", "NAK") for each transmitted packet. In this paper we select the "dual" approach and show that it is possible to adapt to varying channel conditions by using the same decoding procedures as in "regular" ARQ, but slightly increasing the amount of feedback information. The increased information is based on the bit-error pattern in the packet that is known to the receiver when the decoding is successful. Of course, when the decoding is unsuccessful, no such knowledge exists. The

AHARQ algorithms we present in this paper utilize this information to construct and update a measure for channel quality. The better of two codes available to the transmitter is selected from this measure for use in the next transmission.

This paper is organized as follows: Section II describes the basic channel models and the assumptions made to facilitate performance evaluation of the algorithms. The first model, in which the channel alternates between states of known error distribution with an unknown parameter, is discussed in Section III, where we also present an AHARQ algorithm that tunes the code rate in this situation. An algorithm for adapting the code to variation in a channel that alternate between states of unknown error statistics can be found in Section IV. In both cases, the resulting channel throughput is ascertained by means of a probabilistic model. Section V contains some extensions of these algorithms to allow of more complicated situations to be handled.

II. MODELS AND GENERAL CONSIDERATIONS

Consider the following situation: a transmitter sends to a receiver packets of n bits each. The transmitter has K codes at its disposal, denoted as $c_1, c_2, \ldots, c_K$. A code c_i is characterized by a triplet (n, r_i, t_i), the elements of which are the length of the code-word, the code rate (ratio of the number of data bits in a packet to n), and the error-correcting capability, which is the maximum number of bit errors that can be corrected by c_i. We use the convention that $r_i > r_j$ and $t_i < t_j$ for $i < j$. Note that in this paper all the code words have the same length; the purpose of this is to facilitate presentation and analysis. This assumption can be relaxed very easily.

Assuming random error distribution in the packet, we note that, if the number of errors e in a given packet is smaller than t_1,

then c_1 is the code with the highest rate that can correct the packet. That is, c_1 is the code that conveys the largest number of data bits in a packet of a given size; hence it is the best code for this error pattern. In general, when

$$t_{i-1} < e \le t_i, \quad 1 < i \le K \quad ,$$

the code c_i conveys the maximum number of data bits among the K available codes. When $e > t_K$, none of the codes can decode the packet. Given a set of K codes as described above, the collection of possible error patterns in a packet can be partitioned into $K+1$ regions $(1,2,..,K+1)$ so that, for error pattern $i \le K$, code c_i is the best, whereas, for $i = K+1$, none of the codes is useful.

We assume that the distribution of errors in a packet is governed by a channel parameter, the value of which we denote as the channel *state*. Conditioned on the value of this parameter, the error pattern for each packet is drawn independently of other packets. When the channel state changes, the error distribution also changes. Thus, knowing this distribution and the parameter value provides enough information to select the code. Unfortunately, since full knowledge is not available to the transmitter and it thus has to base its operation on partial knowledge.

We consider here two types of partial knowledge the transmitter possesses: (a) the transmitter knows the general shape of the distribution of the number of errors in the packet but it does not know the value of one of its parameters, and (b) the transmitter does not even know the general shape of the distribution. The first type of knowledge can be encountered in channels with additive white Gaussian noise where the errors are random and thus the number of errors in the packet has a binomial distribution, a fact known to the transmitter. The channel state α_i is represented by the BER, p_i; while in that state, the number of errors e in the packet is given by

$$P(e|\alpha_i) = B(n,e,p_i) = \binom{n}{e} p_i^e (1-p_i)^{n-e} \quad .$$

The BER may be time varying, for example, as a result of on-off jamming. However, the rate of change is relatively slow compared with packet transmission time.

The second type of knowledge may be exemplified by a multiple-access spread-spectrum channel, where the principal source of errors is interference from other transmitted packets. It has been shown that, in a frequency-hopping system, the number of interfering packets n_I determines the bit error rate in a packet [7]. Under random channel access protocol the number of interfering packets is often modeled as a random variable, independent from packet to packet, with a Poisson distribution, $P(n_I|\alpha_i) = \frac{\alpha_i^{n_I}}{n_I!} e^{-\alpha_i}$, where the parameter α_i is called the traffic intensity and, in this case represents the channel state. The traffic intensity is usually time-varying, but its rate of change is slow relative to packet transmission time. Thus, for a given α_i, the BER is not fixed as in the previous example but is a random variable that is independent, identically distributed from packet to packet. However, here we assume that the transmitter does not even know the general shape of the error distribution.

We assume that the transmitter has a long sequence of data to send to the receiver. When the algorithm dictates the next packet to be transmitted at rate r, the transmitter uses for that packet the next nr data bits from that sequence. If the packet's decoding is successful, the transmitter uses the immediately following bits for the next packet. If the decoding fails, however, that packet is discarded by the receiver and the transmitter uses for the next packet the first nr' bits from the sequence it had before transmitting the packet. Here r' is the code rate to be used in the next transmission.

The objective of the AHARQ algorithms is to allow the transmitter to the best code given the limited knowledge it has about the channel state. The performance measure we apply to compare the codes is the average *throughput*, which is defined as the expected number of data bits conveyed in a packet normalized to the packet length.

Under the aforementioned assumptions of conditional i.i.d. error patterns and BERs, a reasonable approach is, for a given channel state, to use one code exclusively. Consider a specific error pattern distribution $P(i)$, $1 \le i \le K+1$, where $P(i)$ is the probability that the number of errors falls into the aforementioned i-th region. If code c_k is used exclusively the resulting throughput will be

$$S(k) = r_k \sum_{i=1}^{k} P(i) \quad , \qquad \#$$

since $\sum_{i=1}^{k} P(i)$ is the probability that c_i decodes the packet successfully. The code that achieves the highest throughput, $S_{\max} = \max_i \{S(i)\}$, is the best one to use for the specific channel state. This code may have to be changed when the channel state changes.

Another way of using the codes is to select a code at random according to a distribution u_j, $1<j<K$. The throughput for this distributions is

$$S = \sum_{j=1}^{K} u_j r_j \sum_{i=1}^{j} P(i) \quad . \qquad \#$$

We thus want to choose $\{u_j\}$, $\sum_{j=1}^{K} u_j = 1$, such that S is maximum. It can easily be shown that assigning $u_k = 1$ to the c_k that maximizes Eq. (0), and $u_j = 0$ otherwise, also maximizes S in Eq. (0).

Thus, the problem of adapting the code rate to the channel conditions can be coveniently stated as the problem of determining which of the K available codes achieves the highest throughput for each channel state. Note that a given code can be the best for more than one state and that, in fact, we may face a situation in which all the possible channel states are best served by a single code of the code set available to the transmitter. If the state of the channel is known to the transmitter, the problem is trivial because the best code can be determined from Eq. (0). However, the channel state is usually not known to the transmitter; it thus has to make a decision based on the sequence of codes it used in the past and the feedback it obtained from the receiver.

We turn our attention now to the receiver. An observation that is fundamental to the operation of the AHARQ algorithms presented here is the fact that, upon the correct decoding of a packet, the receiver can tell how many errors the received packet has contained. There is, of course, the possibility of decoding error, but this can be made very unlikely when a small number of error-detecting bits (outer code) is added to the data portion of the packet.

The receiver of course knows the set of available codes and their error- correcting capabilities. Thus, from the error pattern in the packet the receiver can also tell which of the codes could have decoded the packet correctly. It is obvious that, when the receiver fails to decode a packet, it gains none of the foregoing information. In fact, all the receiver then knows is that the decoding has failed.

The information gained by the receiver can be fed back to the transmitter by very few bits: in the first case in which the number of errors is sent back, the maximum length of the feedback message is less than or equal to $\log_2 t_K + 1$, which is much smaller than n. For example, the feedback for a ($code255$,0.5,19) code can be represented by 5 bits. In the second case, the receiver conveys to the transmitter only the index of the highest-rate code that could have corrected the packet, i.e., the feedback message is $\log_2(K+1)$ bits long. Note that the receiver does not have to send the raw information to the transmitter; it can carry out the algorithm to determine the next code that should be used and notify the transmitter only as to the index of that code via a $\log_2 K$ bit message. For the rest of this paper we will refer to the transmitter as the one who executes the algorithms; however, it should be understood that the receiver could also execute it, as explained above.

In the next two sections we present and analyze the performance of two AHARQ algorithms for which $K=2$. These are basic algorithms in the sense that AHARQ for $K>2$ can be built by $K-1$ operating basic algorithms in parallel, as discussed in Section V.

III. CODE SELECTION FOR CHANNEL STATES WITH KNOWN STATISTICS

In this section we present an AHARQ algorithm that assumes that the transmitter knows the general shape of the packet error distribution and uses the number of errors in the previous packets to adapt its code rate to channel conditions.

Suppose the channel alternates between two states, $\{\lambda_1,\lambda_2\}$, each of which has a completely known error distribution and known parameters. That is, the transmitter knows the conditional distribution of number of errors e in a packet, given the channel state, $P(e|\lambda_i)$, $i\epsilon\{1,2\}$. Further suppose that the transmitter has at its disposal two error correction codes $\{c_1,c_2\}$ with the parameters (n,r_k,t_k), $k=1,2$. Recall that the expected normalized number of bits conveyed by a packet encoded by c_k, when the channel is in state λ_i, is given by $S(k,i)=r_kP(e\leq t_k|\lambda_i)$. If $S(1,1)>S(2,1)$ and $S(1,2)>S(2,2)$, code c_1 is the best under both states and the transmitter should use that code exclusively. When both inequalities are reversed, only code c_2 should be used. However, if $S(1,1)>S(2,1)$ and $S(2,2)>S(1,2)$ then the transmitter should use code c_i whenever the channel is in state λ_i , $i=1,2$. This is the case when the AHARQ algorithm is needed most.

Since the times of transitions between the channel states are unknown to the transmitter, it does not know for sure what state the channel is in at any given time. However, it can represent its level of belief that the channel is in a certain state, say λ_1, by a number q $(0<q<1)$, which will be referred to as the *subjective probability*. The subjective probability will be updated according to the decoding results of every packet and will be used by the transmitter to select the code for the next packet. The larger q is, the more certain the transmitter is that the channel state is λ_1 and the more likely it is to use code c_1. Let us examine this notion for the simple case of myopic policy, in which the transmitter wishes to maximize throughput for the next transmission only.

A. Myopic Policy

When q is the probability that the channel is in state λ_1 at a given instant, and code c_k is used, the expected immediate throughput for the next packet transmission is given by

$$E(S|k,q) = r_k[qP_c(r_k|1)+(1-q)P_c(r_k|2)], \quad k\epsilon\{1,2\} \quad , \qquad \#$$

where

$$P_c(r_k|j) = \sum_{e=0}^{t_k} P(e|\lambda_j) \qquad \#$$

is the probability of correct decoding under code c_k and channel state λ_j. The optimal myopic policy is one that maximize the immediate throughput, that is, it uses the code c_i, which achieves

$$E(S|i,q) = \max_k\{E(S|k,q)\} \quad .$$

Using Eq. (0), we find that this policy gives rise to the following threshold criterion for selecting a code: if

$$q \geq \frac{r_2P_c(r_2|2)-r_1P_c(r_1|2)}{r_1[P_c(r_1|1)-P_c(r_1|2)]-r_2[P_c(r_2|1)-P_c(r_2|2)]} \quad , \qquad \#$$

then code c_1 should be selected; otherwise code c_2.

B. Updating q

Usually the transmitter is interested in transmitting a sequence of packets and in maximizing the average throughput for the whole sequence. To this end it should update the value of q after every transmission to obtain a better estimate of the channel state. Let $q(k)$ be the value of the subjective probability just before the transmission of the k-th packet. Using Bayes' rule, the transmitter computes the conditional probability that the channel is in state λ_1, given that there were e errors in the previous packet and that the current value is $q(k)$. This conditional probability becomes the new value. That is,

$$q(k+1) = \frac{q(k)P(e|1)}{q(k)P(e|1)+(1-q(k))P(e|2)} \quad . \qquad \#$$

The new value can be evaluated by Eq. (0) only if the decoding is successful. If code c_i was used and the decoding failed, the value of $q(k+1)$ is given by:

$$q(k+1) = \frac{q(k)P(e>t_i|1)}{q(k)P(e>t_i|1)+(1-q(k))P(e>t_i|2)} \quad . \qquad \#$$

As in the case of the myopic policy, the code for the k-th packet, $c(k)$, is determined by $q(k)$: if $q(k)$ is greater than or equal to some predetermined threshold, the transmitter selects c_1, otherwise it uses code c_2. The threshold may be different, though, from the one in Eq. (0).

Being a probability measure, $0 \le q(k) \le 1$. We of course want to have $q(k) \approx 1$ when the channel is in state λ_1 and $q \approx 0$ otherwise, and to have that probability move from one extreme to another soon after the channel switches states. First it should be noted that, if $q(k)=0,1$, then $q(k+1)=0,1$ respectively, regardless of the feedback. This means that, if the transmitter starts with an overly confident view of the channel, or ever arrives at such an extreme value, it never discards it. Thus, the foregoing extreme values are useless for our algorithm and the transmitter should never use them. It can easily be shown that $q(k+1)$ does not achieve the above extremes if $q(k)$ is not at these values. Therefore, if the transmitter does not use an extreme value for $q(1)$, then $q(k)$ never acquires an extreme value. In the following discussion, we thus assume that $0<q(k)<1$ for all k.

When the channel is in state λ_i, the probability that $q(k+1)=q_2$ given that $q(k)=q_1$ is given by

$$Pr\{q_2|q_1\} = \sum_e P(e|\lambda_i, q_1) \qquad \#$$

where the sum is over all values of e that, when used in Eqs. (0) (0), result in q_2.

The direction in which the sequence $\{q(k)\}$ moves as a function of k is represented by the expected drift which, given that channel is in state λ_i, is defined by

$$E[q(k+1)-q(k)|q(k),\lambda_i] = \sum_{e=0}^{t} \left[\frac{q(k)P(e|\lambda_1)}{q(k)P(e|\lambda_1) + (1-q(k))P(e|\lambda_2)} - q(k)\right] P(e|\lambda_i)$$
$$+ \left[\frac{q(k)P(e>t|\lambda_1)}{q(k)P(e>t|\lambda_1) + (1-q(k))P(e>t|\lambda_2)} - q(k)\right] P(e>t|\lambda_i) \quad , \#$$

where t relates to the code dictated by the value of $q(k)$ and the threshold. In the appendix we prove that the expected drift, given that the channel is in state λ_1, is positive while for λ_2 it is negative. That is, $\{q(k)\}$ moves in the right directions.

C. Channel States with Unknown Parameters

It very rarely happens that the channel alternates exactly between two states in which the transmitter knows the parameter of their error distribution. It is more likely that the channel alternates among several states, say $\alpha_1,..,\alpha_M$, for which only the general shape of the distribution is known but not the values of the parameter. It turns out that the aforementioned algorithm can be extended to this case too. The transmitter selects two states, say λ_1 and λ_2, which have the same error distribution as the α_i's. For example, if the α_i's have binomial distribution with unknown parameters, the λ_i's have the same distribution -- each with a specific parameter value. The parameter values for $\{\lambda_i\}$ are selected so that code c_1 achieves the higher throughput under λ_1 and c_2 is preferred under λ_2. The states $\{\lambda_i\}$ are called the decision states. Given the outcome of the decoding, i.e., a number of errors in the packet or a decoding failure, the value of $q(k+1)$ is still calculated according to Eqs. (0), (0). That is, the decision states are used in calculating $q(k+1)$. However, the decoding outcome is determined by the true channel state, say α_i, which means that the *probability* of moving from $q(k)=q_1$ to $q(k+1)=q_2$ is given by

$$Pr\{q_2|q_1\} = \sum_e P(e|\alpha_i,q_1) \quad , \qquad \#$$

where the summation is taken over all values of e such that Eqs. (0) and (0) result in q_2.

D. Quantization of q

The metric q is continuous and may acquire any value $0<q<1$. However, for practical reasons it is convenient to allow q to take only a finite number of values in the (0,1) interval. Quantizing q will also facilitate the performance evaluation of the algorithm over a time-varying channel. As it turns out, quantizing q down to very few levels does not degrade the performance of the algorithm if the quantization levels are chosen properly.

A quantization scheme for q is defined by the quantization interval boundaries, $0=d_0<d_1<d_2< \cdots <d_{M-1}<d_M=1$, and by M discrete values of q, namely, $q_1,q_2,\ldots,q_M$, where $d_{i-1}<q_i<d_i$. Now, if there are e errors in the packet and $q(k)=q_i$, the state changes to $q(k+1)=q_j$ such that

$$d_{j-1}<\frac{q_i P(e|\lambda_1)}{q_i P(e|\lambda_1)+(1-q_i)P(e|\lambda_2)}\leq d_j \qquad \#$$

if $e\leq t$, and if $e>t$ the new value is q_j for which

$$d_{j-1}<\frac{q_i P(e>t|\lambda_1)}{q_i P(e>t|\lambda_1)+(1-q_i)P(e>t|\lambda_2)}\leq d_j \quad . \qquad \#$$

Note that, if some quantization intervals are too large, it may happen that for a given q_i all values of e lead to $q_j=q_i$ in Eqs. (0) and (0). If this is allowed to happen, the algorithm will never leave the value q_i and will thus lose its adaptivity. Therefore, the quantization intervals should be small enough to ensure high probability for leaving each of them.

E. Performance Evaluation

We use the following model to evaluate the throughput achieved by the aforementioned algorithm: time is slotted into equally long slots, each of which fits an n bit packet. The transmitter has two codes c_1 and c_2, where $c_i = (n,r_i,t_i)$. The transmitter has a long data sequence to send which it does so by taking enough data bits in each slot to fit into an n bit packet, which is encoded by one of the above codes. The channel can be in one of N states, $\alpha_1,\ldots,\alpha_N$, where each state has a known error distribution, possibly with an unknown parameter. The channel changes states at the slot boundaries and the channel state transitions are done according to a Markov chain with transition probabilities $P_c(\alpha_j|\alpha_i)$, which are not necessarily known to the transmitter.

Following each transmission, the receiver sends back to the transmitter the number of errors if the packet decoding was successful or a NAK if it was not. The feedback message is sent instantly on an error-free channel.

The transmitter keeps the link quality measure q that can acquire one of of M possible values. Based on feedback information, the value of q is updated according to Eqs. (0) and (0); two states λ_1 and λ_2 with known statistics are used in these equations. These two states are used for decision purposes, as discussed above, and they are chosen to make the code c_1 achieve the higher throughput under λ_1 and c_2 do so under λ_2.

Whenever the value of q is greater than or equals a predetermined threshold q_{th}, the transmitter uses code c_2; otherwise it uses code c_1.

Under the above assumptions the dynamics of the algorithms can be described as a Markov chain $\{(q(k),\alpha(k)),\ k=1,2,\ldots\}$, where $q(k)$ and $\alpha(k)$ are the value of q and the channel state, respectively, just before the k-th slot. The chain has MN states and the transition probabilities are given by

$$P\{q(k+1),\alpha(k+1)\,|\,q(k),\alpha(k)\} = \Bigg[\sum_{e=0}^{t_{q(k)}} P(e\,|\,\alpha(k))\delta(e,q(k),q(k+1)) + P(e>t_{q(k)}\,|\,\alpha(k))\xi(e,q(k),q(k+1))\Bigg] \quad , \qquad \#$$

where $\delta(e,q(k),q(k+1))=1$ if using the values of e and $q(k)$ in Eq. (0) yields $q(k+1)$, and 0 otherwise. Similarly, $\xi(e,q(k),q(k+1))=1,0$ if using e and $q(k)$ in Eq. (0) yields $q(k+1)$ or not, respectively. Also, $t_{q(k)}=t_1$ if $q(k)\geq q_{th}$, and $t_{q(k)}=t_2$ otherwise.

When the quantization levels are chosen carefully as explained above, none of the states is an absorbing state and the chain is thus ergodic with a steady-state probability vector $\boldsymbol{\pi}$ given by

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P} \quad , \qquad \#$$

where $\mathbf{P}$ is the state transition matrix.

The throughput S is given by

$$S = \sum_{i=1}^{N}[\sum_{j=1}^{th-1} \pi_{j,i} r_2 P(e \le t_2 | \alpha_i) + \sum_{j=th}^{M} \pi_{j,i} r_1 P(e \le t_1 | \alpha_i)] \quad , \qquad \#$$

where $\pi_{j,i}$ is the steady-state probability that the channel is in state (q_j, α_i).

F. Discussion of Numerical Results

The throughput of the algorithm was ascertained numerically and the results are shown in Figures (2)-(4). In all these figures the transmitter has two codes, (255,1,0) and (255,0.5,19). The errors in a packet are random and the conditional distribution of the number of packets is binomial with parameter (BER) α_i which varies with time. In computing q, the transmitter uses two decision states with the above error distribution and with parameters $\lambda_1 = 0.001$ and $\lambda_2 = 0.01$. The channel alternates between two states α_1 and α_2, which are not necessarily the same as the decision states.

To appreciate how well the algorithm works, we compare the throughput it achieves to that achieved by three other simple schemes:

(1) Transmitter uses code c_1 only, regardless of feedback.

(2) Transmitter uses code c_2 only, regardless of feedback.

(3) Transmitter knows the exact channel state at all times and thus always matches the best code for that state. This scheme, called the ideal observer, clearly sets an upper bound on the throughput achievable by a transmitter with limited information.

Figure 2 depicts the throughput achieved by the algorithm as a function of the threshold level. The channel transitions are symmetric, which means that the channel's average stay in a state before making a transition is equal for both states. The abscissa in Figures 2(a)-(c) is the sojourn time - the average stay in a state expressed in packet transmission lengths. Four values of sojourn time are depicted in these figures: 100, 10,3. In Figure 2(a) the channel states are $\alpha_1 = 0.0008$ and $\alpha_2 = 0.03$. In this case we see that the algorithm's throughput is better than that achieved by either of the codes when it is used exclusively even for very small sojourn times. When the channel stays at the state longer, the throughput approaches the upper bound achieved by the ideal observer. Note that the threshold level has little effect on the throughput, so that its selection is not a critical design issue.

In Figure 2(b) the states are $\alpha_1=0.004$ and $\alpha_2=0.06$; in both of them code c_2 is preferred. In this case the threshold level and the sojourn time have only slight effect on the throughput. In Figure 2(c) the separation between the channel states is smaller and here, at short sojourn times, the threshold determines whether the algorithm is better than c_1 or worse. When the sojourn time is medium to large, the algorithm's throughput is larger than that of c_1 and approaches the ideal curve.

Figure 3 depicts the effect of the sojourn time on the throughput in a channel with symmetric transitions. Note that the algorithm approaches the ideal curve for relatively short sojourn times. Figure 4 depicts the performance of the algorithm for a channel with asymmetric transitions. In this case the channel spends an average of 20 packet transmission times in state α_1 and a different time in α_2. The abscissa of Figure 4 is the average time spent in state α_2. Note that the algorithm follows the ideal curve quite closely.

IV. AN ALGORITHM FOR A CHANNEL WITH UNKNOWN ERROR STATISTICS

The algorithm in the previous section made direct use of the number of errors in the packet, which was possible because the general characteristics of the error statistics were assumed to be known. This knowledge may not always be available, since very often there are many diverse factors that influence the error pattern and their overall effect cannot be quantified. In this section we consider a channel with unknown distribution of the error patterns.

As in the previous case, here too, when a packet is sent by code c_i and it is successfully decoded, the number of errors in that packet is known to the receiver. Also known to it is the set of codes available to the transmitter and their characteristics $\{(n,r_i,t_i)\}$. Thus, for each error pattern observed in a decoded packet, the receiver can tell which of the other available codes could have been used to decode that packet successfully. It is reasonable to assume that, if the code of rate r_i could decode the packet all other codes with rates $r_j<r_i$ could have done so too. Thus, the receiver checks only those codes with rates higher than the code that was actually used. The receiver sends to the transmitter on the feedback channel the index of the highest-rate code that could have decoded that packet successfully.

As before, when the packet's decoding is unsuccessful, the receiver sends only a NAK to the transmitter. In both cases, the decoding's success information is used by the transmitter to update its knowledge about the channel and to determine the code for the next transmission.

The basic algorithm is the one that decides between two codes, say, c_1 and c_2, where, say, $r_1>r_2$, $(t_1<t_2)$. In this case, the set of all the possible error patterns is partitioned into three groups: (1) those that can be corrected by both codes, (2) those that can be corrected only by c_2, and (3) error patterns that can be corrected by neither of these codes. In the following discussion we assume that, while time the channel is in a given state, say α_j, the error patterns are independent from packet to packet. The probability that an error pattern falls into group i, when given the channel is in state α_j is denoted by $P_j(i)$,

$i=1,2,3$.

We propose that the transmitter keeps a metric, denoted by x, that measures how well one code performs as compared with the other code. The transmitter bases its code selection on this metric. Specifically, when the transmitter uses code c_2 and the error pattern falls into group 1, this means that the transmitter could have achieved throughput greater by r_1-r_2, had it used code c_1 instead. Thus, in this case x is increased by the amount r_1-r_2. If the error patterns falls into group 2, the meaning is that, if the transmitter had used c_1 for this packet, it would have lost the whole packet whereas under c_2 it obtained the throughput of r_2. The transmitter, therefore, subtracts r_2 from the value of x in this case. If the error pattern falls into group 3, then x remained unchanged since none of the codes can convey any data.

Consider now the case in which the transmitter uses code c_1. If the error pattern falls into group 1, here too, x is increased by r_1-r_2. However, the receiver cannot distinguish between error patterns 2 and 3 since in both cases the decoding is unsuccessful under c_1. Thus, when the transmitter is operating with c_1 the x can be either decreased by r_2 or remain unchanged for both error pattern groups. In the following discussion we assume that the former possibility is used.

We denote by $x(k)$ the value of x just before the k-th packet transmission and, as before, $c(k)$ is the code used for the k-th packet. The transmitter starts with $x(1)=0$ and uses c_2 for the first packet. It updates x according to the feedback information as described above and uses the value of $x(k)$ to determine $c(k+1)$ according to the following rule:

$$\textit{If } x(k)\le 0 \textit{ then } c(k+1)=c_2, \textit{ else } c(k+1)=c_1 \quad .$$

Suppose now that the channel is in a given state α_j, with error region distribution $\{P_j(1),P_j(2),P_j(3)\}$. Recall that since the packet error patterns are i.i.d., one code should be used exclusively while the channel is in this state. If code c_k is used in state α_j, the throughput will be

$$S_j(k) = r_k\sum_{i=1}^{k}P_j(i) \quad i=1,2 \quad . \qquad \#$$

For the distribution, $\{P_j(i)\}$, code c_1 is better than c_2 if

$$r_1P(1) > r_2[P(1)+P(2)] \quad , \qquad \#$$

and code c_2 is better otherwise. Since $x(k)$ determines $c(k)$ according to the above rule, we want $\{x(k)\}$ to be positive when $\{P_j(i)\}$ is such that inequality Eq. (0) holds and be negative when it is reversed.

The expected drift of $\{x(k)\}$ for that distribution is given by

$$E[x(k+1)-x(k)|x(k)\leq 0,\alpha_j] = [r_1-r_2]P_j(1)-r_2P_j(2) \qquad \#$$

and

$$E[x(k+1)-x(k)|x(k)>0,\alpha_j] = [r_1-r_2]P_j(1)-r_2[P_j(2)+P_j(3)] \quad . \qquad \#$$

Note that for $x(k)\leq 0$ the drift is positive when the inequality Eq. (0) is true and negative when it is false. That is, when c_2 is used ($x(k)\leq 0$), the feedback provides the transmitter with enough information so that the sequence $\{x(k)\}$ increases or decreases according to the conditions in Eq. (0). Thus, under $\{P_j(i)\}$ for which Eq. (0) is satisfied, if $x(k)\leq 0$, the sequence $\{x(k)\}$ will move up and eventually cross the threshold so that the correct code, c_1 will be used. When $x(k)\leq 0$ and the inequality sign in Eq. (0) is reversed, $\{x(k)\}$ will tend to remain negative, thereby causing c_2 to be used, which is the better code for this case.

The situation is slightly different for positive values of $x(k)$ in which c_1 is used. For this code, the error pattern groups 2 and 3 are indistinguishable because c_1 cannot correct any of them. This causes the drift when $x(k)>0$ to be positive or negative under slightly different conditions from those given in Eq. (0). Although in most cases these different conditions do not degrade the algorithm's performance, there are some distributions $\{P_j(i)\}$ for which such degradation may occur. This happens when the channel is in state α_j for which the following double inequality holds:

$$r_2[P(1)+P(2)] < r_1P(1) < r_2 \quad . \qquad \#$$

For this distribution, Eq. (0) holds, implying that c_1 should be used. Indeed, the drift for $x(k)\leq 0$ is positive, which makes the algorithm less likely to use c_2. However, when $x(k)>0$, Eq. (0) yields negative drift, because of the right inequality in Eq. (0), which tends to push $\{x(k)\}$ down. The total effect is that

$\{x(k)\}$ oscillates around the zero level, thereby accepting negative values more often than if it had stayed at a level far away from the threshold. Note that when $x(k)\leq 0$, code c_2 is used with the attendant loss of throughput, since c_1 is better for this distribution.

The region of $P(1)$ and $P(2)$ for which Eq. (0) holds is depicted in Figure 5. For all other possible combinations $\{P_j(i)\}$, the drift has the proper sign for both negative and positive values of $x(k)$, thus making $\{x(k)\}$ move in the right direction according to Eq. (0).

When the channel state changes so that the inequality Eq. (0) is reversed, the code should also change to provide the highest possible throughput. That is, $\{x(k)\}$ should change sign as fast as possible. If $\{x(k)\}$ is not bounded from above and below, when the channel conditions persist for a long time $\{x(k)\}$ will grow in either the positive or negative direction. A large value of $\{x(k)\}$ results in a long time to reach the threshold and to switch codes after channel state changes, implying loss of throughput.

To expedite threshold crossing upon state change we set upper and lower bounds $B_u>0$ and $B_l<0$, respectively which we do not allow $\{x(k)\}$ to cross. Thus, when the error pattern is in group 1

$$x(k+1) = \min(B_u, x(k)+r_1-r_2) \quad , \qquad \#$$

and when the pattern is in group 2 and $x(k)\leq 0$, or in groups 2 or 3 and $x(k)>0$:

$$x(k+1) = \max(B_l, x(k)-r_2) \quad . \qquad \#$$

The boundaries, denote as B_u, B_l should be low enough to allow for quick code change when it is needed, as well as high enough to keep small the probability of level crossing as a result of statistical fluctuations.

When $\{P(i)\}$ is such that for $x(k)>0$ the drift is negative and for $x(k)\leq 0$ it is positive (Eqs. (0), (0)), $\{x(k)\}$ fluctuates around the zero level regardless of the level of B_u. Since at this region c_1 should be used, we increase the time $\{x(k)\}$ is positive by occasionally setting $x(k)$ to B_u upon zero crossing from below. We thus modify the algorithm as follows: whenever an error pattern of type 1 occurs such that the upward step (of size r_2-r_1) of $x(k)$ causes it to cross the zero level, $x(k)$ makes this normal step with probability $1-q_c$ and with probability q_c $x(k)$ is set to the value of B_u. Eq. (0) now applies only in the case of $x(k)>0$ or $x(k) \leq -(r_1-r_2)$. However when $-(r_1-r_2)<x(k)\leq 0$, the new value of x upon error pattern of type 1 is given by:

$$x(k+1) = \begin{cases} x(k)+r_1-r_2 & w.p.\ 1-q_c \\ B_u & w.p.\ q_c \end{cases} \quad . \qquad \#$$

A. Performance Evaluation

The model we use to evaluate the performance of this algorithm is similar to that of the previous section. Channel time is composed of fixed-size slots equal in length to the packet size. The feedback is sent to the transmitter instantly on an error- free channel. The channel alternates among N possible states, $(\alpha_1, \ldots, \alpha_N)$ where each state α_j, $(j=1,2,..,N)$, is characterized by a distribution $\{P_j(i)\}$ of the error pattern groups. The channel may switch states only at slot boundaries and doing according to a Markov chain with transition matrix $P(\alpha(k+1)=\alpha_j|\alpha(k)=\alpha_i)=P_{tr}(j|i)$.

Since r_1, r_2 are rationals, we can represent the two nonzero steps of x, namely, r_1-r_2 and r_2 by integers. As before, B_u and B_l are the lower and upper bounds, respectively which means that the channel metric x can take $B = B_u+B_l+1$ values. Thus, the process $\{(\alpha(k),x(k)),\ k=1,2,...\}$ is an ergodic Markov chain.* The chain's transition probabilities are given by

$$P(\alpha(k+1)=\alpha_j,x(k+1)=x_l|\alpha(k)=\alpha_i,x(k)=x_n) = P_{tr}(j|i)P_i(j|n) \quad , \qquad \#$$

where

$$P_i(j|n) = \begin{cases} P_i(1)q_c & \text{if } j=B_u \text{ and } -(r_1-r_2)<n\leq 0 \\ P_i(1)(1-q_c) & \text{if } j=n+r_1-r_2 \text{ and } -(r_1-r_2)<n\leq 0 \\ P_i(1) & \text{if } j=min(B_u,n+r_1-r_2) \text{ and } n\leq -(r_1-r_2) \text{ or } n>0 \\ P_i(2) & \text{if } j=max(B_l,n-r_2),\ n<0 \\ P_i(2)+P_i(3) & \text{if } j=n-r_2,\ n>0 \\ P_i(3) & \text{if } j=n,\ n\leq 0 \end{cases} \qquad \#$$

*For some combinations of r_1, r_2, $x(k)$ never acquires some of the values between B_l and B_u we consider only the values $x(k)$ actually takes.

If we denote by $\pi(j,n)$ its steady state probability that the chain is in state (α_j,x_n), the throughput of the algorithms can be written as:

$$S = \sum_{i=1}^{N} \left[\sum_{n=-B_l}^{0} \pi(i,n) r_2 [P(1)+P(2)] + \sum_{n=1}^{B_u} \pi(i,n) r_1 P(1) \right] \quad . \qquad \#$$

B. Discussion of Numerical Results

The performance of the algorithm is depicted in Figures 6-8. The two codes used in these curves are $c_1 = (255,1,0)$ and $c_2 = (255,0.5,19)$. Figure 6 shows the throughput obtained by the algorithm when the channel alternates between two states with equal expected sojourn time in both states (symmetric transitions). We see that, for very short sojourn times, the throughput is better than that achieved by using a single code only, that is, using no information. As the sojourn time increases, the throughput gets closer to the one achieved by an ideal observer.

Figure 7 shows the effect of q_c on the throughput. In Figures 7(a) and 7(b) the channel stays in one state only. However, the distribution $\{P(i)\}$ of this state is such that the drift has positive sign for $x \le 0$ and negative otherwise, a situation that causes $\{x(k)\}$ to oscillate around the zero level and thereby to lower throughput. For this distribution code c_1 is better and we can see that, as the probability of jumping to B_u upon zero crossing from below increases, so does the throughput. Thus, one should operate in such situation with a large value of q_c. One may wonder whether increasing q_c will have negative effect when $\{P(i)\}$ is not in the region depicted by Figure 5. Figure 7(c) shows that, for one such distribution, q_c has very little effect.

Figure 8 shows the improvement in throughput that can be achieved by increasing B_l and B_u. We see that the improvement is rather small. Figure 9 compares the performance of the algorithms described in the current and the previous sections. To compare them on an equal basis we assumed the number of errors in the packet to be binomially distributed and the two BERs to be 0.008 and 0.02. The channel alternates between these two states and make symmetric transitions. The same two codes that were mentioned above serve both algorithms. The curves in Figure 9 show that both algorithms increase their throughput as the sojourn time increases. However, the algorithm of the previous section, which uses knowledge of the error distribution, achieves higher throughput than does the algorithm of the current sectionm which does not use such knowledge.

V. EXTENSIONS

The AHARQ algorithms we have presented in Sections III and IV select one of two codes to use in a time-varying channel with some unknown characteristics. The code selection is based on a measure for channel quality that is updated by utilizing enhanced feedback sent from the receiver to the transmitter. The performance curves for these algorithms show that the throughput achieved is not very far from that achieved by an ideal observer who has all the channel information.

These algorithms can be extended to the case in which the transmitter holds $K>2$ codes. For a set of codes $c_1,..,c_K$ with $r_1>...>r_K$, the transmitter performs out $K-1$ two-code algorithms in parallel for the pairs of codes (c_1,c_2), (c_2,c_3),.., (c_{K-1},c_K) and selects the code with the highest throughput.

If the k-th packet is transmitted with code c_i and the decoding is successful the receiver knows the number of errors in this packet and also which of the K codes could have been used to quality measures, $q_{i,i+1}$ or $x_{i,i+1}$. This updating is more complicated when the decoding of the c_i packet fails. The receiver then knows that, for all j, $j<i$, the code c_j would have also failed. However, there is not much that can be said about the possible success of c_j, $j>i$. One option is not to update the measures for the algorithms that compare these codes. Another option is to assume that, for all $j>i$, code c_j could have decoded that packet correctly. Which of these alternatives is better will depend on the channel mode.

The selection of the best of $K>2$ codes is a little more complex than in the case $K=2$. We propose that the algorithms select the highest rate code, say, c_i for which the quality measure $q_{i,i+1}$ is above the threshold (or $x_{i,i+1}$ is above zero). The details of this selection and the performance evaluation of multiple-code algorithms will be presented in a forthcoming paper.

Other extensions, such as operation over a channel with a large propagation delay and a noisy feedback channel, may also be of interest. They will be the subject to future research.

APPENDIX: Evaluation Of The Expected Drift For q

In this appendix we prove that, for channel states λ_1, λ_2 the drift under λ_1 is positive, $E(q(k+1)-q(k)|q(k),1)>0$, and under λ_2 $E(q(k+1)-q(k)|q(k),2)<0$.

Proof: Denote by q, q' the values of $q(k+1)$, $q(k)$, respectively. By definition, when code c_l is used

$$q'-q=\begin{cases}\dfrac{P(e|\lambda_1)q}{P(e|\lambda_1)q+P(e|\lambda_2)(1-q)}-q & e\leq t_l\\ \dfrac{P(e>t_l|\lambda_1)q}{P(e>t_l|\lambda_1)q+P(e>t_l|\lambda_2)(1-q)}-q & e>t_l\end{cases}\tag{A1}$$

Following some simple algebraic manipulation, the drift can be written as

$$E[q'-q|q\lambda_1]=q(1-q)\left[\sum_{e=0}^{t_l}\frac{P(e|\lambda_1)-P(e|\lambda_2)}{P(e|\lambda_1)q+(1-q)P(e|\lambda_2)}P(e|\lambda_1)+\frac{P(e>t_l|\lambda_1)-P(e>t_l|\lambda_2)}{P(e>t_l|\lambda_1)q+(1-q)P(e>t_l|\lambda_2)}P(e>t_l|\lambda_1)\right]\tag{A2}$$

Consider first the denominator of the term in the summation in Eq. (A2). Let e_b be the set of values of e such that $P(e|\lambda_1)\geq P(e|\lambda_2)$ when $e\epsilon e_b$. Since $0<q<1$,

$$P(e|\lambda_2)\leq P(e|\lambda_1)q+(1-q)P(e|\lambda_2)\leq P(e|\lambda_1)\quad \text{for } e\epsilon e_b\tag{A3}$$

and

$$P(e|\lambda_1)\leq P(e|\lambda_1)q+(1-q)P(e|\lambda_2)\leq P(e|\lambda_2)\quad \textit{otherwise};\tag{A4}$$

Thus, for $e\epsilon e_b$ the numerator is positive; otherwise it is negative. Hence, for both cases:

$$\frac{P(e|\lambda_1)-P(e|\lambda_2)}{P(e|\lambda_1)q+(1-q)P(e|\lambda_2)}P(e|\lambda_1)\geq\frac{P(e|\lambda_1)-P(e|\lambda_2)}{P(e|\lambda_1)}\geq P(e|\lambda_1)-P(e|\lambda_2)\quad .\tag{A5}$$

Similarly, for $e>t_l$

$$\frac{P(e>t_l|\lambda_1)-P(e>t_l|\lambda_2)}{P(e>t_l|\lambda_1)q+(1-q)P(e>t_l|\lambda_2)}P(e>t_l|\lambda_1)>\frac{P(e>t_l|\lambda_1)-P(e>t_l|\lambda_2)}{P(e>t_l|\lambda_1)}P(e>t_l|\lambda_1)\quad .\tag{A6}$$

Thus, for all $0<q<1$

$$E[q'-q|q,\lambda_1]>q(1-q)[\sum_{e=0}^{t_l}[P(e|\lambda_1)-P(e|\lambda_2)]+P(e>t_l|\lambda_1)-P(e>t_l|\lambda_2)]=q(1-q)0\quad .\tag{A7}$$

Under state λ_2 the drift is given by

$$E[q'-q\,|\,q,\lambda_2] = q(1-q)\left[\sum_{e=0}^{t_i}\frac{P(e|\lambda_1)-P(e|\lambda_2)}{P(e|\lambda_1)q+(1-q)P(e|\lambda_2)}P(e|\lambda_2) + \frac{P(e>t_i|\lambda_1)-P(e>t_i|\lambda_2)}{P(e>t_i|\lambda_1)q+(1-q)P(e>t_i|\lambda_2)}P(e>t_i|\lambda_2)\right] \; .$$

$$< q(1-q)[\sum_{e=0}^{t_i}[P(e|\lambda_1)-P(e|\lambda_2)]+P(e>t_i|\lambda_1)-P(e>t_i|\lambda_2)] = q(1-q)0 \quad . \qquad \text{(A8)}$$

References

1. H. Burton and D. Sullivan, "Error and Error Control," *Proc. of the IEEE*, vol. 60, no. 11, pp. 1293-1301, November 1972.

2. E. Rocher and R. Pickholtz, "An Analysis of the Effectiveness of Hybrid Transmission Schemes," *IBM J. RES. DEVELOP.*, vol. 14, no. 4, pp. 426-433, July 1970.

3. J.J. Metzner, "Improvements in Block-Retransmission Schemes," *IEEE Transactions on Communications*, vol. COM-27, no. 2, pp. 525-532, February 1979.

4. J.J. Metzner and D. Chang, "Efficient Selective Repeat ARQ Strategies for Very Noisy and Fluctuating Channels," *IEEE Transactions on Communications*, vol. COM-33, no. 5, pp. 409-416, May 1985.

5. S. Lin and P.S. Yu, "A Hybrid ARQ Scheme with Parity Retransmission for Error Control of Satellite Channels," *IEEE Transactions on Communications*, vol. COM-30, no. 7, pp. 1701-1719, July 1982.

6. D. Chase, "Code Combining - A Maximum Decoding Approach for Combining an Arbitrary Number of Noisy Packets," *IEEE Transactions on Communications*, vol. COM-33, no. 5, pp. 385-393, May 1985.

7. M. Pursley, "Frequency-Hop Transmission For Satellite Packet Switching and Terrestrial Packet Radio Networks," T-144, Coordinated Science Laboratory - University of Illinois, Urbana-Champaign, June 1984.

SRNTN DISTRIBUTION LIST

Bolt Beranek and Newman, Inc. (2)
ATTN: Jil Westcott
10 Moulton Street
Cambridge, MA 02238

Defense Advanced Research (2)
Projects Agency/IPTO
ATTN: Richard L. DesJardins
1400 Wilson Boulevard
Arlington, VA 22209

MIT Lincoln Laboratories
ATTN: Richard Ralston
Box 73, Room C-117
Lexington, MA 02173

Naval Warfare Systems Command
ATTN: Barry Hughes
Code 611
Washington, D.C. 20363-5100

Polytechnic Institute
ATTN: Bob Boorstyn
333 Jay Street
Brooklyn, NY 11201

SRI International (2)
ATTN: Janet Tornow
Office EJ131
333 Ravenswood Avenue
Menlo Park, CA 94025

SRI International
ATTN: Mike Frankel
Office EJ347
Menlo Park, CA 94025

UCLA
Computer Science Dept.
ATTN: Leonard Kleinrock
3732 BH
Los Angeles, CA 90024

USC-ISI
ATTN: Jon Postel
4676 Admiralty Way
Marina Del Rey, CA 90292-6695

Naval Research Lab
ATTN: J.E. Wieselthier
Code 7520
Washington, D.C. 20375

Commander (2)
ACEC
ATTN: AMSEL-COM-RF-2, P.Sass
Fort Monmouth, NJ 07703

Hazeltine Corporation (2)
Building 1
ATTN: Jeff Markel
Cuba Hill Road
Green Lawn, NY 11740

National Security Agency
ATTN: S743, S. Spano
Fort George Meade, MD 20755

Naval Ocean Systems Center
ATTN: Dan Leonard
271 Catalina Boulevard
San Diego, CA 92152

Rockwell International (2)
Collins Defense Communications
ATTN: John Jubin
M/S 460-215
3200 E. Renner Road
Richardson, TX 75081

SRI International
ATTN: K. Thames
Office EJ125
333 Ravenswood Avenue
Menlo Park, CA 94025

Stanford University
Computer Systems Laboratory
ATTN: Fouad Tobagi
Stanford, CA 94305

University of Illinois
CSL
ATTN: Mike Pursley
1101 W. Springfield Road
Urbana, IL 61801

Rome Air Development Center
ATTN: Dan McAuliffe
DICEF-Section (DCLS)
Griffiss AFB, NY 13441

Linkabit Corporation
ATTN: Richard Binder
3033 Science Park Road
San Diego, CA 92121

Revised 7/29/85