General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

AUTOMATIC-REPEAT-REQUEST ERROR CONTROL SCHEMES

Shu Lin
Department of Electrical Engineering
University of Hawaii at Manoa
Honolulu, Hawaii 96822
Tel. (808) 948-7443

Daniel J. Costello, Jr.
Department of Electrical Engineering
Illinois Institute of Technology
Chicago, Illinois 60616
Tel. (312) 567-3404

Michael J. Miller
Department of Electronics
South Australian Institute of Technology
Ingle Farm, South Australia
Australia

ABSTRACT

Error detection incorporated with automatic-repeat-request (ARQ) is widely used for error control in data communication systems. This method of error control is simple and provides high system reliability. If a properly chosen code is used for error detection, virtually error-free data transmission can be attained.

This paper surveys various types of ARQ and hybrid ARQ schemes, and error detection using linear block codes.

This work was supported by the National Science Foundation under grant ESC-81-02894 and by the National Aeronautics and Space Administration under grant NAG 5-234.

1. Introduction

A major concern in data communications is how to control transmission errors caused by the channel noise so that error-free data can be delivered to the user. An approach to this problem is the application of coding, i.e., the use of error-detecting or error-correcting codes [1-5]. There are two basic categories of error control schemes for data communications: the automatic-repeat-request (ARQ) schemes and the forward-error-correction (FEC) schemes.

In an ARQ error control system, a high-rate error-detecting code, say an (n,k) linear block code, incorporated with a certain retransmission protocol is used. When a message of k information bits is ready for transmission, n-k parity-check bits are appended to it to form a codeword. These n-k parity-check bits are formed based on the code used by the system. The codeword is then transmitted to the receiving end. The transmitted codeword is contaminated by the channel noise, and the word received may contain transmission errors. When a word is received, the receiver (or decoder) computes its syndrome. If the syndrome is zero, the received word is a codeword in the code being used. In this case, the received word is assumed to be error-free and delivered (with parity-check bits removed) to the user (or data sink). If the syndrome of the received word is not zero, the presence of errors is detected. In this case, the receiver discards the erroneously received word, and requests a retransmission of the same codeword via a feedback channel. Retransmission continues until the codeword is successfully received. With this error control system, erroneous data is delivered to the user only if the receiver

fails to detect the presence of errors. Using a proper error-detecting code, the probability of an undetected error can be made very small [3,6-15]. ARQ schemes are widely used in data communication systems for error control because they are simple and provide high system reliability. However, they have one drawback: the throughput is not constant and it falls rapidly with increasing channel error rate.

In an FEC error control system, an error-correcting code (block or convolutional) is used for combating transmission errors. Again parity-check bits are added to each transmitted message to form a codeword (or a code sequence) based on the code used by the system. When the receiver detects the presence of errors in a received word, it attempts to locate and correct the errors. After the error correction has been performed, the decoded word is then delivered to the user. A decoding error is committed if the receiver either fails to detect the presence of errors or fails to determine the exact locations of the errors. In either case, an erroneous word is delivered to the user. Since no retransmission is required in an FEC error control system, no feedback channel is needed. The throughput of the system is constant and is equal to the rate of the code used by the system. However, FEC error control systems have some drawbacks. When a received word is detected in error, it must be decoded, and the decoded word must be delivered to the user regardless of whether it is correct or incorrect. Since the probability of a decoding error is much greater than the probability of an undetected error, it is harder to achieve high system reliability with FEC schemes. In order to attain high system reliability, a long powerful error-correcting code

must be used and a large collection of error patterns needs to be corrected. This makes decoding hard to implement and expensive. For these reasons, ARQ schemes are often preferred over FEC schemes for error control in data communication systems, such as packet-switching data networks and computer communication networks. However, in communication (or data storage) systems where feedback channels are not available or retransmission is not suitable for some reason, FEC is the only choice.

This paper surveys a number of ARQ schemes. They represent alternative solutions to the design of retransmission protocols, particularly the mode in which the transmitter stores, orders, and retransmits the codewords which have been received in error. These different schemes have arisen primarily in an attempt to combat the problem that, when the channel error rate increases, the throughput of an ARQ error control system may deteriorate very rapidly. This is because of the time wasted in retransmitting the codewords detected in error. This problem becomes particularly severe if there is significant round-trip delay between the transmission of a codeword and the receipt of its error status information back at the transmitter. Long delay is inevitable when satellite or long terrestrial channels are being used.

Another approach to error control is through the use of hybrid ARQ schemes which incorporate both forward-error-correction and retransmission. Hybrid ARQ schemes offer the potential for better performance if appropriate ARQ and FEC schemes are properly combined. Either block or convolutional codes may be used for forward error correction. This paper also discusses different classes of hybrid ARQ

schemes and their performance.

2. Basic ARQ Schemes

Based on the retransmission strategies, there are three basic types of ARQ schemes: stop-and-wait ARQ, go-back-N ARQ, and selective-repeat ARQ [3,16-21]. The stop-and-wait scheme represents the simplest ARQ procedure and was implemented in early error control systems. For example, the IBM Binary Synchronous Communication (BISYNC) procedure was of the stop-and-wait type [22]. In a stop-and-wait ARQ error control system, the transmitter sends a codeword to the receiver and waits for an acknowledgement as shown in Figure 1. A positive acknowledgement (ACK) from the receiver indicates that the transmitted codeword has been successfully received, and the transmitter sends the next codeword in the input queue. A negative acknowledgement (NAK) from the receiver indicates that the transmitted codeword has been detected in error; and the transmitter resends the codeword and again waits for an acknowledgement. Retransmissions continue until the transmitter receives an ACK.

This scheme is simple but is inherently inefficient because of the idle time spent waiting for an acknowledgement of each transmitted codeword. One possible remedy is to make the block (or code) length n extremely long. However the use of a very long block length does not really provide a solution since the probability that a block contains errors increases with the block length. Hence, using a long block length reduces the idle time but increases the frequency of retransmissions for each codeword. Moreover, a long block length may

be impractical in many applications because of restrictions imposed by the data format.

By the 1970's, ARQ systems were in extensive use in packet-switched and other data networks. Higher data rates and utilization of satellite channels with long round-trip delays established the need for continuous transmission strategies to replace the stop-and-wait procedures. International standards organizations such as CCITT (the International Telegraph and Telephone Consultative Committee) began making efforts for protocol standardization. This resulted in the HDLC (high-level data link control) and the CCITT X.25 standards. These envisaged the use of a go-back-N ARQ system on full duplex links. This remains the standard for packet-switching networks.

The basic go-back-N ARQ scheme is illustrated in Figure 2. The transmitter continuously transmits codewords in order and then stores them pending receipt of an ACK/NAK for each. The acknowledgement for a codeword arrives after a round-trip delay. The round-trip delay is defined as the time interval between the transmission of a codeword and the receipt of an acknowledgement for that codeword. During this interval N-1 other codewords are also transmitted. Whenever the transmitter receives a NAK indicating that a particular codeword, say codeword i, was received in error, it stops transmitting new codewords. Then it goes back to codeword i and proceeds to retransmit that codeword and the N-1 succeeding codewords which were transmitted during one round-trip delay. At the receiving end, the receiver discards the erroneously received word i and all N-1 subsequently received words no matter whether they are error-free or not. Retransmission continues until codeword i is positively acknowledged.

In each retransmission for codeword i, the transmitter resends the same sequence of codewords. As soon as codeword i is positively acknowledged, the transmitter proceeds to transmit new codewords.

The main drawback of go-back-N ARQ is that, whenever a received word is detected in error, the receiver also rejects the next N-1 received words even though many of them may be error-free. As a result, they must be retransmitted. This represents a waste of transmissions which can result in severe deterioration of throughput performance if large round-trip delay is involved. For example, consider a satellite channel with a round-trip delay of approximately 700 milliseconds. If the codeword length n is 1000 bits long and the bit-rate is 1 Mb/sec, then in one round-trip delay N = 700 codewords are transmitted. Therefore, when one received word is detected in error, 700 received words are rejected. If errors occur often enough, the system throughput may fall off very rapidly.

The go-back-N ARQ scheme becomes quite ineffective for communication systems with high data rates and large round-trip delays. This ineffectiveness is caused by the retransmission of many error-free codewords following a codeword detected in error. This can be overcome by using the selective-repeat ARQ protocol. In a selective-repeat ARQ error control system, codewords are also transmitted continuously. However, the transmitter only resends those codewords that are negatively acknowledged (NAK'ed). As illustrated in Figure 3, after resending a NAK'ed codeword, the transmitter continues transmitting new codewords in the transmitter buffer. With this scheme, a buffer must be provided at the receiver to store the error-free codewords following a received word detected in error because ordinarily

codewords must be delivered to the end user in correct order, e.g., in point-to-point communications. When the first NAK'ed codeword is successfully received, the receiver then releases any error-free codewords in consecutive order from the receiver buffer until the next erroneously received word is encountered. Sufficient receiver buffer storage must be provided in a selective-repeat ARQ system; otherwise, buffer overflow may occur and codewords may be lost.

3. Reliability and Throughput Efficiencies of the Basic ARQ Schemes

The performance of an ARQ error control system is normally measured by its reliability and throughput efficiency. In an ARQ system, the receiver commits a decoding error whenever it accepts a received word with undetected errors. Such an event is called an error event. Let P(E) denote the probability of an error event. Clearly, for an ARQ system to be reliable, P(E) should be made very small. The reliability of an ARQ system is measured by its error probability P(E). The throughput efficiency (simply throughput) of an ARQ system is defined as the ratio of the average number of information bits successfully accepted by the receiver per unit time to the total number of bits that could be transmitted per unit time. All three basic ARQ schemes achieve the same system reliability, but they provide different throughput efficiencies.

Suppose that an (n,k) linear block code C is used for error detection in an ARQ system. Let us define the following probabilities:

P_c = probability that a received word contains no error,

P_d = probability that a received word contain a detectable error

pattern, and

P_e = probability that a received word contains an undetectable error pattern.

Obviously, $P_c + P_d + P_e = 1$. The probability P_c depends on the channel error statistics, and the probabilities P_d and P_e depend on both the channel statistics and the choice of the (n,k) error-detecting code C. P_e is normally called the probability of undetected error of the code. A received word is accepted by the receiver only if it either contains no errors or contains an undetectable error pattern. Hence, the probability P(E) that the receiver in an ARQ system commits a decoding error is given by

$$P(E) = \frac{P_e}{P_c + P_e} \tag{1}$$

If the error-detecting code C is properly chosen, P_e can be made very small relative to P_c, and hence P(E) can be made very small.

For a random error channel with bit-error rate ε,

$$P_c = (1-\varepsilon)^n . \tag{2}$$

It has been proved that there exists linear block codes with the probability of undetected error P_e satisfying the following upper bound [6,7,10]:

$$P_e \leq [1 - (1-\varepsilon)^k]2^{-(n-k)} . \tag{3}$$

Codes satisfying the above bound have been found and will be discussed in a later section. If a code satisfying the bound given by (3) is used for error detection and if the number of parity bits, n-k, is sufficiently large, P_e can be made very small relative to P_c and hence $P(E) \ll 1$. For example, let C be the (2047,2014) triple-error-

correcting primitive BCH code. This code satisfies the bound given by (3) [14]. Suppose that this code is used for error detection in an ARQ system. Let $\varepsilon = 10^{-3}$ (a very high bit-error rate). Then $P_c \approx 1.25\times10^{-1}$, and $P_e \leq 10^{-10}$. From (1), we have

$$P(E) \leq 8\times10^{-10} .$$

From this example we see that high system reliability can be achieved by an ARQ error control scheme using very little parity overhead.

Now we examine the throughput performance of the three basic ARQ schemes. For simplicity, we assume that the forward channel is a random-error channel with bit-error rate ε and that the feedback channel is noiseless. First we consider the stop-and-wait ARQ scheme. Let λ be the idle time of the transmitter between two successive transmissions. Let δ be the bit rate of the transmitter. Even though the transmitter does not transmit during the idle period, the effect of the idle period on the throughput must be taken into consideration. In one round-trip delay time, the transmitter could transmit $n+\lambda\delta$ bits if it did not stay idle. For a codeword to be received correctly, the average number of bits that the transmitter could have transmitted is

$$T_{SW} = \sum_{i=1}^{\infty} i(n+\lambda\delta)P_c(1-P_c)^{i-1} = (n+\lambda\delta)P_c \sum_{i=1}^{\infty} i(1-P_c)^{i-1} = \frac{n+\lambda\delta}{P_c} .$$

Therefore, the throughput of a stop-and-wait ARQ system is

$$\eta_{SW} = \frac{k}{T_{SW}} = \frac{P_c\cdot(\frac{k}{n})}{1+\lambda\delta/n} , \qquad (4)$$

where k/n is the rate of the code used by the system. For data communication systems with low data rates and small round-trip delays,

$\lambda\delta/n$ can be made much smaller than one by using a reasonably long code. In this case, the stop-and-wait ARQ scheme provides satisfactory throughput performance. However, for a high speed data communication system with large round-trip delay, $\lambda\delta/n$ becomes very large because it is impractical to use a very large n. In this case, the throughput performance of the stop-and-wait ARQ scheme becomes unacceptable. For example, consider a satellite communication system with a data rate of 1 M bits per second. Assume the round-trip delay for a satellite channel to be 700 milliseconds. Then $\lambda\delta = 0.7\times10^6$. To make $\lambda\delta/n$ small compared to one, n should be chosen in the range of one million bits! It is impractical to use such a long code. Suppose that we choose $n = 10{,}000$ bits. Then $\lambda\delta/n = 70$. In this case, the throughput of system becomes negligible.

In a go-back-N ARQ system, a retransmission of an NAK'ed codeword always involves resending N codewords. Consequently, for a codeword to be successfully received, the average number of transmissions is

$$T_{GBN} = \sum_{i=1}^{\infty} [(i-1)N+1]P_c(1-P_c)^{i-1} = \frac{P_c + (1-P_c)N}{P_c},$$

and the throughput of a go-back-N ARQ system is

$$\eta_{GBN} = \frac{1}{T_{GBN}} \cdot \left(\frac{k}{n}\right) = \frac{P_c \cdot \left(\frac{k}{n}\right)}{P_c + (1-P_c)N}. \qquad (5)$$

We see that the throughput depends on both the channel error rate and the round-trip delay N. The term $(1-P_c)N$ represents the effect of the channel error rate and the round-trip delay. For communication systems where the data rate is not too high and the round-trip delay is small, N can be made small by choosing a reasonably long code. In

this case, the effect of the round-trip delay is insignificant and the go-back-N ARQ scheme provides high throughput performance. However, for a communication system with a high data rate and long round-trip delay, N may become very large. As a result, $(1-P_c)N$ becomes significantly large, especially when the channel error rate ε is high. This would make the throughput performance of go-back-N ARQ inadequate. For example, consider a satellite communication system with a data rate of 1 M bits per second and a round-trip delay of 700 milliseconds. If we choose n = 10,000 bits, then N = 70. For $\varepsilon = 10^{-4}$, we have $P_c = 0.886$ and $(1-P_c)N \approx 8$. This gives a throughput of less than 10% of the code rate k/n. Figure 4 shows how η_{GBN} varies with bit-error rate ε for various values of code block length n. Throughput values are computed for two values of round-trip delay, namely 30 milliseconds, such as might be the case for a long terrestrial line, and 700 milliseconds, which is typical for a satellite channel. In both cases, the bit-rate is 64 kb/s. Each code block is assumed to contain 32 bits of overhead, including parity and control bits. We see that, for such a low data rate, the go-back-N ARQ system does provide satisfactory throughput performance when the round-trip delay is not too large, such as in a terrestrial system. For a satellite system, however, the throughput becomes unacceptable for a bit-error rate $\varepsilon > 10^{-5}$.

Now consider a selective-repeat ARQ system for which the receiver has an infinite buffer to store the error-free codewords when a received word is detected in error. We will call such a system an ideal selective-repeat ARQ system. For a codeword to be successfully accepted by the receiver, the average number of transmissions needed

is

$$T_{SR} = \sum_{i=1}^{\infty} i \cdot P_c(1 - P_c)^{i-1} = \frac{1}{P_c}.$$

Hence the throughput of an ideal selective-repeat ARQ system is

$$\eta_{SR} = \frac{1}{T_{SR}} \left(\frac{k}{n}\right) = P_c\left(\frac{k}{n}\right). \tag{6}$$

We see that the throughput does not depend on the round-trip delay. As a result, selective-repeat ARQ offers significant benefits for satellite and long terrestrial channels. Figure 5 gives a comparison of ideal selective-repeat ARQ and go-back-N ARQ for a satellite channel with a data rate of 1.5 Mb/s and a round-trip delay of 700 milliseconds. We see that selective-repeat ARQ significantly outperforms go-back-N ARQ, and provides acceptable throughput even at a bit-error rate of $\epsilon = 10^{-4}$.

The high throughput performance of ideal selective-repeat ARQ is achieved at the expense of extensive buffering (theoretically infinite buffering) and more complex logic at both transmitter and receiver. If a finite buffer is used at the receiver (this is the case in practical systems), buffer overflow may occur which reduces the throughput performance of the system. However, if a sufficiently long buffer is used and if buffer overflow is properly handled, even with a reduction in throughput selective-repeat ARQ still significantly outperforms the other two ARQ schemes. Selective-repeat ARQ schemes with finite receiver buffers will be discussed in the next section.

4. Variations of the Basic ARQ Schemes

Go-Back-N Variations

The throughput performance of the basic go-back-N ARQ scheme deteriorates rapidly with increasing bit-error rate, especially if there is a significant delay in the channel. To improve the throughput performance, a number of variations on the basic go-back-N ARQ scheme have been proposed. The first such variation is due to Sastry [23]. With Sastry's scheme, when a NAK is received, the transmitter backs up N codewords to the NAK'ed codeword and resends that codeword repeatedly until an ACK is received. At the receiving end, when the first NAK'ed codeword is recovered, the receiver has to wait N-1 codewords for the next codeword in the original sequence. Sastry's scheme provides higher throughput efficiency than the basic go-back-N ARQ scheme only for high bit-error rates such that $1 - P_c > 0.5$. For bit-error rates with $1 - P_c \leq 0.5$, it is inferior to basic go-back-N ARQ. Following Sastry's retransmission strategy, Morris [24] proposed to use a buffer of size N to store those error-free codewords following an erroneously received word. As soon as the first NAK'ed codeword is recovered, the receiver delivers to the user that codeword and other error-free codewords (if any) that are stored in the buffer in consecutive order until the next erroneously received word is encountered. At the transmitting end, when the transmitter receives an ACK for the first NAK'ed codeword, it either begins transmitting new codewords or initiates a retransmission of the next NAK'ed codeword. Morris' scheme only yields marginal improvement over the basic go-back-N ARQ scheme. For large N (large round-trip delay), the improvement becomes negligible.

Another variation of basic go-back-N ARQ is due to Lin and Yu [25]. With Lin-Yu's scheme, which is called SETRAN ARQ, the receiver

also stores up to N error-free codewords following a detected error. However the scheme uses a retransmission strategy different from Sastry's. When the transmitter receives the first NAK, it backs up to the first NAK'ed codeword and resends that codeword and any subsequently NAK'ed codewords. The time slots that correspond to the ACK'ed codewords are used to repeat the first NAK'ed codeword. This retransmission strategy reduces the effect of the round-trip delay, and hence provides higher throughput than basic go-back-N ARQ. Figure 6 shows the throughput of basic go-back-N ARQ and the three variations described above. We see that Lin-Yu's SETRAN ARQ outperforms the other schemes.

Selective-Repeat Variations

To achieve the throughput performance of ideal selective-repeat ARQ, infinite buffering is required at the receiver. If a finite buffer is used at the receiver, buffer overflow may occur, thereby reducing the throughput of the system. However, if sufficient buffer storage is provided, and if buffer overflow is properly handled, selective-repeat ARQ still significantly outperforms the other two basic ARQ schemes and their variations, especially for systems where the data rate is high and the round-trip delay is large.

There are two methods for handling buffer overflow in a selective-repeat ARQ system with a finite receiver buffer: one is to devise retransmission strategies so that buffer overflow can be prevented, and the other is to devise a mechanism for the transmitter to detect the occurrence of a receiver buffer overflow event so that the lost codewords can be properly retransmitted. The first method is

usually simpler but the second method provides better throughput performance.

Finite receiver buffer selective-repeat ARQ schemes which do not allow buffer overflow are reported in Metzner [26], Miller and Lin [27], and Weldon [28]. These schemes employ mixed-mode retransmission strategies. One such scheme is the selective-repeat plus go-back-N (SR+GBN) ARQ scheme [27]. When the transmitter fist receives a NAK for a given codeword (say codeword i), it retransmits that codeword and then continues transmitting other new codewords as in the basic selective-repeat (SR) mode (see Figure 7). If another NAK is received for codeword i, indicating its second transmission attempt was unsuccessful, the transmitter switches to the go-back-N (GBN) retransmission mode. That is, it sends no more new codewords but backs up to codeword i and resends that codeword and the N-1 succeeding codewords that were transmitted after the previous transmission attempt of codeword i (see Figure 7). The transmitter stays in the GBN mode until codeword i is positively acknowledged. At the receiver, when the second transmission attempt of codeword i is detected in error, the subsequent N-1 received words are discarded regardless of whether they were received error-free or not. This scheme achieves superior throughput performance compared with go-back-N ARQ and its variations. This is because of the benefits gained from the use of the SR mode for the first retransmission attempt. The use of the secondary mode (GBN) guarantees that buffer overflows cannot occur at the receiver as long as a buffer is provided for storing N codewords. The scheme can have an even higher throughput if more buffer storage is provided at the receiver and the transmitter is designed to permit more than one

retransmission attempt for a given NAK'ed codeword in the SR mode before switching to the GBN mode. If ν retransmissions in the SR mode are allowed before the transmitter switches to the GBN mode, the receiver buffer must be able to store $\nu(N+1) + 1$ codewords to prevent buffer overflow. The throughput for ν retransmissions in the SR mode is given by

$$\eta_{SR+GBN} = \frac{P_c(\frac{k}{n})}{1+(N-1)(1-P_c)^{\nu+1}} . \qquad (7)$$

Figure 8 shows how the throughput of the SR+GBN ARQ scheme varies with bit-error rate for several values of ν. As ν increases, the throughput performance of the SR+GBN approaches the ideal case.

Another mixed mode scheme is called the selective-repeat plus stutter (SR+ST) ARQ scheme [27]. This is the same as the SR+GBN scheme except that, instead of using the GBN mode after ν retransmission attempts of a given NAK'ed codeword, the transmitter switches to the ST mode in which it repeatedly retransmits that codeword until it receives an ACK. The SR+ST ARQ scheme is simpler but less efficient than the SR+GBN ARQ scheme.

A variation of the above mixed-mode ARQ scheme was suggested by Weldon [28]. When the transmitter is in the SR mode following the receipt of a NAK, it retransmits the NAK'ed codeword q_1 times (stuttering). Then it proceeds to transmit other codewords waiting in the transmitter buffer in sequence from where it left off. The number q_1 can be chosen to provide maximum throughput for a given error rate and delay. Typically, $q_1 = 3$ provides good results. If all q_1 retransmissions of a codeword are received with detectable errors, then the transmitter reverts to the GBN mode. For $q_1 = 1$, Weldon's

scheme becomes SR+GBN ARQ with $\nu = 1$. Actually Weldon's scheme is a clever combination of the SR+GBN and SR+ST ARQ schemes. Weldon's scheme can be generalized to have multi-level repetitions of a NAK'ed codeword before the transmitter switches to the GBN mode. As the number of levels increases, the throughput performance approaches the ideal case, and of course the size of the receiver buffer also increases. Weldon's scheme provides higher throughput than the SR+GBN ARQ scheme for very high bit-error rates. However, it is also more complex.

Selective-repeat ARQ schemes with finite receiver buffers that allow buffer overflow have recently been reported by Yu and Lin [29] and Wang and Lin [30]. In Yu-Lin's scheme, a receiver buffer of size N is used, where N is the number of codewords transmitted in one round-trip delay period. Each transmitted codeword has a sequence number ranging from 1 to 3N. The sequence numbers are reused cyclically. When a codeword is ready for transmission, it is numbered and stored in the input queue. After its transmission, it is saved in the retransmission buffer until it is positively acknowledged. After one round-trip delay time, it becomes a time-out word and should be either ACK'ed or NAK'ed. If it is not acknowledged (unACK'ed), its transmission is then regarded as unsuccessful, and it will be retransmitted. Whether the transmitter transmits a new codeword or retransmits a codeword that is NAK'ed or rejected at the receiver due to buffer overflow depends on the acknowledgement status and an index number of the current time-out word in the retransmission buffer of the transmitter. Let x_0 be the sequence number of the earliest NAK'ed or unACK'ed codeword in the retransmission buffer. The forward index f_T

of a codeword with sequence number x in the retransmission buffer or in the input queue is defined as

$$f_T \equiv x - x_0 \pmod{3N}.$$

When the transmitter is sending a codeword, it also computes the forward index f_T of the codeword in the retransmission buffer that is to become a time-out word, called the current time-out word. Normally, Yu-Lin's scheme operates in the same manner as the basic selective-repeat ARQ as illustrated in Figure 9. When receiver buffer overflow occurs, the transmitter is able to detect it and retransmits the codewords that were rejected by the receiver. The transmission and retransmission procedure is given as follows:

1) If the current time-out word is ACK'ed and its forward index f_T is less than N, the first (new) codeword in the input queue is transmitted. If the current time-out word is NAK'ed (or unACK'ed) and $0 < f_T < N$, a retransmission of the current time-out word is initiated. This is the selective-repeat process.
2) If $f_T = 0$, the current time-out word is the earliest word in the retransmission buffer that has not been ACK'ed. If the current time-out word is NAK'ed, then receiver buffer overflow occurs. In this case, all the codewords in the retransmission buffer with forward indices equal to or greater than N are moved back to the input queue for retransmission. These are codewords that have been transmitted; however, when they arrive at the receiver, the

receiver buffer has no space to store them (buffer overflow). Therefore these codewords must be retransmitted (see Figure 9).

3) If $f_T \geq N$, the first codeword in the input queue is the next to be transmitted (this may be a codeword that was moved back to the input queue from the retransmission buffer due to receiver buffer overflow).

With Yu-Lin's scheme, the receiver is also capable of detecting whether a received word is a new word or a word that his already been accepted and delivered to the user. Yu-Lin's selective-repeat ARQ provides significantly higher throughput than the SR+GBN ARQ for high bit-error rates as shown in Figure 10. This scheme can be generalized for a receiver buffer of size N. Of course, as ν increases, the throughput increases as shown in Figure 10.

The other selective-repeat ARQ scheme allowing receiver buffer overflow is due to Wang and Lin [30]. With Wang-Lin's scheme, when a codeword is rejected by the receiver due to buffer overflow, an overflow acknowledgement is sent back to the transmitter. Wang and Lin analyzed their scheme and obtained the throughput performance of any receiver buffer of finite size. The throughput performance of Wang-Lin's scheme is about the same as that of Yu-Lin's scheme.

5. Linear Block Codes for Error Detection

The reliability of an ARQ system depends on the choice of the code used for error detection. If the code for error detection is properly chosen, the probability that the receiver commits a decoding

error can be made very small.

For the past three decades, coding theorists have been primarily concerned with the problems of error correction such as deriving bounds on the minimum distance of codes, construction of codes with good error-correcting capability, and error-correction methods. Hundreds of papers have been published. However, very little amount of work has been done in the area of error-detection such as construction of good error-detecting codes. Only a handful of papers have been published [6-15].

Consider error detection with linear block codes on a binary symmetric channel (BSC) with transition probability ε. Korzhik [6] proved that there exists (n,k) linear codes with probability P_e of an undetected error satisfying the following bound

$$P_e \leq 2^{-(n-k)}\{(1-(1-\varepsilon)^k\} \tag{8}$$

for all n and k and all ε, $0 \leq \varepsilon \leq 1$. He proved this result by averaging P_e over all the systematic (n,k) linear block codes. It is an existence proof and no method has been found for constructing codes to satisfy the bound given by (8). However, a few small classes of known codes have been proved to satisfy a weaker bound,

$$P_e \leq 2^{-(n-k)} \tag{9}$$

for $0 \leq \varepsilon \leq 1/2$. These are the linear perfect codes [such as Hamming codes and the (23,12) Golay code], their dual codes,and the distance-5 primitive BCH codes [8,9]. If a code satisfying the bound of (9) is used for error detection in an ARQ system, the error probability P(E) of the system can be made very small by using a moderate number of

parity bits, say 32.

Recently, Kasami, Kløve, and Lin [13,14] considered the ensemble of all even-weight (n,k) linear codes and proved that there exists codes with P_e satisfying the following bound,

$$P_e \leq 2^{-(n-k)}\{1+(1-2\varepsilon)^n - 2(1-\varepsilon)^n\} \tag{10}$$

for $0 \leq \varepsilon \leq 1/2$. This bound is tighter than the bounds given by (8) and (9). They also established sufficient conditions for codes to satisfy their bound. Based on the sufficient conditions, they were able to prove that the following classes of codes satisfy the bound given by (10):

1) distance-8 (24,12) Golay code;
2) distance-4 Hamming codes of lengths 2^m-1 and 2^m;
3) distance-6 primitive BCH codes of length 2^m-1 with $m \geq 5$ (X+1 is a factor of the generator polynomial);
4) distance-6 extended primitive BCH codes of length 2^m;
5) distance-8 primitive BCH codes of length 2^m-1 with m odd and $m \geq 5$;
6) distance-8 extended primitive BCH codes of length 2^m with m odd and $m \geq 5$.

Of course, these classes of codes also satisfy the bounds given by (8) and (9), and they are good for error detection. For example, the CCITT recommendation X.25 for packet-switched data networks [31] adopts the distance-4 Hamming code with 16 parity bits for error detection. The natural length of this code is $n = 2^{15} - 1 = 32,767$ and the generator polynomial is

$$g(X) = (X+1)(X^{15} + X^{14} + X^{13} + X^{12} + X^4 + X^3 + X^2 + X + 1)$$

$$= X^{16} + X^{12} + X^5 + 1.$$

Let C be an (n,k) linear code which satisfies the bound $2^{-(n-k)}$ given by (9). It is important to note that a shortened code [1-5] obtained from C by deleting ℓ information bits does not necessarily satisfy the bound [9,11,12]. Whether a shortened code satisfies the bound depends on the degree ℓ of shortening [15]. Also a code obtained by interleaving C to degree λ [1-5] does not necessarily obey the bound.

One may wonder if an (n,k) linear code with minimum distance $d \geq 2t + 1$ is used for error detection, how its probability of error will behave. Kasami, Kløve, and Lin [14] proved the following results:

$$P_e \leq \begin{cases} \dfrac{1}{\binom{\lceil n/2 \rceil + t}{t}} 2^{-nE((2t+1)/n,\varepsilon)}, & \text{for } 0 \leq \varepsilon < \dfrac{2t+1}{n} < \dfrac{1}{2}, \\ \dfrac{1}{\binom{n-m+t}{t}} + \dfrac{1}{\binom{\lceil n/2 \rceil + t}{t}} 2^{-nE(m/n,\varepsilon)}, & \text{for } \dfrac{2t+1}{n} < \varepsilon < \dfrac{m}{n} \leq \dfrac{1}{2}, \end{cases} \quad (11)$$

where $E(x,\varepsilon) = H(\varepsilon) + (x-\varepsilon)\dfrac{d}{d\varepsilon} H(\varepsilon) > 0$ for $0 \leq \varepsilon < x$, and $H(\varepsilon)$ is the binary entropy function. We see from (11) that, if a code with $d/n > \varepsilon$ is used for error detection, its probability P_e of an undetected error decreases exponentially with n. If d and n are properly chosen, P_e can be made very small relative to P_c, thereby proving high reliability for an ARQ system.

Even though the double-error-correcting and some triple-error-correcting primitive BCH codes have been shown to obey the $2^{-(n-k)}$ bound, it is still unknown whether the entire class of primitive BCH codes obeys the bound. However, a large number of primitive BCH codes satisfy the following bound [3,32]: For a t-error-correcting primitive BCH code of length $n = 2^m-1$ with $n-k = mt$ parity-check digits and

$m \geq m_0(t)$,

$$P_e \leq \begin{cases} (1+\lambda_0 \cdot n^{-1/10})2^{-[n-k+nE((2t+1)/n,\varepsilon)]}, & \text{for } \varepsilon < \frac{2t+1}{n} \\ (1+\lambda_0 \cdot n^{-1/10})2^{-(n-k)}, & \text{for } \frac{2t+1}{n} \leq \varepsilon \end{cases} \quad (12)$$

where λ_0 is some constant. We see that these primitive BCH codes are quite good for error detection.

6. Hybrid ARQ Error Control Schemes

The drawbacks of the ARQ and FEC schemes can be overcome if the two basic error control schemes are properly combined. Such a combination of the two basic error control schemes is referred to as hybrid ARQ [3,19,33,34]. A hybrid ARQ system consists of an FEC subsystem contained in an ARQ system. The function of the FEC system is to reduce the frequency of retransmission by correcting the error patterns which occur most frequently. This increases the system throughput performance. However, when a less frequent error pattern occurs and is detected, the receiver requests a retransmission rather than passing the unreliably decoded message to the user. This increases the system reliability. As a result, a proper combination of FEC and ARQ provides higher reliability than an FEC system alone and higher throughput than a system with ARQ alone.

Hybrid ARQ schemes can be classified into two categories, namely type-I and type-II schemes [3,34]. A straightforward type-I hybrid ARQ scheme is to use a code which is designed for simultaneous error correction and error detection [1-5]. When a received word is detected in error, the receiver first attempts to correct the errors. If the number of errors (or the length of an error burst) is within

the designed error-correcting capability of the code, the errors will be corrected and the decoded message will be delivered to the user or saved in the buffer until it is ready to be passed to the user. If an uncorrectable error pattern is detected, the receiver rejects the received word and requests a retransmission. The retransmission is the same codeword. When the retransmitted codeword is received, the receiver again attempts to correct the errors (if any). If the decoding is not successful, the receiver again rejects the received word and requests another retransmission. This continues until the codeword is either successfully received or successfully decoded. The error-correction may be included in any type of ARQ scheme. Since a code is used for both error correction and detection in a type-I hybrid ARQ system, it requires more parity-check bits than a code used only for error detection in a pure ARQ system. As a result, the overhead for each transmission is increased. When the channel error rate is low, a type-I hybrid ARQ system has lower throughput than its corresponding ARQ system. However, when the channel error rate is high, a type-I hybrid ARQ system provides higher throughput than its corresponding ARQ system because its error-correction capability reduces the retransmission frequency as illustrated in Figure 11.

Type-I hybrid ARQ schemes are best suited for communication systems in which a fairly constant level of noise and interference is anticipated on the channel. In this case, enough error correction can be designed into the system to correct the vast majority of received words, thereby greatly reducing the number of retransmissions and enhancing the system performance. However, for a nonstationary channel where the bit-error rate changes, type-I hybrid ARQ scheme has

some drawbacks. When the channel error rate is low (e.g., a satellite channel in good weather), the transmission is smooth and no (or little) error-correction is needed. As a result, the extra parity-check bits for error correction included in each transmission represent a waste. When the channel is very noisy, the designed error-correcting capability may become inadequate. As a result, the frequency of retransmission increases and hence reduces the throughput. Several type-I hybrid ARQ schemes using either block or convolutional codes have been proposed and analyzed [33,35-41].

For a channel with a nonstationary bit-error rate, one would like to design an adaptive hybrid ARQ system. When the channel is quiet, the system behaves just like a pure ARQ system, with only parity-check bits for error detection being included in each transmission. Therefore the throughput performance is the same as that of a pure ARQ system. However, when the channel becomes noisy, extra parity-check bits are needed. This concept forms the basis of the type-II hybrid ARQ schemes. A message in its first transmission is coded with parity-check bits for error detection only, just like a pure ARQ scheme. When the receiver detects the presence of errors in a received word, it saves the erroneous word in a buffer, and at the same time requests a retransmission. The retransmission is not the original codeword but a block of parity-check bits which is formed based on the original message and an error correcting code. When this block of parity-check bits is received, it is used to correct the errors in the erroneous word that is stored in the receiver buffer. If error correction is not successful, the receiver requests a second retransmission of the NAK'ed word. The second retransmission may be

either a repetition of the original codeword or another block of parity-check bits. This depends on the retransmission strategy and the type of error-correcting code to be used.

The concept of parity retransmission for error correction was first introduced by Mandelbaum [42]. The first type-II hybrid ARQ using a parity-retransmission strategy was proposed by Metzner [43,44]. Metzner's scheme was later extended and modified by many others [34,45-53]. Among all the type-II hybrid ARQ schemes that have been reported, the scheme proposed by Lin and Yu [34] is the best analyzed and forms the basis for the schemes reported in References 46 to 53. The Lin-Yu scheme provides both high system reliability and high throughput performance.

With Lin-Yu's type-II hybrid ARQ scheme, two codes are used; one is a high rate (n,k) code C_0 which is designed for error detection only, and the other is a half-rate invertible (2k,k) code C_1 which is designed for simultaneous error correction and error detection [1-5], e.g., correcting t or fewer errors and simultaneously detecting d (d > t) or fewer errors. A (2k,k) code is said to be invertible if, knowing only the k parity-check bits of a codeword, the corresponding k information bits can be uniquely determined by an inversion operation [3,34]. That is to say, the parity-check section contains the same amount of information as the message section.

When a message D of k information bits is ready for transmission, it is encoded into a codeword (D,Q) of n bits based on the error-detecting code C_0, where Q denotes the n-k parity-check bits. The codeword (D,Q) is then transmitted. At the same time, the transmitter computes the k parity-check bits, denoted P(D), based on the message D

and the half rate invertible (2k,k) code C_1. Thus, (D,P(D)) is a codeword in C_1. The k-bit parity block P(D) is not transmitted but stored in the retransmission buffer of the transmitter for later use.

Let $(\tilde{D},\tilde{Q})$ denote the received word corresponding to (D,Q). When $(\tilde{D},\tilde{Q})$ is received, the receiver computes the syndrome of $(\tilde{D},\tilde{Q})$ based on C_0. If the syndrome is zero, then $\tilde{D}$ is assumed to be error-free and will be accepted by the receiver. If the syndrome is nonzero, the presence of errors in $(\tilde{D},\tilde{Q})$ is detected. The erroneous message $\tilde{D}$ is then saved in the receiver buffer and a NAK is sent to the transmitter. Upon receiving this NAK, the transmitter encodes the k-bit parity block P(D) into a codeword $(P(D),Q^{(1)})$ of n bits based on the error-detecting code C_0, where $Q^{(1)}$ denotes the n-k parity-check digits for P(D). Then the parity word $(P(D),Q^{(1)})$ is transmitted. Let $(\tilde{P}(D),\tilde{Q}^{(1)})$ denote the received word corresponding to $(P(D),Q^{(1)})$. When $(\tilde{P}(D),\tilde{Q}^{(1)})$ is received, its syndrome is computed based on C_0. If the syndrome is zero, then $\tilde{P}(D)$ is assumed to be error-free and the message D is recovered from $\tilde{P}(D)$ by inversion. If the syndrome is nonzero, then $\tilde{P}(D)$ and the erroneous message $\tilde{D}$ (stored in the receiver buffer) together are used for error correction based on the half-rate code C_1. If the errors in $(\tilde{D},\tilde{P}(D))$ form a correctable error pattern, they will be corrected. The decoded message D is then accepted by the receiver and an ACK is sent to the transmitter. If the errors in $(\tilde{D},\tilde{P}(D))$ form a detectable but not a correctable error pattern, then $\tilde{D}$ is discarded, the erroneous parity block $\tilde{P}(D)$ is stored in the receiver buffer, and a NAK is sent to the transmitter.

Upon receiving the second NAK for the message D, the transmitter resends the codeword (D,Q). When $(\tilde{D},\tilde{Q})$ is received, its syndrome is

again computed based on C_0. If the syndrome is zero, $\tilde{D}$ is assumed to be error-free and is accepted by the receiver, and the erroneous parity block $\tilde{P}(D)$ is then discarded. If the syndrome is nonzero, then $\tilde{D}$ and the erroneous parity block $\tilde{P}(D)$ (stored in the receiver buffer) together are used for error correction based on C_1. If error correction is not successful, then $\tilde{P}(D)$ is discarded, $\tilde{D}$ is stored in the receiver buffer, and a NAK is sent to the transmitter. The next retransmission will be the parity-word $(P(D),Q^{(1)})$. Therefore, the retransmissions are alternate repetitions of the parity-word $(P(D),Q^{(1)})$ and the information codeword (D,Q). The receiver stores the received message $\tilde{D}$ and the received parity block $\tilde{P}(D)$ alternately. The retransmissions continue until the message D is finally recovered either by inversion or by decoding. The throughput behavior of the above type-II hybrid ARQ scheme in the selective-repeat mode is illustrated in Figure 11.

The alternate parity-data retransmission strategy can be incorporated with any of the three basic types of ARQ schemes and their variations. It is particularly effective when it is used in conjunction with selective-repeat ARQ. Type-II selective-repeat hybrid ARQ schemes using the parity-data retransmission strategy and a finite receiver buffer have been proposed and analyzed by Lin and Yu [34] and Wang and Lin [48]. Lin and Yu showed that, even with a receiver buffer of size N, their type-II selective-repeat hybrid ARQ scheme can achieve the same throughput performance as the ideal selective-repeat ARQ with an infinite receiver buffer, by using a half-rate invertible code C_1 which is designed for correcting only a few errors, say $t = 4$ or 5. If C_1 is designed to correct more than 5 errors, their scheme

can provide much higher throughput than the ideal selective-repeat ARQ with an infinite receiver buffer, as shown in Figure 12. We see that there is a substantial trade-off between a small amount of error-correction and a large amount of buffer storage. Lin and Yu also showed that their hybrid scheme provides the same order of reliability as a pure ARQ scheme.

The decoding complexity for the type-II hybrid ARQ scheme using alternate parity-data retransmission and a half-rate invertible code is only slightly greater than that of a corresponding type-I hybrid ARQ scheme with the same designed error-correcting capability. The extra circuits needed are an inversion circuit based on the half-rate invertible code C_1, which is simply a linear sequential circuit, and an error detection circuit based on the error-detecting code C_0.

7. Type-II Hybrid ARQ Schemes Using Convolutional Codes

The alternate parity-data retransmission type-II hybrid ARQ scheme can be incorporated with a rate-1/2 convolutional code using either Viterbi or majority-logic decoding [49,50,52-54]. The error-detecting code C_0 is still an (n,k) block code. However, the error-correcting code C_1 is a half-rate (2,1,m) convolutional code with memory order m [3]. In the following we will describe a scheme using Viterbi decoding.

Let $G_1(X)$ and $G_2(X)$ be the two generator polynomials of the half-rate convolutional code C_1. When a data sequence of k bits is ready for transmission, it is first encoded into a codeword I(X) in C_0. The information sequence I(X) is then transformed into two sequences $P_1(X) = I(X)G_1(X)$ and $P_2(X) = I(X)G_2(X)$, each n+m bits long.

Note that the 2(n+m)-bit sequence obtained by interleaving $P_1(X)$ and $P_2(X)$ is a code sequence for $I(X)$ based on the half-rate convolutional code C_1. The sequence $P_1(X) = I(X)G_1(X)$ is then transmitted to the receiver and the sequence $P_2(X)$ is stored in the transmitter buffer for possible retransmission at a later time. Let $\tilde{P}_1(X)$ denote the received sequence corresponding to $P_1(X)$. When $\tilde{P}_1(X)$ is received, it is divided by $G_1(X)$. Let $\tilde{I}_1(X)$ and $\tilde{R}_1(X)$ be the quotient and remainder, respectively. If $\tilde{R}_1(X) = 0$, $\tilde{I}_1(X)$ is checked based on the error-detecting code C_0. If its syndrome $S_1(X)$ is zero, $\tilde{I}_1(X)$ is assumed to be error-free and identical to the transmitted information sequence $I(X)$. The receiver then accepts $\tilde{I}_1(X)$ (with n-k parity bits deleted). If $\tilde{R}_1(X) \neq 0$ or $S_1(X) \neq 0$, errors are detected in $\tilde{P}_1(X)$. $\tilde{P}_1(X)$ is then saved in the receiver buffer for reprocessing at a later time. At the same time, the receiver sends a NAK to the transmitter. Upon receiving this NAK, the transmitter sends the sequence $P_2(X) = I(X)G_2(X)$ to the receiver [first retransmission for $I(X)$]. Let $\tilde{P}_2(X)$ be the received sequence corresponding to $P_2(X)$. When $\tilde{P}_2(X)$ is received, it is divided by $G_2(X)$. Let $\tilde{I}_2(X)$ and $\tilde{R}_2(X)$ be the quotient and remainder, respectively. If $\tilde{R}_2(X) = 0$, $\tilde{I}_2(X)$ is then checked based on the error-detecting code C_0. If its syndrome $S_2(X)$ is zero, then $\tilde{I}_2(X)$ is assumed to be error-free and identical to the transmitted sequence $I(X)$. In this case, the receiver accepts $\tilde{I}_2(X)$ and discards the sequence $\tilde{P}_1(X)$. If $\tilde{R}_2(X) \neq 0$ or $S_2(X) \neq 0$, then $\tilde{P}_2(X)$ together with $\tilde{P}_1(X)$ (which is stored in the receiver buffer) are then decoded based on the half-rate convolutional code C_1 using a Viterbi decoder. Let $\tilde{I}^*(X)$ be the decoded sequence. $\tilde{I}^*(X)$ is then checked based on the error-detecting code C_0. If the syndrome $S(X)$ is zero, then the

receiver accepts $\tilde{I}^*(X)$. If its syndrome S(X) is not zero, then $\tilde{P}_1(X)$ is discarded, $\tilde{P}_2(X)$ is stored in the receiver buffer, and the receiver sends another NAK to the transmitter. The next retransmission will be $P_1(X)$. The alternate retransmissions of $P_1(X)$ and $P_2(X)$ continue until I(X) is finally recovered.

For receiver buffer size N, the throughput efficiencies of the above hybrid ARQ scheme for block length n = 1024 and various memory orders m are shown in Figure 13. We see that the scheme offers significantly better throughput performance than pure selective-repeat ARQ with infinite receiver buffer when the channel error rate is high.

One disadvantage of the above scheme is that the throughput efficiency drops rapidly to 0.5 when the channel is noisy enough to require retransmissions. A refinement of the above scheme, first introduced by Lugand and Costello [50], uses higher rate convolutional codes to achieve a higher throughput when the channel is noisy, at a cost of some increased receiver complexity.

In the Lugand-Costello scheme, the error-detecting code C_0 is still an (n,k) block code. However, the error-correcting code is a (3,2,m) convolutional code [3] denoted C_3 with generator matrix

$$G(X) = \begin{bmatrix} G_1(X) & 0 & G_{31}(X) \\ 0 & G_2(X) & G_{32}(X) \end{bmatrix}.$$

After a data sequence of k bits is encoded into a codeword I(X) in C_0, I(X) is transformed into two sequences $P_1(X) = I(X)G_1(X)$ and $P_{31}(X) = I(X)G_{31}(X)$. Note that the sequence obtained by interleaving $P_1(X)$ and $P_{31}(X)$ is a code sequence for I(X) based on the half-rate

convolutional code C_1 with generator polynomials $G_1(X)$ and $G_{31}(X)$. The sequence $P_1(X)$ is then transmitted to the receiver, while $P_{31}(X)$ is stored in the transmitter buffer for possible use at a later time. When $\tilde{P}_1(X)$ is received, the same procedure is followed as in the half-rate case. If errors are detected in $\tilde{P}_1(X)$, then $\tilde{P}_1(X)$ is saved in the receiver buffer, and the receiver sends a NAK1 message to the transmitter. Upon receiving a NAK1 message, the transmitter switches to transforming codewords $J(X)$ in C_0 into the two sequences $P_2(X) = J(X)G_2(X)$ and $P_{32} = J(X)G_{32}(X)$. These sequences, when interleaved, form a code sequence for $J(X)$ based on the half-rate convolutional code C_2 with generator polynomials $G_2(X)$ and $G_{32}(X)$. The sequence $P_2(X)$ is transmitted to the receiver, and $P_{32}(X)$ is stored in the transmitter buffer. When $\tilde{P}_2(X)$ is received, the same procedure is again followed. If errors are detected in $\tilde{P}_2(X)$, a NAK2 message is sent to the transmitter and $\tilde{P}_2(X)$ is stored. Upon receiving a NAK2, the transmitter sends the sequence $P_3(X) = P_{31}(X) + P_{32}(X) = I(X)G_{31}(X) + J(X)G_{32}(X)$ to the receiver (the first retransmission for both $I(X)$ and $J(X)$). Note that the sequence obtained by interleaving $P_1(X)$, $P_2(X)$, and $P_3(X)$ is a code sequence for $I(X)$ and $J(X)$ based on the two-thirds-rate convolutional code C_3. When $\tilde{P}_3(X)$ is received, then $\tilde{P}_3(X)$ together with $\tilde{P}_1(X)$ and $\tilde{P}_2(X)$ (which are stored in the receiver buffer) are decoded based on the code C_3 using a Viterbi decoder. Let $\tilde{I}^*(X)$ and $\tilde{J}^*(X)$ be the decoded sequences. They are then both checked based on the error-detecting code C_0. If both syndromes are zero, the receiver accepts $\tilde{I}^*(X)$ and $\tilde{J}^*(X)$. If both syndromes are nonzero, $\tilde{P}_1(X)$ is discarded, $\tilde{P}_2(X)$ and $\tilde{P}_3(X)$ are stored, and the receiver sends a NAK3 message to the transmitter. This instructs the

transmitter to resend $P_1(X)$. Retransmissions of $P_1(X)$, $P_2(X)$, and $P_3(X)$ continue until both I(X) and J(X) are finally recovered. If only one syndrome is zero, several options are available. These include subtracting the effect of the decoded sequence from $\tilde{P}_3(X)$, and then trying to decode the other sequence based on one of the half-rate codes C_1 or C_2. A full discussion and analysis of these options is given in [54].

The transmitter must keep track of all data sequences which have not been decoded and ACK'ed. In this way it knows whether to send new sequences encoded with C_1 or C_2. If more data sequences encoded with C_1 than with C_2 have not been ACK'ed, the transmitter must continue to encode new sequences using C_2 until the situation reverses. Then C_1 can be used for encoding new sequences again.

The size of the receiver buffer limits the length of time the receiver can wait before both a $\tilde{P}_1(X)$ and a $\tilde{P}_2(X)$ are received in error. If the receiver buffer fills up before a $\tilde{P}_2(X)$ is received in error, a special NAK can be sent to the transmitter requesting a transmission of $P_{31}(X)$ only. Then, if errors are detected in $\tilde{P}_{31}(X)$, the half-rate code C_1 can be used to decode $\tilde{P}_1(X)$ and $\tilde{P}_{31}(X)$. Details are given in [54].

Although the above scheme requires a more complex retransmission protocol, a more complex decoder (for the same undetected error probability), and a longer receiver buffer than the half-rate scheme, its throughput efficiency drops only to 0.66 when the channel becomes noisy enough to require retransmissions. A similar scheme using three-fourths-rate convolutional codes maintains a throughput efficiency of at least 0.75. The throughput efficiencies of the three

schemes discussed, all using a memory order 3 code, are compared in Figure 14.

8. Type-I Hybrid ARQ Schemes Using Convolutional Codes

On channels where a fairly constant noise level is anticipated, type-I hybrid ARQ schemes can offer a throughput advantage over type-II schemes (see Figure 11), as well as a simpler protocol. Type-I schemes using convolutional codes have been proposed by several authors [38,39,55-57]. In this section, we will review two recent type-I schemes for use in packet switching networks [39,57]. Both these schemes make use of convolutional codes with sequential decoding [2,3,5].

In conventional sequential decoding of convolutional codes, decoding proceeds until the received message is completely decoded or an erasure is declared. Erasures are normally declared when decoding time becomes excessive. In type-I hybrid ARQ sequential decoding, this time-out condition can be used as a signal to request a retransmission. Assume that a data sequence I(X) ready for transmission is encoded into a codeword V(X) in an (n,k,m) convolutional code C_1. V(X) is then transmitted over the channel, and the received sequence $\tilde{V}(X)$ is decoded by a sequential decoder. If decoding is completed within the time-out limit, the decoded sequence $\tilde{I}^*(X)$ is accepted by the receiver. Otherwise, a retransmission is requested. This is called the time-out algorithm (TOA) [57]. The time required to decode provides a natural error detecting mechanism with sequential decoding. When the received sequence is not very noisy, it is decoded quickly, and no retransmission is necessary. When the received

sequence is noisy, however, decoding takes a longer time due to the tree-searching rules of the decoder. If the time-out limit is exceeded, a retransmission is requested.

Since each decoded branch in the code tree results in k data bits being delivered to the user, the throughput efficiency of the TOA assuming an ideal SR protocol is defined as

$$\eta_{TOA} = \frac{1}{C_A} \left(\frac{k}{n}\right) ,$$

where C_A is the average number of computations the decoder performs per decoded branch, <u>including retransmissions</u>. In the noiseless case, $C_A = 1$ and $\eta_{TOA} = k/n$, the code rate. Note that since the number of computations performed by a sequential decoder is a random variable which depends on the noisiness of the received sequence, C_A must be computed as a statistical average. The existence of the optimum time-out limit which minimizes C_A and hence maximizes η_{TOA} has been shown analytically, and has also been verified by computer simulations [39]. When the optimum time-out limit is used, C_A increases only slightly with increasing code memory order m. Since the undetected error probability P_e is an exponentially decreasing function of m [39], large memory orders can be used to achieve extremely small undetected error probabilities without reducing the throughput.

A more sophisticated approach to detecting errors can be used to further improve the throughput efficiency. The metric of the best path in a sequential decoder is monitored. When the slope of this metric becomes too negative, i.e., the metric of the best path is decreasing too rapidly, decoding stops and a retransmission is requested. As in the TOA, the metric of the best path provides a

natural error-detecting capability. If the received sequence is not very noisy, the metric along the best path will tend to increase at a fairly uniform rate, with only occasional dips. The slope of this metric, averaged over an appropriate number of branches, will then remain positive. However, if the received sequence is noisy, the decoder may follow an incorrect path whose metric declines over several branches. The slope of the metric will then turn negative, and may exceed a threshold thereby triggering a retransmission request. This is called the slope control algorithm (SCA) [39]. The advantage of the SCA is that it can recognize a noisy received sequence as soon as erroneous bits are processed by the decoder, rather than waiting for the time-out limit to be exceeded. This reduces C_A, and hence increases throughput efficiency, compared to the TOA. The existence of an optimum threshold on the slope of the metric, which minimizes C_A and hence maximizes the throughput efficiency η_{SCA} of the SCA, has been shown analytically and verified by computer simulation [39]. Figure 15 compares the throughput efficiency of the TOA and SCA, both operating under optimum conditions, over a range of bit error rates corresponding to a very noisy channel. The code used in the comparison was a (2,1,11) convolutional code with optimum distance properties for sequential decoding. Clearly the SCA has a smaller C_A, and hence a larger throughput, than the TOA. Therefore the SCA provides a very efficient means of obtaining a high throughput and a low undetected error probability on noisy channels with relatively constant error statistics.

REFERENCES

1. E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

2. G.C. Clark, Jr. and J.B. Cain, Coding for Error Control, Plenum, New Jersey, 1981.

3. S. Lin and D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, New Jersey, 1983.

4. F.J. MacWilliams and N.J.A. Sloane, Theory of Error-Correcting Codes, North Holland, Amsterdam, 1977.

5. W.W. Peterson and E.J. Weldon, Jr., Error-Correcting Codes, 2nd Ed., MIT Press, Cambridge, Mass., 1972.

6. V.I. Korzhik, "Bounds on Undetected Error Probability and Optimum Group Codes in a Channel with Feedback", Radiotecknika, 20, Vol. 1, pp. 27-33, 1965. (English translation: Telecommunication and Radio Engineering, 2, pp. 87-92, January, 1965.)

7. V.I. Levenshtein, "Bounds on the Probability of Undetected Error", Problemy Perdachi Informatsii, Vol. 13, No. 1, pp. 3-18, 1977. (English translation: Problem of Information Transmission, Vol. 13, No. 1, pp. 1-12, 1978.)

8. S.K. Leung-Yan-Cheong, M.E. Hellman, "Concerning a Bound on Undetected Error Probability", IEEE Transactions on Information Theory, Vol. IT-22, No. 2, pp. 235-237, March, 1976.

9. S.K. Leung-Yan-Cheong, E.R. Barnes, and D.U. Friedman, "Some Properties of Undetected Error Probability of Linear Codes", IEEE Transactions on Information Theory, Vol. IT-25, No. 1, pp. 110-112, January, 1979.

10. J. Massey, "Coding Techniques for Digital Data Networks", Proceedings of the International Conference on Information Theory and Systems, NTG-Fachberichte, Vol. 65, Berlin, Germany, September 18-20, 1978.

11. J.K. Wolf, A.M. Michelson, and A.H. Levesque, "On the Probability of Undetected Error for Linear Block Codes", IEEE Transactions on Communications, Vol. COM-30, No. 2, pp. 317-324, February, 1982.

12. R. Padovani and J.K. Wolf, "Data Transmission Using Error Detection Codes", GLOBECOM '82 Conf. Rec., C6.2, Miami, November 29 to December 2, 1982.

13. T. Kasami, T. Kløve, and S. Lin, "On the Probability of Undetected Error of Linear Block Codes", GLOBECOM '82 Conf. Rec., Miami, November 29 to December 2, 1982.

14. T. Kasami, T. Kløve, and S. Lin, "Error Detection with Linear Block Codes", IEEE Transaction on Information Theory, Vol. IT-29, No. 1, January, 1983.

15. T. Fujiwara, A. Kitai, S. Yamamura, T. Kasami, and S. Lin, "On the Probability of Undetected Error for Shortened Cyclic Hamming Codes", Proceedings of the Fifth Conference on Information Theory and Its Application, Hachimantal, Japan, October, 1982.

16. J.J. Metzner and K.C. Morgan, "Coded Feedback Communication Systems", Proceedings of Nat. Electron. Conf., Chicago, Illinois, 1960.

17. B. Reiffen, W.G. Schmidt, and H.L. Yudkin, "The Design of an Error-Free Data Transmission System for Telephone Circuits", AIEE Trans. Commun. Electron., Vol. 80, pp. 224-231, 1961.

18. W.R. Cowell and H.O. Burton, "Computer Simulation of the Use of a Group Code with Retransmission on a Gilbert Burst Channel", AIEE Trans. Commun. Electron., pt. 1, pp. 577-580, January, 1962.

19. R.J. Benice and A.H. Frey, Jr., "An Analysis of Retransmission Systems", IEEE Trans. Commun. Syst., Vol. CS-12, pp. 135-144, 1964.

20. J.J. Metzner and K.C. Morgan, "Word Selection Procedures for Error-Free Communication Systems", New York University, New York, 2nd Sci. Rep., Contract AF 19(628)-4321, June, 1965.

21. H.O. Burton and D.D. Sullivan, "Errors and Error Control", Proc. IEEE, 60(11), pp. 1293-1310, November, 1972.

22. General Information: Binary Synchronous Communication, IBM Publication GA27-3004, IBM Corp., White Plains, N.Y., 1969.

23. A.R.K. Sastry, "Improving Automatic Repeat-Request (ARQ) Performance on Satellite Channels Under High Error Rate Conditions", IEEE Trans. Commun., COM-23, pp. 436-439, April, 1975.

24. J.M. Morris, "On Another Go-Back-N ARQ Technique for High Error Rate Conditions", IEEE Trans. Commun., COM-26, pp. 187-189, January, 1978.

25. S. Lin and P.S. Yu, "An Efficient Error Control Scheme for Satellite Communications", IEEE Trans. Commun., COM-28, pp. 395-401, March, 1980.

26. J.J. Metzner, "A Study of an Efficient Retransmission Strategy for Data Links", NTC'77 Conf. Rec., pp. 3B:1-1 to 3B:1-5.

27. M.J. Miller and S. Lin, "The Analysis of Some Selective-Repeat ARQ Schemes with Finite Receiver Buffer", IEEE Trans. Commun., COM-29, pp. 1307-1315, September, 1981.

28. E.J. Weldon, Jr., "An Improved Selective-Repeat ARQ Strategy", IEEE Trans. Commun., Vol. COM-30, No. 3. pp. 480-486, March, 1982.

29. P.S. Yu and S. Lin, "An Efficient Selective-Repeat ARQ Scheme for Satellite Channels and Its Throughput Analysis", IEEE Trans. Commun., COM-29, pp. 353-363, March 1981.

30. Y.-M. Wang and S. Lin, "Throughput Analysis of Selective-Repeat ARQ Schemes with Finite Receiver Buffer", presented at the 1982 IEEE International Symposium on Information Theory, Les Ares, France, June 21-25, 1982.

31. CCITT: Recommendation X.25 "Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode on public data networks", with Plenary Assembly, Doc. No. 7, Geneva, 1980.

32. V.M. Sidelnikov, "Weight Spectrum of Binary Bose-Chaudhuri-Hoquinghem Codes", Problemy Peridachi Informatsii, Vol. 7, No. 1, pp. 14-22, January-March 1971.

33. E.Y. Rocher and R.L. Pickholtz, "An Analysis of the Effectiveness of Hybrid Transmission Schemes", IBM J. Res. Dev., pp. 426-433, July, 1970.

34. S. Lin and P.S. Yu, "A Hybrid ARQ Scheme with Parity Retransmission for Error Control of Satellite Channels", IEEE Trans. Commun., COM-30, No. 7, July, 1982.

35. K. Brayer, "Error Control Techniques Using Binary Symbol Burst Codes", IEEE Trans. Commun., COM-16, pp. 199-214, April, 1968.

36. A.R.K. Sastry, "Performance of Hybrid Error Control Schemes on Satellite Channels", IEEE Trans. Commun., COM-23, pp. 689-694, July, 1975.

37. A.R.K. Sastry and L.N. Kanal, "Hybrid Error Control Using Retransmission and Generalized Burst-Trapping Codes", IEEE Trans. Commun., COM-24, pp. 385-393, April, 1976.

38. H. Yamamoto and K. Itoh, "Viterbi Decoding Algorithm for Convolutional Codes with Repeat Requests", IEEE Trans. on Inf. Theory, IT-26, pp. 540-547, September, 1980.

39. A. Drukarev and D.J. Costello, Jr., "Hybrid ARQ Error Control Using Sequential Decoding", IEEE Trans. on Inf. Theory, IT-29, July, 1983.

40. A. Drukarev and D.J. Costello, Jr., "A Comparison of Block and Convolutional Codes in ARQ Error Control Schemes", IEEE Trans. Commun., Vol. COM-30, No. 11, pp. 2449-2455, November, 1982.

41. C.S.K. Leung and A. Lam, "Forward Error Correction for an ARQ Scheme", IEEE Trans. Commun., Vol. COM-29, No. 10, pp. 1514-1519, October 1981.

42. D.M. Mandelbaum, "Adaptive-feedback Coding Scheme Using Incremental Redundancy", IEEE Trans. Information Theory, Vol. IT-20, pp. 388-389, May, 1974.

43. J.J. Metzner, "Improvements in Block-retransmission Schemes", presented at the IEEE Int. Symp. Inform. Theory, Ithaca, NY, October 10-14, 1977.

44. J.J. Metzner, "Improvements in Block-retransmission Scheme", IEEE Trans. Commun., Vol. COM-27, pp. 525-532, February, 1979.

45. T.C. Ancheta, "Convolutional Parity Check Automatic Repeat Request", presented at the 1979 IEEE Int. Symp. Inform. Theory, Grignano, Italy, June 25-29, 1979.

46. S. Lin and J.S. Ma, "A Hybrid ARQ System with Parity Retransmision for Error Correction", IBM Res. Rep. 7478(32232), January 11, 1979.

47. S. Lin and .S. Yu, "SPEC--An Effective Hybrid-ARQ Scheme", IBM Res. Rep. 7591 (32852), April 4, 1979.

48. Y.-M. Wang and S. Lin, "A Modified Selective-Repeat Type-II Hybrid ARQ System and Its Performance Analysis", IEEE Trans. Commun., Vol. 31, No. 5, May, 1983.

49. Y.-M. Wang and S. Lin, "A Parity-Retransmission Hybrid ARQ Using a Convolutional Code and Viterbi Decoding for Error Control", GLOBECOM'82, Conf. Rec., E7.1, Miami, November 29 to December 2, 1982.

50. L. Lugand and D.J. Costello, Jr., "A Comparison of Three Hybrid ARQ Schemes Using Convolutional Codes on a Nonstationary Channel", GLOBECOM'82, Conf. Rec., Miami, November 29 to December 2, 1982.

51. R.A. Comroe and D.J. Costello, Jr., "ARQ Schemes for Data Transmission in Mobile Radio Systems", submitted to IEEE Trans. Comm., 1983.

52. M.J. Miller and S. Lin, "Parity Retransmission ARQ Using Convolutional Codes", submitted to IEEE Trans. Information Theory, 1982.

53. M.J. Miller, "ARQ Systems", Ph.D. Dissertation, Electrical Engineering Department, University of Hawaii, Honolulu, Hawaii, 1982.

54. L.R. Lugand, "ARQ Schemes Using Convolutional Codes and Viterbi Decoding over Non-Stationary Channels", Elec. Engr. Dept. Tech. Rept. EE8212-32-02, Ill. Inst. of Tech., Chicago, IL, 1982.

55. K.E. Perry and J.M. Wozencraft, "SECO: A Self-Regulating Error Correcting Coder-Decoder", IRE Trans. Information Theory, Vol. IT-8, pp. 128-135, September, 1962.

56. R.J. Fang, "Lower Bounds on Reliability Functions of Variable Length Nonsystematic Convolutional Codes for Channels with Noiseless Feedback", IEEE Trans. Information Theory, Vol. IT-17, pp. 161-171, March, 1971.

57. R.E. Kahn, S.A. Gronemeyer, J. Burchfiel, and R.C. Kunzelman, "Advances in Packet Radio Technology", Proc. IEEE, Vol. 66, pp. 1468-1496, November, 1978.

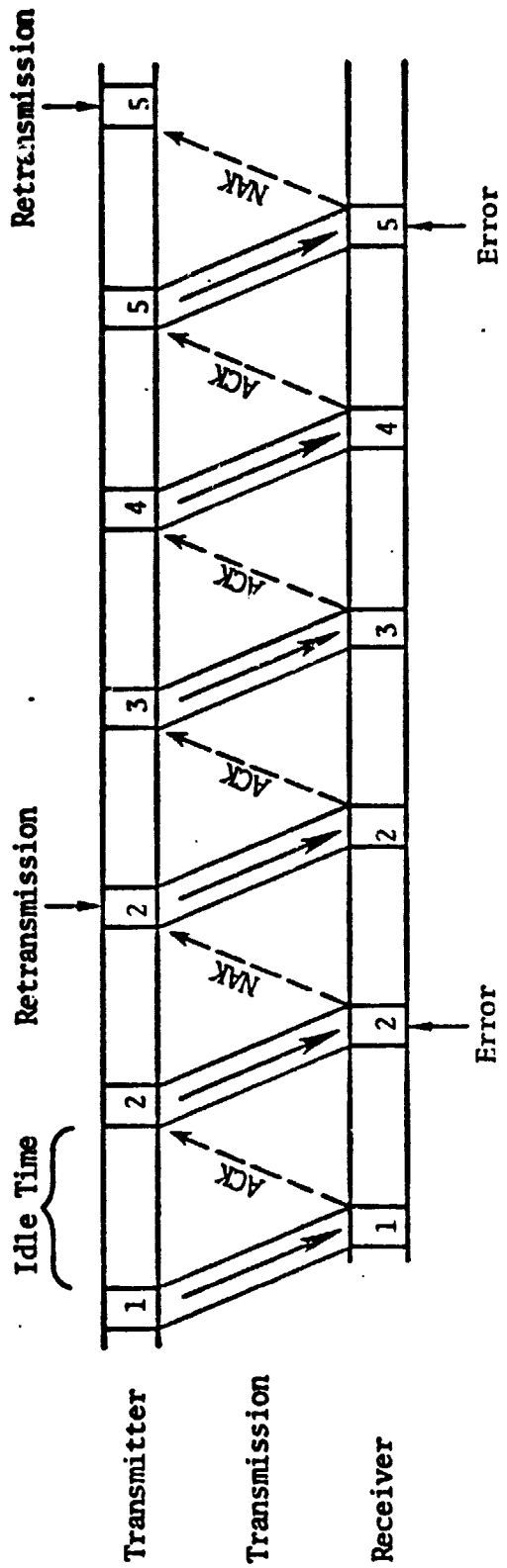

Figure 1 Stop-and-wait ARQ

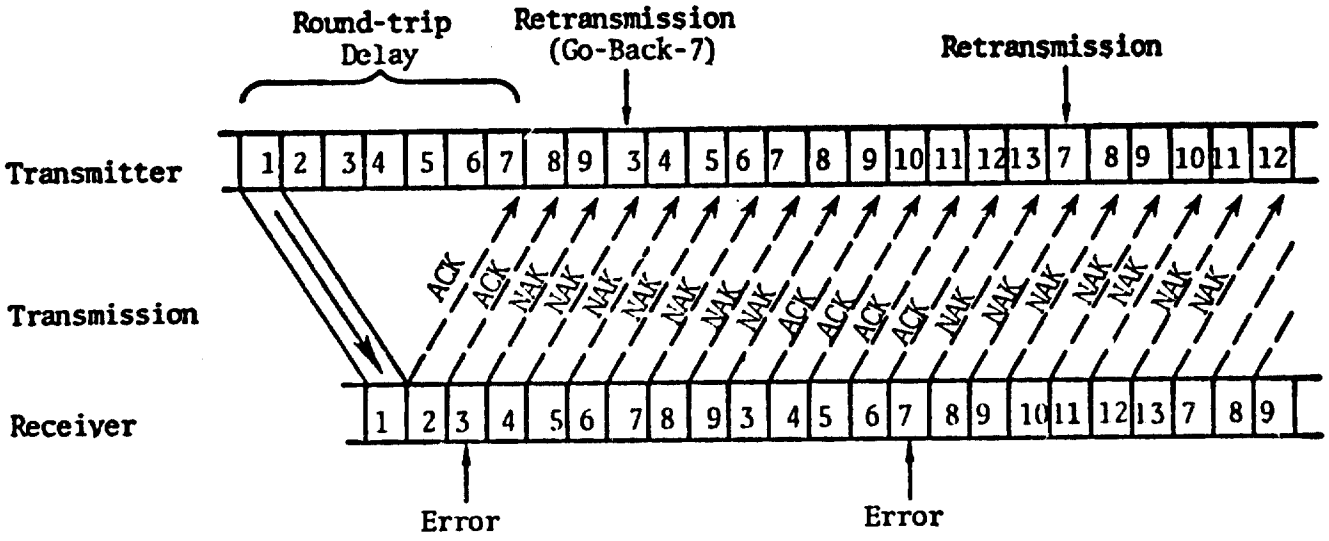

Figure 2 Go-Back-N ARQ with N=7

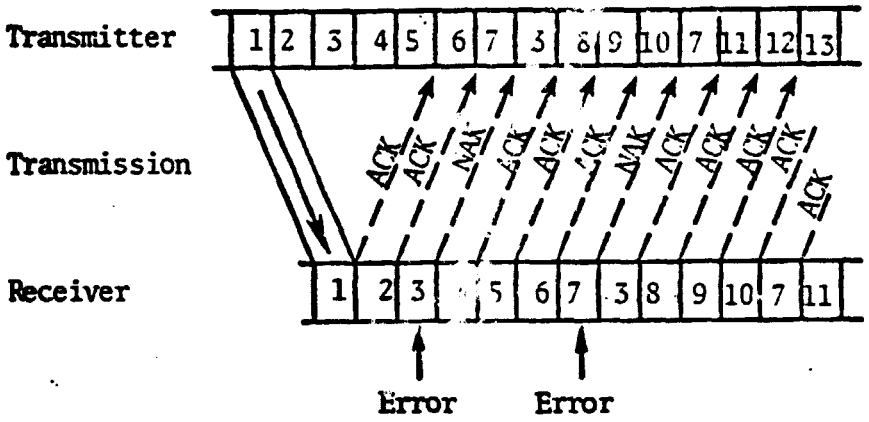

Figure 3 Selective-repeat ARQ

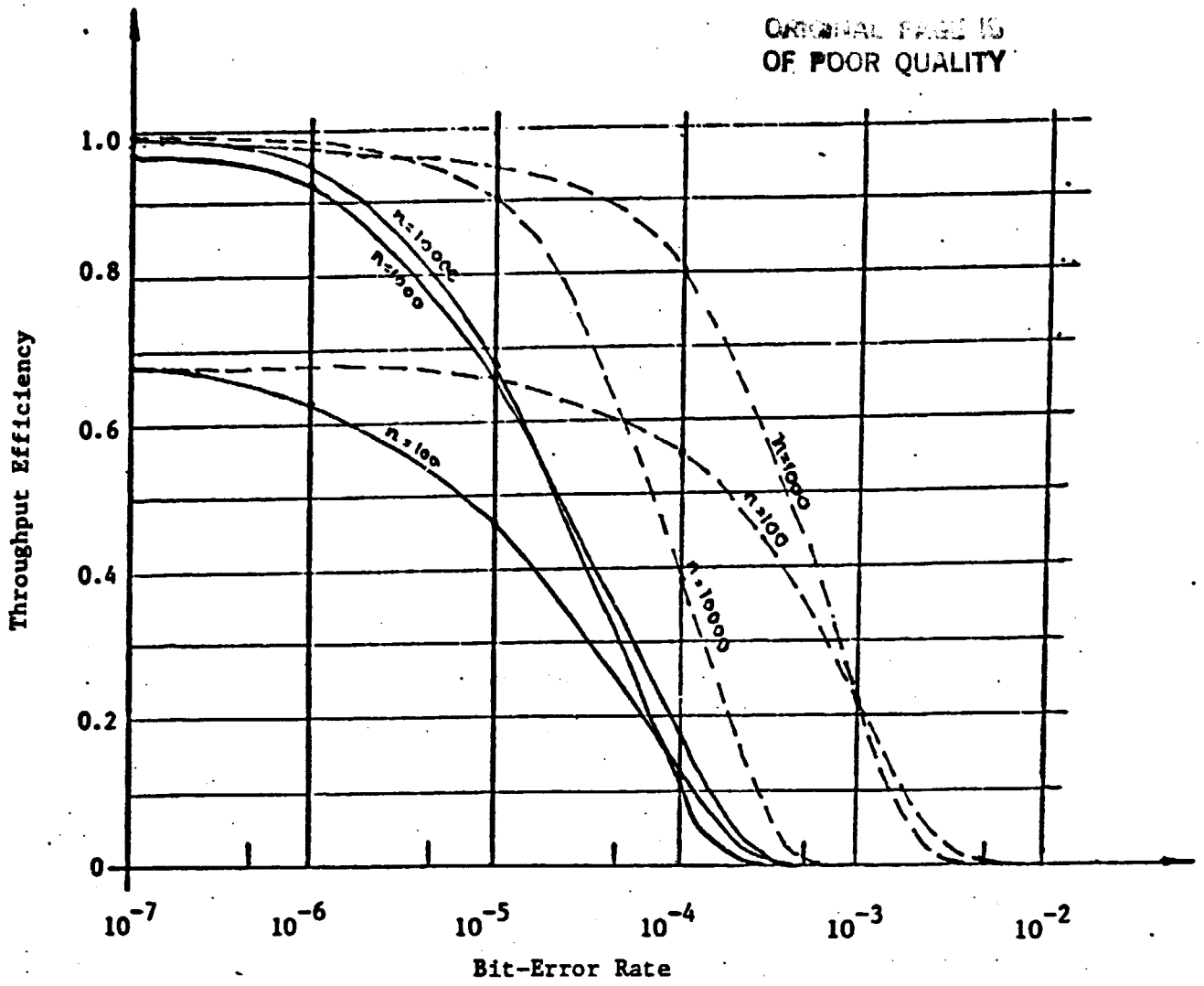

Figure 4 Throughput versus channel BER for go-Back-N ARQ for various code block lengths (n)

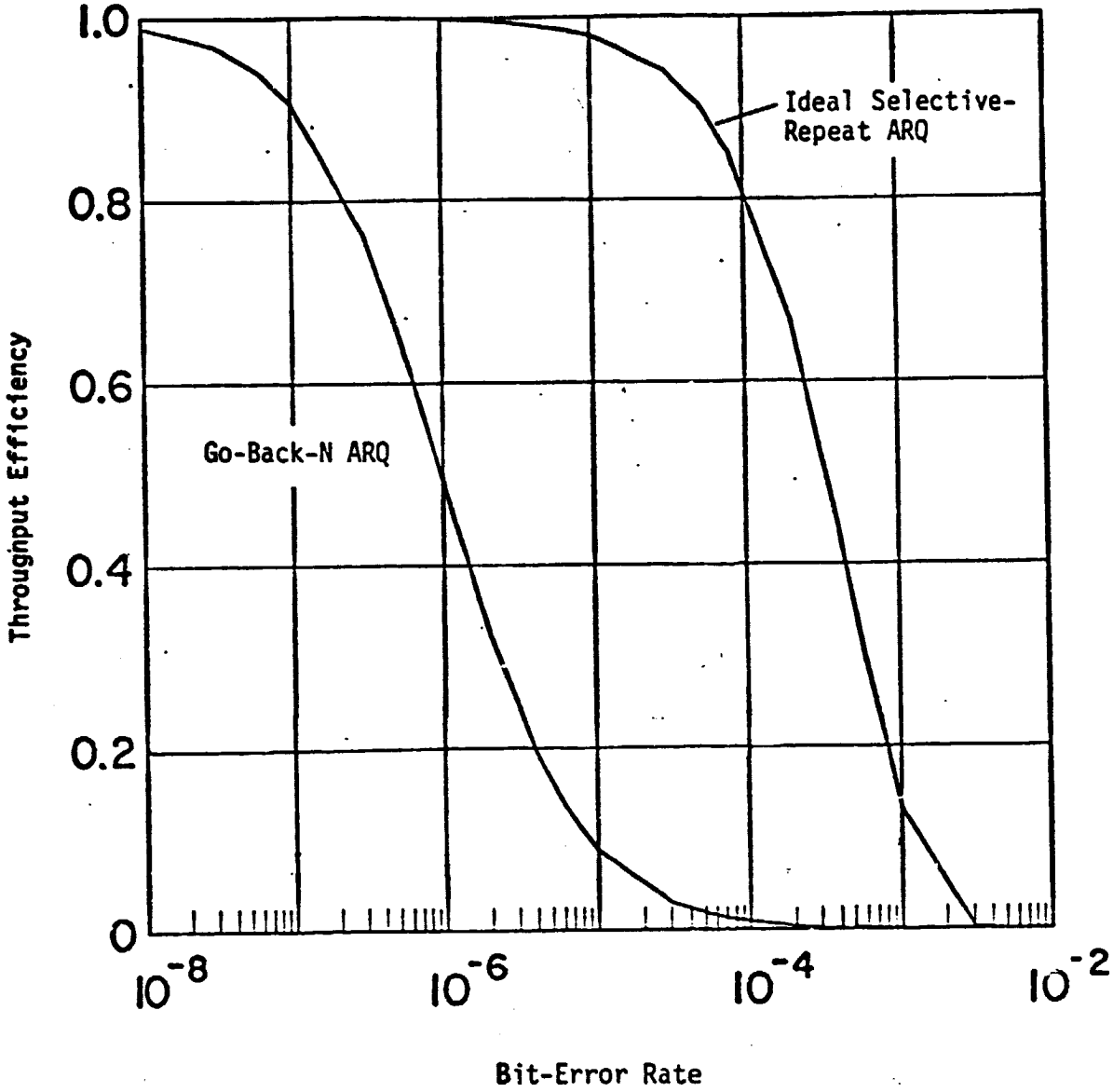

Figure 5 Throughput efficiencies: the ideal selective-repeat ARQ with infinite receiver buffer and the go-back-N ARQ with code block length n=2024.

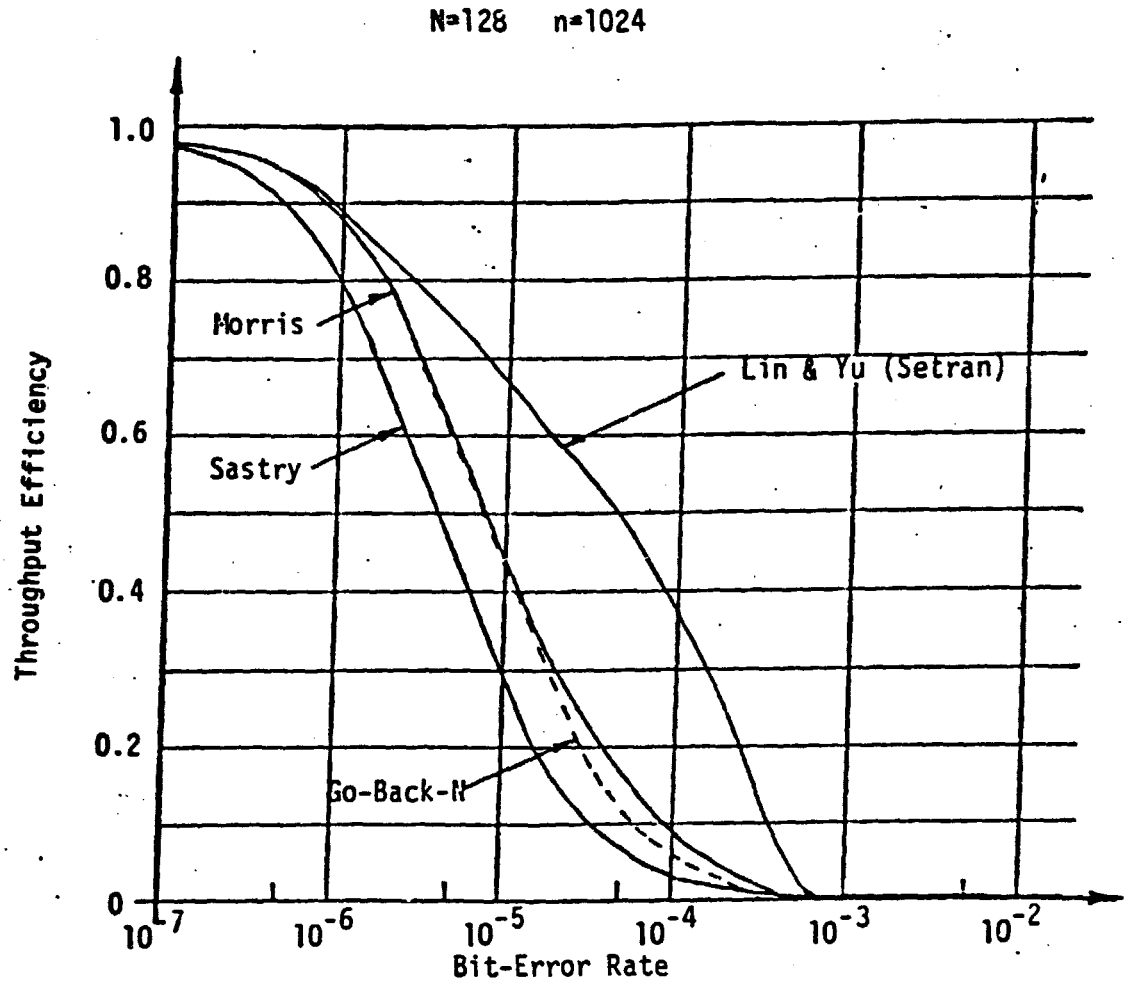

Figure 6 Basic go-back-N scheme and its variations

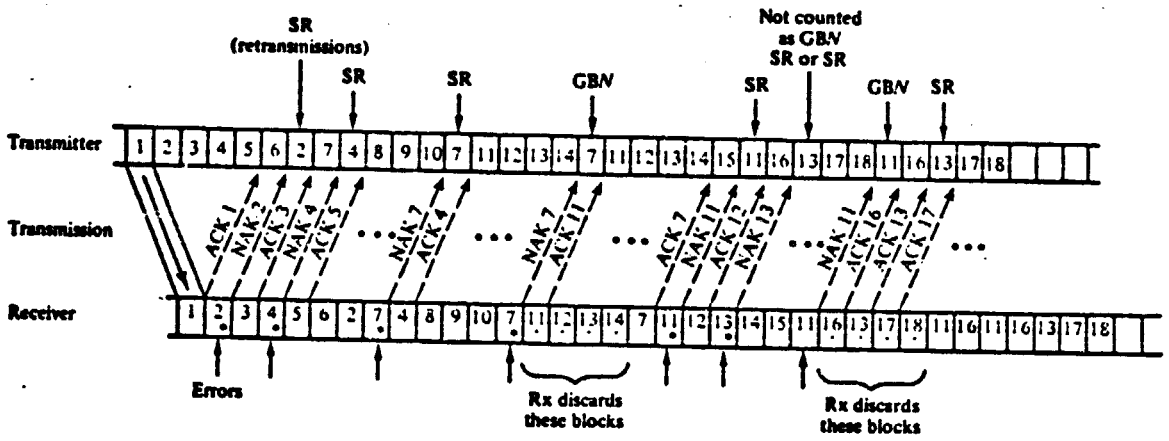

Figure 7 SR + GBN Scheme for $\nu = 1$

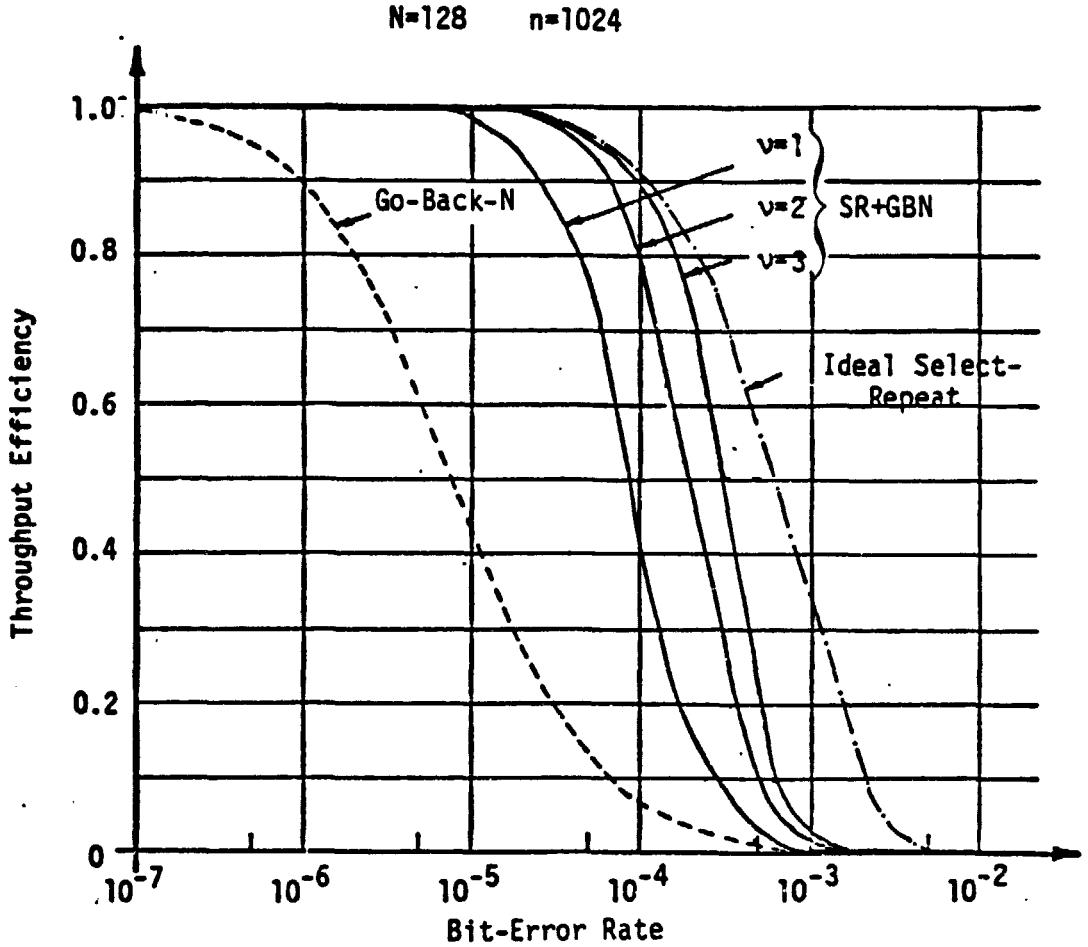

Figure 8 Throughput performance of the SR+GBN ARQ for various values of ν

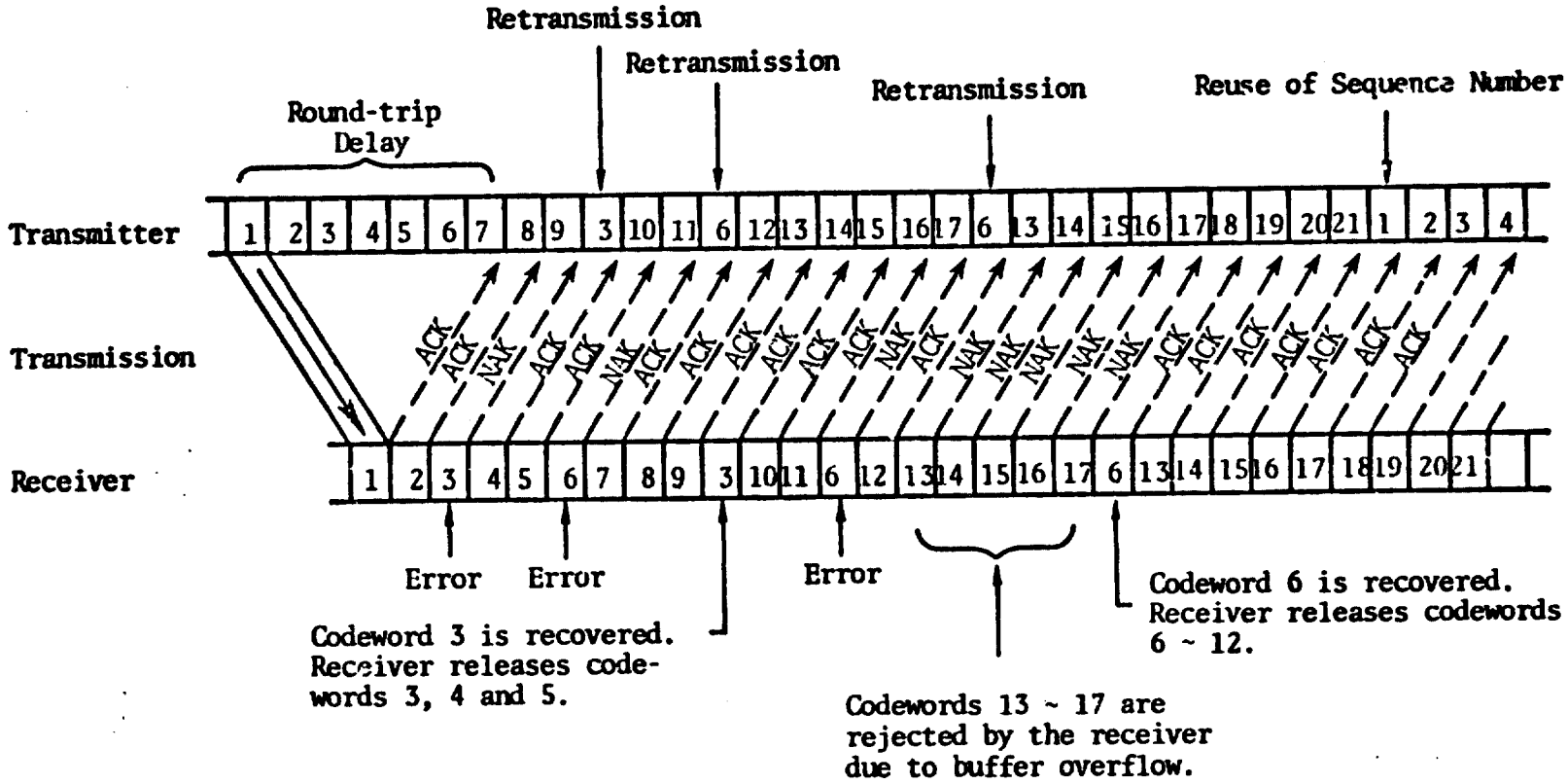

Figure 9 Selective-repeat ARQ with finite receiver buffer of size N=7

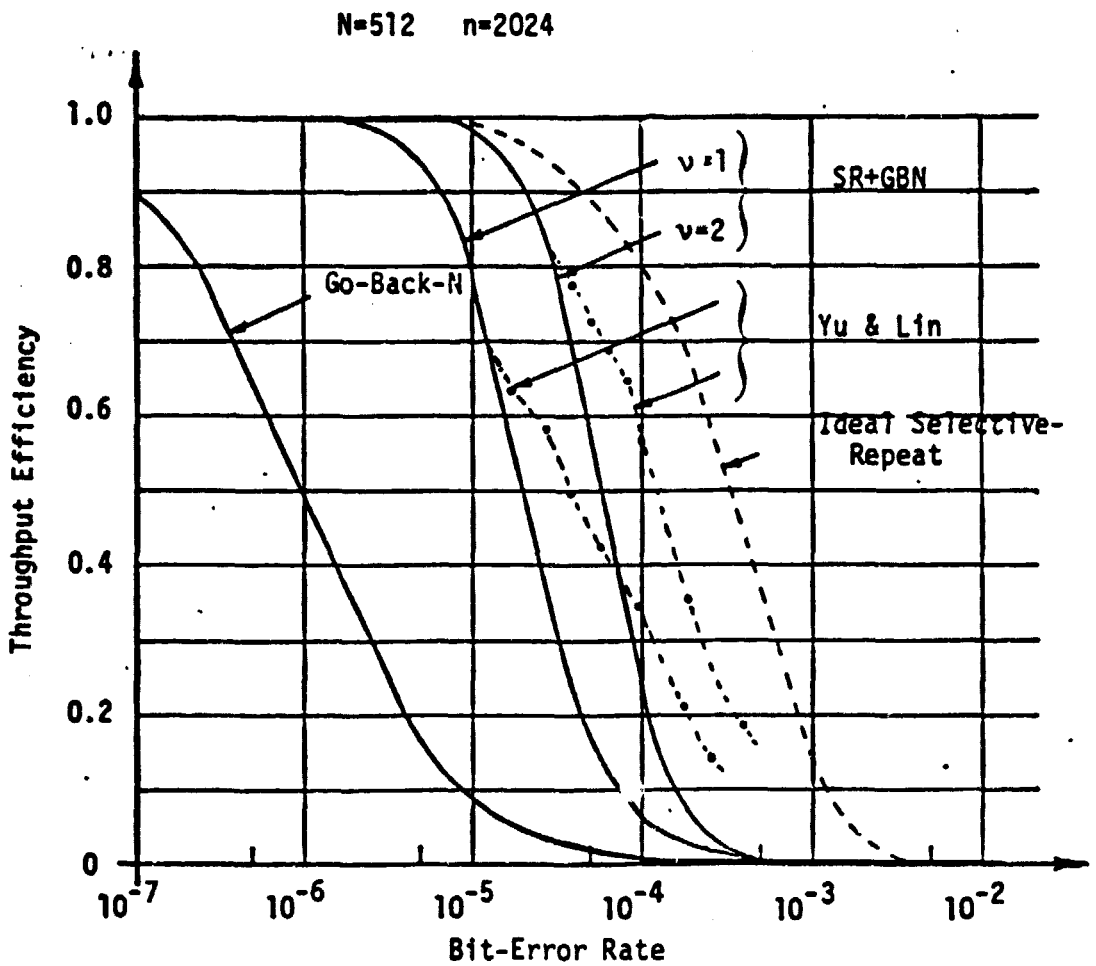

Figure 10 Throughput performance of Yu-Lin's selective-repeat ARQ

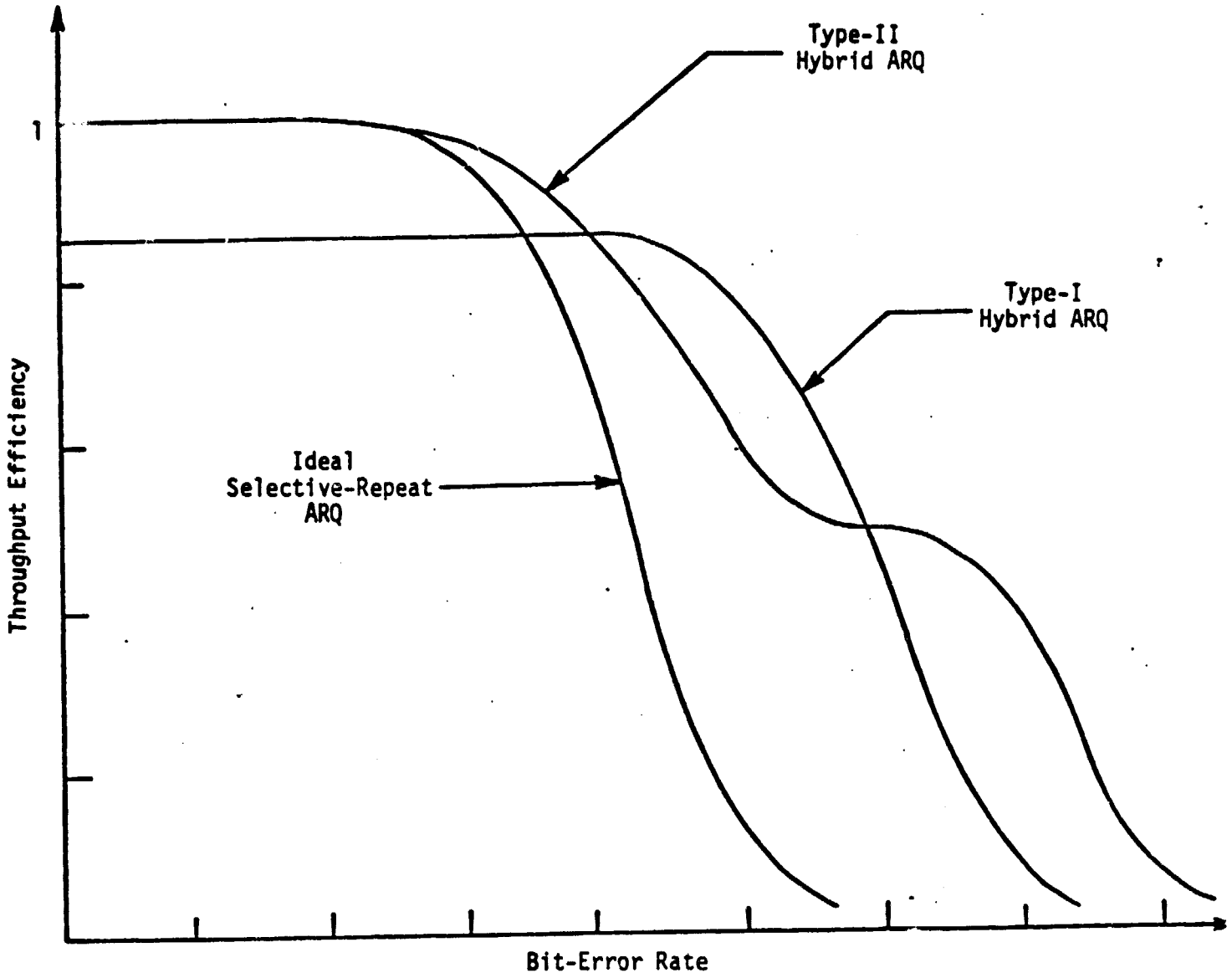

Figure 11 Comparison of Hybrid ARQ schemes

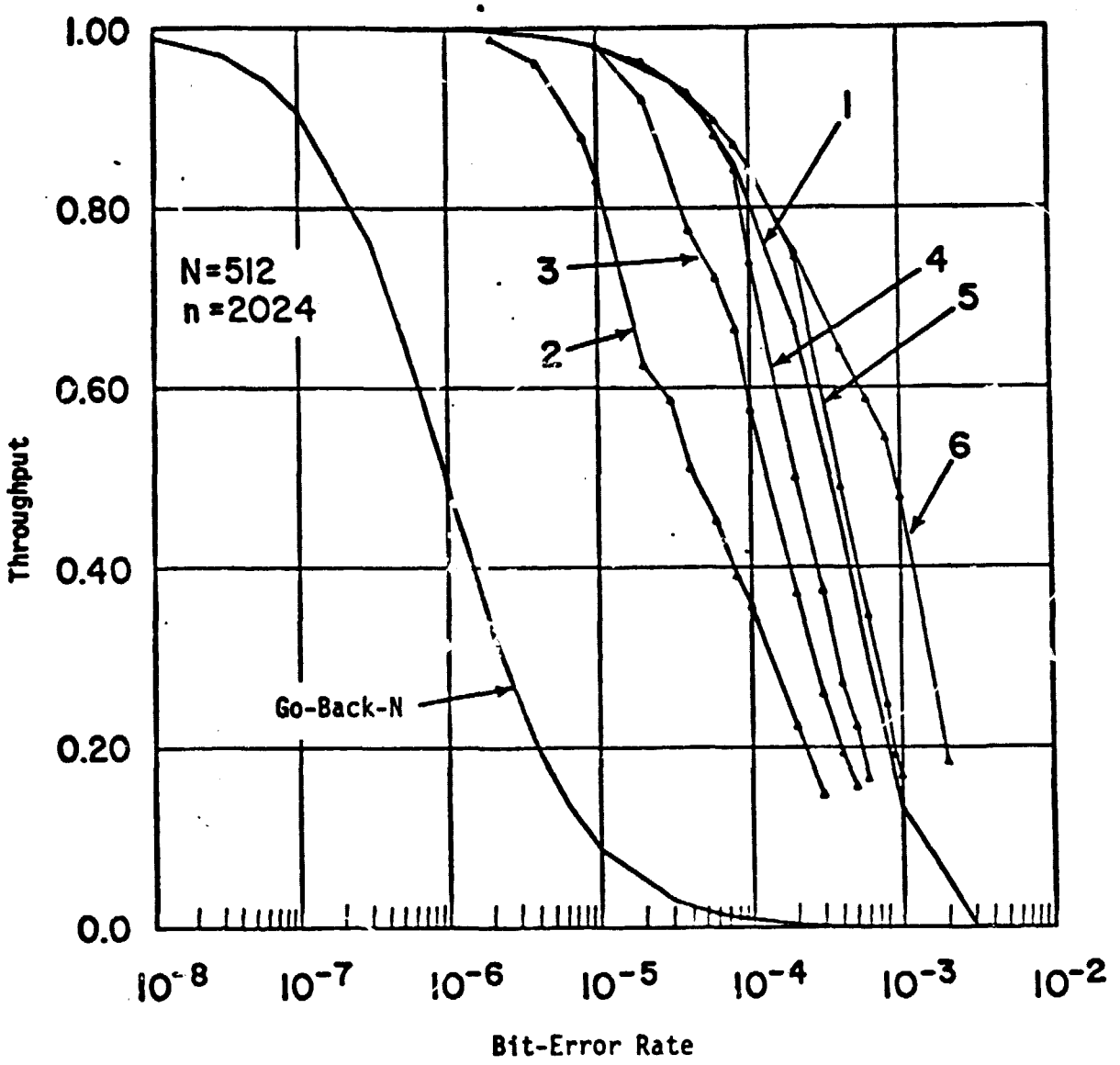

Figure 12 The throughput of various ARQ schemes with N=512 and n=2024: (1) ideal selective-repeat with infinite receiver buffer; (2) and (3) Yu-Lin selective-repeat ARQ with receiver buffer of size N and 2N; (4), (5) and (6) the type-II selective-repeat hybrid ARQ with receiver buffer of size N and error correction parameter t=3, 5 and 10.

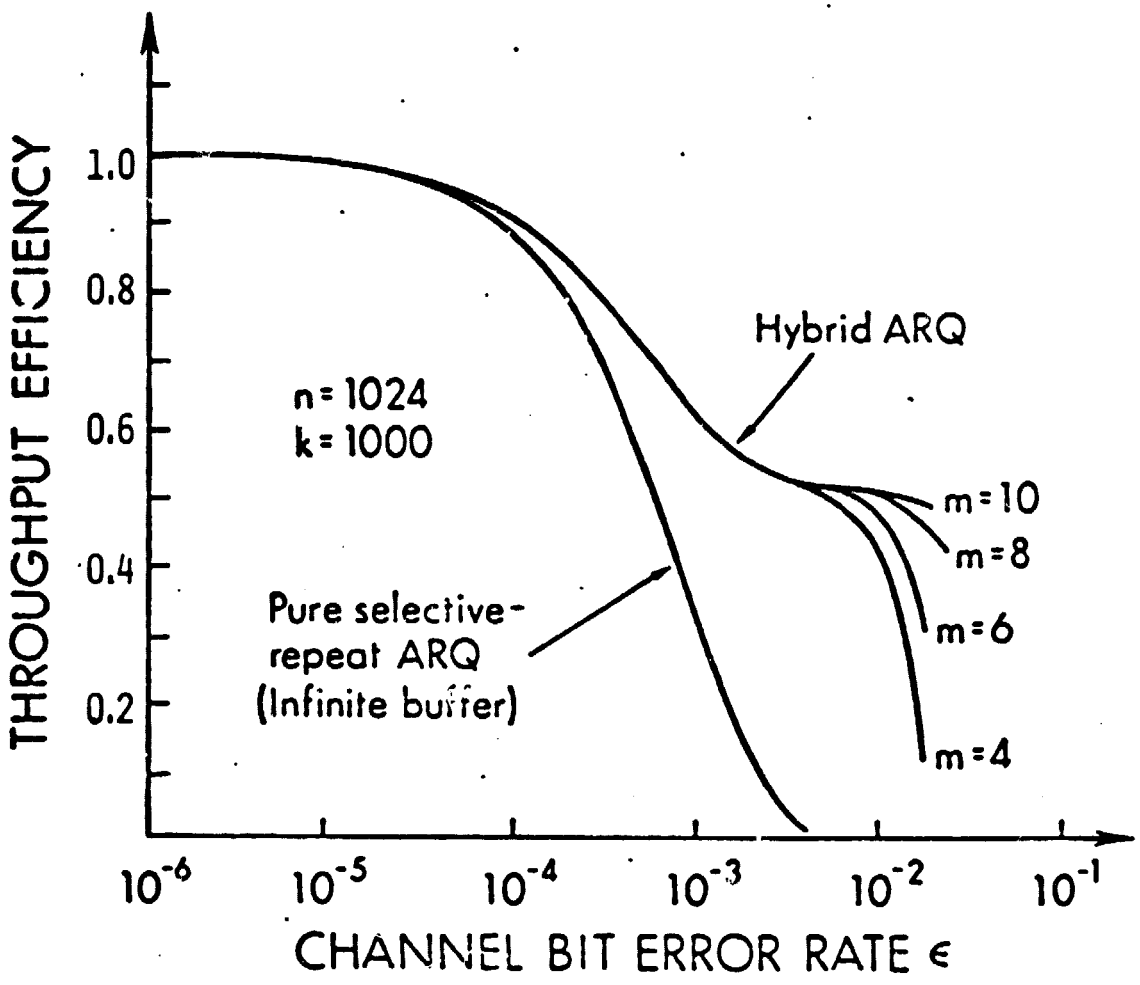

Figure 13 Throughput performance of the type-II hybrid ARQ using a half-rate convolutional code with n=1024 and k=1000.

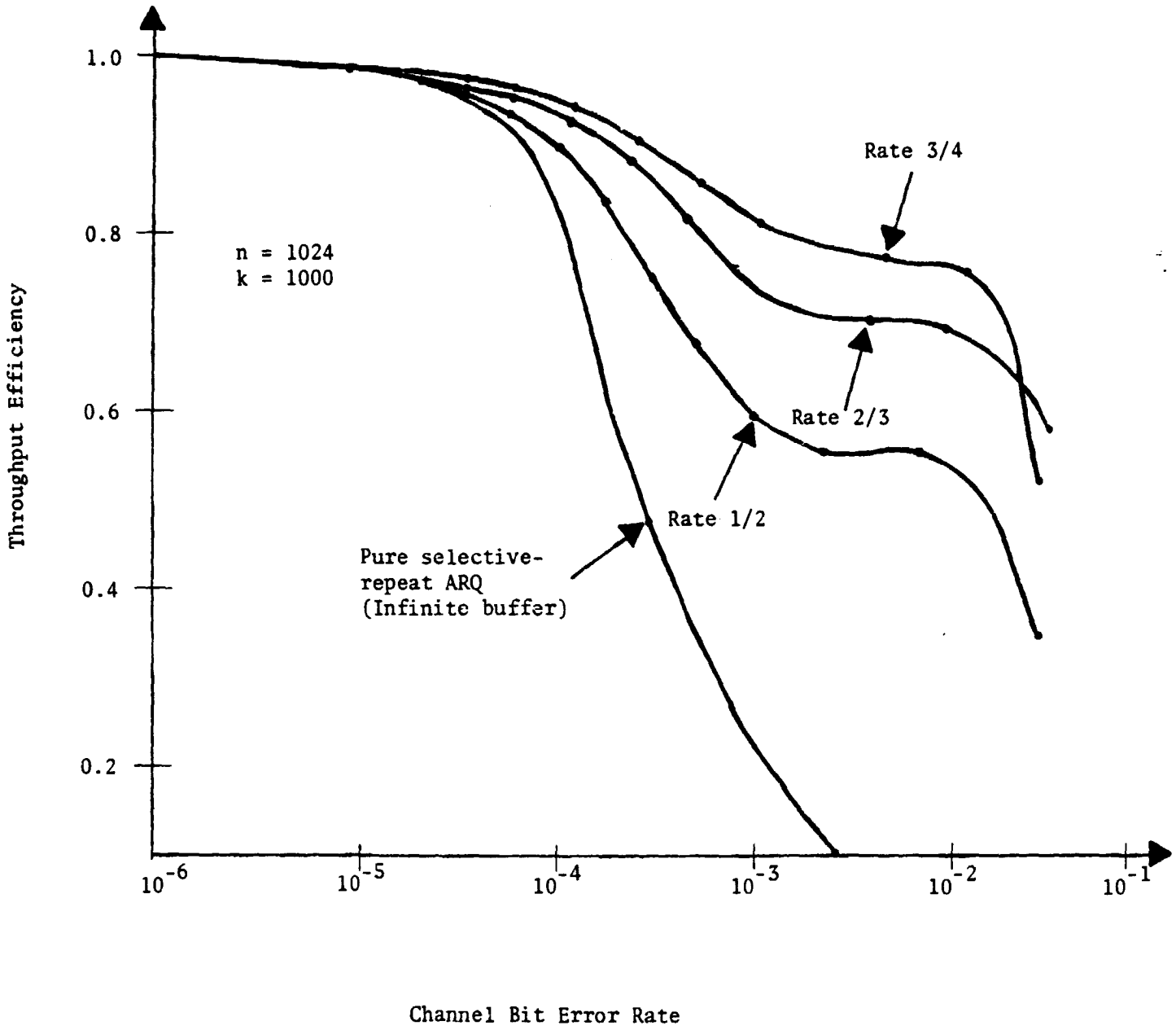

Figure 14 Throughput performance of three type-II hybrid ARQ schemes using convolutional codes.

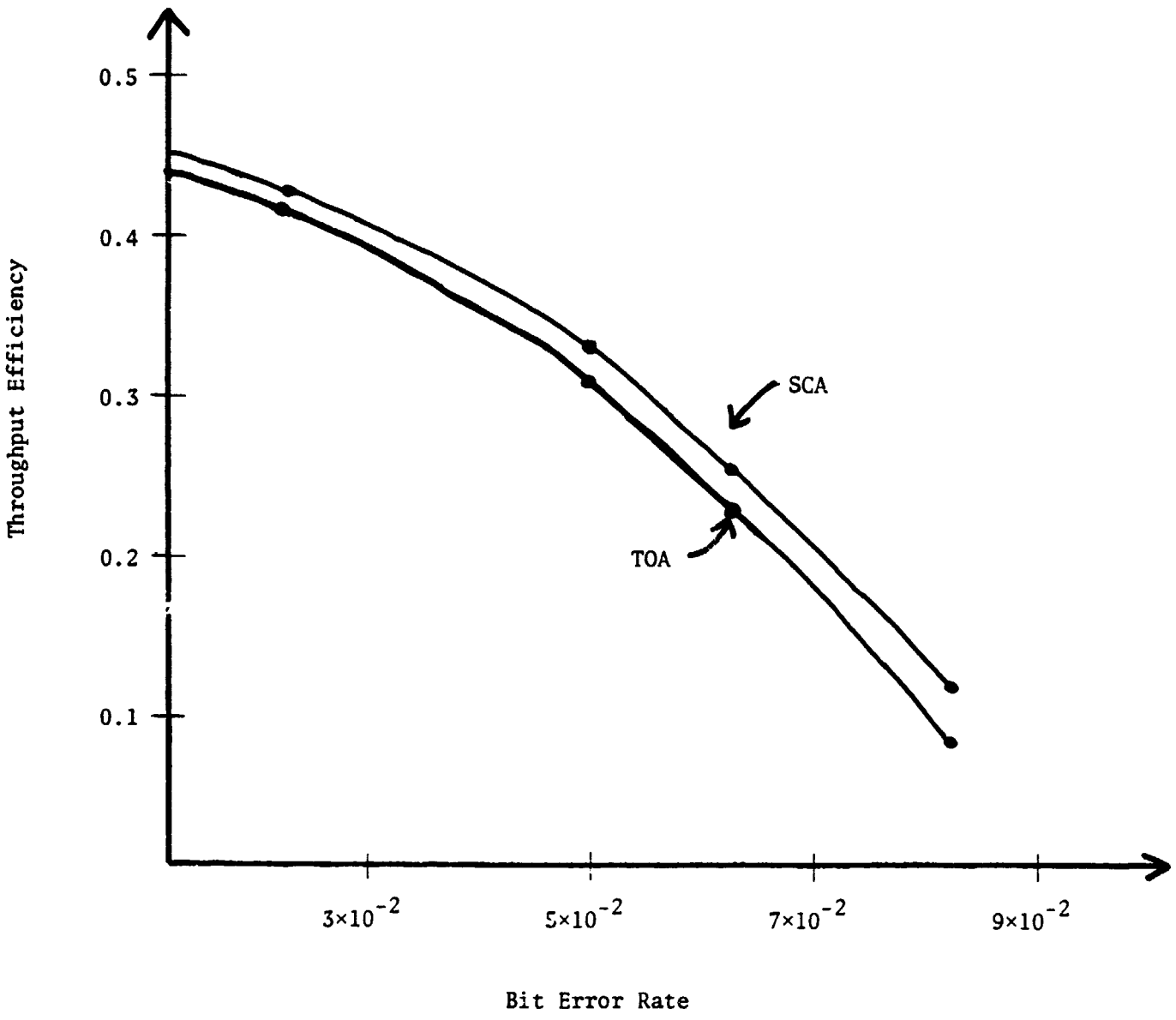

Figure 15 Performance comparison of two type-I hybrid ARQ schemes using sequential decoding.