Embedding a secondary communication channel transparently within a cyclic redundancy check (CRC)

by D. R. Irvin

Techniques are proposed for transparently recapturing part of the capacity of a cyclic redundancy check (CRC) and using the recaptured capacity to carry a secondary communication channel. To do this, a transmitter induces CRC violations in outbound packets of a primary channel according to a set of error templates that are known to both the transmitter and the receiver, and that map one-to-one onto a set of secondary-channel-data words. The receiver corrects the errors by constrained trial and error to deduce which template the transmitter used, and consequently the intended secondary data. Imposing a *P*-bit-per-packet secondary channel is shown to reduce the performance of the primary channel's CRC as though the degree of the generator polynomial had been reduced by *P*, where *P* is an integer. The technique is extended to recapture CRC capacity in fractional-bit increments by permitting only certain codewords to bear the secondary channel. A further extension uses a set of generators mapped one-to-one onto the set of secondary-channel words and known *a priori* to both the transmitter and receiver. A secondary channel is provided by selecting each packet's primary-channel CRC generator in response to a desired secondary-channel word. A variant of this technique improves the performance of the CRC by selecting the generator according to transmission performance or packet length rather than carrying a secondary channel.

Introduction

Several techniques are proposed here for transparently recapturing a small portion of the capacity of a cyclic

0018-8646/01/$5.00 © 2001 IBM

redundancy check (CRC) that is used to protect a data packet from transmission errors, and employing the recaptured capacity to carry a secondary communication channel. To do this, a transmitter purposefully induces CRC violations in outbound packets of a primary communication channel according to a set of error patterns or templates that are known *a priori* to both the transmitter and the receiver, and that correspond to words of data carried by the secondary channel. The receiver corrects these deliberate errors by constrained trial and error, thereby deducing which error pattern the transmitter used and which secondary data the transmitter intended. So, in a sense, a variety of template-matching multiplexing is proposed.

The discussion begins with a conventional explanation of how a CRC works, and goes on to summarize a well-known model for evaluating its performance. Next, the basic technique is presented for embedding a secondary channel in the CRC of a primary channel, a variation of the technique is developed, a table-lookup implementation is described, and the performance model is extended to encompass the foregoing. The model shows that imposing a P-bit-per-packet secondary channel reduces the performance of the primary channel's CRC as though the degree of the generator polynomial had been reduced by P, as would be expected intuitively. In effect, the secondary channel recaptures P bits of capacity from the CRC, where P is an integer.

Attention then turns to extending the basic technique to recapture CRC capacity in fractional-bit increments. The extension works by permitting only certain codewords to bear the secondary channel. Again, the performance model is brought along, and its implications and limitations are discussed.

Finally, two complementary variations of the basic technique are proposed. Neither is based on inducing CRC violations. Rather, the first variation selects a generator packet-by-packet from a pre-established set of possibilities which are mapped one-to-one onto the set of secondary-channel words. Secondary data is carried implicitly by association with the choice of the generator itself. A mechanism is included for resolving ambiguity that arises when a packet is a codeword according to more than one of the generators. In the second of the two variations, which does not bear a secondary channel, the transmitter improves the performance of the CRC by selecting the generator packet-by-packet in response to changes in transmission performance or packet length.

Operation and nomenclature of the cyclic redundancy check

The cyclic redundancy check is based upon the division algorithm for the ring of polynomials with coefficients taken from an algebraic field. For any two polynomials $I(x)$ and $G(x)$, there exist two unique polynomials $Q(x)$, a quotient, and $R(x)$, a remainder, such that

$$I(x) = Q(x)G(x) + R(x),$$

where the degree of $R(x)$ is less than the degree of $G(x)$. In the context of data communications, $I(x)$ is aptly called the "information," and $G(x)$ the "generator."

A data packet to be transmitted, here called $T(x)$, is formed by adding the information $I(x)$ and the remainder $R(x)$:

$$T(x) = I(x) + R(x) = Q(x)G(x) + R(x) + R(x).$$

Again in the context of data communications, the polynomials' coefficients are normally drawn from the field of integers modulo-2. As a result, the algebraic operation "addition" is the same as the logical operation "exclusive-or," and

$$R(x) + R(x) = 0.$$

Hence,

$$T(x) = I(x) + R(x) = Q(x)G(x),$$

which shows that the packet, or more precisely the corresponding polynomial, is a multiple of the generator. Ordinarily, a packet is formed by shifting the information bits up k places to make room for the remainder. This is equivalent to multiplying $I(x)$ by x^k, where k is the degree of the generator polynomial. With the shift, the packet becomes

$$T(x) = I(x)x^k + R(x).$$

Packets constructed in this way are known to be the codewords of a shortened cyclic code [1].

The number of bits needed to convey a polynomial as a word of data is one greater than the degree of the polynomial. Thus, if the degree of $I(x)$ is $N - k - 1$, the degree of the polynomial $T(x)$ that represents the packet is $N - 1$, which corresponds to a packet of length N bits. Of the N bits, the remainder $R(x)$ corresponds to a field of length k bits, here called the "redundancy bits." Thus, the packet is conveyed by $N - k$ information bits followed by k redundancy bits.

To detect the presence of errors introduced by transmission impairment, the receiver evaluates an incoming packet to see whether it is still a multiple of the generator, i.e., a codeword. If the packet is a codeword, the receiver presumes that the information received is the same as the information sent; if the packet is not a codeword, the receiver presumes that the information received was unintentionally altered in transit.

Performance of a CRC

Because a CRC is linear, the sum of two codewords is a codeword. Suppose that the transmitter sends $T(x)$, and

the receiver receives $T(x) + e(x)$, where $e(x)$ represents a pattern of errors imposed upon $T(x)$ by transmission impairment. Because $T(x)$ is a codeword, and because the sum of two codewords is a codeword, the sum $T(x) + e(x)$ is also a codeword whenever $e(x)$ is itself a codeword. This means that the CRC cannot detect transmission errors caused by error patterns that are themselves codewords. Whenever $e(x)$ is a multiple of $G(x)$, a packet contaminated by $e(x)$ passes the CRC check because it is a codeword, although not the intended codeword.

The problem of undetected errors is a central problem in the study of CRCs, and therefore a subject that is well examined in the literature. Boudreau et al., for example, gather and summarize a number of results that concern undetected errors [2], and note the following: A CRC that is based on a primitive degree-k generator polynomial fails to detect the fraction 2^{-k} of all error patterns that have a length (the span from the pattern's first bit-error to its last) greater than $k + 1$. *When all error patterns are equally likely to occur*, the fraction of undetected error patterns is the probability of undetected error. This is now re-derived in preparation for some new results.

Let the transmission process be modeled as a binary symmetric channel (BSC) with a crossover probability (bit-error probability) of $\varepsilon = 0.5$. The BSC provides the receiver with a stream of random bits, which the receiver frames into a stream of packets. The length of the information (i.e., not including the redundancy) is $N - k$ bits. Each possible binary pattern of the information defines a codeword, where the appropriate redundancy bits follow deterministically. Because there are 2^{N-k} patterns of information bits, there are 2^{N-k} codewords. Overall, the packet length is N bits, and these N bits define 2^N patterns which appear at random. A random pattern leads to an undetected error if it happens to be a codeword other than the codeword intended by the transmitter. The probability of occurrence of such an event, and therefore the probability of undetected error, is

$$P_{ue} = (2^{N-k} - 1)/2^N$$
$$= 2^{-k} - 2^{-N}, \quad (1)$$

which for large N becomes the familiar

$$P_{ue} = 2^{-k}.$$

This completes the groundwork.

Secondary channels

A secondary channel carried transparently by a primary channel may be used for a number of purposes, for example

- To support a second logical link that carries traffic from a second end user or application, thereby providing a kind of multiplexing.
- To carry network-management traffic between nodes deep within a network that are unseen by any end user.
- To convey a digital signature that accompanies traffic on a primary channel.
- To supplement the transmission capacity of a primary channel, for example by conveying a control field implicitly rather than explicitly.

This last application may be useful in its own right to shorten a primary-channel packet, as well as providing an alternative to redesigning the packet's format when the bits of a control field have all been assigned yet a need arises for additional control function.

The basic technique

In its most basic form, the secondary channel works by altering the bits of primary-channel packets in accordance with a set of error patterns or templates that are known *a priori* to both the transmitter and the receiver. The transmitter induces deliberate CRC violations by altering bits of an outbound packet, and the receiver determines that bits of an incoming packet have been altered—by the transmitter deliberately or by transmission impairment accidentally—using the CRC.

Rather than discard an incoming packet when the CRC indicates a violation, the receiver perturbs the packet using the commonly known set of error patterns. If a codeword emerges, the receiver presumes that the transmitter intended to send data across the secondary channel, and that in doing so the transmitter employed the particular error pattern that resulted in the emergence of the codeword. If a codeword does not emerge, the receiver discards the incoming packet as flawed in transit.

Operation of the transmitter

To encode a P-bit-per-packet secondary channel, the transmitter draws from $M = 2^P$ pre-established error patterns [$E_i(x)$, $i = 0$ to $M - 1$]. The set of error patterns maps one-to-one onto the set of P-bit words that are to be sent across the secondary channel. For example, to provide a two-bit-per-packet secondary channel, the transmitter has $M = 4$ error patterns in its catalog, consisting of $E_0(x)$, $E_1(x)$, $E_2(x)$, and $E_3(x)$. The pattern $E_0(x)$ identifies the two-bit word 00 to be sent across the secondary channel, $E_1(x)$ identifies the two-bit word 01 to be sent, and so forth.

To construct an outbound packet, the transmitter computes redundancy bits conventionally from the primary-source information, appends the redundancy bits to the primary-source information bits, and then induces a deliberate CRC violation in the outgoing packet by adding

one of the error patterns. With these operations, the outbound packet becomes

$$T(x) = I(x)x^k + R(x) + E_i(x),$$

where $E_i(x)$ corresponds to the desired secondary-channel word.

The receiver has in its catalog the same error patterns mapped one-to-one onto the same set of P-bit secondary-channel words. To determine which error pattern the transmitter used, the receiver adds the various error patterns to the incoming packet and evaluates the CRC of each resulting sum, advancing through the catalog until a positive check is found. In effect, the receiver tries to invert the operation of the transmitter by template matching (constrained trial and error), in order to deduce the transmitter's choice of an error pattern. To do this, the receiver performs the operations

$$\begin{aligned} T(x) + E_j(x) &= [I(x)x^k + R(x) + E_i(x)] + E_j(x) \\ &= [I(x)x^k + R(x)] + [E_i(x) + E_j(x)], \end{aligned}$$

searching over the range $j = 0$ to $j = M - 1$ for j such that

$$E_i(x) + E_j(x) = 0, \tag{2}$$

at which point the CRC gives a positive check. The value of j that satisfies Equation (2) is the value of j the transmitter used in selecting an error pattern, and reveals the corresponding secondary-channel-data word intended by the transmitter.

Choosing the error patterns

With the possible exception of an all-zero error pattern, none of the error patterns $[E_i(x), i = 0 \text{ to } M - 1]$ is allowed to be a codeword. Further, it is important to note that the error patterns $E_i(x)$ are not necessarily the P-bit secondary-channel words themselves. This is because new codewords are introduced by adding the error patterns to the existing codewords. In the extended codeword space, maximizing the minimum Hamming distance between codewords optimizes CRC performance. Error patterns should be devised with this in mind. Moreover, the error patterns should be devised, to the extent possible, so that the extended set of codewords is distributed symmetrically in N-space [3].

When the secondary channel is known *a priori* to be operating, i.e., each packet is known or assumed to carry both primary-channel and secondary-channel data, $E_0(x)$ may advantageously be the all-zero error pattern. Then, when the receiver detects no errors in a packet, the receiver outputs the secondary-channel word that is paired with $E_0(x)$, which most logically may be the all-zero secondary-channel word.

Conversely, when packets that carry secondary-channel data are intermingled with packets that do not carry secondary-channel data, and the receiver must deduce which is which, packet by packet, $E_0(x)$ should not be the all-zero error pattern. Then, when the receiver detects no errors in a packet, the receiver presumes that the secondary channel is inactive rather than sending the all-zero secondary-channel word.

Of the two options, the first—assigning the all-zero pattern to $E_0(x)$—provides the better performance, because it introduces fewer new codewords than its alternative. For this reason, the performance models developed below assume that the secondary channel is always operating. The models are easily adapted, however, to fit the alternative option if desired.

A variation of the basic technique for providing a secondary channel

The transmitter described above alters the outgoing packet by adding the error pattern after the redundancy bits are computed. This requires the receiver to reconstitute the primary-channel information by subtracting the error pattern that was added by the transmitter. As an alternative that does not require the receiver to perform this subtraction, the transmitter may construct the outgoing packet as follows: The error pattern and the primary information are added, and the redundancy bits are computed after the addition. But the primary information is sent *without* the addition of the error pattern. Here, the packet to be transmitted is given by

$$T(x) = I(x)x^k + \rho(x),$$

where $\rho(x)$ is the remainder computed such that

$$I(x)x^k + E_i(x) + \rho(x) = Q'(x)G(x).$$

The receiver inverts the operation of the transmitter in the general way already described, again using constrained trial and error to find a positive CRC check.

To achieve this same result, however, it is not necessary for the transmitter to add the error pattern to the primary information in order to find $\rho(x)$. Rather, $R(x)$—the remainder of the primary information computed conventionally according to the CRC—may be modified as proposed in [4]. Let $\delta_i(x)$ be the remainder found by treating $E_i(x)$ as an information field in its own right; i.e., $\delta_i(x)$ is the remainder that makes $E_i(x)$ a codeword:

$$E_i(x) + \delta_i(x) = Q_i(x)G(x).$$

Then, as a consequence of linearity,

$$\rho(x) = R(x) + \delta_i(x).$$

This relationship suggests a convenient implementation in which the full set $[\delta_i(x), i = 0 \text{ to } M - 1]$ is precomputed and kept in nonvolatile memory at the transmitter. The transmitter constructs an outgoing packet

conventionally (i.e., without regard to the secondary channel), and then modifies the redundancy bits in response to the desired secondary-channel word. To do this, the P-bit secondary-channel word is used as an address to the nonvolatile memory. From the memory location selected, $\delta_i(x)$ is read out and then added to the redundancy bits of the outbound packet.

Performance implications

Unfortunately, the embedded secondary channel does not come at no cost; rather, it comes at the expense of reducing the performance of the primary channel's CRC. Adding $M - 1$ nonzero error patterns to a primary-source codeword gives rise to $M - 1$ new codewords (adding the all-zero error pattern does not produce a new codeword), so there are now $2^{N-k} + (M - 1)2^{N-k}$ codewords. Hence, the probability of an undetected error, P_{ue}, which is just the probability that the at-random BSC produces a packet that is by chance a codeword, but not the intended codeword, is

$$P_{ue} = [2^{N-k} - 1 + (M - 1)2^{N-k}]/2^N$$
$$= M(2^{-k}) - 2^{-N}, \quad (3)$$

which for large N becomes

$$P_{ue} = M(2^{-k}).$$

As before, let $M = 2^P$, where M error patterns (one of which is all-zero) are used to support a P-bit-per-packet secondary channel. With this substitution, Equation (3) may be recast as

$$P_{ue} = 2^{-(k-P)} - 2^{-N}. \quad (4)$$

Comparing Equation (4) with Equation (1) shows that the probability of undetected primary-channel error that results when a CRC supports a P-bit-per-packet secondary channel is the same as the probability of undetected error that would have resulted had the generator polynomial been reduced from degree k to degree $k - P$, and the secondary channel carried explicitly by redeploying the P redundancy bits that were freed. In effect, the secondary channel steals P bits from the capacity of the CRC, where the theft is implicit and transparent except for its performance impact. The difference between an implicit and an explicit secondary channel is, of course, that the implicit channel preserves both the CRC mechanism and the packet format unchanged, whereas the explicit theft and redeployment of redundancy bits does not.

Recapturing the capacity of fractions of redundancy bits

The methods described thus far permit every packet to bear secondary-channel data. This enables a P-bit-per-packet secondary channel to recapture P bits of redundancy, where P is an integer. The technique is now extended to recapture fractional bits of redundancy by restricting the secondary channel to ride upon only selected codewords. Here, a codeword that is used to carry secondary-channel information is called a *bearer codeword*.

Performance implications of fractional-bit recapture

In the extreme case, let the secondary channel ride upon the occurrences of only a single bearer codeword. When the transmitter has secondary-channel data to send, it waits until the primary source provides the bearer codeword. The bearer codeword is then altered by adding one of the $M - 1$ nonzero error patterns. But this time only $M - 1$ new codewords are introduced, rather than $(M - 1)2^{N-k}$ new codewords as before, since only a single codeword bears the secondary channel. Consequently, the probability of undetected error over a BSC with $\varepsilon = 0.5$ and a single bearer codeword is

$$P_{ue} = 2^{-k} + (M - 2)2^{-N}. \quad (5)$$

Comparing fractional-bit recapture with full-bit recapture

It is interesting to examine the decrease in primary-channel CRC performance that accompanies the use of one codeword as a bearer codeword. At what point does the performance loss become a full bit of primary-channel CRC capacity (i.e., the effect of operating the secondary channel is equal to the effect of reducing the generator from degree k to degree $k - 1$)? When one bit of primary-channel CRC capacity is taken away explicitly by reducing the degree of the generator polynomial, the probability of undetected error is given by Equation (1) as

$$P_{ue} = 2^{-(k-1)} - 2^{-N}. \quad (6)$$

Solving Equations (5) and (6) simultaneously gives

$$M - 1 = 2^{N-k}.$$

Thus, the following are equivalent: $N - k$ error patterns applied to one bearer codeword, and one nonzero error pattern applied to $N - k$ bearer codewords. Looked at another way, when the number of error patterns applied to a single bearer codeword is equal to the number of primary-channel information patterns, the number of codewords is doubled. Hence, the primary-channel CRC performance is decreased as though the degree of the generator polynomial had been reduced from degree k to degree $k - 1$. Effectively, the single-bearer-codeword fractional channel operates at its maximum capacity when 2^{N-k} error patterns are used—one full bit is stolen from the performance of the CRC. Conversely, a one-bit-per-packet secondary channel ($M - 1 = 1$) carried by a single bearer codeword

effectively recaptures the fraction $1/2^{N-k}$ of one bit of capacity from the primary channel's CRC.

Although the use of a single bearer codeword recaptures CRC capacity in fractional-bit increments, there are at least three drawbacks: 1) the capacity of the secondary channel is capped at one full bit per packet, 2) the timing of the secondary channel is awkward due to stochastic variation in interarrival times of the bearer codeword, and 3) the codewords are inopportunely clustered without regard to symmetry or minimum Hamming distance.

The situation can be improved by using more than one of the codewords as bearers. Specifically, let A be the number of bearer codewords put at the disposal of the secondary channel, where $0 \le A \le 2^{N-k}$. The performance of the primary-channel CRC is then given by

$$P_{ue} = [2^{N-k} - 1 + (M-1)A]/2^N$$

or

$$P_{ue} = 2^{-k} + (M-1)A(2^{-N}) - 2^{-N}. \quad (7)$$

To express A as a fraction of the total number of primary-source codewords available for use as bearer codewords, let

$$A = \alpha(2^{N-k}),$$

where $0 \le \alpha \le 1$. With this substitution, Equation (7) becomes

$$P_{ue} = 2^{-k} + \alpha(M-1)2^{-k} - 2^{-N}. \quad (8)$$

Equation (8) reduces to Equation (5) for $\alpha = 2^{k-N}$, which is the case of the single bearer codeword, and to Equation (3) for $\alpha = 1$, which is the case in which all of the codewords are employed as bearers. The use of the fraction α in this way serves to illustrate the recapture of fractional-bit capacity from the primary-channel CRC by the secondary channel, where the fraction recaptured increases as the number of bearer codewords increases.

Performance when all patterns are not equally probable

The performance models developed thus far assume that all patterns of unintentional transmission errors are equally likely to occur, or equivalently that the transmission process can be modeled properly as a BSC with $\varepsilon = 0.5$. Four arguments support the use of this model:

1. This is the standard way of evaluating CRC performance, with longstanding acceptance.
2. The model applies to any CRC that has a primitive generator without regard to nuances of a particular generator's behavior.
3. The mathematics is straightforward.
4. The model is accurate and complete in a number of useful situations, for example when applied to fiber-optic transmission that is interrupted by route switching, and when applied to wireless transmission in a deep, extended fade.

Nevertheless, in 1976 Leung-Yan-Cheong and Hellman [3] demonstrated that models of this sort give a limit (as block length N becomes large and as ε approaches 0.5) but not necessarily a bound on P_{ue}. Witzke and Leung [5] and Castagnoli et al. [1] develop this proposition further. It follows, then, that the performance of the techniques proposed here is an open question for smaller values of N and ε. In any case, the question can be answered for a particular generator polynomial and a particular set of error patterns by heroic but straightforward persistence, applying the weight distribution of the code or of its dual, as described by Castagnoli et al.

Conveying a secondary channel by selecting from a set of generators

Another way to support a P-bit-per-packet secondary channel is to provide the transmitter and receiver with a set of $M = 2^P$ different generator polynomials, $[G_i(x), i = 0 \text{ to } M-1]$, which map one-to-one onto the set of secondary-channel words. For each packet, the transmitter computes redundancy bits for the primary-channel data using the generator that is paired with the desired secondary-channel data. A two-bit-per-packet secondary channel, for example, draws from four different generators. When the secondary-channel word is 00, the redundancy bits are computed from the primary information according to $G_0(x)$; when the secondary-channel word is 01, the redundancy bits are computed according to $G_1(x)$, and so on.

In principle, the receiver evaluates an incoming packet according to all M generators and deduces which generator the transmitter used by noting which generator gives the receiver a positive check. In practice, however, things do not work so simply, since an occasional packet may be a codeword in the space of more than one of the generators. When this happens, the receiver's CRC checks positive according to more than one of the generators, and the output of the secondary channel is ambiguous.

To eliminate such ambiguity, the transmitter first constructs a packet according to the generator that corresponds to the desired secondary-channel word, and then—before sending the packet—checks the packet using all of the other generators (or, equivalently, the transmitter computes redundancy bits according to each of the M generators, and looks for duplicates). If the packet is found not to be a codeword in any space except the space according to the intended generator, the packet is sent. Otherwise (i.e., when the packet is a codeword in the

space of more than one generator), the transmitter adds to the packet an error pattern that uniquely identifies the intended generator.

When the receiver determines that an inbound packet does not check according to any of the generators, the receiver adds the various error patterns, again looking through the possibilities by constrained trial and error for the error pattern that was employed by the transmitter and the attendant secondary-channel word. In effect, one of the techniques mentioned above is used sparingly to signal the choice of generator implicitly over a secondary channel, or, more precisely, over a tertiary channel, to resolve occurrences of ambiguity.

More specifically, when a packet is a codeword in the space of more than one generator, the packet is altered before transmission to become

$$T(x) = I(x)x^k + R_i(x) + d_i(x),$$

where $R_i(x)$ is computed conventionally for $I(x)x^k$ according to $G_i(x)$, and $d_i(x)$ is a distinguishing error pattern associated uniquely with $G_i(x)$. The receiver evaluates an incoming packet according to each generator. Because of the addition of $d_i(x)$, however, the packet will not check positively according to any of the generators.

The receiver then evaluates the set of packets $[T(x) + d_j(x)$, for $j = 0$ to $M - 1]$, where, for each choice of j, the packet $T(x) + d_j(x)$ is evaluated according to the generator $G_j(x)$. Unless $T(x)$ has been corrupted by channel errors, $T(x) + d_j(x)$ will check positively when and only when $j = i$, thereby identifying the generator $G_i(x)$ used by the transmitter, and hence identifying the secondary-channel word intended by the transmitter.

Because of the unique associations of $d_j(x)$ with $G_j(x)$ for $j = 0$ to $M - 1$, the complexity of the decoder grows approximately linearly with M rather than according to M^2 or combinatorially. In other words, it is not necessary for the receiver to evaluate $T(x) + d_j(x)$ by any generator except $G_j(x)$, as the receiver knows *a priori* that the transmitter never uses the distinguishing error pattern $d_j(x)$ except with generator $G_j(x)$.

The same performance model that holds for the basic secondary-channel technique holds also for a secondary channel that is conveyed by selecting a generator polynomial as just described. Equation (3) again gives the limit on the probability of undetected error, since the introduction of $M - 1$ additional generators expands the number of codewords by the factor $M - 1$.

Adapting the generator to the characteristics of the channel

Even though every primitive generator polynomial provides a CRC that obeys the limit on P_{ue} mentioned earlier, individual approaches to the limit may differ greatly, with one polynomial providing a significantly lower P_{ue} over a first range of ε or N, and another polynomial providing a significantly lower P_{ue} over a second range. The performance of the CRC may be improved by exploiting these differences, using the multigenerator technique described immediately above to adapt the CRC to short-term changes in transmission performance or block size by changing the generator. The choice of generator may be signaled implicitly from transmitter to receiver using the technique just developed, or signaled explicitly by some other mechanism. Signaling may not be necessary at all if the raw information needed to select the proper generator is available to both the transmitter and the receiver. This would often be the case when the generator is selected according to block size, since the block size would ordinarily be carried in the packet's header.

An apparent shortcoming of this proposition is that the capacity devoted to signaling the choice of generator, whether the signaling is implicit or explicit, might instead be better used to carry additional redundancy bits. This would presume that the performance of a higher-degree generator is necessarily better than the performance of a lower-degree generator for any value of ε or N. The presumption of superiority may be incorrect, however, because a lower-degree generator that is carefully chosen in view of particular values of ε or N may well outperform a higher-degree generator that is chosen as a compromise, for all values of ε and N.

The veracity of this assertion is suggested, although not broadly confirmed, by the large performance differences, dependent upon packet length, among degree-16 generators evident in Figure 2 of Castagnoli et al. [1] for a BSC with crossover probability $\varepsilon = 10^{-5}$. There, the probability of undetected error for packets between 130 and 150 bits in length, given a first degree-16 generator polynomial, is about six orders of magnitude better than the probability of undetected error given a second degree-16 generator, whereas for packets between 150 and 250 bits in length, the performance of the second degree-16 generator exceeds the performance of the first degree-16 generator by about seven orders of magnitude. Given the very large differences in performance among same-degree generators, it is plausible to assume that the performance of degree-k generators and the performance of degree-k' generators, where k' is greater than k, would overlap over various ranges of N *and* ε.

Concluding remarks

A family of template-matching techniques for multiplexing a secondary channel onto a primary channel has been presented. These techniques recapture capacity transparently from a cyclic redundancy check. An attempt

has been made to discuss the techniques in a rigorous way and to quantify aspects of their performance. Nevertheless, Wicker observes that although CRCs are one of the most widely used error-control methods, "they are not so easily understood conceptually" [6]. This would seem to be true, especially regarding undetected errors when a CRC is modified as described here. Fortunately, the literature offers a number of good theoretical and practical papers that should help address such concerns and suggest how to select generators and error patterns for good results [7–11].

Acknowledgment

The author thanks Ali S. Khayrallah for many generous and helpful discussions on the topic of error-detecting codes.

References

1. G. Castagnoli, J. Ganz, and P. Graber, "Optimum Cyclic Redundancy-Check Codes with 16-bit Redundancy," *IEEE Trans. Commun.* **38,** No. 1, 111–114 (January 1990).
2. P. E. Boudreau, W. C. Bergman, and D. R. Irvin, "Performance of a Cyclic-Redundancy Check and Its Interaction with a Data Scrambler," *IBM J. Res. & Dev.* **38,** No. 6, 651–658 (November 1994).
3. S. K. Leung-Yan-Cheong and M. E. Hellman, "Concerning a Bound on Undetected Error Probability," *IEEE Trans. Info. Theory* **IT-22,** 235–237 (March 1976).
4. D. R. Irvin, "Preserving the Integrity of Cyclic-Redundancy Checks When Protected Text Is Intentionally Altered," *IBM J. Res. & Dev.* **33,** No. 6, 618–626 (November 1989).
5. K. A. Witzke and C. Leung, "A Comparison of Some Error-Detecting CRC Code Standards," *IEEE Trans. Commun.* **COM-33,** No. 9, 996–998 (September 1985).
6. S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, Upper Saddle River, NJ, 1995, p. 121.
7. S. K. Leung-Yan-Cheong, E. R. Barnes, and D. U. Friedman, "On Some Properties of the Undetected Error Probability of Linear Codes," *IEEE Trans. Info. Theory* **IT-25,** No. 1, 110–112 (January 1979).
8. V. K. Agarwal and A. Ivanov, "Computing the Probability of Undetected Error for Shortened Cyclic Codes," *IEEE Trans. Commun.* **40,** No. 3, 494–499 (March 1992).
9. G. Castagnoli, S. Bräuer, and M. Herrmann, "Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits," *IEEE Trans. Commun.* **41,** No. 6, 883–892 (June 1993).
10. B. Wong and C. Leung, "On Computing Undetected Error Probabilities on the Gilbert Channel," *IEEE Trans. Commun.* **43,** No. 11, 2657–2661 (November 1995).
11. G. Funk, "Determination of Best Shortened Linear Codes," *IEEE Trans. Commun.* **44,** No. 1, 1–6 (January 1996).

Received November 1, 2000; accepted for publication September 17, 2001

David R. Irvin *IBM Corporation, 3039 Cornwallis Road, P.O. Box 12195 (T81/B503), Research Triangle Park, North Carolina 27709 (dirvin@us.ibm.com).* Mr. Irvin joined IBM in 1974, and worked in the IBM laboratory at Research Triangle Park, North Carolina, from 1974 until 1994 as an engineer–scientist in the field of data communications. He then taught for an academic year (1994–1995) in the North Carolina State University Department of Electrical and Computer Engineering. From 1995 through 2000, he was Senior Consulting Engineer and manager of invention-development activities at the Ericsson, Inc. laboratory, Research Triangle Park. Mr. Irvin, who is also a registered United States Patent Agent, rejoined IBM in early 2001 and now works in the Intellectual Property Law Department of the IBM Software Group, where he does patent preparation and prosecution on behalf of IBM Global Services clients. Mr. Irvin received the B.E.S. from Johns Hopkins University (1970), the M.E.E. from North Carolina State University (1971), and the P.D.E. from the University of Wisconsin at Madison (1990). He is a member of Phi Beta Kappa and Tau Beta Pi, a full member of Sigma Xi, and a Senior Member of IEEE; he currently holds 25 U.S. patents, four of which pertain to cyclic redundancy checks.