Pakistan Journal of Applied Sciences 2(5): 574-578, 2002


# Error Control Protocols Used in File Transfer Environment

Muhammad Sher and Khalid Rashid
Department of Computer Science, International Islamic University, Islamabad

**Abstract:** File transfer is the movement of digital information from one data terminal to another. This digital information is organized into characters, frames, and files. File transfer applications typically demand a very high level of reliability; unlike voice application, a single erroneous bit can render a multi megabyte file useless to the end user. Communication links in wireless and wired file transfer systems are designed to be highly reliable. The error control systems for these applications use error detection coupled with retransmission requests to maximize reliability at some cost to throughput. At there simplest, such error control systems use parity-check bits or CRC codes to trigger retransmission requests. Since the retransmission requests occur well below the application layer, and such protocols are called *automatic repeat request* (ARQ) protocols. More complicated protocol includes elements of FEC and packet combining to reduce the frequency of retransmission requests. In this article Parity, CRC, and pure ARQ protocols are discussed. Several popular file transfer protocols are provided as examples. Consideration is then given to hybrid protocols and packet combining.

**Keywords:** ARQ Protocols, CRC, FCS, IP, TCP, UDP

## Parity Bits and CRC Protocols

In asynchronous file transfer systems, each character is treated as a separate entity. Seven bit ASCII is the most famous example. Standard error control technique for asynchronous ASCII is the simplest possible: a single parity bit is appended to each ASCII character. The value of the appended bit is selected so that the sum of all eight transmitted bits is 0 modulo 2 (even parity) or 1 modulo 2 (odd parity). In either case, the resulting code can detect all single error as well as a large number of error patterns of higher weight. The encoding and decoding circuit is extremely simple, and can be constructed using a small number of exclusive-OR gates or a few lines of code. The Kermit data modem protocol is a typical example of an asynchronous file transfer system that uses simple parity checks.

In synchronous transmission systems, error control can be expanded to cover entire frames of data. Since the frame is a larger transmission element than the character, the number of encoding and decoding operations is reduced, along with the number of potential requests for retransmission. The most popular synchronous error control techniques are based on shortened linear cyclic codes. and C be an (n, k) cyclic code with a systematic encoder. and S be the subset of code words in C whose j high-order information coordinates (i.e. the j rightmost coordinates in the codeword) have the value zero. Let C´ be the set of words obtained by deleting the j rightmost coordinates from all of the words in S. C´ is an ( $n - j, k - j$) shortened systematic code that is usually referred to as a CRC, or Cyclic Redundancy Check code. CRC codes are also called polynomial codes.

CRC is the most common and the most powerful error-detecting code. It can be described as follows: Given a K bit block of bits, or message, the transmitter generates an n-bit sequence, known as a Frame Check Sequence (FCS), so that the resulting frame, consisting of k + n bits, is exactly divisible by some predetermined number. The receiver then divides the incoming frame by that number and if there is no remainder, assumes there was no error.

$$\frac{2^n M}{P} = Q + \frac{R}{P} \quad (1)$$

To clarify this, we present the procedure in three ways: modulo 2 arithmetic, polynomials, and digital logic.

**Modulo 2 Arithmetic:** Modulo 2 arithmetic uses binary addition with no carries, which is just the exclusive operation. Now define:

T = (k + n)-bit frame to be transmitted, with n < k.
M = k-bit message, the first k bits of T
F = n-bit FCS, the last n bits of T
P = pattern of n +1 bits; this is the predetermined divisor

We would like T / P to have no remainder. It should be clear that

$T = 2^nM + F$ (2)

That is, by multiplying M by $2^n$, we have, in fact, shifted it to the left by n bits and padded out the result with zeroes. Adding F yields the concatenation of M and F, which is T. We want T to be exactly divisible by P. Suppose that we divided $2^nM$ by P:

$$\frac{2^n M}{P} = Q + \frac{R}{P} \quad (3)$$

There is a quotient and a remainder. Because division is binary, the remainder is always at least one bit less than the divisor. We will use this remainder as our FCS. Then

$T = 2^nM + R$ (4)

There is a quotient and a remainder. Because division is binary, the remainder is always at least one bit less than the divisor. We will use this remainder as our FCS. Then

$T = 2^nM + R$ (5)

There is no remainder, and, therefore, T is exactly divisible by P. Thus, the FCS is easily generated: Simply divide $2^nM$ by P and use the remainder as the FCS. On reception, the receiver will divide T by P and will get no remainder if there have been no errors.

The pattern P is chosen to be one bit longer than the desired FCS, and the exact bit pattern chosen depends on the type of errors expected. At minimum, both the high and low order bits of P must be 1.

The occurrence of an error is easily expressed. An error results in the reversal of a bit. This is equivalent to taking the exclusive-OR of the bit and 1 (modulo 2 addition of 1 to the bit): 0 + 1 = 1; 1 + 1 = 0. Thus, the errors in an (n + k)-bit frame can be represented by an (n + k)-bit field with is in each error position. The resulting frame $T_r$ can be expressed as

$T_r = T + E$, Where (6)

T = transmitted frame

E = error pattern positions where errors occur

T = received frame

The receiver will fail to detect an error if and only if $T_r$ is divisible by P, which is equivalent to E divisible by P. Intuitively, this seems an unlikely occurrence.

**Polynomials:** A second way of viewing the CRC process is to express all values as polynomials in a dummy variable X, with binary coefficients. The coefficients correspond to the bits in the binary number. Thus, for M = 110111, we have $M(X) = X^5 + X^4 + X^2 + X + 1$, and, for P = 11001, we have $P(X) = X^4 + X^3 + 1$. Arithmetic operations are again modulo 2. The CRC process can now be described as

$$\frac{X^n M(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)} \quad (7)$$

$$T(X) = X^n M(X) + R(X) \quad (8)$$

An error E(X) will only be undetectable if it is divisible by P(X). It can be shown that all of the following errors are not divisible by a suitably chosen P(X) and, hence, are detectable:

- All single bit errors.
- All double-bit errors, as long as P(X) has at least three 1s.
- Any odd number of errors, as long as P(X) contains a factor (X + 1).
- Any burst error for which the length of the burst is less than the length of the divisor polynomial; that is, less than or equal to the length of the FCS.
- Most largely burst errors.

In addition, it can be shown that if all error patterns are considered equally likely, then for a burst error of length r +1, the probability that E(X) is divisible by P(X) is $1 / 2^{r-1}$, and for a longer burst, the probability is $1/2^r$, where r is the length of the FCS.

Three versions of P(X) are widely used:

$CRC\text{-}16 = X^{16} + X^{15} + X^2 + 1$ (9)

$CRC\text{-}CCITT = X^{16} + X^{12} + X^5 + 1$ (10)

$CRC\text{-}32 = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1$ (11)

**Digital Logic:** The CRC process can be represented by, and indeed implemented as, a dividing circuit consisting of exclusive-OR gates and a shift register. The shift register is a string of 1-bit storage devices. Each device has an output line, that indicates the value currently stored, and an input line. At discrete time instants, known as clock times, the value in the storage device is replaced by the value indicated by its input line. The entire register is clocked simultaneously, causing a 1-bit shift along the entire register.

The circuit is implemented as follows:

- The register contains n bits, equal to the length of the FCS.
- There are up to n exclusive-OR gates.
- The presence or absence of a gate corresponds to the presence or absence of a term in the divisor polynomial. P(X).

The architecture of this circuit is best explained by first considering an example, which is illustrated in Fig. 1. In this example, we use

Message M = 1010001101;

$M(X) = X^9 + X^7 + X^3 + X^2 + 1$

Divisor P = 110101; $P(X) = X^5 + X^4 + X^2 + 1$

Fig. 1 shows the shift register implementation. The process begins with the shift register cleared (all zeros). The messages, or dividend, is then entered, one bit at a time, starting with the most significant bit. Table 1 shows the step by step operation, as the input is applied one bit at a time. Each row of the Table shows the values currently stored in the five shift-register elements. In addition, the row shows the values that appear at the outputs of the three exclusive-OR circuits. Finally, the row shows the value of the next input bit, which is available for the operation of the next step.

Because no feed back occurs until a 1-divided bit arrives at the most significant end of the register (C4), a 1 is subtracted (exclusive-OR) from the second (C3), fourth (C1), and sixth (input) bits on the next shift. This is identical to the binary long-division process. The process continues through all the bits of the message plus five zero bits. These letters bits account for shifting M to the left five positions to accommodate the FCS. After the last bit is processed, the shift register contains the remainder (FCS), which can then be transmitted.

At the receiver, the same logic is used. As each bit of M arrives, it is inserted into the shift register. If there have been no errors, the shift register should contain the bit pattern for R at the conclusion of M. The transmitted bits of R now begins to arrive, and the effect is to zero-out the register, At the conclusion of reception, the register contains all 0s.

Fig. 2 indicates the general architecture of the shift register implementation of a CRC for the polynomial

$$P(X) = \sum_{i=0}^{n} a_i X^i,$$

Where $a_0 = a_n = 1$ and all other $a_i$ equal either 0 or 1

The error-detecting capability of a CRC code is best measured in terms of the percentage of possible error patterns that are detectable. Channel error tend to occur in bursts in the wired file-transfer applications that make the most use of CRC codes, so the memory less channel assumption that underlies a Hamming distance analysis is not valid. An r-bit CRC can detect $100*(1- 2^{-r})\%$ of all error patterns.

CRC codes are often altered in practice to avoid the possibility of confusing the check bits with protocol flag fields (frame delimiters, for example) or to maintain a minimum transition density (an important consideration in magnetic recording applications). The modification to the code typically involves

complementing some or all of the check bits. Though the code is no longer linear. Error-detection performance is unaffected. In some Applications it is easier to generate an r-bit check-sum by simply segmenting the information to be protected into r-bit words and computing the sum (without carry). As with the unmodified CRC, the resulting code
is linear and can detect 100. $(1 - 2^{-r})$% of all error patterns.

**Retransmission Request Modes:** In this section we Derive the basic relations and design choices that determine the throughput performance of an ARQ protocol. We begin by determining the number of times That a given packet can expect to be transmitted before it is accepted by the receiver. Given the probability $P_r$ that an error is detected and a retransmission request is generated, the expected number of transmission $T_r$ is computed as follows:

$$T_r = (1-P_r) + 2P_r(1-P_r) + 3P_r^2(1-P_r) + \ldots\ldots + kP_r^{k-1}(1-P_r) + \ldots\ldots$$

$$T_r = (1-P_r)\sum_{k=1}^{\infty} kP_r^{k-1} = \frac{1}{1-P_r}$$

There are three basic modes that are used in the implementation of retransmission-request protocol: Stop-and-Wait (SW-ARQ), Go-Back-N (GBN-ARQ), and Selective-Repeat (SR-ARQ). These three modes provide varying levels of throughput by changing the amount of buffering required in the transmitter and/or receiver. Throughput (η) is defined here to be the average number of encoded data blocks (packets) accepted by the receiver in the time it takes the transmitter to send a single k-bit data packet.

SW-ARQ is by far simplest of the three modes. In this mode, the transmitter sends a single packet and waits for an acknowledgment (ACK). If an ACK is received, the transmitter sends the next packet. Otherwise, the first packet is retransmitted. Since the transmitter is idle while waiting for the response, this mode of ARQ is often called "Idle RQ". Let $\Gamma$ be the number of bits that the transmitter could have transmitted during this idle time. An SW-ARQ
protocol based on a rate $R = k/n$ error-detecting code will provide the following throughput:

$$\eta sw = \frac{k}{T_r(n+\Gamma)} = R\left(\frac{1-P_r}{1+\Gamma/n}\right)$$

SW-ARQ protocols require very little buffering in the transmitter, and virtually none in the receiver. For this reason such protocols are often used in simple data modems (the "Kermit" protocol is a typical). The disadvantage, of course, lies in the inefficient use of the channel, an inefficiency that increases with the distance between transmitter and receiver.

GBN-ARQ protocols assume that the transmitter is capable of buffering N packets. When the transmitter is informed by the received of an error in some packet, say packet j, the transmitter "goes back" to that packet and resumes transmission from there. The parameter N is the number of packets that have been subsequently sent by the transmitter since the jth packet, and that must now be retransmitted. N is a function of the roundtrip propagation and processing delays for the transmitter and receiver and can be approximated by $[\Gamma/n]$. The throughput for a GBN-ARQ protocol based on a rate R = k/n error-detecting code is as follows:

$$\eta GNB = \left(\frac{k}{n}\right)\frac{1}{1+(\Gamma_r - 1)N} = R\left(\frac{1-P_r}{1+P_r(N-1)}\right)$$

Note that the transmitter in a GBN-ARQ protocol is continuously transmitting. GBN-ARQ protocols are thus often called "Continuous RQ" protocols.

SR-ARQ protocols are also Continuous RQ protocols. In this case the transmitter only retransmits the specific packets for which an ACK is not received. This requires significant buffering in both the transmitter and the receiver, but maximizes channel efficiency. The throughput in this case is no longer a function of channel delay. The throughput for a rate $R = k/n$ code is quickly computed as

$$\eta SR = \left(\frac{k}{n}\right)\left(\frac{1}{\Gamma_r}\right) = R(1 - P)_r$$

In the next section we consider the means by which error detection retransmission requests are incorporated into two popular file-transfer applications.

**The Internet and TCP/IP:** The key technology underlying the Internet (and many internets, intranets, and variations thereof) is the suite of protocols that go by the name TCP/IP. TCP/IP is named after two constituent protocols, the Transmission Control Protocol and the Internet Protocol, but the suite actually contains several dozen protocols. TCP/IP allows for the internetworking of computer networks that are based on different networking technologies. For example, TCP/IP allows for the seamless interconnection of token rings (IEEE 802.5) and Ethernet-based systems (IEEE 802.3)

A TCP/IP internetworking architecture can be divided into four basic layers.

- An *Application Layer* consisting of user processes (TEL-NET, FTP, SMTP, etc,).
- A *Transport Layer* that provides end-to-end data transfer.
- An *Internet work Layer* (also called the Internet layer) that provides the image of a single virtual network to upper layers by routing datagram's among the interconnected networks.
- A *Network Interface Layer* (also called the link or data-link layer) that provides the actual interface to the hardware of the individual networks (e.g.X.25, ATM, FDDI, and packet radio networks).

The Internet Protocol (IP) is the heart of the Internet work layer. It is a connectionless protocol that does not assume or provide any measure of link reliability. The only error control incorporated into the IP datagram is a 16-bit checksum that only covers the header. The checksum is obtained by computing the 16-bit 1's complement of the 1's complement sum of all 16-bit words in the header (the checksum itself is assumed to be zero during this computation). If a received IP datagram has an incorrect checksum, the entire datagram is discarded. There is no provision for a transmission request, in part because it would not be

Table 1: Result of Circuit with Shift Registers for Dividing by the Polynomial $X^5 + X^4 + X^2 + 1$

| | $C_4$ | $C_3$ | $C_2$ | $C_1$ | $C_0$ | $C_4 + C_3$ | $C_4 + C_1$ | $C_4$ + input | input |
|---|---|---|---|---|---|---|---|---|---|
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Step 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Step 2 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| Step 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Step 4 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| Step 5 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| Step 6 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Step 7 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| Step 8 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| Step 9 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Step10 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| Step11 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| Step12 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| Step13 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| Step14 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| Step15 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | |

Message to be sent
Five zero added

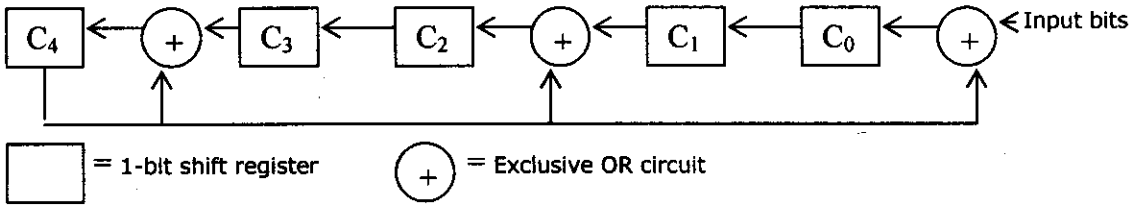


Fig. 1: Shift-register Implementation.

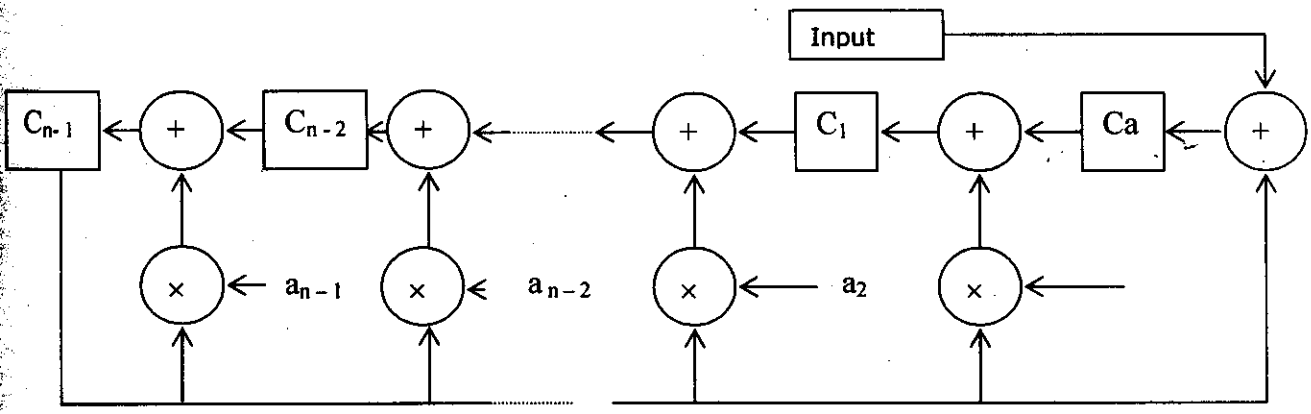


Fig. 2: General CRC Architecture to Implement Devisor $1 + a_1x + a_2x^2 + \ldots + a_{n-1}X^{n-1} + X^n$

known from whom to request the retransmission. It should also be noted that IP does not provide a check on the data field within the datagram. This must be provided at the Transport layer.

There are two basic transport-layer protocols: the Transmission Control Protocol (TCP-connection oriented) and the User Datagram Protocol (UDP-connectionless). Both protocols "encapsulate" IP datagram's by appending transport-layer headers. UDP has an optional 26-bit checksum that, when used, is calculated in a manner similar to the IP datagram checksum. The UDP checksum is a 26-bit 1's complement of the 1's complement sum of the encapsulated IP header, the UDP header, and the UDP data. As with IP, UDP does not allow for acknowledgments. UDP simply arranges received packets in the proper order and passes them on to the application. Internet telephony is typical of the applications that use UDP in that it emphasizes latency over reliability.

TCP, on the other hand, makes full use of its 16-bit checksum. TCP assigns sequence number to each transmitted data block, and expects a positive acknowledgment from the transport layer of the receiving entity. If an acknowledgment is not received in a timely manner, the block is retransmitted. The full retransmission protocol is a variation of SR-ARQ that uses a window to control the number of pending acknowledgments. The transmitter is allowed to transmit all packets within the window, starting a retransmission clock for each. The window is moved to account for each received acknowledgment.

**Hybrid-ARQ Protocols and Packet Combining:** In 1960, Wozencraft and Horstein described and analyzed a system that allowed for both error correction and error detection with retransmission requests. Their system, now known as a type-1 Hybrid-ARQ protocol, provides significantly improved performance over "pure" ARQ protocols. The goal in designing such systems is used FEC to handle the most frequently occurring error patterns. The less frequently occurring (those most likely to cause FEC decoder errors) are handled through error detection and the request of a retransmission.

Error-control protocols based on algebraic block codes (e.g., RS and BCH codes) and hard-decision decoding provide a natural form of type-1 Hybrid-ARQ protocol. The most commonly used hard-decision decoders for BCH and RS codes use bounded distance decoding algorithms that correct a number of error up to some design distance $t$. If there is no codeword within distance $t$ of the received word, the bounded distance-decoding algorithm fails, providing an indication of uncorrectable errors. When decoder failures are used as a trigger for a retransmission request, the BCH and RS-based systems become example of type-1 Hybrid-ARQ protocols. The Reflex and Cellular Digital Packet Data (CDPD) wireless data protocols are typical of the applications that use these strategies.

Type-1 Hybrid-ARQ protocols can also be based on complete decoders. Drukarev and Costello developed the "Time-Out" and "Slope Control" algorithms for type-1 protocols based on Convolution codes with sequential decoding. Yamamoto at all developed a similar protocol based on Viterbi decoding.

Pure ARQ and Type-1 Hybrid-ARQ protocols generally discard received packets that trigger retransmission requests. In 1977, Sindhu discussed a scheme that made use of these packets. Sindhu's idea was that such packets can be stored and later combined with additional copies of the packet, creating a single packet that is more reliable than any of its constituent packets. Since 1977 there have been innumerable systems proposed that involve some form of packet combining. These packet-combining systems can be loosely arranged into two categories: code-combining systems and diversity-combining systems.

In code-combining systems, the packets are concatenated to form noise-corrupted code works from increasingly longer and lower rate codes. Code combining was first discussed in a 1985 paper by Chase, who coined the term. Subsequent code-combining systems are exemplified by the work of Krishna, Morgera, and Odoul. An early version of a code-combining system was the type-II Hybrid-ARQ protocol invented by Lin and Yu. The type-II system allows for the combination of two packets, and is thus a truncated code-combining system. The system devised by Lin and Yu was developed for satellite channels. Type-II systems based on RS and RM codes were later developed by Pursley and Sandberg and by Wicker and Bartz, and similar systems, which utilize RCPC code, were introduced by Hagenauer.

In diversity-combining systems, individual symbols from multiple, identical copies of a packet are combined to create a single packet with more reliable constituent symbols. Diversity-combining systems are generally suboptimal with respect to code-combining systems, but are simpler to implement. These schemes are represented by the work of Sindhu, Benelli, Metzner and Harvey and Wicker.

## References

Chin-Cheng Shih, C.R. Wulff, C.R.P. Hartmann, C.K. Mohan, 1998. "Efficient Heuristic Search Agorithms for Soft-Deision Decoding of Linear Block Codes" IEEE Trans. Information Theory, IETTAW-44: 7.

D.J. Costello, J. Hagernauer, Hideki Imai, Stephen B. Wicher, 1998. "Application of Error-Control Coding" IEEE Trans. Information Theory, 44: 6.

J.L. Massey, "Deep-space Communication and Coding: A Marriage Made In Heaven," In Lecture Notes on Control and Information Sciences 82, J.

JM Wozencraft and B. Reiffen, 1961. Sequential Decoding, Cambridge MA: MIT Press.

RR Green, 1966. "A Serial Orthogonal Decoder," in Jet Propulsion Laboratory Space Programmes Summery, 37-38: 247-251.

RM Fano, 1963. "A Heuristic Discussion of Probabilistic Decoding" IEEE Trans, Inform Theory, 9: Pp 64-74.

S. Lin and H Lyne, 1967. "Some Results on Binary Convolutional Codes Generators," IEEE Trans, Inform Theory, IT 13, pp. 134-139.