ERROR CONTROL AND DETECTION

Error Control:

When data is transmitted over a cable or channel, there is always a chance that some of the bits will be changed (corrupted) due to noise, signal distortion or attenuation. If errors do occur, then some of the bits will either change from 0 to 1 or from 1 to 0.

Error Control allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. Error control is divided in two main categories:
Error Detection It allows a receiver to check whether received data has been corrupted during transmission. It can, for example, request a retransmission.

Error Correction This type of error control allows a receiver to reconstruct the original information when it has been corrupted during transmission.

In the data link layer, the term error control refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

There are 2 ways to correct found errors:

- Forward error correction (FEC is accomplished by adding redundancy to the transmitted information using a predetermined algorithm. Each redundant bit is invariably a complex function of many original information bits. The original information may or may not appear in the encoded output; codes that include the unmodified input in the output are systematic, while those that do not are nonsystematic.) and
- Automatic repeat request (ARQ) (in which the receiver detects transmission errors in a message and automatically requests a retransmission from the transmitter. Usually, when the transmitter receives the ARQ, the transmitter retransmits the message until it is either correctly received or the error persists beyond a predetermined number of retransmissions.A few types of ARQ protocols are Stop-and-wait ARQ, Go-Back-N ARQ and Selective Repeat ARQ.).

Hamming distance:

One of the central concepts in coding for error control is the idea of the Hamming distance. The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words x and y as d(x, y). The Hamming distance can easily be found if we apply the XOR operation on the two words and count the number of 1s in the result. Note that the Hamming distance is a value greater than zero.

Let us find the Hamming distance between two pairs of words.
1. The Hamming distance d(OOO, 011) is 2 because 000 XOR 011 is 011 (two 1s).
2. The Hamming distance d(10101, 11110) is 3 because 10101 XOR 11110 is 01011 (three 1s).

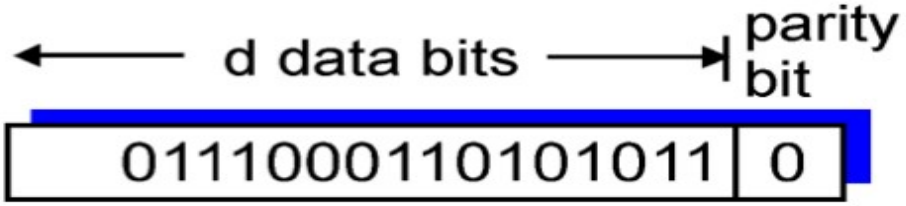
Error Detection:

There are three ways to detect errors.

1. Parity check
2. CRC
3. Checksum

1. Parity Check:

The simplest error-detection scheme is to append a parity bit to the end of a block of data . A typical example is ASCII transmission, in which a parity bit is attached to each 7-bit ASCII character. The value of this bit is selected so that the character has an even number of 1s (even parity) or an odd number of 1s (odd parity).

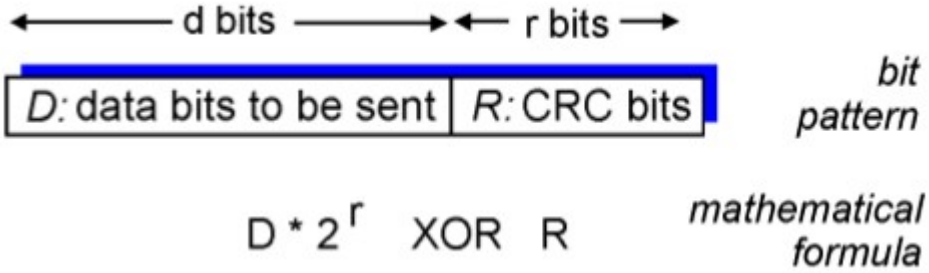


One bit Even Parity

So, for example, if the transmitter is transmitting an ASCII G (1110001) and using odd parity, it will append a 1 and transmit 11100011. The receiver examines the received character and, if the total number of 1s is odd, assumes that no error has occurred. If one bit (or any odd number of bits) is erroneously inverted during transmission (for example, 11QO0011), then the receiver will detect an error. Note, however, that if two (or any even number) of bits are inverted due to error, an undetected error occurs. Typically, even parity is used for synchronous transmission and odd parity for asynchronous transmission. The use of the parity bit is not foolproof, as noise impulses are often long enough to destroy more than one bit, particularly at high data rates.

2. Cyclic Redundancy Check:

CRC codes operate as follows. Consider the d-bit piece of data, D, that the sending node wants to send to the receiving node. The sender and receiver must first agree on a r+1 bit pattern, known as a generator, which we will denote as G. We will require that the high and low order bits of G must be 1 (e.g., 10111 is acceptable, but 0101 and 10110 are not) . The key idea behind CRC codes is shown in Figure. For a given piece of data, D, the sender will choose r additional bits, R, and append them to D such that the resulting d+r bit pattern (interpreted as a binary number) is exactly divisible by G using modulo 2 arithmetic. The process of error checking with CRC's is thus simple: the receiver divides the d+r received bits by G. If the remainder is non-zero, the receiver knows that an error has occurred; otherwise the data is accepted as being correct.



All CRC calculations are done in modulo 2 arithmetic without carries in addition or borrows in subtraction. This means that addition and subtraction are identical, and both are equivalent to the bitwise exclusive-or (XOR) of the operands. Thus, for example,

1011 XOR 0101 = 1110
1001 XOR 1101 = 0100

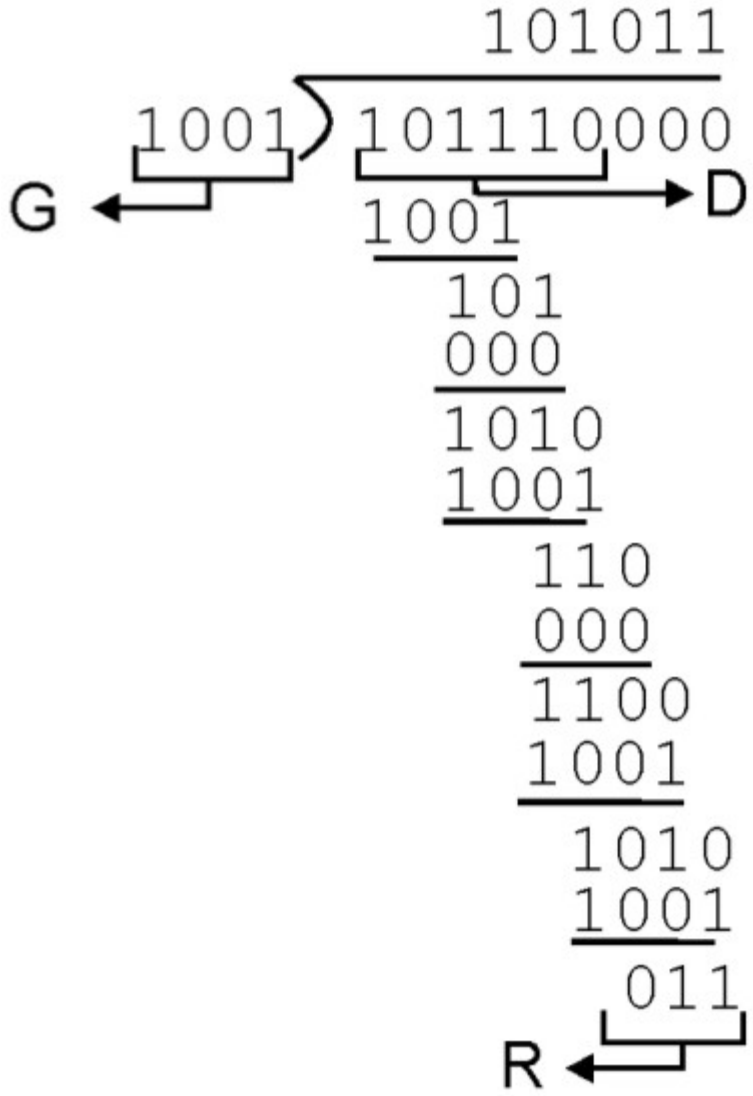
Also, we similarly have

1011 - 0101 = 1110
1001 - 1101 = 0100

Multiplication and division are the same as in base 2 arithmetic, except that any required addition or subtraction is done without carries or borrows. As in regular binary arithmetic, multiplication by 2^k left shifts a bit pattern by k places. Thus, given D and R, the quantity $D*2^r$ XOR R yields the d+r bit pattern shown in Figure above.

International standards have been defined for 8-, 12-, 16- and 32-bit generators, G. An 8-bit CRC is used to protect the 5-byte header in ATM cells.

Figure below illustrates this calculation for the case of D = 101110, d = 6 and G = 1001, r=3. The nine bits transmitted in this case are 101110 011. You should check these calculations for yourself and also check that indeed $D2^r$ = 101011 * G XOR R.



A second way to viewing the CRC process is to express all values as polynomials in a dummy variable X, with binary coefficients. The coefficients corresponds to the bits in the binary number. Thus for D=110011 we have, $D(X)=X^5+X^4+X+1$ for P=11001 we have $P(X)=X^4+X^3+1$. Arithmetic operations are again modulo 2.

3. Checksum:

The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking.

Example1:
Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12,0,6,36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

Example2:
We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12,0,6, -36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error

Internet Checksum
Traditionally, the Internet has been using a 16-bit checksum. The sender calculates the checksum by following these steps.
Sender site:
1. The message is divided into 16-bit words.
2. The value of the checksum word is set to O.
3. All words including the checksum are added ushtg one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

The receiver uses the following steps for error detection.

Receiver site:
1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3.The sum is complemented and becomes the new checksum.
4.If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

Source : http://dayaramb.files.wordpress.com/2011/03/computer-network-notes-pu.pdf