Communication and Networking
Error Detection Basics

D. Richard Brown III

(selected figures from Stallings Data and Computer Communications 10th edition)

Probability of Bit and Frame Errors

Probability of bit error: $0 \leq P_b \leq 1$.

Suppose we have F bits in a frame (message). Denote

$P_1 =$ probability frame arrives with no bit errors

$P_2 =$ probability frame arrives with one or more undetected errors

$P_3 =$ probability frame arrives with one or more detected errors and no undetected errors

Suppose we have no error detection so that $P_3 = 0$. Then we can write

$$P_1 = (1 - P_b)^F$$
$$P_2 = 1 - (1 - P_b)^F$$

Example: $P_b = 10^{-5}$, $F = 2048$. In this case, we have

$$P_2 = 1 - (1 - 10^{-5})^{2048} \approx 0.02.$$

Approximately 2% of the frames are received in error (98% of the frames are received without error).

Error Detection

The receiver needs a way to detect errors in frames in order to discard bad frames and request a replacement.

Three common methods:

1. Parity Check
2. Checksum
3. Cyclic Redundancy Check

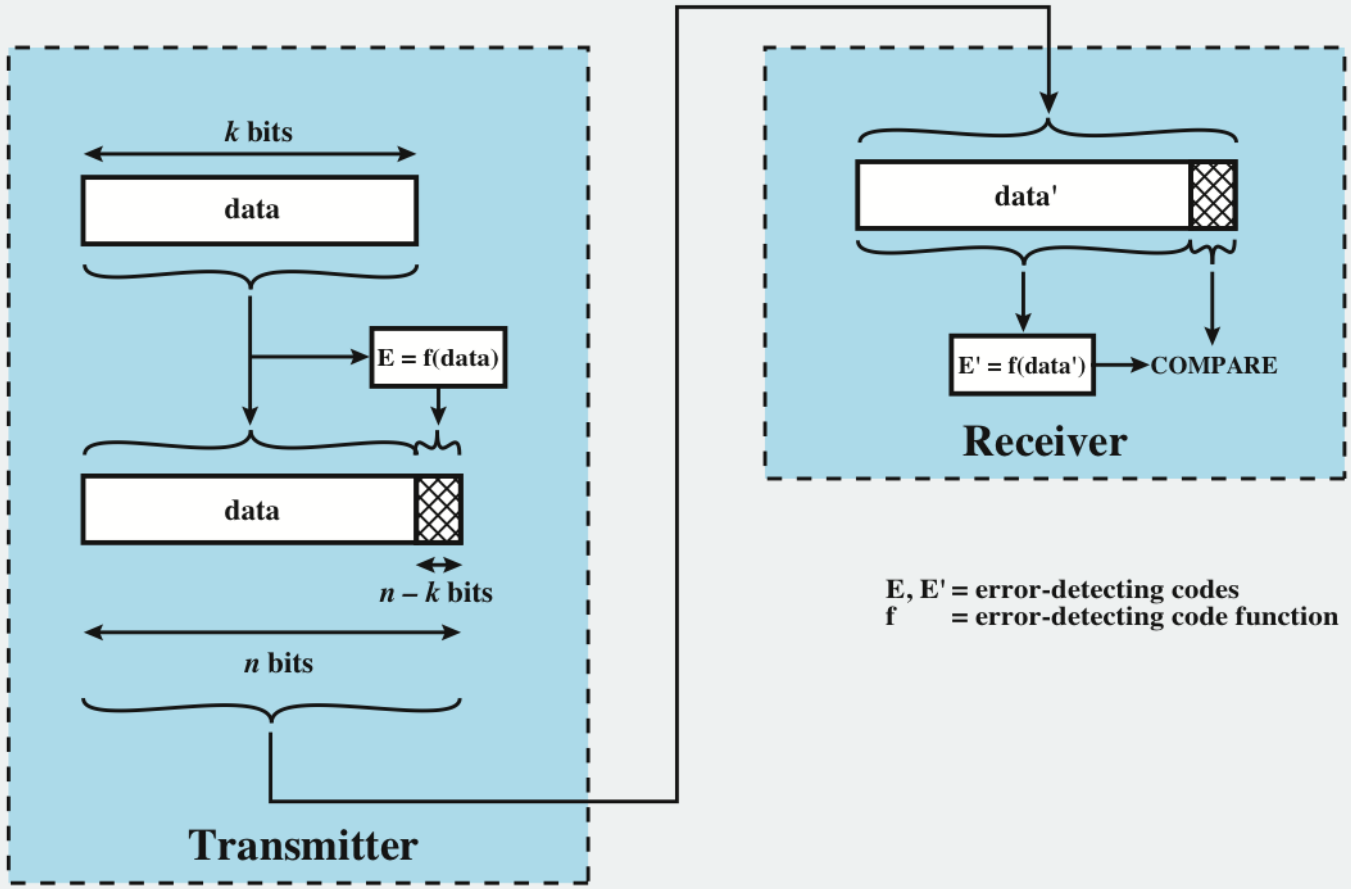


Figure 6.2 Error Detection Process

Parity Check

Main idea: Let N_1 be the number of bits equal to one in a F-bit frame.

- "Even parity": make the number of 1's even.
  - If N_1 is odd, append a one to the message.
  - If N_1 is even, append a zero to the message.
- "Odd parity": make the number of 1's odd.
  - If N_1 is odd, append a one to the message.
  - If N_1 is even, append a zero to the message.

Example: Encode message "1110001" (these bits are in the usual order with MSB on the left and LSB on the right) with odd parity. Note that $N_1 = 4$. So the transmitted bits would be 11110001 (LSB transmitted first, so "appending" the parity puts the parity bit left of the MSB).

Parity Check: Remarks

- Main advantage: simple to implement.
- Can detect any odd number of bit errors.
- Main disadvantage: can't detect an even number of bit errors

Example: Suppose we transmitted the odd parity frame

$$11110001$$

and we received

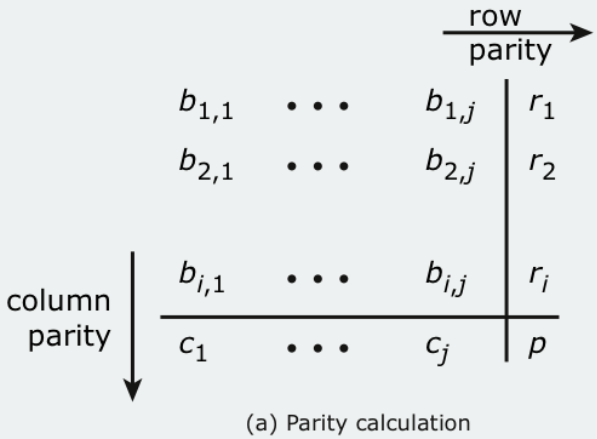
$$10110000$$

Received frame still has odd parity, so errors would not be detected.

Two-dimensional parity check:

- Arrange a block of $F = MN$ bits into an $M \times N$ array.
- Add a parity bit to each row (example is even parity)
- Add a parity bit to each column (example is even parity)
- Add one more parity bit on the bottom right corner (example is even parity)
- Total parity bits added is $M + N + 1$

Can detect an odd number of bit errors. Can also correct some error patterns.



(a) Parity calculation

0	1	1	1	0	1
0	1	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	0	0	1	1	0

(b) No errors

0	1	1	1	0	1
0	(0)	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	0	0	1	1	0

row parity error

column parity error

(c) Correctable single-bit error

0	1	1	1	1	1	0	1
0	(0)	1	1	0	(1)	1	0
0	0	1	1	0	0	1	1
0	(0)	0	0	0	(0)	0	0
1	0	1	1	1	1	1	0
1	1	0	0	0	1	1	0

(d) Uncorrectable error pattern

Figure 6.3 A Two-Dimensional Even Parity Scheme

Checksums

Basic idea is to compute some function of the bits and append the result to the frame/message. Most common implementation is the "internet checksum":

1. Take two numbers with the same number of bits from the frame header and add them.
2. If the result has one more bit than the original numbers (a final carry bit), remove this bit and add one to the previous sum.

Simple example of steps 1-2:

$$\begin{array}{r} 1101 \\ +1011 \\ \hline 11000 \\ +1 \\ \hline 1001 \end{array}$$

Checksum Example: 00 01 F2 03 F4 F5 F6 F7 00 00

Partial sum	0001 F203 F204
Partial sum	F204 F4F5 1E6F9
Carry	E6F9 1 E6FA
Partial sum	E6FA F6F7 1DDF1
Carry	DDF1 1 DDF2
Ones complement of the result	220D

(a) Checksum calculation by sender

Partial sum	0001 F203 F204
Partial sum	F204 F4F5 1E6F9
Carry	E6F9 1 E6FA
Partial sum	E6FA F6F7 1DDF1
Carry	DDF1 1 DDF2
Partial sum	DDF2 220D FFFF

(b) Checksum verification by receiver

Figure 6.4 Example of Internet Checksum

Result of all ones (FFFF) indicates checksum verified.

Cyclic Redundancy Check (CRC) (part 1 of 2)

Basic idea:

1. Take a k-bit block of data D.
2. Calculate a $n-k$ "frame check sequence" (FCS) denoted as F.
3. Append the FCS to the data to form an n-bit frame $T = [D\ F]$.

The FCS is calculated so the resulting n-bit frame T is exactly divisible by some predetermined number P that is $n-k+1$ bits long.

Note that $T = 2^{n-k}D + F$. At the transmitter, to compute F, we compute

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P}$$

and set $F = R$ (the FCS is just the remainder). Note that R will have $n-k$ bits since the divisor has $n-k+1$ bits.

Cyclic Redundancy Check (CRC) (part 2 of 2)

Assuming no errors, at the receiver we compute

$$\frac{T}{P} = \frac{2^{n-k}D + F}{P} = Q + \frac{R}{P} + \frac{R}{P}$$

The addition is performed "modulo 2" (XOR), e.g.,

$$\begin{array}{r} 1101 \\ +1011 \\ \hline 0110 \end{array}$$

Since $\frac{R}{P} + \frac{R}{P} = \frac{R+R}{P}$ and modulo 2 addition of any number with itself is zero, we get

$$\frac{T}{P} = Q + 0$$

The receiver declares the frame to be correct if there the remainder is zero.

Polynomial Representation of Binary Numbers

One convenient way to divide binary numbers is to use to use polynomial representations. Given a length-n binary number N with bits $[b_{n-1}, b_{n-2}, \ldots, b_1, b_0]$ (usual bit ordering), the polynomial representation of that number is

$$N(x) = \sum_{k=0}^{n-1} b_k x^k$$

Example: $N = 11001$. We get

$$N(x) = 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot 1 = x^4 + x^3 + 1$$

CRC Example Setup

Suppose we have $k = 6$ and $n = 8$. Our data D and predetermined divisor P are given as

$$D = 111011$$
$$P = 101$$

in the usual bit ordering (MSB on left and LSB on right). Note the three bit divisor P implies a $n - k = 2$ bit remainder R. The polynomial representations are

$$D(x) = x^5 + x^4 + x^3 + x + 1$$
$$P(x) = x^2 + 1$$

At the transmitter, we compute $\frac{x^{n-k}D(x)}{P(x)} = Q(x) + \frac{R(x)}{P(x)}$. We don't care about $Q(x)$

CRC Calculation at the Transmitter

$$x^{n-k}\,D(x) = x^2\left(x^5+x^4+x^3+x+1\right) = x^7+x^6+x^5+x^3+x^2$$

long division...

$$x^2+1 \;\overline{)\; x^7+x^6+x^5+0x^4+x^3+x^2}$$

$$x^5+x^4+0x^3+x^2+x$$

$$x^7+0x^6+x^5$$

$$x^6+0x^5$$

$$x^6+0x^5+x^4$$

$$x^4$$

$$x^4+0x^3+x^2$$

$$x^3+0x^2$$

$$x^3+0x^2+x$$

$$x+0$$

all math modulo 2

- $x^a x^b = x^{a+b}$
- $x^a + x^a = 0$
- $x^a - x^a = 0$
- $x^a - 0 = x^a$
- $0 - x^a = x^a$

Remainder $R(x) = x+0 \Rightarrow R = 10$

Transmitted frame $T = \boxed{D}\boxed{R} = \boxed{111011}\boxed{10}$

CRC Check at the Receiver

now compute $\frac{T(x)}{P(x)}$ assuming no errors...

$$
\begin{array}{r}
x^5 + x^4 + 0x^3 + x^2 + x + 0 \\
x^2+1 \;\overline{)\; x^7 + x^6 + x^5 + 0x^4 + x^3 + x^2 + x + 0} \\
\underline{x^7 + 0x^6 + x^5} \\
x^6 + 0x^5 \\
\underline{x^6 + 0x^5 + x^4} \\
x^4 \\
\underline{x^4 + 0x^3 + x^2} \\
x^3 + 0x^2 \\
\underline{x^3 + 0x^2 + x} \\
0x^2 + 0x
\end{array}
$$

modulo-2 math again

can stop here since we know remainder will be zero.

$R(x) = 0 \Rightarrow$ no errors detected

CRC Check at the Receiver

What if $T(x) = x^7 + 0x^6 + x^5 + 0x^4 + x^3 + x^2 + x + 0$?

↑ error here

$$
\begin{array}{r|l}
x^2+1 & x^5 + 0x^4 + 0x^3 + 0x^2 + x + 1 \\
\hline
 & x^7 + 0x^6 + x^5 + 0x^4 + x^3 + x^2 + x + 0 \\
 & \underline{x^7 + 0x^6 + x^5} \\
 & \quad 0 + 0x^4 + x^3 + x^2 + x + 0 \\
 & \quad\quad \underline{x^3 + 0x^2 + x + 0} \\
 & \quad\quad\quad x^2 + 0 \\
 & \quad\quad\quad \underline{x^2 + 1} \\
 & \quad\quad\quad 1
\end{array}
$$

remainder not zero $\rightarrow$ 1

$\Rightarrow$ error detected.

Final Remarks

- All of these methods are for error detection.
- Generally, they do not provide any way to locate/correct the errors.
- Erroneous frames are typically discarded and requested again.
- Parity check:
  - Extremely simple
  - Can be fooled by certain types of errors (e.g., two bits in error)
  - Useful for small frame sizes when the typical error is a single bit.
- Checksum:
  - Pretty simple
  - More powerful than parity check (lower probability of undetected errors)
  - Often used as a secondary mechanism for error detection.
- CRC:
  - Most complicated
  - Most powerful (extremely low probability of undetected errors for longer CRC checks)
  - IEEE 802 specifies a CRC-32 check (33 bit polynomial $P(x)$)