CS244a: An Introduction to Computer Networks

Handout 13:

Error Detection and Correction

Nick McKeown

Professor of Electrical Engineering and Computer Science, Stanford University

nickm@stanford.edu

http://www.stanford.edu/~nickm

Outline

- Basic ideas: BER, PER and Hamming Distance
- Parity
- Error Detection: Cyclic Redundancy Codes
- Error Correction
- Error Detection versus Correction

Errors

- A bit error occurs when a source sends a bit, b, and the destination receives NOT b.
  i.e. $b \rightarrow b \oplus 1$.
- The error can take place on the link (e.g. EM interference, or signal loss), in a (malfunctioning) switch or router along the path, or in the source or destination (e.g. failed hardware, or bit errors in memories).
- The bit error rate (BER) tells us the probability of any given bit being in error. Typical values are BER = 10^{-9} for an electrical link, and 10^{-12} for an optical link.

Bit error rate

An example

Assume an N-bit packet, with known BER and independent errors:

Packet Error Rate = PER = $1 - (1 - \text{BER})^N$

PER ~= N (BER) if N (BER) << 1

e.g. N = 10^4, BER = 10^{-7} = PER = 10^{-3}

In practice, bit errors occur in *bursts*:

- Perhaps caused by mechanical switches that switch slowly relative to a bit-time.
- If a bit is in error, it is likely that the next bit is in error too. Therefore, bit errors are not independent.
- So for a given BER, PER < N (BER)

Detecting and Correcting Errors

- When we transmit a message, we typically append a checksum to the message.
- The checksum is calculated by performing a function over all the bits in the message.
- For example, the Internet (IP and TCP) checksum is a 16-bit ones-complement sum of the data.
- What sort of errors can we expect it to catch?

Detecting Errors

- The IP and TCP checksum will catch any burst error or 15 or fewer bits.
- In general, it will catch approximately 1 in 2^{16} of all possible errors. (Why?)
- As we will see, stronger checksums are possible.
- Q: In general, can we design a checksum that will always catch errors?

Encoding to detect errors

- We use codes to help us detect errors.
- The set of possible messages is mapped by a function onto the set of codes.
- We pick the mapping function so that it is easy to detect errors among the resulting codes.
- Example: Consider the function that duplicates each bit in the message. E.g. the message 1011001 would be mapped to the code 11001111000011, and then transmitted by the sender. The receiver knows that bits always come in pairs. If the two bits in a pair are different, it declares that there was a bit error.
- Of course, this code is quite inefficient…

Hamming Distance

Number of bits that differ between two codes

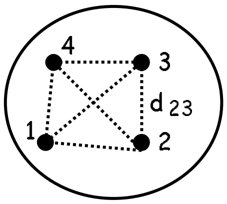
e.g. 1 0 0 1 0 1 0 1
1 0 1 1 1 0 0 1

0 0 1 0 1 1 0 0 → HD=3

In our example code (replicated bits), all codes have at least two bits different from every other code. Therefore, it has a Hamming distance of 2.

Hamming Distance

Set of codes



HD = $\min_{ij} (d_{ij})$

To reliably detect a d-bit error: HD > d
To reliably correct a d-bit error: HD > 2d

Parity: A simple error detecting code

Parity added to make @ 1s even/odd

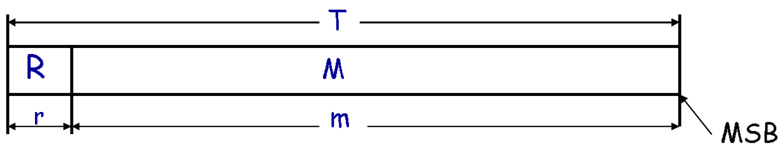
0 1 1 1 0 1 0 1 | 1

If parity is wrong ➡ ERROR

If parity is right ➡ NO ERROR
(or an even number of errors has occurred)

Q: What is the minimum Hamming distance for this code?

Error Detection
Cyclic Redundancy Code (CRC)

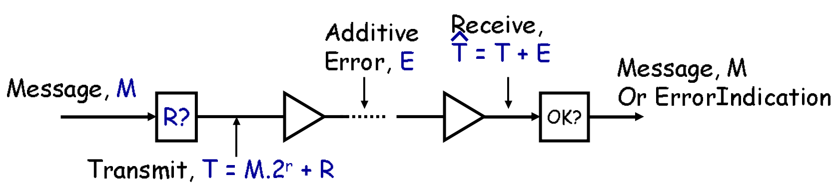


i.e. $T = M.2^r + R$

Modulo-2 addition (XOR)

Cyclic Redundancy Code (CRC)

1. Choose R to maximize the Pr{detecting error}
2. Agree on G, a generator.
3. Choose R such that T = AxG, for some A.
4. Transmit T.
5. On reception, check that $\hat{T}$ = AxG.



So, $\hat{T} = T + E = AxG + E$

Cyclic Redundancy Code (CRC)

Choosing G is critical to detecting errors.
In general, G is chosen to:

1. Detect single-bit errors.
2. Detect any 2 single-bit errors.
3. Detect burst errors.
4. Detect other errors not considered here…

Since: $\hat{T} = T + E = AxG + E$
If E(mod)G = 0 then error is not detected.

Cyclic Redundancy Code (CRC)

A convenient representation:
If, for example, M = 1 0 1 1, then $M(x) = x^3 + x + 1$

(i) Detecting Single-bit errors.

Single-bit errors can be represented as:
$E(x) = x^i$

If $G(x) = x^k + x^j$, *i.e.* G(x) has 2 terms, then since G(x) cannot divide E(x):
ALL single bit errors can be detected.

Cyclic Redundancy Code (CRC)

(ii) Two isolated errors can be represented as:

$E(x) = x^i + x^j = x^j (x^{i-j} + 1)$, where $i-j > 1$

If $(x^k + 1)$ is not a factor of $G(x)$ for any $k>1$ then $G(x)$ does not divide $E(x)$, and so

ALL two-bit errors are detected.

(iii) Detecting an odd number of errors

Theorem: If $(x + 1)$ is a factor of $G(x)$ then:

ALL odd errors are detected.

Proof by contradiction:

Assume E has odd # of 1's and $(x + 1)$ is a factor of $E(x)$:

i.e. $E(x) = (x + 1)E'(x)$ for some $E'(x)$

Cyclic Redundancy Code (CRC)

iii) Proof (cont.)

We can evaluate $E(x)$ for $x=0$ or $x=1$.

Evaluating $E(x)$ for $x=1$:

$E(1) = (1 + 1) E'(1) = 0$

But E has an odd # of 1's so : $E(1) = 1$

Therefore, if E has an odd # of 1's,

it is not divisible by $(x + 1)$

Cyclic Redundancy Code (CRC)

(iv) Bursts up to length r (degree of G)

Burst of length k: $x^i (x^{k-i} + \ldots + 1)$

If G(x) is of the form $\ldots + x^o$ then:

x^i is not a factor

(v) Longer bursts are not guaranteed to be detected.

(vi) In general, an r-bit generator can detect 1 in 2^r errors.

An example of a widely used 16-bit CRC generator:

CRC-CCITT = $x^{16} + x^{12} + x^5 + 1$

Calculating a CRC

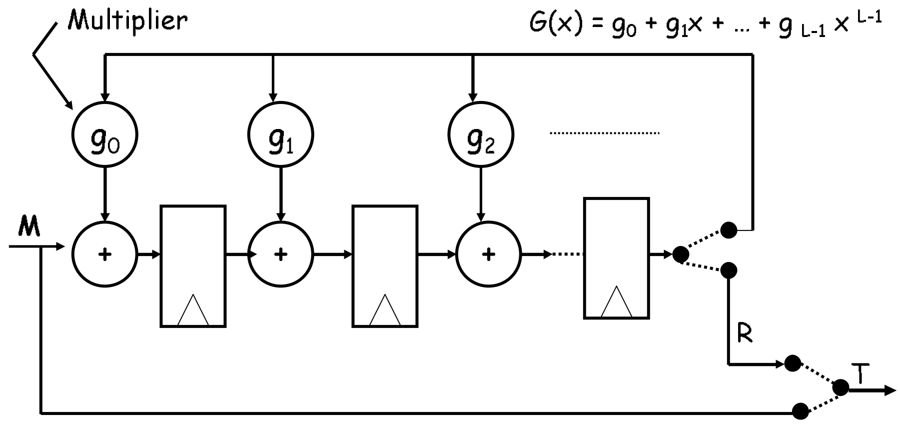
Example: M= 110101, G = 1001

```
1001 | 110101000
       1001
        1000
        1001
         0011
         0000
          0110
          0000
           1100
           1001
            1010
            1001
             011  <----- R
```

T = 110101| 011

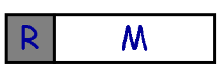
Circuit for Calculating CRC

$G(x) = g_0 + g_1 x + \ldots + g_{L-1} x^{L-1}$



Error Correction

Common code used: Bose-Chaudhuri-Hocquenghem (BCH)



BCH (R + M, M, t)

e.g. BCH (1023, 923, 10)

Can detect all "t" bit errors

If t=1 then the code is called Reed-Solomon and is used in CD players

Detect or Correct?

Advantages of Error Detection
- Requires smaller number of bits/overhead.
- Requires less/simpler processing.

Advantages of Error Correction
- Reduces number of retransmissions.

Most data networks today use error detection, not error correction.

Detect or Correct?

An example

Assume: 1. Packets are of lengths 923 bits
2. PER = 10^{-5}

Overhead of Error Correction:

Assume we use: BCH (1023, 923, 10)
Therefore, we send 923 data bits as 1023 bits.

$$\text{Transmission Overhead} = \frac{100}{923} \approx 10\%$$

Overhead of Error Detection:

Assume we use: 32-bit CRC; one retransmission per error.
Therefore, we send 923 data bits as 955 bits.

$$\text{Transmission Overhead} = \frac{(923 + 32)\, 10^{-5} + 32}{923} \approx 3\%$$