Getting Connected

Chapter 2, Part 2

Networking
CS 3470, Section 1
Sarah Diesburg

Five Problems

- Encoding/decoding
- Framing
- Error Detection
- Error Correction
- Media Access

Five Problems

- Encoding/decoding
- Framing
- Error Detection
- Error Correction
- Media Access

Why Error Detection?

- Bit errors are sometimes introduced into frames
- Some mechanism is needed to detect these errors!
  - Otherwise, strange file corruptions could occur on the receiver's end

Basic Idea

- Add redundant information to a frame
  - Can be used to determine if errors
- Want redundant information to be as small as possible
- Redundant information is often called ***error-detecting codes*** or ***checksums***

Error Detection/Correction

- 2-D Checks
- Internet checksums
- Cyclic Redundancy Check

Error Detection/Correction

- 2-D Checks
  - Divides bytes into even rows and columns (e.g. 8 rows of 8 bytes)
  - Computes even parity across rows and down columns
  - Extra parity bits are sent along with the data
    - # of parity bits scales up with the amount of data

Error Detection/Correction

- 2-D Checks

```
0 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1
1 1 0 1 1 0 1 1
0 0 0 0 0 0 0 1
1 1 1 1 1 0 0 0
1 0 0 1 0 1 1 1
0 0 1 0 0 0 0 0
1 1 1 0 0 0 0 1
```

Error Detection/Correction

- 2-D Checks

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

Error Detection/Correction

- 2-D Checks : Error

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

Error Detection/Correction

- 2-D Checks
  - Catches all 1,2,3-bit errors
  - Catches most 4-bit errors

Error Detection/Correction

- Internet Checksums
- View payload as 16-bit integers
- Sum up payload, using 1's complement, with carry wraparound.
- Transmit payload along with the result for validation on the receiving side.
- Easy to implement in software/hardware (see the code-snippet in the text)

Error Detection/Correction

- Internet Checksums
- Negative number -x is x with all bits inverted
- When two numbers are added, the carry-on is added to the result
- Example: -15 + 16 (assume 8-bit)

```
15 = 00001111 →   -15 = 11110000
                            +
                     16 = 00010000
                        1 00000000
                            +
                                 1
                          00000001
                       -15+16 = 1
```

Error Detection/Correction

- Internet Checksums
- Usability
  - 16 bits for messages of any length
  - Strength of error-detection not as good as 2D
    - Pair of single-bit errors could go unnoticed

Error Detection/Correction

- Cyclic Redundancy Check
  - Based upon polynomial division
  - Bit strings are considered to represent the coefficients of a polynomial:

    m+1 bits <=> degree m polynomial
    - eg:
      01101101 $\Leftrightarrow 0x^7 + 1x^6 + 1x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 1x^0$
      $\Leftrightarrow x^6 + x^5 + x^3 + x^2 + 1$
  - Long division is carried out as usual, but polynomial subtraction is done modulo-2:
    - eg:
      $(x^6 + x^5 + x^3 + x^2 + 1) - (x^6 + x^4 + x^3 + x + 1) = ??$

Error Detection/Correction

- Cyclic Redundancy Check
  - Based upon polynomial division
  - Bit strings are considered to represent the coefficients of a polynomial:

    m+1 bits <=> degree m polynomial
    - eg:
      01101101 <=> $0x^7 + 1x^6 + 1x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 1x^0$
      <=> $x^6 + x^5 + x^3 + x^2 + 1$
  - Long division is carried out as usual, but polynomial subtraction is done modulo-2:
    - eg:
      $(x^6 + x^5 + x^3 + x^2 + 1) - (x^6 + x^4 + x^3 + x + 1) = x^5 + x^4 + x^2 + x$

Error Detection/Correction

- Cyclic Redundancy Check
  - Long division is carried out as usual, but polynomial subtraction is done modulo-2:
    - eg:
      $(x^6 + x^5 + x^3 + x^2 + 1) - (x^6 + x^4 + x^3 + x + 1) = x^5 + x^4 + x^2 + x$
    - 1 – 0 = 1
    - 0 – 1 = 1 (modulo 2, remember)
    - 1 + 1 = 0 (modulo 2!)
    - 1 – 1 = 0
    - 0 – 0 = 0
  - Hey! This is just XOR. That makes it easy.

Error Detection/Correction

- Cyclic Redundancy Check
  - Sender and receiver have a common generator polynomial (determined in advance, by the protocol).
  - Eg:
    - 1101011011 (Frame) $x^9 + x^8 + x^6 + x^4 + x^3 + x + 1$
    - 10011 (Generator) $x^4 + x + 1$
  - Procedure:
    - degree(G)=r (hence r+1 bits)
    - degree(M) = m (or, m+1 bits in frame)
    - Promote the frame's polynomial by r bits, so that it is degree m+r. I.e., $x^r M(x)$ is the polynomial we're working with.

Error Detection/Correction

- Cyclic Redundancy Check
  - Procedure:
    - degree(G)=r (hence r+1 bits)
    - degree(M) = m (or, m+1 bits in frame)
    - Promote the frame's polynomial by r bits, so that it is degree m+r. I.e., $x^r M(x)$ is the polynomial we're working with.
    - Divide this polynomial by G(x) using polynomial division.
    - Subtract the remainder from $x^r M(x)$ using modulo-2 subtraction. The result is the transmission payload T(x)

Error Detection/Correction

10011 <=> $G(x) = x^4 + x + 1$

1101011011 <=> $M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x + 1$

$10011\ \overline{\big)\,11010110110000}$

Promote by x^r; $x^rM(x)$

Error Detection/Correction

```
              1
10011 |11010110110000
       10011
        10011
```

Error Detection/Correction

```
             11
       ____________________
10011 |11010110110000
       10011
        10011
        10011
        _____
            1
```

Error Detection/Correction

```
              110
       ________________
10011 |11010110110000
       10011    |
        10011   |
        10011   |
             10
```

Error Detection/Correction

```
             1100
10011 |11010110110000
       10011    |
        10011   |
        10011   |
             101
```

Error Detection/Correction

```
              11000
        _______________
10011 |11010110110000
       10011
        10011
        10011
             1011
```

Error Detection/Correction

```
             110000
10011 |11010110110000
       10011
        10011
        10011
             10110
```

Error Detection/Correction

```
             1100001
       _______________
10011 |11010110110000
       10011
        10011
        10011
             10110
             10011
             -----
               1010
```

Error Detection/Correction

```
           11000010
      _______________
10011 |1101011011 0000
       10011
        10011
        10011
             10110
             10011
               10100
```

Error Detection/Correction

```
             110000101
       ________________
10011 |11010110110000
       10011
        10011
        10011
             10110
             10011
               10100
               10011
               -----
                1110  <---- Remainder
```

Error Detection/Correction

```
           1100001011  <------  Discarded
       ______________
10011 |11010110110000
       10011
        10011
        10011
             10110
             10011
               10100
               10011
               -----
                 1110
```

Error Detection/Correction

```
T(x) = x^r M(x) - r(x)

          11010110110000
        -           1110
          --------------
          11010110111110
```

A simple XOR

Error Detection/Correction

- So what does the receiver do?
  - G(x) should divide evenly into T(x)!
    - If it doesn't, receiver asks for frame again or tries to correct the errors
- Where does G(x) come from?
  - Looked up ahead of time
  - Pick G(x) so that it cannot easily be divided evenly
- CRC-32 (for TCP)

$$G(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$