Discussion 3: CRC, Sliding Window Protocol

CSE 123: Computer Networks

Marti Motoyama & Chris Kanich

UCSDCSE

Modulo-2 Arithmetic

- Multiplication

```
    1101
     110
  ------
    0000
   11010
  110100
  ------
  101110
```

- Division

```
        1101
110 | 101110
       110
      ----
        111
        110
      ----
         011
         000
      ----
         110
```

Cyclic Remainder Check

- Idea is to *divide* the incoming data, D, rather than add
  - The divisor is called the generator, g
- We can make a CRC resilient to k-bit burst errors
  - Need a generator of $k+1$ bits
- Divide $2^k D$ by g to get remainder, r
  - Remainder is called frame check sequence
- Send $2^k D+r$
  - Note $2^k D$ is just D shifted left k bits
  - Remainder must be at most k bits
- Receiver checks that $(2^k D+r)/g = 0$

CRC: Rooted in Polynomials

- We're *actually* doing polynomial arithmetic
  - Each bit is actually a coefficient of corresponding term in a kth-degree polynomial

$$1101 \text{ is } (1 * X^3) + (1 * X^2) + (0 * X^1) + (1 * X^0)$$

- Why do we care?
  - Can use the properties of finite fields to analyze effectiveness
  - Says any generator with two terms catches single bit errors

CRC Example Encoding

$x^3 + x^2 + 1$ = 1101 Generator

$x^7 + x^4 + x^3 + x$ = 10011010 Message

```
1101 | 10011010000
       1101
        1001
        1101
         1000
         1101
          1011
          1101
           1100
           1101
             1000
             1101
              101
```

Message plus *k* zeros (*2^k)

k + *1* bit check sequence *g*, equivalent to a degree-k polynomial

Remainder

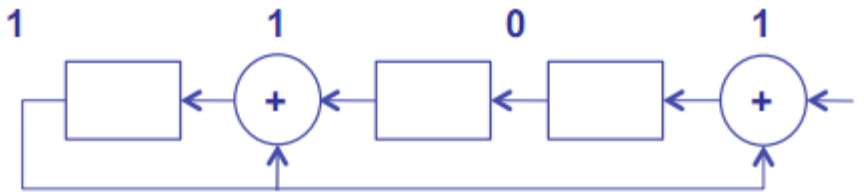
D mod *g*

Result:

Transmit message followed by remainder:

10011010101

CRC in Hardware



- Key observation is only subtract when MSB is one
  - Recall that subtraction is XOR
  - No explicit check for leading one by using as input to XOR
- Hardware cost very similar to checksum
  - We're only interested in remainder at the end
  - Only need *k* registers as remainder is only *k* bits

CRC Example Decoding

$x^3 + x^2 + 1$	= 1101	Generator
$x^{10} + x^7 + x^6 + x^4 + x^2 + 1$	= 10011010101	Received Message

```
1101 | 10011010101
       1101
        1001
        1101
         1000
         1101
          1011
          1101
           1100
           1101
             1101
             1101
                0
```

Received message, no errors

k + 1 bit check sequence *g*, equivalent to a degree-k polynomial

Remainder

D mod *g*

Result:

CRC test is passed

CRC Example Failure

$x^3 + x^2 + 1$	= 1101	Generator
$x^{10} + x^7 + x^5 + x^4 + x^2 + 1$	= 10010110101	Received Message

```
1101 | 10010110101
       1101
        1000
        1101
         1011
         1101
          1101
          1101
           0101
```

Received message

Two bit errors

k + 1 bit check sequence *g*, equivalent to a degree-k polynomial

Remainder

D mod *g*

Result:

CRC test failed

More than one way…

- Hardware based – encode and decode with XOR and shift register
- Software based – encode by padding and dividing, decode by divide and check for zero remainder

- Any other ways to implement this? Easier? Slower?

Homework #1, Problem #4

USB uses a 5 bit CRC with polynomial $x^5 + x^2 + 1$. Find a pair of distinct values that share the same CRC value, and show that they are the same either via a mathematical argument or polynomial long division as in the book. You may find it helpful to write the CRC portion of the project, modify it for this CRC length and polynomial, and use that code as part of a program that searches for CRC value pairs.

- What real-world situation does this problem mimic?
- What are a few approaches for finding a collision?

Sliding Window Protocol

- Sliding window basics (slides from class)
- Sliding window in wireshark
- Sliding window example (on the board)
- Sliding window question from Moodle

Sliding Window Protocol

- See: Alex's slides for lecture 5

Sliding Window Protocol

- SWS = RWS = 4
- One way delay 200ms
- Delay acks by 50ms, batch acks together
- Assume infinite sequence numbers, in order delivery
- Simulate 6 data packets worth of transmission

Sliding Window Protocol

- Use wireshark to inspect a flow.
- Disc.3.pcap will be available along with these slides online.

Sliding Window Protocol

Seq and Ack fields
by Robert Condy - Sunday, 10 October 2010, 01:11 PM

In the book's implementation of sliding window, it uses two different fields in the frame for sequence number and acknowledgement number. I am wondering why this is necessary. It seems to me that they could use the same field. More specifically, I would think that if the frame is flagged as an ack it should insert the ack number into this field; if it is flagged as data, it should insert the sequence number into this field. I googled the topic, and came to discover that TCP uses two different fields as well. Still couldn't find an explanation why though. Seems like a waste of bandwidth to me. Am I missing something obvious?

Edit | Delete | Reply

Sliding Window Protocol

Re: Seq and Ack fields
by Christopher Kanich - Sunday, 10 October 2010, 06:15 PM

This is a really good question. So good that I don't want to give the answer away right away, and it's something we can talk about during discussion tomorrow. I won't hold the question hostage though - I'll post a full response tomorrow evening for those that can't make it to discussion.

One hint I can give right now - consider what effects this optimization would have on two way communication. The project really only mimics one way communication, and we've mainly been talking about communication in terms of a sender and a receiver; in reality, communication is most useful when data can flow both ways between two nodes.

Specifically, think about something like a game server where input commands from the client and updates about the game world from the server are happening very, very frequently. What sort of optimizations might you want to happen, and why wouldn't you be able to do them with only one sequence number field?

Show parent | Edit | Split | Delete | Reply