CMSC 417 Spring 2016 Lecture #19 4/18/2016

Agenda

⇒ p4 due friday

⇒ office hours this week

- Rama: Th 11a-1p
- Colin: W 3-5p

⇒ Internet Checksum cont'd

⇒ CRC

CMSC 417 Spring 2016 Lecture #19 4/18/2016

Internet Checksum cont'd

⇒ how well does it work?

- catches all single bit errors
- misses any errors which keep the sum the same, e.g.,
  - add X to one word
  - subtract X from another word

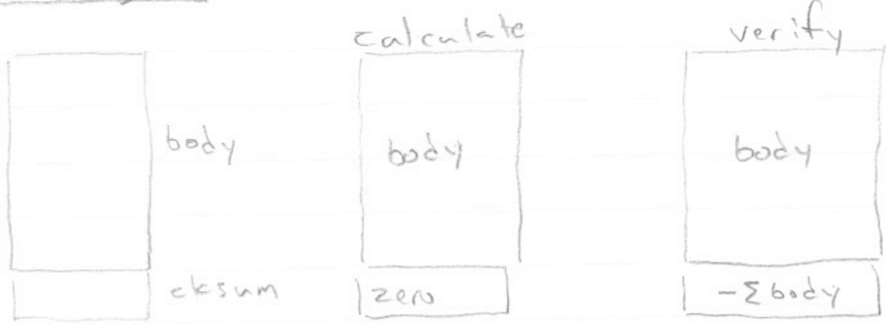
⇒ $Pr\{\geq 2 \text{ bit errors}\} \approx Pr\{1 \text{ bit error}\}^2$

⇒ generally it reduces errors quite a bit, but not nearly as well as CRC

⇒ used as a defense against errors escaping the Link layer, e.g., CRC in Ethernet

⇒ also, it's really simple to write code to do it!

Basic Logic



$\Sigma \text{body} + 0 = \Sigma \text{body}$

$\Sigma \text{body} + -\Sigma \text{body} = 0$

CMSC417 Spring 2016 Lecture #19 4/18/2016

Internet Checksum Examples with Carries

5+ -0

```
   0101   //5
 + 1111   //-0
  10100
 +    1
   0101   //5.
```

5+ -4

```
   0101   //5
 + 1011   //-4
  10000
 +    1
      1
```

```
   0101   //5
   1011   //-4
   0001   //1
   0110   //6
 + 1101   //-2
 100100
     10
   0110   //6
```

⇒ 1s: 4 ⇒ 0 carry 2
⇒ 2s: 4 ⇒ 0 carry 2
⇒ 4s: 5 ⇒ 1 carry 2
⇒ 8s: 4 ⇒ 10

```
   0101   //5
   1011   //-4
   0001   //1
   0110   //6
   1101   //-2
 + 1001   //-6
 101101
 +    10
   1111   //-0 == 0
```

⇒ if we flip the bits, we get -6 = 1001 now sum
⇒ we get zero
⇒ to check a checksum, make sure it's zero
⇒ to calculate, set the checksum field to zero and do the math
⇒ why make things sum to zero rather than just storing the sum?

CMSC 417 Spring 2016 Lecture #19 4/18/2016

TCP pseudo header

⇒ TCP checksum is over TCP header, data and TCP pseudo header

bits: 0 8 16 24 32

IP src addr		
IP dst addr		
zero	IP proto	TCP seg. length.

} 12 bytes

⇒ IP src, IP dst, IP proto from IP header

⇒ TCP seg length is TCP data length + TCP header length

checksum

TCP pseudo header
TCP header checksum=0
TCP data

} 16-bit one's complement

⇒ pad odd # of bytes in data with zeros to even

SYN cookies

⇒ people can send you SYNs and make you keep state

- ☐ can you push this off onto the client?
- ☐ Yes! pick your initial chesum carefully

0 5 8 32

time	state	cryptographic hash

- ☐ need to encode state in 3-bits
  - ⇒ originally just encoding 8 values for MSS
- ☐ recover state by (1) subtract one from ACK, (2) check time is recent (3) verify crypto hash

CMSC 417 Spring 2016 Lecture #19 4/18/2016

Cyclic Redundancy Check

⇒ powerful checksum with deep mathematical foundations (finite field theory)

⇒ think of $(n+1)$-bit message as an nth-degree polynomial (in math $\in \{0,1\}[x]$)

$b_0 + b_1 x + b_2 x^2 + \ldots b_n x^n = M(x)$ // message

key idea

⇒ pick a divisor polynomial $C(x)$

- there are well-known good choices

⇒ left shift M by k bits

⇒ compute a $(k-1)$th degree polynomial $R(x)$ then let $P(x) = M(x) \cdot \underbrace{x^k}_{\text{left shift}} + R(x)$

⇒ compute $R(x)$ s.t. $P(x) / C(x) = 0$

polynomial division w/ binary coefficients

⇒ if you have $B(x)$ and $C(x)$ where $\text{degree}(B(x)) \geq \text{degree}(C(x))$, then $B(x) \% C(x) =$ xor coefficients

⇒ example $(x^3+1) \% (x^3+x^2+1)$

$1001 \% 1101 = 1001$

xor 1101

$0100 \Rightarrow x^2$

- note that

$(x^3+1) = (x^3+x^2+1) \times \underbrace{1}_{\text{quotient}} + \underbrace{x^2}_{\text{remainder}}$

example, not a proof

⇒ $\text{degree}(B(x)) = \text{degree}(C(x)) \Rightarrow B(x)/C(x) = 1$

⇒ $\text{degree}(B(x)) \leq \text{degree}(C(x)) \Rightarrow B(x) \% C(x) = B(x)$ and $B(x) / C(x) = 0$

CMSC 417 Spring 2016 Lecture #19 4/18/2016

CRC algorithm)

1.) divide $M(x) \cdot x^k$ by $C(x)$ to get the remainder $R(x)$

2.) send $P(x) = M(x) \cdot x^k - R(x)$

$\Rightarrow$ since $M(x) \cdot x^k \ \% \ C(x) = R(x)$

$\exists \ D(x)$ s.t. $M(x)\, x^k = C(x)\, D(x) + R(x)$

thus $P(x) = M(x)\, x^k - R(x) = C(x)\, D(x)$

thus $P(x) \ \% \ C(x) = 0$

example)

$M(x) = x^7 + x^4 + x^3 + x = 10011010$

$C(x) = x^3 + x^2 + 1 = 1101$

```
           ?????   <-- don't care about the quotient
      ______________
1101 ) 10011010 000      left shift
       1101
       ----
        1001
        1101
        ----
         1000
         1101
         ----
          1011
          1101
          ----
           1100
           1101
           ----
            1000
            1101
            ----
             101  <-- remainder
```

$\Rightarrow$ called the "generator"

$\Rightarrow$ subtraction is also xor so $M(x)x^k - R(x)$ is $M(x)x^k$ xor $R(x) = \underbrace{10011010}_{M}\underbrace{101}_{R}$

CMSC417 Spring 2016 Lecture #19 4/18/2016

How to pick $C(x)$

⇒ want to pick $C(x)$ s.t. $P(x)+E(x) \% C(x) \neq 0$ with high probability ($E(x)$ is error)

⇒ $(P(x)+E(x)) \% C(x) = 0 = P(x) \% C(x) + E(x) \% C(x)$
implies $E(x) \% C(x) = 0$ since $P(x) \% C(x) = 0$

⇒ question is how can we minimize $Pr\{E(x) \% C(x) = 0\}$

common errors

⇒ single bit: $E(x) = x^i$
- □ $x^i \% C(x) = 0$ ⇒ only the ith coefficient of $C(x)$ is nonzero b/c % is xor of coefficients
- □ by contrapositive if $C(x)$ has a nonzero coefficient other than the ith, then $E(x) \% C(x) \neq 0$
- □ if $C(x)$ has two nonzero coefficients then, $C(x)$ has a nonzero coefficient other than the ith
- □ if $C(x)$ has two nonzero coefficients then, CRC will catch all single bit errors

⇒ all 2-bit errors if $C(x) \nmid x$ and $C(x) \nmid (x^j+1)$ where $j \le$ max frame len.

⇒ all odd # of bit errors if $(x+1) \mid C(x)$

⇒ all bursts with length $< k$ bits

CMSC 417 Spring 2016 Lecture #19 4/18/2016

2-bit errors proof}

$\Rightarrow E(x) = x^i + x^j$ where $i \neq j$, $j \geq 0$, $i \geq 0$

assume w/o loss of generality $j < i$

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

case 1: $j = 0$

$E(x) = x^i + 1$

if $C(x) \nmid (x^i + 1)$ for $i \leq$ max frame length we're good

case 2: $j \neq 0$

if $C(x) \nmid x$ then $C(x) \mid (x^{i-j} + 1)$ for $C(x) \mid E(x)$

so, we need

$\underbrace{C(x) \nmid x}_{\text{trivially true if } C(x) \text{ has an } x^0 \text{ term}}$ and $\underbrace{C(x) \nmid (x^{i-j} + 1)}_{\text{same as case 1}}$

amazingly simple $C(x)$ have the property that $C(x) \nmid x$ and $C(x) \nmid (x^k + 1)$ for large values of k

e.g.) $x^{15} + x^{14} + 1$ works for $k \leq 2^{15}$

$\Rightarrow$ why was this harder than just having 3+ terms