Error Detection and Correction

David Wetherall
djw@cs.washington.edu

Last Time

- Different media have different properties that affect higher layer protocols.
- We abstract media into a simple model of a link
- To send messages over a link we must frame them
This Lecture

- Error detection and correction
- Focus: How do we detect and correct messages that are garbled during transmission?
- The responsibility for doing this cuts across the different layers

Errors and Redundancy

- Noise can flip some of the bits we receive
  - We must be able to detect when this occurs!
- Basic approach: add redundant data
  - Error detection codes allow errors to be recognized
  - Error correction codes allow errors to be repaired too
Motivating Example

- A simple error detection scheme:
  - Just send two copies. Differences imply errors.

- Question: Can we do any better?
  - With less overhead
  - Catch more kinds of errors
- Answer: Yes – stronger protection with fewer bits
  - But we can’t catch all inadvertent errors, nor malicious ones

- We will look at basic block codes
  - K bits in, N bits out is a (N,K) code
  - Simple, memoryless mapping

Detection vs. Correction

- Two strategies to correct errors:
  - Detect and retransmit, or Automatic Repeat reQuest. (ARQ)
  - Error correcting codes, or Forward Error Correction (FEC)
- Satellites, real-time media tend to use error correction
- Retransmissions typically at higher levels (Network+)

- Question: Which should we choose?
Retransmissions vs. FEC

• The better option depends on the kind of errors and the cost of recovery
• Example: Message with 1000 bits, Prob(bit error) 0.001
  – Case 1: random errors
  – Case 2: bursts of 1000 errors
  – Case 3: real-time application (teleconference)

The Hamming Distance

• Errors must not turn one valid codeword into another valid codeword, or we cannot detect/correct them.
• Hamming distance of a code is the smallest number of bit differences that turn any one codeword into another
  – e.g., code 000 for 0, 111 for 1, Hamming distance is 3
• For code with distance d+1:
  – d errors can be detected, e.g., 001, 010, 110, 101, 011
• For code with distance 2d+1:
  – d errors can be corrected, e.g., 001 000
**Parity**

- Start with n bits and add another so that the total number of 1s is even (even parity)
  - e.g. 0110010 01100101
  - Easy to compute as XOR of all input bits

- Will detect an odd number of bit errors
  - But not an even number
- Does not correct any errors

---

**2D Parity**

- Add parity row/column to array of bits

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Detects all 1, 2, 3 bit errors, and many errors with >3 bits.
- Corrects all 1 bit errors

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0110111 0
Checksums

- Used in Internet protocols (IP, ICMP, TCP, UDP)
- Basic Idea: Add up the data and send it along with sum

Algorithm:
- checksum is the 1s complement of the 1s complement sum of the data interpreted 16 bits at a time (for 16-bit TCP/UDP checksum)
- 1s complement: flip all bits to make number negative
  - Consequence: adding requires carryout to be added back

CRCs (Cyclic Redundancy Check)

- Stronger protection than checksums
  - Used widely in practice, e.g., Ethernet CRC-32
  - Implemented in hardware (XORs and shifts)

Algorithm: Given n bits of data, generate a k bit check sequence that gives a combined n + k bits that are divisible by a chosen divisor C(x)

- Based on mathematics of finite fields
  - “numbers” correspond to polynomials, use modulo arithmetic
  - e.g., interpret 10011010 as $x^7 + x^4 + x^3 + x^1$
How is $C(x)$ Chosen?

- Mathematical properties:
  - All 1-bit errors if non-zero $x^k$ and $x^0$ terms
  - All 2-bit errors if $C(x)$ has a factor with at least three terms
  - Any odd number of errors if $C(x)$ has $(x + 1)$ as a factor
  - Any burst error $< k$ bits

- There are standardized polynomials of different degree that are known to catch many errors
  - Ethernet CRC-32: 100000100110000010001110110110111

Reed-Solomon / BCH Codes

- Developed to protect data on magnetic disks
- Used for CDs and cable modems too
- Property: $2t$ redundant bits can correct $\leq t$ errors
- Mathematics somewhat more involved …
Key Concepts

- Redundant bits are added to messages to protect against transmission errors.
- Two recovery strategies are retransmissions (ARQ) and error correcting codes (FEC)
- The Hamming distance tells us how much error can safely be tolerated.