Transmission Errors

Error Detection and Correction
Transmission Errors

• Transmission errors are caused by:
  – thermal noise \{Shannon\}
  – impulse noise (e..g, arcing relays)
  – signal distortion during transmission (attenuation)
  – crosstalk
  – voice amplitude signal compression (companding)
  – quantization noise (PCM)
  – jitter (variations in signal timings)
  – receiver and transmitter out of synch.
Error Detection and Correction

• *error detection*: adding enough “extra” bits to deduce that there is an error but not enough bits to correct the error.

• If only error detection is employed in a network transmission → retransmission is necessary to recover the frame (data link layer) or the packet (network layer).

• *At the data link layer, this is referred to as ARQ (Automatic Repeat reQuest).*
Error Detection and Correction

- **error correction** :: requires enough additional (redundant) bits to deduce what the correct bits must have been.

**Examples**

Hamming Codes

FEC = Forward Error Correction *found in* MPEG-4.
Hamming Codes

codeword :: a legal dataword consisting of $m$ data bits and $r$ redundant bits.

Error detection involves determining if the received message matches one of the legal codewords.

Hamming distance :: the number of bit positions in which two bit patterns differ.

Starting with a complete list of legal codewords, we need to find the two codewords whose Hamming distance is the smallest. This determines the Hamming distance of the code.
### Error Correcting Codes

<table>
<thead>
<tr>
<th>Char.</th>
<th>ASCII</th>
<th>Check bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1001000</td>
<td>001100100000</td>
</tr>
<tr>
<td>a</td>
<td>1100001</td>
<td>10111001001</td>
</tr>
<tr>
<td>m</td>
<td>1101101</td>
<td>11101010101</td>
</tr>
<tr>
<td>m</td>
<td>1101101</td>
<td>11101010101</td>
</tr>
<tr>
<td>i</td>
<td>1101001</td>
<td>01101011001</td>
</tr>
<tr>
<td>n</td>
<td>1101110</td>
<td>01101010110</td>
</tr>
<tr>
<td>g</td>
<td>1100111</td>
<td>01111001111</td>
</tr>
<tr>
<td>c</td>
<td>0100000</td>
<td>10011000000</td>
</tr>
<tr>
<td>o</td>
<td>1100011</td>
<td>11111000011</td>
</tr>
<tr>
<td>d</td>
<td>1100100</td>
<td>11111001100</td>
</tr>
<tr>
<td>e</td>
<td>1100101</td>
<td>001111000101</td>
</tr>
</tbody>
</table>

**Note**

Check bits occupy power of 2 slots

**Figure 3-7. Use of a Hamming code to correct burst errors.**
(a) A code with poor distance properties

(b) A code with good distance properties

\[ x = \text{codewords} \quad o = \text{non-codewords} \]
Hamming Codes

- To detect $d$ single bit errors, you need a $d+1$ code distance.
- To correct $d$ single bit errors, you need a $2d+1$ code distance.

⇒ In general, the price for redundant bits is too expensive to do error correction for network messages.

⇒ Network protocols use error detection and ARQ.
Error Detection

Remember – errors in network transmissions are bursty.
→ The percentage of damage due to errors is lower.
→ It is harder to detect and correct network errors.

• Linear codes
  – Single parity check code :: take $k$ information bits and append a single check bit to form a codeword.
  – Two-dimensional parity checks

• IP Checksum

• Polynomial Codes
  Example: CRC (Cyclic Redundancy Checking)
General Error-Detection System

All inputs to channel satisfy pattern/condition

Channel output

Deliver user information or set error alarm
Error-Detection System using Check Bits

Information bits

Calculate check bits

Channel

Received information bits

Recalculate check bits

Compare

Information accepted if check bits match
### Two-dimensional parity check code

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

- Last column consists of check bits for each row
- Bottom row consists of check bit for each column
Figure 3.53

Arrows indicate failed check bits

Networks: Transmission Errors

Copyright ©2000 The McGraw Hill Companies
unsigned short cksum(unsigned short *addr, int count) {
    /*Compute Internet Checksum for “count” bytes
     * beginning at location “addr”.
     */
    register long sum = 0;
    while ( count > 1 ) {
        /* This is the inner loop*/
        sum += *addr++;
        count -=2;
    }

    /* Add left-over byte, if any */
    if ( count > 0 )
        sum += *addr;

    /* Fold 32-bit sum to 16 bits */
    while (sum >>16)
        sum = (sum & 0xffff) + (sum >> 16) ;

    return ~sum;
}
Polynomial Codes [LG&W pp. 161-167]

• Used extensively.
• Implemented using **shift-register circuits** for speed advantages.
• Also called CRC (cyclic redundancy checking) because these codes generate check bits.
• **Polynomial codes**: bit strings are treated as representations of polynomials with ONLY binary coefficients (0’s and 1’s).
Polynomial Codes

• The $k$ bits of a message are regarded as the coefficient list for an information polynomial of degree $k-1$.

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_1x + i_0$$

Example:

$$1 0 1 1 0 0 0$$

$$i(x) = x^6 + x^4 + x^3$$
Polynomial Notation

• Encoding process takes $i(x)$ produces a codeword polynomial $b(x)$ that contains information bits and additional check bits that satisfy a pattern.
• Let the codeword have $n$ bits with $k$ information bits and $n-k$ check bits.
• We need a generator polynomial of degree $n-k$ of the form

$$G = g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \ldots + g_1x + 1$$

Note – the first and last coefficient are always 1.
CRC Codeword

- k information bits
- n-k check bits
- n bit codeword
Polynomial Arithmetic

Addition: \( (x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + (1 + 1)x^6 + x^5 + 1 = x^7 + x^5 + 1 \)

Multiplication: \( (x + 1)(x^2 + x + 1) = x^3 + x^2 + x + x^2 + x + 1 = x^3 + 1 \)

Division:

\[
\begin{array}{c|cccc}
\text{dividend} & x^6 & x^4 & x^3 & \\
\hline
\text{divisor} & x^3 + x + 1 & \\
\text{quotient} & x^3 + x^2 + x & \\
\hline
\text{remainder} & x^5 + x^4 + x^3 & \\
\end{array}
\]

\[
\begin{array}{c|cccc}
\text{remainder} & x^5 & x^3 & x^2 & \\
\hline
\text{remainder} & x^4 & x^2 & \\
\text{remainder} & x^4 & x^2 + x & \\
\hline
\text{remainder} & x & \\
\end{array}
\]
CRC Steps:
1) Multiply $i(x)$ by $x^{n-k}$ (puts zeros in $(n-k)$ low order positions)

2) Divide $x^{n-k} \cdot i(x)$ by $g(x)$

$$x^{n-k}i(x) = g(x) \cdot q(x) + r(x)$$

quotient \hspace{1cm} remainder

3) Add remainder $r(x)$ to $x^{n-k} \cdot i(x)$
(puts check bits in the $n-k$ low order positions):

$$b(x) = x^{n-k}i(x) + r(x) \hspace{1cm} \text{transmitted codeword}$$
Information: $(1,1,0,0) \implies i(x) = x^3 + x^2$

Generator polynomial: $g(x) = x^3 + x + 1$

Encoding: $x^3 i(x) = x^6 + x^5$

\[
\begin{array}{c}
\begin{array}{c}
 x^3 + x^2 + x \\
 x^3 + x + 1 \end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
 x^6 + x^5 \\
 x^6 + x^4 + x^3 \\
 x^5 + x^4 + x^3 \\
 x^5 + x^3 + x^2 \\
 x^4 + x^2 \\
 x^4 + x^2 + x \\
 x
\end{array}
\end{array}
\]

Transmitted codeword: $b(x) = x^6 + x^5 + x$

$\implies b = (1,1,0,0,0,1,0)$
Frame: 1101011011
Generator: 10011
Message after 4 zero bits are appended: 11010110110000

\[\begin{array}{c}
\text{Frame} : \quad 1101011011 \\
\text{Generator:} \quad 10011 \\
\text{Message after 4 zero bits are appended:} \quad 11010110110000
\end{array}\]

\[\begin{array}{c}
\text{10011} \\
\text{110101101101100000}
\end{array}\]

\[\begin{array}{c}
\text{10011} \\
\text{10011} \\
\text{10011} \\
\text{000001} \\
\text{000000} \\
\text{00010} \\
\text{00000} \\
\text{00101} \\
\text{00000} \\
\text{01011} \\
\text{00000} \\
\text{10110} \\
\text{10011} \\
\text{01010} \\
\text{00000} \\
\text{10100} \\
\text{10011} \\
\text{01110} \\
\text{00000} \\
\text{1110} \\
\end{array}\]

\text{Remainder}

**Cyclic Redundancy Checking**

Figure 3-8. Calculation of the polynomial code checksum.

Transmitted frame: 11010110111110

Networks: Transmission Errors 22
Generator Polynomial Properties for Detecting Errors

1. Single bit errors: \( e(x) = x^i \quad 0 \leq i \leq n-1 \)

   If \( g(x) \) has more than one term, it cannot divide \( e(x) \)

2. Double bit errors: \( e(x) = x^i + x^j \quad 0 \leq i < j \leq n-1 \)

   \[ e(x) = x^i (1 + x^{j-i}) \]

   If \( g(x) \) is primitive, it will not divide \( (1 + x^{j-i}) \) for \( j-i \leq 2^{n-k}-1 \)

3. Odd number of bit errors: \( e(1) = 1 \) If number of errors is odd.

   If \( g(x) \) has \((x+1)\) as a factor, then \( g(1) = 0 \) and all codewords have an even number of 1s.
Generator Polynomial Properties for Detecting Errors

4. Error bursts of length $b$: \[
\begin{array}{c}
0000110 \cdots 0001101100 \cdots 0 \\
\end{array}
\]

$e(x) = x^i \ d(x)$ \hspace{1cm} \text{where } \deg(d(x)) = L-1

$g(x)$ has degree $n-k$;

$g(x)$ cannot divide $d(x)$ \hspace{1cm} \text{if } \deg(g(x)) > \deg(d(x))$

- $L = (n-k)$ or less: all errors will be detected
- $L = (n-k+1)$: \hspace{1cm} $\deg(d(x)) = \deg(g(x))$
  \hspace{1cm} i.e. $d(x) = g(x)$ is the only undetectable error pattern,
  \hspace{1cm} fraction of bursts which are undetectable \hspace{1cm} $= 1/2^{L-2}$
- $L > (n-k+1)$: \hspace{1cm} fraction of bursts which are undetectable \hspace{1cm} $= 1/2^{n-k}$
Basic ARQ with CRC

Packet sequence

Transmitter

Information frames

Receiver

Error-free packet sequence

Station A

Station B

Information frames

Control frames

Information Frame

Information packet

Header

CRC

Information Frame

Control frame

Header

CRC

Copyright ©2000 The McGraw Hill Companies

Leon-Garcia & Widjaja: *Communication Networks*

Figure 5.8