Course Outline

~ Apr 16  ➔ L7  Application
~ Mar 21  ➔ L4  Transport
~ Feb 26  ➔ L3  Network
~ Feb 15  ➔ L2  Data link
Today  ➔ L1  Physical
Outline for Today

• Today: The Physical Layer

• How to encode data over a link

• How to detect and correct errors
A Brief Overview of Physical Media
Links - Copper

- Copper-based Media
  - Category 3 Twisted Pair
  - Category 5 Twisted Pair
  - ThinNet Coaxial Cable
  - ThickNet Coaxial Cable

- More twists, less crosstalk, better signal over longer distances
  - 10-100Mbps 100m
  - 10-100Mbps 200m
  - 10-100Mbps 500m

- More expensive than twisted pair
- High bandwidth and excellent noise immunity
Links - Optical

- Optical Media
  - Multimode Fiber 100Mbps 2km
  - Single Mode Fiber 100-2400Mbps 40km
Links - Optical

- Single mode fiber
  - Expensive to drive (Lasers)
  - Lower attenuation (longer distances) \( \leq 0.5 \text{ dB/km} \)
  - Lower dispersion (higher data rates)

- Multimode fiber
  - Cheap to drive (LED’s)
  - Higher attenuation
  - Easier to terminate

---

Core of single mode fiber

\( \approx 1 \text{ wavelength thick} = \approx 1 \text{ micron} \)

Core of multimode fiber (same frequency; colors for clarity)

\( \text{O}(100 \text{ microns}) \text{ thick} \)
Encoding
How can two hosts communicate?

- Encode data as variations in electrical/light/EM
  - Phase, frequency, and signal strength modulation, and combinations thereof
- Simple scheme: voltage encoding
  - Encode 1’s and 0’s as variations in voltage
  - How to do that?
Non-Return to Zero (NRZ)

- **Signal to Data**
  - High $\Rightarrow$ 1
  - Low $\Rightarrow$ 0

- **Comments**
  - Transitions maintain clock synchronization
  - Long strings of 0s confused with no signal
  - Long strings of 1s causes baseline wander
  - Both inhibit clock recovery

```
Bits:     0  0  1  0  1  1  1  1  0  1  0  0  0  0  1  0
NRZ:  
```
Non-Return to Zero Inverted (NRZI)

- **Signal to Data**
  - Transition \( \Rightarrow \) 1
  - Maintain \( \Rightarrow \) 0

- **Comments**
  - Solves series of 1s, but not 0s

![NRZ and NRZI waveforms](image-url)
Manchester Encoding

- **Signal to Data**
  - XOR NRZ data with clock
  - High to low transition $\Rightarrow 1$
  - Low to high transition $\Rightarrow 0$

- **Comments**
  - Used by old 10Mbps Ethernet
  - Solves clock recovery problem
  - Only 50% efficient (½ bit per transition)
4B/5B

- **Signal to Data**
  - Encode every 4 consecutive bits as a 5 bit symbol

- **Symbols**
  - At most 1 leading 0
  - At most 2 trailing 0s
  - Never more than 3 consecutive 0s
  - Transmit with NRZI

- **Comments**
  - 16 of 32 possible codes used for data
  - At least two transitions for each code
  - 80% efficient
  - Used by old 100Mbps Ethernet
  - Variation (64B/66B) used by modern 10Gbps Ethernet
### 4B/5B – Data Symbols

**At most 1 leading 0**

- \(0000 \Rightarrow 11110\)
- \(0001 \Rightarrow 01001\)
- \(0010 \Rightarrow 10100\)
- \(0011 \Rightarrow 10101\)
- \(0100 \Rightarrow 01010\)
- \(0101 \Rightarrow 01011\)
- \(0110 \Rightarrow 01110\)
- \(0111 \Rightarrow 01111\)

**At most 2 trailing 0s**

- \(1000 \Rightarrow 10010\)
- \(1001 \Rightarrow 10011\)
- \(1010 \Rightarrow 10110\)
- \(1011 \Rightarrow 10111\)
- \(1100 \Rightarrow 11010\)
- \(1101 \Rightarrow 11011\)
- \(1110 \Rightarrow 11100\)
- \(1111 \Rightarrow 11101\)
4B/5B – Control Symbols

- 11111 ⇒ idle
- 11000 ⇒ start of stream 1
- 10001 ⇒ start of stream 2
- 01101 ⇒ end of stream 1
- 00111 ⇒ end of stream 2
- 00100 ⇒ transmit error
- Other ⇒ invalid
Binary Voltage Encodings

• Problem with binary voltage (square wave) encodings
  • Wide frequency range required, implying
    • Significant dispersion
    • Uneven attenuation
  • Prefer to use narrow frequency band (carrier frequency)

• Types of modulation
  • Amplitude (AM)
  • Frequency (FM)
  • Phase/phase shift
  • Combinations of these
  • Used in wireless Ethernet, optical communications
Example:
AM/FM for continuous signal

- Original signal
- Amplitude modulation
- Frequency modulation
Amplitude Modulation

idle 1 0
Frequency Modulation
Phase Modulation
Phase Modulation

- Phase shift in carrier frequency
- $180^\circ$ difference in phase
- Collapse for $180^\circ$ shift
Phase Modulation Algorithm

- Send carrier frequency for one period
  - Perform phase shift
  - Shift value encodes symbol
    - Value in range [0, 360º)
    - Multiple values for multiple symbols
    - Represent as circle

8-symbol example
You can combine modulation schemes

Example: QAM (Quadrature Amplitude Modulation)

For a given symbol:
- Perform phase shift and change to new amplitude

2-dimensional representation:
- Angle is phase shift
- Radial distance is new amplitude
QAM: Example transmission
Real constellation with noise
Suppose you have the following 1Hz signal being received.

How fast to sample, to capture the signal?
Sampling

- Sampling a 1 Hz signal at 2 Hz is enough
  - Captures every peak and trough
Sampling

- Sampling a 1 Hz signal at 3 Hz is also enough
  - In fact, more than enough samples to capture variation in signal
Sampling

• Sampling a 1 Hz signal at 1.5 Hz is not enough
  • Why?
• Sampling a 1 Hz signal at 1.5 Hz is not enough
  • Not enough samples, can’t distinguish between multiple possible signals
In general

• Sampling a 1 Hz signal at 2 Hz is both necessary and sufficient

• In general: sampling twice rate of signal is enough
What about more complex signals?

- Fourier’s theorem: any continuous signal can be decomposed into a sum of sines and cosines at different frequencies
- Example: Sum of 1 Hz, 2 Hz, and 3 Hz sines
  - How fast to sample?
What about more complex signals?

- Fourier’s theorem: any continuous signal can be decomposed into a sum of sines and cosines at different frequencies
- Example: Sum of 1 Hz, 2 Hz, and 3 Hz sines
  - How fast to sample?
  - Answer: Twice rate of fastest signal (bandwidth): 6 Hz
Nyquist–Shannon sampling theorem

• If a function \( x(t) \) contains no frequencies higher than \( B \) hertz, it is completely determined by giving its ordinates at a series of points spaced \( 1/(2B) \) seconds apart

• In other words:
  • If the bandwidth of your channel is \( B \)
  • Your sampling rate should be \( 2B \)
  • Higher sampling rates are pointless
  • Lower sampling rates lead to aliasing/distortion/error
Related Question: How much data can you pack into a channel?

• If I sample at a rate of 2B, I can precisely determine the signal of bandwidth B

• If I have data coming in at rate 2B, I can encode it in a channel of rate B
  • Similar argument to above, but in reverse
  • Instead of “reading” a sample, we “write” a sample

• More generally:
  • Transmitting N distinct signals over a noiseless channel with bandwidth B, we can achieve at most a data rate of
  • 2B log2 N
Noiseless Capacity

• Nyquist’s theorem: $2B \log_2 N$

• Example 1: sampling rate of a phone line
  • $B = 4000$ Hz
  • $2B = 8000$ samples/sec.
    • sample every 125 microseconds

• Example 2: noiseless capacity
  • $B = 1200$ Hz
  • $N = \text{each pulse encodes 16 levels}$
  • $C = 2B \log_2 (N) = D \times \log_2 (N)$
    = $2400 \times 4 = 9600$ bps.
What can Limit Maximum Data Rate?

- **Noise**
  - E.g., thermal noise (in-band noise) can blur symbols

- **Transitions between symbols**
  - Introduce high-frequency components into the transmitted signal
  - Such components cannot be recovered (by Nyquist’s Theorem), and some information is lost

- **Examples**
  - **Phase modulation**
    - Single frequency (with different phases) for each symbol
    - Transitions can require very high frequencies
How does Noise affect these Bounds?

- In-band (thermal, not high-frequency) noise
  - Blurs the symbols, reducing the number of symbols that can be reliably distinguished.

- Claude Shannon (1948)
  - Extended Nyquist’s work to channels with additive white Gaussian noise (a good model for thermal noise)

  channel capacity \( C = B \log_2 (1 + S/N) \)

  - \( B \) is the channel bandwidth
  - \( S/N \) is the ratio between
    - the average signal power and
    - the average in-band noise power
Noisy Capacity

- Telephone channel
  - 3400 Hz at 40 dB SNR
  - $C = B \log_2 (1+S/N)$ bits/s
  - SNR = 40 dB
    \[ 40 = 10 \log_{10} (S/N) \]
    \[ S/N = 10,000 \]
  - $C = 3400 \log_2 (10001) = 44.8$ kbps

\[
\text{SNR (dB)} = 10 \log_{10} \left( \frac{P_{\text{signal}}}{P_{\text{noise}}} \right)
\]
Summary of Encoding

• Problems
  • Attenuation, dispersion, noise
• Digital transmission allows periodic regeneration
• Variety of binary voltage encodings
  • High frequency components limit to short range
  • More voltage levels provide higher data rate
• Carrier frequency and modulation
  • Amplitude, frequency, phase, and combinations
  • Quadrature amplitude modulation: amplitude and phase, many signals
• Nyquist (noiseless) and Shannon (noisy) limits on data rates
Error
Detection/Correction
Error Detection

- **Encoding** translates symbols to signals
- **Framing** demarcates units of transfer
- **Error detection** validates correctness of each frame
Error Detection

• Key idea: Add redundant information that can be used to determine if errors have been introduced, and potentially fix them

• Errors checked at many levels
  • Demodulation of signals into symbols (analog)
  • Bit error detection/correction (digital)—our main focus
    • Within network adapter (CRC check)
    • Within IP layer (IP checksum)
    • Possibly within application as well
Error Detection

• Analog Errors
  • Example of signal distortion

• Hamming distance
  • Parity and voting
  • Hamming codes

• Error bits or error bursts?

• Digital error detection
  • Two-dimensional parity
  • Checksums
  • Cyclic Redundancy Check (CRC)
Analog Errors

- Consider RS-232 encoding of character ‘Q’
  - ASCII Q = 1100001

- Assume idle wire (-15V) before and after signal
RS-232 Encoding of 'Q'

![Diagram showing RS-232 encoding for 'Q' with voltage levels and start-stop sequence.](image-url)
Limited-Frequency Signal Response
(bandwidth = baud rate)
Limited-Frequency Signal Response
(bandwidth = baud rate/2)
Symbols

possible binary voltage encoding
symbol neighborhoods and erasure
region

possible QAM symbol
neighborhoods in green; all
other space results in erasure
Symbols

- Inputs to digital level
  - valid symbols
  - erasures

- Hamming distance
  - Definition
  - 1-bit error-detection with parity
  - 1-bit error-correction with voting
  - 2-bit erasure-correction with voting
  - Hamming codes (1-bit error correction)
Hamming Distance

• The Hamming distance between two code words is the minimum number of bit flips to move from one to the other
  • Example:
  • 00101 and 00010
  • Hamming distance of 3
Detecting bit flips with Parity

• 1-bit error detection with parity
  • Add an extra bit to a code to ensure an even (odd) number of 1s
  • Every code word has an even (odd) number of 1s
Correcting bit flips with Voting

• 1-bit error correction with voting
  • Every codeword is transmitted n times
2-bit **Erasure** Correction with Voting

- Every code word is copied 3 times

2-erasure planes in green remaining bit not ambiguous cannot correct 1-error and 1-erasure
Minimum Hamming Distance

• The minimum Hamming distance of a code is the minimum distance over all pairs of codewords
  • Minimum Hamming Distance for parity
    • 2
  • Minimum Hamming Distance for voting
    • 3
Coverage

• N-bit error detection
  • No code word changed into another code word
  • Requires Hamming distance of N+1

• N-bit error correction
  • N-bit neighborhood: all codewords within N bit flips
  • No overlap between N-bit neighborhoods
  • Requires hamming distance of 2N+1
Hamming Codes

- Linear error-correcting code, Named after Richard Hamming
  - Simple, commonly used in RAM (e.g., ECC-RAM)
- Can detect up to 2 simultaneous bit errors
- Can correct single-bit errors
- Construction
  - number bits from 1 upward
  - powers of 2 are check bits
  - all others are data bits
  - Check bit $j$ is XOR of all bits $k$ such that $(j \text{ AND } k) = j$
- Example: 4 bits of data, 3 check bits

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
C & C & D & C & D & D & D & C \\
\end{array}
\]
Hamming Codes

\[
\begin{align*}
C_1 &= D_3 \oplus D_5 \oplus D_7 \\
C_2 &= D_3 \oplus D_6 \oplus D_7 \\
C_4 &= D_5 \oplus D_6 \oplus D_7
\end{align*}
\]
Hamming Codes
Error Bits or Bursts?

• Common model of errors
  • Probability of error per bit
  • Error in each bit independent of others
  • Value of incorrect bit independent of others

• Burst model
  • Probability of back-to-back bit errors
  • Error probability dependent on adjacent bits
  • Value of errors may have structure

• Why assume bursts?
  • Appropriate for some media (e.g., radio)
  • Faster signaling rate enhances such phenomena
Digital Error Detection Techniques

- Two-dimensional parity
  - Detects up to 3-bit errors
  - Good for burst errors

- IP checksum
  - Simple addition
  - Simple in software
  - Used as backup to CRC

- Cyclic Redundancy Check (CRC)
  - Powerful mathematics
  - Tricky in software, simple in hardware
  - Used in network adapter
Two-Dimensional Parity

- Use 1-dimensional parity
  - Add one bit to a 7-bit code to ensure an even/odd number of 1s

- Add 2nd dimension
  - Add an extra byte to frame
    - Bits are set to ensure even/odd number of 1s in that position across all bytes in frame

- Comments
  - Can detect and correct any 1-bit error
  - Can detect any 1-, 2- and 3-bit, and most 4-bit errors
Two-Dimensional Parity

\[
\begin{array}{cccccccc}
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]
What happens if...

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

Can detect exactly which bit flipped
Can also correct it!
What happens if...

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

Can detect the two-bit error, but can’t tell which bits are flipped, so can’t correct

No longer a problem here
What happens if…

Suppose these four parity bits don’t match. Which bits could be in error?

Could be the blue pair, OR, could be the orange pair. So, can’t correct.
What about 3-bit errors?

Can detect exactly which bit flipped
You can correct in this case
What about 3-bit errors?

Can detect exactly which bit flipped
But you can’t correct (eg if orange bits got flipped instead of the blue ones)

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
What about 4-bit errors?

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

Are there any 4-bit errors this scheme *can* detect?

Can you think of a 4-bit error this scheme can’t detect?
Internet Checksum

- Idea: Add up all the words, transmit the sum
- Internet Checksum
  - Use 1’s complement addition on 16bit codewords
  - Example
    - Codewords: -5  -3
    - 1’s complement binary: 1010 1100
    - 1’s complement sum 1000

- Comments
  - Small number of redundant bits
  - Easy to implement
  - Not very robust
IP Checksum

u_short cksum(u_short *buf, int count) {
    register u_long sum = 0;
    while (count--) {
        sum += *buf++;
        if (sum & 0xFFFF0000) {
            /* carry occurred, so wrap around */
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(sum & 0xFFFF);  
}
Cyclic Redundancy Check (CRC)

• Non-secure hash function based on cyclic codes

• Idea
  • Add $k$ bits of redundant data to an $n$-bit message
  • $N$-bit message is represented as a $n$-degree polynomial with each bit in the message being the corresponding coefficient in the polynomial

• Example
  • Message = 10011010
  • Polynomial
    \[= 1 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 0\]
    \[= x^7 + x^4 + x^3 + x\]
Overly simplified CRC-like protocol, using regular numbers

- Both endpoints agree in advance on a divisor value \( C=3 \)
- Sender wants to send a message \( M=10 \)
- Sender computes a value \( P=M+X=10+2=12 \) that is evenly divisible by \( C \)
- Sender sends \( P \) and \( M \) to receiver
- Receiver checks to make sure \( P=12 \) is evenly divisible by \( C=3 \)
  - If it is not, then there’s error(s)
  - If it is, then there are probably no errors

- CRC is vaguely like this, but uses polynomials instead of numbers
  - CRC can reconstruct \( M \) from \( P \) and \( C \), so just needs to send \( P \)
CRC Approach

1. Given
   - Message $M(x) = 10011010$
   - Represented as $x^7 + x^4 + x^3 + x$

2. Select a divisor polynomial $C(x)$ with degree $k$
   - Example with $k = 3$:
     - $C(x) = x^3 + x^2 + 1$
     - Represented as 1101

2. Transmit a polynomial $P(x)$ that is evenly divisible by $C(x)$
   - $P(x) = M(x) + k$ bits

How can we determine these $k$ bits?
Properties of Polynomial Arithmetic

• Divisor
  • Any polynomial $B(x)$ can be divided by a polynomial $C(x)$ if $B(x)$ is of the same or higher degree than $C(x)$

• Remainder
  • The remainder obtained when $B(x)$ is divided by $C(x)$ is obtained by subtracting $C(x)$ from $B(x)$

• Subtraction
  • To subtract $C(x)$ from $B(x)$, simply perform an XOR on each pair of matching coefficients

• For example: $(x^3+1)/(x^3+x^2+1) = ?$
CRC - Sender

- **Given**
  - \( M(x) = 10011010 = x^7 + x^4 + x^3 + x \)
  - \( C(x) = 1101 = x^3 + x^2 + 1 \)

- **Steps**
  - \( T(x) = M(x) \times x^k \) (add zeros to increase degree of \( M(x) \) by \( k \))
  - Find remainder, \( R(x) \), from \( T(x)/C(x) \)
  - \( P(x) = T(x) - R(x) \Rightarrow M(x) \) followed by \( R(x) \)

- **Example**
  - \( T(x) = 10011010000 \)
  - \( R(x) = 101 \)
  - \( P(x) = 10011010101 \)
CRC - Receiver

- Receive Polynomial $P(x) + E(x)$
  - $E(x)$ represents errors
  - (if no errors then $E(x) = 0$)
- Divide $(P(x) + E(x))$ by $C(x)$
  - If result = 0, either
    - No errors ($E(x) = 0$, and $P(x)$ is evenly divisible by $C(x)$)
    - $(P(x) + E(x))$ is exactly divisible by $C(x)$, error will not be detected
CRC – Example Encoding

\[ C(x) = x^3 + x^2 + 1 \]
\[ M(x) = x^7 + x^4 + x^3 + x \]

\[ C(x) = 1101 \]
\[ M(x) = 10011010 \]

Generator
Message

Message plus \( k \) zeros

Result:

Transmit message followed by remainder:

\[ 10011010101 \]
CRC – Example Decoding – No Errors

\[ C(x) = x^3 + x^2 + 1 = 1101 \] \hspace{2cm} \text{Generator}

\[ P(x) = x^{10} + x^7 + x^6 + x^4 + x^2 + 1 = 10011010101 \] \hspace{2cm} \text{Received Message}

\[ 1101 \]

\[ 10011010101 \]

\[ 1101 \]

\[ 1001 \]

\[ 1101 \]

\[ 1000 \]

\[ 1101 \]

\[ 1011 \]

\[ 1101 \]

\[ 1100 \]

\[ 1101 \]

\[ 1101 \]

\[ 1101 \]

\[ 1101 \]

\[ 0 \]

Result:
CRC test is passed
CRC – Example Decoding – with Errors

\[ C(x) = x^3 + x^2 + 1 = 1101 \]
\[ P(x) = x^{10} + x^7 + x^5 + x^4 + x^2 + 1 = 10010110101 \]

Result:
CRC test failed
CRC Error Detection

• Properties
  • Characterize error as $E(x)$
  • Error detected unless $C(x)$ divides $E(x)$
    • ($i.e.$, $E(x)$ is a multiple of $C(x)$)
Example of Polynomial Multiplication

- Multiply
  - $1101$ by $10110$
  - $x^3 + x^2 + 1$ by $x^4 + x^2 + x$

This is a multiple of $c$, so that if errors occur according to this sequence, the CRC test would be passed.
On Polynomial Arithmetic

• Polynomial arithmetic
  • A fancy way to think about addition with no carries.
  • Helps in the determination of a good choice of C(x)
  • A non-zero vector is not detected if and only if the error polynomial E(x) is a multiple of C(x)

• Implication
  • Suppose C(x) has the property that C(1) = 0 (i.e. (x + 1) is a factor of C(x))
  • If E(x) corresponds to an undetected error pattern, then it must be that E(1) = 0
  • Therefore, any error pattern with an odd number of error bits is detected
CRC Error Detection

- What errors can we detect?
  - All single-bit errors, if \( x^k \) and \( x^0 \) have non-zero coefficients
  - All double-bit errors, if \( C(x) \) has at least three terms
  - All odd bit errors, if \( C(x) \) contains the factor \( (x + 1) \)
  - Any bursts of length < \( k \), if \( C(x) \) includes a constant term
  - Most bursts of length \( \geq k \)
## Common Polynomials for C(x)

<table>
<thead>
<tr>
<th>CRC</th>
<th>C(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC-8</td>
<td>$x^8 + x^2 + x^1 + 1$</td>
</tr>
<tr>
<td>CRC-10</td>
<td>$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$</td>
</tr>
<tr>
<td>CRC-12</td>
<td>$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$</td>
</tr>
<tr>
<td>CRC-16</td>
<td>$x^{16} + x^{15} + x^2 + 1$</td>
</tr>
<tr>
<td>CRC-CCITT</td>
<td>$x^{16} + x^{12} + x^5 + 1$</td>
</tr>
<tr>
<td>CRC-32</td>
<td>$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$</td>
</tr>
</tbody>
</table>