The Data Link Layer
Chapter 3

- Data Link Layer Design Issues
- Error Detection and Correction
- Elementary Data Link Protocols
- Sliding Window Protocols
- Example Data Link Protocols

Revised: August 2011
The Data Link Layer

Responsible for delivering frames of information over a single link

- Handles transmission errors and regulates the flow of data
Data Link Layer Design Issues

- Frames
- Possible services
- Framing methods
- Error control
- Flow control
Error Detection and Correction

Error codes add structured redundancy to data so errors can be either detected, or corrected.

Error correction codes:
- Hamming codes
- Binary convolutional codes
- Reed-Solomon and Low-Density Parity Check codes
  - Mathematically complex, widely used in real systems

Error detection codes:
- Parity
- Checksums
- Cyclic redundancy codes
Error Bounds – Hamming distance

Code turns data of \( n \) bits into codewords of \( n+k \) bits

Hamming distance is the minimum bit flips to turn one valid codeword into any other valid one.

- Example with 4 codewords of 10 bits \((n=2, k=8)\):
  - 0000000000, 0000011111, 1111100000, and 1111111111
  - Hamming distance is 5

Bounds for a code with distance:

- \( 2d+1 \) – can correct \( d \) errors (e.g., 2 errors above)
- \( d+1 \) – can detect \( d \) errors (e.g., 4 errors above)
Error Correction – Hamming code

Hamming code gives a simple way to add check bits and correct up to a single bit error:

- Check bits are parity over subsets of the codeword
- Recomputing the parity sums (syndrome) gives the position of the error to flip, or 0 if there is no error

(11, 7) Hamming code adds 4 check bits and can correct 1 error
Error Correction – Convolutional codes

Operates on a stream of bits, keeping internal state

- Output stream is a function of all preceding input bits
- Bits are decoded with the Viterbi algorithm

Popular NASA binary convolutional code (rate = $\frac{1}{2}$) used in 802.11
Error Detection – Parity (1)

Parity bit is added as the modulo 2 sum of data bits
- Equivalent to XOR; this is even parity
- Ex: 1110000 → 11100001
- Detection checks if the sum is wrong (an error)

Simple way to detect an *odd* number of errors
- Ex: 1 error, 11100101; detected, sum is wrong
- Ex: 3 errors, 11011001; detected sum is wrong
- Ex: 2 errors, 11101101; *not detected*, sum is right!
- Error can also be in the parity bit itself
- Random errors are detected with probability $\frac{1}{2}$
Error Detection – Parity (2)

Interleaving of N parity bits detects burst errors up to N
• Each parity sum is made over non-adjacent bits
• An even burst of up to N errors will not cause it to fail
Error Detection – Checksums

Checksum treats data as N-bit words and adds N check bits that are the modulo $2^N$ sum of the words

- Ex: Internet 16-bit 1s complement checksum

Properties:
- Improved error detection over parity bits
- Detects bursts up to N errors
- Detects random errors with probability $1-2^N$
- Vulnerable to systematic errors, e.g., added zeros
Error Detection – CRCs (1)

Adds bits so that transmitted frame viewed as a polynomial is evenly divisible by a generator polynomial.

Start by adding 0s to frame and try dividing.

Offset by any reminder to make it evenly divisible.
Error Detection – CRCs (2)

Based on standard polynomials:
- Ex: Ethernet 32-bit CRC is defined by:
  \[ x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1 \]
- Computed with simple shift/XOR circuits

Stronger detection than checksums:
- E.g., can detect all double bit errors
- Not vulnerable to systematic errors
End