Bit Stream Encoding

• Specify how bits are represented in the analog signal
• This service is provided by the physical layer
• Challenges:
  -- Efficiency: ideally, bit rate is maximized
  -- Robust: avoid de-synchronization between sender and receiver when there is a large sequence of 1’s or 0’s

Assumptions

• We use two discrete signals, high and low, to encode 1 and 0
• The transmission is synchronous, i.e., there is a clock used to sample the signal
• If the amplitude and duration of the signals is large enough, the receiver can do a reasonable job of looking at the distorted signal and estimating what was sent.
Non-Return to Zero (NRZ)

- 1 → high signal; 0 → low signal
- Disadvantages: when there is a long sequence of 1's or 0's
  - Sensitive to clock skew, i.e., difficult to do clock recovery

Non-Return to Zero Inverted (NRZI)

- 1 → make transition; 0 → stay at the same level
- Solve previous problems for long sequences of 1's, but not for 0's

Manchester

- 1 → high-to-low transition; 0 → low-to-high transition
- Addresses clock recovery problems
- Disadvantage: signal transition rate doubled
  - i.e. useful data rate on same physical medium halved
  - Efficiency of 50%
4-bit/5-bit (100Mb/s Ethernet)

- Goal: address inefficiency of Manchester encoding, while avoiding long periods of low signals
- Solution:
  - Use 5 bits to encode every sequence of four bits such that no 5 bit code has more than one leading 0 and two trailing 0’s
  - Use NRZI to encode the 5 bit codes
  - Efficiency is 80%

<table>
<thead>
<tr>
<th>4-bit</th>
<th>5-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>11110</td>
</tr>
<tr>
<td>0001</td>
<td>01001</td>
</tr>
<tr>
<td>0010</td>
<td>10110</td>
</tr>
<tr>
<td>0011</td>
<td>10111</td>
</tr>
<tr>
<td>0100</td>
<td>01010</td>
</tr>
<tr>
<td>0101</td>
<td>01011</td>
</tr>
<tr>
<td>0110</td>
<td>01100</td>
</tr>
<tr>
<td>0111</td>
<td>01101</td>
</tr>
<tr>
<td>1000</td>
<td>10010</td>
</tr>
<tr>
<td>1001</td>
<td>10011</td>
</tr>
<tr>
<td>1010</td>
<td>10110</td>
</tr>
<tr>
<td>1011</td>
<td>10111</td>
</tr>
<tr>
<td>1100</td>
<td>11010</td>
</tr>
<tr>
<td>1101</td>
<td>11011</td>
</tr>
<tr>
<td>1110</td>
<td>11100</td>
</tr>
<tr>
<td>1111</td>
<td>11101</td>
</tr>
</tbody>
</table>

Framing

- Specify how blocks of data are transmitted between two nodes connected on the same physical media
  - This service is provided by the data link layer
- Challenges
  - Decide when a frame starts/ends
  - If use special delimiters, differentiate between the true frame delimiters and delimiters appearing in the payload data

Byte-Oriented Protocols: Sentinel Approach

- STX – start of text
- ETX – end of text
- Problem: what if ETX appears in the data portion of the frame?
- Solution
  - If ETX appears in the data, introduce a special character DLE (Data Link Escape) before it
  - If DLE appears in the text, introduce another DLE character before it
  - Like in C programming, “Say "Hello!"", (i is the escape character)
Byte-Oriented Protocols: Byte Counting Approach

• Sender: insert the length of the data (in bytes) at the beginning of the frame, i.e., in the frame header
• Receiver: extract this length and decrement it every time a byte is read. When this counter becomes zero, we are done

Bit-Oriented Protocols

• Both start and end sequence can be the same
  – E.g., 01111110 in HDLC (High-level Data Link Protocol)
• Sender: in data portion inserts a 0 after five consecutive 1s
  – “Bit stuffing”
• Receiver: when it sees five 1s makes decision on the next two bits
  – If next bit 0 (this is a stuffed bit), remove it
  – If next bit 1, look at the next bit
    • If 0 this is end-of-frame (receiver has seen 01111110)
    • If 1 this is an error, discard the frame (receiver has seen 01111111)

Clock-Based Framing (SONET)

• SONET (Synchronous Optical NETwork)
• Developed to transmit data over optical links
  – Example: SONET STS-1: 51.84 Mbps
• SONET maintains clock synchronization across several adjacent links to form a path
**STS-1 Frame**

- First two bytes of each frame contain a special bit pattern that allows to determine where the frame starts
- No bit-stuffing is used, frame is fixed size
- Receiver looks for the special bit pattern every 810 bytes
  - Size of frame = 9x90 = 810 bytes

**Clock-Based Framing (SONET)**

- Details:
  - Bits are encoded using NRZ
  - To avoid long sequences of 0's or 1's the payload is XOR-ed with a special 127-bit pattern with many transitions from 1 to 0

**Error detection**

- How to determine if errors (via noise) were introduced?

- Could send 2 copies of data
  - Has poor efficiency
  - Poor protection against errors

- Will discuss three approaches
  - Two-dimensional parity
  - Checksum
  - CRCs
Two-dimensional parity

- Add extra bits to keep number of 1s even
  - Add parity bits and parity bytes

```
0101001
1101001
1011110
0001110
0110100
1011111
1111011
```
Parity bit for each 7 bits
Parity byte for each frame

- Can detect all 1-, 2-, and 3-bit errors!
  - But with at least 14% overhead

Checksums

- Simple: add up bytes of messages, include the sum
  - Hence check-sum

- View data as series of unsigned 16-bit integers
  - Use ones-complement arithmetic

- Much lower overhead (16 bits/frame)
- But, not resilient to errors
  - Why? Error which increments/decrements any two ints
- Used in UDP, TCP, and IP, though

CRCs

- Cyclic redundancy check (CRC)
- Addresses limitations of prior approaches
  - Uses field theory

- Much better performance
  - Fixed overhead per frame
  - Only 1 in $2^{32}$ chance of missed error with 32-bit CRC

- Details in the book, if you're curious