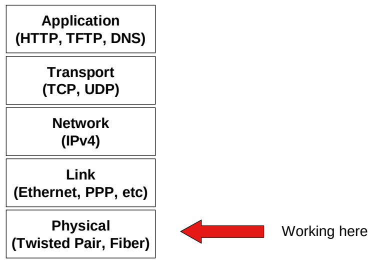
COMP312-09A
Communications and Systems Software

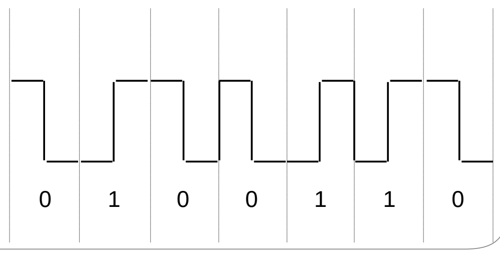
Lecture 13 – Digital Data, Digital Signals

Overview



Overview

- Today, we're looking at methods to encode a series of bits into voltage signals
- COMP202: Manchester encoding

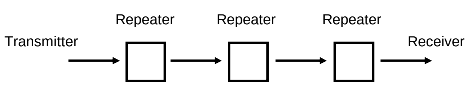


Some evaluation criteria

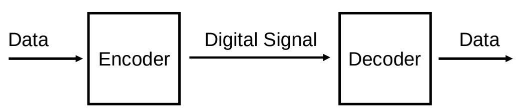
- How efficient is the technique?
  - How many elements are required to signal one bit?
- How immune to noise is the technique?
  - Can the message be decoded in the presence of some background interference?
- What happens if a long series of zeros or ones are transmitted?
  - Does the signal require precise, synchronised, but expensive clocks at each end of the link?
- Can any errors be detected without additional techniques?
- Is the signal DC-balanced?
  - Will use the signalling method result in charging the receiver?

Digital Signals

- Discrete, discontinuous signals
  - usually indicated with a voltage change
- We like digital signals as they fit well with computers
  - zeros and ones
- They are also good for transmitting over long distances
  - signal is decoded and then transmitted, removing most of the effect of any interference
  - compared with analouge amplifier which amplifies noise and all

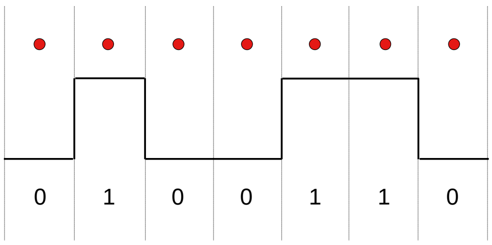


Digital Signals



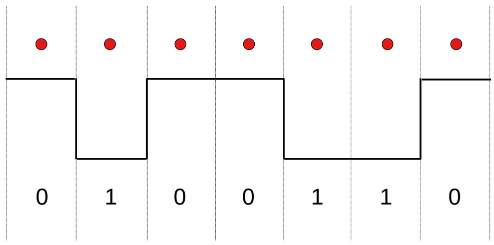
- Data is already in digital form
- Encoder and decoder logic not terribly complicated

Non return to zero (NRZ)



Zero: low voltage
One: high voltage

Non return to zero level



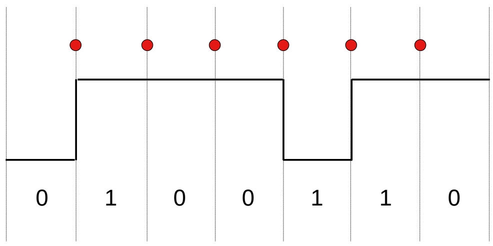
More common in practice

Zero: high voltage
One: low voltage

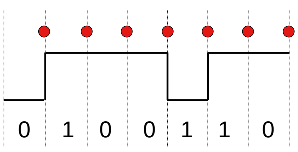
NRZ evaluation

- Bandwidth efficient
  - Each bit requires one signalling element
- Synchronisation problems
  - long sequences of zeroes and ones are difficult to recover
- Simple to implement
- Poor noise immunity
  - need to be able to measure voltage level accurately to determine signal
  - noise can make voltage level ambiguous

Non return to zero invert on ones (NRZI)



NRZI



- Differential encoding
  - changes in voltage level are more robust to decode than voltage levels themselves
- Used in USB with ‘bit stuffing’

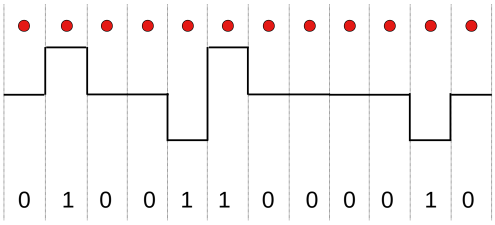
NRZI evaluation

- Bandwidth efficient
- Synchronisation problems for long periods of zeros
- Simple to implement
- Better noise immunity
  - differential encoding

Multilevel schemes

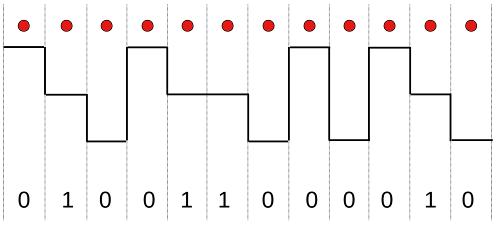
- Instead of using two voltage levels, use three

Bipolar Alternate Mark Inversion



Mark is a '1' bit.
use alternating high and low voltages to signal one.
use medium voltage to signal zero.

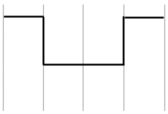
Pseudoternary



Mark is a '0' bit.
use alternating high and low voltages to signal zero.
use medium voltage to signal one.

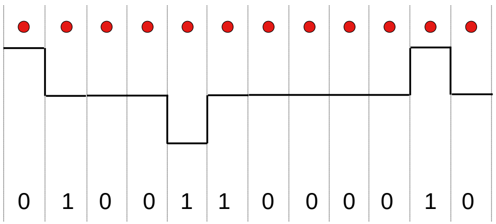
Multilevel scheme evaluation

- Bandwidth efficient
- More complicated to implement than NRZ
- Can detect some errors, e.g:



- Synchronisation on mark bits, but not non-mark bits

Multilevel Transmit (MLT-3)



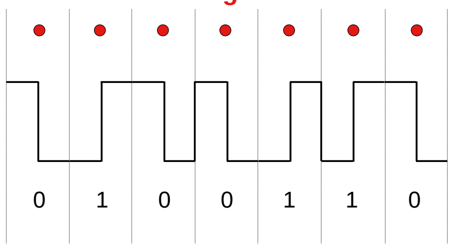
Mark: '1'. Change voltage on mark bit.
Cycle through voltages on each mark bit, rather than alternate

Used as part of 100Mbps Ethernet on UTP (with 4B5B, coming soon)

Symbol rate

- Previous schemes have had a 1:1 mapping of bit to symbol emitted
- Not good at keeping clock synchronisation
- What if we increased the ratio of symbols per bit?
- **Baud**: unit of symbol rate

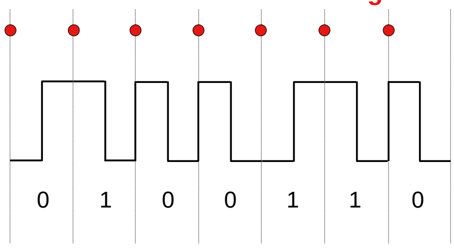
Manchester encoding



Used in 10Mbps Ethernet over Unshielded Twisted Pair (UTP)
High to low: zero
Low to high: one

Two symbols per bit. Less bandwidth efficient.
Good clocking characteristics

Differential Manchester encoding



Used in Token Ring Ethernet
At least one voltage transition per bit
zero: transition at beginning of interval
one: no transition

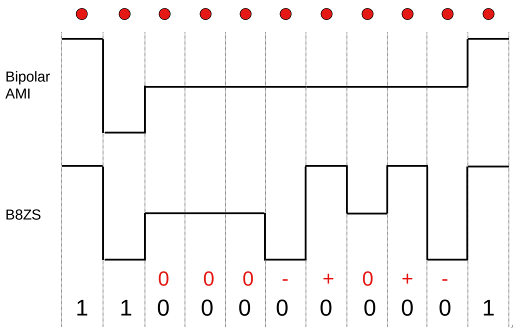
Scrambling

- Manchester is a biphase scheme
  - bandwidth inefficient
- Goal is to remove long strings of zeroes and ones
  - to provide good clocking

Scrambling

- NRZ Codes are scrambled by using a mathematical transformation to produce a random looking bit stream with many transitions.
  - The receiver reverses the transformation to produce the original data stream.
- Multilevel schemes can be scrambled by replacing long sequences of non-marks with defined patterns using polarity violations.
- Example: B8ZS based on Bipolar-AMI with replacement of strings of eight zeroes
  - Following a positive mark use 000+-0-+
  - Following a negative mark use 000-+0+-

Scrambling



Scrambling evaluation

- Excellent bandwidth characteristics (1:1)
- Good clocking
- Poor noise immunity
- More complicated to implement

100Mbps Ethernet: 4B5B

- 10Mbps Ethernet uses Manchester encoding:
  - Requires twice as much bandwidth as the original signal
  - That is, 10Mbps Ethernet requires 20Mhz clock
- 100Mbps Ethernet uses 4B5B:
  - Encodes a set of 4 bits using a 5 bit code defined in dictionary ...
  - ... designed to provide clock synchronisation.
- Comes at a cost of overhead; 5 signal elements are required to encode 4 bits.
- That is, 100Mbps Ethernet requires 125Mhz clock

4B5B

•0000 → 11110	1000 → 10010
•0001 → 01001	1001 → 10011
•0010 → 10100	1010 → 10110
•0011 → 10101	1011 → 10111
•0100 → 01010	1100 → 11010
•0101 → 01011	1101 → 11011
•0110 → 01110	1110 → 11100
•0111 → 01111	1111 → 11101

4B5B

- Good clocking characteristics
  - Can implement with MLT3 (100baseT UTP)
  - or NRZI (100baseT Fiber)
- More complicated to implement
  - need same dictionary in both transmitter and receiver
- Not DC balanced
  - of the 80 bits in the previous slide, 49 were ‘1’.
  - effectively charging receiver device

8B10B

- Encode 8 bits of information with 10 signalling elements
  - still require 25% more bandwidth than bit rate
- DC-balanced
  - dictionary better defined.
- Used in
  - 1Gbps Ethernet
  - SATA
  - Firewire
  - PCI Express
  - DVI.

Summary

- Many, many ways to encode digital data into a digital signal
- Trade offs are in:
  - efficiency of signalling
  - complexity to implement
  - clocking characteristics
  - noise immunity
  - DC balance