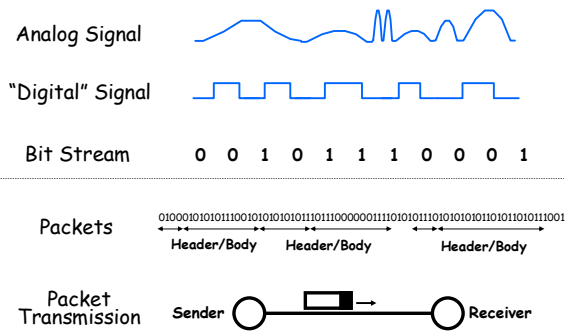
CS 640: Introduction to Computer Networks

Aditya Akella

Lecture 5 -
Encoding and Data Link Basics

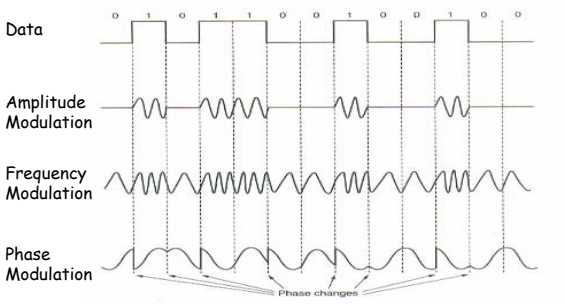
Signals, Data and Packets



Binary data to Signals

- Modulation: changing attributes of signal to effect information transmissions
- Encoding
  - How to convert bits to "digital" signals
  - Very complex, actually
  - Error recovery, clock recovery,…

Modulation Schemes



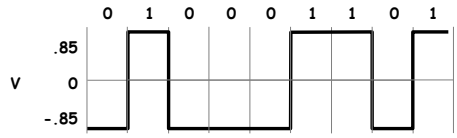
Why Do We Need Encoding?

- Meet certain electrical constraints.
  - Receiver needs enough "transitions" to keep track of the transmit clock
  - Avoid receiver saturation
- Create control symbols, besides regular data symbols.
  - E.g. start or end of frame, escape, …
  - Important in packet switching
- Error detection or error corrections.
  - Some codes are illegal so receiver can detect certain classes of errors
  - Minor errors can be corrected by having multiple adjacent signals mapped to the same data symbol
- Encoding can be very complex, e.g. wireless.

Encoding

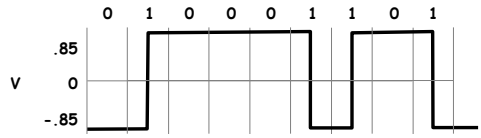
- Use two signals, high and low, to encode 0 and 1.
- Transmission is synchronous, i.e., a clock is used to sample the signal.
  - In general, the duration of one bit is equal to one or two clock ticks
  - Receiver's clock must be synchronized with the sender's clock
- Encoding can be done one bit at a time or in blocks of, e.g., 4 or 8 bits.

Non-Return to Zero (NRZ)



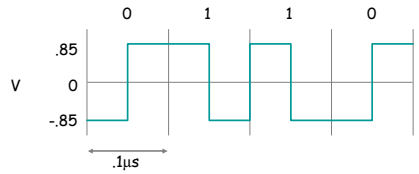
- 1 -> high signal; 0 -> low signal
- Long sequences of 1's or 0's can cause problems:
  - Hard to recover clock
  - Difficult to interpret 0's and 1's

Non-Return to Zero Inverted (NRZI)



- 1 -> make transition; 0 -> signal stays the same
- Solves the problem for long sequences of 1's, but not for 0's.

Ethernet Manchester Encoding



- Positive transition for 0, negative for 1
- XOR of NRZ with clock
- Transition every cycle communicates clock (but need 2 transition times per bit)
- Problem: doubles the rate at which signal transitions are made
  - Less efficient
  - Receiver has half the time to detect the pulse

4B/5B Encoding

- Data coded as *symbols* of 5 line bits => 4 data bits, so 100 Mbps uses 125 MHz.
  - Uses less frequency space than Manchester encoding
- Each valid symbol has no more than one leading zero and no more than two trailing zeros
  - At least two 1s ➔ Get dense transitions
- Uses NRZI to encode the 5 code bits
  - What happens if there are consecutive 1s?
- Example: FDDI.

4B/5B Encoding

- 16 data symbols, 8 control symbols
  - Control symbols: idle, begin frame, etc.
  - Remaining 8 are invalid

Data	Code	Data	Code
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

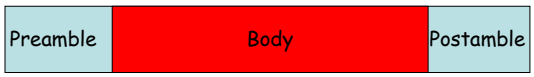
Other Encodings

- 8B/10B: Fiber Channel and Gigabit Ethernet
  - DC balance
- 64B/66B: 10 Gbit Ethernet
- B8ZS: T1 signaling (bit stuffing)

Datalink Protocol Functions

1. **Framing**: encapsulating a network layer
   - Add header, mark and detect frame boundaries, …
2. **Error control**: error detection and correction to deal with bit errors.
   - May also include other reliability support, e.g. retransmission
3. **Error correction**: Correct bit errors if possible
4. **Flow control**: avoid sender outrunning the receiver.
5. **Media access**: controlling which frame should be sent over the link next
   - Easy for point-to-point links
     - Half versus full duplex
   - Harder for multi-access links
     - Who gets to send?
6. **Switching**: How to send frames to the eventual destination?

Framing



- A link layer function, defining which bits have which function
- *Minimal functionality*: mark the beginning and end of packets (or frames).
- Some techniques:
  - frame delimiter characters with character stuffing
  - frame delimiter codes with bit stuffing
  - synchronous transmission (e.g. SONET) out of band delimiters

Byte Stuffing

- Mark end of frame with special character
  - BISYNC uses "ETX"
  - What happens when the user sends this character?
    - Use escape character when controls appear in data
  - Very common on serial lines; old technique
  - View frame as a collection of bytes

Byte Counting

- An alternative is to include a count of number of bytes
  - Next to the start of frame
  - E.g. DDCMP
  - Corruptions of count field may cause receiver to receive incorrectly
  - Include an error-check to help receiver realize this

Bit Stuffing

- Treat frames as a sequence of bits
- Mark frames with special bit sequence
  - Example, HDLC: 01111110 is a special sequence or "flag"
    - Used at the beginning and end of frame
  - But, must ensure data containing this sequence can be transmitted
    - Flag can cross byte boundaries
  - transmitter inserts a 0 when this is likely to appear in the data:
    - 111111 → 1111101
    - must stuff a zero any time five 1s appear:
  - receiver unstuffs.
- Problem with stuffing techniques: frame size depends on data
  - Frames can be of different size
  - Could lead to some inefficiencies