CSCI-1680
Physical Layer
Link Layer I

John Jannotti

Based partly on lecture notes by David Mazières, Phil Levis, Rodrigo Fonseca

Today

- **Physical Layer**
  – Modulation and Channel Capacity
  – Encoding
- **Link Layer I**
  – Framing

Layers, Services, Protocols

Application	Service: user-facing application. Application-defined messages
Transport	Service: multiplexing applications Reliable byte stream to other node (TCP), Unreliable datagram (UDP)
Network	Service: move packets to any other node in the network IP: Unreliable, best-effort service model
Link	Service: move frames to other node across link. May add reliability, medium access control
Physical	Service: move bits to other node across link

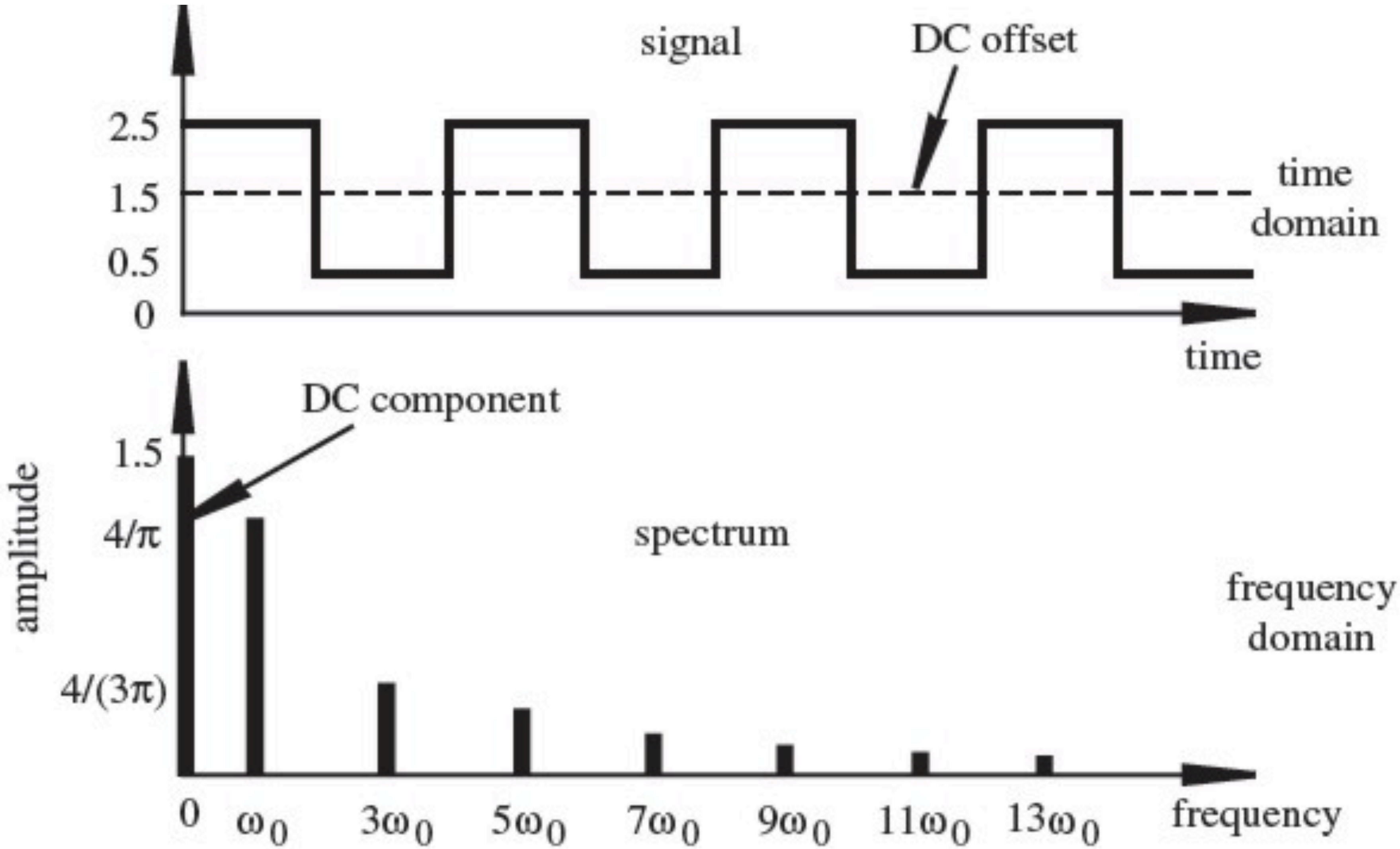
Physical Layer (Layer 1)

- **Responsible for specifying the physical medium**
  - Type of cable, fiber, wireless frequency
- **Responsible for specifying the signal (modulation)**
  - Transmitter varies *something (*amplitude, frequency, phase)
  - Receiver samples, recovers signal
- **Responsible for specifying the bits (encoding)**
  - Bits above physical layer -> *chips*

Modulation

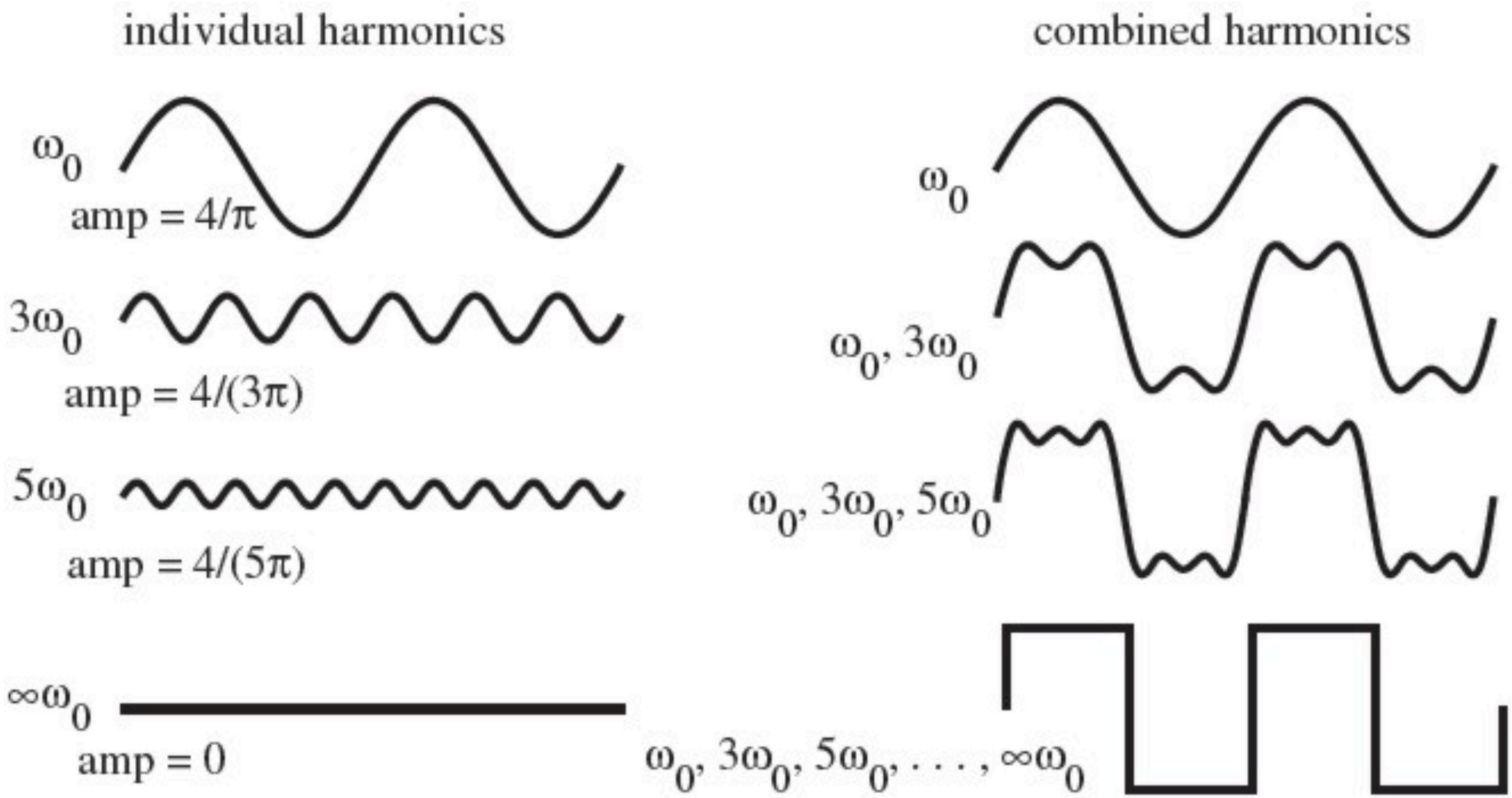
- **Specifies mapping between digital signal and some variation in analog signal**
- **Why not just a square wave (1v=1; 0v=0)?**
  - Not square when bandwidth limited
- **Bandwidth – frequencies that a channel propagates well**
  - Signals consist of many frequency components
  - Attenuation and delay frequency-dependent

Components of a Square Wave



Graphs from Dr. David Alciatore, Colorado State University

Approximation of a Square Wave

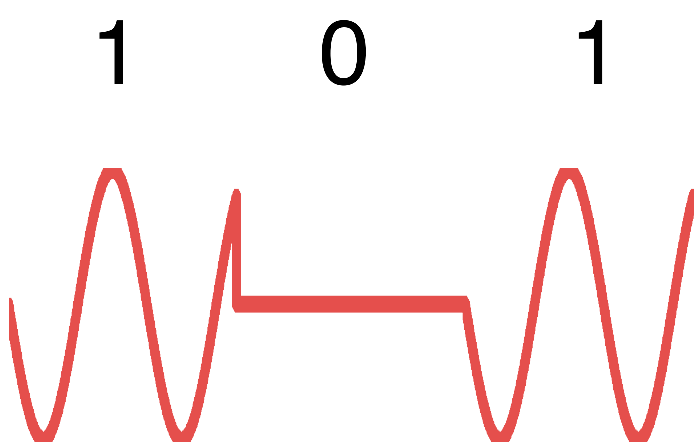


Graphs from Dr. David Alciatore, Colorado State University

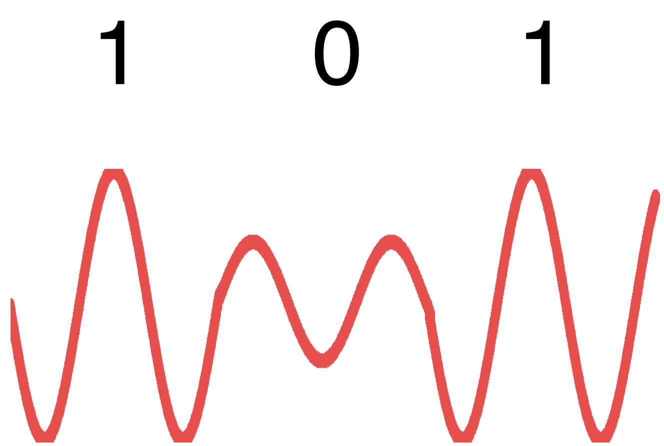
Idea: Use Carriers

- Only use frequencies that transmit well
- *Modulate* the signal to encode bits

OOK: On-Off Keying



ASK: Amplitude Shift Keying

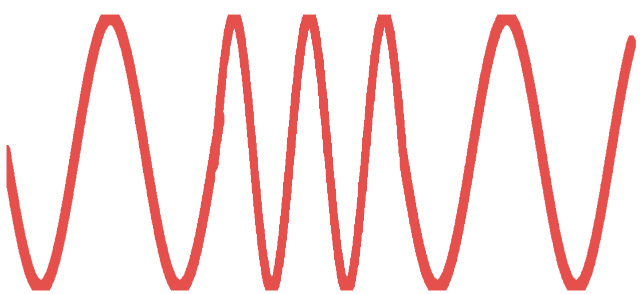


Idea: Use Carriers

- Only use frequencies that transmit well
- *Modulate* the signal to encode bits

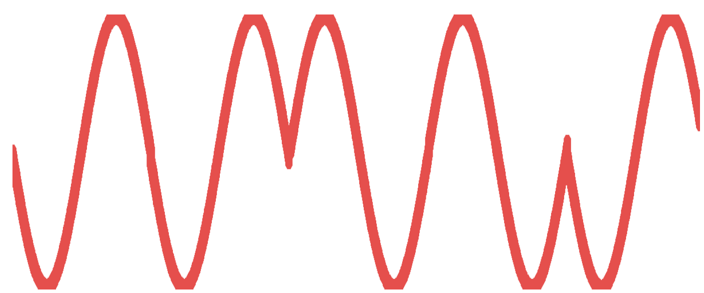
FSK: Frequency Shift Keying

1 0 1



PSK: Phase Shift Keying

1 0 1



How Fast Can You Send?

- Encode information in some varying characteristic of the signal.
- If B is the bandwidth of the signal

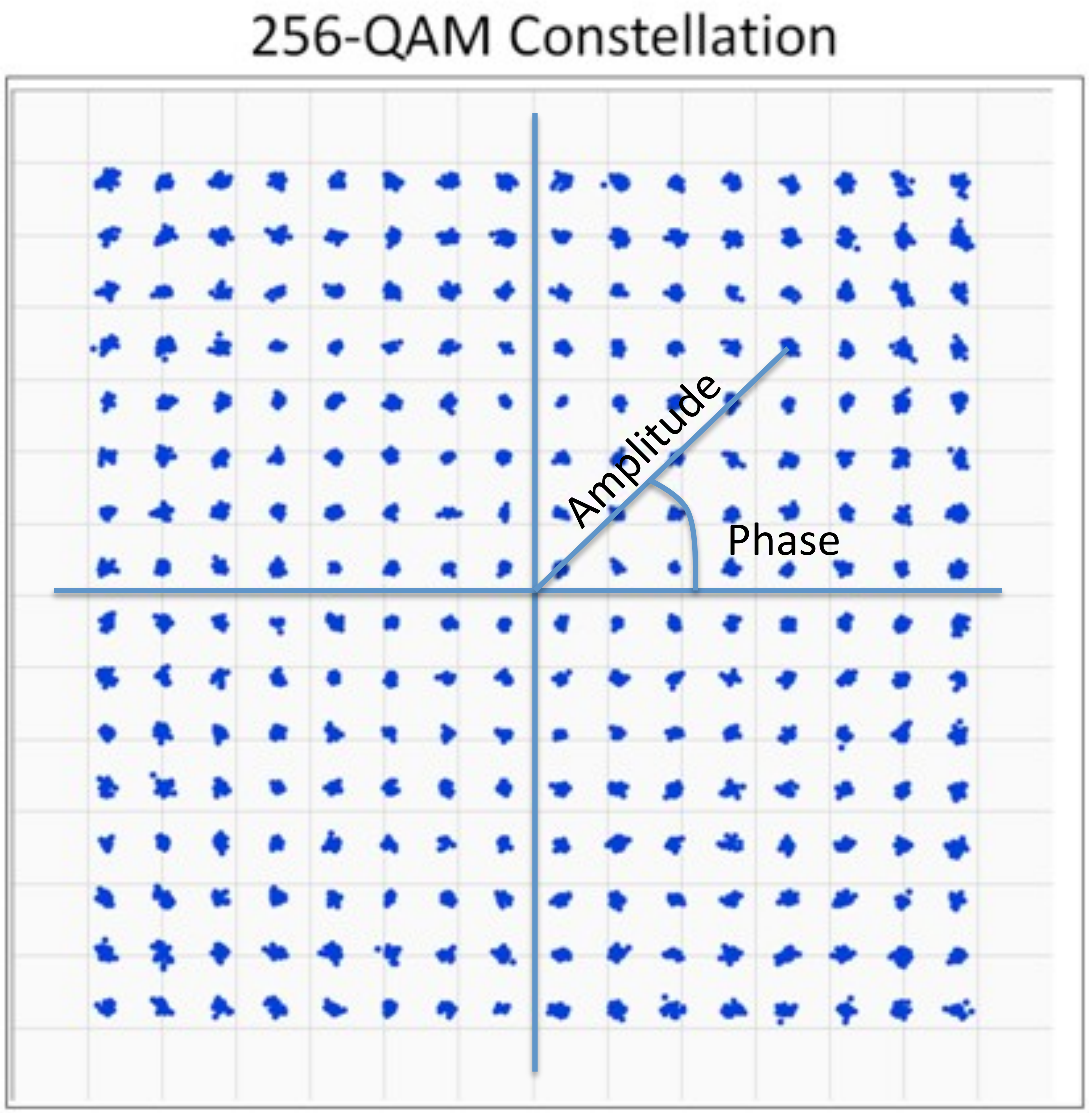
$$C = 2B \text{ bits/s}$$

(Nyquist, 1928)

Can we do better?

- **So we can only change 2B/second, what if we encode more bits per sample?**
  - Baud is the frequency of changes to the physical channel
  - Not the same thing as bits!
- **Suppose channel passes 1KHz to 2KHz**
  - 1 bit per sample: alternate between 1KHz and 2KHz
  - 2 bits per sample: send one of 1, 1.33, 1.66, or 2KHz
  - Or send at different amplitudes: A/4, A/2, 3A/4, A
  - n bits: choose among 2^n frequencies!
- **What is the capacity if you can distinguish M levels?**

Example



Hartley's Law

$$C = 2B \log_2(M) \text{ bits/s}$$

Great. By increasing M, we can have as large a capacity as we want!

Or can we?

The channel is noisy!

The channel is noisy!

- **Noise prevents you from increasing M arbitrarily!**
- **This depends on the signal/noise ratio (S/N)**
- **Shannon:** $C = B \log_2(1 + S/N)$
  - C is the channel capacity in bits/second
  - B is the bandwidth of the channel in Hz
  - S and N are average signal and noise power
  - Signal-to-noise ratio is measured in dB = $10\log_{10}(S/N)$

Putting it all together

- **Noise limits M!**

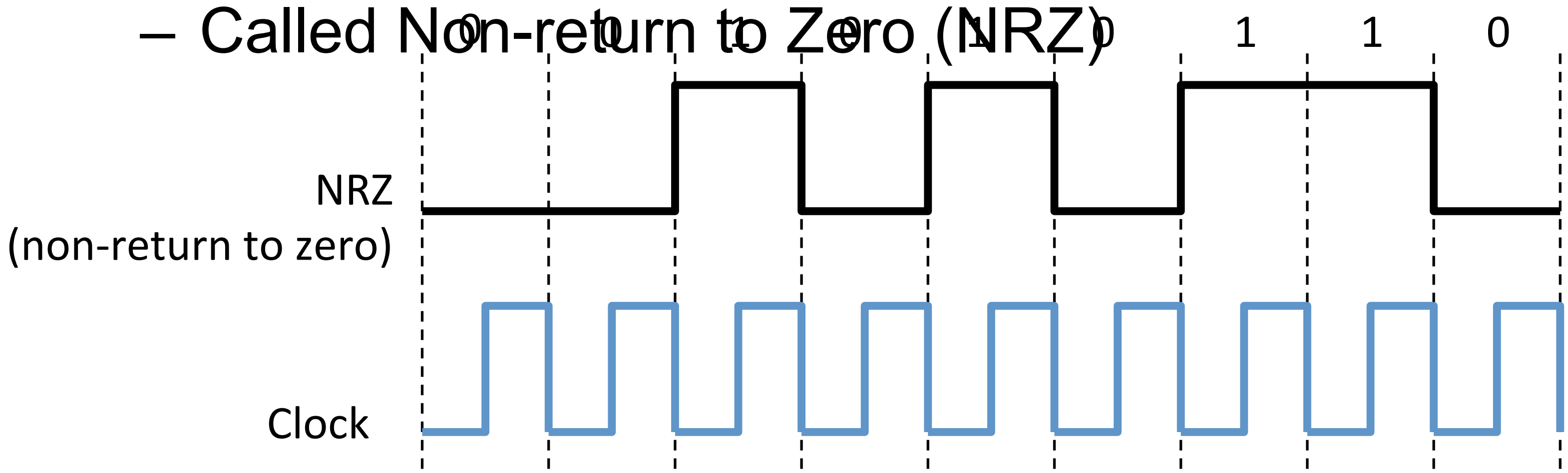
$$2B \log_2(M) \leq B \log_2(1 + S/N)$$

$$M \leq \sqrt{1+S/N}$$

- **Example: Telephone Line**
  - 3KHz b/w, 30dB S/N = 10ˆ(30/10) = 1000
  - C = 3KHz $\log_2(1001) \approx$ 30Kbps

Encoding

- **Now assume that we can somehow modulate a signal: receiver can decode our binary stream**
- **How do we encode binary data onto signals?**
- **One approach: 1 as high, 0 as low!**
  - Called Non-return to Zero (NRZ)

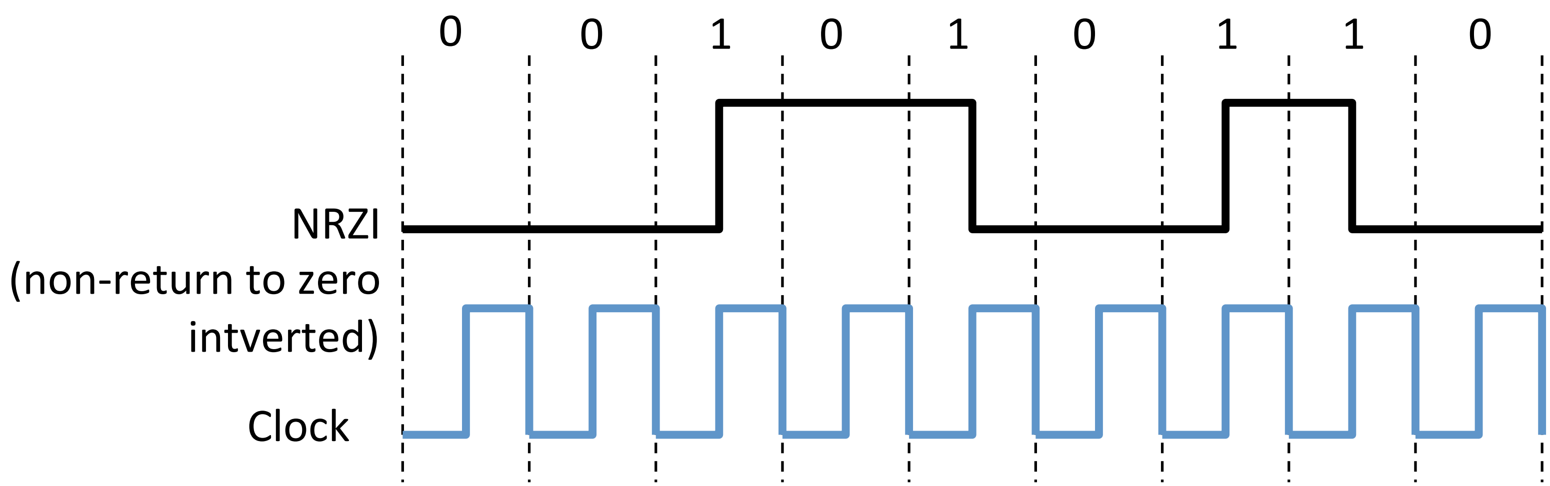


Drawbacks of NRZ

- **No signal could be interpreted as 0 (or vice-versa)**
- **Consecutive 1s or 0s are problematic**
- **Baseline wander problem**
  - How do you set the threshold?
  - Could compare to average, but average may drift
- **Clock recovery problem**
  - For long runs of no change, could miscount periods

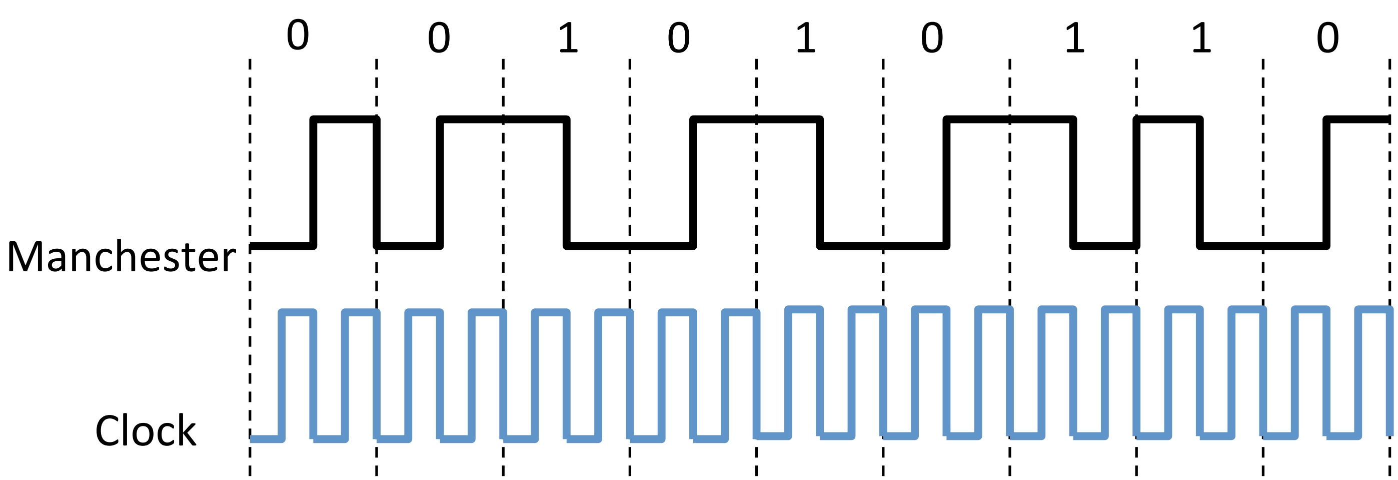
Alternative Encodings

- **Non-return to Zero Inverted (NRZI)**
  - Encode 1 with transition from current signal
  - Encode 0 by staying at the same level
  - At least solve problem of consecutive 1s



Manchester

- **Map 0 → chips 01; 1 → chips 10**
  - Transmission rate now 1 bit per two clock cycles
- **Solves clock recovery, baseline wander**
- **But cuts transmission rate in half**



4B/5B

- **Can we have a more efficient encoding?**
- **Every 4 bits encoded as 5 *chips***
- **Need 16 5-bit codes:**
  - selected to have no more than one leading 0 and no more than two trailing 0s
  - ***Never get more than 3 consecutive 0s***
- **Transmit chips using NRZI**
- **Other codes used for other purposes**
  - E.g., 11111: line idle; 00100: halt
- **Achieves 80% efficiency**

4B/5B Table

0	0000	11110
1	0001	01001
2	0010	10100
3	0011	10101
4	0100	01010
5	0101	01011
6	0110	01110
7	0111	01111
8	1000	10010
9	1001	10011
A	1010	10110
B	1011	10111
C	1100	11010
D	1101	11011
E	1110	11100
F	1111	11101

Encoding Goals

- **DC Balancing (same number of 0 and 1 chips)**
- **Clock synchronization**
- **Can recover some chip errors**
- **Constrain analog signal patterns to make signal more robust**
- **Want near channel capacity with negligible errors**
  - Shannon says it's possible, doesn't tell us how
  - Codes can get computationally expensive
- **In practice**
  - More complex encoding: fewer bps, more robust
  - Less complex encoding: more bps, less robust

Last Example: 802.15.4

- **Standard for low-power, low-rate wireless PANs**
  - Must tolerate high chip error rates
- **Uses a 4B/32B bit-to-chip encoding**

Symbols

Bits	Chips
0000	1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
0001	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
0010	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
0011	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1
⋮	
1111	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

Questions so far?

Photo: Lewis Hine

Today

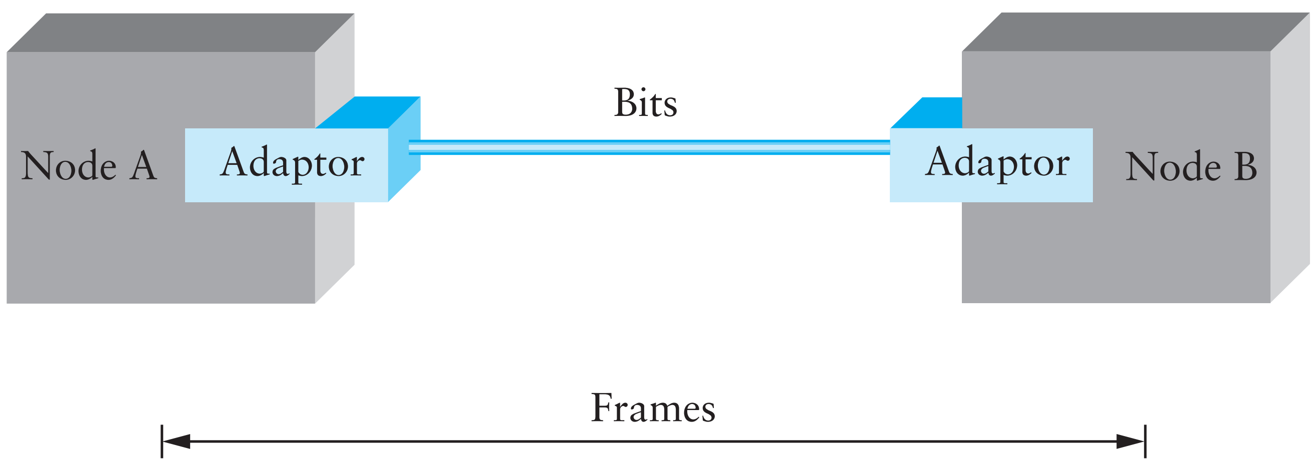
- **Physical Layer**
  - Modulation and Channel Capacity
  - Encoding
- **Link Layer I**
  - Framing

Layers, Services, Protocols

Application	Service: user-facing application. Application-defined messages
Transport	Service: multiplexing applications Reliable byte stream to other node (TCP), Unreliable datagram (UDP)
Network	Service: move packets to any other node in the network IP: Unreliable, best-effort service model
Link	Service: move frames to other node across link. May add reliability, medium access control
Physical	Service: move bits to other node across link

Framing

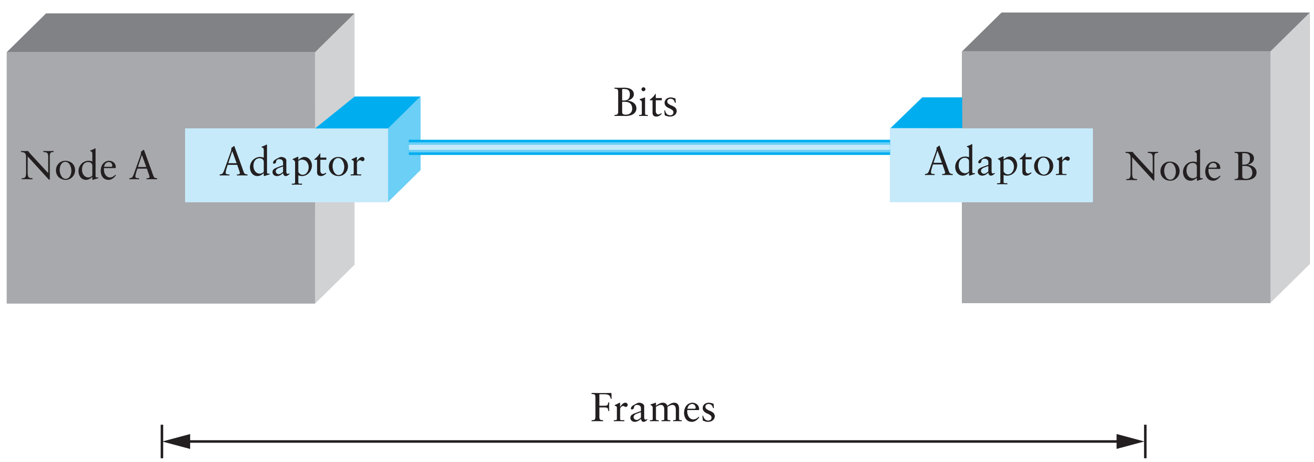
- Given a stream of bits, how can we represent boundaries?
- Break sequence of bits into a frame
- Typically done by network adaptor



Link Layer
Framing

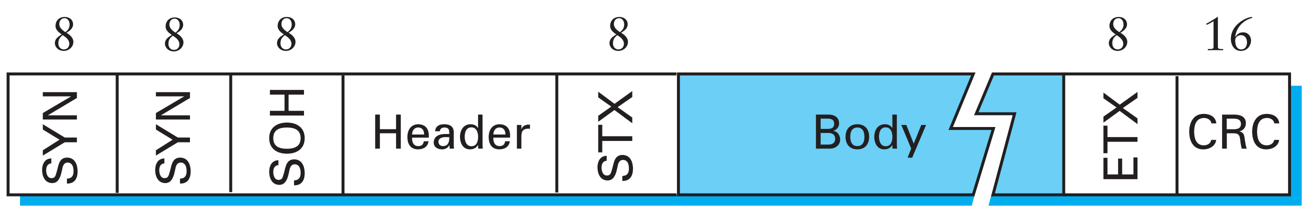
Representing Boundaries

- Sentinels
- Length counts
- Clock-based



Sentinel-based Framing

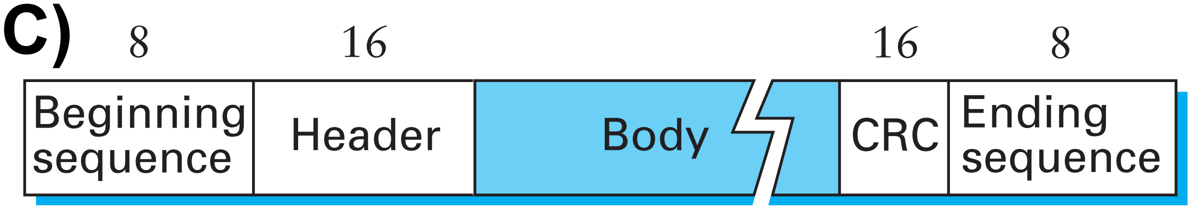
- **Byte-oriented protocols (*e.g.* BISYNC, PPP)**
  – Place special bytes (SOH, ETX,…) in the beginning, end of messages



- **What if ETX appears in the body?**
  – Escape ETX byte by prefixing DEL byte
  – Escape DEL byte by prefixing DEL byte
  – Technique known as *character stuffing*

Bit-Oriented Protocols

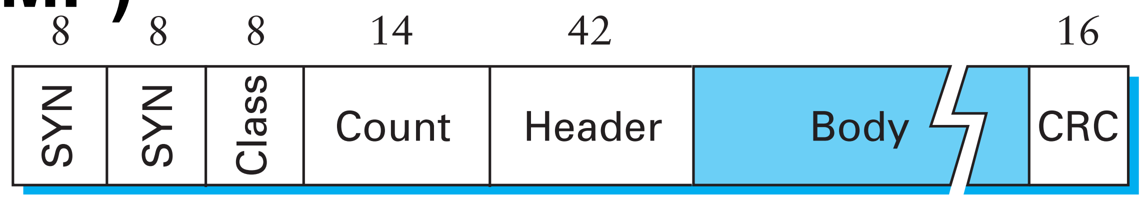
- **View message as a stream of bits, not bytes**
- **Can use sentinel approach as well (*e.g.*, HDLC)**



  - HDLC begin/end sequence 01111110
- **Use *bit stuffing* to escape 01111110**
  - Always append 0 after five consecutive 1s in data
  - After five 1s, receiver uses next two bits to decide if stuffed, end of frame, or error.

Length-based Framing

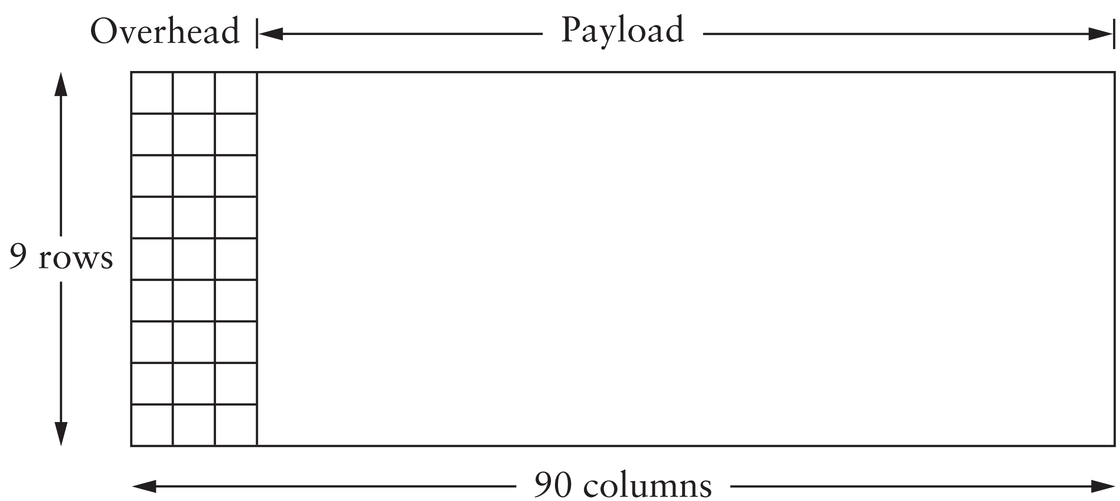
- **Drawback of sentinel techniques**
  - Length of frame depends on data
- **Alternative: put length in header (*e.g.*, DDCMP)**



- **Danger: Framing Errors**
  - What if high bit of counter gets corrupted?
  - Adds 8K to length of frame, may lose many frames
  - CRC checksum helps detect error

Clock-based Framing

- *E.g.*, **SONET (Synchronous Optical Network)**
  - Each frame is 125μs long
  - Look for header every 125μs
  - Encode with NRZ, but first XOR payload with 127-bit string to ensure lots of transitions



Error Detection

- **Basic idea: use a checksum**
  - Compute small checksum value, like a hash of the frame data
- **Good checksum algorithms**
  - Want several properties, *e.g.*, detect any single-bit error
  - Details in a later lecture

Next Week

- **Next week: more link layer**
  - Flow Control and Reliability
  - Ethernet
  - Sharing access to a shared medium
  - Switching
- **Next Thursday (Sep 24th): Snowcast due, HW1 out**