

Lecture 26:

Modulation & Clock Recovery

CSE 123: Computer Networks

Aaron Schulman





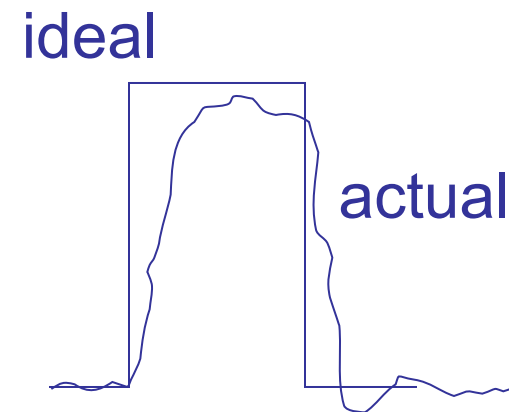
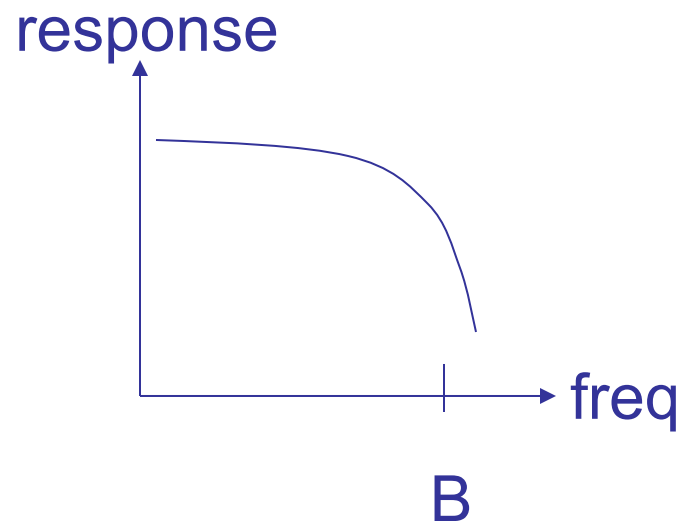
Lecture 26 Overview

- Channel constraints
- Signaling constraints
 - ◆ Shannon's Law
 - ◆ Nyquist Limit
- Encoding schemes
 - ◆ Clock recovery
 - ◆ Manchester, NRZ, NRZI, etc.

Channel Challenges



- Every channel degrades a signal
 - ◆ Distortion impacts how the receiver will interpret signal





Channel Properties

- **Bandwidth**-limited
 - ◆ Range of frequencies the channel will transmit
 - ◆ Means the channel is slow to react to change in signal
- Power **attenuates** over distance
 - ◆ Signal gets softer (harder to “hear”) the further it travels
 - ◆ Different frequencies have different response (**distortion**)
- Background **noise** or interference
 - ◆ May add or subtract from original signal
- Different physical characteristics
 - ◆ Point-to-point vs. shared media
 - ◆ Very different price points to deploy

Copper Channels



- Typical examples

- ◆ Category 5/6 Twisted Pair
- ◆ Coaxial Cable

10M-10Gbps

50-100m

10-100Mbps

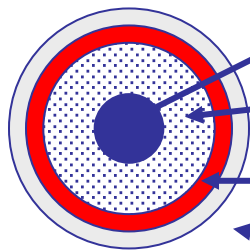
200m

twisted pair



Northwire, Inc. — DataCELL® GEV-1000™ Cable

coaxial
cable
(coax)

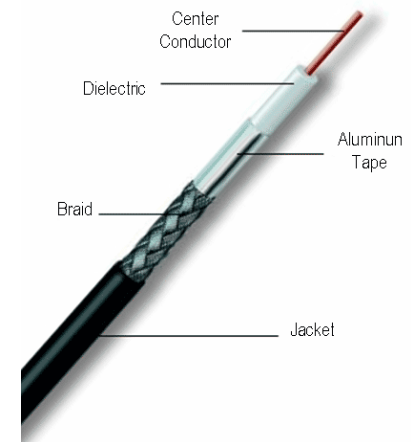


copper core

insulation

braided outer conductor

outer insulation

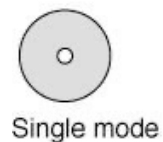
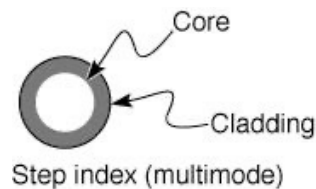




Fiber Optic Channels

- Typical examples

- | | | |
|---------------------|--------------|-----------|
| ◆ Multimode Fiber | 100Mbps-10Gb | 500-2000m |
| ◆ Single Mode Fiber | 1-10Tbps | 100m-40km |



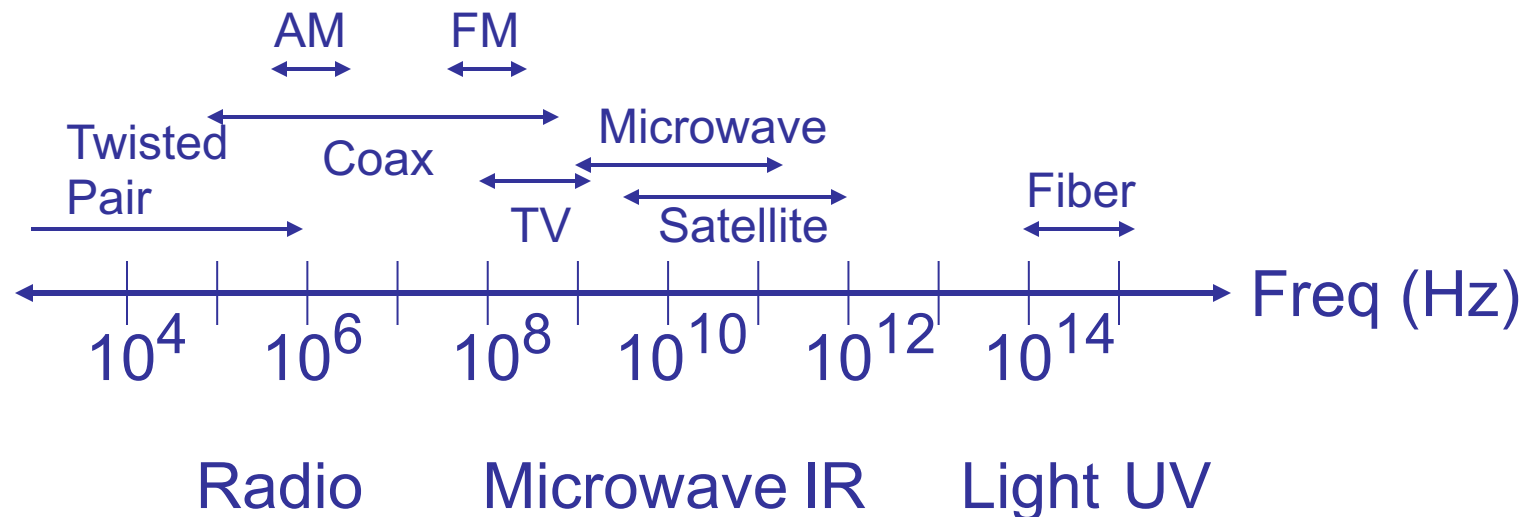
**Cheaper to drive
(LED vs laser) &
terminate**

**Longer distance
(low attenuation)
Higher data rates
(low dispersion)**



Wireless Channels

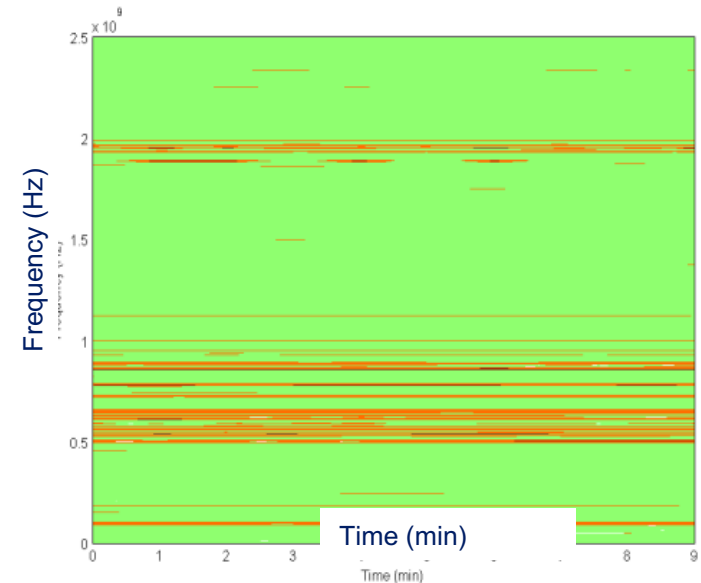
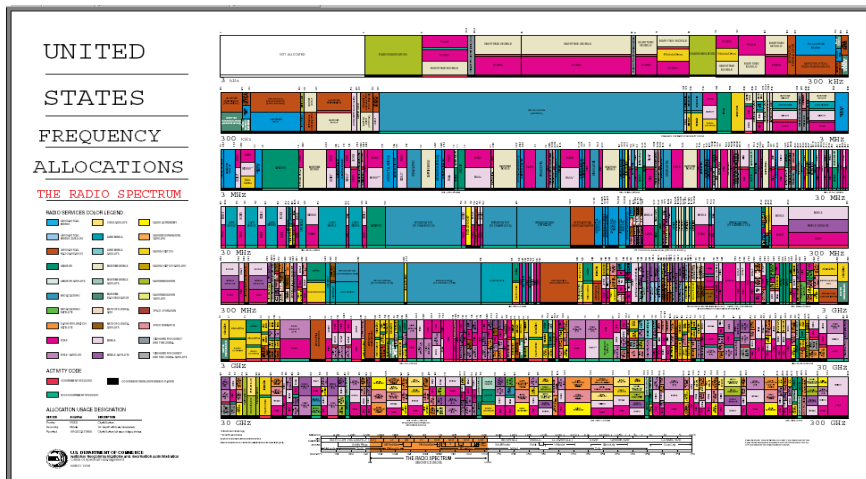
- Widely varying channel bandwidths/distances
- Extremely vulnerable to noise and interference



Wireless Spectrum Allocation



- Policy approach forces spectrum to be allocated like a fixed spatial resource (e.g. land, disk space, etc)
- Reality is that spectrum is time and power shared
- Measurements show that fixed allocations are poorly utilized



Hot topic: Sharing spectrum with military



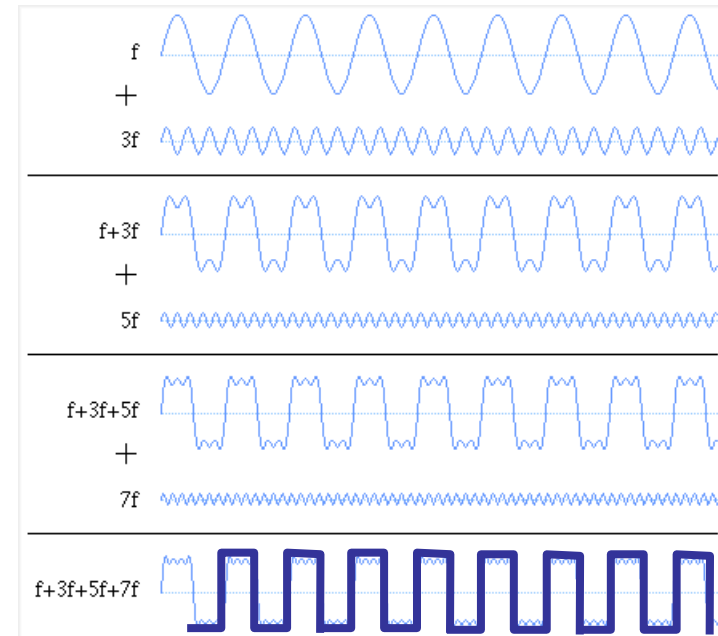
Two Main Tasks at Layer-1

- 1. First we need to transmit data as a signal (modulation)
 - ◆ Determine how to send the data, and how quickly
- 2. Then we need to receive data from a degraded signal (demod)
 - ◆ Figure out when someone is sending bits
 - ◆ Determine which bits they are sending
- A lot like a conversation
 - ◆ “WhatintheworldamIsaying” – needs punctuation and pacing
 - ◆ Helps to know what language I’m speaking



The Magic of Sine Waves

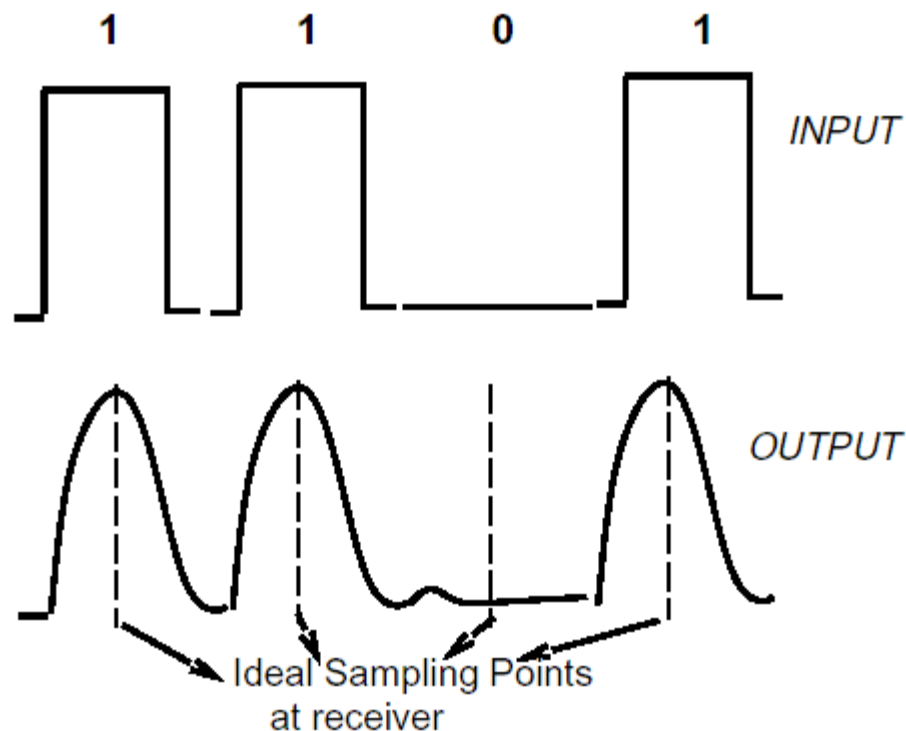
- Movement of energy in nature is often involving waves
- All periodic signals can be expressed as sine waves
 - ◆ Component waves are of different frequencies
- Sine waves are “nice”
 - ◆ Phase shifted or scaled by most channels
- “Easy” to analyze
 - ◆ Fourier analysis can tell us how signal changes
 - ◆ But not in this class...



Sampling into Symbols – From Continuous to Discrete



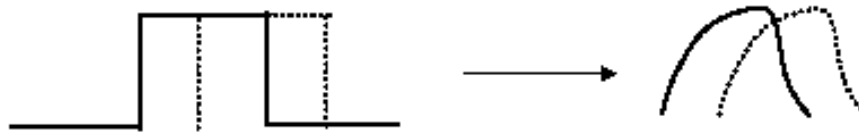
- To decode a signal in a computer, we need to sample it
- We are converting **continuous** wave to **discrete** bits
- We always have a limit in sampling frequency f_s





Intersymbol Interference

- Bandlimited channels cannot respond faster than some maximum frequency $f_{bandwidth}$
 - ◆ Channel takes some time to settle
- Attempting to signal too fast for channel will mix symbols
 - ◆ Previous symbol still “settling in”
 - ◆ Mix (add/subtract) adjacent symbols
 - ◆ Leads to **intersymbol interference** (ISI)

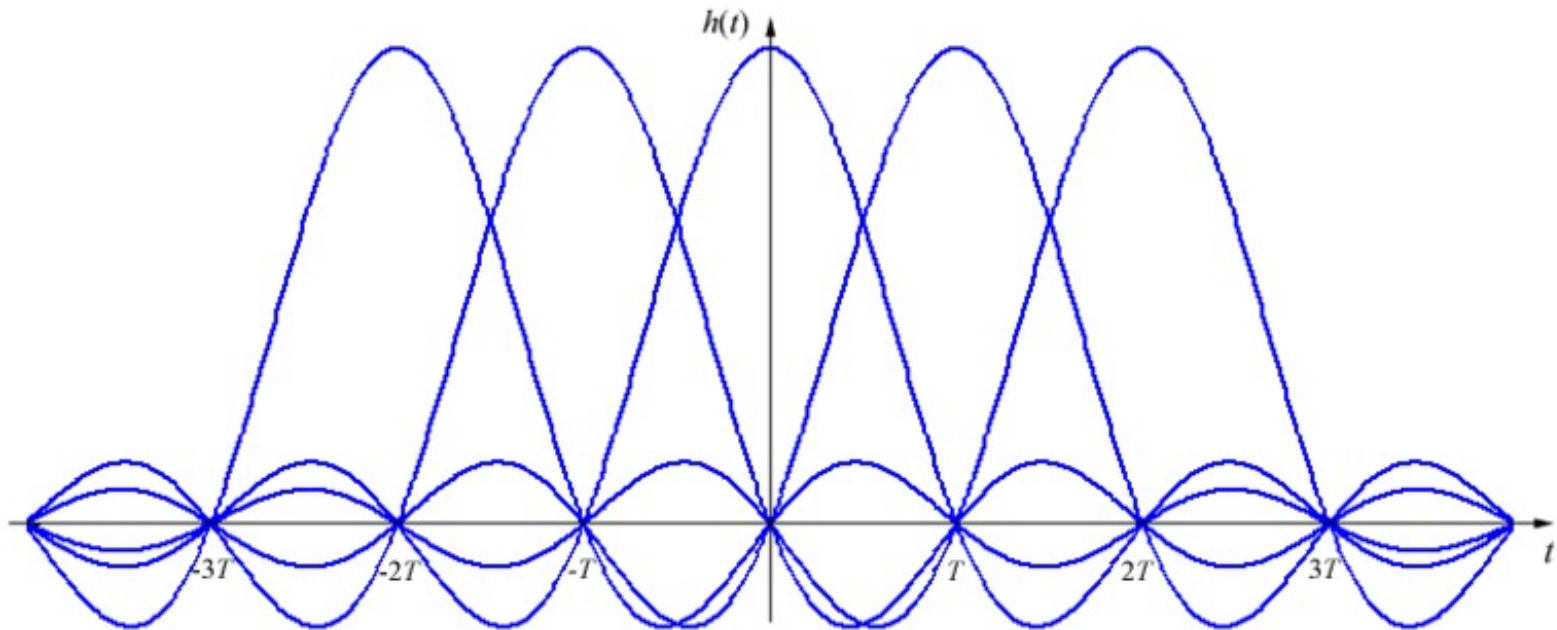


- OK, so just how fast can we send symbols?



Speed Limit: Nyquist

- In a channel bandlimited to a sample rate of f_s , we can send at maximum symbol (**baud**) rate of $2f$ without ISI





Multiple Bits per Symbol

- Nyquist limits the number of symbols per second we can send, but doesn't talk about the amount of information content in each symbol
- Couldn't we send *multiple* bits per symbol
 - ◆ E.g., multiple voltage levels instead of just high/low
 - ◆ Four levels gets you two bits, $\log_2 M$ in general (M levels)
- Can combine this observation with Nyquist
 - ◆ *Channel capacity:* $C < 2 B \log_2(M)$
- Why not infinite levels? Infinite bandwidth no?



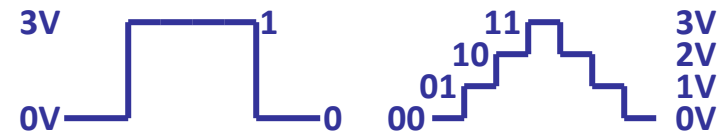
Noise matters

- Real channels are **noisy**... noise creates measurement challenges

- Example:

- ◆ Encode 4 values using voltage

- » 2 bits per symbol
 - » Symbols at 3V, 2V, 1V and 0V



- ◆ What if noise is 0.5V?
 - » If you get line level of 2.5V then what symbol is it? 11 or 10?

- Limited to $\sim \log_2 (S/2N)$ bits per symbol
(S = signal power, N = Noise)

- ◆ Previous example: $S = 3V - 0V = 3V$, $N = 0.5V$, so we can have $\log_2(3/1) = 1.58$ bits per symbol



Shannon's Law

- Shannon considered noisy channels and derived

$$C = B \log (1 + S/N)$$

- Gives us an upper bound on any channel's bitrate regardless of signaling scheme
- Old school modems approached this limit
 - ◆ $B = 3000\text{Hz}$, $S/N = 30\text{dB} = 1000$
 - ◆ $C = 3000 \times \log(1000) \approx 30\text{kbps}$
 - ◆ 28.8Kbps – anyone remember dialup?

Next problem: Clock recovery



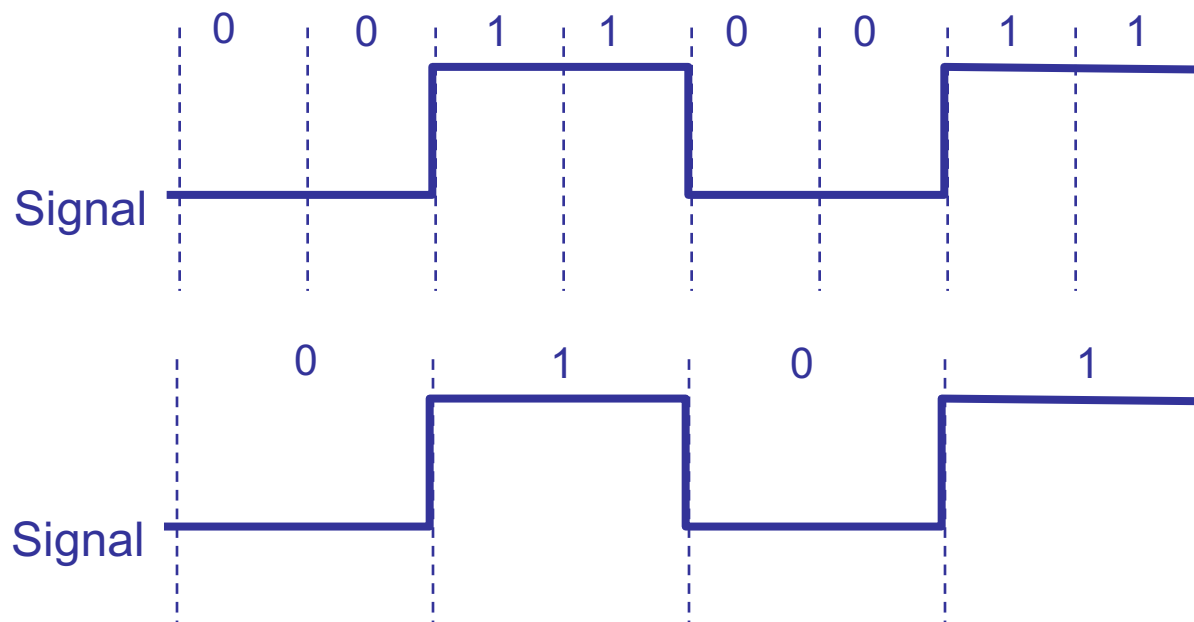
- How does the receiver know when to sample the signal?
 - ◆ Sampling rate: How often to sample?
 - ◆ Sampling phase:
 - » When to start sampling? (getting in phase)
 - » How to adjust sampling times (staying in phase)

Why the sampling rate matters



- Signal could have multiple interpretations

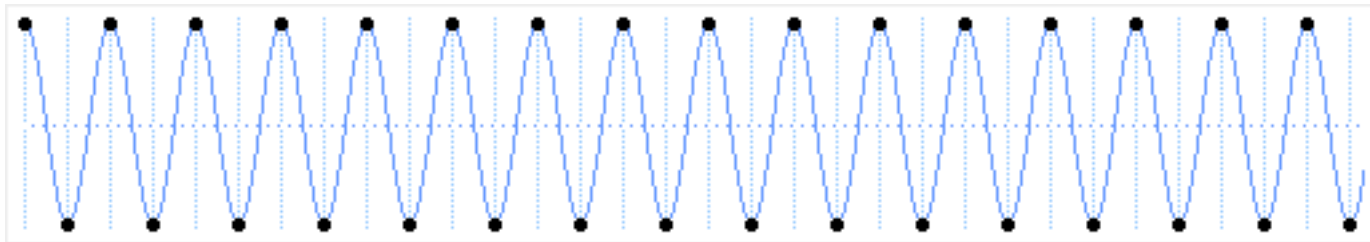
Which of these is correct?



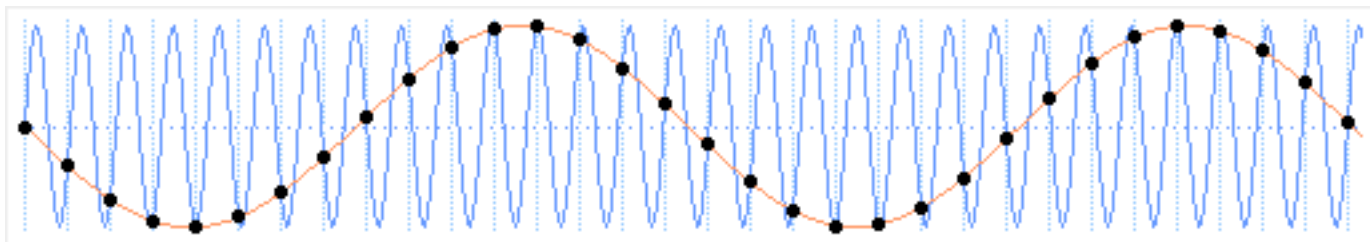


Nyquist Revisited

- Sampling at the correct rate ($2f$) yields actual signal
 - ◆ Always assume lowest-frequency wave that fits samples



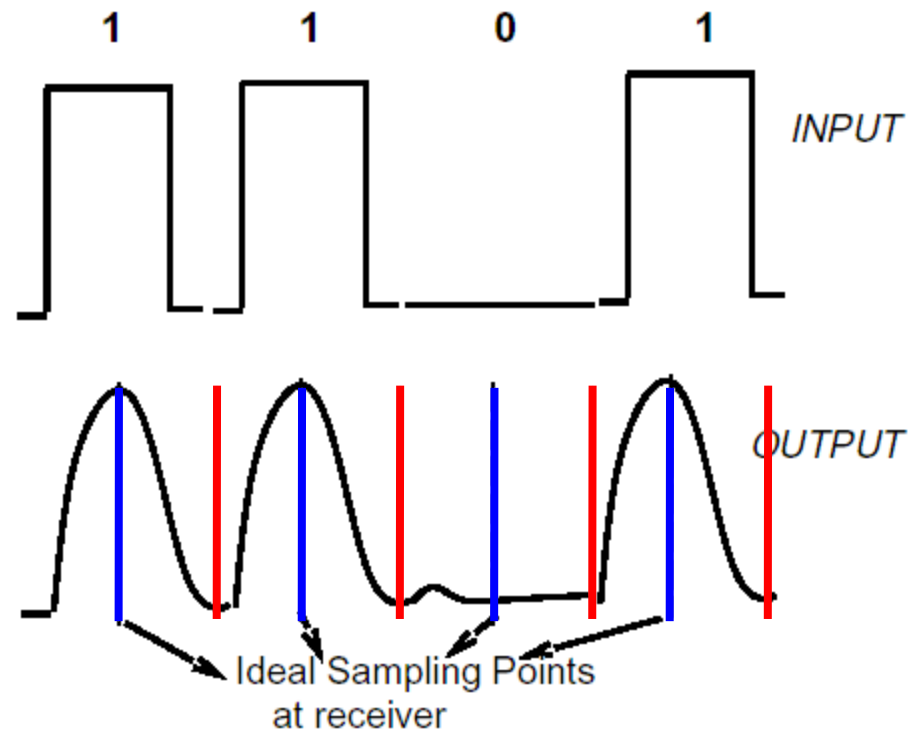
- Sampling too slowly yields aliases





The Importance of Phase

- Need to determine when to START sampling, too





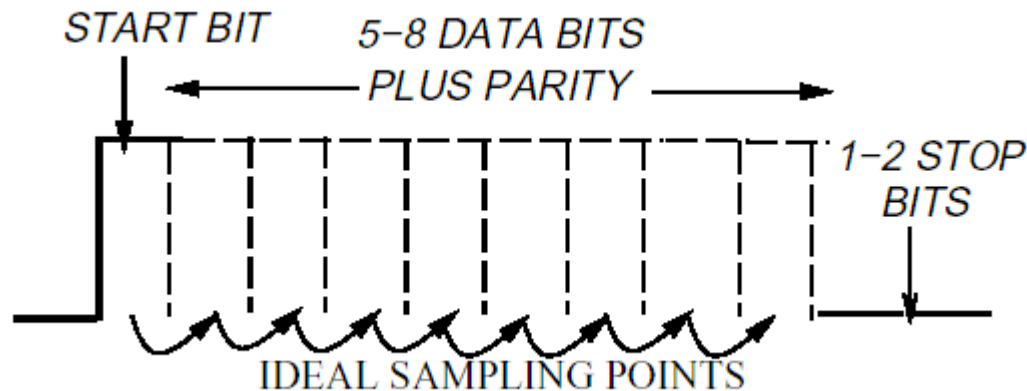
Clock Recovery

- Using a training sequence to get receiver lined up
 - ◆ Send a few, known **initial training bits**
 - ◆ Adds inefficiency: only m data bits out of n transmitted
- Need to combat clock drift as signal proceeds
 - ◆ Use **transitions** to keep clocks sync'ed up
- Question is, how often do we do this?
 - ◆ Quick and dirty – every message: **asynchronous** coding
 - ◆ Spend a lot of effort to get it right, but amortize over lots of data: **synchronous** coding



Asynchronous Coding

- Encode several bits (e.g. 7) together with a leading “start bit” and trailing “stop bit”
- Data can be sent at any time

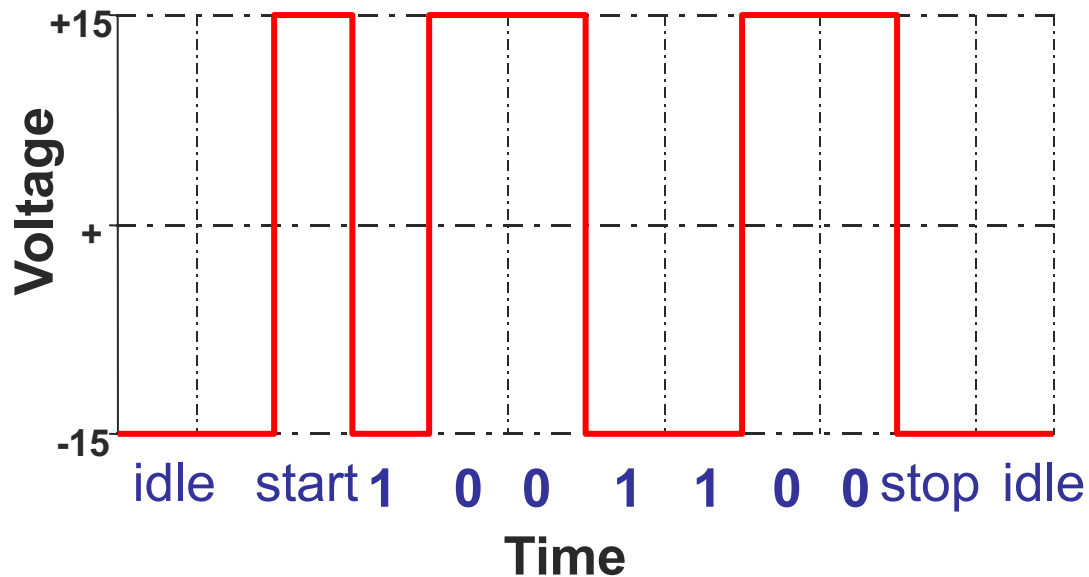


- Start bit transition kicks off sampling intervals
 - Can only run for a short while before drifting

Example: RS232 serial lines



- Uses two voltage levels (+15V, -15V), to encode single bit binary symbols
- Needs long idle time – limited transmit rate





Synchronous Coding

- Asynchronous receiver phase locks each symbol
 - ◆ Takes time, limiting transmission rates
- So, start symbols need to be extra slow
 - ◆ Need to fire up the clock, which takes time
- Instead, let's do this training once, then just keep sync
 - ◆ Need to continually adjust clock as signal arrives
 - ◆ Ever hear of Phase Lock Loops (PLLs) ?
- Basic idea is to use **transitions** to lock in



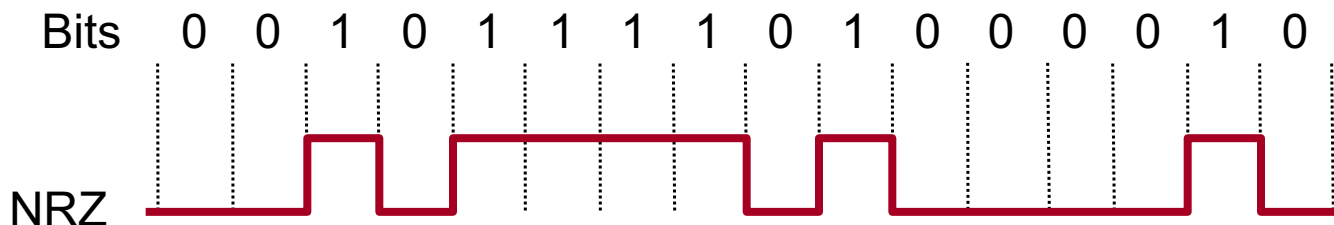
Synchronous coding

- Encode many bits (thousands) together
 - ◆ Amortize cost of learning clock information from start bits (preamble) and stop bits (trailer)
 - ◆ Continuously “learn” clock from data stream
 - » Watch for 0-1 or 1-0 transitions, and adjust clock
 - » Called clock recovery process
- Examples
 - ◆ NRZ
 - ◆ NRZI
 - ◆ Manchester
 - ◆ 4B/5B
 - ◆ Many others...



Non-Return to Zero (NRZ)

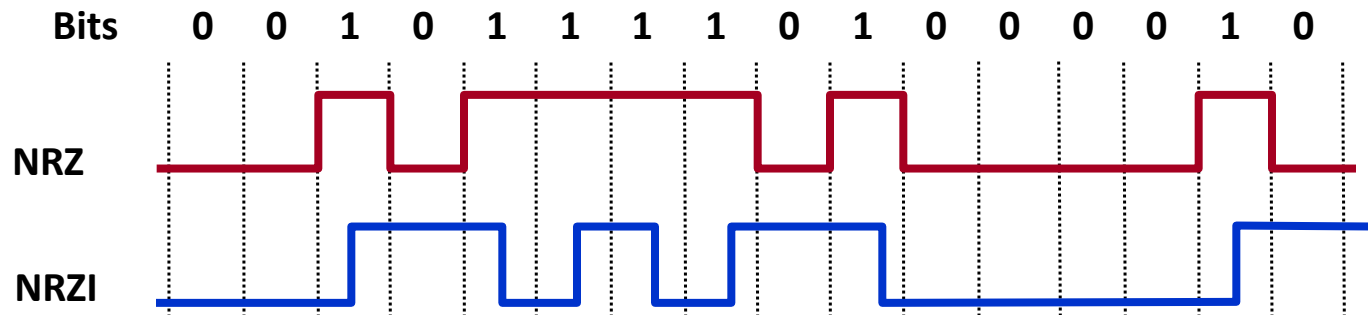
- Signal to Data
 - ◆ High $\Rightarrow$ 1
 - ◆ Low $\Rightarrow$ 0
- Comments
 - ◆ Transitions maintain clock synchronization
 - ◆ Long strings of 0s confused with no signal
 - ◆ Long strings of 1s causes *baseline wander*
 - » We use average signal level to infer high vs low
 - ◆ Both inhibit clock recovery



Non-Return to Zero Inverted (NRZI)



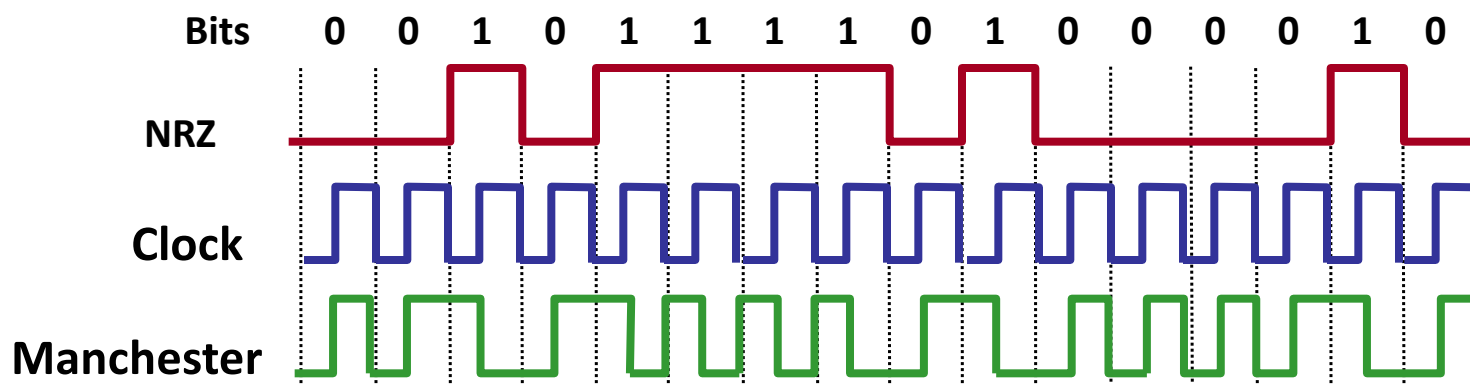
- Signal to Data
 - ◆ Transition $\Rightarrow$ 1
 - ◆ Maintain $\Rightarrow$ 0
- Comments
 - ◆ Solves series of 1s, but not 0s



Manchester Encoding (10Mbps Ethernet)



- Signal to Data
 - ◆ XOR NRZ data with senders clock signal
 - ◆ High to low transition $\Rightarrow$ 1
 - ◆ Low to high transition $\Rightarrow$ 0
- Comments
 - ◆ Solves clock recovery problem
 - ◆ Only 50% efficient ($\frac{1}{2}$ bit per transition)
 - ◆ Still need preamble (typically 0101010101... trailing 11 in Ethernet)



4B/5B (100Mbps Ethernet)



- Goal: address inefficiency of Manchester encoding, while avoiding long periods of low signals
- Solution:
 - ◆ Use five bits to encode every sequence of four bits
 - ◆ 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%

4-bit	5-bit	4-bit	5-bit
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

Encoding/Decoding Summary



- Signaling & Modulation
 - ◆ Transforming digital signal to and from analog representation
 - ◆ Fundamental limits (Shannon)
 - ◆ Lots of ways to encode signal (modulation) onto a given medium
- Clock recovery
 - ◆ Receiver needs to adjust its sampling times to best extract signal from channel
 - ◆ Sender can code signal to make it far easier to do this



For Next Class

- We'll cover some examples of media access control in the most difficult scenario, wireless communication.
- Read 2.6