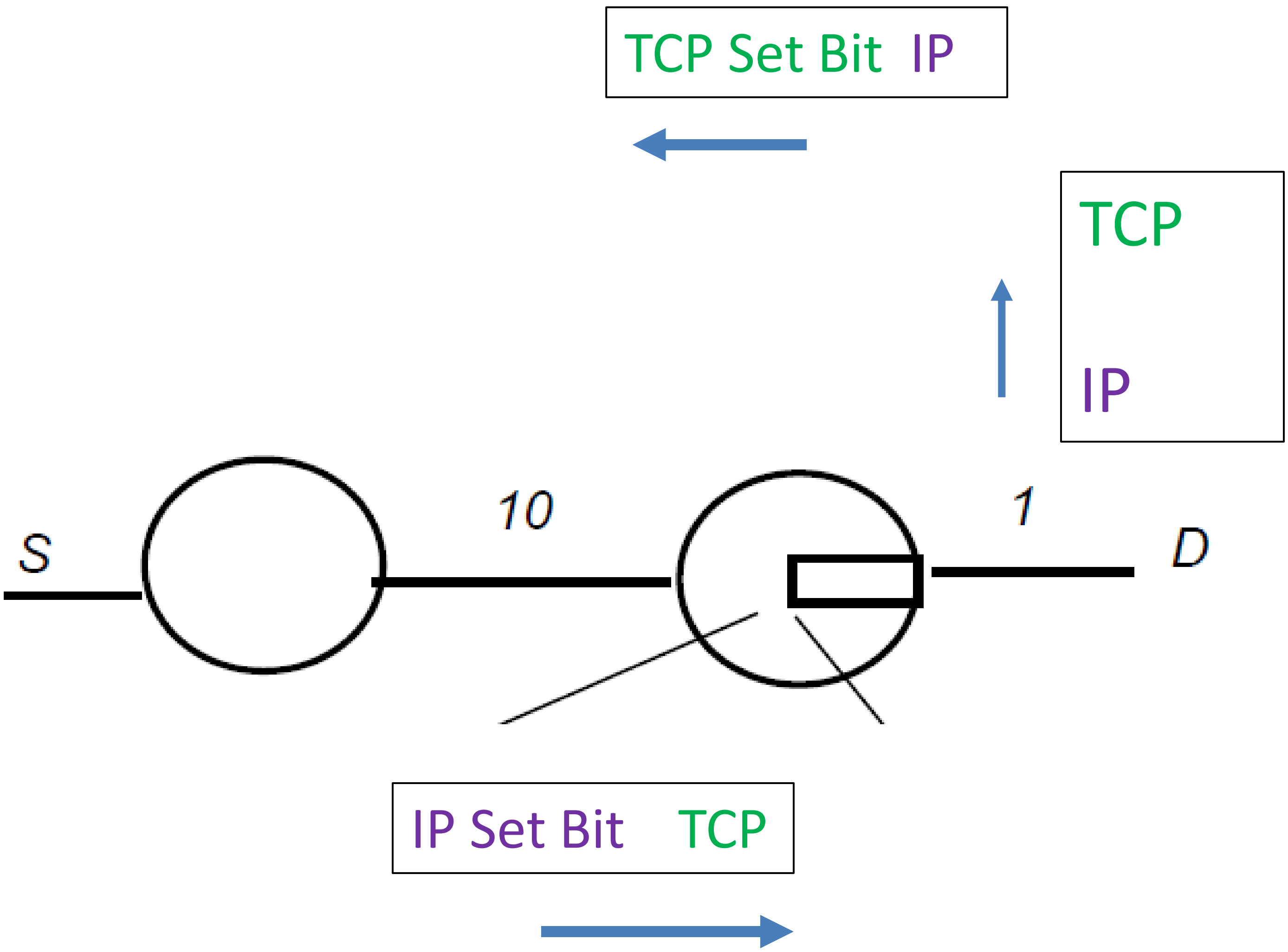
CS118: Lecture 3, Getting in Synch

George Varghese

October 3, 2022

Main Idea: Receivers learn when to sample the continuous wave received from sender by cueing on transitions!

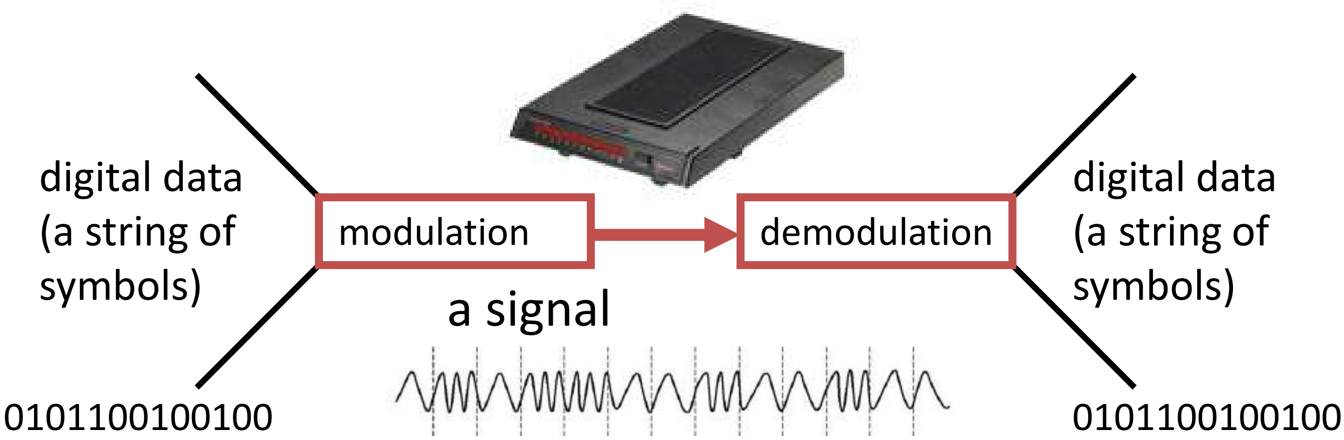
Sample Midterm Question on Layering



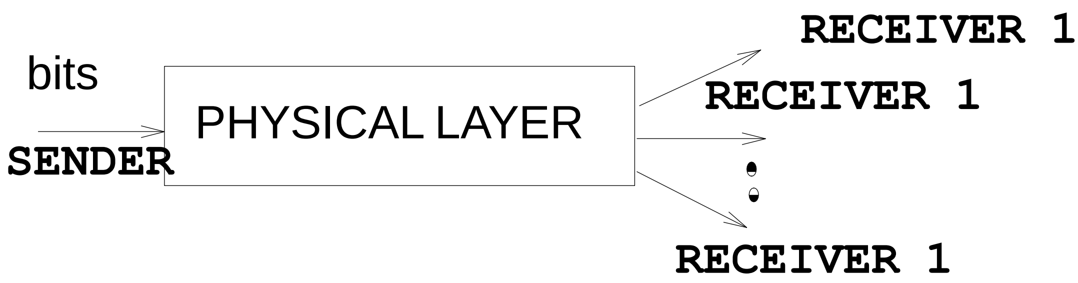
Recall Sending Bits

A three-step process

- Take an input stream of bits (digital data)
- Modulate some physical media to send data (analog)
- Demodulate the signal to retrieve bits (digital again)
- Anybody heard of a modem (Modulator-demodulator)?



What does the Physical Layer Do?



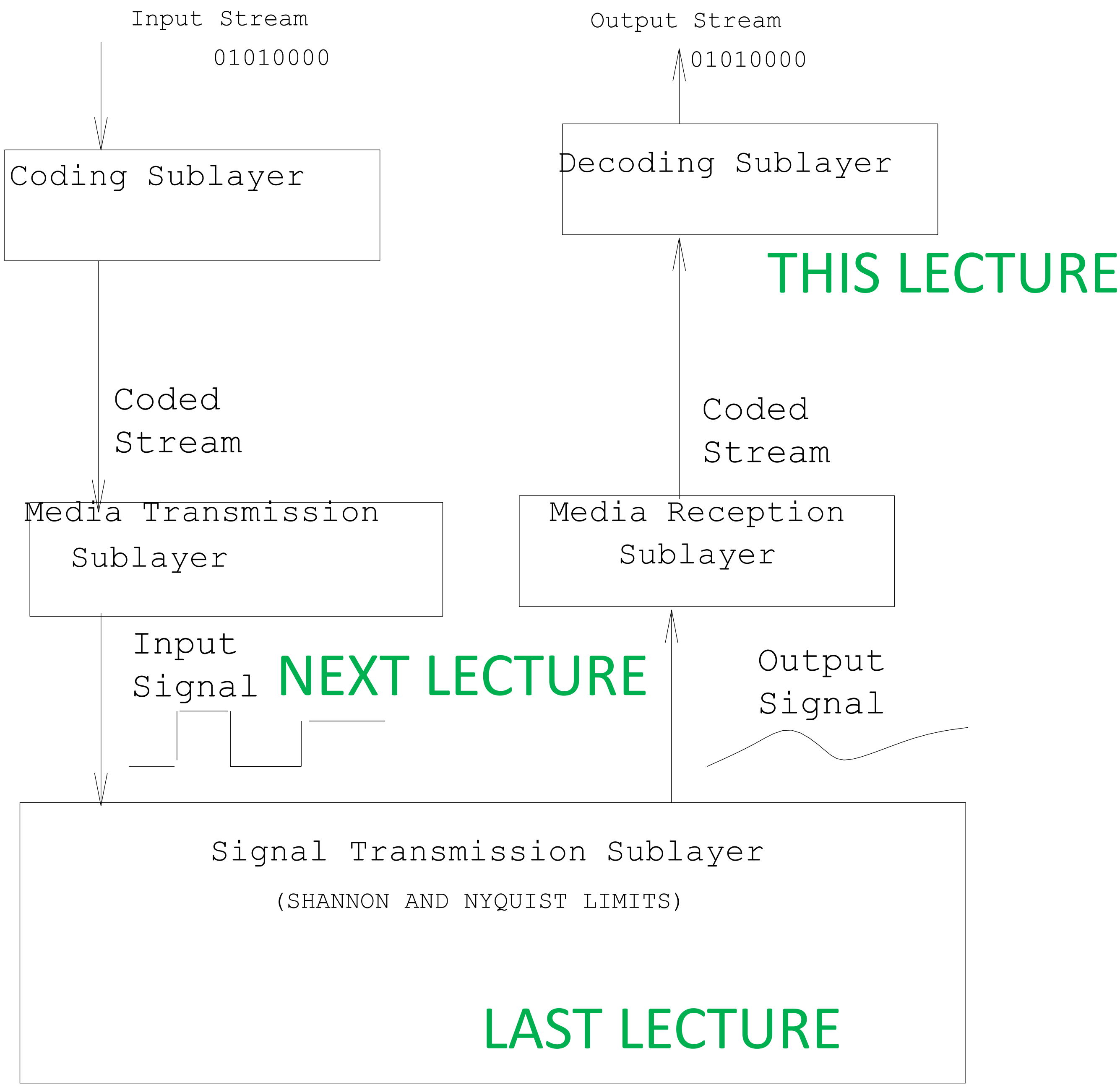
- A possibly faulty, single-hop, bit pipe that connects a sender to possibly multiple receivers

Morse Code Analogy

Example bit pipe: sending Morse Code to receivers using a flashlight. Issues:

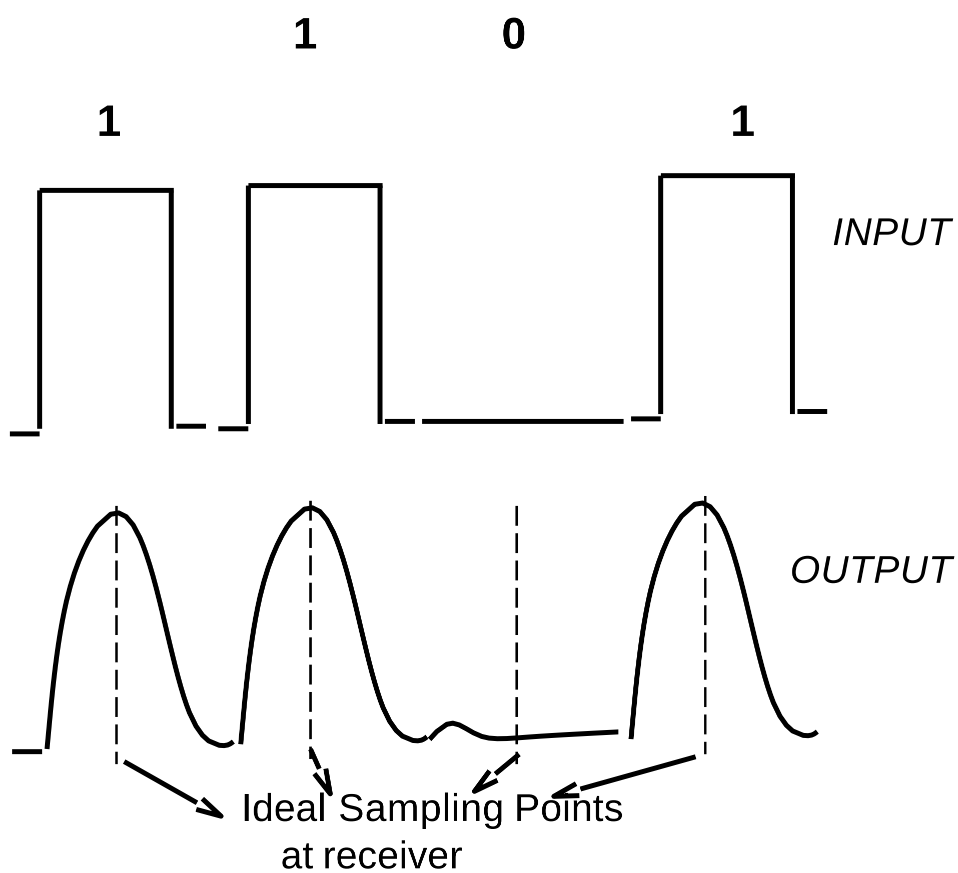
- Fundamental Limits: Brain-eye system processing limits leads to Inter Symbol Interference
- Media Issues: Flashlight, semaphore
- Coding: Morse code, getting in synch, knowing receiver rate.

PHYSICAL LAYER: SUBLAYERS



Imagine you work at Infinera and wish to run at 20 Terabit/second for the Marea cable from USA to Spain. While you can use multiple colors of light you have to to really nail the receiver sampling instants when the bits are so closely spaced.

Sampling Bits



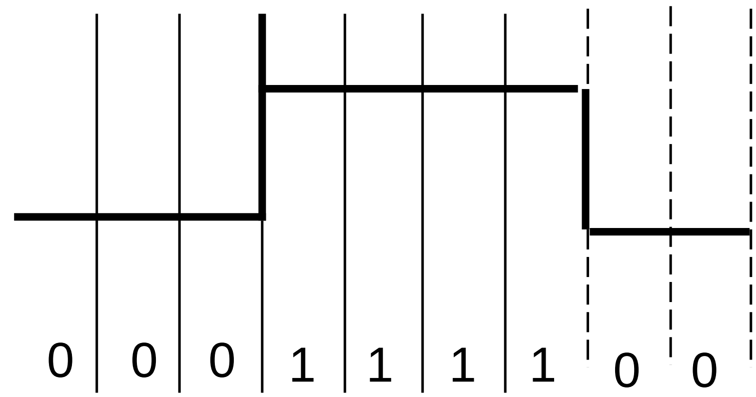
- Receivers recover the bits in the input signal by *sampling* output signal close to middle of bit period.
- Two limits to bit rate: channel bandwidth (Nyquist) and noise (Shannon). Last lecture explains why output is way it is.

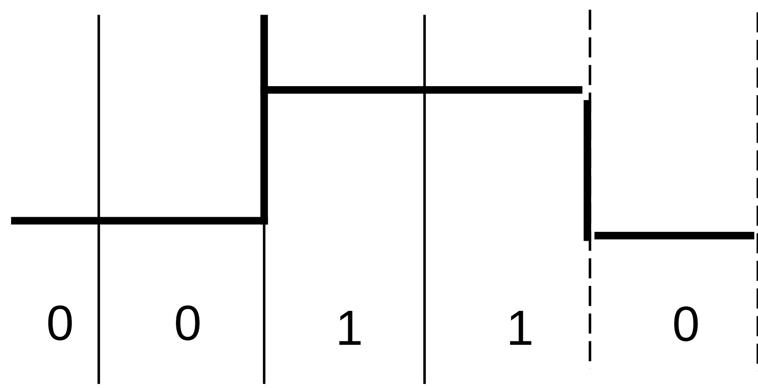
Main Topic of Lecture 3

Q: How does a receiver know when to sample bits and how can the sender help?

A: By cueing on transitions ensured by sender sending “extra bits”

Who needs a clock anyway?



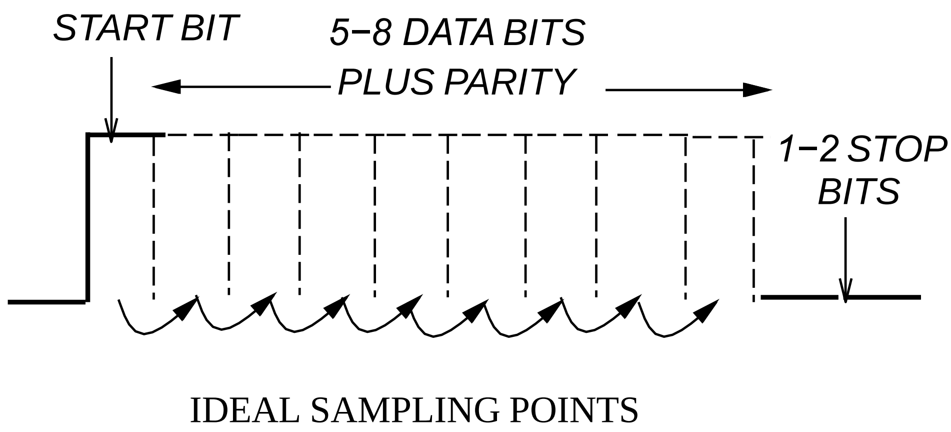


- How to initially synchronize receiver clock with sender clock? Initial Training Bits
- **Stay in Synch (first cut):** Receiver has a very similar clock as sender (say both run at 1 Ghz)
- **Problem:** All real physical clocks drift over time. Crucial at high speeds. Small drift leads to sampling error. How to keep in synch? Transitions

Transitions and Coding

- Parse: nohewontgosoon. Need spaces and punctutations to parse speech.
- Stream of bits without transitions (change in signal value) equally hard to parse.
- Real data may contain all 0's. How can you ensure transitions. Coding. Adds cost.
- Code to ensure that every n bits you get at least m transitions. Different coding schemes parameterized by n and m.

Asynchronous Coding



- Codes a character. Assume 8 bits in ASCII.
- We add "extra" start bits to get into synch and one or two stop bits. 1 is encoded as lowe voltage, 0 as high.

Receiver Code Assumes

- **Edge Detector**: A hardware gadget used to tell whether a 0 goes to a 1 or a 1 to a zero that we abstract as a function to tell where a transition is
- **Sampling:** We abstract sampling and comparing to a threshold as a function SampleSignal
- **Clock Drift:** Receiver has a clock that runs at approximately same rate as sender

```
Receiver Code

Data Structures:
C[0..10];  ARRAY, to store bits in current character

On Transition:
   StartTimer ( ? bit)
   For (i = 0 to ? do
      Wait (TimerExpiry);
      C[i] = SampleSignal;
      StartTimer ( ? )
   End;
   If (C[0] = 1) and (C[9] = C[10] = 0) then Output C[1..8]
```

EDGE DETECTOR FIRES

SAMPLE AND THRESHOLD

WHY?

Breakout: Fill in the details in the Code

- How long to wait (in bit times, possibly a fraction of a bit time) before first sampling point (first question mark)?
- How long to wait between sampling points?
- Why only pass up bits C[1..8]. How may stop bits are we assuming?
- For asynchronous, how do we get in synch? How do we stay in synch?

- **Think and engage in the Breakout. You will learn a lot more if you do!**

Breakout Answers

- **Edge Detector**: A hardware gadget used to tell whether a 0 goes to a 1 or a 1 to a zero that we abstract as a function to tell where a transition is
- **Sampling:** We abstract sampling and comparing to a threshold as a function SampleSignal
- **Clock Drift:** Receiver has a clock that runs at approximately same rate as sender

```
Receiver Code

Data Structures:
C[0..10];  ARRAY, to store bits in current character

On Transition:          EDGE DETECTOR FIRES
   StartTimer (1/2 bit)
   For (i = 0 to 10) do
      Wait (TimerExpiry);
      C[i] = SampleSignal;    SAMPLE AND THRESHOLD
      StartTimer (1 bit)
   End;
   If (C[0] = 1) and (C[9] = C[10] = 0) then Output C[1..8]
                                                     WHY?
```

SLOW CLOCK PROBLEM: HOW MUCH SLOWER CAN THE RECEIVER CLOCK BE WITHOUT CAUSING BIT ERRORS

"SYNCHRONOUS" TRANSMISSION

same as asynchronous except larger frame sizes . It requires better clock tolerance and more sophisticated coding.

Why Synchronous Transmission

- For Clock Recovery, receiver must know when to start its receive clock (phase). Then can sample the line at periodic intervals at the same rate as sender clock with some help from transitions in data.
- In asynchronous, receiver gets locked in phase when the voltage goes from low to high (start bit). Need to have fairly large idle time between characters for receiver to get locked in phase for each character; slows transmission and limits it to low rates.
- Two overheads to start bits: extra bit but also extra time needed for reliable detection. (Starting up receiver clock is expensive)
- In synchronous, we put a large number of start bits at the start of a large number of data bits. This allows the startup overhead to be amortized.

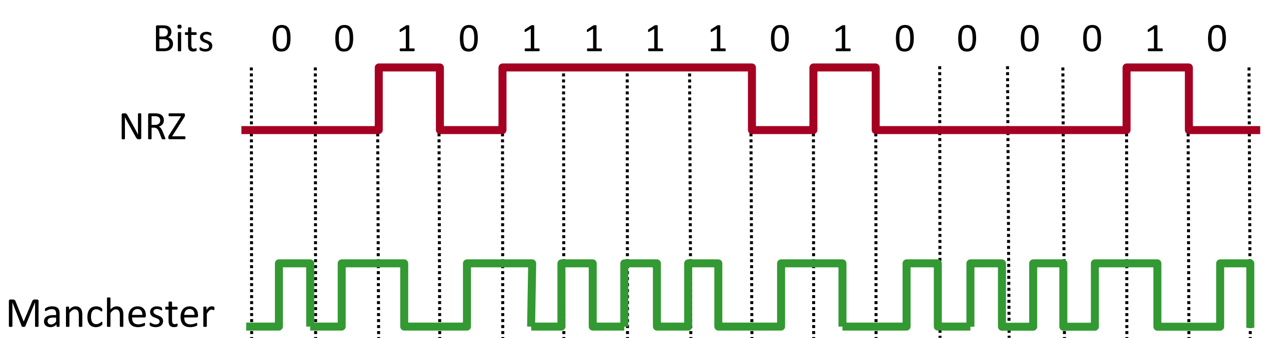
Example 2: Manchester Encoding

Signal to Data

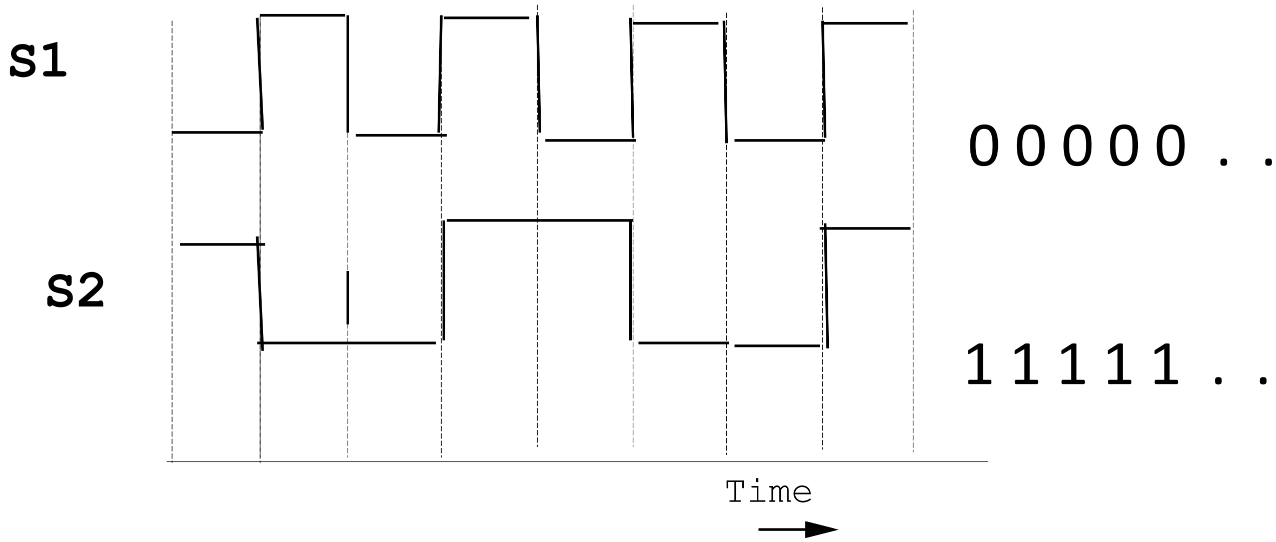
- High to low transition (10) ⇨ 1
- Low to high transition (01) ⇨ 0

Comments

- Solves clock recovery problem
- Only 50% efficient (½ bit per transition)
- Still need preamble (typically 0101010101… trailing 11 in Ethernet)



Getting locked in phase



- In asynchronous you get in phase by a single 0 to 1 transition. Not very reliable in the presence of noise.

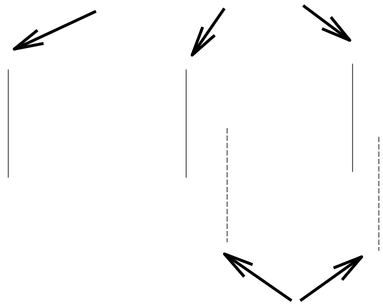
- But in Manchester, a sequence of 0's sent can be read off as a sequence of 1's if receiver is out of phase

- In Manchester, you get in phase by sending a **preamble** or group of start bits of the form 010101 in which the only transitions are at mid bit; easy to recognize and get locked in phase. Need to end with 11 to know where data starts. So preamble is 010101010111.

- Why does counting off say 5 "01" pairs not work?

Phase Locked Loops

RECEIVER SAMPLING CLOCK



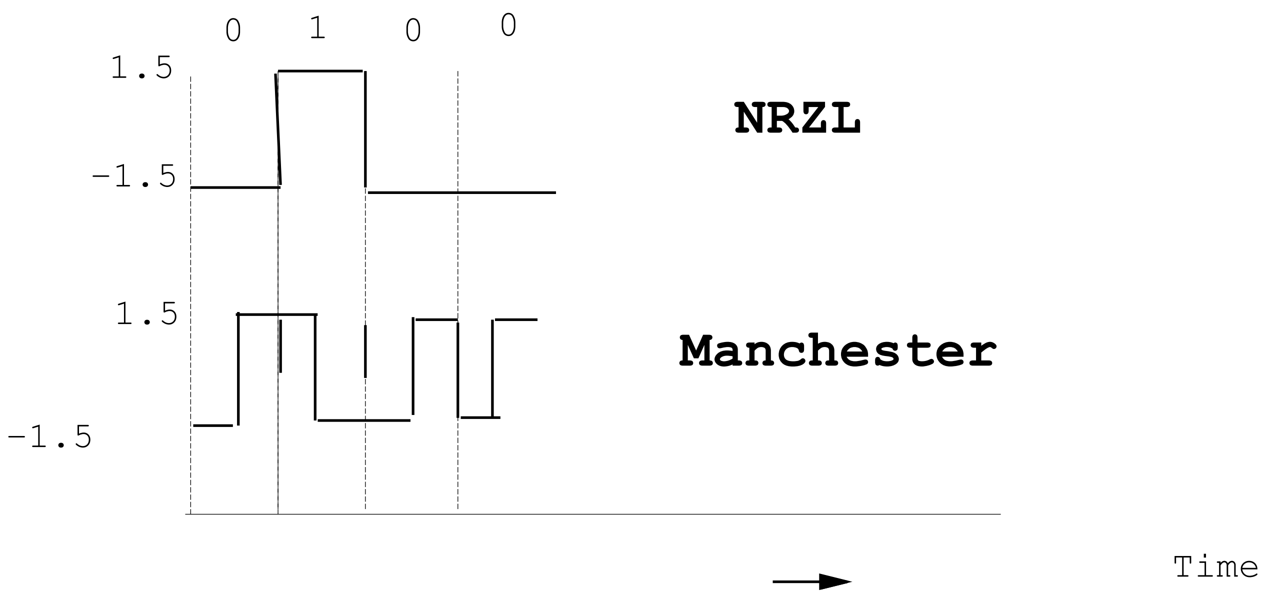
OBSERVED TRANSITIONS IN DATA

- Once you lock in at the start of a data unit, you can rely on accuracy of receiver clock frequency (as in asynchronous). Can't do that if data unit is large (as in synchronous).
- Could try resetting receiver clock on every observed transition. Susceptible to noise. Better to use more gradual adjustment.
- Phase Locked Loops *measure phase difference and speed up or slow down receiver clock* to reduce phase difference. Commonly used.

Example 3: Modern codes

- **8-10 coding**: Every group of 8 data bits is encoded as 10 bits. Easily done by table lookup
- **Why transitions**: Because all 10-bit patterns with no transitions (0000000000 and 1111111111) are not used. In fact can rule out even patterns with 1 transition as we have 1024 patterns to code a byte.
- **Receiver decoding**: A 1024 entry table that lookups the 8-bit data corresponding to each received 10-bit pattern,

What makes a code good?



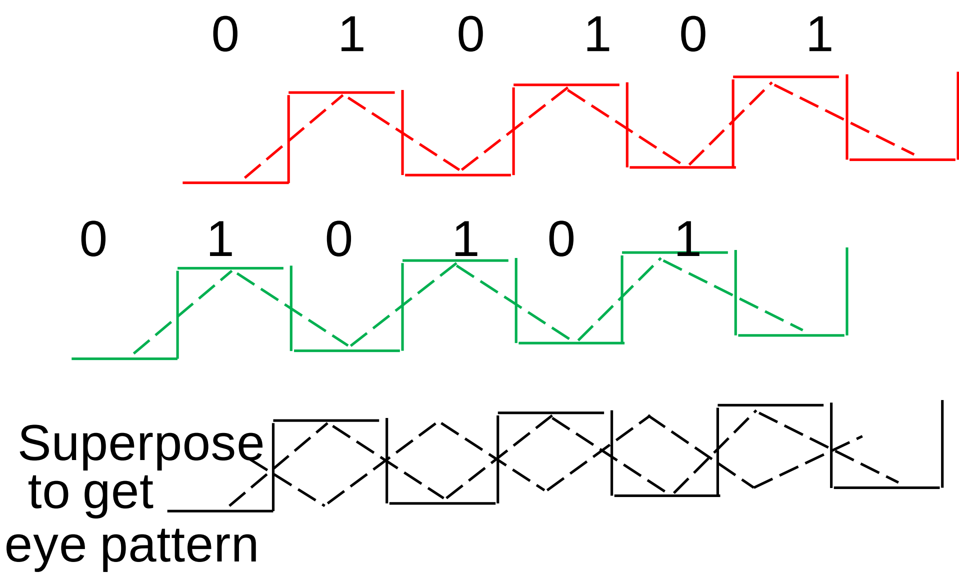
Evaluation Criteria

1. Coding Efficiency(Real Bits/Coded Bits)
2. Signal to Noise Ratio
3. DC Balance
4. Implementation Complexity

Student Clarifications

- **Edge Detector**: A hardware gadget used to tell whether a 0 goes to a 1 or a 1 to a zero that we abstract as a function to tell where a transition is
- **Why 11 in Manchester**: Because the receiver gets locked in after a variable number of preamble bits and so counting does not work, need a pattern in data
- **Eye Pattern**: A way to superimpose arbitrary bit sequences *on screen* as opposed to sending them in parallel

Eye Patterns



- In a perfect system, we will have a well-defined eye. Should sample at center of eye. Nice visual test of line quality.

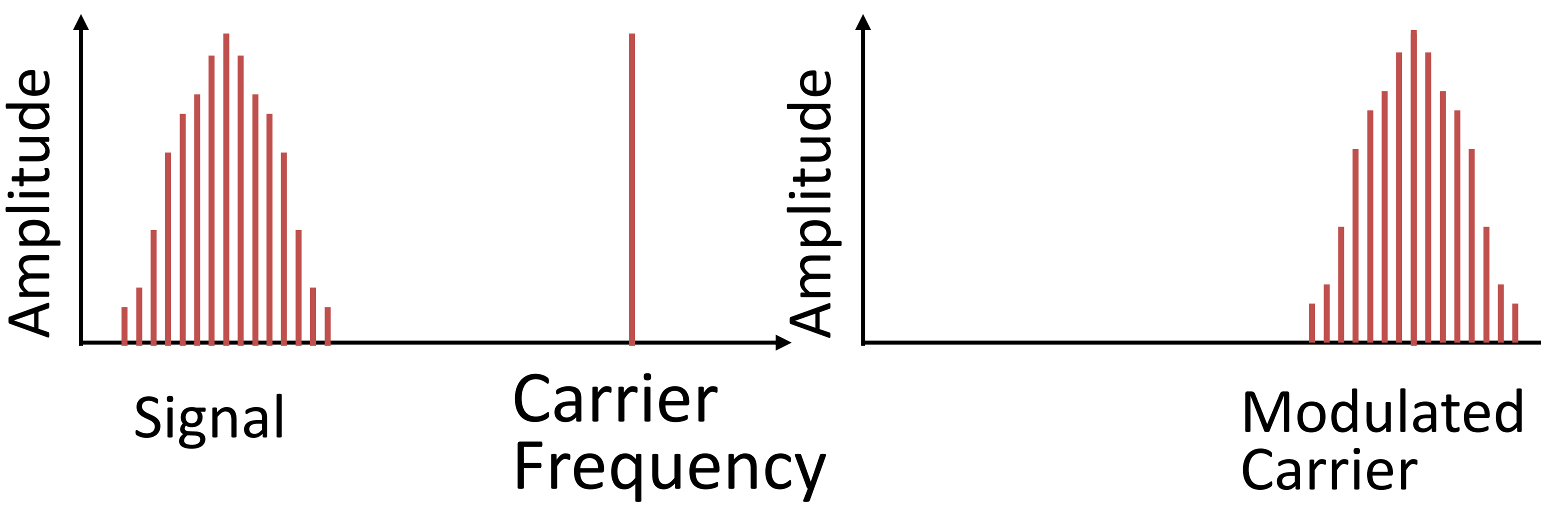
Broadband Coding

- (So far) Baseband Coding using energy levels such as voltage or light. In ***broadband coding*** information is ***modulated*** on a carrier wave of a certain frequency. Used by modems.
- Modulation refers to changing the properties of the carrier to convey the required information.
- Frequency Shift Keying (FSK): high frequency encodes a 1 and low a 0, Amplitude Shift Keying (ASK): high amplitude encodes a 1. QAM (multiple amplitudes and phases)

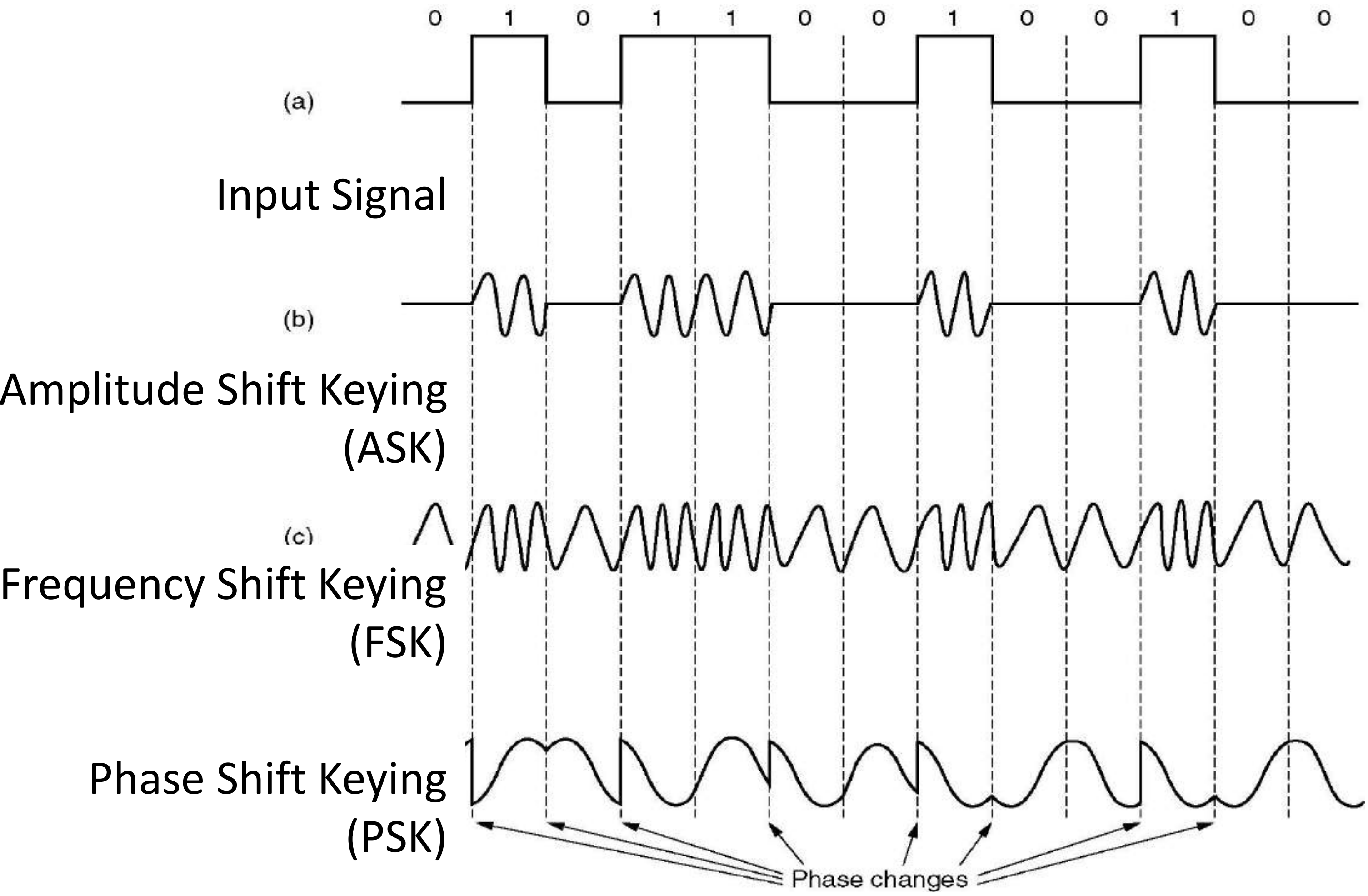
Baseband versus Broadband

Baseband modulation: send the "bare" signal
- E.g. +5 Volts for 1, -5 Volts for 0
- All signals fall in the same frequency range

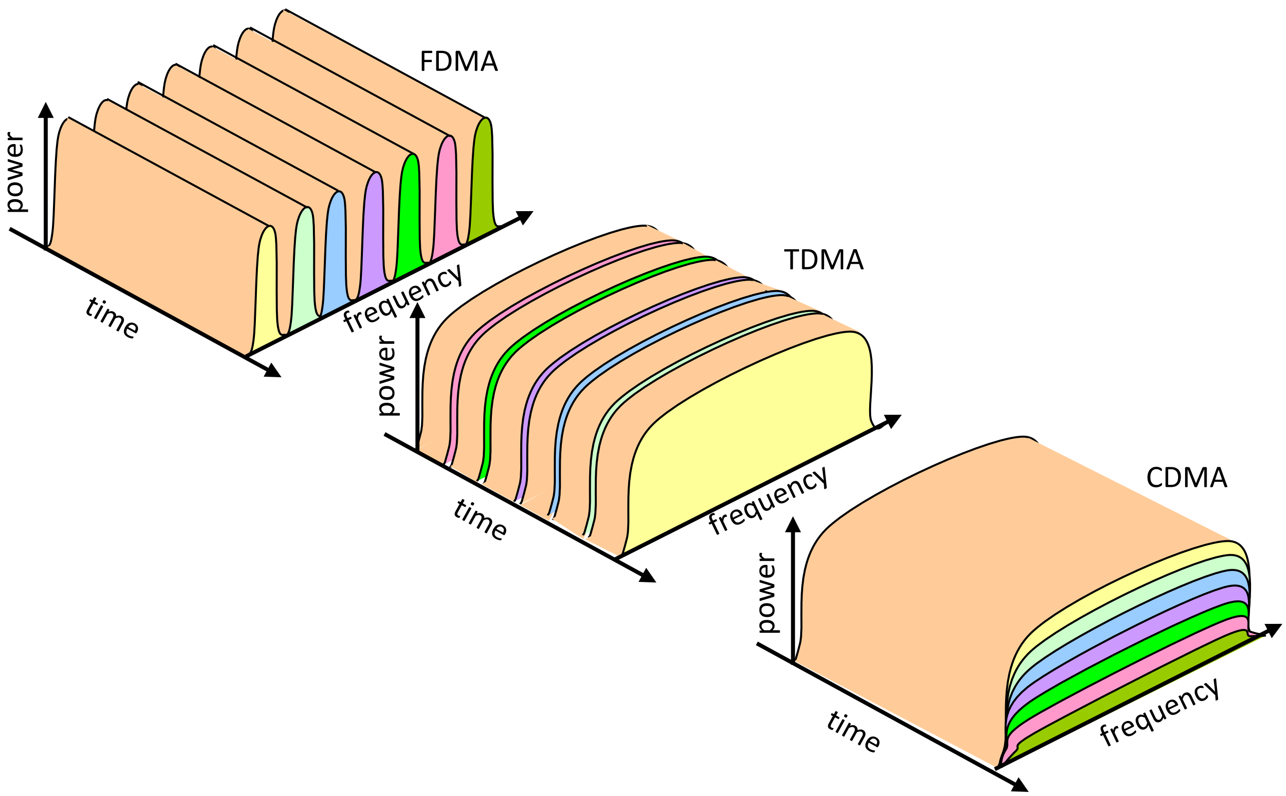
Broadband modulation
- Use the signal to modulate a high frequency signal (carrier).
- Can be viewed as the product of the two signals



Digital Modulation Methods (from Tanenbaum)

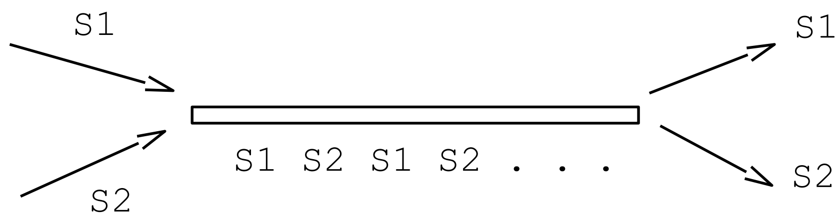


Three Levers for Modulation

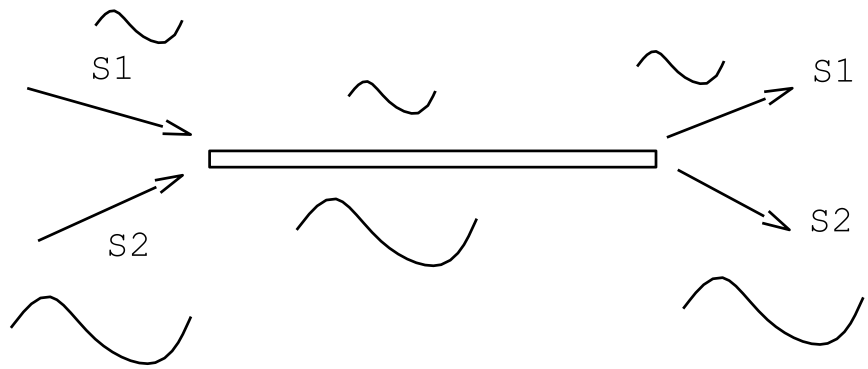


Courtesy Takashi Inoue

MULTIPLEXING (SHARING)



TIME DIVISION MULTIPLEXING (TDM)



FREQUENCY DIVISION MULTIPLEXING (FDM)

and Wavelength division multiplexing for fiber

Summary of Lecture 3

- **Clock Synchronization**: Getting receiver clock in synch with sender transmission using transitions guaranteed a by a code
- **Coding Schemes**: Asynchronous (transitions at start and end, small number of bits) versus Synchronous (many transitions at start and guaranteed transitions in data as well.
- **Real engineering**: Phase Locked loops not programs we used, and eye patterns to visualize
- **Broadband coding**: Modulate 1's and 0's on a *high frequency carrier wave* in wireless and cable modems