Discussion 1

* Interface
  + Communicate vertically (e.g., data link sends data to physical layer through an interface)
* Protocol
  + Communicate horizontally (e.g., communicate between 2 TCP apps)
  + Protocol between 2 layers uses interfaces of previous layer to implement
* Strict layering
  + Information between layers must be passed through an interface
  + Do not read another layer’s header
* Layers
  + Application
  + Transport
    - TCP
    - Guarantee all info is reliably sent over network
    - Resend dropped data
    - Establishes a reliable connection between 2 apps
    - IN -> QUEUE 1 … network … QUEUE 2 -> OUT
  + Routing
    - Given an IP address (ultimate destination, larger area), knows which path to take
    - Recomputes the path if a node goes down
    - Given IP addresses A and C, we want to go from A to C
    - Routing layer computes that we need to go through B first
    - Places “dest C, src A” in the header
    - Pass next hop “B” to data link through the interface
  + Data link
    - Low-level routing – Only knows info about 1 hop on the network; knows the hardware
    - Mac address corresponds to the medium data can be sent through (e.g., wire)
    - Knows which wire to send the bits along
    - Places “dest B, src A” into header (it got B from routing layer)
    - Node B’s data link layer will see that dest is B, so it will remove the data link header and pass the data to routing
    - Routing will compute next node and place “dest C, src A” into the header and pass next hop “C” to data link layer
  + Physical
    - The edge between 2 nodes (wire (ethernet), wireless (wifi), light (fiber))

Physical

* Fundamental issues
  + Nyquist (signal units) – each signal unit can be sent with freq 2f if f is the freq of 1 signal unit. How close together can we send each signal?
  + Shannon (bits) – Can send B log (1 + S/2N) bps. How close the voltages can be before noise makes them ambiguous?
  + Input signal (discrete) != output signal (continuous)
  + Most channels don’t distort sine waves too much though
    - So we can pass any input as sines through Fourier
  + Most sine wave distortions are shifting and scaling, which are manageable
  + Each channel does the same distortions to all signals
  + Signal ---Fourier---> sines ----channel---> sines 2.0 (shifted and scaled) ---inverse Fourier---> signal 2.0
  + Bandwidth (Hz) == width of freqs not attenuated
  + More bandwidth -> more harmonics -> more sine waves -> better reconstruction of the original signal
* Encoding and decoding issues
  + How to differentiate 0s and 1s (freq or ampltidue?)

Discussion 2

* Manchester encoding
  + Sync encoding
  + Initially clocks not in sync
  + Need some preamble to sync up the clocks
  + 11111..1 could be confused as 0000..0 when clocks are not in sync (and vice versa)
  + Using 101010 -> No matter where we sample (no matter what our clock is), we will read the preamble due to no extra transitions in the bit pattern, and our clocks can sync up.
    - Send an extra 11 to indicate we are done with the preamble phase (once clocks are now in sync)
  + How many preamble 10 bits?
    - Probabilistic proofs -> Send n 10 bits will have high enough probability to be detected
* Phase Locked Loops
  + In HW, algo detected transitions where they shouldn’t be one and added lag offset
  + But in practice, this is bad due to noise transitions
  + Look at distance between expected and observed and only shift a little bit according to how far it is (clock moves a bit toward detected transitions. If we are actually off, we are going to see repeated transitions and we will take smaller and smaller steps toward right phase)
* Eye pattern