Lecture 1

* Routing
  + Paths
* Data Links – physical layer
  + Between 2 nodes (error detection, etc.)
* Narrow physical layer restriction:
  + Need to send data in small bits
* DNS – directory service -> name to ip address
* Network header
  + Routing info (all intermediary routers)
* Data link header
  + Only info about next router
* Physical layer
  + No header, splits DPDU (?) (frame) into bits
* At destination node, data link header is taken off, then the network header. Then, it sees final destination is this router and proceeds to take off other headers
* Data links – fiber, wireless, cable, Bluetooth, etc.
* TCP vs. UDP transport layer
  + UDP – no resending/retransmission
  + TCP – more reliable
* TCP abstraction
  + Sender and receiver have 2 connected queues and data reliably is sent from sender to receiver queue. No packets.
  + Uses data link to send packets between 2 routers, which sends bits at physical layer

Lecture 2

* TCP
  + Queue abstraction
* IP layer
  + IP header places source and destination
  + If IP addresses match, then we are at the right destination
  + Else, figure out the next router
* Data link
  + From mac address x to y
* Mac address
  + Unique 48 bit addresses uniquely identify everyone on a link
  + E.g., x->y z->l, y,z are mac addresses for the same router
* IP address
  + Temporarily assigned to device
* Physical
  + Bits
* RPC (remote procedure calls) vs TCP (Queue abstraction)
* Socket = queue
* Transmission
  + Bandwidth
    - Range of frequencies a particular signal is created from (infinite sum of which sine wave freqs)
  + Strict layering
    - Only read headers at current layer (e.g., IP only reads IP header)
    - Information can only be passed across interfaces
  + Physical layer
    - Frequency shifting vs amplitude shifting (baseband)
    - Bits to coded bits to add transitions
    - Send bits to modem to convert to energy
  + Channel
    - Physical medium to transmit from sender to receiver
    - A channel does not change frequency of sine waves (only maybe the amplitude and phase)
  + Any periodic function can be written as an infinite sum of sin waves with diff freqs
  + More harmonics/bandwidth -> better bit recovery
  + Harmonic = multiple of the fundamental frequency
  + Channels are sluggish (take time to respond to a change in the signal)
  + Channels have noise, resulting in a change in amplitude
  + Sample at middle points
  + Two limits
    - Bandwidth (Nyquist rate) – limit on symbols (signals)
      * Input: square wave, output: falling sine wave
        + When can be send the second signal such that it does not interfere with previous signal?

Bandwidth of input signal is 1/T

You can theoretically send signals every T/2 seconds (Send at frequency: 2 \* bandwidth = 2/T)

Max rate of sending symbols, not bits (baud rate)

* + - * Why not send multiple bits per symbol (6v – 11, 4v – 10, …)
        + Noise
    - Noise (Shannon) – limit on bits
      * Maximum number of noise to tolerate to distinguish 6V and 4V is 1V, which limits the number of bits per symbol
      * Shannon limit applies to any channel (wire, wireless, etc.)
      * S – max signal
      * N – noise
      * Signals must be spaced 2\*N
      * B – bandwidth
      * Shannon limit is B \* log (1 + S/2N) bits per second
* Channel needs to pass a range of frequencies in a range from highest to 0Hz
  + 010101010101 – highest
  + 000000000000 – 0 Hz
  + Non-periodic bit patterns

Lecture 4

Diagram

Description automatically generated

* Physical layer
  + Coding sublayer
    - Add transition bits
    - Initial training bits used to sync clock at the beginning
      * Physical clocks drift over time -> we need transition bits to keep them in sync
    - Example
      * Diagram

        Description automatically generated
      * Transition bits
        + Start detected by rising signal (edge detector)
        + Sample every 1 bit after that
      * Initial transition bit after each character being transmitted
    - Manchester
      * Sends double the number of bits (if Nyquist limit is B, we send B/2 amount of useful data)
      * 1->0 : 1
      * 0->1 : 0
      * Do not look at the signal for about 20-30%.
        + E.g., to send data 1111111
        + We need, 10-10-10-10-10-10-10
        + We must ignore the 01 transitions
        + No need to look at a specific point (middle of a peak), instead we watch for a transition from 0 to 1 or 1 to 0
      * The data 0101011 indicates start of the bit pattern (preamble phase)
      * Looking for rising and falling edges is dangerous due to noise
    - Phase Locked Loops (?)
    - Eye pattern (?)
      * Superimpose 2 signals together
    - Multiplexing (n times more bandwidth by sending more data at once)
      * Time
      * Frequency
      * Wavelength (visible light)
        + Send red, blue, etc. signals at the same time at the sender end
      * CDMA
        + Different frequencies at different times
    - Async vs sync
      * Async: Sender and receiver don’t have the same clock. Need start and stop bits and an accepted number of data bits (e.g., 1 byte)
      * Sync: Both the sender and receiver have synchronized clocks. Need transitions to ensure clocks do not drift.
  + Decoding sublayer
    - Clock recovery
  + Media transmission sublayer
    - Convert digital bits to energy (electricity or light)
  + Signal transmission sublayer
    - What the channel does to the signal (Nyquist and Shannon limits)

Lecture 5

* [www.google.com](http://www.google.com) -> DNS -> IP address
  + IP -> 32 bits
* HTTP atop TCP
  + HTTP header
* TCP adds sequence numbers to order data when received (after retransmission)
* Routing
  + Each router knows the next hop (forwarding table) and passes that to data link
* Data link
  + MAC in header
* RS-232
  + Interface between CPU and modem connected to twisted pair
* Cable
  + Twisted pair (last mile)
    - Signal is difference between both pairs -> signal gets cancelled out
    - Used to be low battery and low throughput and long distance
      * Now higher throughput less distance (cat 5/6 twisted pair)
    - Cable = coaxial
  + Cable is good at sending data to you
  + Coaxial
    - Used bw buildings
  + Twisted pair
    - Local
* Fibre
  + A lot of bandwidth
  + Each fibre cable is unidirectional
  + Total internal reflection – no refraction into the fibre cable
  + Multimode fibre
    - One part reflects (long path)
    - One part doesn’t (short path)
    - Limits bandwidth
  + Single mode -- Thin fibre
    - Wave will not be spread into parts... just one thin palse
    - Higher bandwidth because there is no dispersion (wave is not spread out)
  + Use monochromatic lasers so that the light waves do not interfere
* Wireless
  + Spectrum
    - Radio waves
      * Omnidirectional – goes in all direction
      * Goes through obstacles
    - Satiates, microwaves
      * Directional
      * Absorbed by obstacles
      * Signal to satellite from dish will be unidirectional
      * The signal back from satellite to its destination will have a larger direction range
      * Geosynchronous
        + Satellite should rotate around earth at same speed earth rotates itself

Antenna only needs to point in one directions

* + - * + Satellite needs to be far to achieve geosynchrony

Huge latency

Not as much bandwidth (cannot keep too many satellites in the sky)

* + - * Right of way
        + Satellites are not on the ground (bypass wires, …)
      * Low orbiting satellites
        + More latency
        + Cover earth with low orbiting satellites… handoff connection to closest satellite as one rotates away from dish
        + (satellites are not geosynchronous since they are too close)
        + Disadvantage: You need a lot of satellites that span around earth
* FMD = freq division multiplex
* TDM = time
* WDM = wavelength
* Interconnection
  + Router
    - More latency
    - Can redirect if a link goes down
    - Check for errors
  + Repeater (“hubs”)
    - Less latency
    - Cheaper
    - Physical layer
    - “Blindly” repeat bits
    - Fibre stays at the repeater level

Data link

* Hop by hop should be an optimization (so that routers retransmit before TCP times out… not worth it if links are reliable)
* End-to-end ack is required to guarantee successful transmission
* Why framing
  + Headers
    - Individual bits cannot have headers (which would include destination, source, etc.)
  + Multiplexing
    - 2 senders and 2 receivers share a link
      * Each sender sends 1 frame at a time
    - Need frames with headers to multiplex (need discrete units to multiplex)
  + Error recovery
    - Easier to retransmit frames rether than the entire data stream
  + Frame boundaries
* Sol 1 : High Level Data-Link Control (HDLC)
  + Flag: 01111110
    - Start and stop delimeter
  + Bit stuffing
    - If data has pattern 11111, then add a 0
    - So, 111110 -> 11111
    - So, 1111100 -> 111110
    - Stuff an extra 0 after 5 ones
* Sol 2 : Ethernet
  + Inter-frame gap
    - Period of silence (0 V)
    - For 96 bit times
  + Period of preamble (56 bits)
    - 10101010…
    - For receiver to synchronize clock with sender
  + Start of frame delimeter
    - 10101011
  + Data
    - Includes header (dest, src, …)
  + Inter-frame gap
    - Period of silence (0V)
* Too small frames
  + Error rate
* Too large frames
  + Higher chance of error and need to recover a larger number of bits
* Usual amount: 64 – 1500 bytes
* If flag is 01111110, should add 0 after 5 1’s, not after 011111
  + First case: 0111111111110 -> 011111011111010 – transmit – decode -> 011111111110 (good)
  + Second case: 01111101111110 -> deflagger will see 01111110 as a EOF delimeter -> data after this pattern is lost
* If flag is 01010101, cannot stuff 1 after 010101
  + Data = 0101
  + Transmitted = 01010101 0101 01010101
    - * + -------------
        + Receiver sees this as a end flag -> data lost
* Sol 3 : Change the physical API
  + E.g., 4-5 encoding gives 0000 to 1111 data values, but there are many other bit patterns not used. Use those as control characters (SOF and EOF)
  + Add a third symbol at the physical layer (0 -> 0V, 1 -> 1V, frame boundary -> 5V?)
* Errors
  + Random error
    - 0 or 1 corrupted
    - Noise corrupts 1 bit
  + Burst error of length n
    - Bit 1 and n are corrupted. Bits in between may be corrupted
  + H(M) is 32 bits -> 2^32 possibilities -> 1/2^32 chance of getting the same hash for M and M’ such that M != M’ -> undetected error
    - Assuming errors and hash are random
  + Error correction is more expensive, not worth it when retransmission is few ms
  + xxxxxxx -> xxxxxxxppp -> ppp -> needed addition to make xxxxxxxppp divisible by 7 -> receiver will ensure xxxxxxxppp is divisible by 7 -> extract xxxxxxx as data bits (we add 3 bits because we add up to 6 to make num divisible by 7 -> 110 is 3 bits.
  + CRC uses mod 2 arithmetic
    - Addition = subtraction = xor
      * 10001 + 11100 = 01101
    - Multiply
      * Shift + XOR
      * 10111 \* 11 = 11001 (shifting just as normal mult -> use XOR when adding the terms)
    - Division
      * M = 1001, G = 101 => r = 3 (num bits of G) [If first bit of G is 1, then the remainder will be r-1 bits)
      * M’ = 100100 (add r-1 = 2 bits to M)
      * 1011
      * \_\_\_\_\_\_\_
      * 101 | 100100
      * 101
      * 0011
      * 000
      * 110
      * 101
      * 0110
      * 101
      * 011
      * Remainder t = 11 (CRC). [you no longer take G-t as CRC (only t)]
      * Final message = 100111 (which is mod 2 divisible by G=101

Each codeword has a set of valid codewords.

The idea is to maximize hamming distance between two adjacent valid codewords.

Parity has a hamming distance of 2

0 -> 000

1 -> 111

Hamming distance is 3

Hamming distance – 1 == resilience to error

Hamming distance of d:

C1 -------- d ---------- C2

|

|

|

d

|

|

|

C3

Any error of d – 1 bits on a valid codeword will result in an invalid codeword, so it will be dropped. => All d – 1 bit errors will be detected

Hamming distance of CRC is 4 (all 1-bit and 2-bit errors are detected… all odd bit errors are detected (3), but 4 bit errors are not detected)

Error correction: If hamming distance is 2d + 1, we can correct all d bit errors by going to the closest valid codeword. Error of > d bits will result in an incorrect correction