CSE 461 - Module 5: Dealing with Errors II

Network Errors: Assumptions (for now)

- Our goal is eliable transmission of messages
  - Receiver delivers a single copy of each received message to the app, in order
    - (We're assuming some lower layer deals with bit errors, so we deal only with messsage drops and reordering)
- We're worried only about correctness, not performance (yet)
- Three parts to reliability:
  - [Mainly sender] Make sure at least one copy of the message gets to the receiver
  - [Mainly receiver] Make sure to detect and drop duplicates
  - [Inherent, for now] Make sure you deliver messages in order
    -

ARQ (Automatic Repeat Request)

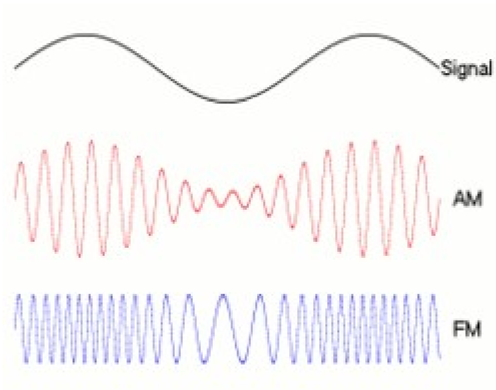
- Positive Acknowledgements
  - The sender knows:
    - What it can sense (e.g., a received ACK)
    - What it can deduce from the causal chain implied by a correct implementation of the protocol
      - The receiver sends an ACK only if it receives a message
      - I got an ACK
      - Therefore the receiver received a message
- Suppose sender observes
  `send, send, send, ACK, send, ACK, send, ACK`
  What has receiver seen?
- How do we fix this?

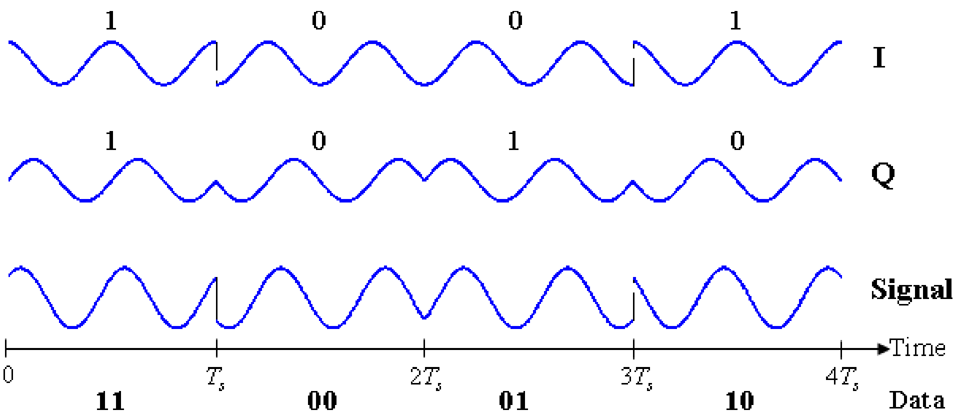
Examples

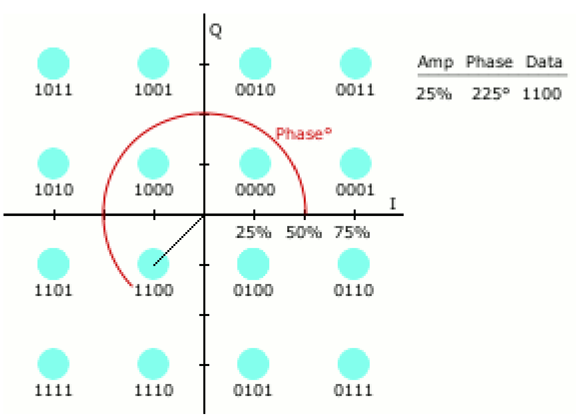
- **TCP**
  - What are the characterisitics of “the channel”
  - How should ARQ work?
- **802.11** (wireless)
  - What are the characteristics of “the channel”
  - How should ARQ work?
- **802.3** (wired Ethernet)
  - What are the characteristics of the channel?
  - How should ARQ work?

Bit Errors

Modulation

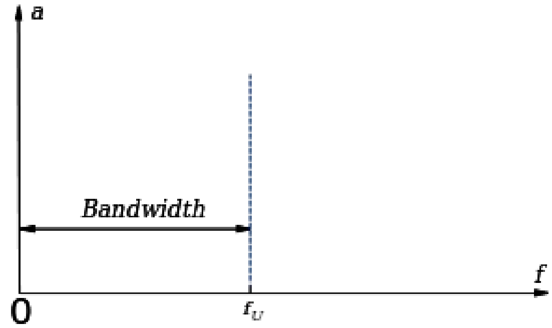


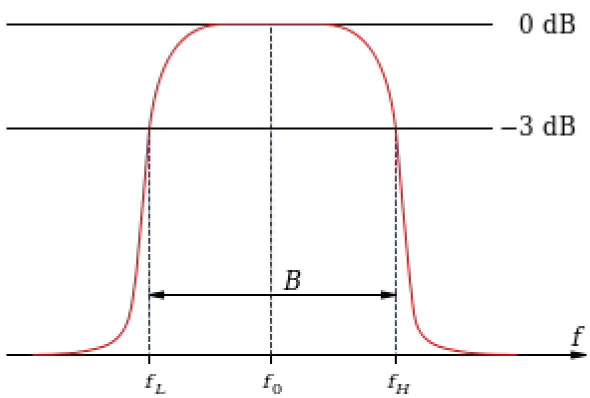




Communication channels

- attenuation
- noise
- limited bandwidth

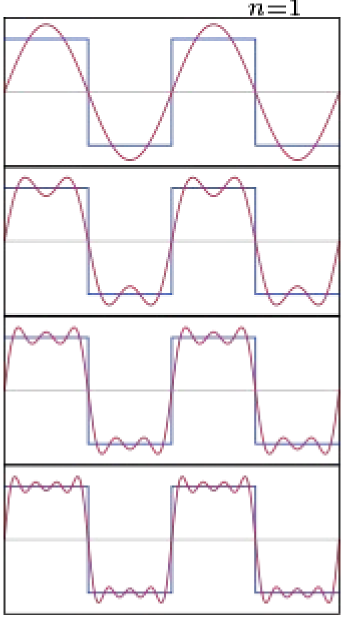




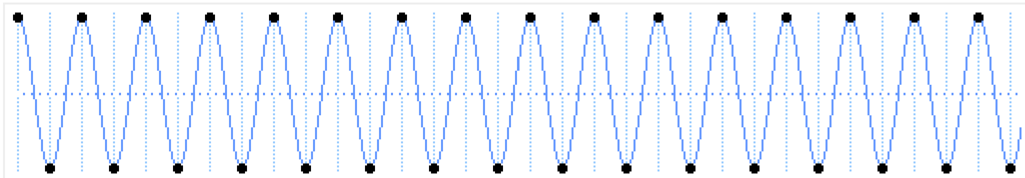
Theoretical Limits

- **Fourier theorem**

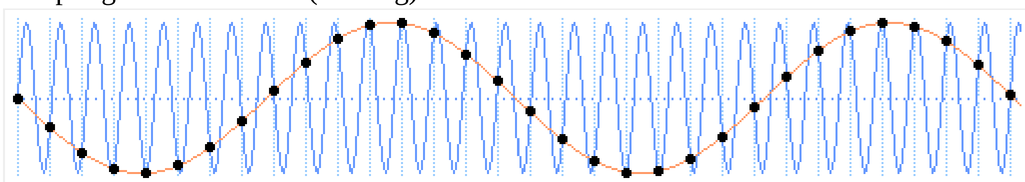
$$s_N(x) = \frac{a_0}{2} + \sum_{n=1}^{N} A_n \cdot \sin\left(\frac{2\pi n x}{P} + \phi_n\right), \quad \text{for integer } N \geq 1.$$



- **Nyquist Limit**
  - If signal has bandwidth B, the maximum symbol rate (i.e. ,noiseless channel) is 2B
    - Sampling at rate 2B is sufficient to reconstruct the signal at all points, so further samples are redundant
    - Sampling at the limit



    - Sampling below the limit (aliasing)



- **Shannon Capacity Theorem**
  - No matter how many bits/symbol, for a channel with bandwidth B the maximum bit rate (capacity) is $B \log_2(1 + S/N)$ where S is the received signal strength and N is the noise.
    - Higher power => higher bit rate or lower bit error rate (BER)
      Lower power => lower bit rate and/or higher BER

Clocking

- Difficult/impossible to have sender and receiver clocks run at exactly the same rate
  - They might run at the same rate for a little while
  - You might need to resynchronize them
- At the extreme, you might synchronize on every sent bit
  - Force an observable transition on every bit (e.g., 0 => low → high; 1 => high → low)
- 4B/5B
  - no change in signal → 0; change in signal → 1
  - Now want to make sure you send a 1 often enough
    - What if the data is a long sequence of 0's?
  - Idea: send 5 bits to represent 4 bits of data
    - Choose 16 of the 32 5-bit combinations that have enough 1's
    - Never send 00000, for instance
- Preambles
  - Some schemes send non-data bits before a frame whose goal is to allow the receiver to lock onto the

senders clock rate

- 802.11:
  - preamble contains 128 bit (essentially random) string for sync'ing
- Ethernet:
  - 7 bytes of 10101010

Error Detection and Correction (TW 3.2)

- Key idea:
  - send k+n bits to represent k bits of data
- Only 2^k valid codes out of the 2^k possible bit strings
  - If you get something that the sender would never send, it's an error
  - If you get something that the sender might have sent, t's not an error, so long as the channel can't produce “too many” bit errors
- Systematic code: k of the bits are the data, and n are function of the data
  - Sender computes the n bits based on the k data bits
  - Receiver computes what function based on the k data bits it actually received, and compares that value to the n bits it actually received
    - If they don't match, there's an error
- Error detection schemes
  - Parity
    - 1-bit odd parity: add a single bit to each block so that total number of 1 bits is odd
      - 01100000 1
01100010 0
      - What is detected? What isn't?
  - Internet checksum
    - (Basically) sum the words of the message and send that result at the end
  - Cyclic redundancy code (CRC)
    - Think of the message as a very big integer
    - Send additional (low-order) bits so that the big integer is evenly divisible by some agreed upon integer
    - Why? Analyzable; good error detection properties (e.g., burst errors); easily implemented
- Error correction
  - **Hamming Distance**
    - Minimum number of bit flips required to go from one legal code word to another legal code word

        - Example: 0 → 00; 1 → 11
        - Example: 0 → 000; 1 → 111
- If the maximum possible number of bit errors is less than half the Hamming distance of the code, then every received bit string will be closest to a unique valid code
    - That valid code is what was sent (under the assumption about the number of errors)
        - Example: Hamming codes
            - Will do in sections