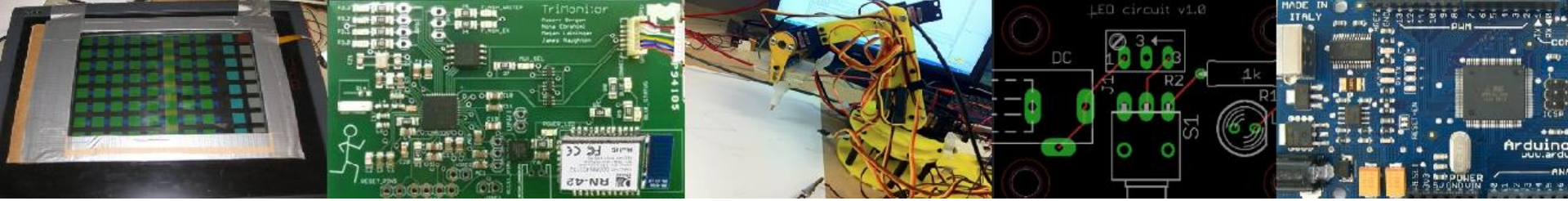
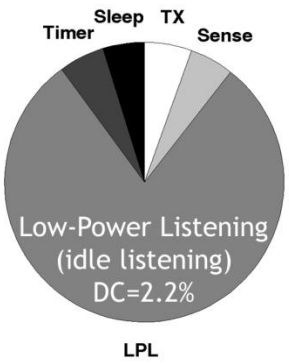
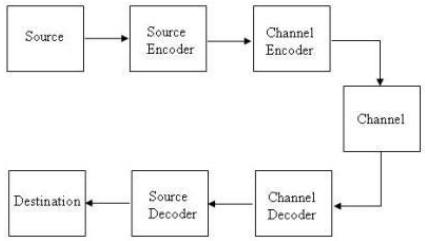


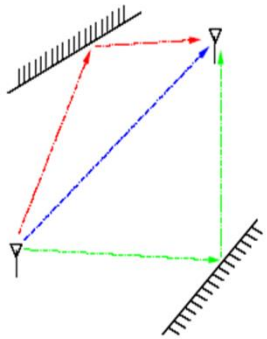
EECS 473
Advanced Embedded Systems

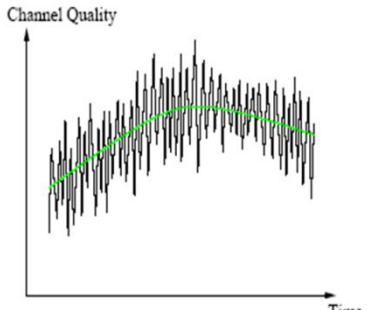
Lecture 13

Start on Wireless









Feedback and Upcoming

- Quick survey
  - Who has gotten their PCB out?
- PCBs
  - All are to be out by 10/31.
  - Strongly suggest you get them reviewed
- MS2
  - Report due on Tuesday 11/7
  - Check-offs due by Wednesday 11/8 @6pm
  - MS2 meetings start on Thursday 11/9

Team status updates for Thursday

- Largest roadblock
- Thing you are most pleased with wrt your project.

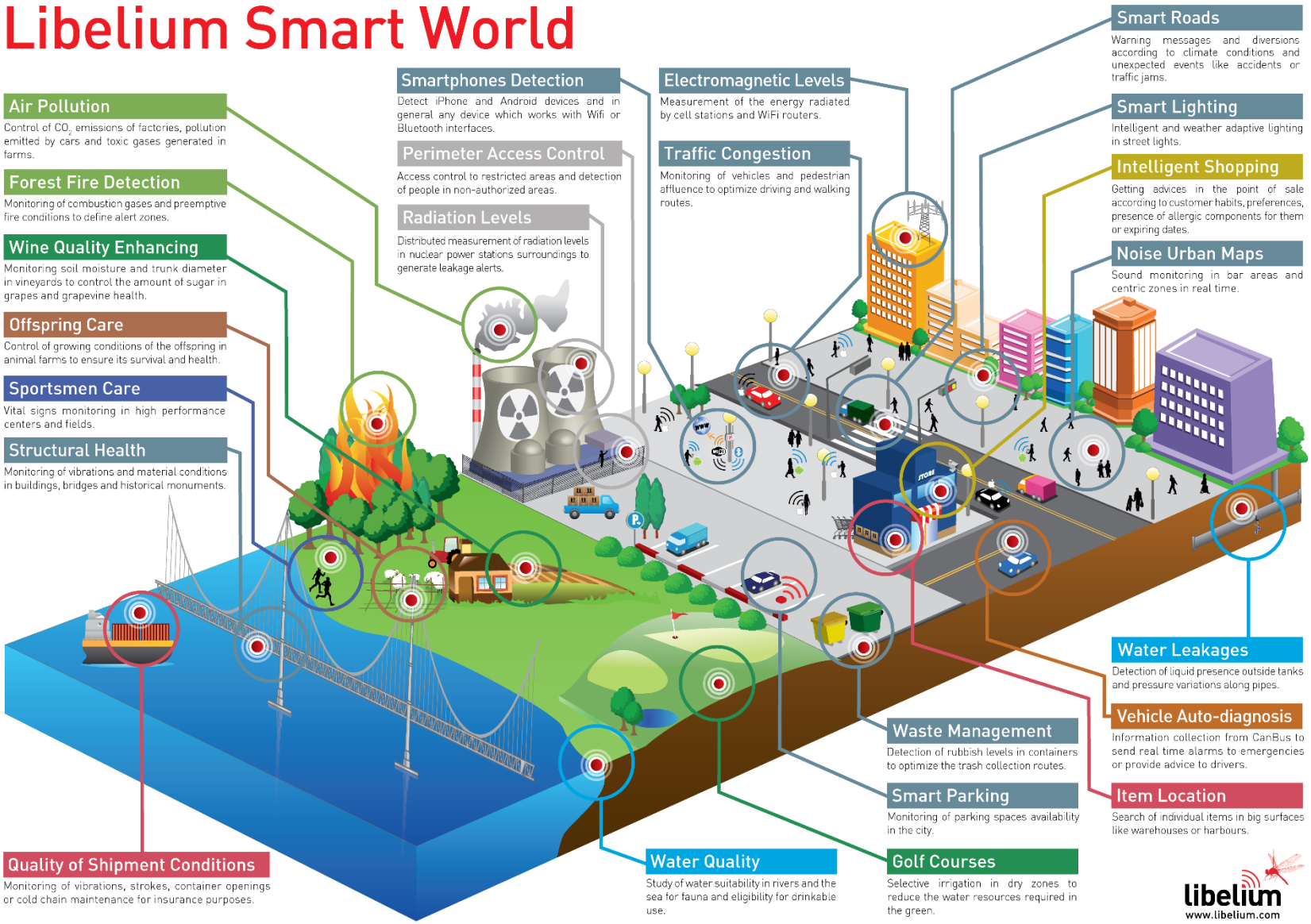
Wireless communications

- Next 2.5 lectures are going to cover wireless communication
  - Both theory and practice.
- If you've had a communication systems class, there will be some overlap
  - And we will be focusing on digital where we can
    - Though that's still a lot of analog.

Wireless and embedded?

- As should be obvious, modern embedded systems are tied closely to wireless communication.
  - Think about your projects.
- Applications include the home…

But certainly reach much farther



Lots of peer-to-peer wireless protocols

- **Wi-Fi** high-speed data with ranges of about 20-40 meters. Fairly high power consumption. If you want to get on the Internet, this is usually the way to go.
- **Bluetooth** is a global 2.4 GHz personal area network for short-range wireless communication. Device-to-device file transfers, wireless speakers, and wireless headsets are common users. Range can get quite long with 5.0 (~200 meters)
  - **BLE** is a version of Bluetooth designed for lower-powered devices that use less data. To conserve power, BLE remains in sleep mode except when a connection is initiated. This makes it ideal for wearable fitness trackers and health monitors.
- **ZigBee** is (mostly) a 2.4 GHz mesh local area network (LAN) protocol built on 802.15.4. It was designed for building automation and still sees a lot of use there.
- **RFID** Allows passive (unpowered) devices to communicate.
  - **NFC** a protocol used for very close communication. If you wave your phone to pay for groceries, you're likely using NFC. Closely related to RFID.
- **LoRaWAN** is a long-range (1-15km), low-power, low-data rate (~1-50kbits/sec) protocol.
- **Z-Wave** is a priority protocol design for home automation. <30 meter range. Has some nice dev kits. Very low power.
- **SigFox, Ingenu, Weightless-(N, P, W), ANT, DigiMesh, MiWi, EnOcean, Dash7, WirelessHART.** Probably a lot more.

Mostly from https://iotdesignpro.com/articles/different-types-of-wireless-communication-protocols-for-iot

Cellular

- Rather than peer-to-peer, we can go through cell phone towers or even satellites.
  - If you want to go **far** the only other real option is shortwave radio (limited to about 300 baud)
- This used to be very expensive, but for small amounts of data, it may be viable.
  - For example, TruPhone will give you 125MB to use for 5 years for $19.00.
  - There are some variations out there (e.g. Cat-M1, G4)
    - Tradeoffs aren't hugely clear.

Two and half Lectures

- Start at the high-level
  - Overview by example: Zigbee/802.15.4
    - OSI model
      - MAC layer
- Go to low-level
  - Source & channel encoding
  - Multi-path issues
  - Modulation
  - Range

Outcomes: Things you should be able to answer after these lectures.

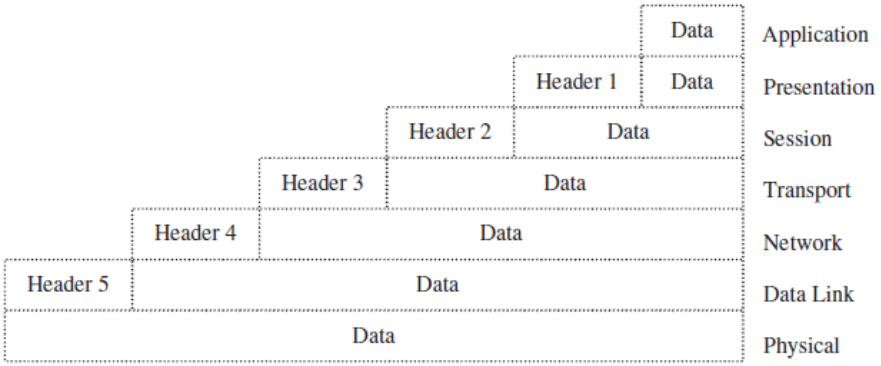
- Why might I choose the (lower bandwidth) 915MHz frequency over the 2.4GHz?
  - Related: Why are those the only bands I can pick?
  - Related: Why can shortwave radio in China reach the US? Why are their so many Cubs fans? (actually related)
- How do I compute "open space" radio distances?
  - How do I convert "open space" radio distances in a specification to indoor distances?
- How do I deal with a dropped packet?
- How much data can I hope to move over this channel?

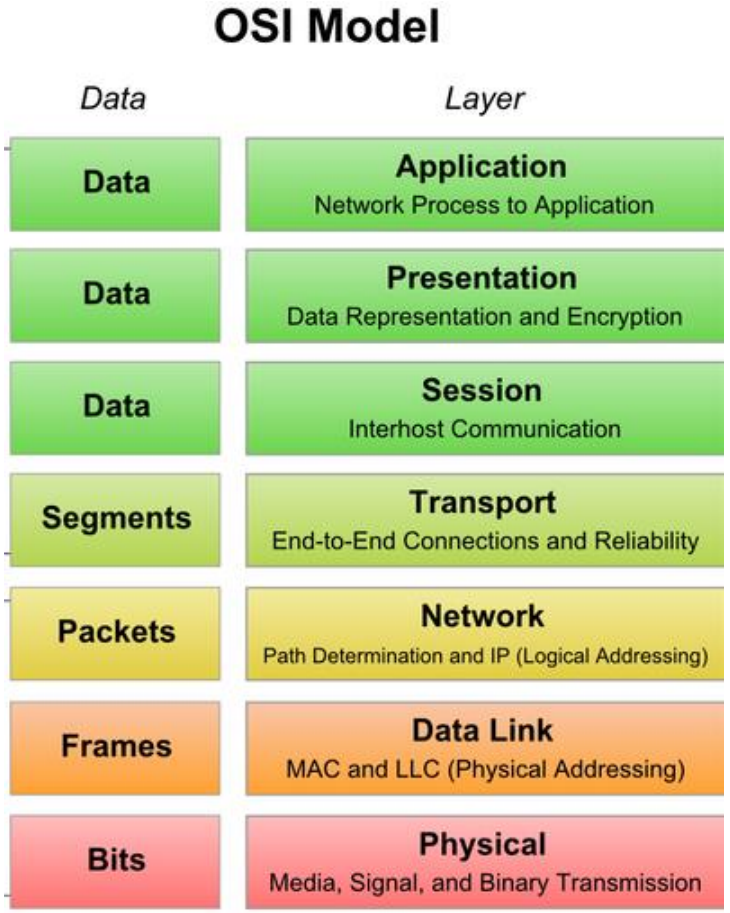
Zigbee

- ZigBee is an IEEE 802.15.4-based protocol
  - used to create personal area networks with small, low-power digital radios.
- Simpler and less expensive than Bluetooth or Wi-Fi.
- Used for wireless light switches, electrical meters with in-home-displays, etc.
- Transmission distances to 10–100 meters line-of-sight
  - Zigbee Pro can hit a mile
- Secure networking
  - (ZigBee networks are secured by 128 bit symmetric encryption keys.)
- ZigBee has a defined rate of 250 kbit/s, best suited for intermittent data transmissions from a sensor or input device.

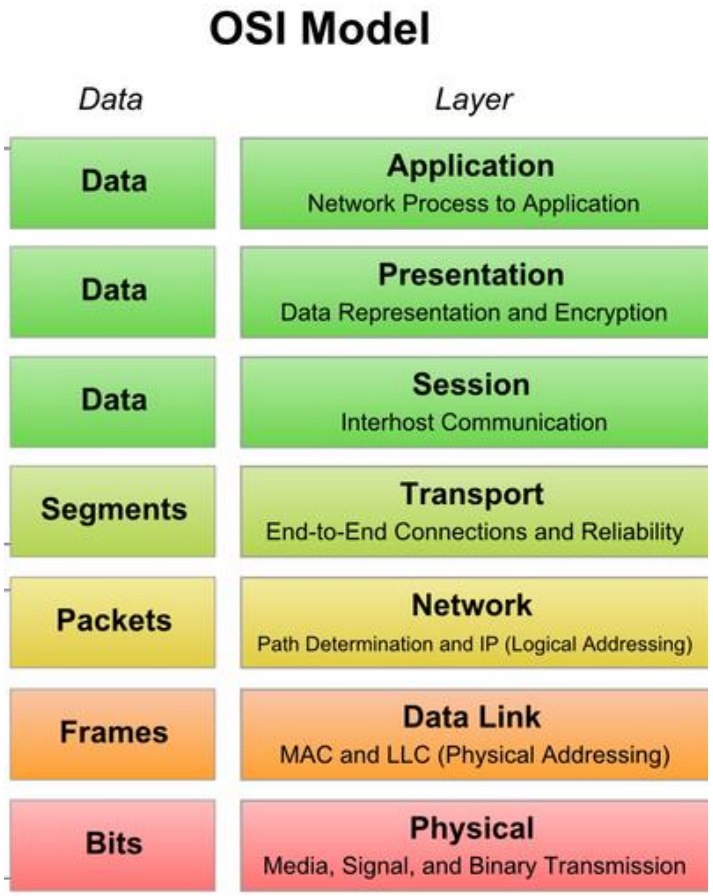
At the end of the day all wireless is just sending bits over the air.

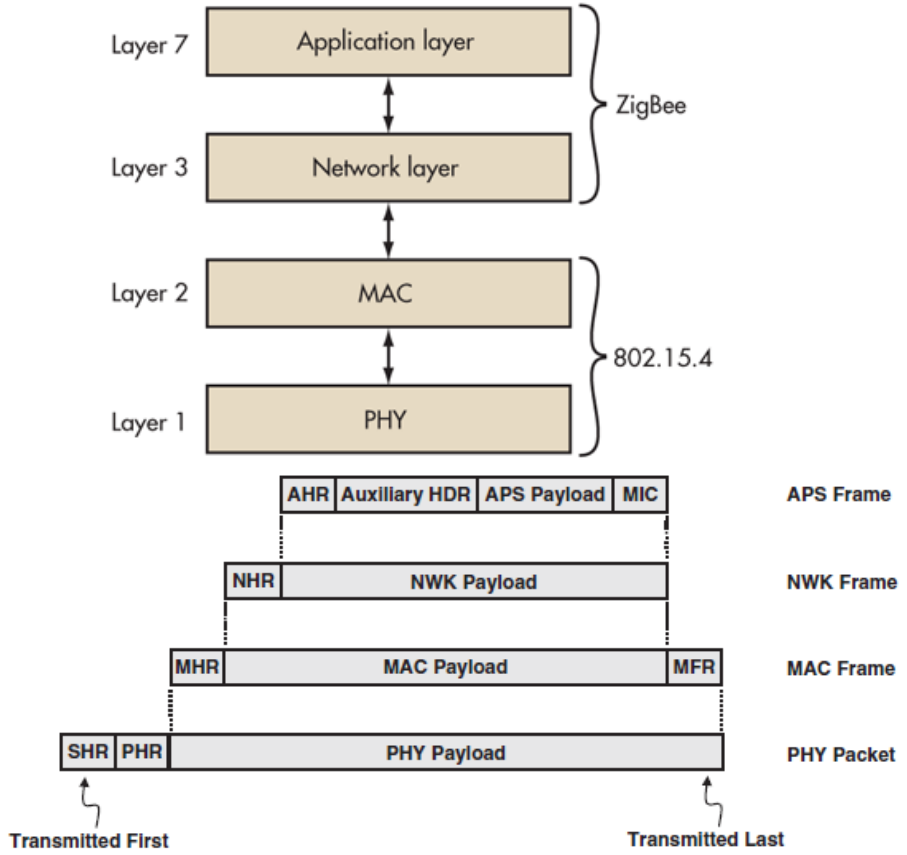
- Two issues:
  - How you send those bits (physical layer)
  - How you use those bits (everything else)
- We'll discuss both

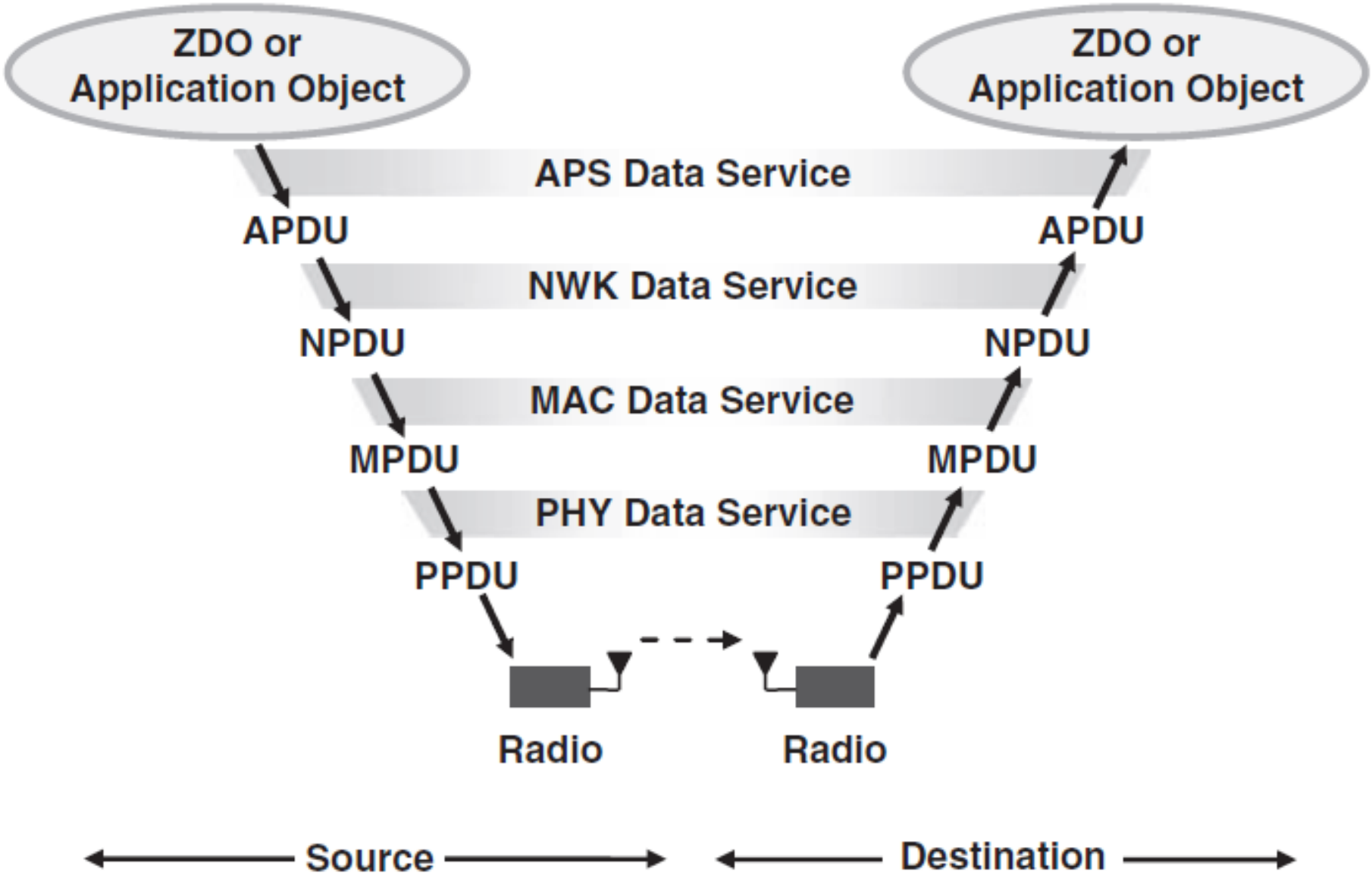




To minimize overhead, Zigbee skips some layers



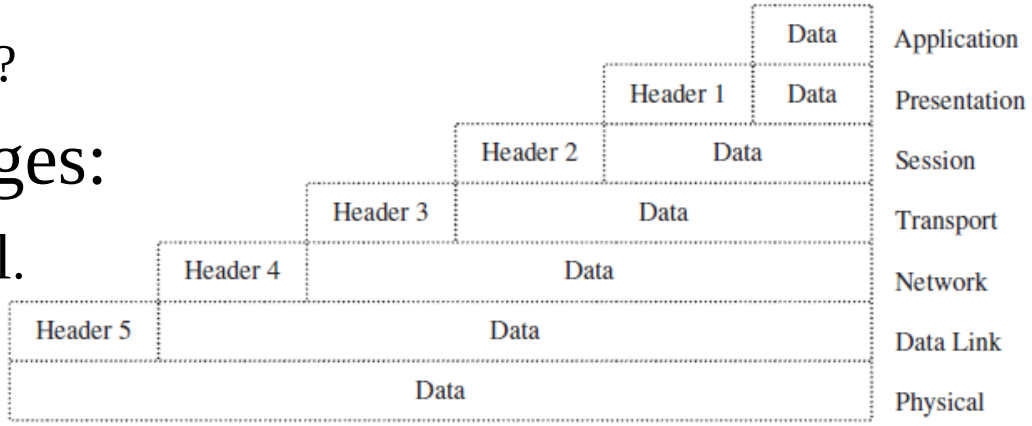




From: ***ZigBee Wireless Networks and Transceivers*** by Shahin Farahani

OSI basic idea

- Each layer adds some header information to address a specific problem.
  - What task on the target is this message related to?
    - A given sensing unit might have a lot of sensors for example.
  - What if we have a longer message than one frame?
    - What if one of those frames gets dropped?
- Example on next pages:
  - Data link to Physical.



MAC (data link) layer (802.15.4)

- Frame control basics:
  - What type of frame?
  - Security enabled?
  - Need to Ack?
- Frame control—frame size
  - Sender/receiver on same PAN?
  - Address size for source and destination (16 or 64 bit)
  - Which standard? (Frame version)

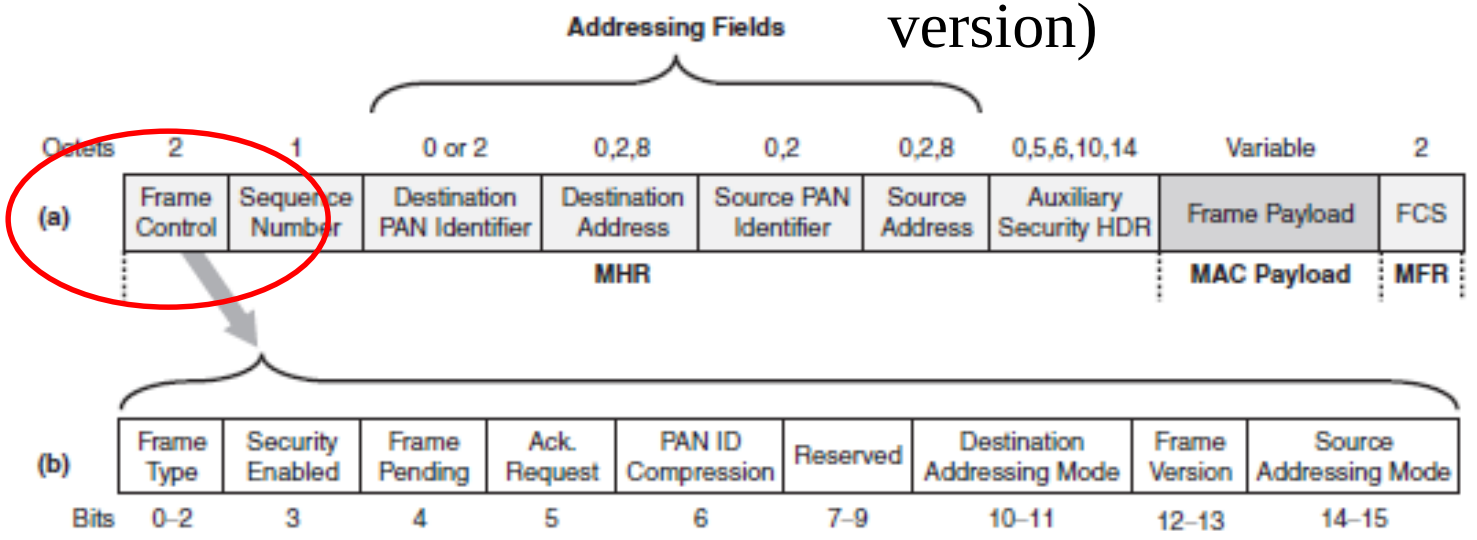
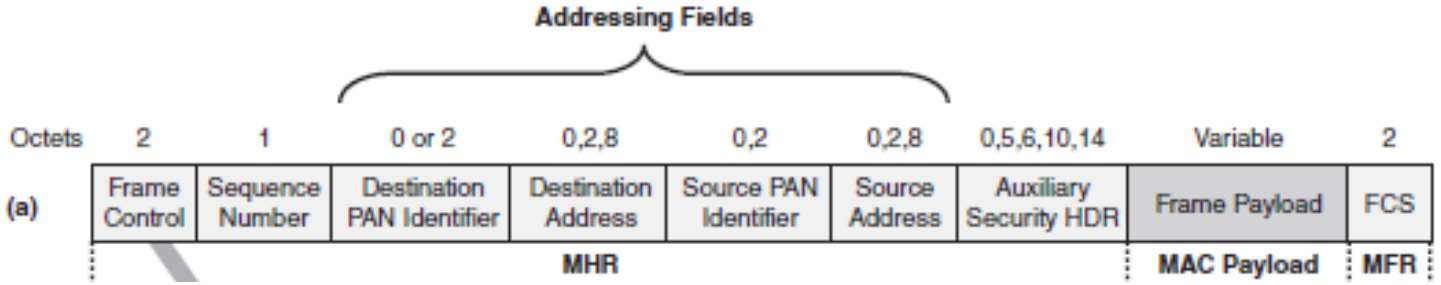


Figure 3.21: (a) General MAC Frame Format and (b) Details of the Frame Control Field

MAC layer (802.15.4)

- Sequence number
  - Used to reassemble packets that came out of order.
  - Or detect a resent packet
- PAN IDs and addresses are just what you'd think.
- Auxiliary Security header specifies encryption schemes
- FCS (Frame check sequence)
  - CRC for detecting errors.



Physical layer

Table 1.1: IEEE 802.15.4 Data Rates and Frequencies of Operation

	Frequency (MHz)	Number of Channels	Modulation	Chip Rate (Kchip/s)	Bit Rate (Kb/s)	Symbol Rate (Ksymbol/s)	Spreading Method
	868–868.6	1	BPSK	300	20	20	Binary DSSS
	902–928	10	BPSK	600	40	40	Binary DSSS
Optional	868–868.6	1	ASK	400	250	12.5	20-bit PSSS
	902–928	10	ASK	1600	250	50	5-bit PSSS
Optional	868–868.6	1	O-QPSK	400	100	25	16-array orthogonal
	902–928	10	O-QPSK	1000	250	62.5	16-array orthogonal
	2400–2483.5	16	O-QPSK	2000	250	62.5	16-array orthogonal

There is a lot about the physical layer to understand. We'll do some on Tuesday.

PHY layer

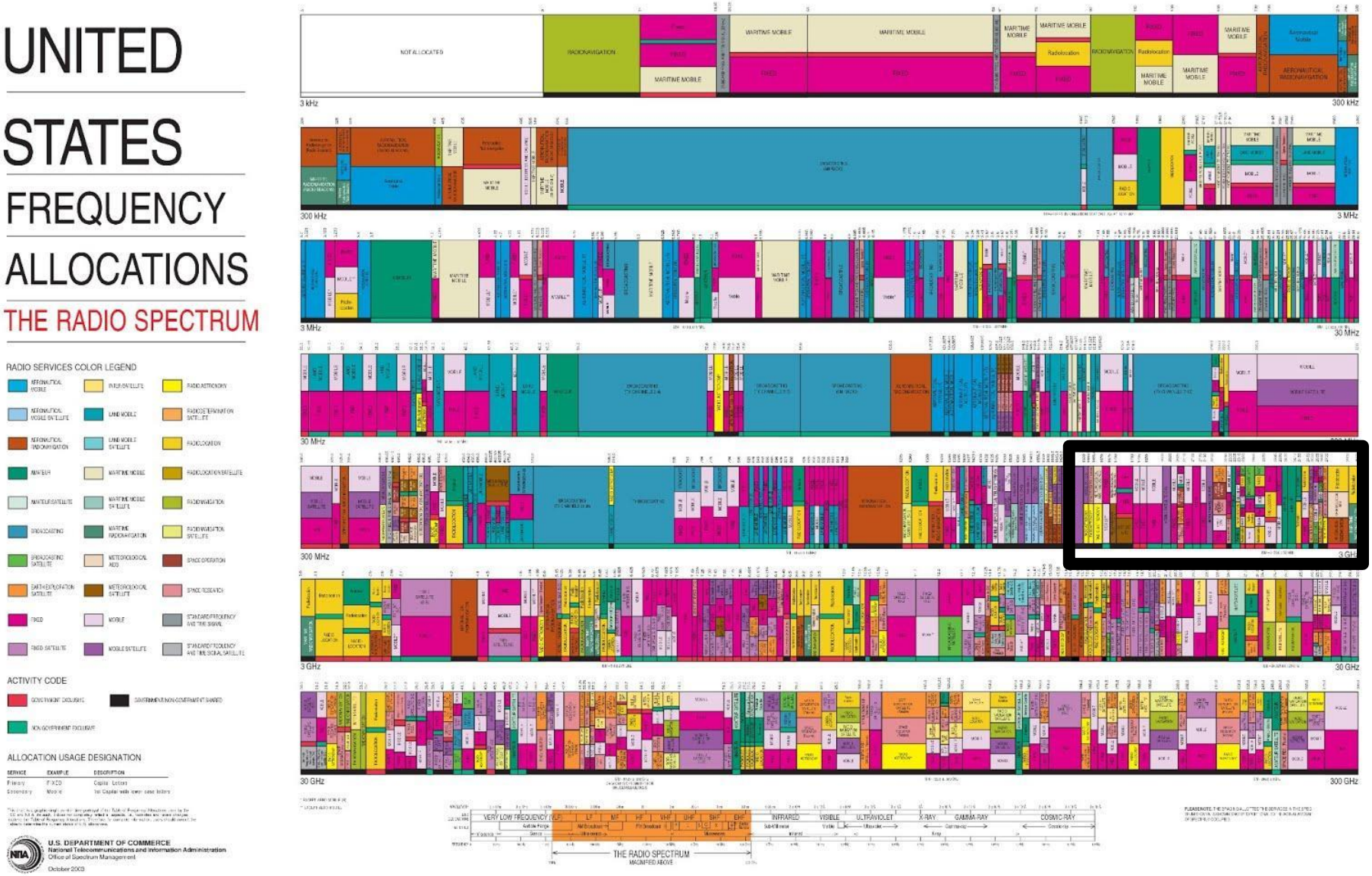


Image taken from: en.wikipedia.org/wiki/File:United_States_Frequency_Allocations_Chart_2003_-_The_Radio_Spectrum.jpg

United States Partial Frequency Spectrum

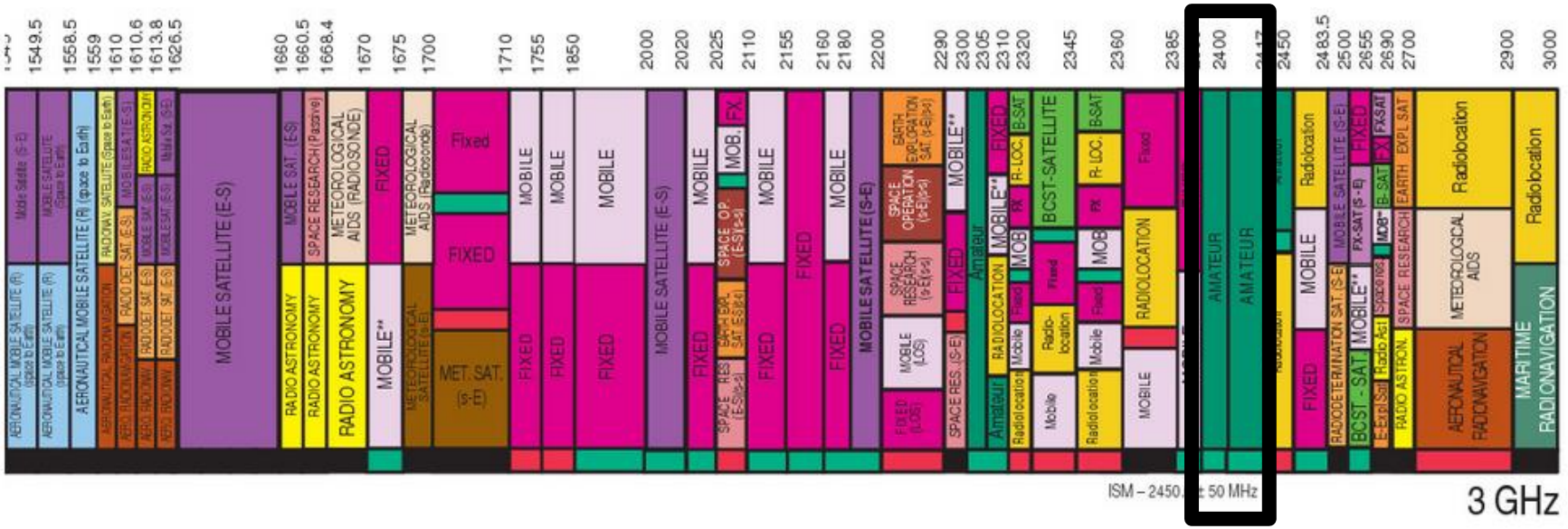


Image taken from: en.wikipedia.org/wiki/File:United_States_Frequency_Allocations_Chart_2003_-_The_Radio_Spectrum.jpg

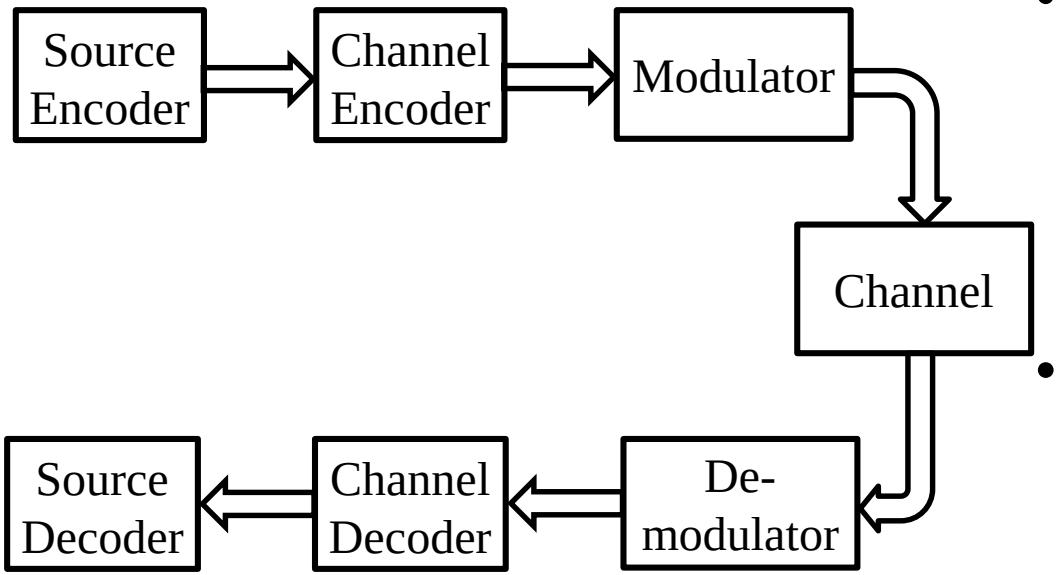
Message, Medium, and Power & noise

- **Message**
  - Source encoding, Channel encoding, Modulation, and Protocol and packets
- **Medium**
  - Shannon's limit, Nyquist sampling, Path loss, Multi-channel, loss models, Slow and fast fading.
- **Signal power & noise power**
  - Receive and send power, Antennas, Expected noise floors.
- **Putting it together**
  - Modulation (again), MIMO

So starting with the **message**

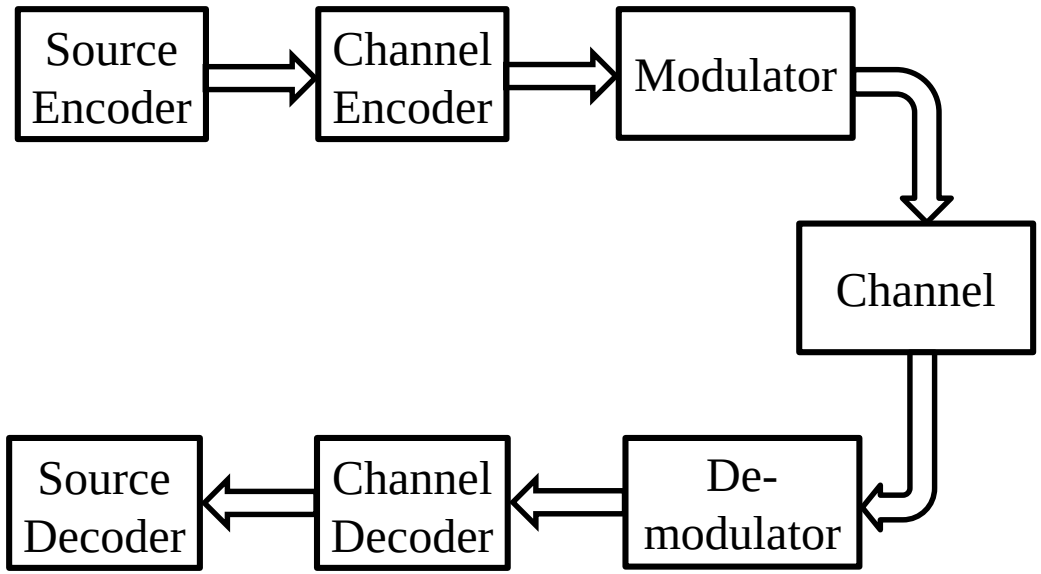
- We are trying to send data from one point to the next over some channel.
  - What should we do to get that message ready to go?
  - The basic steps are
    - Convert it to binary (if needed)
    - Compress as much as we can
      - to make the message as small as we can
    - Add error correction
      - To reduce errors
      - But, unexpectedly, also to speed up communication over the channel.
  - The receiver will need to undo all that work.

Communicating a Message (1/3)



- Source
  - The message we want to send.
  - We'll assume it's in binary already.
- Source encoding
  - Compression; remove redundancies.
  - Could be lossy (e.g. jpeg)
  - Called source encoding because depends on source type (think jpeg vs mp3)

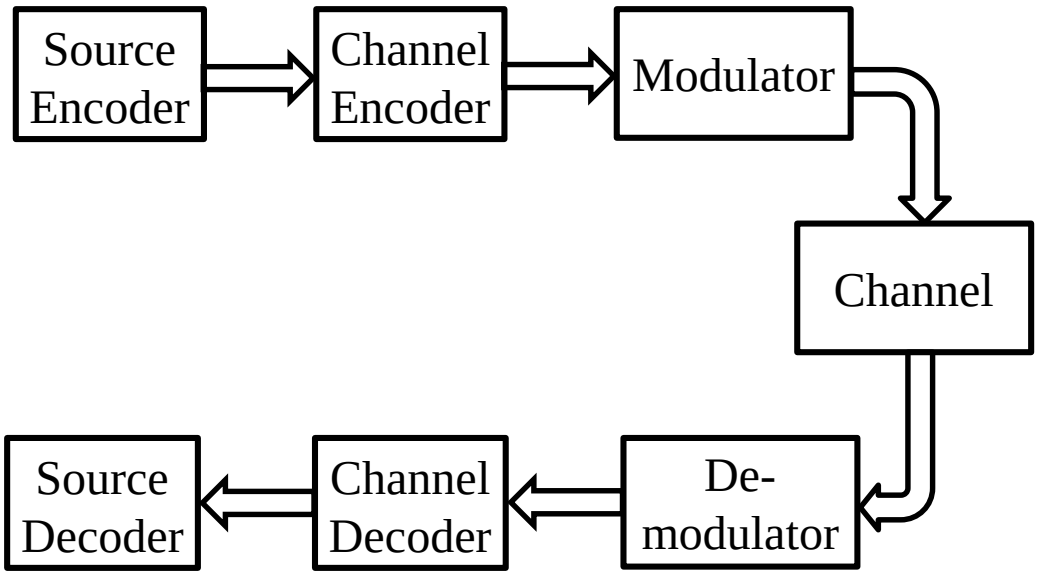
Communicating a Message (2/3)



- Channel encoder
  - Add error correction.
  - Called channel encoder, because error correction choices depend on channel.
- Modulator
  - Convert to analog.
    - Figure out how to move to carrier frequency.
    - Lots of options including:
      - Frequency modulation
      - Amplitude modulation
      - Phase modulation

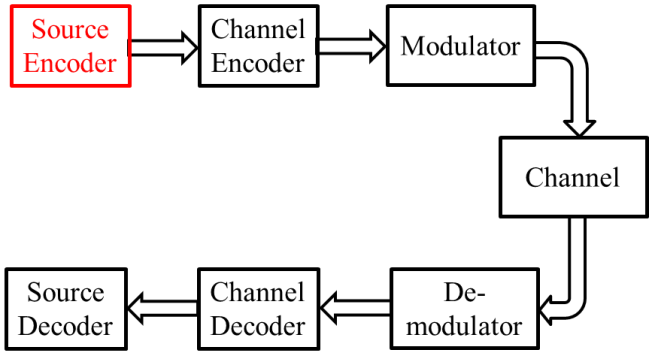
Note: some sources consider modulation to be part of the channel encoder.

Communicating a Message (3/3)



- Then the receiver undoes all that (demodulation and the two decoders)
  - Often more work than sending!

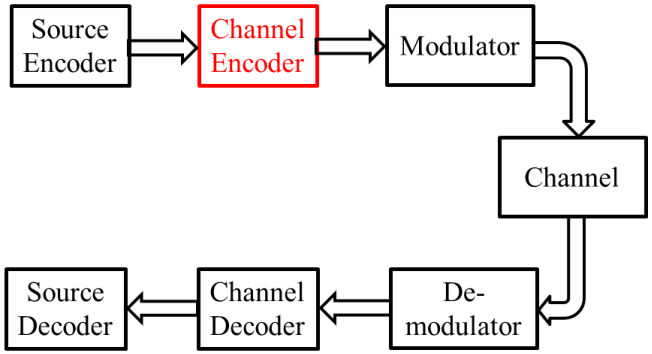
- Channel
  - The medium over which our encoded message is sent.
  - For the type of wireless communication we are doing, we are talking about using radio frequencies (RF) to connect two points not connected by a conductor.
  - **Lossy**.



Source encoding

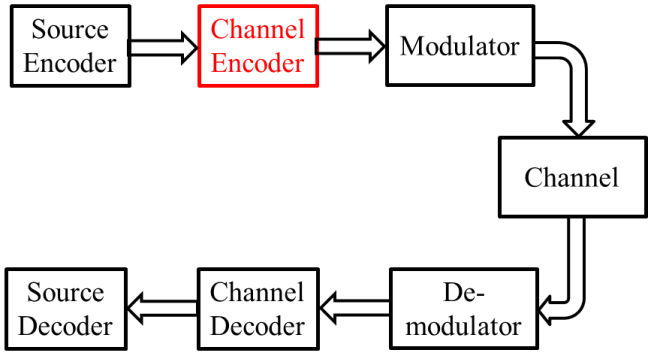
- Pretty much traditional CS techniques for compression
  - Very much dependent on nature of source
    - We use different techniques for different things.
    - Huffman encoding is the basic solution
- Goal here is to remove redundancy to make the message as small (in bits) as possible.
  - Can accept loss in some cases (images, streaming audio, etc.)

For more information: http://en.wikipedia.org/wiki/Data_compression, http://www.ccs.neu.edu/home/jnl22/oldsite/cshonor/jeff.html



Channel encoding (1/3)

- Error correction and detection
  - We are adding bits back into the message (after compression) to reduce errors that occur in the channel.
  - The number of bits added and how we add them depends on characteristics of the channel.
- Idea:
  - Extra bits add redundancy.
  - If a bit (or bits) go bad, we can either repair them or at least detect them.
  - If detect an error, we can ask for a resend.



Channel encoding (2/3)

- Block codes
  - In this case we are working with fixed block sizes.
  - We take a group of N bits, add X bits to the group.
  - Some schemes promise correction of up to Y bits of error (including added bits)
  - Others detect Z bits of error.

- Specific coding schemes
  - Add one bit to each block (parity)
    - Can ***detect*** any one bit error.
  - Take N bits, add ~$\log_2(N)$ bits (for large N)
    - Can ***correct*** any one bit error.
  - Both of the above can be done using Hamming codes.
    - Also Reed-Solomon codes and others.

See http://en.wikipedia.org/wiki/Block_code for more details.

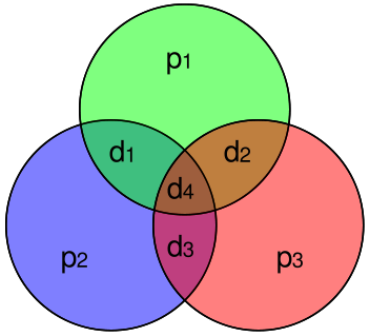
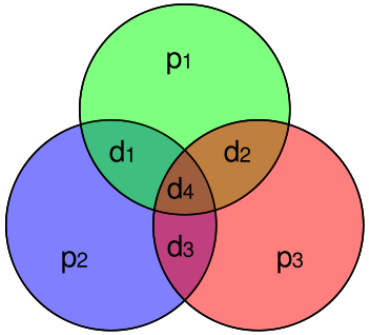


Figure from Wikipedia

Example block code: Hamming(7,4)

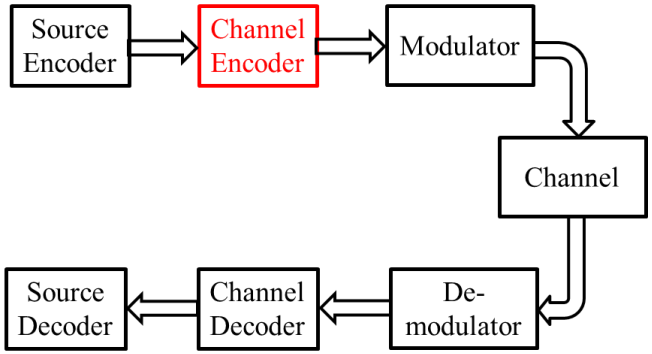
- Hamming(7,4)-code.
  - Take 4 data elements (d_1 to d_4)
  - Add 3 parity bits (p_1 to p_3)
    - $p_1=P(d_1, d_2, d_4)$
    - $p_2=P(d_1, d_3, d_4)$
    - $p_3=P(d_2, d_3, d_4)$
  - If any one bit goes bad (p or d) can figure out which one.
    - Just check which parity bits are wrong. That will tell you which bit went wrong.
    - If more than one went wrong, scheme fails.
- Much more efficient on larger blocks.
  - E.g. (136,128) code exists.
- Example:
  - Say
    - d[1:4]=4'b0011
  - Then:
    - $p_1=P(0,0,1)=1$
    - $p_2=P(0,1,1)=0$
    - $p_3=P(0,1,1)=0$
  - If d_2 goes bad (is 1)
    - Then received p_1 and p_3 are wrong.
    - Only d_3 covered by both (and only both)
      - ***So d_3 is the one that flipped.***



In-Class Example

- Hamming(7,4)-code.
  - Take 4 data elements (d_1 to d_4)
  - Add 3 parity bits (p_1 to p_3)
    - $p_1=P(d_1, d_2, d_4)$
    - $p_2=P(d_1, d_3, d_4)$
    - $p_3=P(d_2, d_3, d_4)$

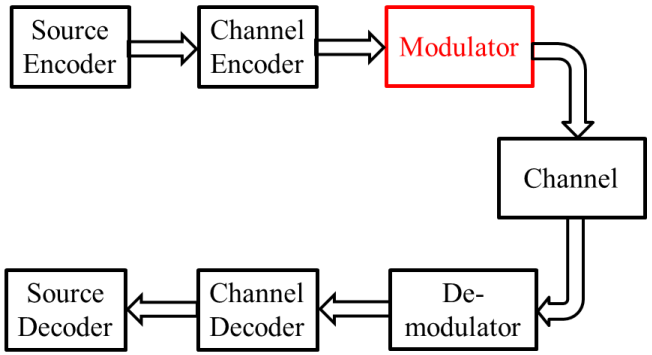
- If we get: 1000101
  - What was the data?
- If we get: 1111111
  - What was the data?
- If we get: 0100100
  - What was the data?



Channel encoding (3/3)

- Convolution codes
  - Work on a sliding window rather than a fixed block.
  - Often send one or even two parity bits per data bit.
- Can be good for finding close solutions even if wrong.
  - Viterbi codes are a very common type of
- Turbo codes are a type of convolution code that can provide near-ideal error correction
  - That's different than perfect, just nearly as good as possible.
  - Approaches Shannon's limit, which we'll cover shortly.
- Low-density parity-check (LDPC) codes are block codes with similar properties.

See http://en.wikipedia.org/wiki/Convolutional_code for more details.



Modulation

- We take an input signal and move it to a carrier frequency (f_c) in a number of way.
  - We can vary the amplitude of the signal
  - We can vary the frequency of the signal.
  - We can vary the phase of the signal.

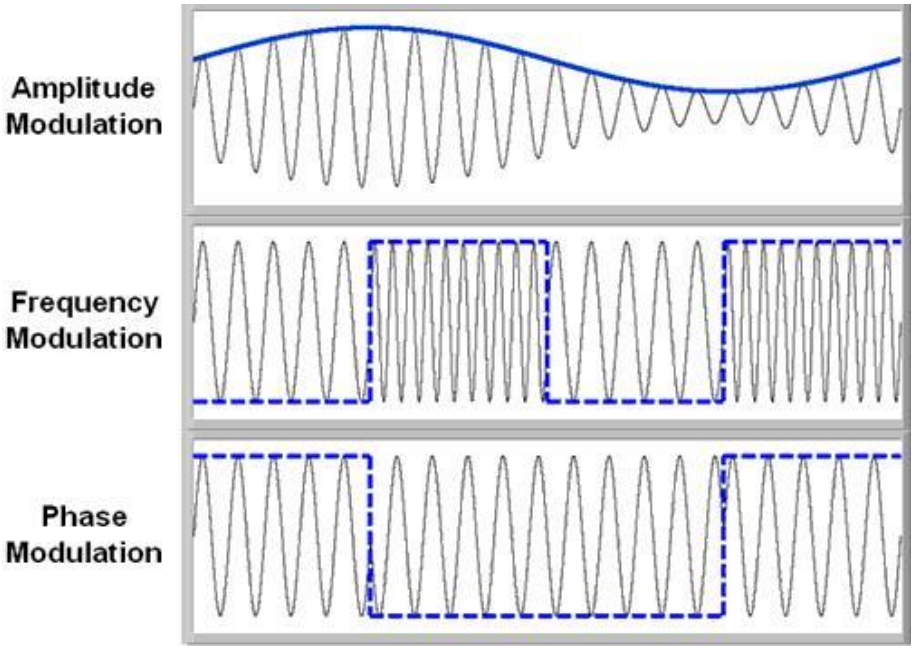
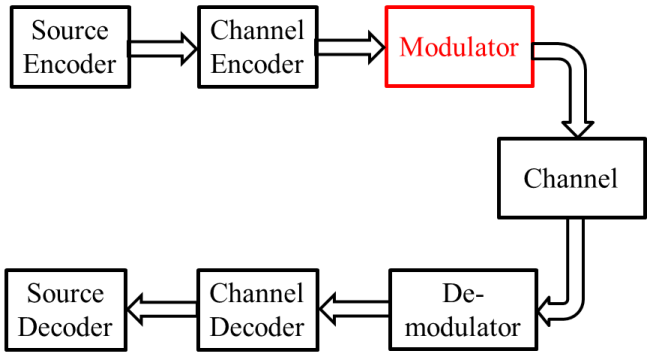
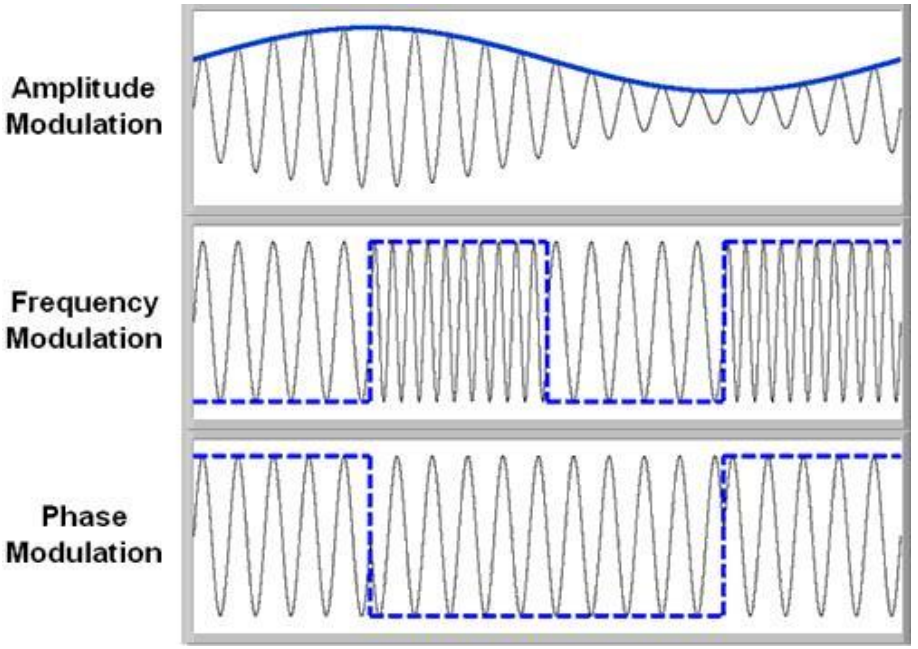


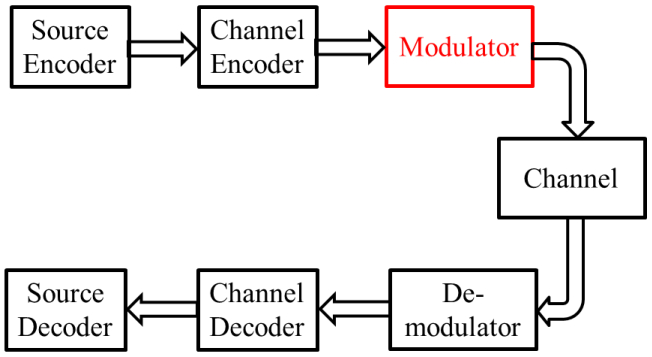
Figure from http://www.ni.com/white-paper/4805/en/



Terms: "keying"

- Keying is a family of modulations where we allow only a predetermined set of values.
  - Here, frequency and phase only have two values, so those two examples are "keying"
    - Note phase and frequency could be continuous rather than discrete.





Example: Amplitude-Shift Keying (ASK)

- Changes amplitude of the transmitted signal based on the data being sent
- Assigns specific amplitudes for 1's and 0's
- On-off Keying (OOK) is a simple form of ASK

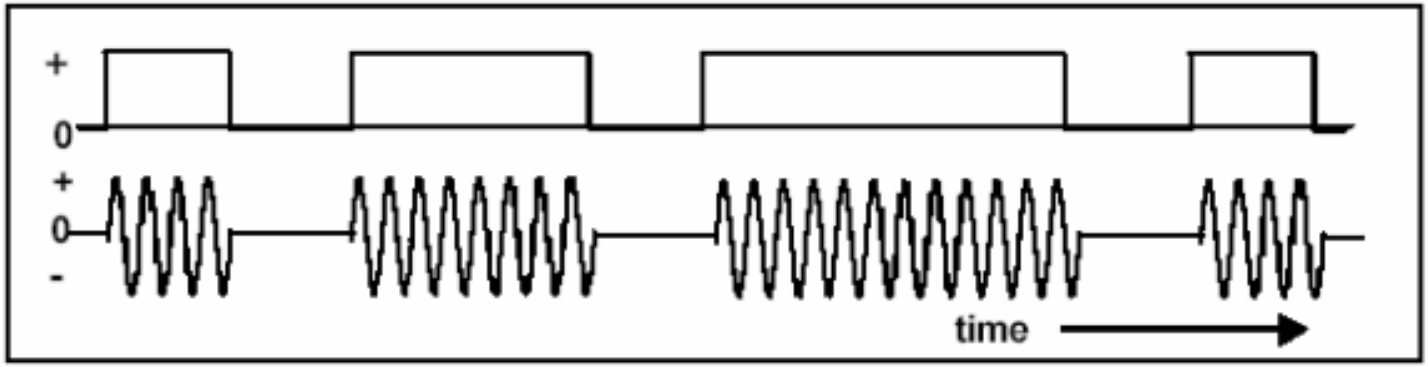
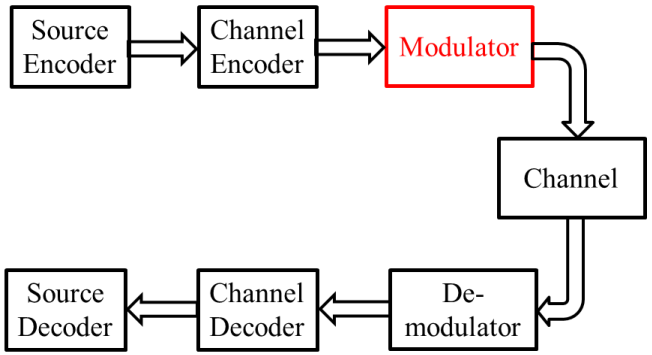


Figure 1: an ASK signal (below) and the message (above)

Figure from http://www.ele.uri.edu/Courses/ele436/labs/ASKnFSK.pdf



Example: Frequency Shift Keying (FSK)

- Changes frequency of the transmitted signal based on the data being sent
- Assigns specific frequencies for 1's and 0's

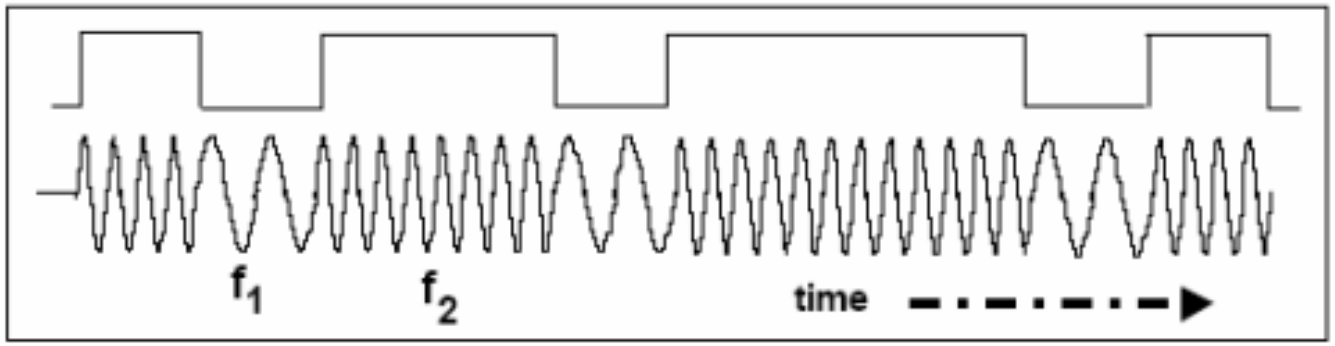
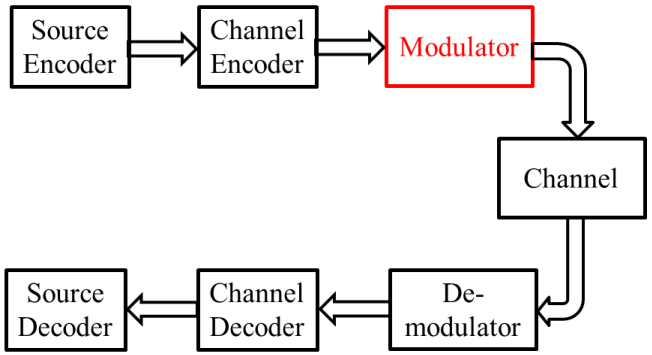


Figure 1: an FSK waveform, derived from a binary message

Figure from http://www.ele.uri.edu/Courses/ele436/labs/ASKnFSK.pdf



Example: Phase Shift Keying (PSK)

- Changes phase of the transmitted signal based on the data being sent
- Send a 0 with 0° phase, 1 with 180° phase
- This case called Binary Phase Shift Keying (BPSK)

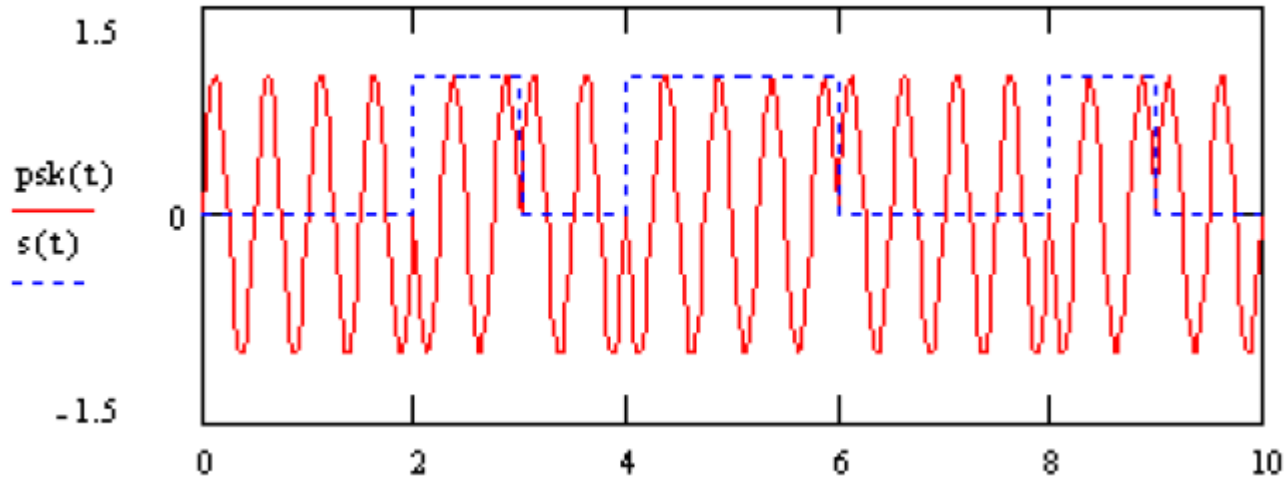


Figure from http://people.seas.harvard.edu/~jones/cscie129/papers/modulation_1.pdf

And we can have modulation of a continuous signal

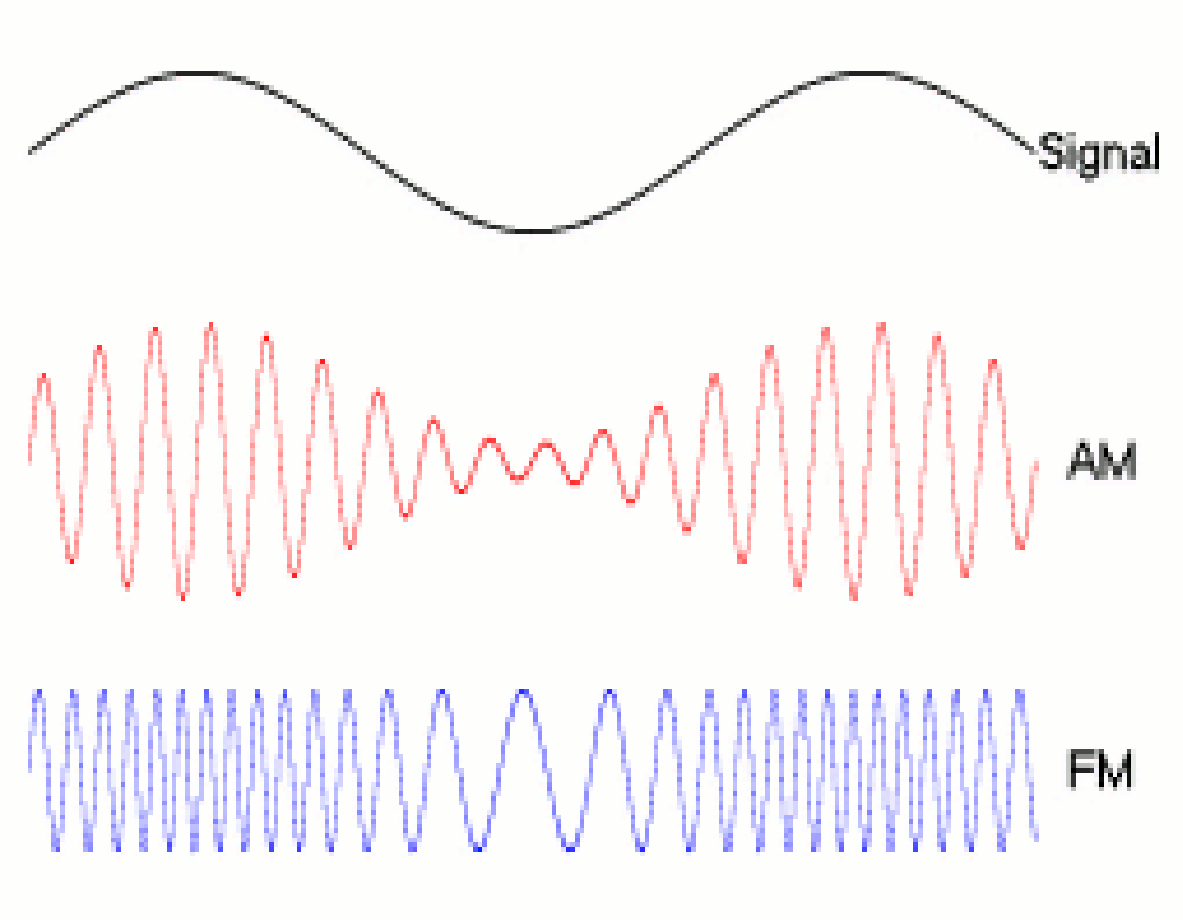


Figure from http://en.wikipedia.org/wiki/Modulation

Back to Keying—M-ary

- It's possible to do more than binary keying.
  - Could use "M-ary" symbols
    - Basically have an alphabet of M symbols.
  - For ASK this would involve 4 levels of amplitude.
    - Though generally it uses 2 amplitudes, but has "negative valued" amplitudes.

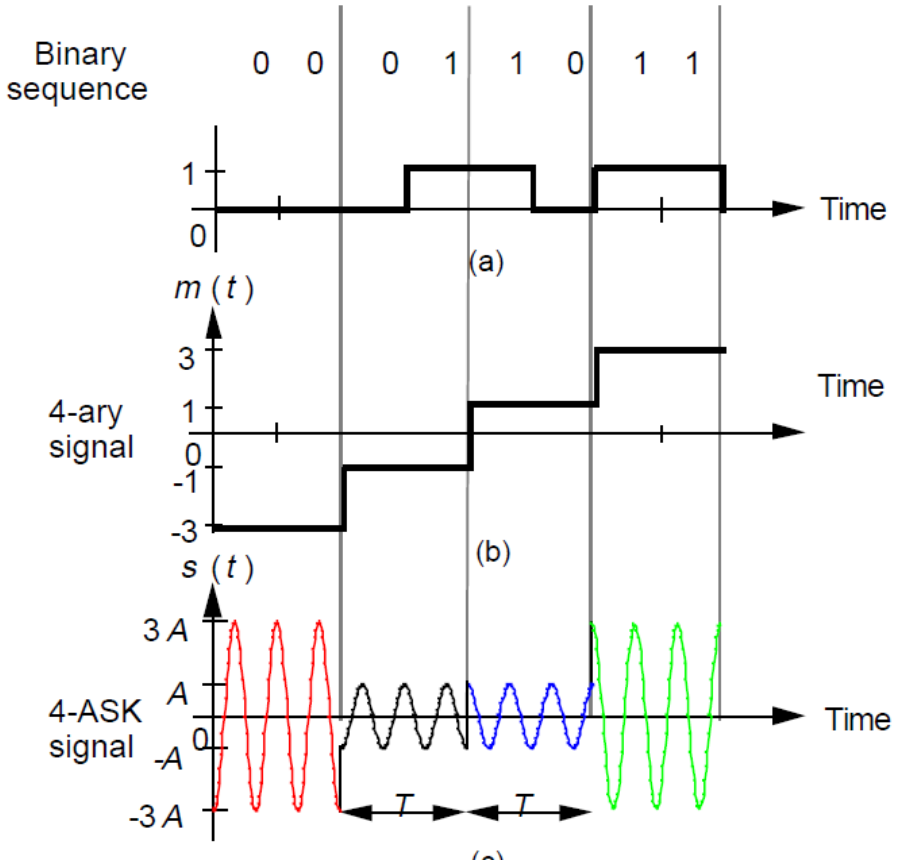
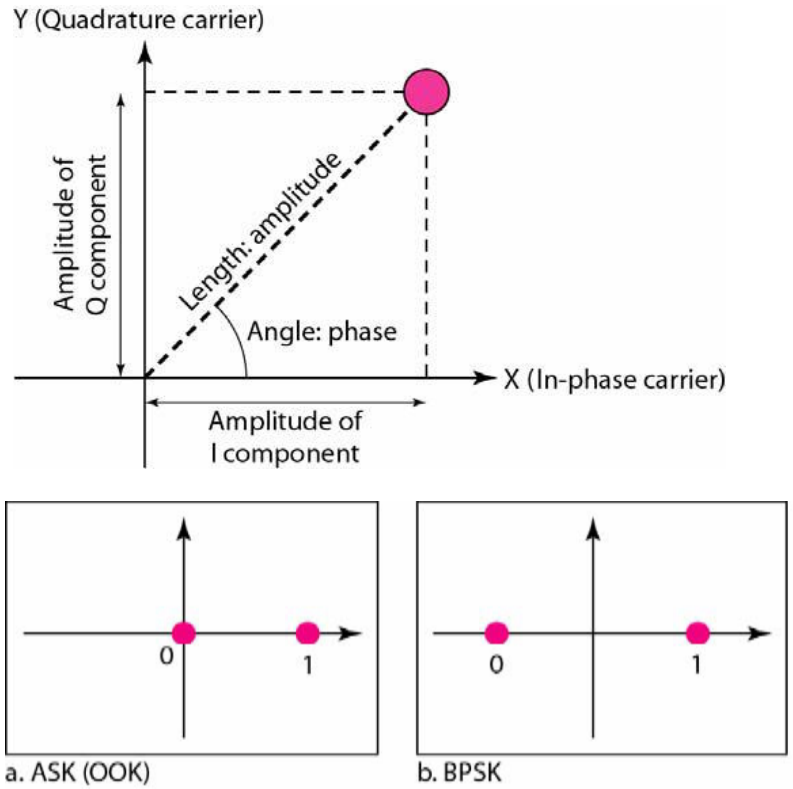
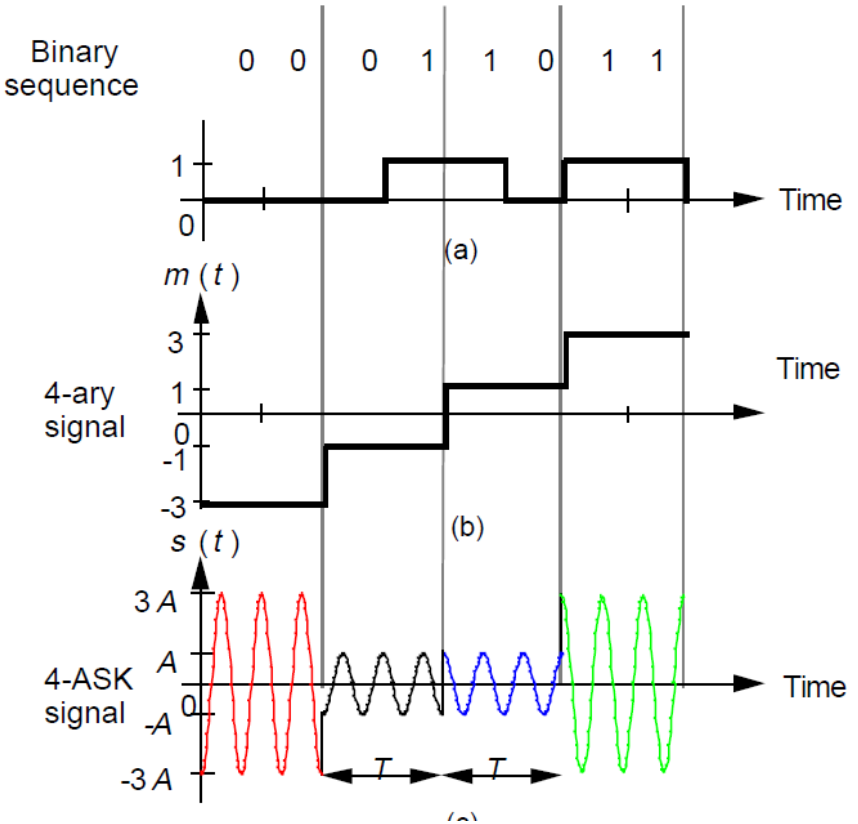


Figure from http://engineering.mq.edu.au/~cl/files_pdf/elec321/lect_mask.pdf

Key “constellations”

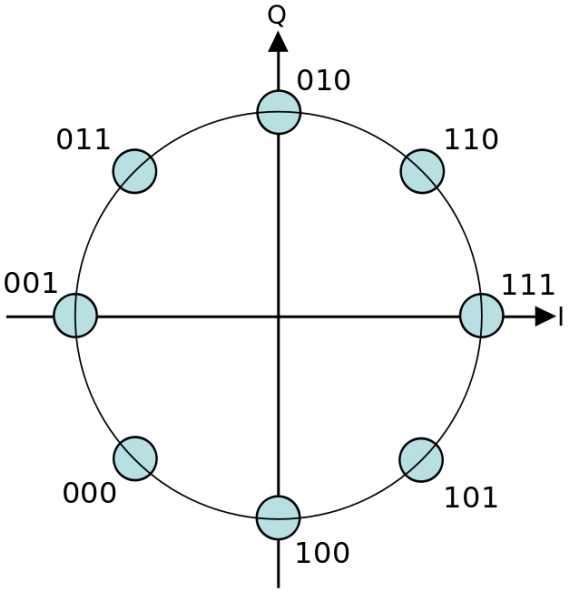




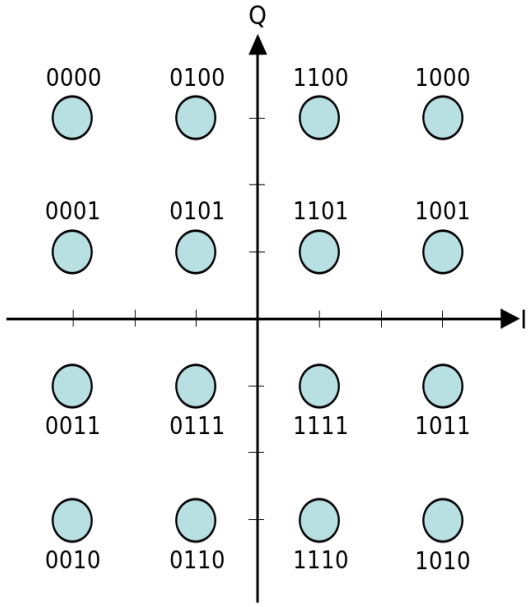
Draw the 4-ASK constellation.

New figures from http://www.eecs.yorku.ca/course_archive/2010-11/F/3213/CSE3213_07_ShiftKeying_F2010.pdf

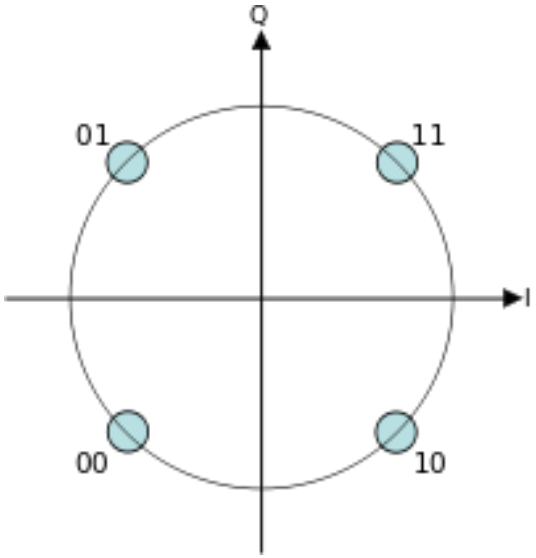
Some constellations



8-PSK



16-QAM
(Quadrature amplitude)

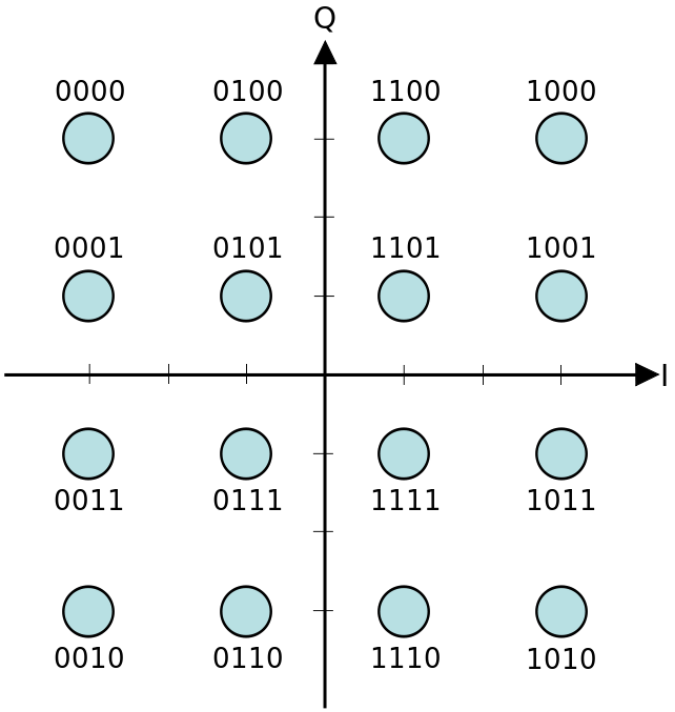


4-PSK
QPSK
4-QAM
(lots of names)

QPSK=quadriphase PSK. Really.

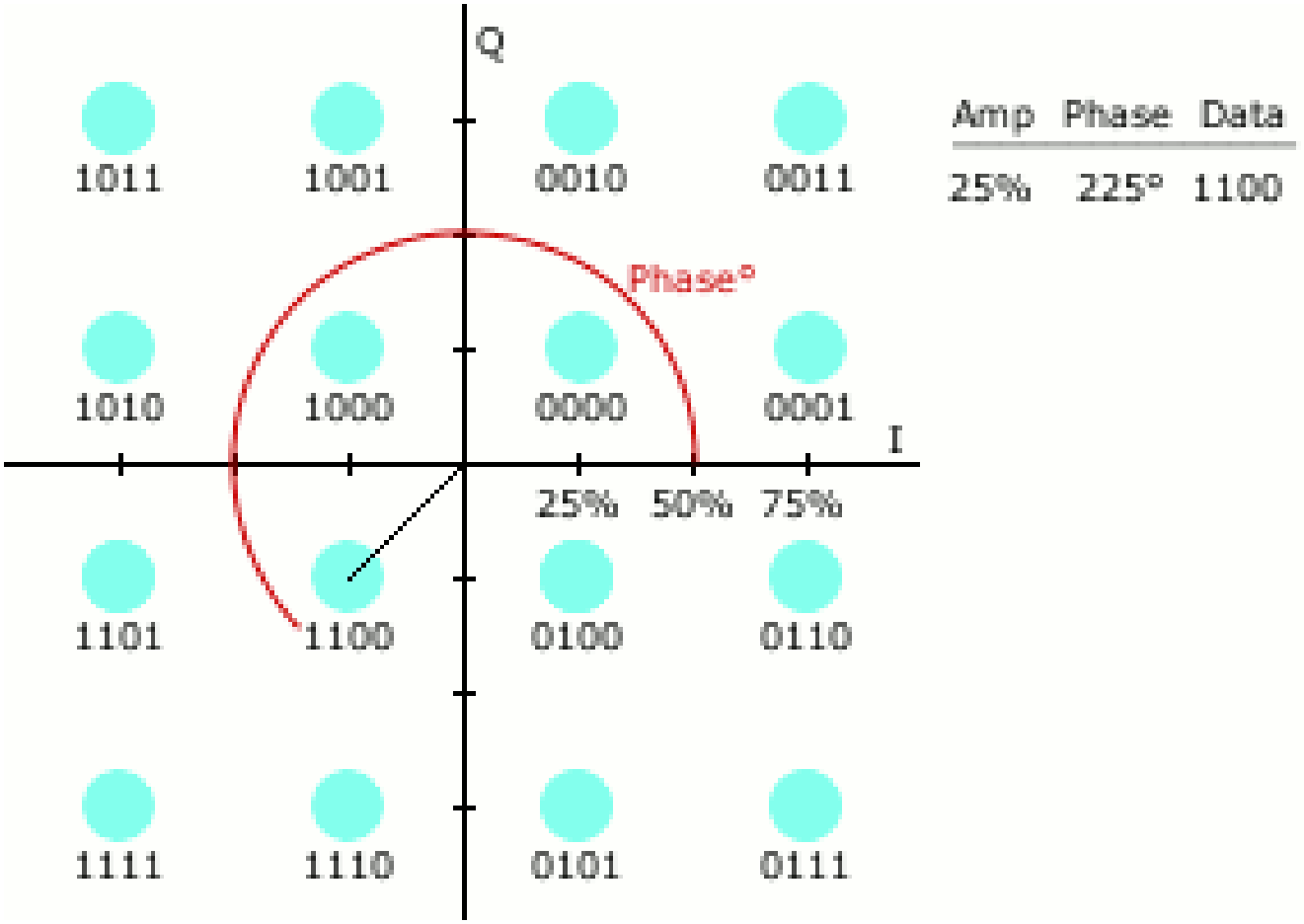
Figures from Wikipedia

QAM



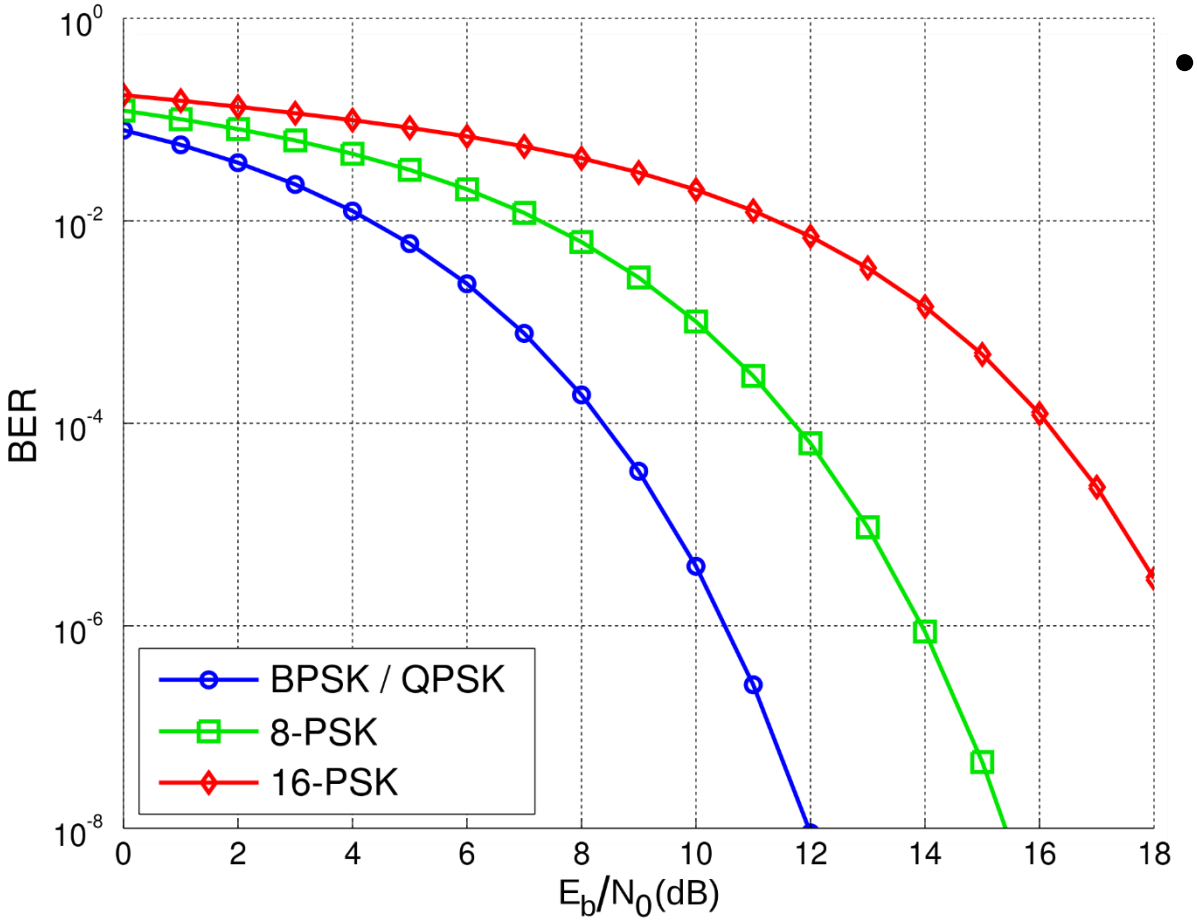
16-QAM
(Quadrature amplitude)

- Can be thought of as varying phase and amplitude for each symbol.
  - Can also be thought of as mixing two signals 90 degrees out of phase.
    - I and Q.



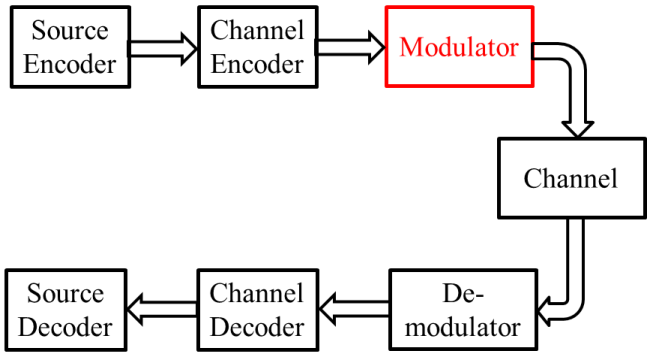
Animation from Wikipedia

So, who cares?
Noise immunity



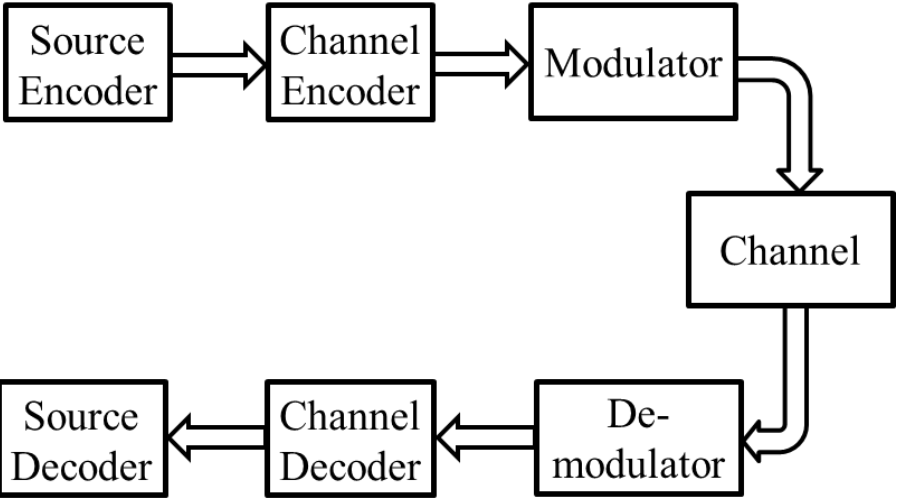
https://en.wikipedia.org/wiki/Eb/N0

- Looking at signal-to-noise ratio needed to maintain a low bit error rate.
  - Notice BPSK and QPSK are least noise-sensitive.
  - And as "M" goes up, we get more noise sensitive.
    - Easier to confuse symbols!



Modulation

- So we have a lot of modulation choices.
  - Could view it all as FSK and everything else.



Wireless messages

- Sending a message
  - We first compress the source (source encoding)
  - Then add error correction (channel encoding)
  - Then modulate the signal
- Each of these steps is fairly complex
  - We spent more time on modulation, because our prereq. classes don't cover it.

Shannon's limit

- First question about the medium:
  - How fast can we hope to send data?
- Answered by Claude Shannon (given some reasonable assumptions)
  - Assuming we have only Gaussian noise, provides a bound on the rate of *information* that can be reliably moved over a channel.
    - That includes error correction and whatever other games you care to play.

Taken from a slide by Dr. Stark

Fundamental Tradeoffs

More than 50 years ago Claude Shannon (U of M EE/Math graduate) determined the tradeoff between data rate, bandwidth, signal power and noise power for reliable communications for an additive white Gaussian noise channel. Let W be the bandwidth (in Hz), R be the data rate (in bits per second), P be the *received* signal power (in watts) and $N_0/2$ the noise power spectral density (in watts/Hz) then reliable communication is possible provided

$$R < W \log_2(1 + \frac{P}{N_0 W}).$$

Shannon–Hartley theorem

- We'll use a different version of this called the Shannon-Hartley theorem.

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

- **C** is the channel capacity in bits per second;
- **B** is the bandwidth of the channel in hertz
- **S** is the total received signal power measured in Watts or Volts^2
- **N** is the total noise, measured in Watts or Volts^2

Adapted from Wikipedia.

Comments (1/2)

- This is a *limit*. It says that you can, in *theory*, communicate that much ***data*** with an arbitrarily tight bound on error.
  - Not that you won't get errors at that data rate. Rather that it's possible you can find an error correction scheme that can fix things up.
    - Such schemes may *require* really <u>really</u> long block sizes and so may be computationally intractable.
- There are a number of proofs.
  - IEEE reprinted the original paper in 1998
    - http://www.stanford.edu/class/ee104/shannonpaper.pdf
  - More than we are going to do.
    - Let's just be sure we can A) understand it and B) use it.

Comments (2/2)

- What are the assumptions made in the proof?
  - All noise is Gaussian in distribution.
    - This not only makes the math easier, it means that because the addition of Gaussians is a Gaussian, all noise sources can be modeled as a single source.
    - Also note, this includes our inability to distinguish different voltages.
      - Effectively quantization noise and also treated as a Gaussian (though it ain't)
- Can people actually do this?
  - They can get really close.
    - Turbo codes,
    - Low density parity check codes.

Examples (1/2)

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

- **C** is the channel capacity in bits per second;
- **B** is the bandwidth of the channel in Hertz
- **S** is the total received signal power measured in Watts or Volts^2
- **N** is the total noise, measured in Watts or Volts^2

- If the SNR is 20 dB, and the bandwidth available is 4 kHz what is the channel capacity?
  - Part 1: convert dB to a ratio (it's power so it's base 10)
  - Part 2: Plug and chug.

$$\mathbf{db = 10 \log P_1/P_0}$$

$$\mathbf{P_1 = 10^{db/10} * P_0}$$

Adapted from Wikipedia.

Examples (2/2)

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

- **C** is the channel capacity in bits per second;
- **B** is the bandwidth of the channel in Hertz
- **S** is the total received signal power measured in Watts or Volts^2
- **N** is the total noise, measured in Watts or Volts^2

- If you wish to transmit at 50,000 bits/s, and a bandwidth of 1 MHz is available, what S/R ration can you accept?

Adapted from Wikipedia.

Summary of Shannon's limit

- Provides an upper-bound on *information* over a channel
  - Makes assumptions about the nature of the noise.
- To approach this bound, need to use channel encoding and modulation.
  - Some schemes (Turbo codes, Low density parity check codes) can get very close.

Acknowledgments and sources

- A 9 hour talk by David Tse has been extremely useful and is a basis for me actually understanding anything (though I'm by no means through it all)
- A talk given by Mike Denko, Alex Motalleb, and Tony Qian two years ago for this class proved useful and I took a number of slides from their talk.
- An hour long talk with Prabal Dutta formed the basis for the coverage of this talk.
- Some other sources:
  - http://www.cs.cmu.edu/~prs/wirelessS12/Midterm12-solutions.pdf -- A nice set of questions that get at some useful calculations.
  - http://people.seas.harvard.edu/~jones/es151/prop_models/propagation.html all the path loss/propagation models in one place
  - http://people.seas.harvard.edu/~jones/cscie129/papers/modulation_1.pdf very nice modulation overview.

- I'm grateful for the above sources. All mistakes are my own.

Additional sources/references

General

- http://www.cs.cmu.edu/~prs/wirelessS12/Midterm12-solutions.pdf

Modulation

- https://fetweb.ju.edu.jo/staff/ee/mhawa/421/Digital%20Modulation.pdf
- http://www.ece.umd.edu/class/enee623.S2006/ch2-5_feb06.pdf
- https://www.nhk.or.jp/strl/publica/bt/en/le0014.pdf
- http://engineering.mq.edu.au/~cl/files_pdf/elec321/lect_mask.pdf (ASK)
- http://www.eecs.yorku.ca/course_archive/2010-11/F/3213/CSE3213_07_ShiftKeying_F2010.pdf

Message, Medium, and Power & noise

- **Message**
  - Source encoding, Channel encoding, Modulation, and Protocol and packets
- **Medium**
  - Shannon's limit, Nyquist sampling, Path loss, Multi-channel, loss models, Slow and fast fading.
- **Signal power & noise power**
  - Receive and send power, Antennas, Expected noise floors.
- **Putting it together**
  - Modulation (again), MIMO