Computer Networks

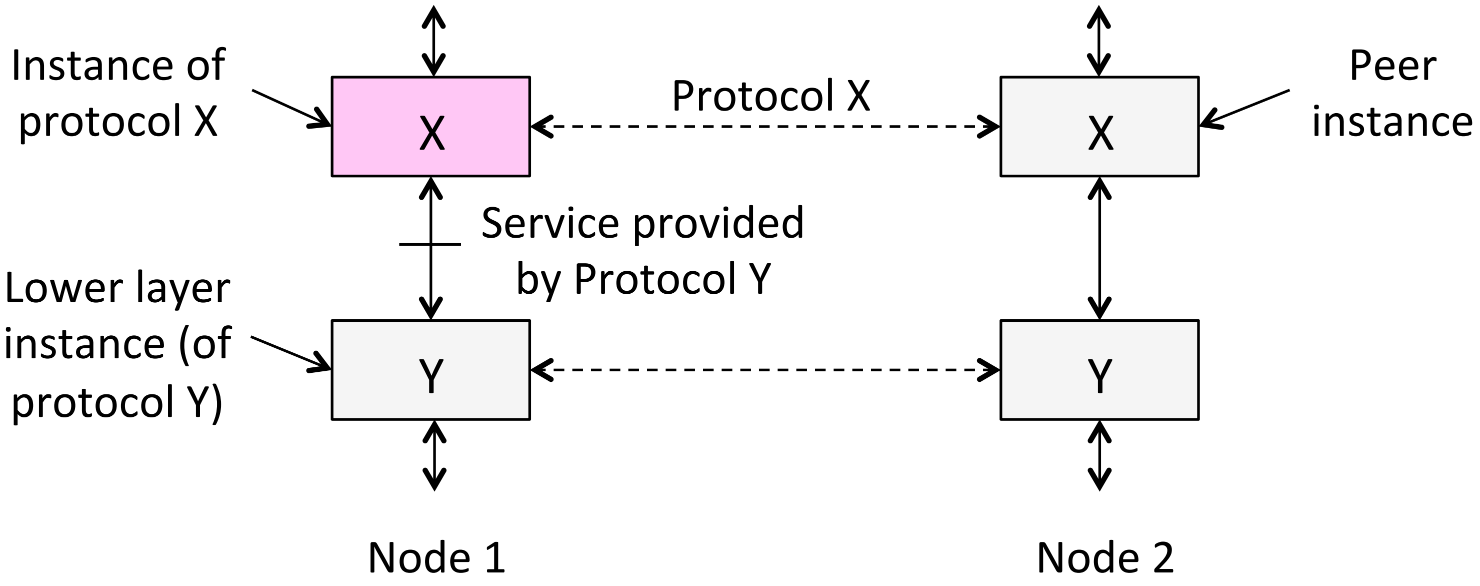
Shyam Gollakota

Protocols and Layers

- Protocols and layering is the main structuring method used to divide up network functionality
  - Each instance of a protocol talks virtually to its peer using the protocol
  - Each instance of a protocol uses only the services of the lower layer

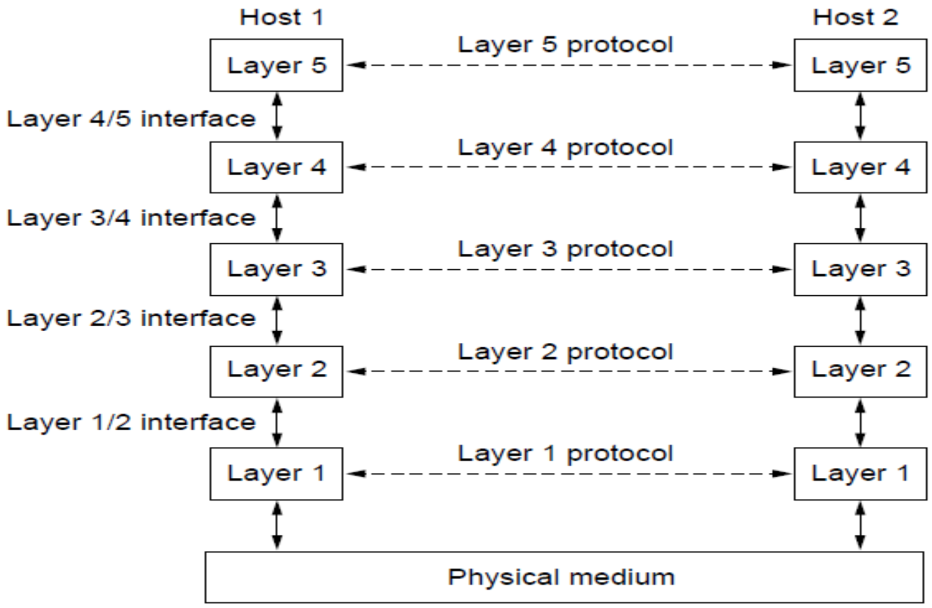
Protocols and Layers (3)

- Protocols are horizontal, layers are vertical



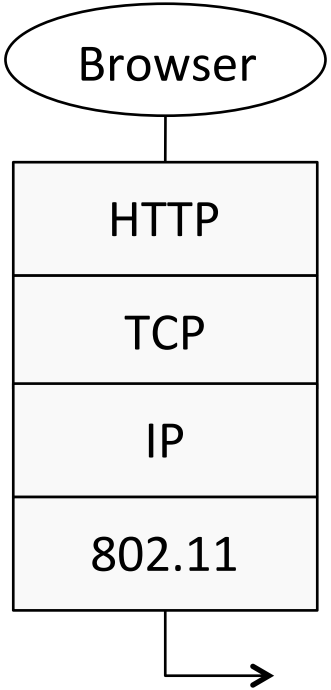
Protocols and Layers (4)

- Set of protocols in use is called a <u>protocol stack</u>



Protocols and Layers (6)

- Protocols you've probably heard of:
  - TCP, IP, 802.11, Ethernet, HTTP, SSL, DNS, … and many more
- An example protocol stack
  - Used by a web browser on a host that is wirelessly connected to the Internet

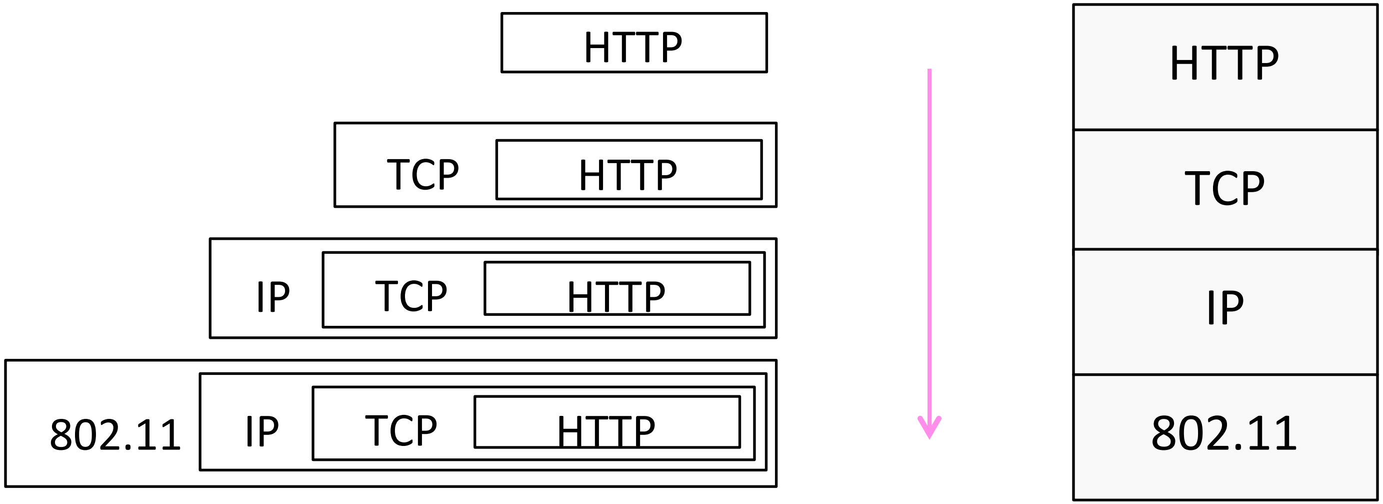


Encapsulation

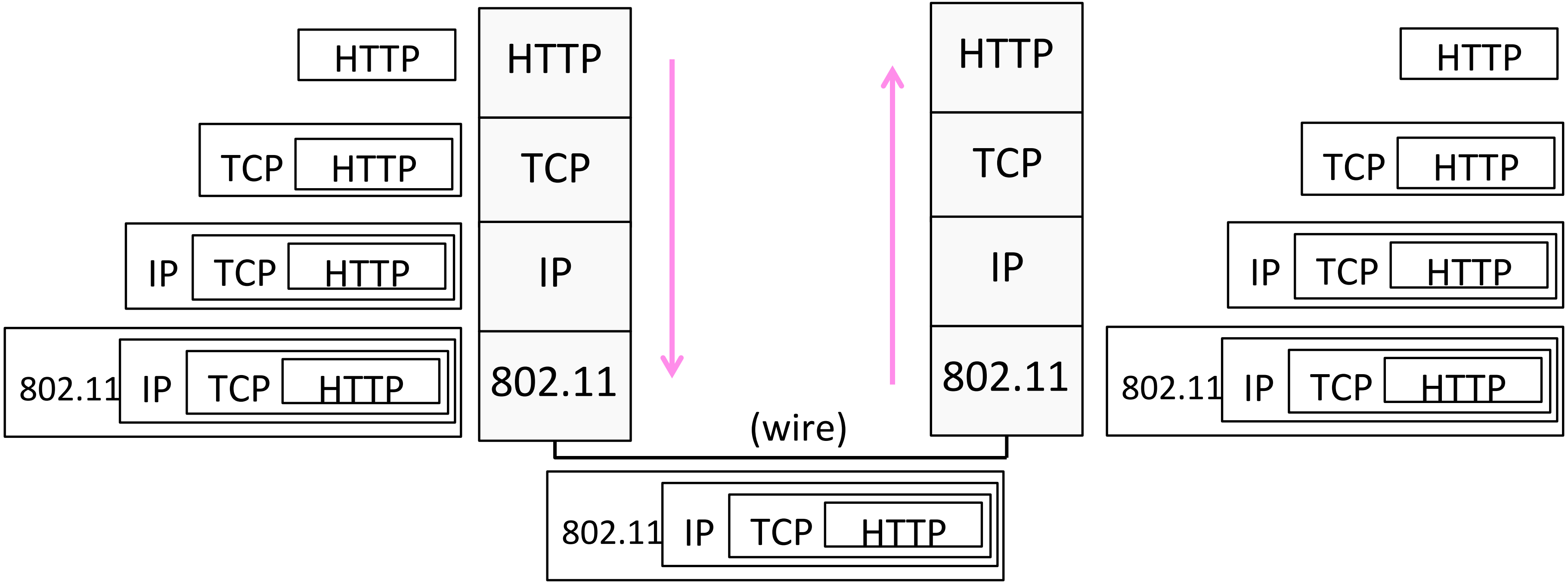
- Encapsulation is the mechanism used to effect protocol layering
  - Lower layer wraps higher layer content, adding its own information to make a new message for delivery
  - Like sending a letter in an envelope; postal service doesn’t look inside

Encapsulation (3)

- Message “on the wire” begins to look like an onion
  - Lower layers are outermost

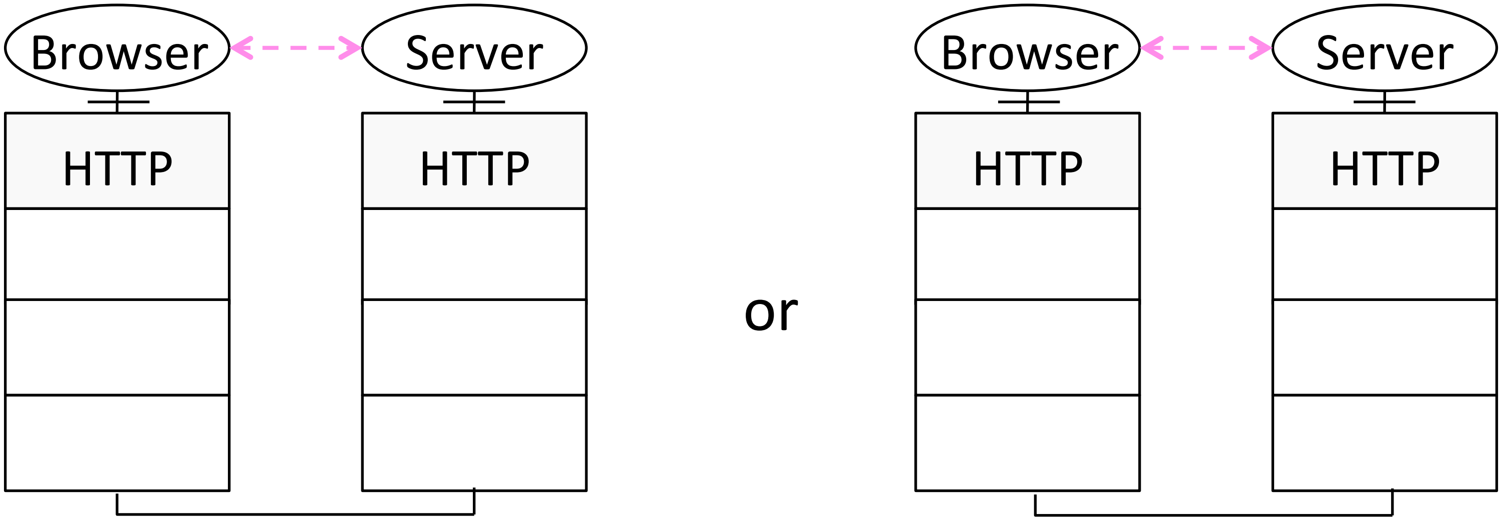


Encapsulation (4)



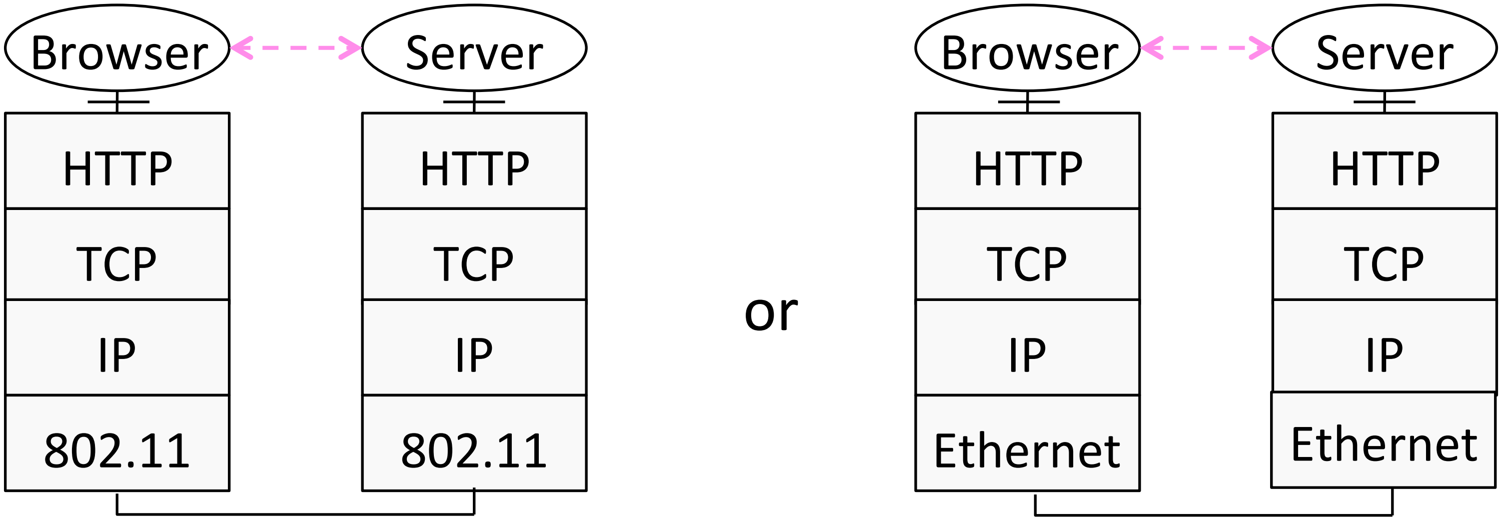
Advantage of Layering

- Information hiding and reuse



Advantage of Layering (2)

- Information hiding and reuse

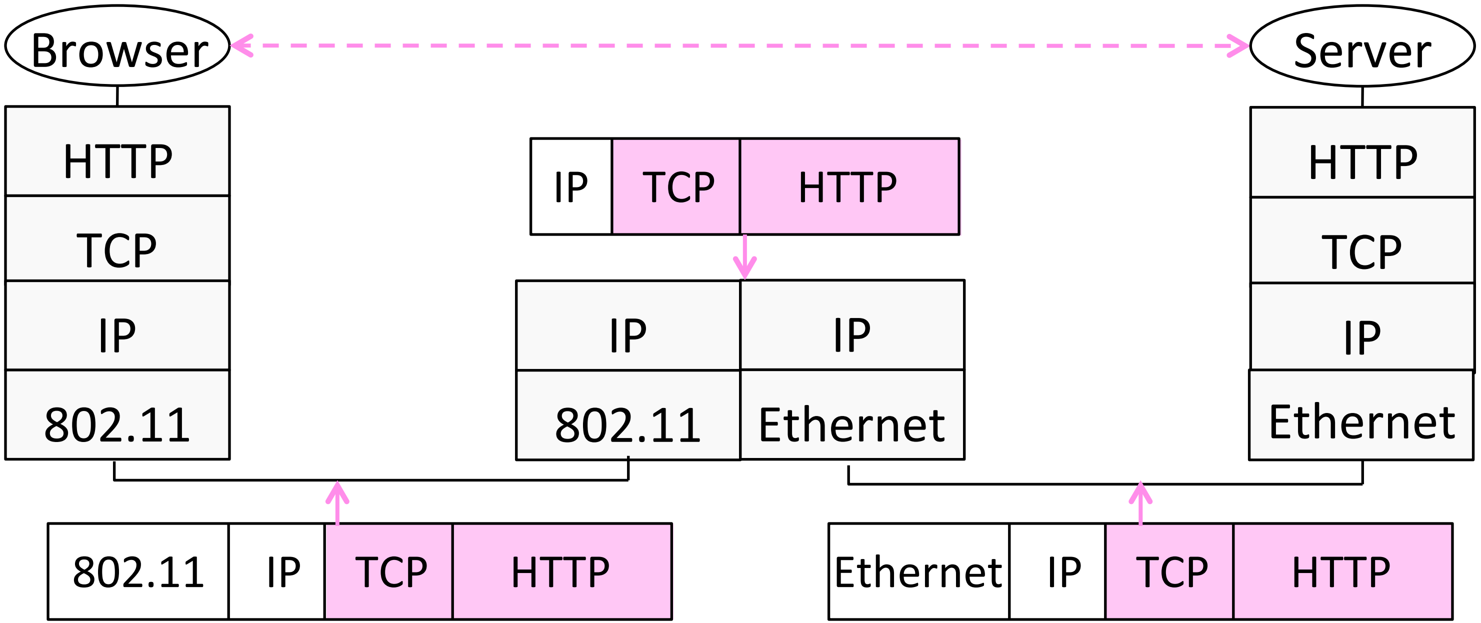


Advantage of Layering (3)

- Using information hiding to connect different systems

Advantage of Layering (4)

- Using information hiding to connect different systems



Disadvantage of Layering

- ??

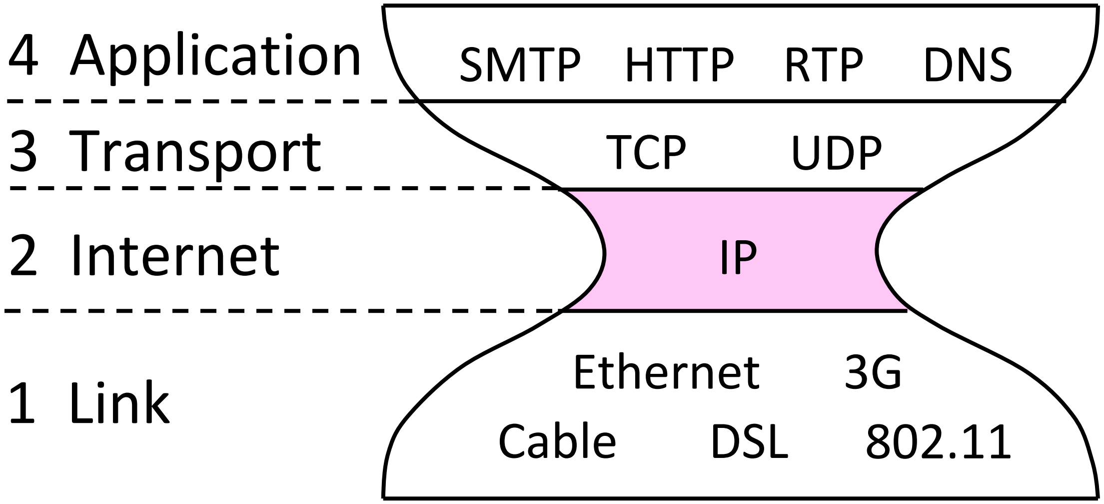
Internet Reference Model

- A four layer model based on experience; omits some OSI layers and uses IP as the network layer.

4	Application	– Programs that use network service
3	Transport	– Provides end-to-end data delivery
2	Internet	– Send packets over multiple networks
1	Link	– Send frames over a link

Internet Reference Model (3)

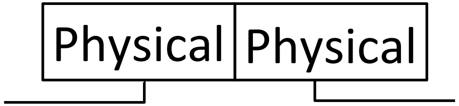
- IP is the "narrow waist" of the Internet
  - Supports many different links below and apps above



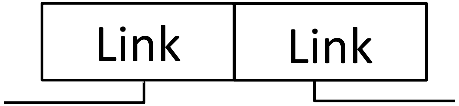
Layer-based Names (2)

- For devices in the network:

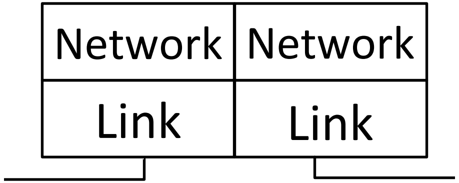
Repeater (or hub)



Switch (or bridge)



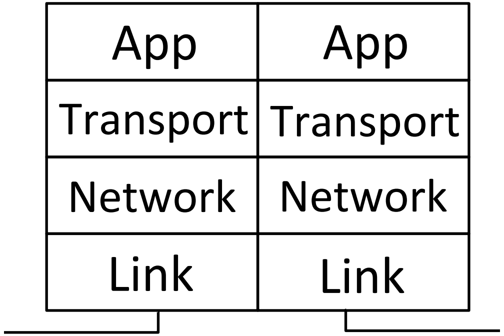
Router



Layer-based Names (3)

- For devices in the network:

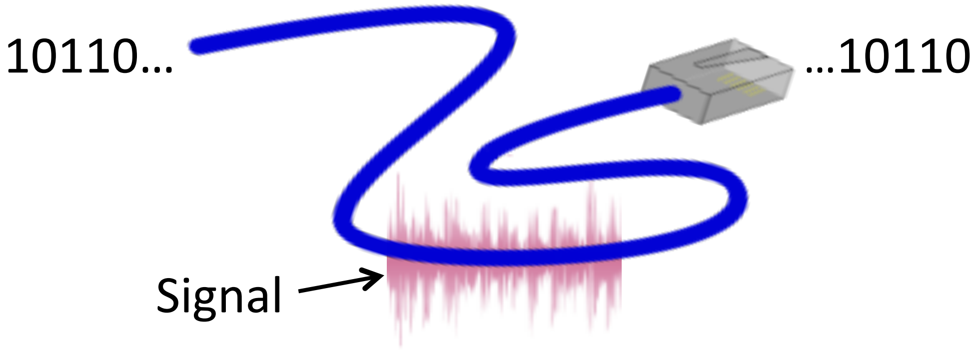
Proxy or middlebox or gateway



But they all look like this!

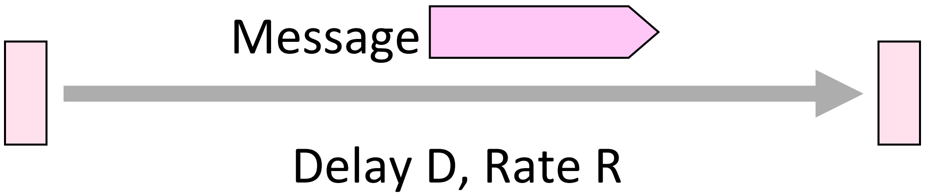
Scope of the Physical Layer

- Concerns how signals are used to transfer message bits over a link
  - Wires etc. carry <u>analog signals</u>
  - We want to send <u>digital bits</u>



Simple Link Model

- We'll end with an abstraction of a physical channel
  - Rate (or bandwidth, capacity, speed) in bits/second
  - Delay in seconds, related to length



- Other important properties:
  - Whether the channel is broadcast, and its error rate

Message Latency

- Latency is the delay to send a message over a link
  - Transmission delay: time to put M-bit message "on the wire"
  - Propagation delay: time for bits to propagate across the wire
  - Combining the two terms we have:

Message Latency (2)

- Latency is the delay to send a message over a link
  - Transmission delay: time to put M-bit message "on the wire"

    T-delay = M (bits) / Rate (bits/sec) = M/R seconds

  - Propagation delay: time for bits to propagate across the wire

    P-delay = Length / speed of signals = Length / ⅔c = D seconds

  - Combining the two terms we have: $\boxed{L = M/R + D}$

Metric Units

- The main prefixes we use:

Prefix	Exp.	prefix	exp.
K(ilo)	10^3	m(illi)	10^{-3}
M(ega)	10^6	μ(micro)	10^{-6}
G(iga)	10^9	n(ano)	10^{-9}

- Use powers of 10 for rates, 2 for storage
  - 1 Mbps = 1,000,000 bps, 1 KB = 2^{10} bytes
- "B" is for bytes, "b" is for bits

Latency Examples (2)

- "Dialup" with a telephone modem:

  D = 5 ms, R = 56 kbps, M = 1250 bytes

  $L = 5 \text{ ms} + (1250 \times 8)/(56 \times 10^3) \text{ sec} = 184 \text{ ms!}$

- Broadband cross-country link:

  D = 50 ms, R = 10 Mbps, M = 1250 bytes

  $L = 50 \text{ ms} + (1250 \times 8) / (10 \times 10^6) \text{ sec} = 51 \text{ ms}$

- A long link or a slow rate means high latency
  - Often, one delay component dominates

Bandwidth-Delay Product

- Messages take space on the wire!

- The amount of data in flight is the <u>bandwidth-delay (BD) product</u>

$$BD = R \times D$$

  - Measure in bits, or in messages
  - Small for LANs, big for “long fat” pipes

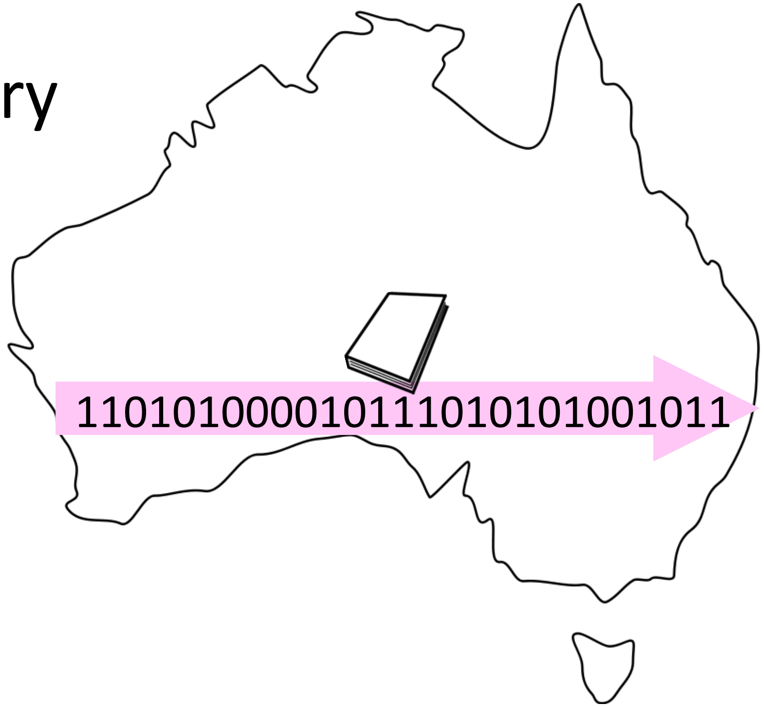
Bandwidth-Delay Example (2)

- Fiber at home, cross-country

  R=40 Mbps, D=50 ms

  $$\begin{aligned} BD &= 40 \times 10^{6} \times 50 \times 10^{-3} \text{ bits} \\ &= 2000 \text{ Kbit} \\ &= 250 \text{ KB} \end{aligned}$$

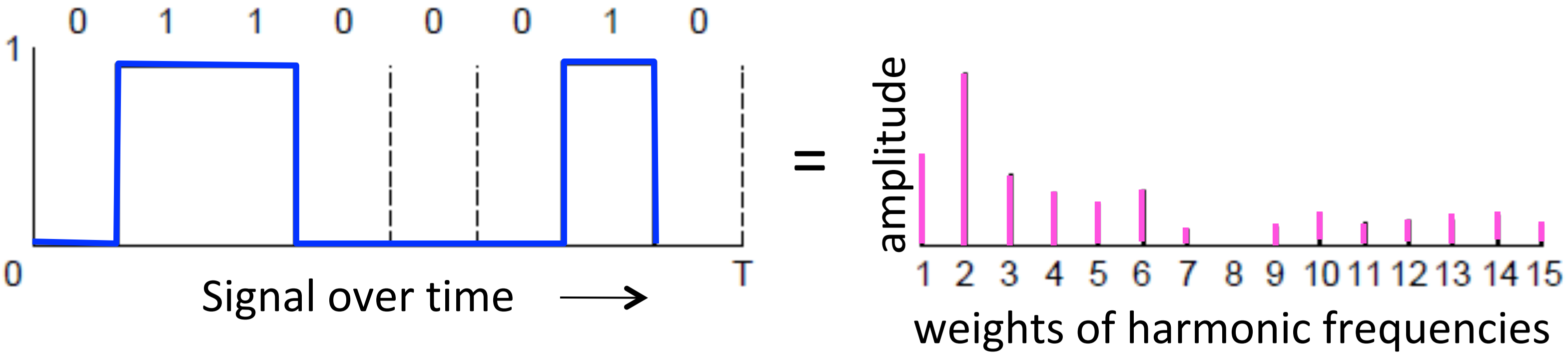
- That's quite a lot of data "in the network"!



Frequency Representation

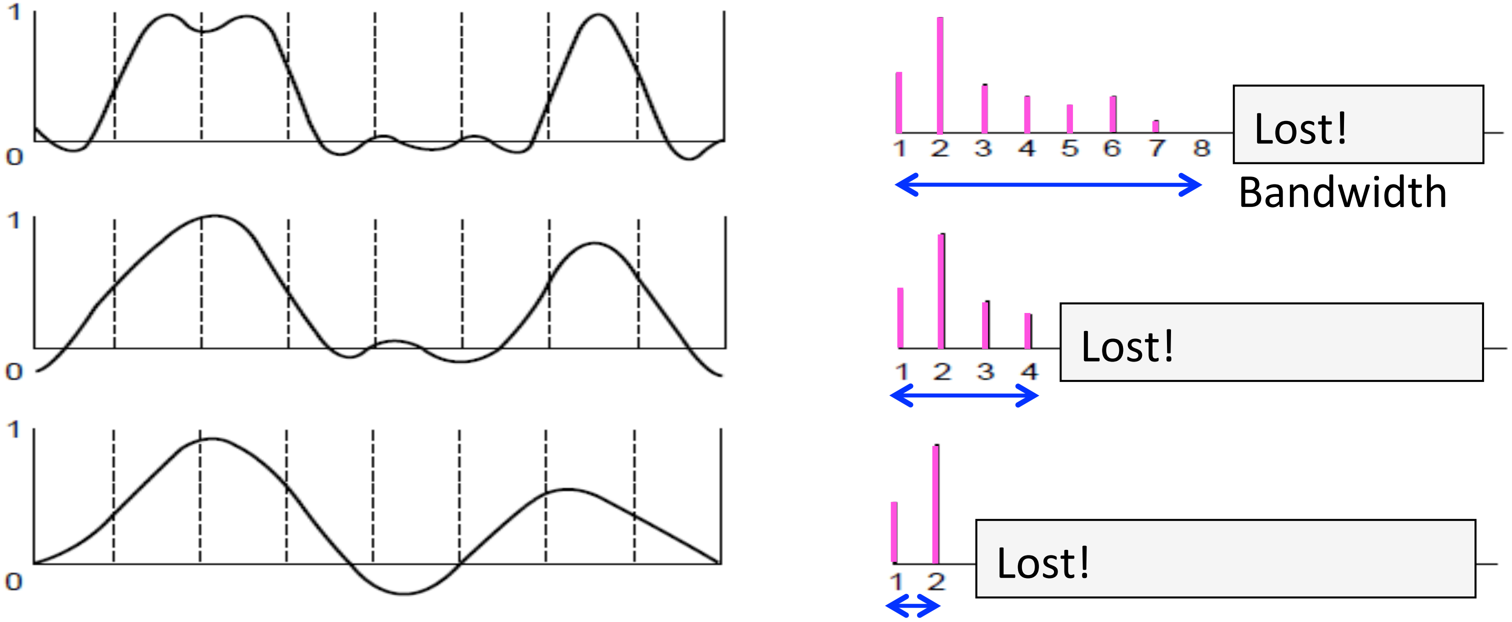
- A signal over time can be represented by its frequency components (called Fourier analysis)

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$



Effect of Less Bandwidth

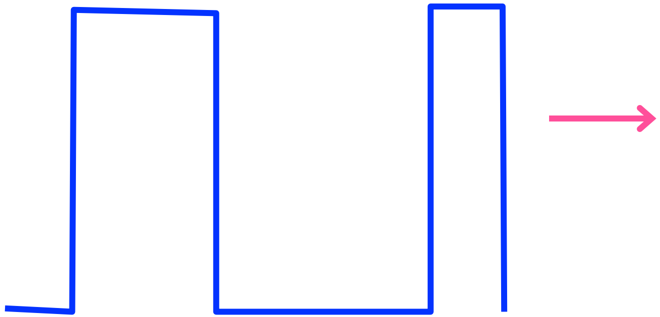
- Fewer frequencies (=less bandwidth) degrades signal



Signals over a Wire (2)

- Example:

Sent signal



2: Attenuation:

3: Bandwidth:

4: Noise:

Signals over Wireless

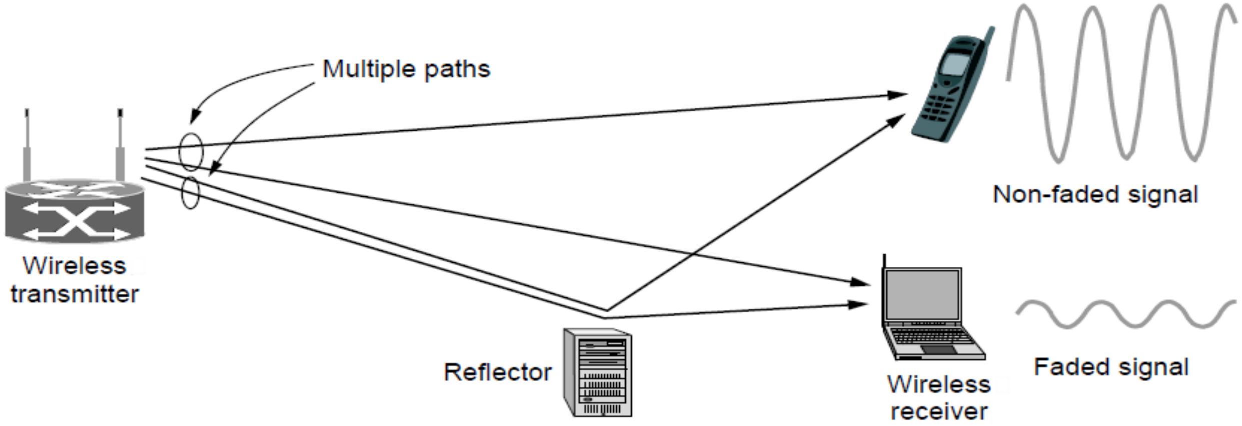
- Signals transmitted on a carrier frequency, like fiber
- Travel at speed of light, spread out and attenuate faster than $1/dist^2$
- Multiple signals on the same frequency interfere at a receiver

Signals over Wireless (5)

- Various other effects too!
  - Wireless propagation is complex, depends on environment
- Some key effects are highly frequency dependent,
  - E.g., <u>multipath</u> at microwave frequencies

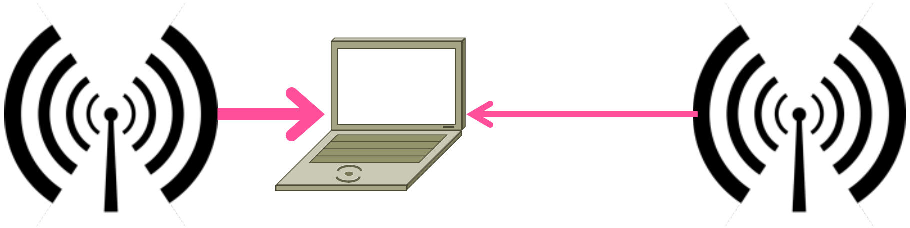
Wireless Multipath

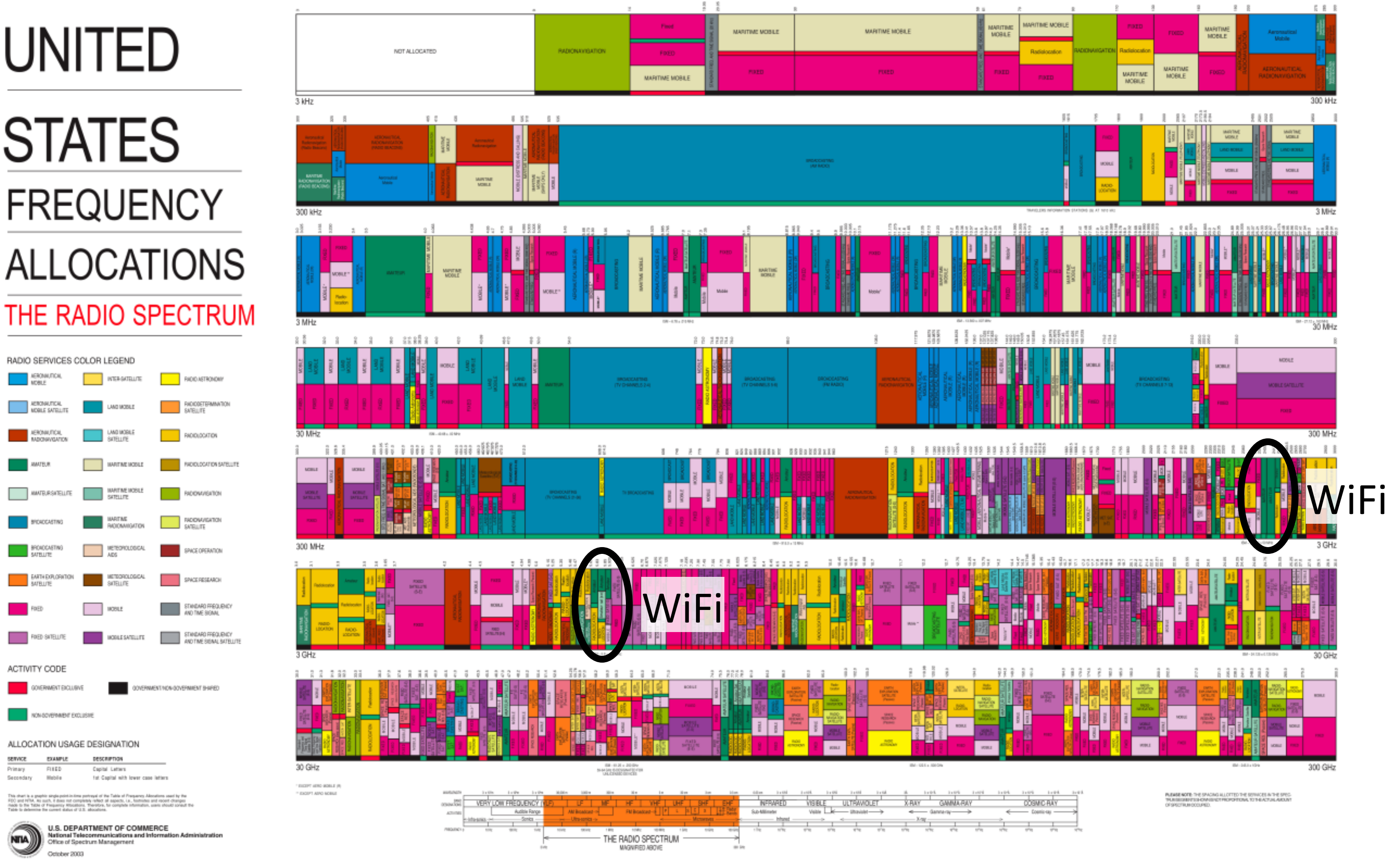
- Signals bounce off objects and take multiple paths
  - Some frequencies attenuated at receiver, varies with location
  - Messes up signal; handled with sophisticated methods



Wireless

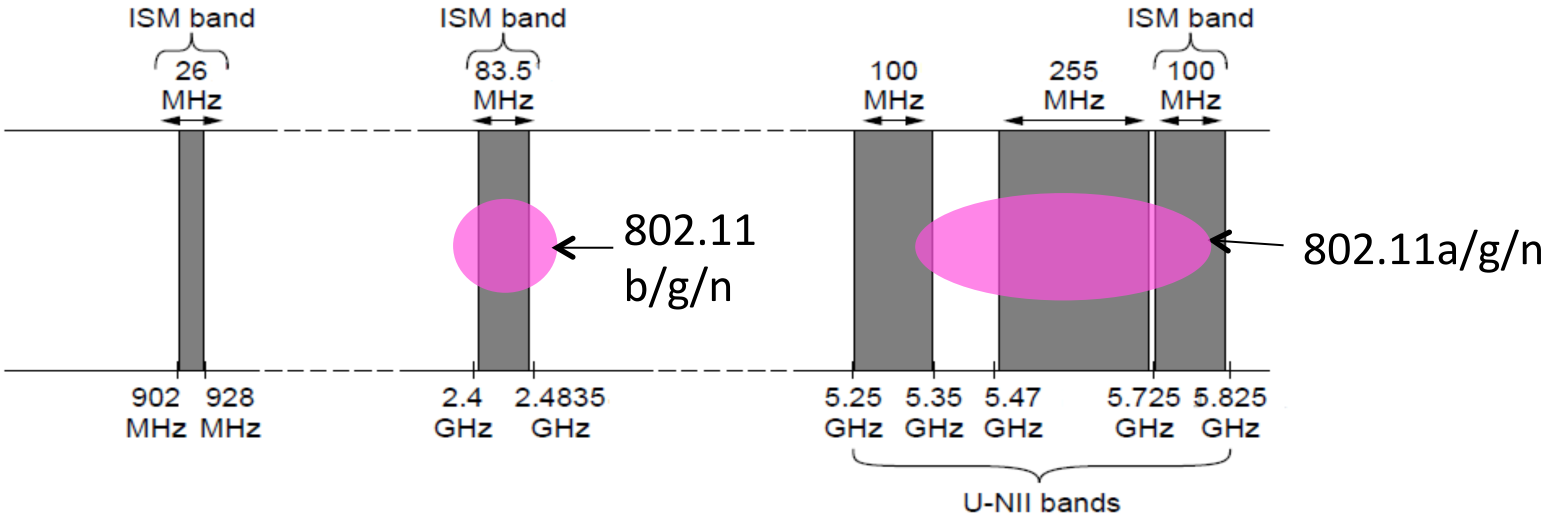
- Sender radiates signal over a region
  - In many directions, unlike a wire, to potentially many receivers
  - Nearby signals (same freq.) <u>interfere</u> at a receiver; need to coordinate use




UNITED
STATES
FREQUENCY
ALLOCATIONS
THE RADIO SPECTRUM
RADIO SERVICES COLOR LEGEND
ACTIVITY CODE
ALLOCATION USAGE DESIGNATION
U.S. DEPARTMENT OF COMMERCE
National Telecommunications and Information Administration
Office of Spectrum Management
October 2003
3 kHz
300 kHz
3 MHz
30 MHz
300 MHz
3 GHz
30 GHz
300 GHz
WiFi
WiFi
THE RADIO SPECTRUM

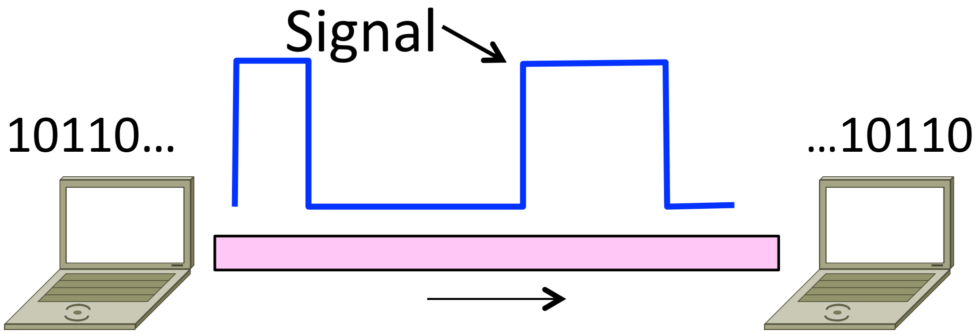
Wireless (2)

- Microwave, e.g., 3G, and unlicensed (ISM) frequencies, e.g., WiFi, are widely used for computer networking



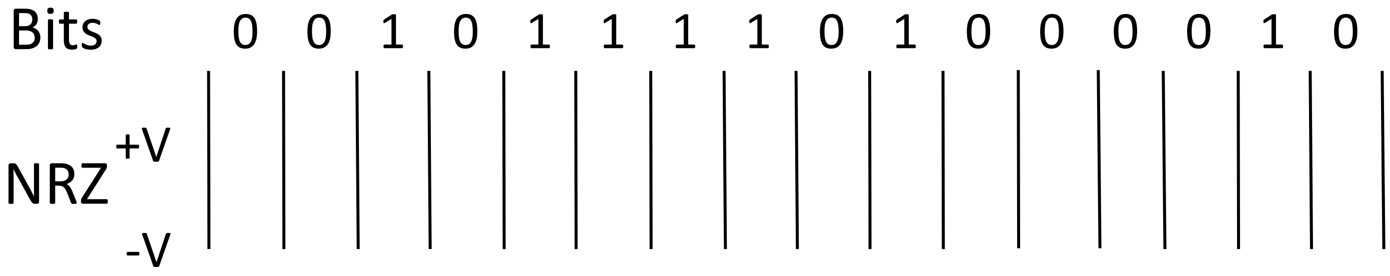
Topic

- We've talked about signals representing bits. How, exactly?
  - This is the topic of <u>modulation</u>



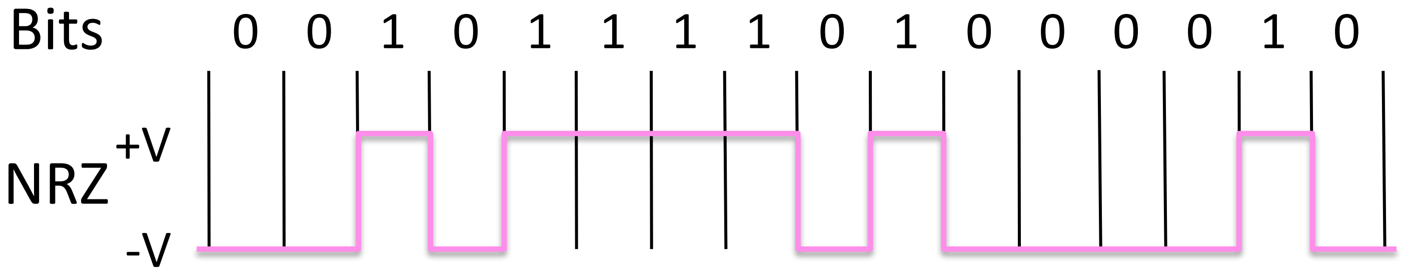
A Simple Modulation

- Let a high voltage (+V) represent a 1, and low voltage (-V) represent a 0
  - This is called NRZ (Non-Return to Zero)



A Simple Modulation (2)

- Let a high voltage (+V) represent a 1, and low voltage (-V) represent a 0
  - This is called NRZ (Non-Return to Zero)



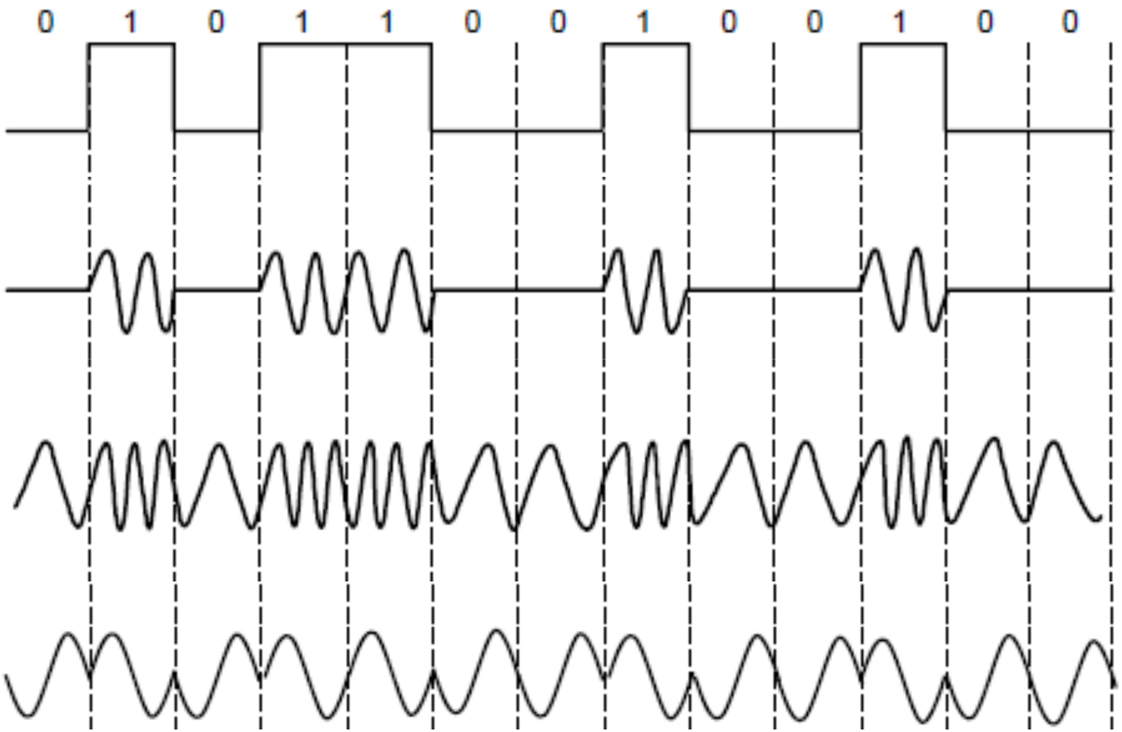
Modulation

NRZ signal of bits

Amplitude shift keying

Frequency shift keying

Phase shift keying



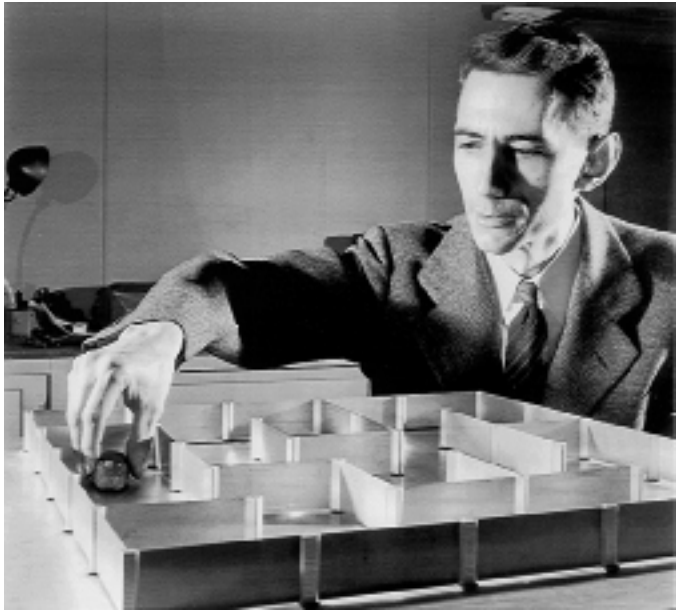
Key Channel Properties

- The bandwidth (B), signal strength (S), and noise strength (N)
  - B limits the rate of transitions
  - S and N limit how many signal levels we can distinguish

Claude Shannon (1916-2001)

- Father of information theory
  - "A Mathematical Theory of Communication", 1948
- Fundamental contributions to digital computers, security, and communications

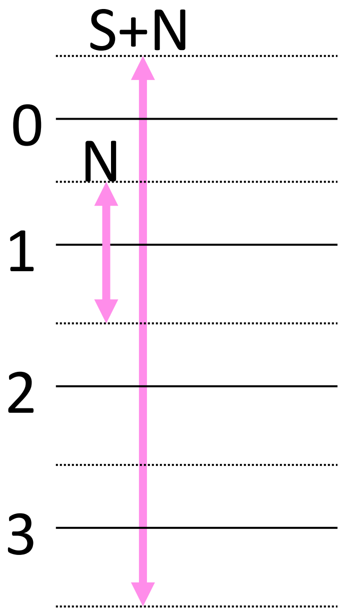
Electromechanical mouse
that "solves" mazes!



Credit: Courtesy MIT Museum

Shannon Capacity

- How many levels we can distinguish depends on S/N
  - Or SNR, the Signal-to-Noise Ratio
  - Note noise is random, hence some errors
- SNR given on a log-scale in deciBels:
  - $SNR_{dB} = 10\log_{10}(S/N)$



Shannon Capacity (2)

- Shannon limit is for capacity (C), the maximum information carrying rate of the channel:

$$C = B \log_2(1 + S/(BN)) \text{ bits/sec}$$

Wired/Wireless Perspective

- Wires, and Fiber
  - Engineer link to have requisite SNR and B
  - →Can fix data rate

- Wireless
  - Given B, but SNR varies greatly, e.g., up to 60 dB!
  - →Can't design for worst case, must adapt data rate

Wired/Wireless Perspective (2)

- Wires, and Fiber
  - Engineer link to have requisite SNR and B

  → Can fix data rate

Engineer SNR for data rate

- Wireless
  - Given B, but SNR varies greatly, e.g., up to 60 dB!

  → Can't design for worst case, must adapt data rate

Adapt data rate to SNR

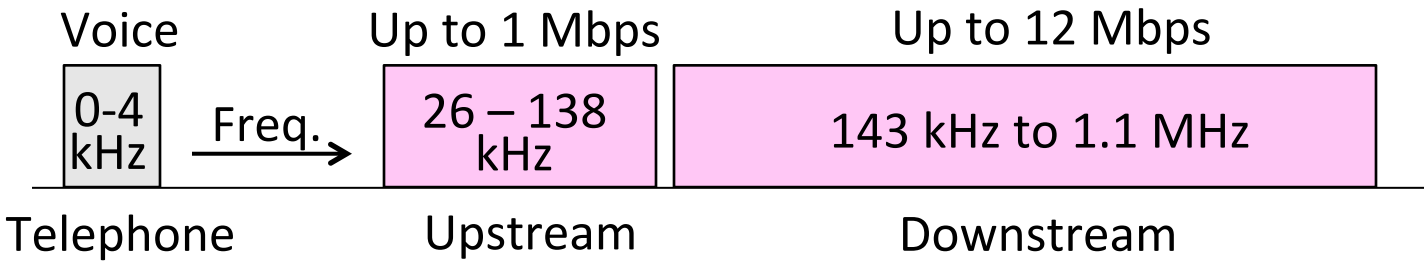
Putting it all together – DSL

- DSL (Digital Subscriber Line) is widely used for broadband; many variants offer 10s of Mbps
  - Reuses twisted pair telephone line to the home; it has up to ~2 MHz of bandwidth but uses only the lowest ~4 kHz

DSL (2)

- DSL uses passband modulation (called OFDM)
  - Separate bands for upstream and downstream (larger)
  - Modulation varies both amplitude and phase (called QAM)
  - High SNR, up to 15 bits/symbol, low SNR only 1 bit/symbol

ADSL2:



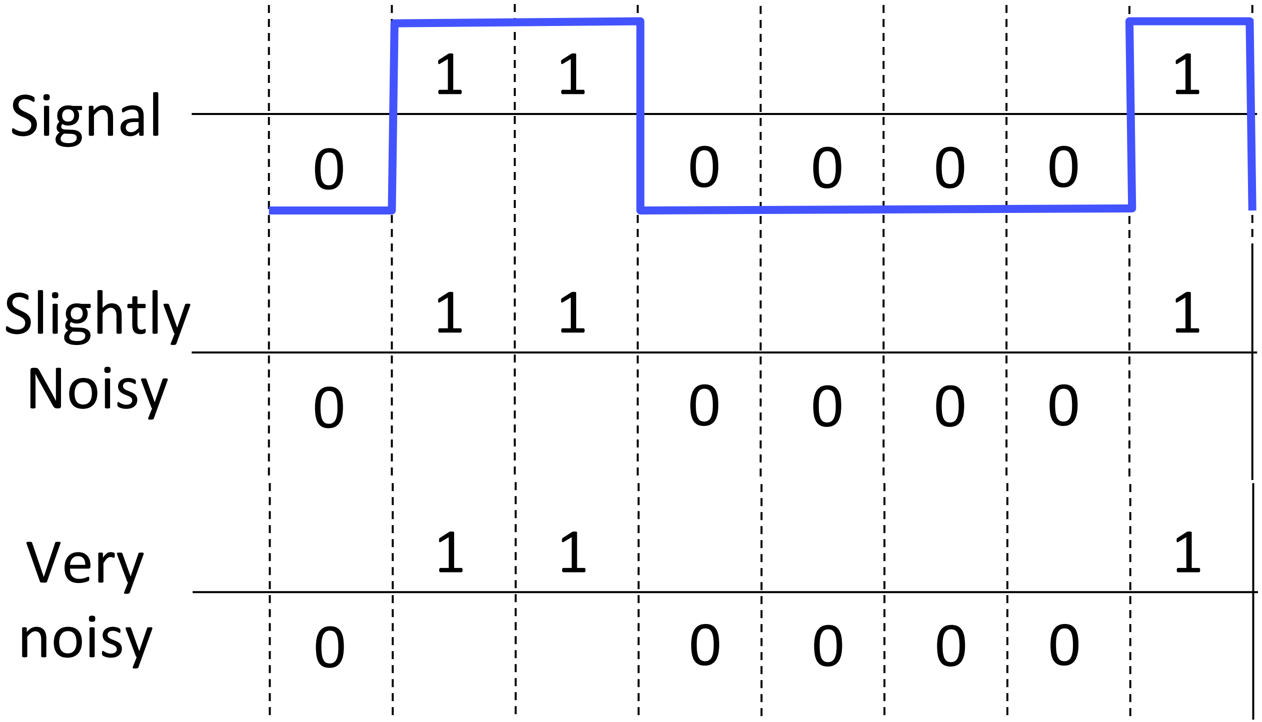
Topic

- Some bits will be received in error due to noise. What can we do?
  - Detect errors with codes »
  - Correct errors with codes »
  - Retransmit lost frames

Later

- Reliability is a concern that cuts across the layers – we'll see it again

Problem – Noise may flip received bits



Approach – Add Redundancy

- Error detection codes
  - Add check bits to the message bits to let some errors be detected
- Error correction codes
  - Add more check bits to let some errors be corrected

- Key issue is now to structure the code to detect many errors with few check bits and modest computation

Motivating Example

- A simple code to handle errors:
  - Send two copies! Error if different.

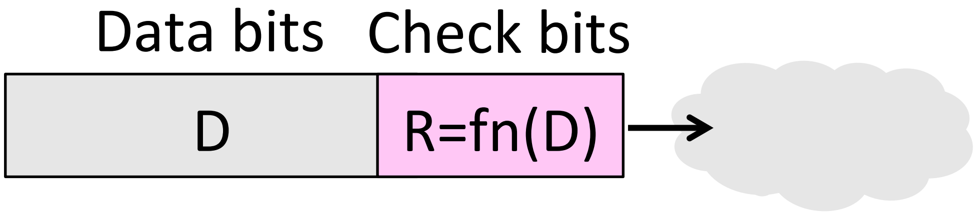
- How good is this code?
  - How many errors can it detect/correct?
  - How many errors will make it fail?

Motivating Example (2)

- We want to handle more errors with less overhead
  - Will look at better codes; they are applied mathematics
  - But, they can’t handle all errors
  - And they focus on accidental errors (will look at secure hashes later)

Using Error Codes

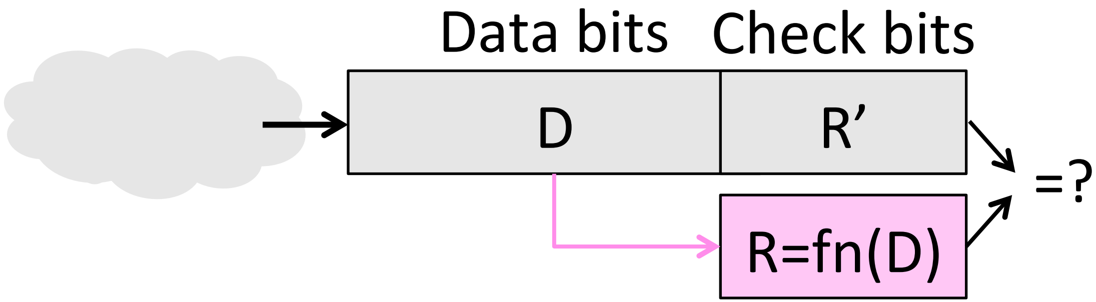
- Codeword consists of D data plus R check bits (=systematic block code)



- Sender:
  - Compute R check bits based on the D data bits; send the codeword of D+R bits

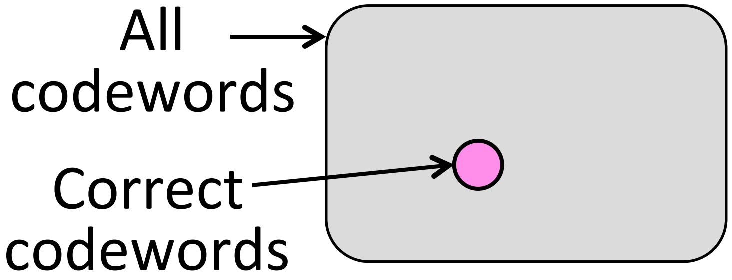
Using Error Codes (2)

- Receiver:
  - Receive D+R bits with unknown errors
  - Recompute R check bits based on the D data bits; error if R doesn't match R'



Intuition for Error Codes

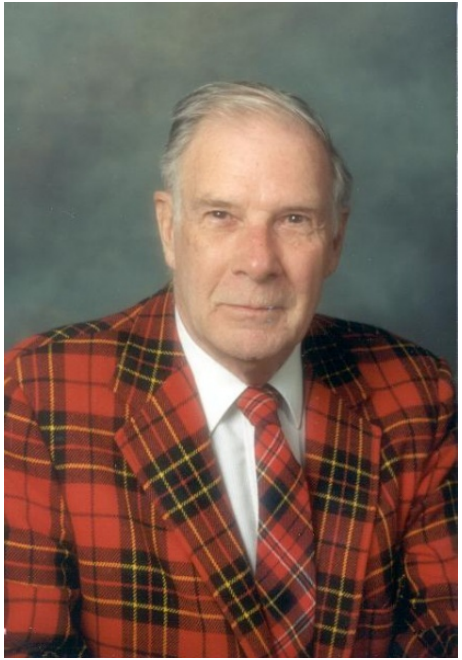
- For D data bits, R check bits:



- Randomly chosen codeword is unlikely to be correct; overhead is low

R.W. Hamming (1915-1998)

- Much early work on codes:
  - "Error Detecting and Error Correcting Codes", BSTJ, 1950
- See also:
  - "You and Your Research", 1986



Source: IEEE GHN, © 2009 IEEE

Hamming Distance

- Distance is the number of bit flips needed to change D_1 to D_2

- Hamming distance of a code is the minimum distance between any pair of codewords

Hamming Distance (2)

- Error detection:
  - For a code of distance d+1, up to d errors will always be detected

Hamming Distance (3)

- Error correction:
  - For a code of distance 2d+1, up to d errors can always be corrected by mapping to the closest codeword

Topic

- Some bits may be received in error due to noise. How do we detect this?
  - Parity »
  - Checksums »
  - CRCs »

- Detection will let us fix the error, for example, by retransmission (later).

Simple Error Detection – Parity Bit

- Take D data bits, add 1 check bit that is the sum of the D bits
  - Sum is modulo 2 or XOR

Parity Bit (2)

- How well does parity work?
  - What is the distance of the code?
  - How many errors will it detect/ correct?
- What about larger errors?

Checksums

- Idea: sum up data in N-bit words
  - Widely used in, e.g., TCP/IP/UDP

1500 bytes	16 bits

- Stronger protection than parity

Internet Checksum

- Sum is defined in 1s complement arithmetic (must add back carries)
  - And it's the negative sum
- *"The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words …"* – RFC 791

Internet Checksum (2)

Sending:

1. Arrange data in 16-bit words
2. Put zero in checksum position, add

3. Add any carryover back to get 16 bits

4. Negate (complement) to get sum

```
0001
f203
f4f5
f6f7
```

Internet Checksum (3)

Sending:

1. Arrange data in 16-bit words
2. Put zero in checksum position, add
3. Add any carryover back to get 16 bits
4. Negate (complement) to get sum

```
   0001
   f203
   f4f5
   f6f7
+(0000)
------
  2ddf0
    ↓
   ddf0
+     2
------
   ddf2
    ↓
   220d
```

Internet Checksum (4)

Receiving:

1. Arrange data in 16-bit words
2. Checksum will be non-zero, add

```
  0001
  f203
  f4f5
  f6f7
+ 220d
------
```

3. Add any carryover back to get 16 bits

4. Negate the result and check it is 0

Internet Checksum (5)

Receiving:

1. Arrange data in 16-bit words
2. Checksum will be non-zero, add
3. Add any carryover back to get 16 bits
4. Negate the result and check it is 0

```
   0001
   f203
   f4f5
   f6f7
+  220d
 ------
  2fffd
    ↓
   fffd
+     2
 ------
   ffff
    ↓
   0000
```

Internet Checksum (6)

- How well does the checksum work?
  - What is the distance of the code?
  - How many errors will it detect/ correct?

- What about larger errors?

Cyclic Redundancy Check (CRC)

- Even stronger protection
  - Given n data bits, generate k check bits such that the n+k bits are evenly divisible by a generator C
- Example with numbers:
  - n = 302, k = one digit, C = 3

CRCs (2)

- The catch:
  - It's based on mathematics of finite fields, in which "numbers" represent polynomials
  - e.g, 10011010 is $x^7 + x^4 + x^3 + x^1$

- What this means:
  - We work with binary values and operate using modulo 2 arithmetic

CRCs (3)

- Send Procedure:

1. Extend the n data bits with k zeros
2. Divide by the generator value C
3. Keep remainder, ignore quotient
4. Adjust k check bits by remainder

- Receive Procedure:

1. Divide and check for zero remainder

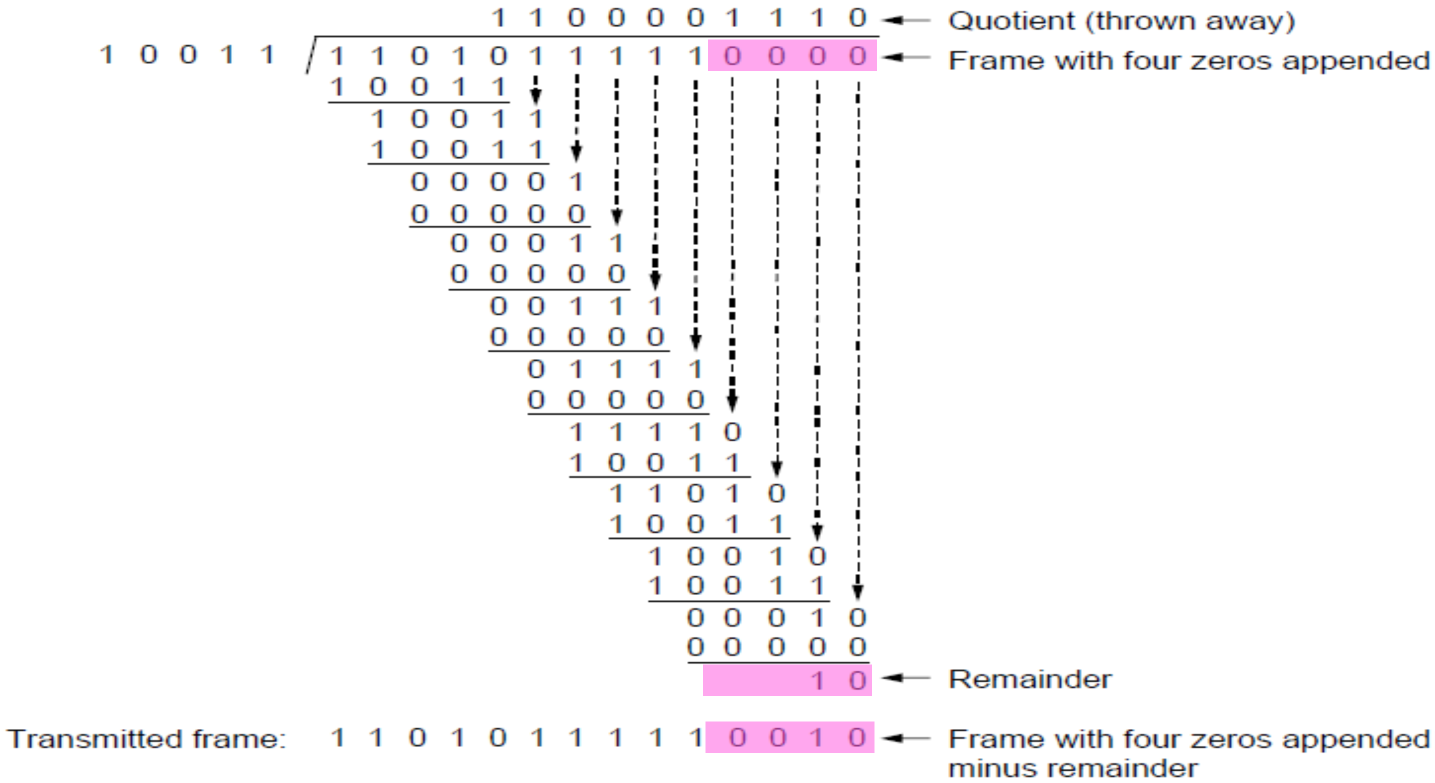
CRCs (4)

Data bits:
1101011111

Check bits:
$C(x)=x^4+x^1+1$
C = 10011
k = 4

$$
10011 \overline{)\,1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\qquad\qquad}
$$

CRCs (5)


1 1 0 0 0 0 1 1 1 0
Quotient (thrown away)
1 0 0 1 1
1 1 0 1 0 1 1 1 1 1 0 0 0 0
Frame with four zeros appended
1 0 0 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 1
0 0 0 0 0
0 0 1 1 1
0 0 0 0 0
0 1 1 1 1
0 0 0 0 0
1 1 1 1 0
1 0 0 1 1
1 1 0 1 0
1 0 0 1 1
1 0 0 1 0
1 0 0 1 1
0 0 0 1 0
0 0 0 0 0
1 0
Remainder
Transmitted frame:
1 1 0 1 0 1 1 1 1 1 0 0 1 0
Frame with four zeros appended minus remainder

CRCs (6)

- Protection depend on generator
  - Standard CRC-32 is 10000010 01100000 10001110 110110111
- Properties:
  - HD=4, detects up to triple bit errors
  - Also odd number of errors
  - And bursts of up to k bits in error
  - Not vulnerable to systematic errors like checksums

Error Detection in Practice

- CRCs are widely used on links
  - Ethernet, 802.11, ADSL, Cable …
- Checksum used in Internet
  - IP, TCP, UDP … but it is weak
- Parity
  - Is little used

Topic

- Some bits may be received in error due to noise. How do we fix them?
  - Hamming code »
  - Other codes »
- And why should we use detection when we can use correction?

Why Error Correction is Hard

- If we had reliable check bits we could use them to narrow down the position of the error
  - Then correction would be easy
- But error could be in the check bits as well as the data bits!
  - Data might even be correct

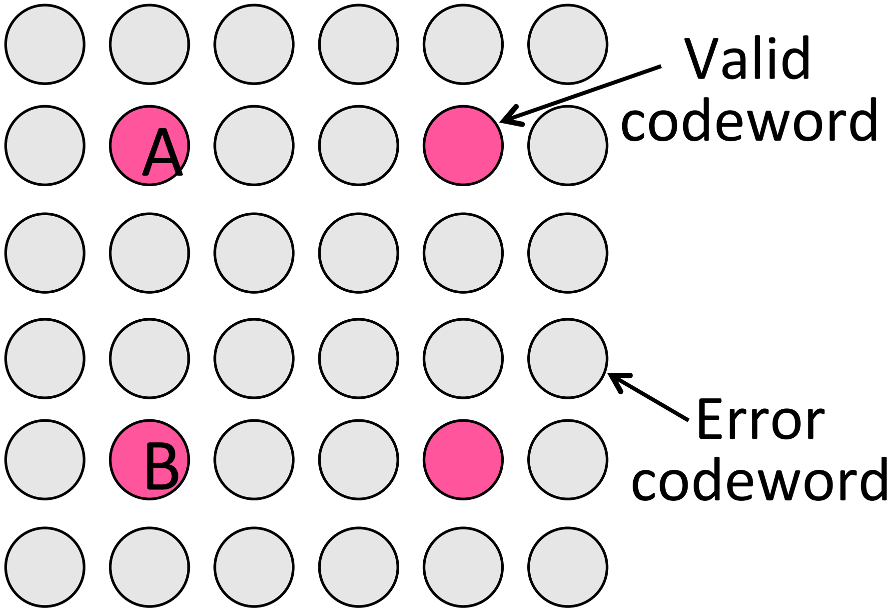
Intuition for Error Correcting Code

- Suppose we construct a code with a Hamming distance of at least 3
  - Need ≥3 bit errors to change one valid codeword into another
  - Single bit errors will be closest to a unique valid codeword

- If we assume errors are only 1 bit, we can correct them by mapping an error to the closest valid codeword
  - Works for d errors if HD ≥ 2d + 1

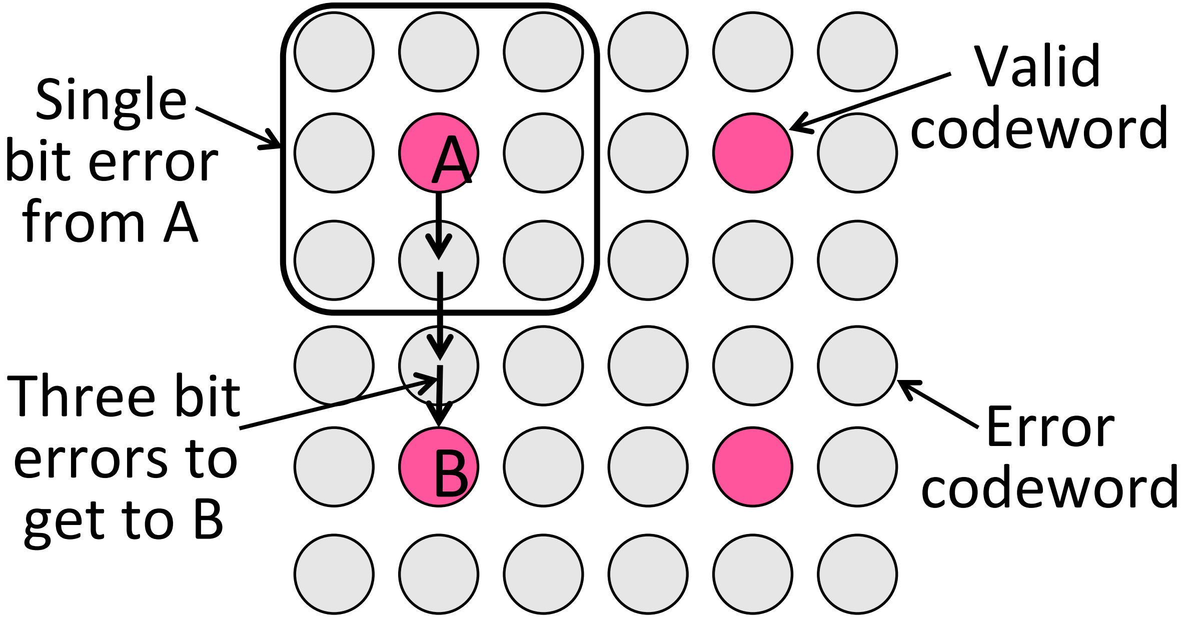
Intuition (2)

- Visualization of code:



Intuition (3)

- Visualization of code:



Hamming Code

- Gives a method for constructing a code with a distance of 3
  - Uses $n = 2^k - k - 1$, e.g., n=4, k=3
  - Put check bits in positions p that are powers of 2, starting with position 1
  - Check bit in position p is parity of positions with a p term in their values
- Plus an easy way to correct [soon]

Hamming Code (2)

- Example: data=0101, 3 check bits
  - 7 bit code, check bit positions 1, 2, 4
  - Check 1 covers positions 1, 3, 5, 7
  - Check 2 covers positions 2, 3, 6, 7
  - Check 4 covers positions 4, 5, 6, 7

_ _ _ _ _ _ _
1 2 3 4 5 6 7

Hamming Code (3)

- Example: data=0101, 3 check bits
  - 7 bit code, check bit positions 1, 2, 4
  - Check 1 covers positions 1, 3, 5, 7
  - Check 2 covers positions 2, 3, 6, 7
  - Check 4 covers positions 4, 5, 6, 7

$$\underline{0}\ \ \underline{1}\ \ 0\ \ \underline{0}\ \ 1\ \ 0\ \ 1 \ \longrightarrow$$

1 2 3 4 5 6 7

$p_1 = 0+1+1 = 0$, $p_2 = 0+0+1 = 1$, $p_4 = 1+0+1 = 0$

Hamming Code (4)

- To decode:
  - Recompute check bits (with parity sum including the check bit)
  - Arrange as a binary number
  - Value (syndrome) tells error position
  - Value of zero means no error
  - Otherwise, flip bit to correct

Hamming Code (5)

- Example, continued

$$
\longrightarrow \begin{array}{ccccccc} \underline{0} & \underline{1} & 0 & \underline{0} & 1 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}
$$

p_1= p_2=

p_4=

Syndrome =

Data =

Hamming Code (6)

- Example, continued

$$\longrightarrow \underline{0}\ \ \underline{1}\ \ 0\ \ \underline{0}\ \ 1\ \ 0\ \ 1$$
$$\begin{matrix}1 & 2 & 3 & 4 & 5 & 6 & 7\end{matrix}$$

$p_1 = 0+0+1+1 = 0$, $p_2 = 1+0+0+1 = 0$,
$p_4 = 0+1+0+1 = 0$

Syndrome = 000, no error
Data = 0 1 0 1

Hamming Code (7)

- Example, continued

$\longrightarrow \quad \underline{0}\ \ \underline{1}\ \ 0\ \ \underline{0}\ \ 1\ \ 1\ \ 1$

$\quad\quad\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7$

p_1= p_2=

p_4=

Syndrome =

Data =

Hamming Code (8)

- Example, continued

$$\longrightarrow \begin{array}{ccccccc} \underline{0} & \underline{1} & 0 & \underline{0} & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

$p_1 = 0+0+1+1 = 0$, $p_2 = 1+0+1+1 = 1$,
$p_4 = 0+1+1+1 = 1$

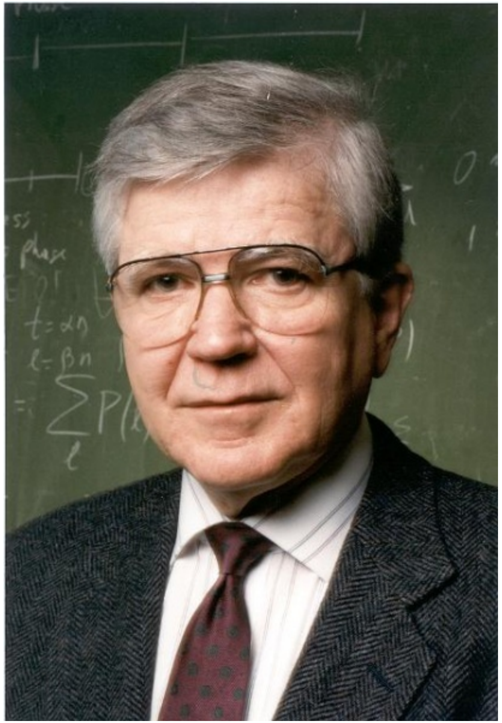
Syndrome = 1 1 0, flip position 6
Data = 0 1 0 1 (correct after flip!)

Other Error Correction Codes

- Codes used in practice are much more involved than Hamming
- Convolutional codes (§3.2.3)
  - Take a stream of data and output a mix of the recent input bits
  - Makes each output bit less fragile
  - Decode using Viterbi algorithm (which can use bit confidence values)

Other Codes (2) – LDPC

- Low Density Parity Check (§3.2.3)
  - LDPC based on sparse matrices
  - Decoded iteratively using a belief propagation algorithm
  - State of the art today
- Invented by Robert Gallager in 1963 as part of his PhD thesis
  - Promptly forgotten until 1996 …



Source: IEEE GHN, © 2009 IEEE

Detection vs. Correction

- Which is better will depend on the pattern of errors. For example:
  - 1000 bit messages with a bit error rate (BER) of 1 in 10000
- Which has less overhead?

Detection vs. Correction

- Which is better will depend on the pattern of errors. For example:
  - 1000 bit messages with a bit error rate (BER) of 1 in 10000
- Which has less overhead?
  - It still depends! We need to know more about the errors

Detection vs. Correction (2)

1. Assume bit errors are random
   - Messages have 0 or maybe 1 error

- Error correction:
  - Need ~10 check bits per message
  - Overhead:

- Error detection:
  - Need ~1 check bits per message plus 1000 bit retransmission 1/10 of the time
  - Overhead:

Detection vs. Correction (3)

2. Assume errors come in bursts of 100
   - Only 1 or 2 messages in 1000 have errors

- Error correction:
  - Need >>100 check bits per message
  - Overhead:

- Error detection:
  - Need 32? check bits per message plus 1000 bit resend 2/1000 of the time
  - Overhead:

Detection vs. Correction (4)

- Error correction:
  - Needed when errors are expected
  - Or when no time for retransmission

- Error detection:
  - More efficient when errors are not expected
  - And when errors are large when they do occur

Error Correction in Practice

- Heavily used in physical layer
  - LDPC is the future, used for demanding links like 802.11, DVB, WiMAX, LTE, power-line, …
  - Convolutional codes widely used in practice
- Error detection (w/ retransmission) is used in the link layer and above for residual errors
- Correction also used in the application layer
  - Called Forward Error Correction (FEC)
  - Normally with an erasure error model
  - E.g., Reed-Solomon (CDs, DVDs, etc.)