Error detection

- **Transmission errors occur**
  - Cosmic rays, radio interference, etc.
  - If error probability is 2^{-20}, that's 1 error per 128 MB!
- **Idea: Detect errors with error-detecting code**
  - Include extra, redundant bits with each message
  - If message changes, extra bits likely to be wrong
- **Examples:**
  - IP, TCP checksums
  - MAC layer error-detection (Ethernet, AAL-5)

Parity

- **Simplest scheme: Parity**
  - For each 7-bits transmitted, transmit an 8th parity bit
  - *Even* parity means total number of 1 bits even
  - *Odd* parity means total number of 1 bits odd
- **Detects any single-bit error (good)**
- **Only detects odd # of bit errors (not so good)**
- **Common errors not caught**
  - E.g., error induces bunch of zeros, valid even parity
- **Can we somehow have multiple parity bits?**

Background: Finite field notation

- **Let Z_2 designate field of integers modulo 2**
  - Two elements are 0 and 1, so an element is a bit
  - Can perform addition and multiplication, just reduce mod 2
  - Example: $1 \cdot 1 = 1$, $1 + 1 = 2 \bmod 2 = 0$
- **Let $\mathbf{Z}_2[x]$ be polynomials w. coefficients in $\mathbf{Z}_2$**
  - I.e., $a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ for $a_i \in \mathbf{Z}_2 = \{0, 1\}$
  - Each a_i is a bit, so can represent polynomial compactly
- **We can multiply, add, subtract polynomials**
  - Example 1: $(x + 1)(x + 1) = x^2 + x + x + 1 = x^2 + 1$ (recall $1 + 1 \equiv 0 \pmod 2$, so $(1 + 1)x = 0$)
  - Example 2: $(x^3 + x^2 + 1) + (x^2 + x) = x^3 + x + 1$
  - Note addition & subtraction are both just XOR

Hamming codes

- **Idea: Use multiple parity bits over subsets of input**
  - Will allow you to detect multiple errors
  - Technique is used by ECC memory
- **Notation: View data as a vector**
  - $D = \begin{pmatrix} d_1 & d_2 & d_3 & d_4 & \cdots \end{pmatrix}$
  - View encoding as multiplication by matrix $G = \begin{pmatrix} I & A \end{pmatrix}$ (where I is the identity matrix)
  - A is specifying how to generate redundant bits
  - Encoded bits $E = D \times G$

Hamming code example

$$
\begin{aligned}
D &= \begin{pmatrix} d_1 & d_2 & d_3 & d_4 \end{pmatrix} \\
G &= \begin{pmatrix} 1 & 0 & 0 & 0 & & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & & 1 & 1 & 0 \end{pmatrix} \\
E &= D \times G = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_1 + d_3 + d_4 \\ d_1 + d_2 + d_4 \\ d_1 + d_2 + d_3 \end{pmatrix}
\end{aligned}
$$

Checking hamming codes

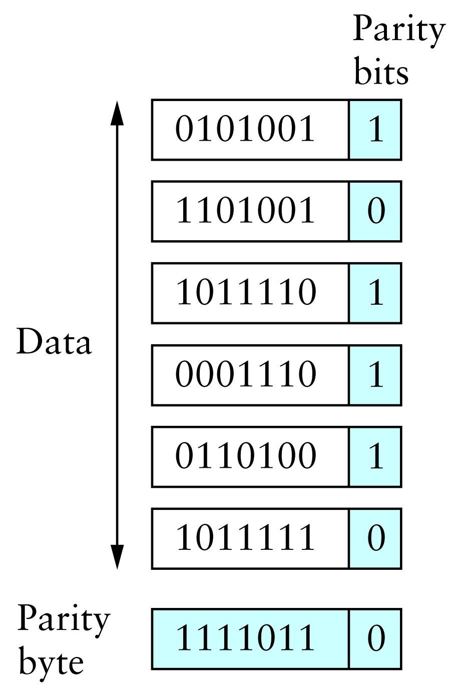
- **Check using** $H = \begin{pmatrix} A^T & I \end{pmatrix}$**:** ***Syndrome*** $s = H \times E$

$$s = \begin{pmatrix} 1 & 0 & 1 & 1 & & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_1 + d_3 + d_4 \\ d_1 + d_2 + d_4 \\ d_1 + d_2 + d_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- **Can detect any two bad bits (if** $s \neq \vec{0}$**)**
- **Can even recover from one incorrect bit!**
  - If one extra bit is 1, it is wrong
  - If two extra bits are 1, d_2, d_3, or d_4 is wrong
  - If all 3 extra bits are 1, d_1 is wrong

2D parity

- **Better if error-detection covers whole message**
- **Idea: Take 2D parity**
  - Catches any 2-bit error, Catches any 1-*byte* error



- **Drawback of all parity schemes: Bandwidth**

Fixed-length codes

- **Idea: Fixed-length code for arbitrary-size message**
  - Calculate code, append to message
  - If code "mixes up the bits" enough, will detect many errors
  - n-bit code should catch all but 2^{-n} faction of errors
  - But want to make sure that includes all common errors
- **Example: IP checksum**

```
u_short
cksum (u_short *buf, int count)
{
  u_long sum = 0;
  while (count--)
    if ((sum += *buf) & 0xffff) /* carry */
      sum = (sum & 0xffff) + 1;
  return ~(sum & 0xffff);
}
```

How good is IP checksum?

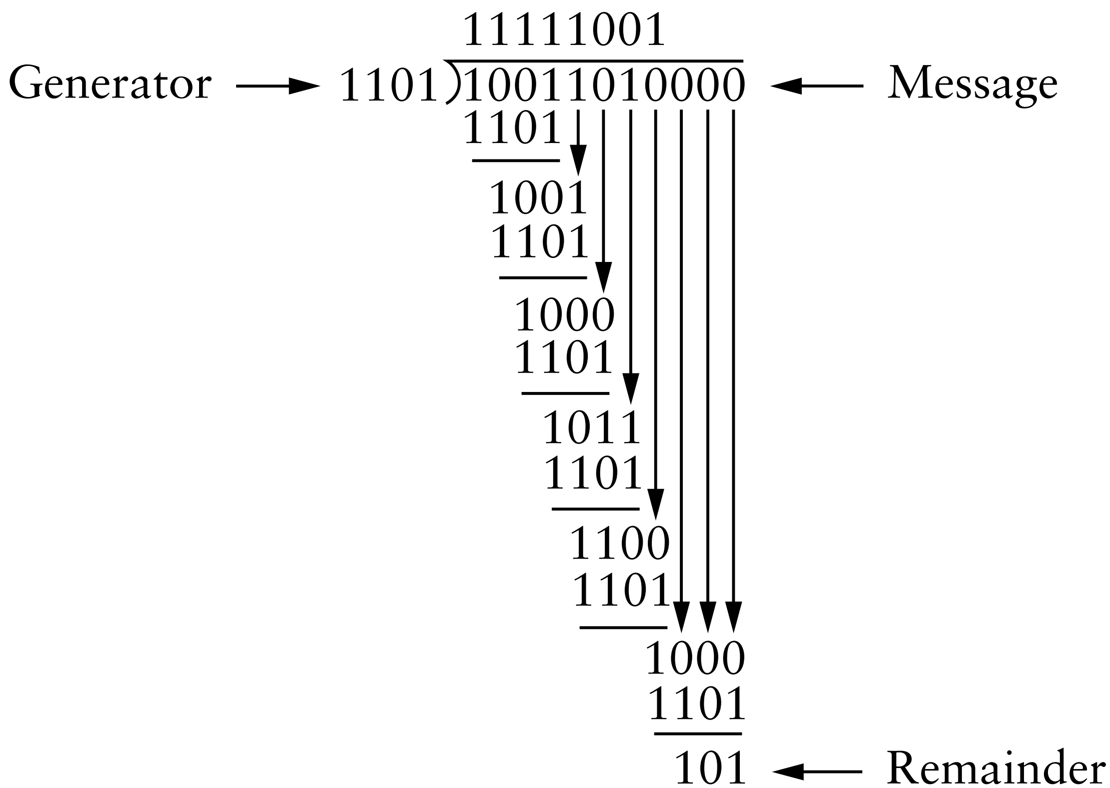
- **16 bits is not very long (misses 1/65K errors)**
- **Checksum does catch any 1-bit error**
- **But not any two-bit error**
  - E.g., increment one word ending 0, decrement one ending 1
- **Checksum also optional on UDP**
  - All 0s means no checksum calculated
  - If checksum word gets wiped to 0 as part of error, bad news

Error-detection with polynomials

- **Consider a message to be a polynomial in $\mathbf{Z}_2[x]$**
  - Each bit corresponds to one coefficient
  - E.g., message $10011010 \Longrightarrow m(x) = x^7 + x^4 + x^3 + x$
- **Can reduce one polynomial *modulo* another**
  - Let $n(x) = m(x)x^3$. Let $C(x) = x^3 + x^2 + 1$.
  - Find $q(x)$ and $r(x)$ such that $n(x) = q(x)C(x) + r(x)$ and the degree of $r(x) <$ degree of $C(x)$.
  - Analogous to computing $11 \bmod 5 = 1$

Polynomial division example

- **Just long division, but addition/subtraction is XOR**



Cyclic Redundancy Check (CRC)

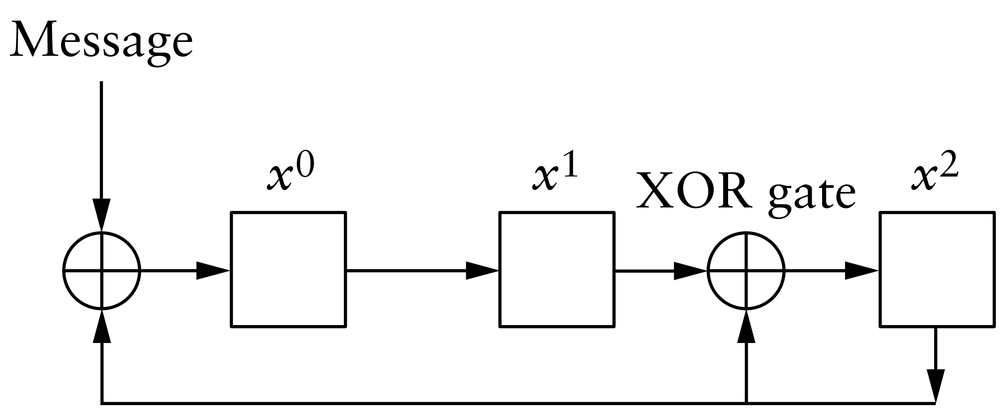
- **Select a divisor polynomial $C(x)$ of degree k**
  - $C(x)$ should be *irreducible*—not expressible as product of two lower-dgree polynomials in $\mathbf{Z}_2[x]$
- **Add k bits to message to make it divisible by $C(x)$**
  - Let $n(x) = m(x)x^k$ (message as polynomial w. k 0s added)
  - Compute $r(x) \leftarrow n(x) \bmod C(x)$
  - Compute $n(x) \leftarrow n(x) - r(x)$, will be divisible by $C(X)$ (Note subtraction is XOR, with 0s just setting lower bits)
- **Checking CRC is easy**
  - Reduce message by $C(x)$, make sure remainder is 0

Why is this good?

- **Suppose you send $m(x)$, recipient gets $m'(x)$**
  - Exact error $E(x) = m'(x) - m(x)$ (all the incorrect bits)
  - If CRC fails to catch error, $C(x)$ divides $m'(x)$
  - Therefore, if CRC fails to catch, $C(x)$ must divide $E(x)$
- **Chose $C(x)$ that doesn't divide any common errors!**
  - All single-bit errors caught if x^k, x^0 coefficients in $C(x)$ are 1
  - All 2-bit errors caught if at least 3 terms in $C(x)$
  - Any odd # errors caught if last two terms $x + 1$
  - Any error burst of less than length k caught

CRC in hardware

- **Recall from long division**
  - Always XOR $C(x)$ with left of message to make first bit 0
  - Build hardware with shift registers
  - Shift in bits starting with highest term coefficient of $m(x)$
  - When top coefficient non-zero, XOR in polynomial
  - I.e., put XOR before x^d box if x^d is term in $C(x)$
- **CRC with $x^3 + x^2 + 1$:**



Error correction

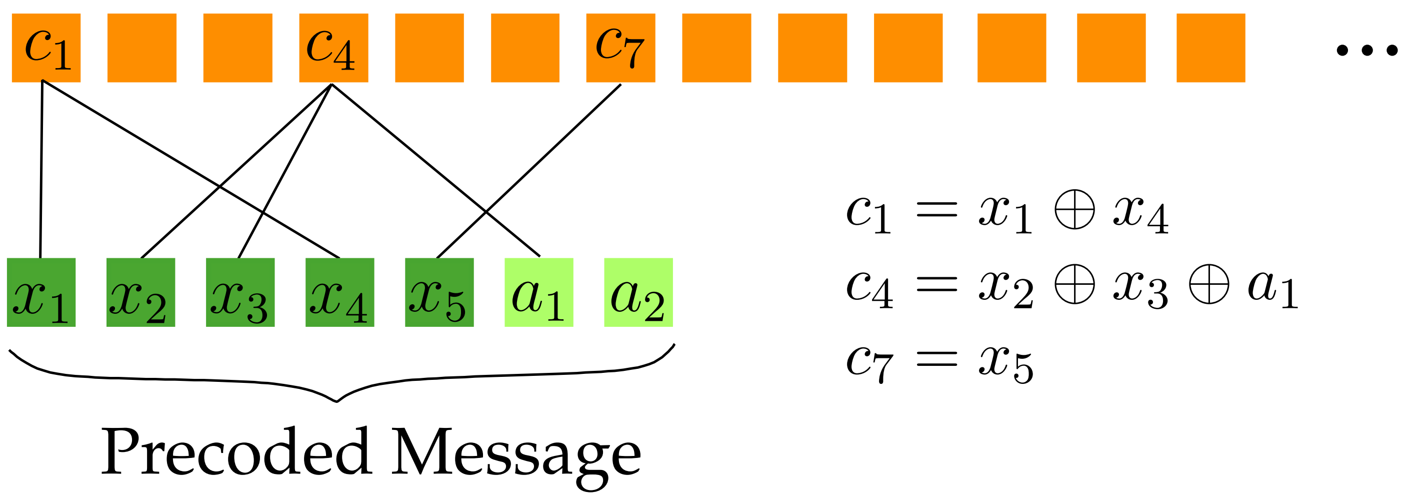
- **Already saw how hamming codes can correct bits**
- **More often interested in recovering lost messages**
  - Can detect bad packets using CRC and discard
  - Might like to recover from lost packets automatically
- **Technique known as *erasure codes***
  - I.e., recover from erased blocks, not corrupt ones
  - Sender just sends more than n pkts for n-pkt message
  - Therefore also known as *forward error correction*

Polynomial interpolation

- **Break message into elements of a finite field F**
  - E.g., $a_n, a_{n-1}, \ldots a_0$—each a_i might be 16 bits
  - Only one degree-n polynomial $m(x) \in F[x]$ will satisfy $m(0) = a_0, m(1) = a_1, \ldots, m(n) = a_n$
  - Use Lagrange interpolation to compute $m(x)$
- **Now evaluate $m(x)$ for $x > n$—creates more blocks**
  - Receiver can interpolate $m(x)$ given any n values
  - Then get message by computing $m(n), m(n-1), \ldots m(0)$
- **Problem: Slow for large messages ($O(n^2)$)**

Efficient codes

- **Recent erasure codes much more efficient—$O(n \log n)$ and $O(n)$**
  - Tornado codes, LT-codes, Raptor codes, On-line codes
  - Require slightly more than n blocks to reconstruct
- **Compute check blocks as XOR of message blocks**
  - But XOR graph structure surprisingly irregular & tricky



$$c_1 = x_1 \oplus x_4$$
$$c_4 = x_2 \oplus x_3 \oplus a_1$$
$$c_7 = x_5$$

When to use error detection & correction

- **At data-link layer, bad to deliver corrupt packets**
  - Actually, theoretically should be fine
  - But IP checksums are not good
- **Often not worth reconstructing packets**
  - Example: Say 1 in 10^6 packets corrupted
  - Retransmission requires negligible bandwidth
  - But sending redundancy for every packet not negligible
- **Want to avoid noticeable loss fraction**
  - Recall TCP uses packet loss as a sign of congestion
  - High loss because of transmission failure hurts performance

Quiz Review

- **Open book**
  - Bring text and papers, you will need them!
  - All class notes on line, feel free to print and bring
  - Books & papers only; no laptops, cell phones, ...
- **Topics:**
  - Most of chapters 1–6 in text + section 9.1
    exact chapters on syllabus web page
  - Lectures 1–13
  - All papers, *except* TCP in ANSNET

Bandwidth

- **Data units (using $K = 1,024 = 2^{10}$):**
  - 1 Byte = 8 bits
  - 1 KByte = 1,024 Bytes
  - 1 MByte = 1,024×1,024 Bytes
  - 1 GByte = 1,024×1,024×1,024 Bytes
- **Clock rates use $K = 1000 = 10^3$:**
  - 1 KHz = 1,000 Hz, 1 Mhz = 1,000,000 Hz, etc.
- **Bandwidth usually uses clock rates:**
  - 1 Mbps = 1 Mega-(bit-per-second) = 1,000,000 bits/sec
  - Note *b* in Mbps can also mean *bytes*
    In this class, will use b for bit, B for byte

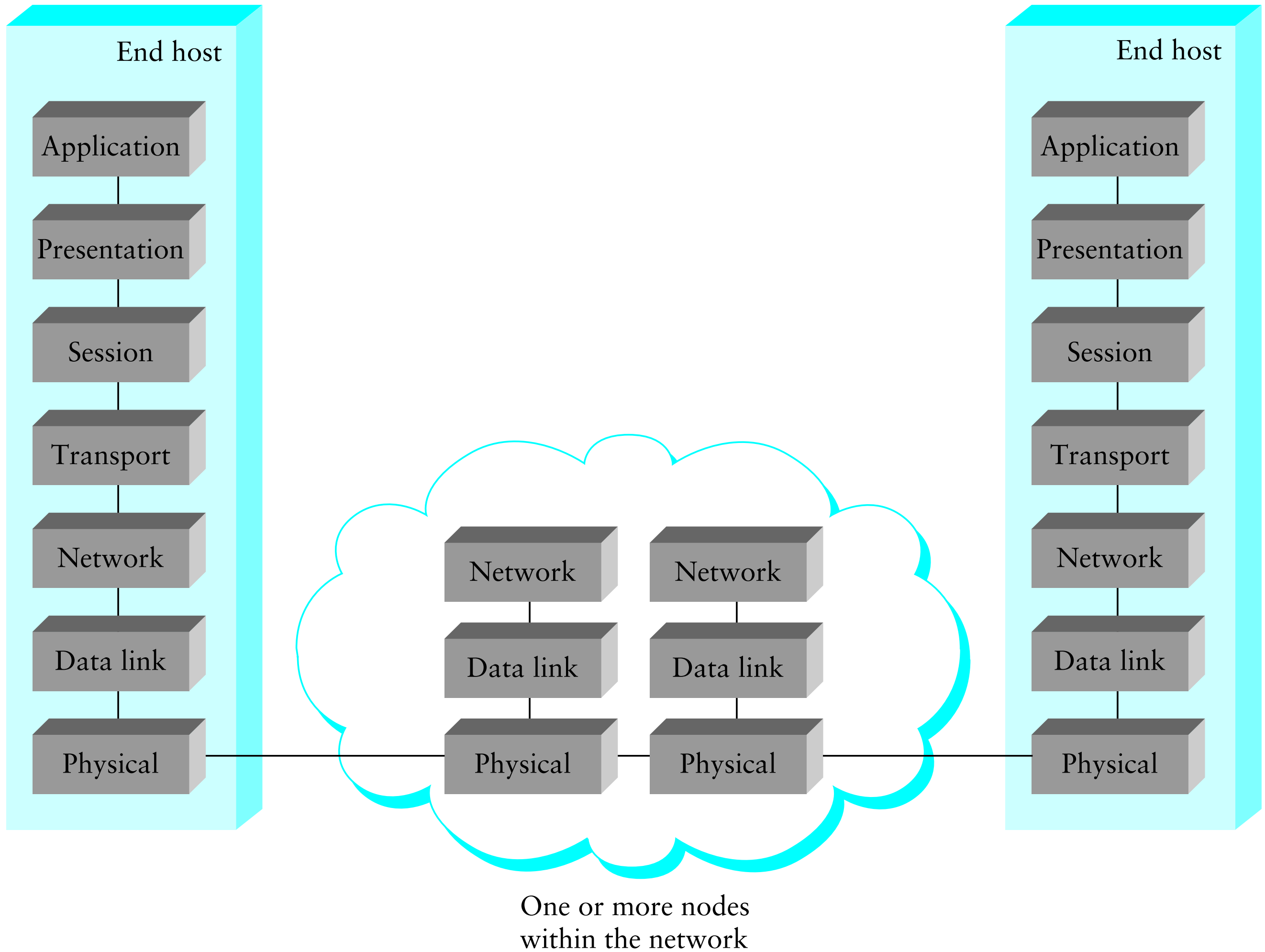
Latency

- **Latency=Propagation+Transmission+Queing**
  - Propagation=Distance/SpeedOfLight
  - Transmission=Size/Bandwidth
- **Transmit 1 Byte @1 Gbps over 1,000 km fiber**
  - Speed of light in fiber is $2 \cdot 10^5$ km/s
  - Propagation $\dfrac{10^3 \text{ km}}{10^5 \text{ km/s}} = 5 \cdot 10^{-3} \text{ sec} = 5 \text{ msec}$
  - Transmission = $8 \cdot 10^{-9}$ sec (negligible)
- **Transmitting 1 GB over same channel**
  - Transmission time 8 sec, dominates

Possible quiz questions

- **What is latency of transmitting message over particular channel?**
- **What is latency of sending when using some particular flow control protocol?**
  - Recall sliding window protocol
  - May not be able to use full link bandwidth, waiting for acknowledgment
- **Why do you need certain features of sliding window protocol?**
  - Example: Here’s a broken protocol, show a scenario in which recipient misinterprets packets

OSI layers



End host
Application
Presentation
Session
Transport
Network
Data link
Physical
Network
Data link
Physical
Network
Data link
Physical
End host
Application
Presentation
Session
Transport
Network
Data link
Physical
One or more nodes
within the network

Physical layer topics

- **How to transmit a bit**
  - NRZ – simplest $0 = 0$, $1 = 1$, but baseline wander & clock recovery problems
  - Manchester, 4B/5B, NRZI
- **Framing**
  - Sentinel based, used character & bit stuffing
  - Counter (length in header); framing errors
  - Clock-based (SONET) – sync with repeated header

Data-link layer

- **Medium Access Control (MAC)**
  - CSMA/CD – Ethernet reacts to collisions
  - Token ring
- **Possible questions: Why are systems as they are?**
  - E.g., Monitor delays bits to ensure token rotation time is less than transmission time…what if it didn't?
  - Answer: When no one is transmitting, need enough storage in the ring to hold token
- **Implementation and driver issues (Afterburner)**

Switching

- **Circuit switched vs. packet switched**
- **Source routing**
- **Avoiding loops (should be familiar by now…)**
- **Also, learning bridges to avoid unnecessary traffic**
- **ATM cells 53-bytes, connection oriented**
  - CS-PDUs: AAL4 and AAL5. Why is AAL5 better?
  - Answer: Uses less overhead, catches as many (more) errors with single bigger checksum

TCP/IP topics

- **Structure of an IP address**
  - Classes & CIDR aggregation
- **IP header fields (how does traceroute work)**
- **TCP and UDP**
  - Headers & Port numbers, how do applications use
  - Hint: Figure 5.7 in textbook should make sense
- **ICMP**
- **How does IP work over Ethernet**
  - ARP protocol translates IP to Ethernet addresses
- **DNS (domain name system)**
  - How name `www.nyu.edu` gets mapped to 128.122.108.74

Routing

- **Distance Vector**
  - Ways of avoiding loops
- **Link state**
  - Dijkstra's algorithm
- **Possible questions:**
  - How will particular system stabilize?
  - What is damage a bad router can do with either approach?
- **BGP concepts**
  - Autonomous Systems (ASes)

More TCP

- **How TCP flow control works (advertised window)**
- **How TCP congestion control works**
  - AIMD Sawtooth pattern
  - Slow start
  - Fast retransmit
- **Routers: Scheduling discipline & Drop policy**
  - Fair queuing, RED, FPQ
- **Possible questions:**
  - What happens to TCP with high transmission error rate?
  - Why are extensions needed for high bandwidth$\times$delay networks?