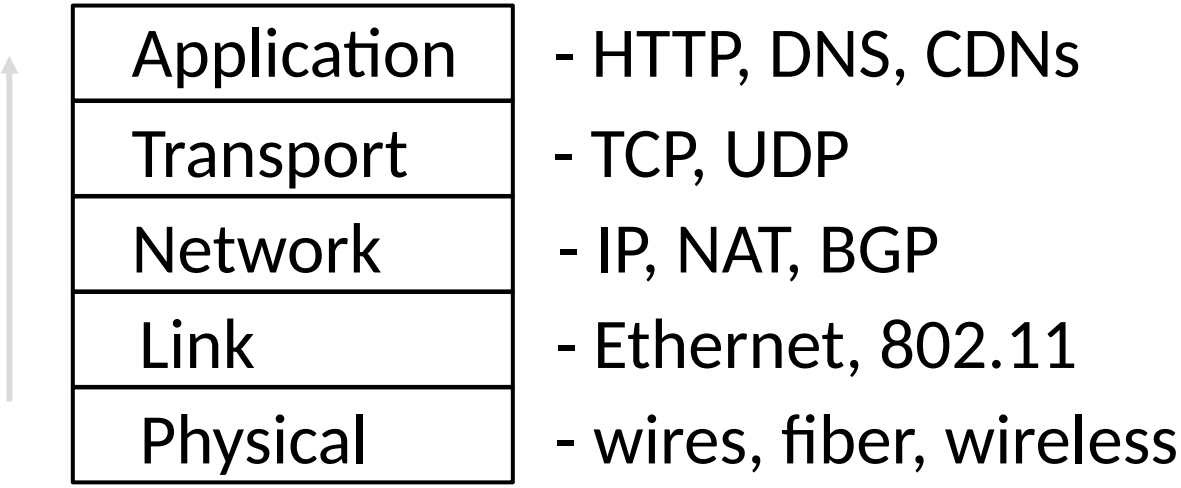
Physical Layer

Lecture Progression

- Bottom-up through the layers:

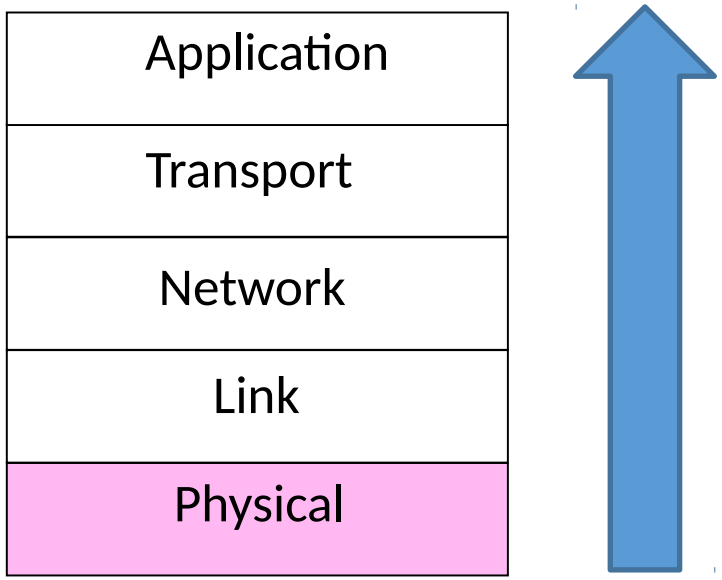


- HTTP, DNS, CDNs
- TCP, UDP
- IP, NAT, BGP
- Ethernet, 802.11
- wires, fiber, wireless

- Followed by more detail on:
  - Quality of service, Security (VPN, SSL)

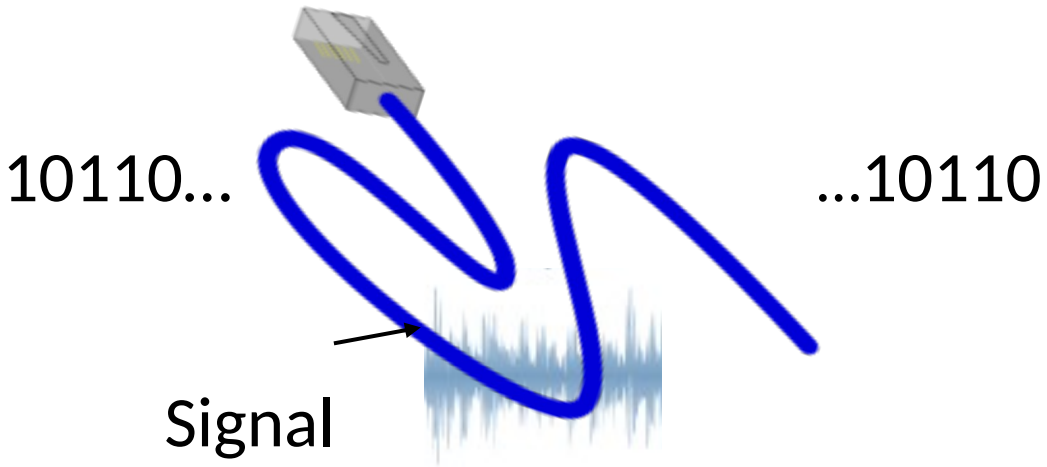
Where we are in the Course

- Beginning to work our way up starting with the Physical layer



Scope of the Physical Layer

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



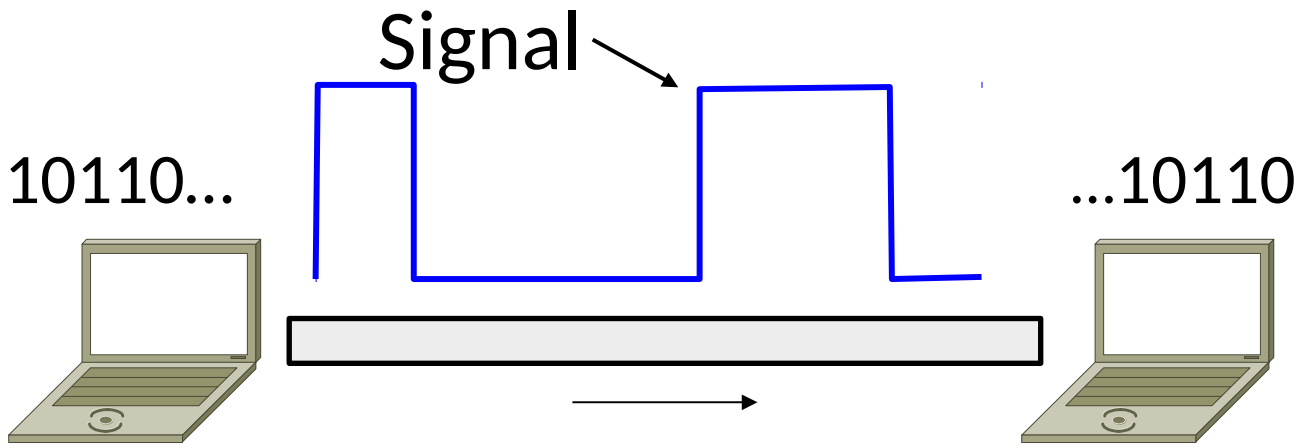
Topics

1. Coding and Modulation schemes
   - Representing bits, noise
2. Properties of media
   - Wires, fiber optics, wireless, propagation
   - Bandwidth, attenuation, noise
3. Fundamental limits
   - Nyquist, Shannon

Coding and Modulation

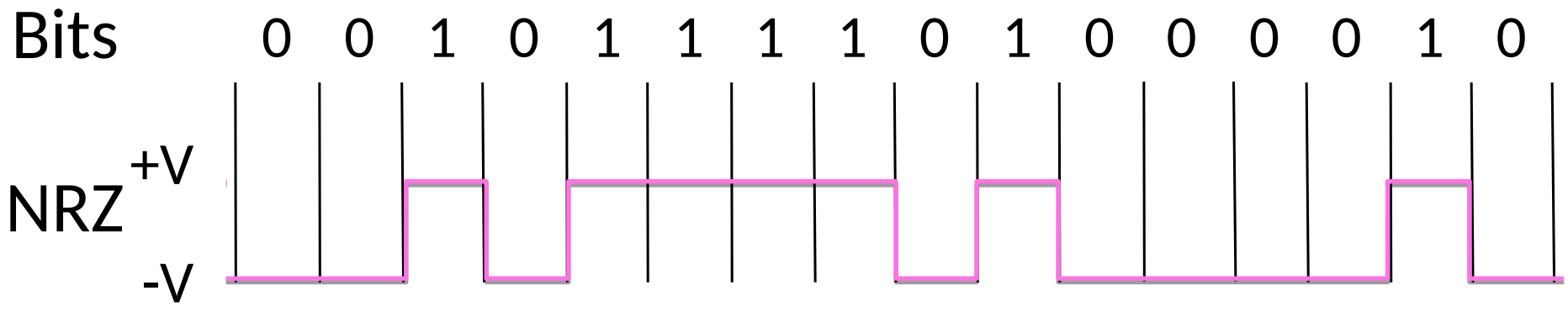
Topic

- How can we send information across a link?
  - This is the topic of coding and modulation
  - Modem (from modulator–demodulator)



A Simple Coding

- 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 (3)

- Problems?

Many Other Schemes

- Can use more signal levels
  - E.g., 4 levels is 2 bits per symbol
- Practical schemes are driven by engineering considerations
  - E.g., clock recovery

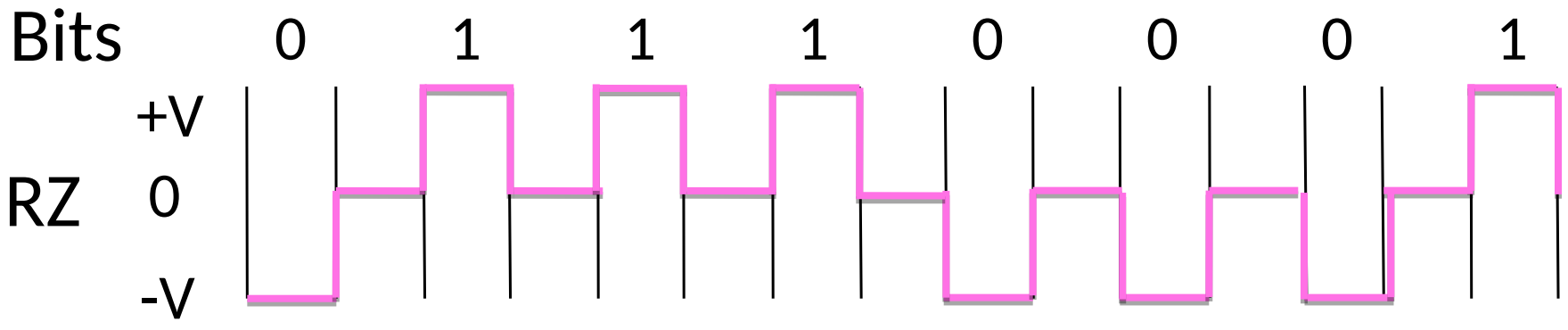
Clock Recovery

- Um, how many zeros was that?
  - Receiver needs frequent signal transitions to decode bits

- Several possible designs
  - E.g., Manchester coding and scrambling (§2.5.1)

Answer 1: A Simple Coding

- Let a high voltage (+V) represent a 1, and low voltage (-V) represent a 0
- Then go back to 0V for a “Reset”
  - This is called RZ (Return to Zero)



Answer 2: Clock Recovery – 4B/5B

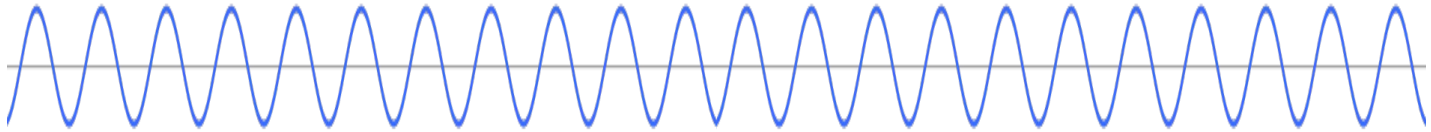
- Map every 4 data bits into 5 code bits without long runs of zeros
  - 0000 → 11110, 0001 →01001, 1110 → 11100, … 1111 → 11101
  - Has at most 3 zeros in a row
  - Also invert signal level on a 1 to break up long runs of 1s (called NRZI, §2.5.1)

Modulation vs Coding

- What we have seen so far is called coding
  - Signal is sent directly on a wire
- These signals do not propagate well as RF
  - Need to send at higher frequencies
- Modulation carries a signal by modulating a carrier
  - Baseband is signal pre-modulation
  - Keying is the *digital* form of modulation (equivalent to coding but using modulation)

Passband Modulation (2)

- Carrier is simply a signal oscillating at a desired frequency:



- We can modulate it by changing:
  - Amplitude, frequency, or phase

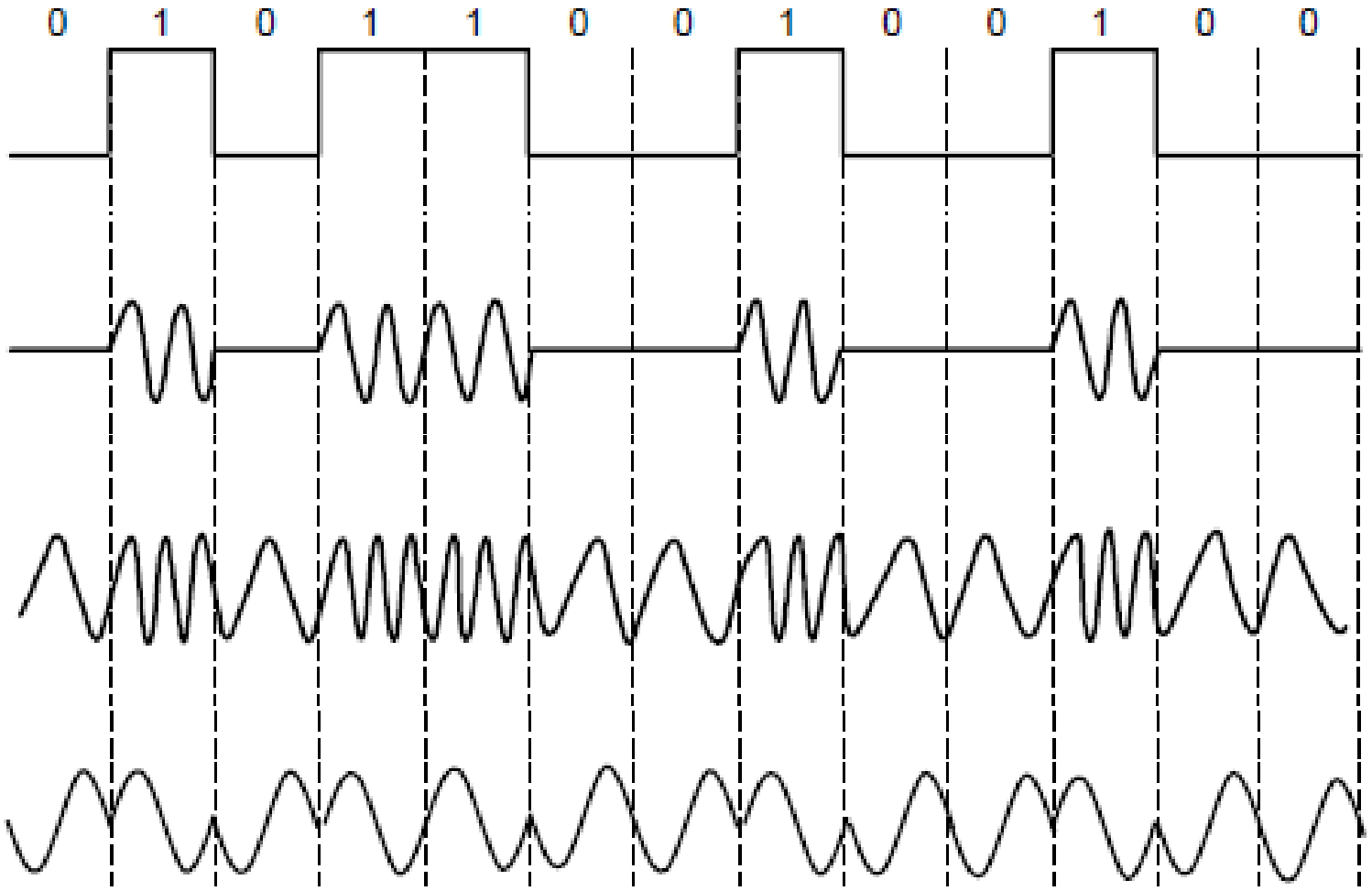
Comparisons

NRZ signal of bits

Amplitude shift keying

Frequency shift keying

Phase shift keying



Philosophical Takeaways

- Everything is analog, even digital signals
- Digital information is a *discrete* concept represented in an analog physical medium
  - A printed book (analog) vs.
  - Words conveyed in the book (digital)

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"

    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: L = M/R + D

Latency Examples

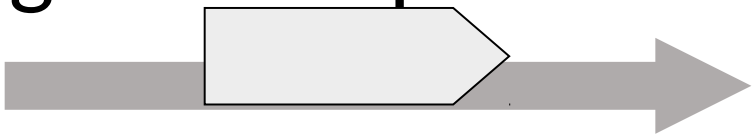
- "Dialup" with a telephone modem:
  - D = 5 ms, R = 56 kbps, M = 1250 bytes
  - L = (1250x8)/(56 x 10^3) sec + 5ms = 184 ms!

- Broadband cross-country link:
  - D = 50 ms, R = 10 Mbps, M = 1250 bytes
  - L = (1250x8) / (10 x 10^6) sec + 50ms = 51 ms

- A long link or a slow rate means high latency: **One component dominates**

Bandwidth-Delay Product

- Messages take space on the wire!



- The amount of data in flight is the bandwidth-delay (BD) product

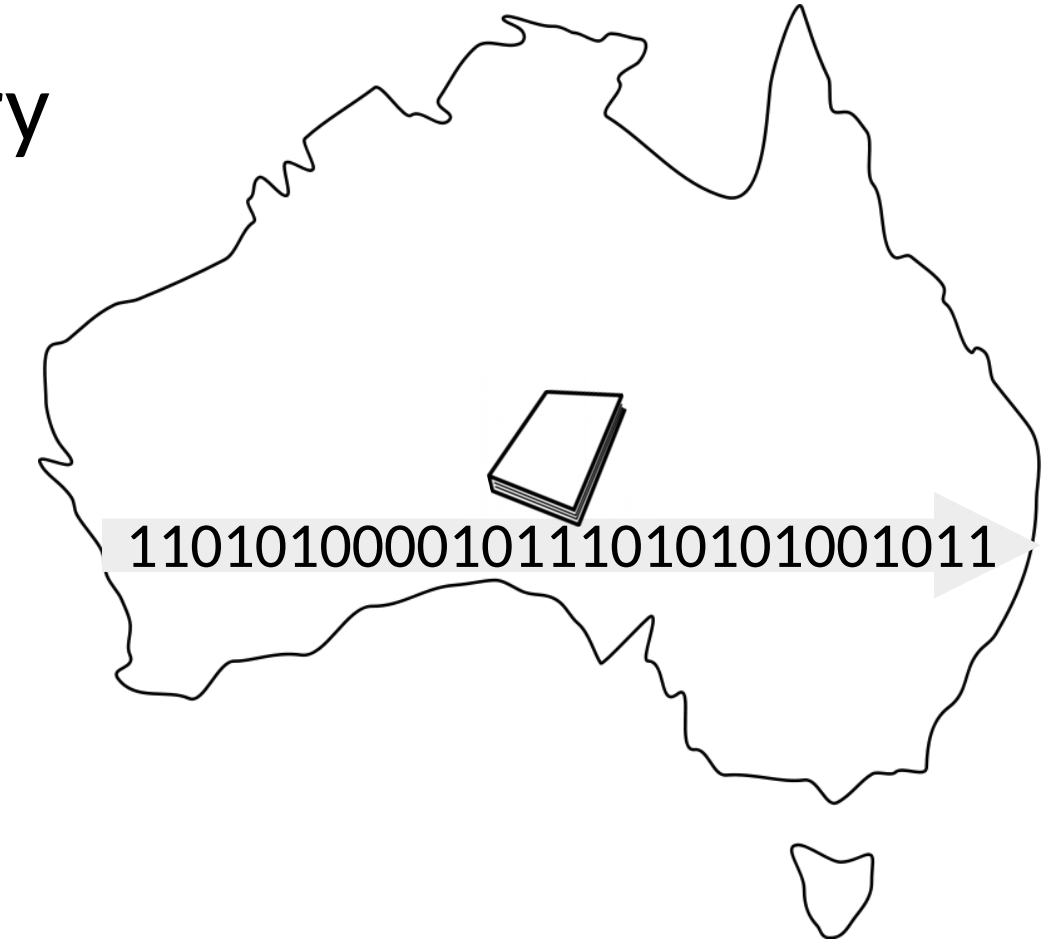
  BD = R x D

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

Bandwidth-Delay Example

- Fiber at home, cross-country

  R=40 Mbps, D=50 ms



Bandwidth-Delay Example (2)

- Fiber at home, cross-country

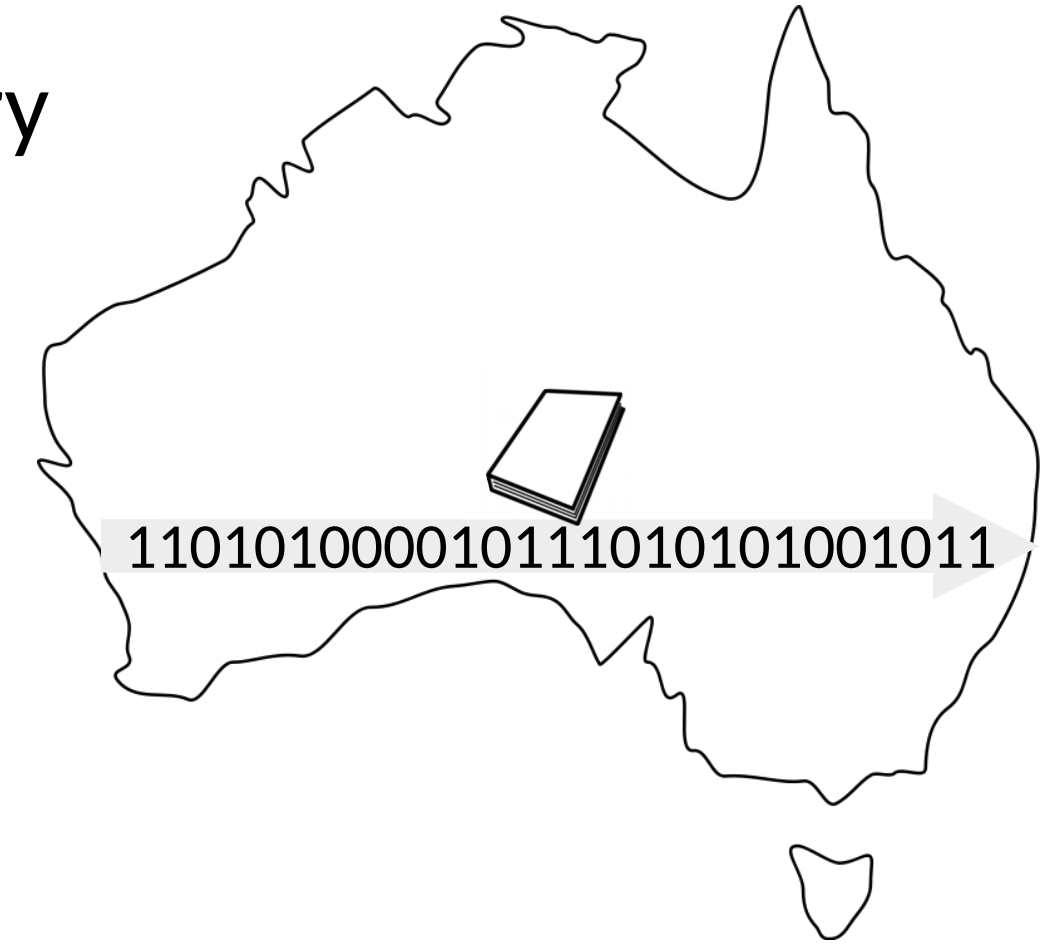
  R=40 Mbps, D=50 ms

  BD = 40 x 10^6 x 50 x 10^{-3} bits

  = 2000 Kbit

  = 250 KB

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



Media

[2]media

noun, often attributive

Definition of MEDIA

plural **medias**

1 : a medium of cultivation, conveyance, or expression • Air is a *media* that conveys sound.; *especially* : MEDIUM 2b

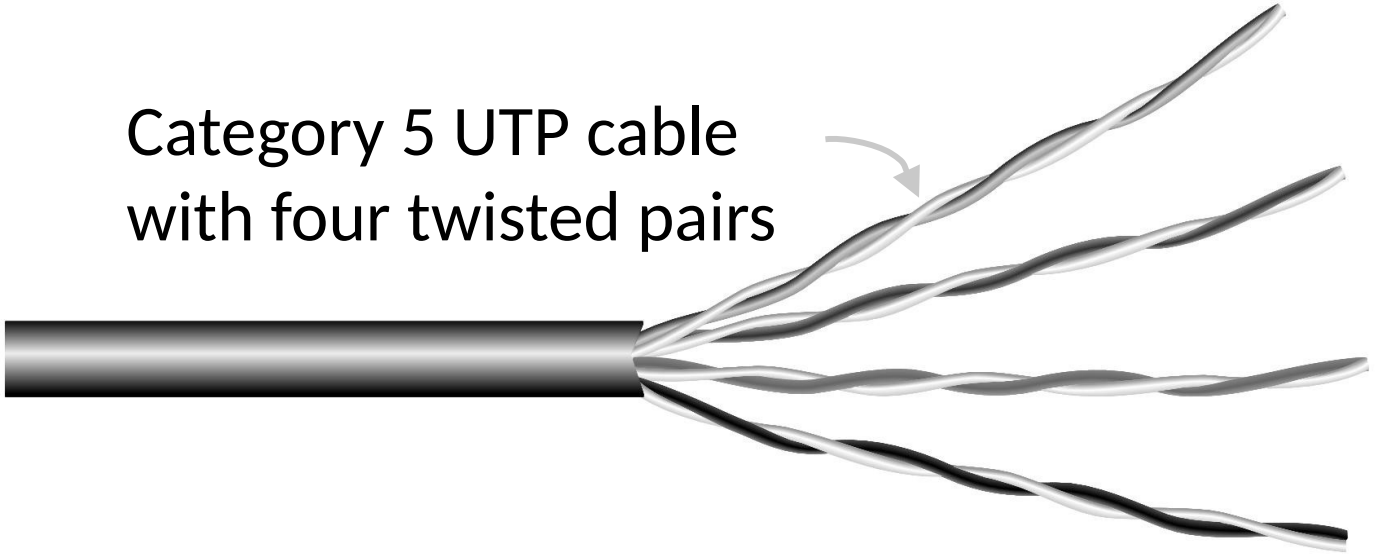
https://www.merriam-webster.com/dictionary/media

Types of Media

- Media propagate signals that carry bits of information
- We'll look at some common types:
  - Wires
  - Fiber (fiber optic cables)
  - Wireless

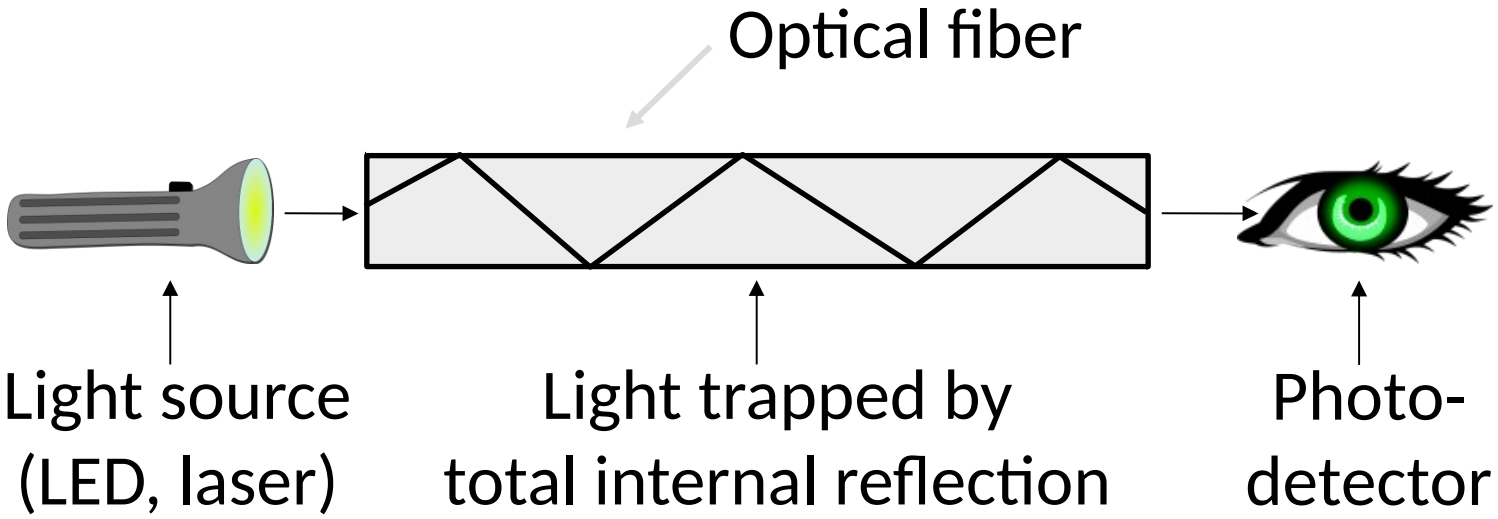
Wires – Twisted Pair

- Very common; used in LANs and telephone lines
  - Twists reduce radiated signal



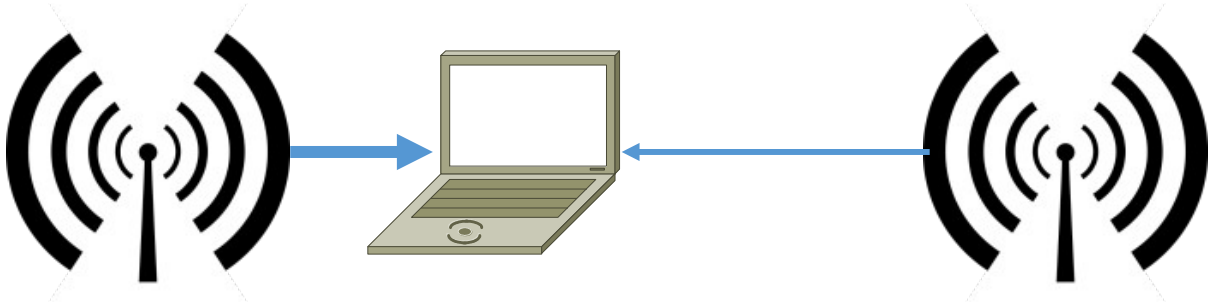
Fiber

- Long, thin, pure strands of glass
  - Enormous bandwidth (high speed) over long distances

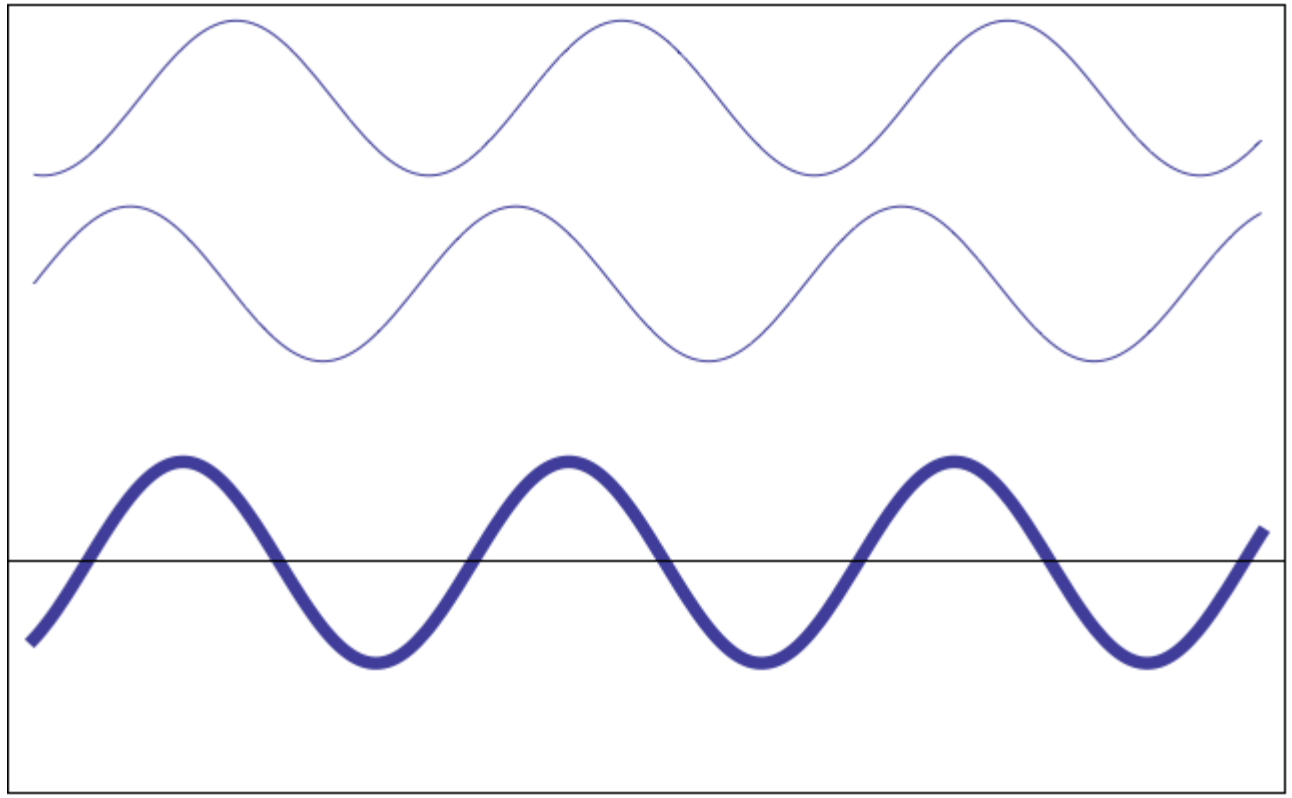


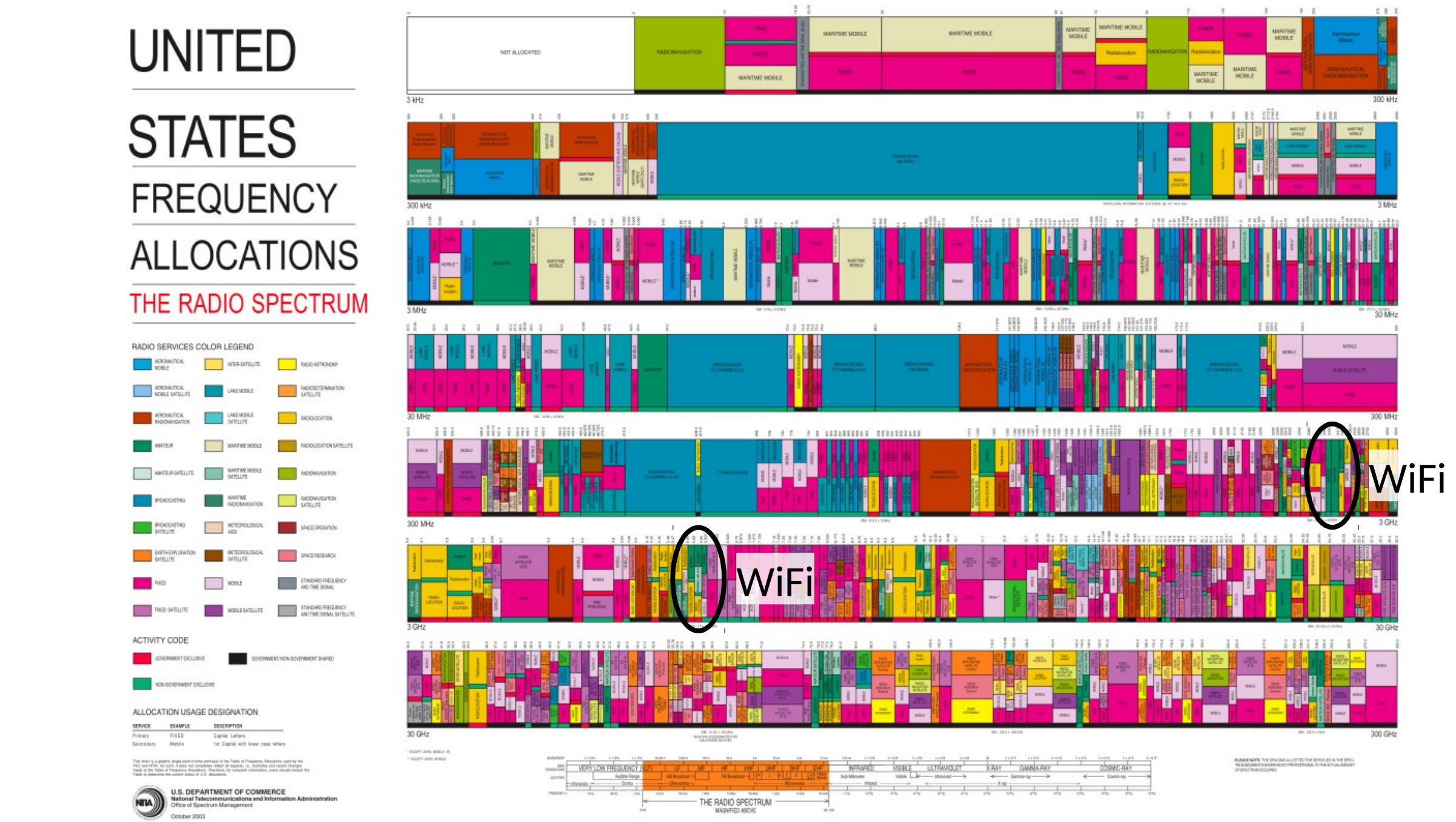
Wireless

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



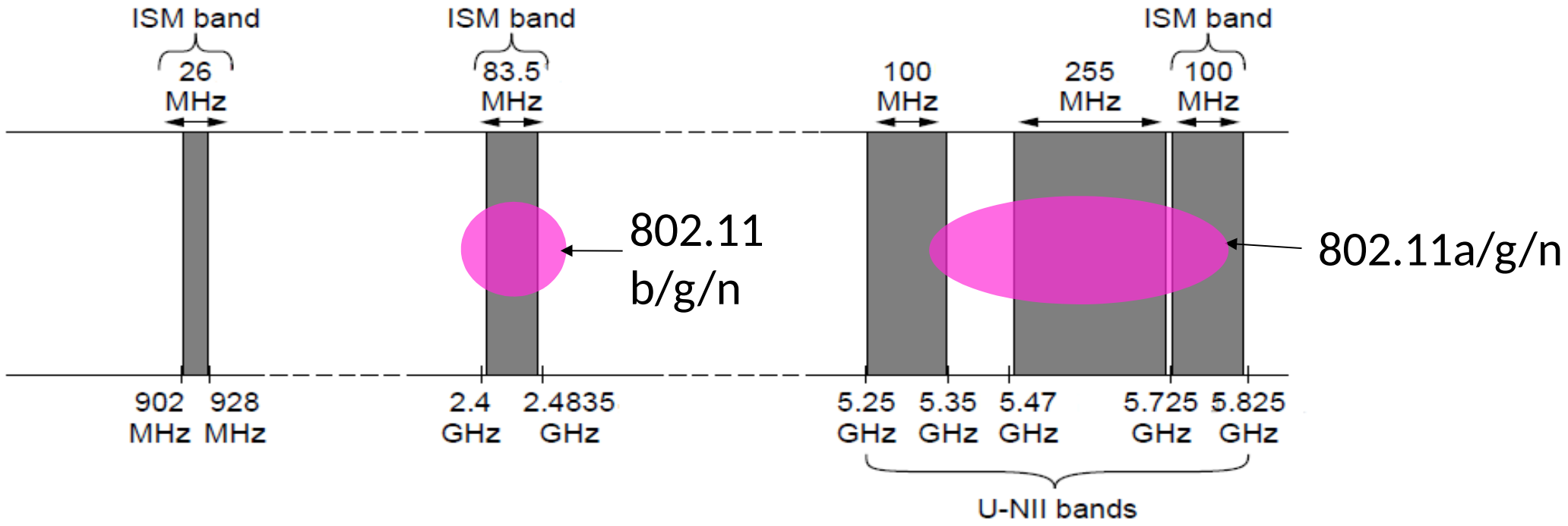
Wireless Interference




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
MAGNIFIED ABOVE

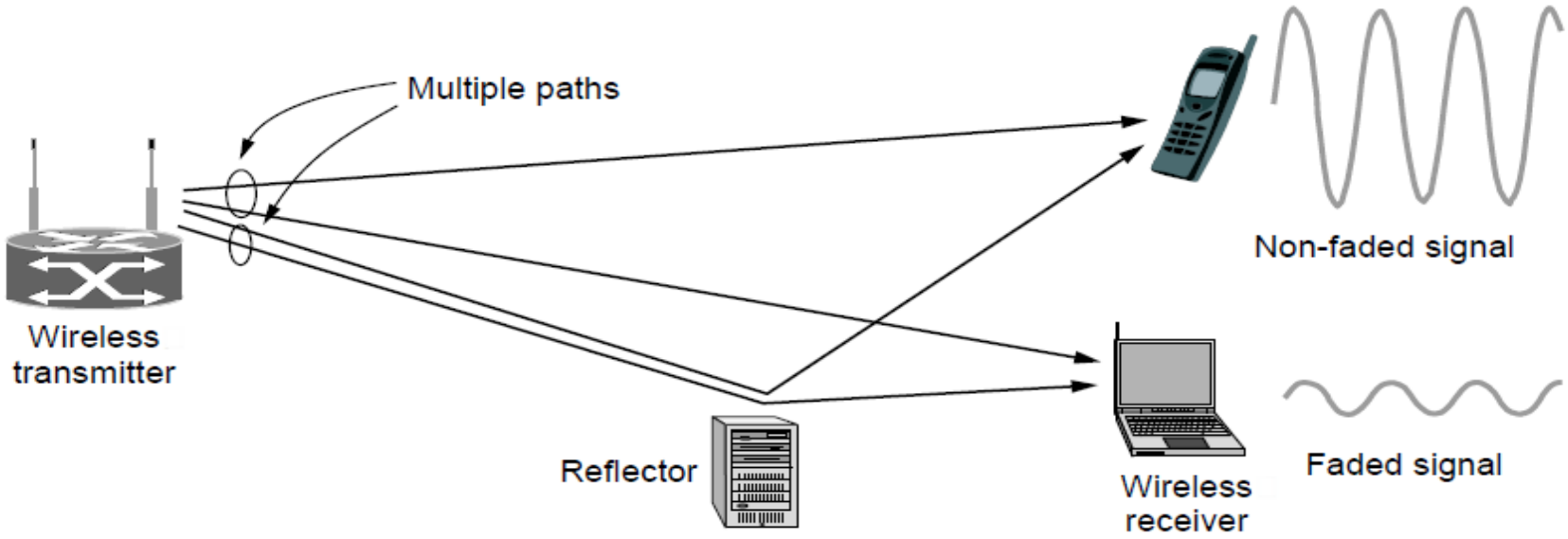
Wireless (2)

- Unlicensed (ISM) frequencies, e.g., WiFi, are widely used for computer networking



Multipath (3)

- Signals bounce off objects and take multiple paths
  - Some frequencies attenuated at receiver, varies with location



Wireless (4)

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

Limits

Topic

- How rapidly can we send information over a link?
  - Nyquist limit (~1924)
  - Shannon capacity (1948)

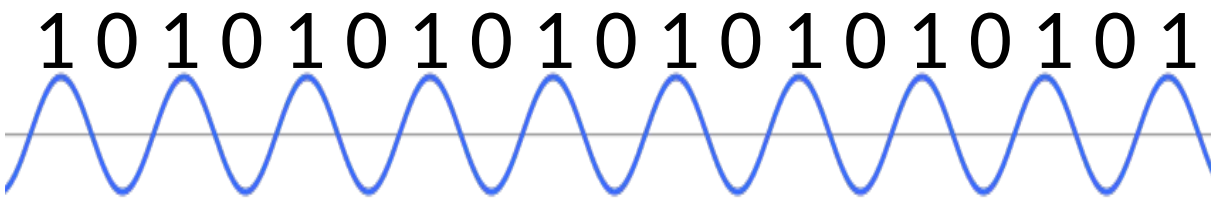
- Practical systems are devised to approach these limits

Key Channel Properties

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

Nyquist Limit

- The maximum <u>symbol</u> rate is 2B



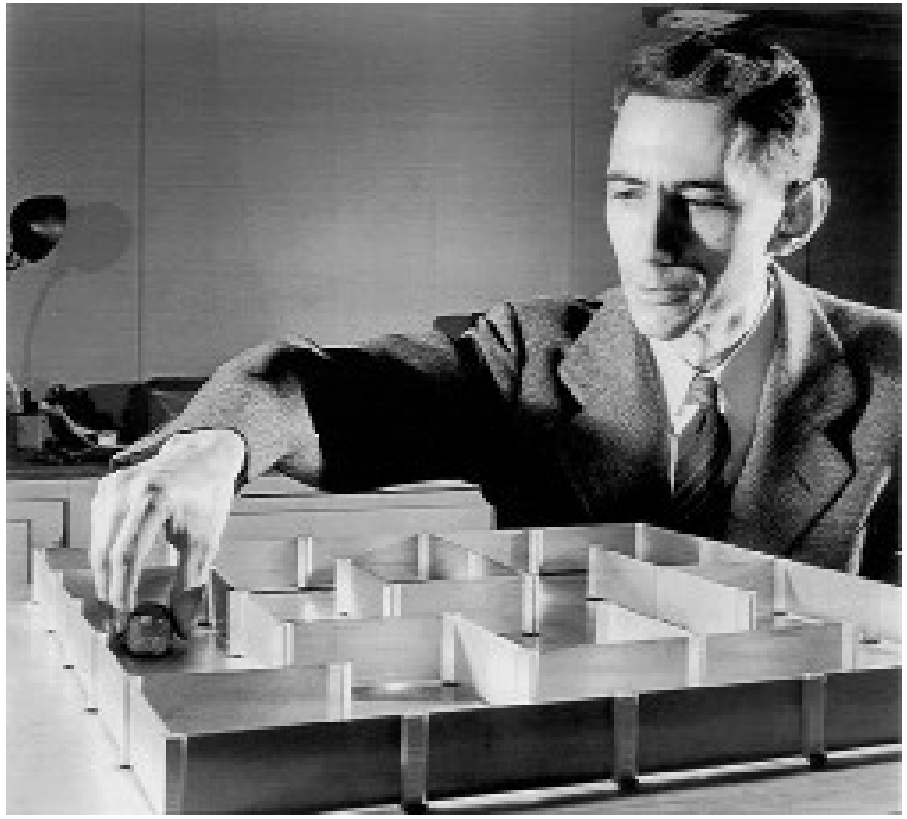
- Thus if there are V signal levels, ignoring noise, the maximum bit rate is:

$$R = 2B \log_2 V \text{ bits/sec}$$

Claude Shannon (1916-2001)

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

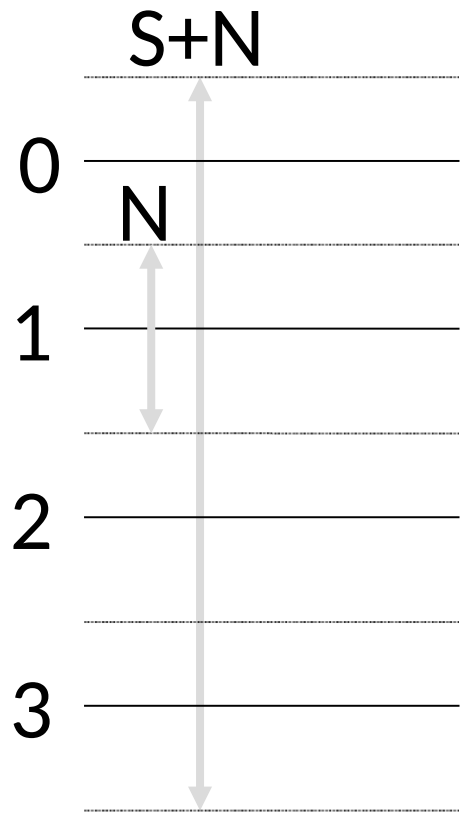
Electromechanical mouse that "solves" mazes!



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:

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

Shannon Capacity Takeaways

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

- There is some rate at which we can transmit data **without loss** over a random channel
- Assuming noise fixed, increasing the signal power yields diminishing returns : (
- Assuming signal is fixed, increasing bandwith increases capacity linearly!

Wired/Wireless Perspective (2)

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

  →Can fix data rate

Engineer SNR for data rate

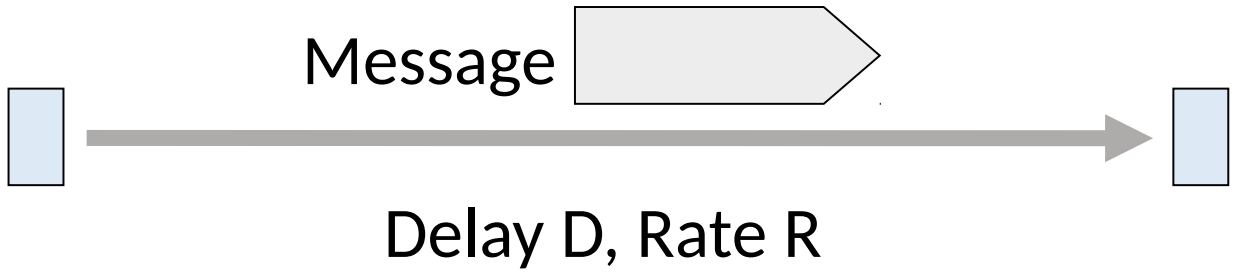
- Wireless
  - Given B, but SNR varies greatly, e.g., up to 60 dB!

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

Adapt data rate to SNR

All distilled to a simple link model

- Rate (or bandwidth, capacity, speed) in bits/second
- Delay in seconds, related to length

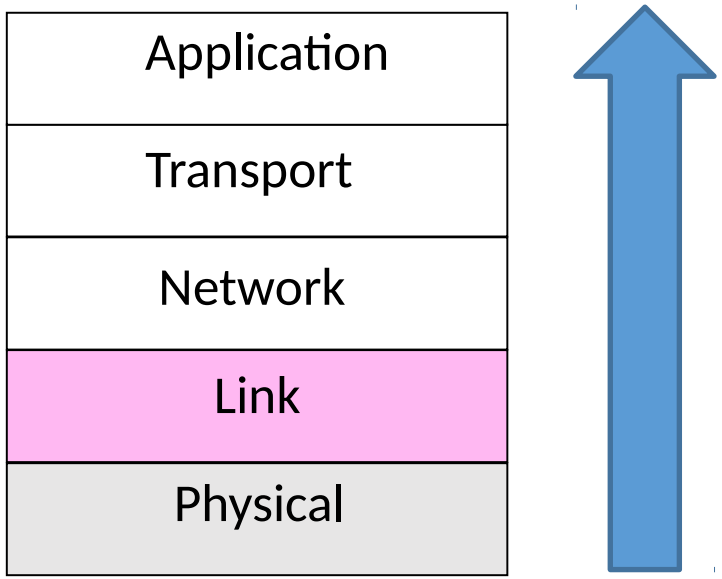


- Other important properties:
  - Whether the channel is broadcast, and its error rate

Link Layer

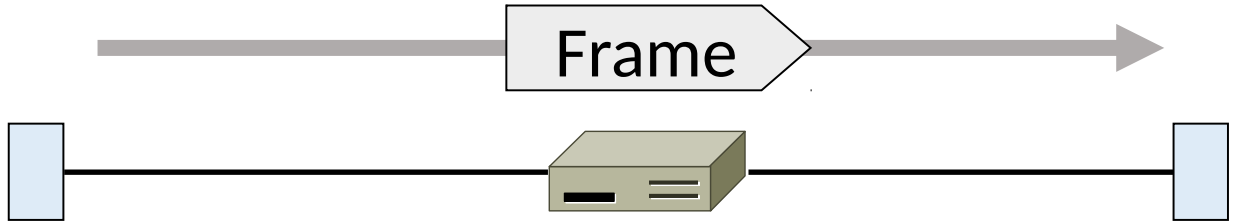
Where we are in the Course

- Moving on up to the Link Layer!

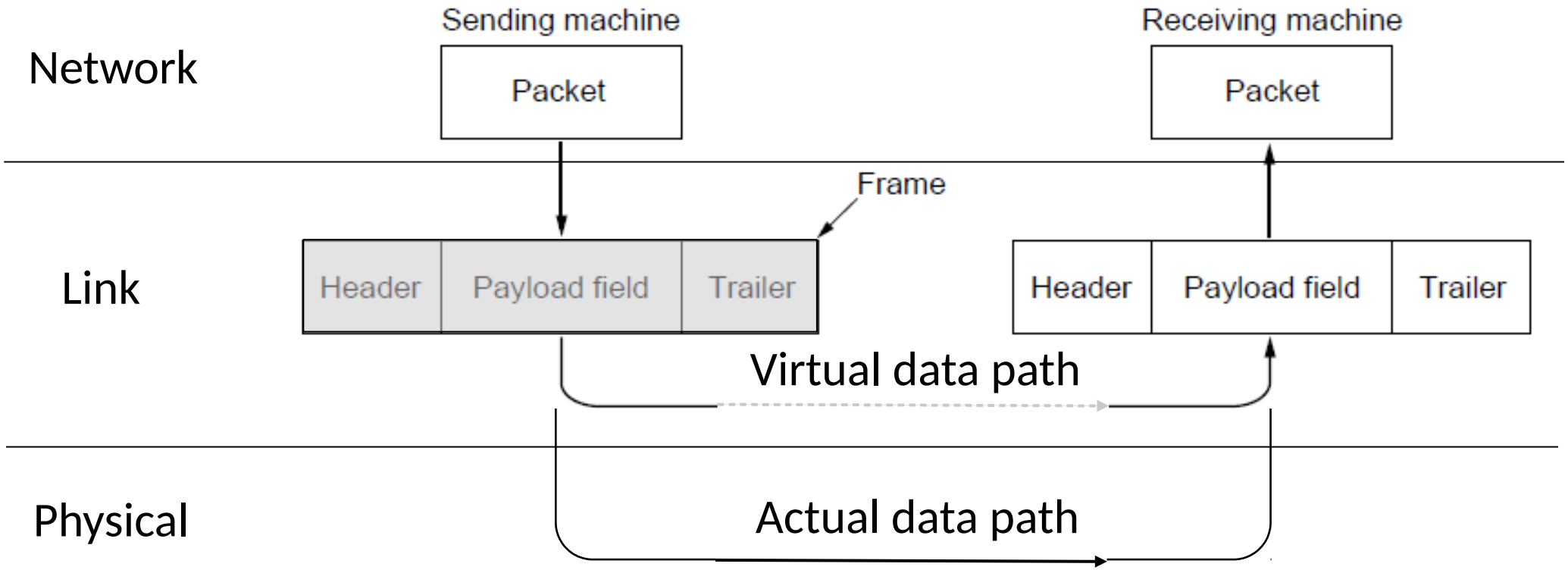


Scope of the Link Layer

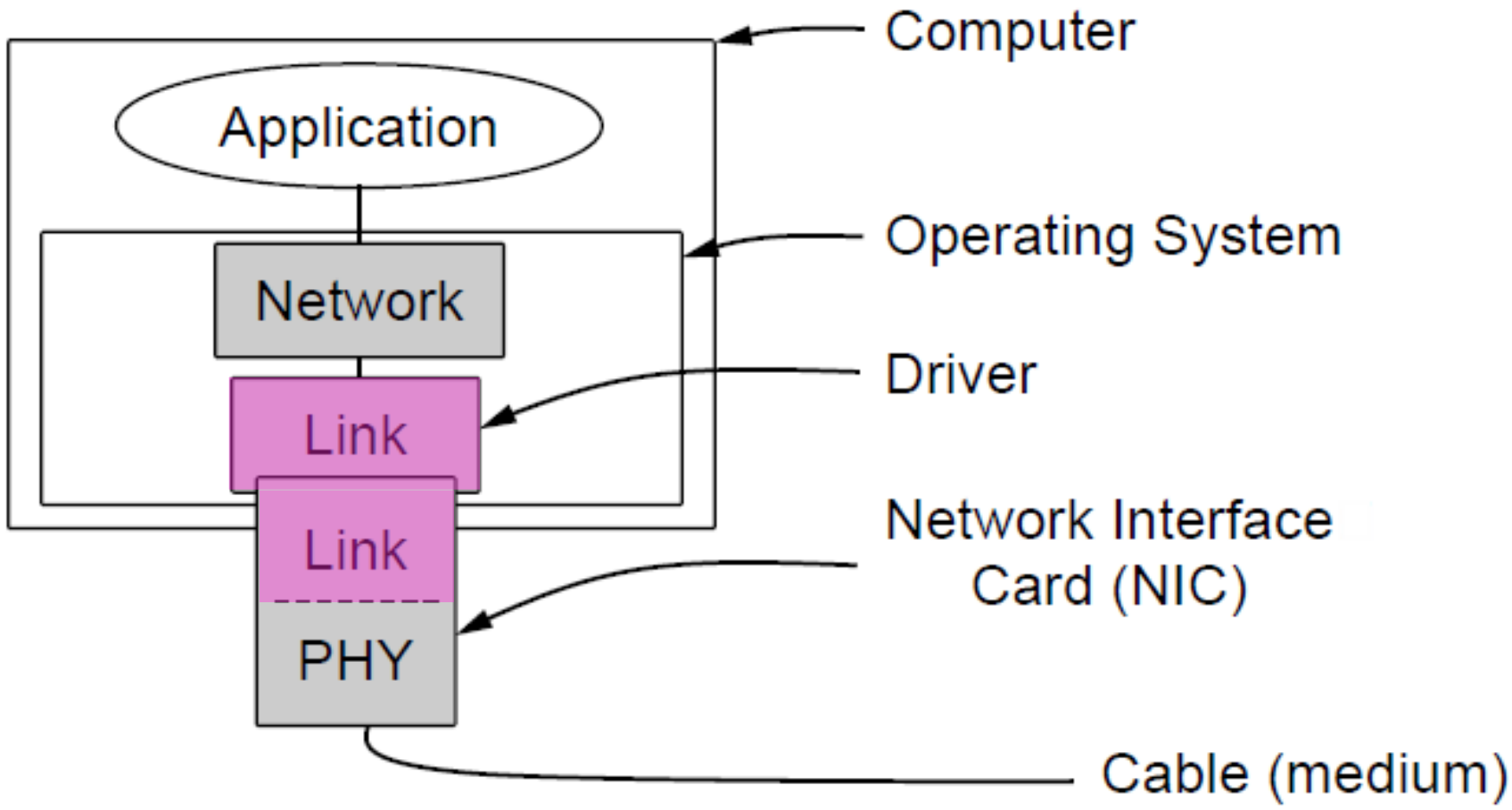
- Concerns how to transfer messages over one or more connected links
  - Messages are frames, of limited size
  - Builds on the physical layer



In terms of layers



Typical Implementation of Layers (2)



Topics

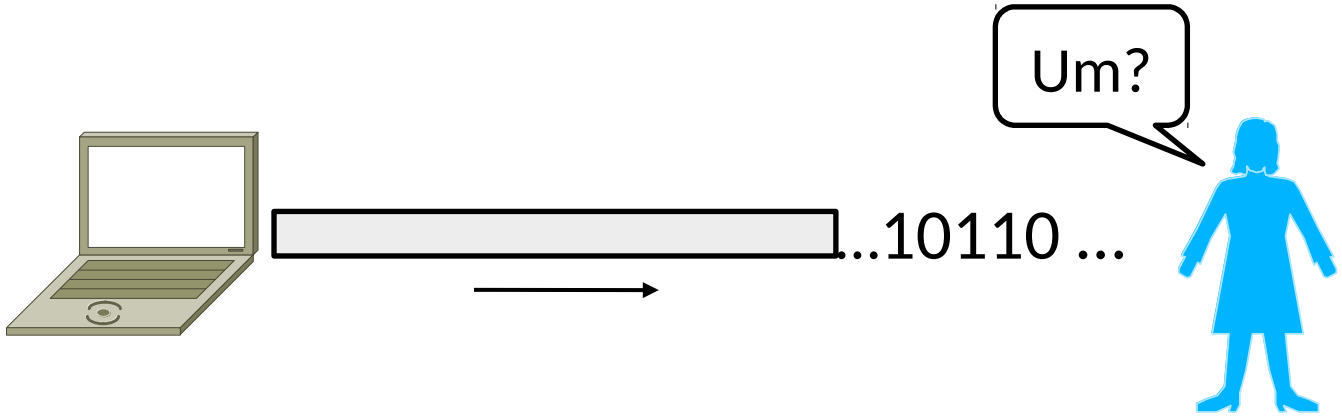
1. Framing
   - Delimiting start/end of frames
2. Error detection and correction
   - Handling errors
3. Retransmissions
   - Handling loss
4. Multiple Access
   - 802.11, classic Ethernet
5. Switching
   - Modern Ethernet

Framing

Delimiting start/end of frames

Topic

- The Physical layer gives us a stream of bits. How do we interpret it as a sequence of frames?



Framing Methods

- We'll look at:
  - Byte count (motivation)
  - Byte stuffing

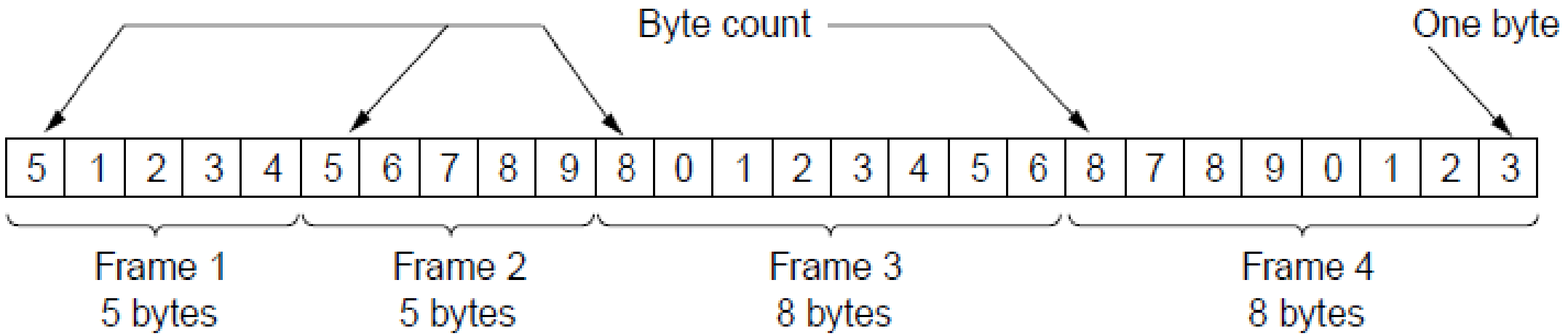
- In practice, the physical layer often helps to identify frame boundaries
  - E.g., Ethernet, 802.11

Simple ideas?

Byte Count

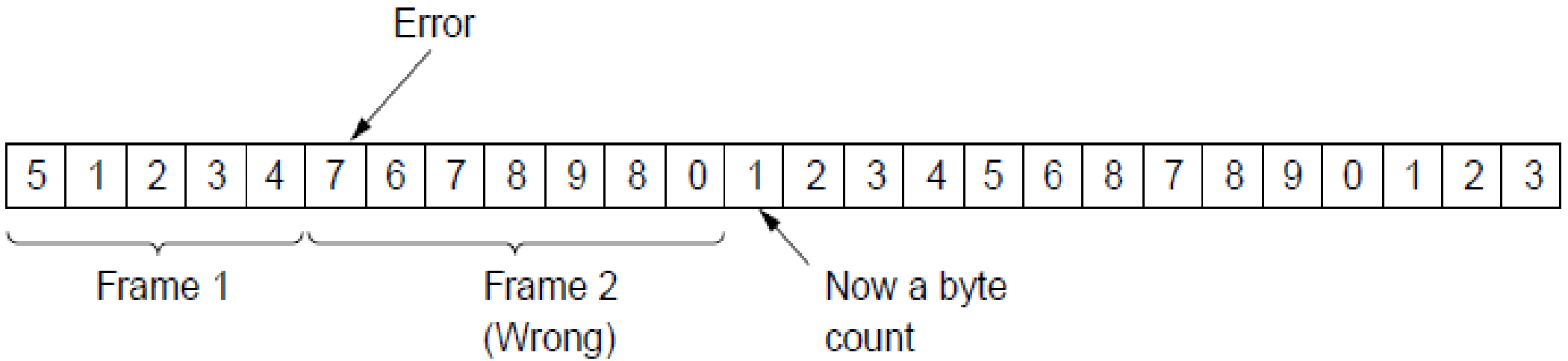
- First try:
  - Let's start each frame with a length field!
  - It's simple, and hopefully good enough …

Byte Count (2)


Byte count
One byte
5 1 2 3 4 5 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Frame 1
5 bytes
Frame 2
5 bytes
Frame 3
8 bytes
Frame 4
8 bytes

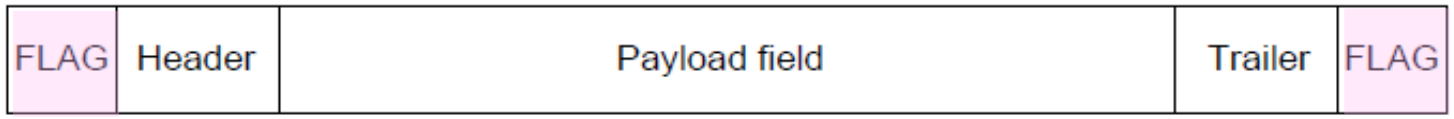
Byte Count (3)

- Difficult to re-synchronize after framing error
  - Want a way to scan for a start of frame



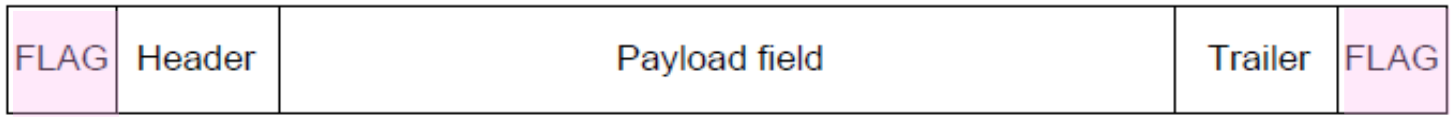
Byte Stuffing

- Better idea:
  - Have a special flag byte value for start/end of frame
  - Replace (“stuff”) the flag with an escape code
  - Problem?



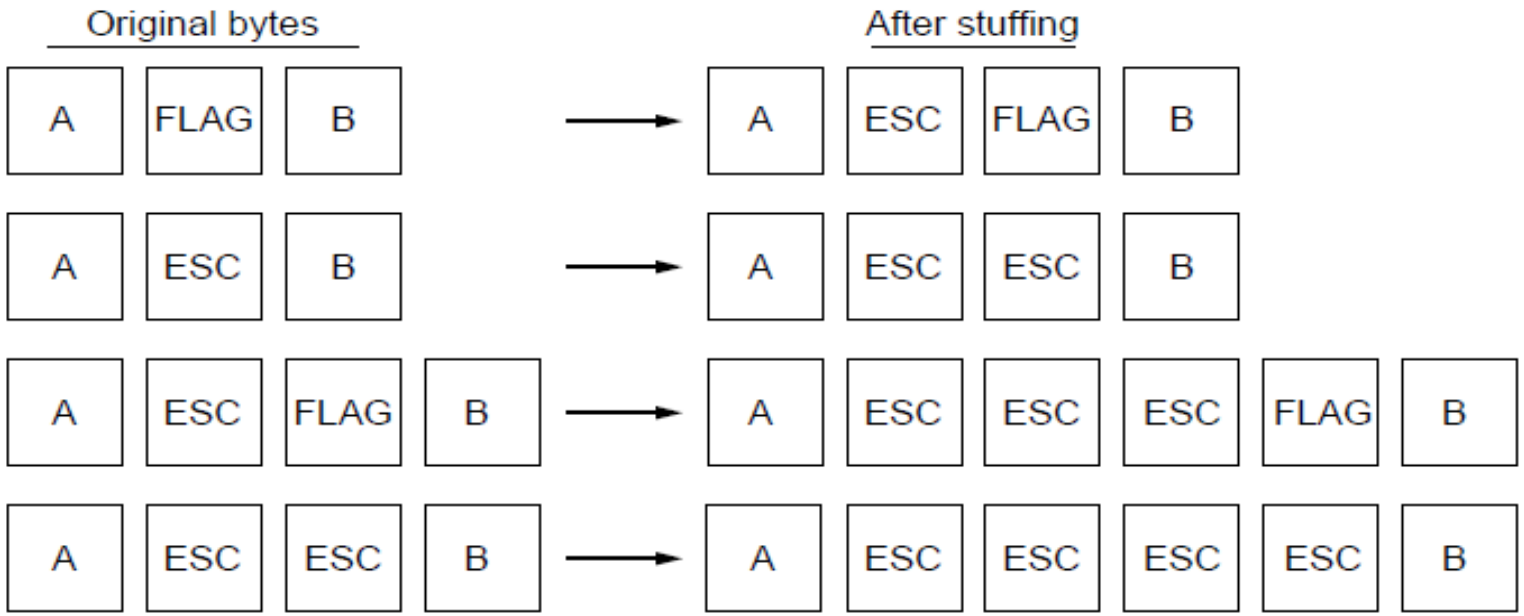
Byte Stuffing

- Better idea:
  - Have a special flag byte value for start/end of frame
  - Replace (“stuff”) the flag with an escape code
  - Complication: have to escape the escape code too!



Byte Stuffing (2)

- Rules:
  - Replace each FLAG in data with ESC FLAG
  - Replace each ESC in data with ESC ESC



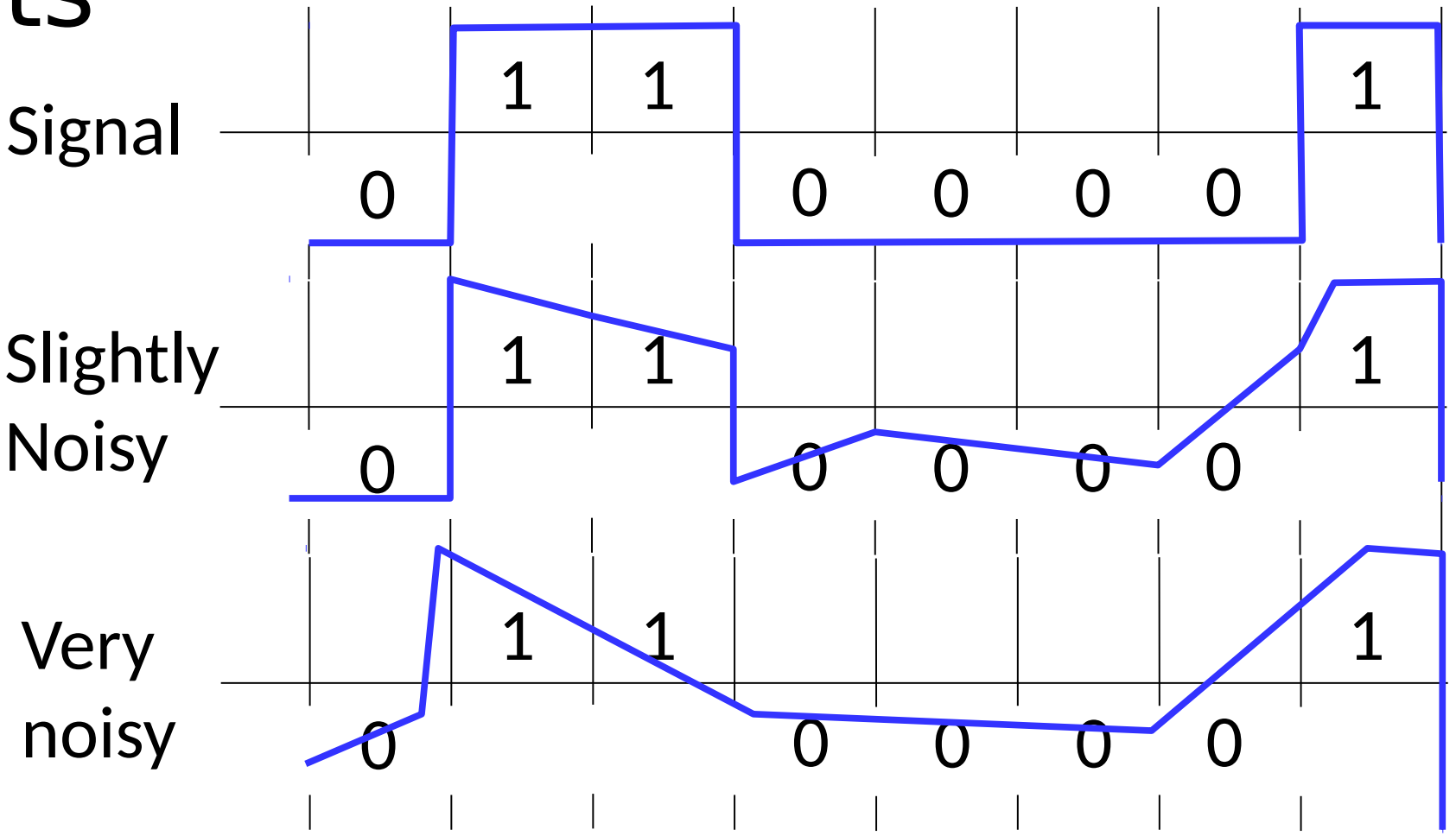
Link Layer: Error detection and correction

Topic

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

- Reliability is a concern that cuts across the layers

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

- Simple Ideas?

Motivating Example

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

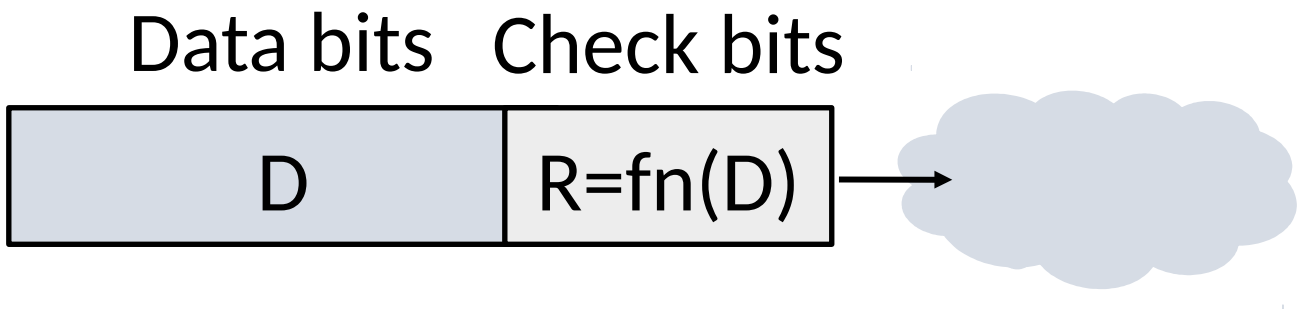
- How good is this code?
  - How many errors can it detect/correct?
  - How many errors will make it fail?

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

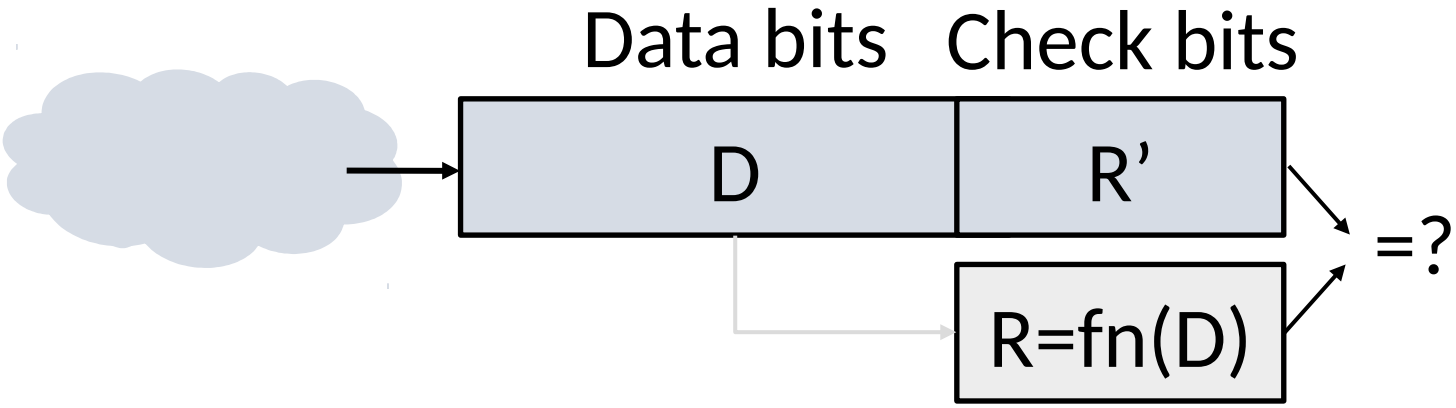
- Codeword consists of D data plus R check bits (=systematic block code)



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

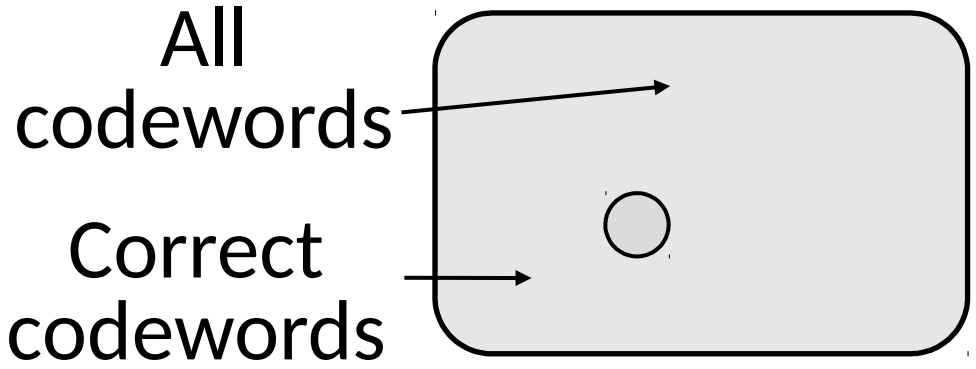
Using Error Codes (2)

- 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:

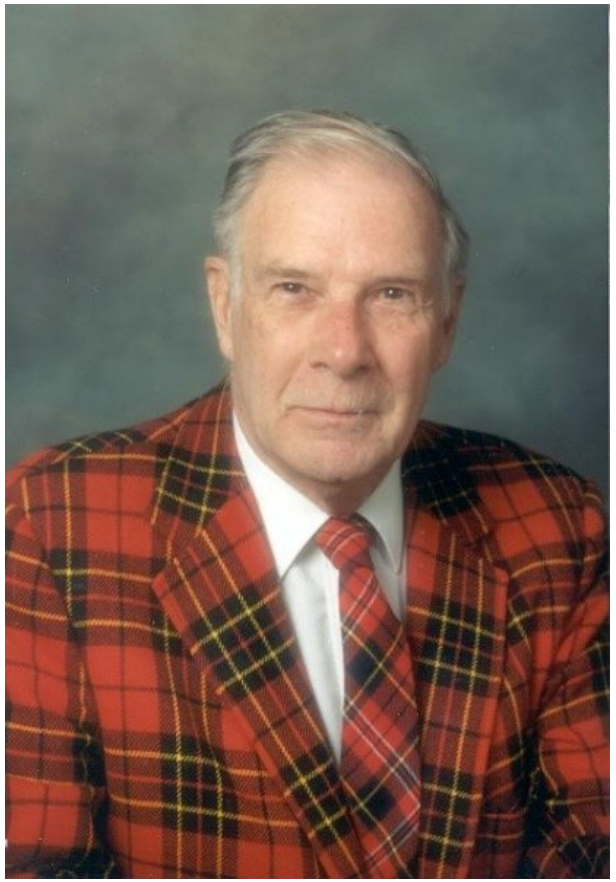


- Randomly chosen codeword is unlikely to be correct; overhead is low

R.W. Hamming (1915-1998)

- Much early work on codes:
  - "Error Detecting and Error Correcting Codes", BSTJ, 1950

- *"If the computer can tell when an error has occurred, surely there is a way of telling where the error is so the computer can correct the error itself"* - Hamming



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 coding is the minimum error distance between any pair of codewords (bit-strings) that cannot be detected

Hamming Distance (2)

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

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

- 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
f204
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
  f204
  f4f5
  f6f7
+(0000)
------
 2ddf1
   |
  ddf1
+    2
------
  ddf3
   |
  220c
```

Internet Checksum (4)

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
  f204
  f4f5
  f6f7
+ 220c
------
```

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
  f204
  f4f5
  f6f7
+ 220c
------
 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?

Why Error Correction is Hard

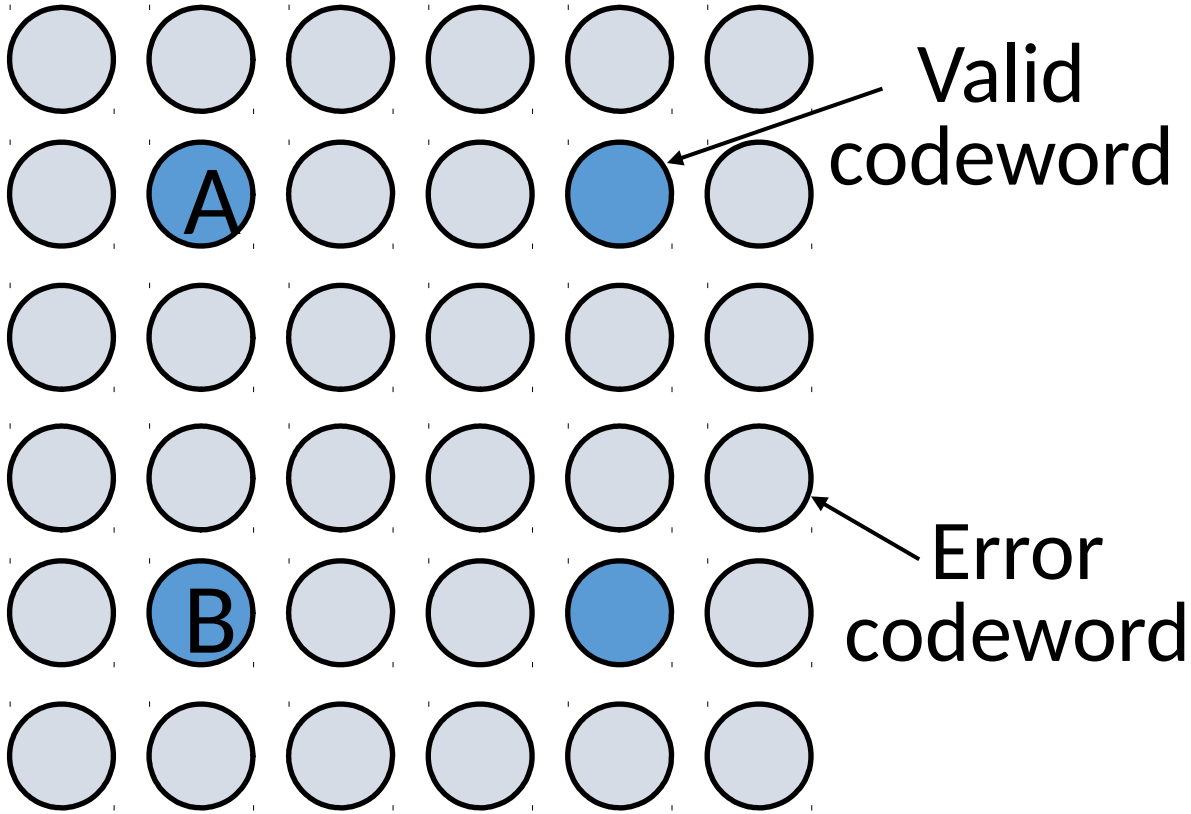
- If we had reliable check bits we could use them to narrow down the position of the error
  - Then correction would be easy
- But error could be in the check bits as well as the data bits!
  - Data might even be correct

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

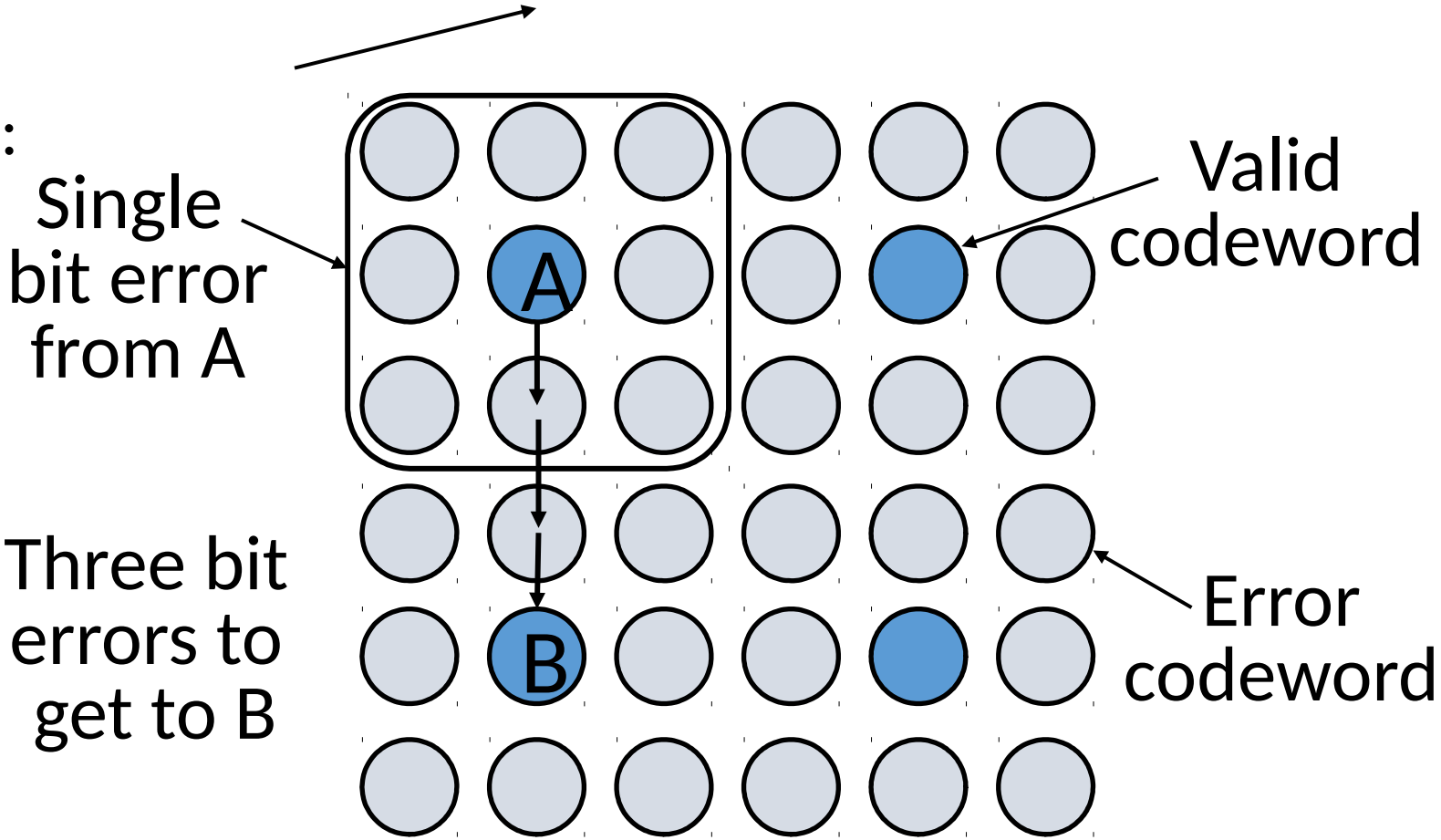
Intuition (2)

- Visualization of code:



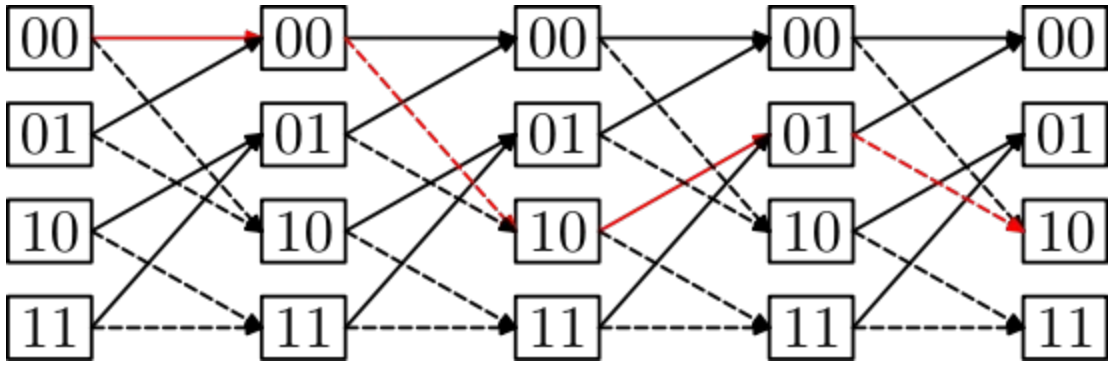
Intuition (3)

- Visualization of code:



Other Error Correction Codes

- Real codes are more involved than Hamming
- E.g., Convolutional codes (§3.2.3)
  - Take a stream of data and output a mix of the input bits
  - Makes each output bit less fragile
  - Decode using Viterbi algorithm (which can use bit confidence values)



More coding theory

- This is a **huge** field.
- See EE 505, 514, 515 for more info
- Key points:
  - Coding allows us to detect and correct bit errors received from the PHY
  - It is very complicated. Abstract away with Hamming Distance

Detection vs. Correction

- 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, 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.)

Link Layer: Retransmissions

So what do we do if a frame is corrupted?

- From sender?
- From receiver?

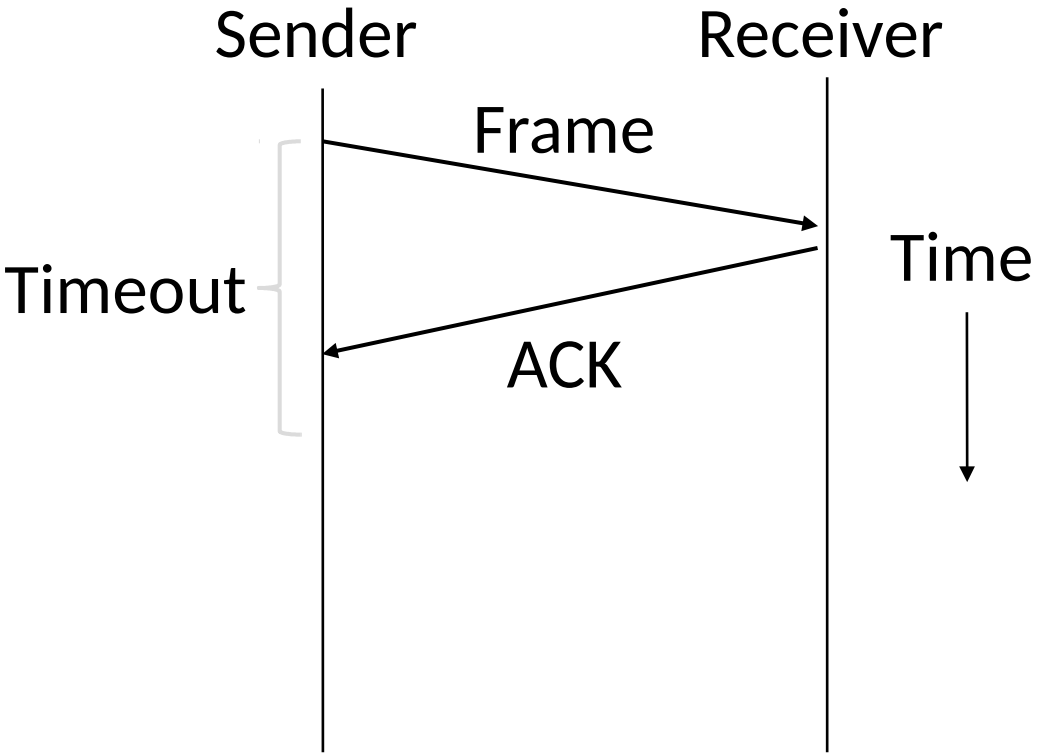
ARQ (Automatic Repeat reQuest)

- ARQ often used when errors are common or must be corrected
  - E.g., WiFi, and TCP (later)

- Rules at sender and receiver:
  - Receiver automatically acknowledges correct frames with an ACK
  - Sender automatically resends after a timeout, until an ACK is received

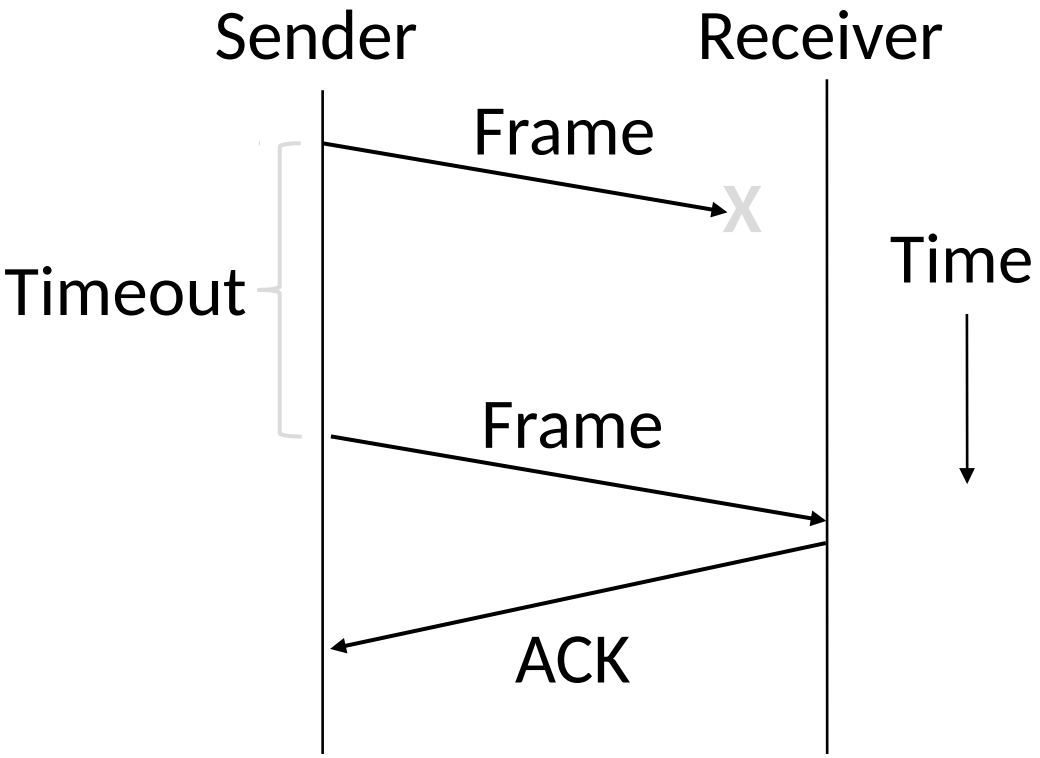
ARQ (2)

- Normal operation (no loss)



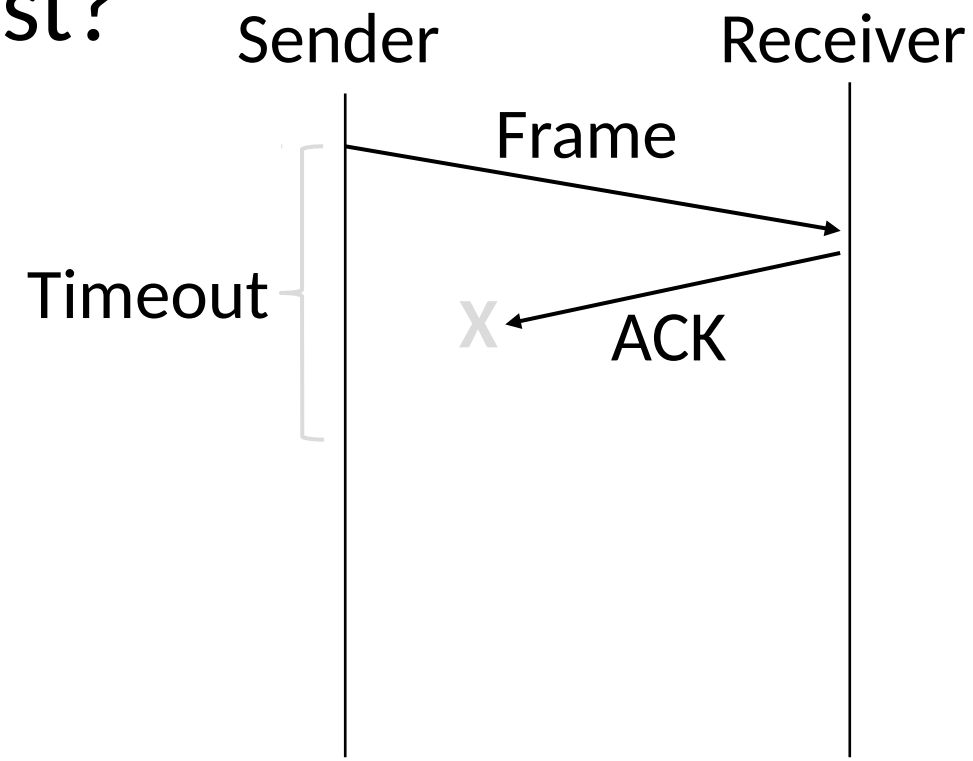
ARQ (3)

- Loss and retransmission



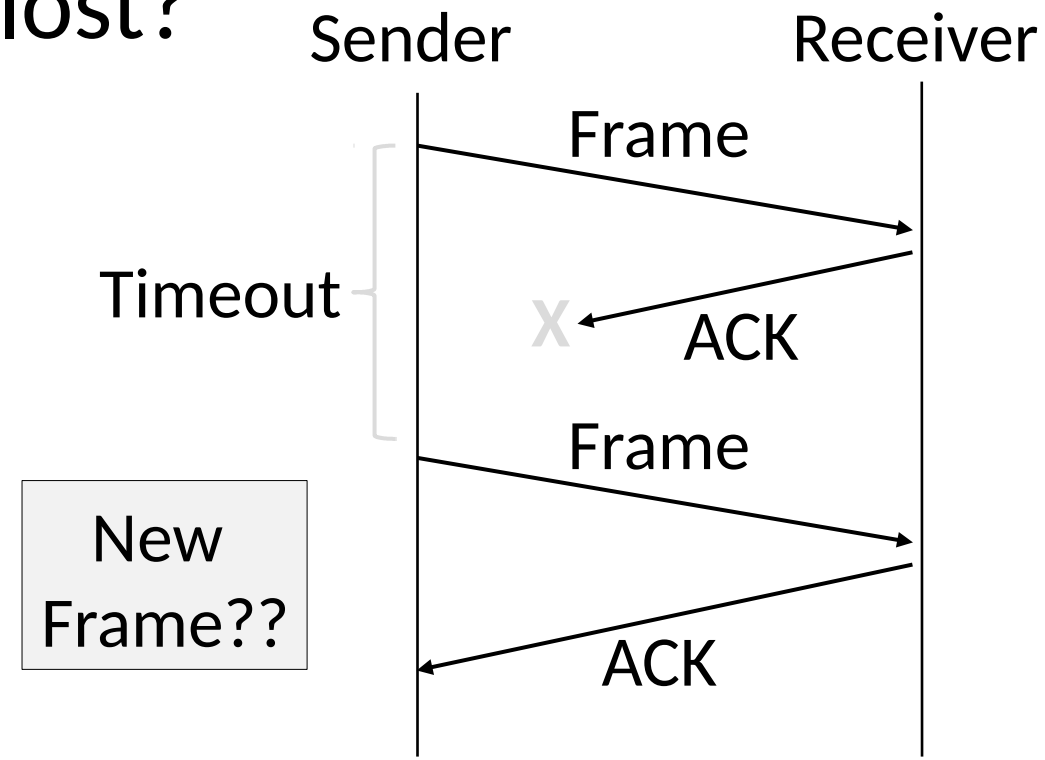
Duplicates

- What happens if an ACK is lost?



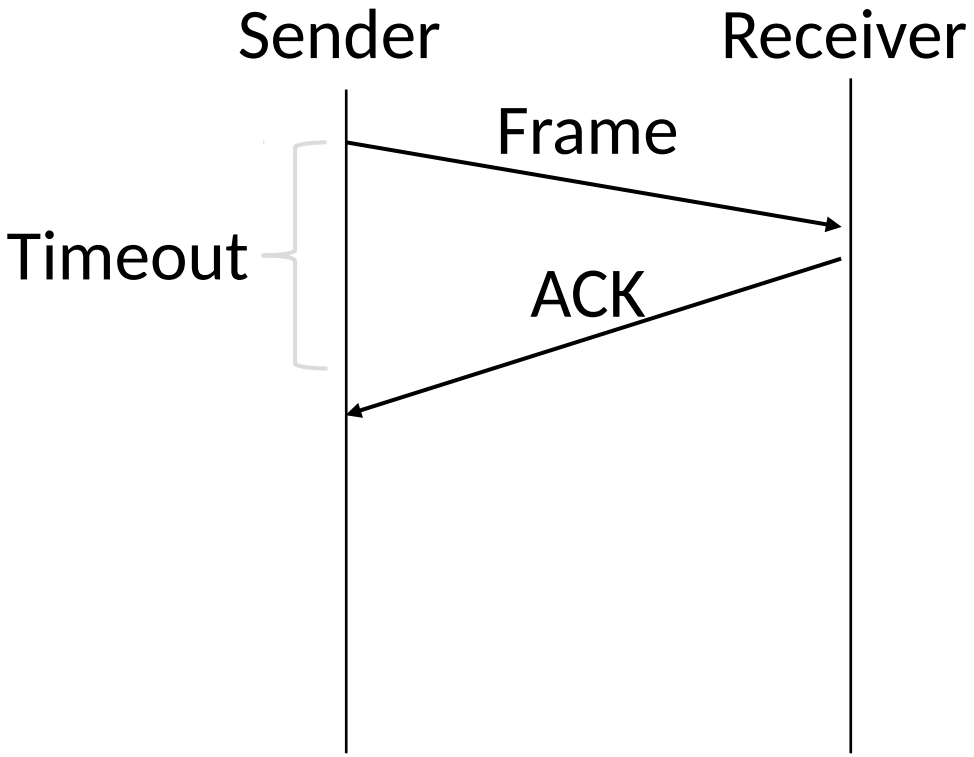
Duplicates (2)

- What happens if an ACK is lost?



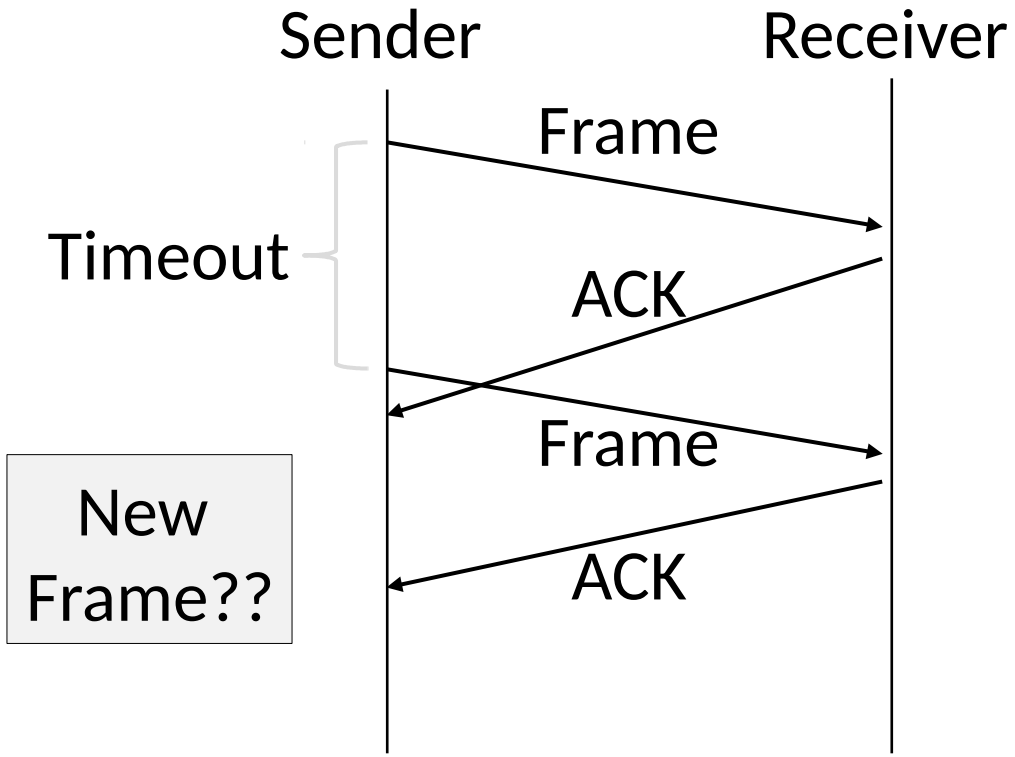
Duplicates (3)

- Or the timeout is early?



Duplicates (4)

- Or the timeout is early?



So What's Tricky About ARQ?

- Two non-trivial issues:
  - How long to set the timeout?
  - How to avoid accepting duplicate frames as new frames
- Want performance in the common case and correctness always
- Ideas?

Timeouts

- Timeout should be:
  - Not too big (link goes idle)
  - Not too small (spurious resend)
- Fairly easy on a LAN
  - Clear worst case, little variation
- Fairly difficult over the Internet
  - Much variation, no obvious bound
  - We'll revisit this with TCP (later)

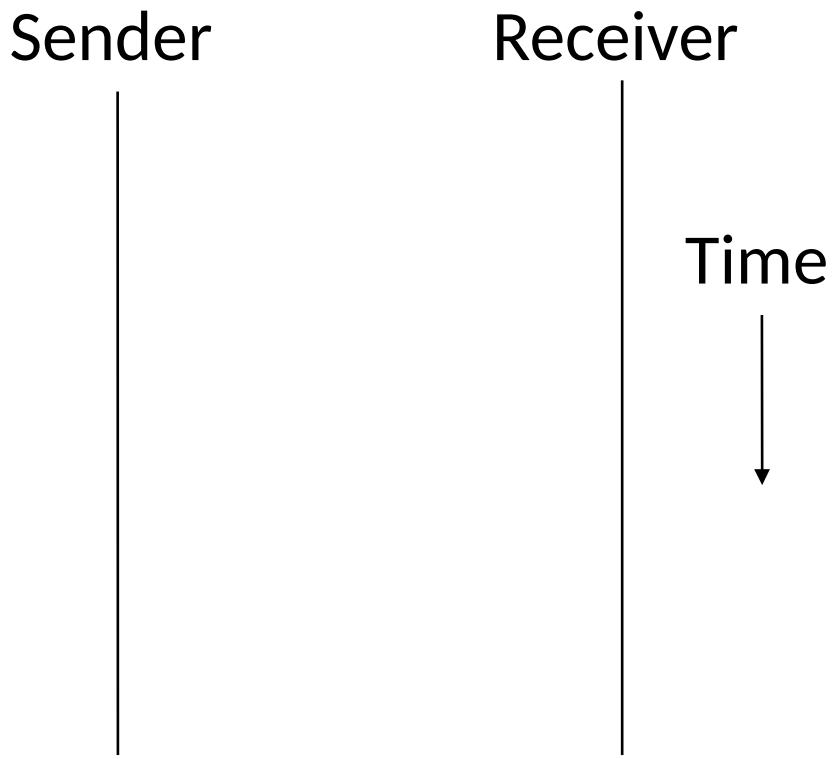
Sequence Numbers

- Frames and ACKs must both carry sequence numbers for correctness

- To distinguish the current frame from the next one, a single bit (two numbers) is sufficient
  - Called Stop-and-Wait

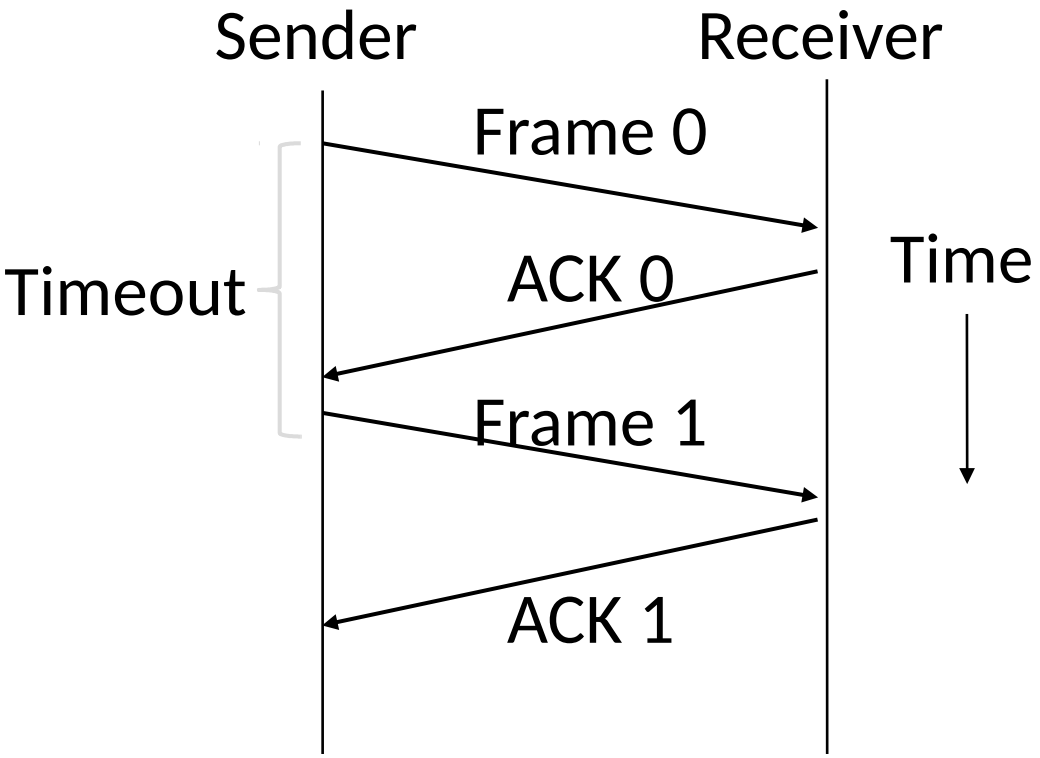
Stop-and-Wait

- In the normal case:



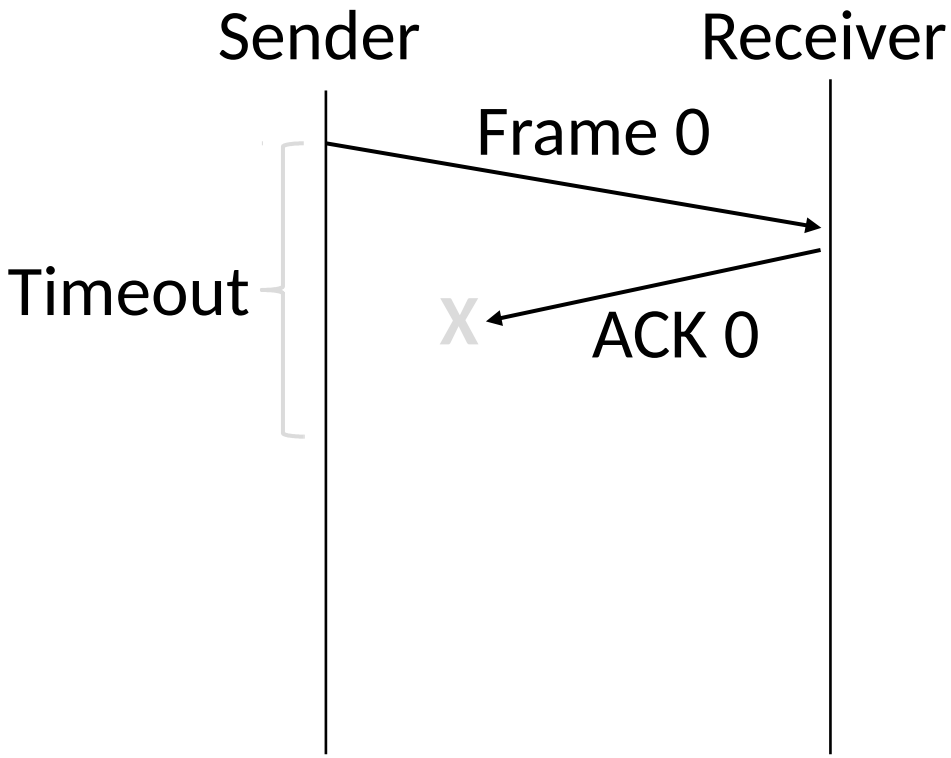
Stop-and-Wait (2)

- In the normal case:



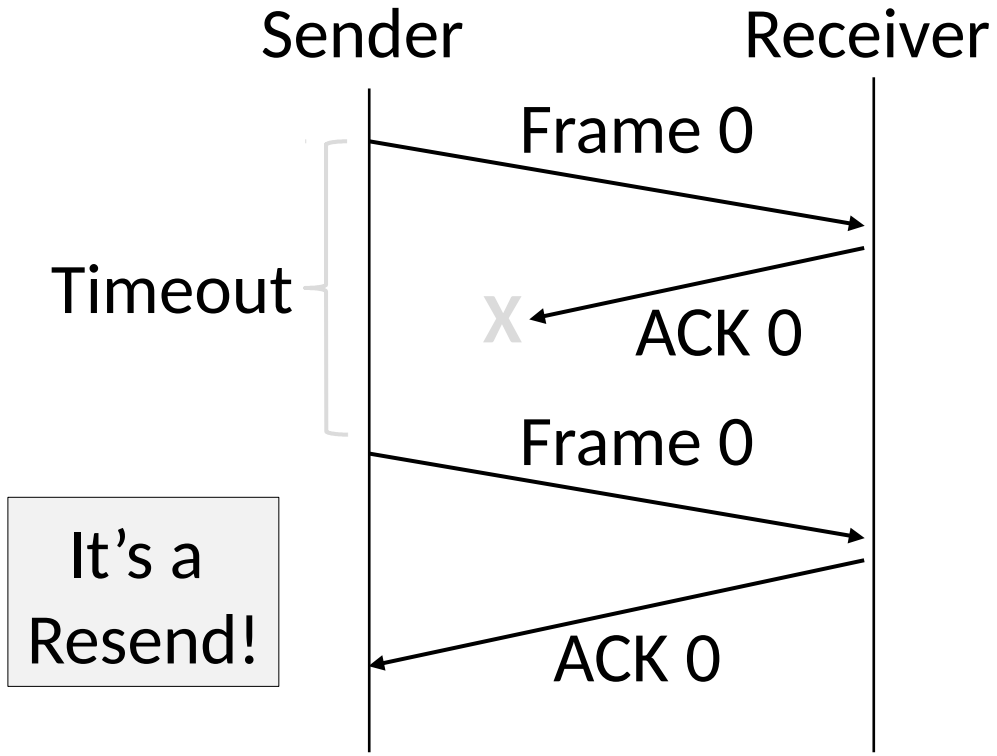
Stop-and-Wait (3)

- With ACK loss:



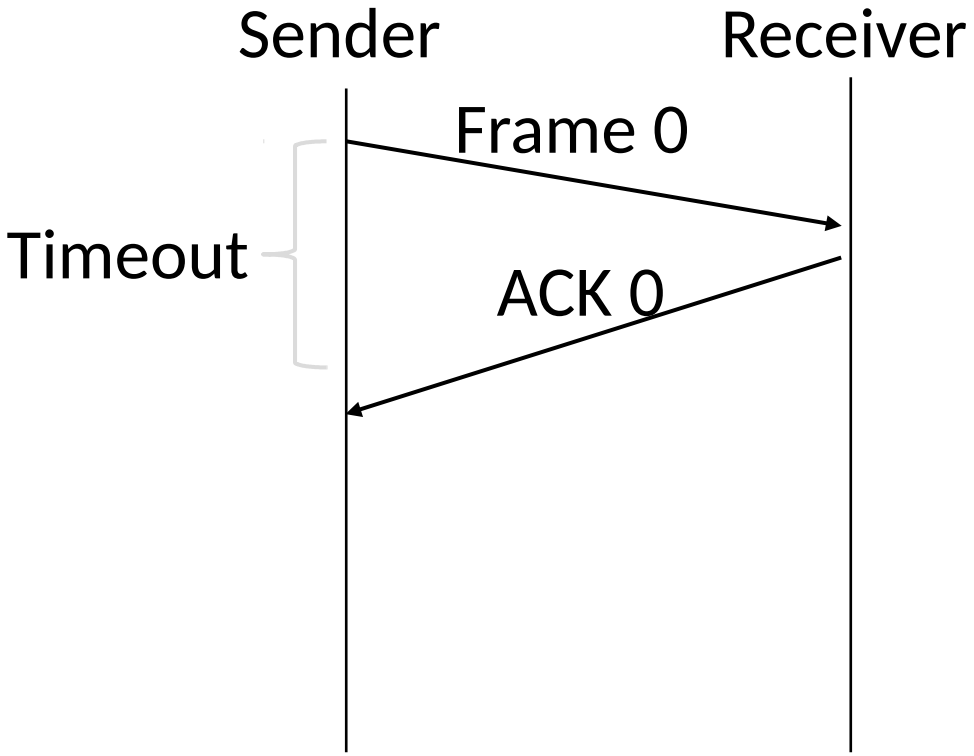
Stop-and-Wait (4)

- With ACK loss:



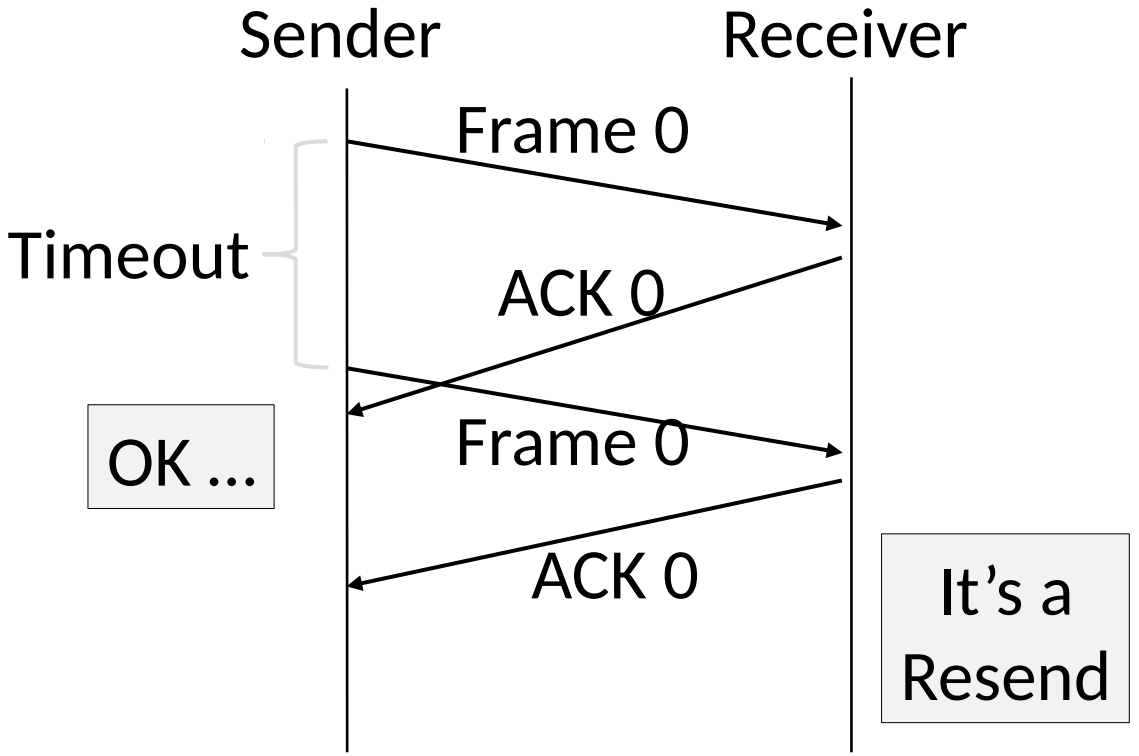
Stop-and-Wait (5)

- With early timeout:



Stop-and-Wait (6)

- With early timeout:



Multiple Access

Topic

- Multiplexing is the network word for the sharing of a resource
- What are some obvious ways to multiple a resource?

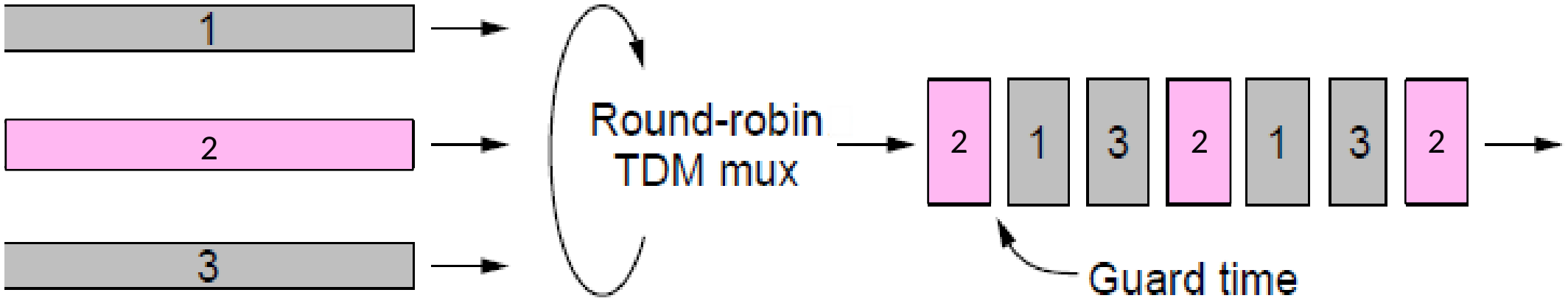
Topic

- Multiplexing is the network word for the sharing of a resource

- Classic scenario is sharing a link among different users
  - Time Division Multiplexing (TDM)
  - Frequency Division Multiplexing (FDM)

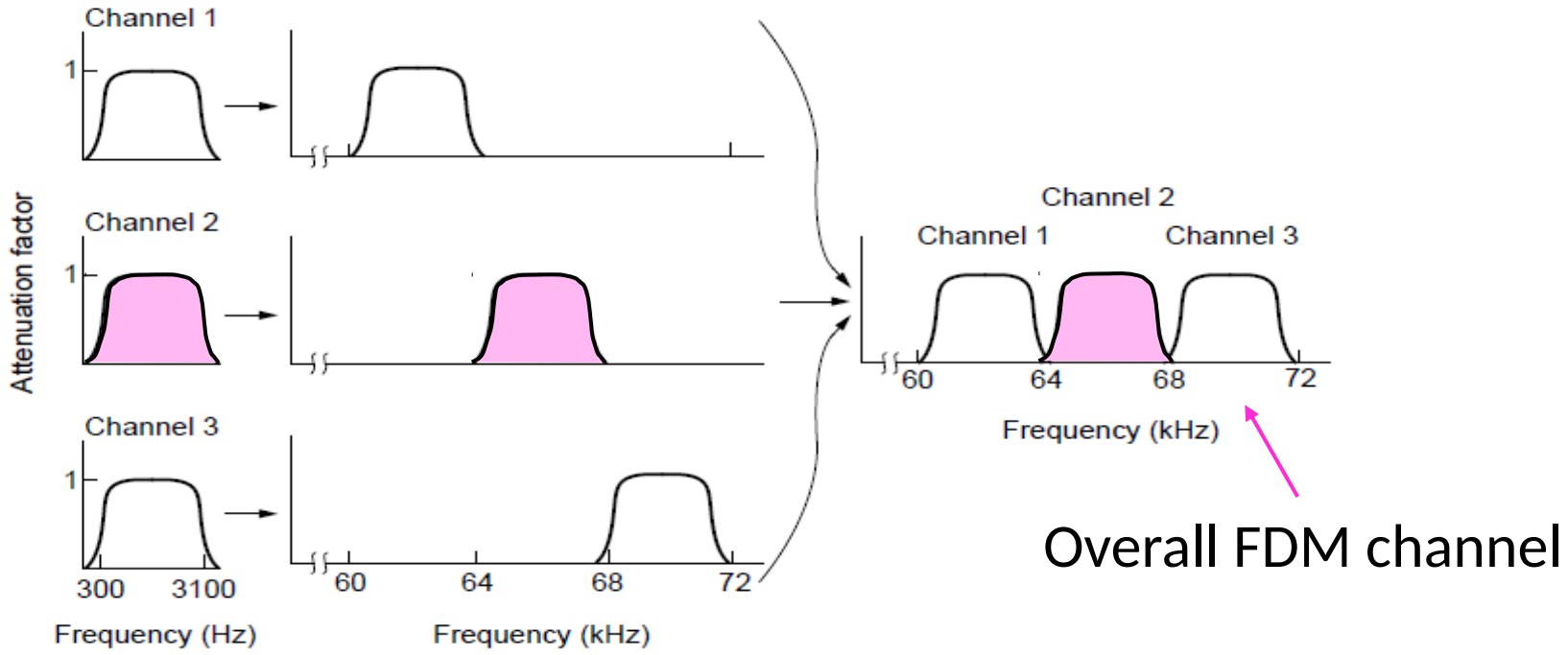
Time Division Multiplexing (TDM)

- Users take turns on a fixed schedule



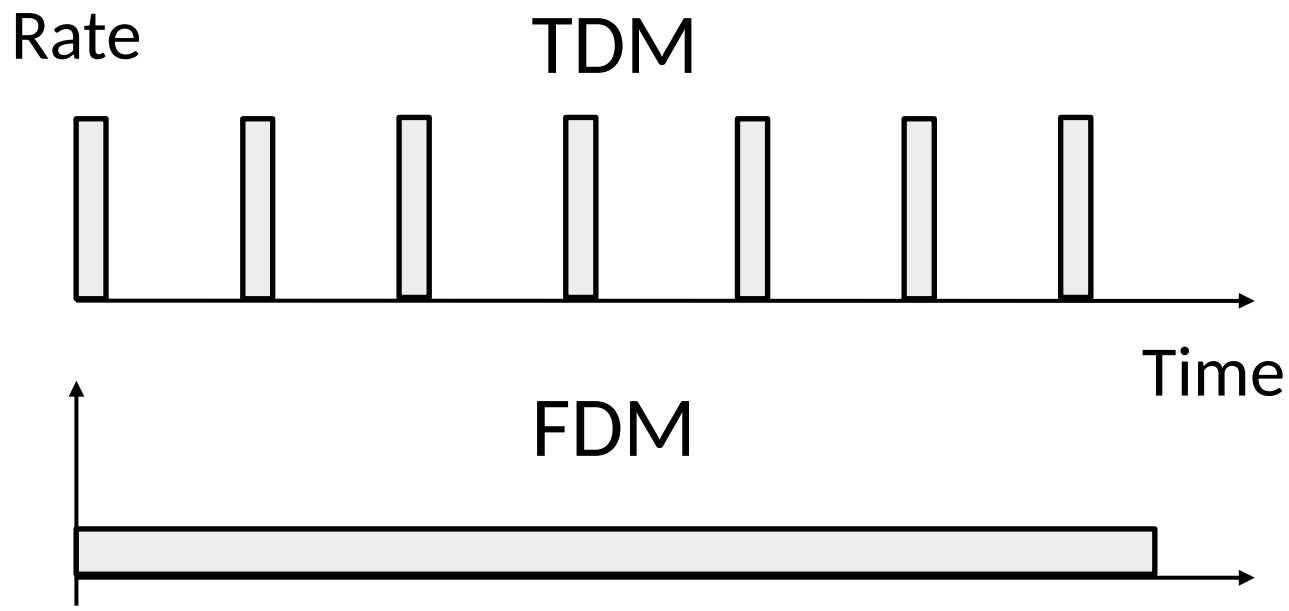
Frequency Division Multiplexing (FDM)

- Put different users on different frequency bands



TDM versus FDM (2)

- In TDM a user sends at a high rate a fraction of the time; in FDM, a user sends at a low rate all the time

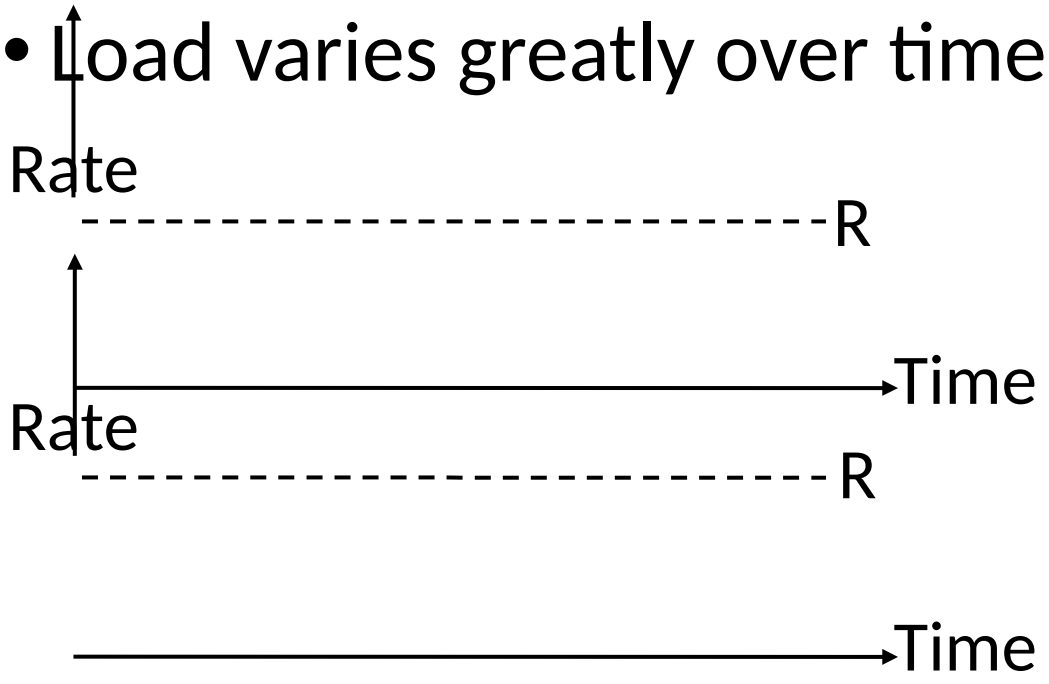


TDM/FDM Usage

- Statically divide a resource
  - Suited for continuous traffic, fixed number of users
- Widely used in telecommunications
  - TV and radio stations (FDM)
  - GSM (2G cellular) allocates calls using TDM within FDM

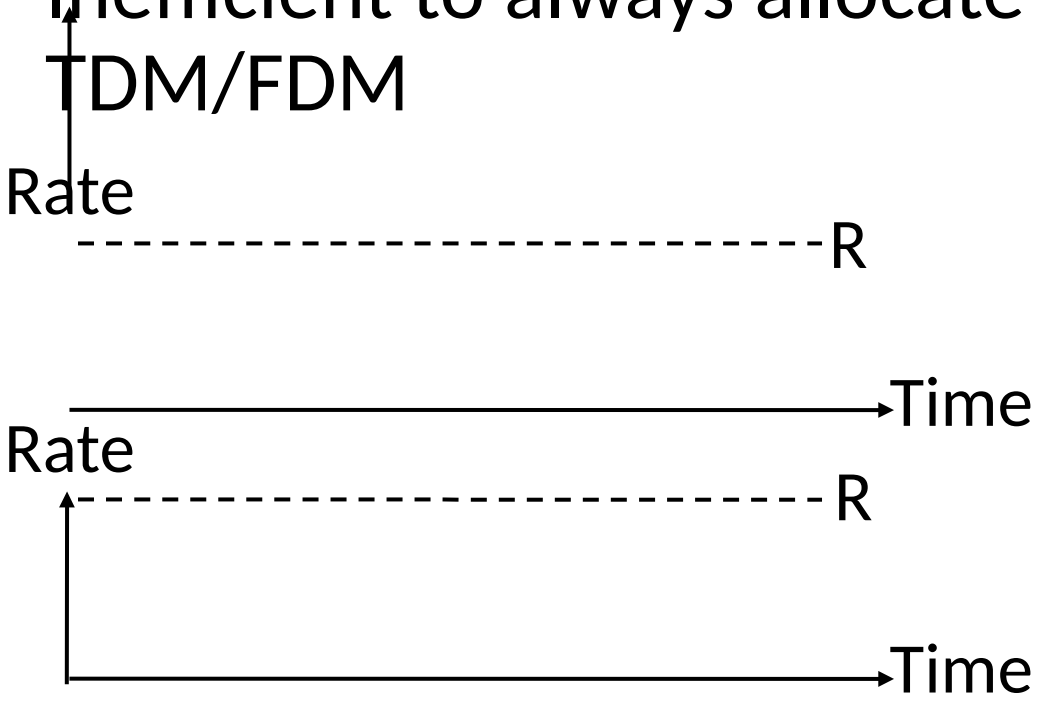
Multiplexing Network Traffic

- Network traffic is bursty
  - ON/OFF sources
  - Load varies greatly over time



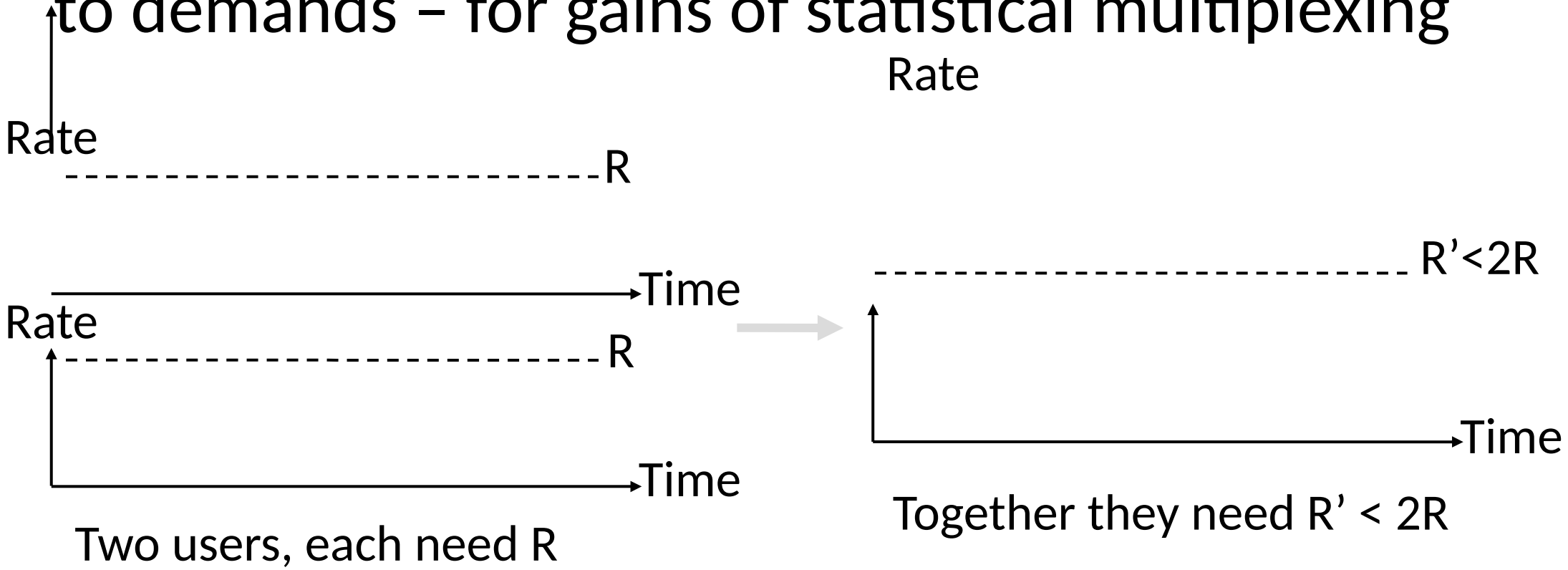
Multiplexing Network Traffic (2)

- Network traffic is bursty
  - Inefficient to always allocate user their ON needs with TDM/FDM



Multiplexing Network Traffic (3)

- Multiple access schemes multiplex users according to demands – for gains of statistical multiplexing



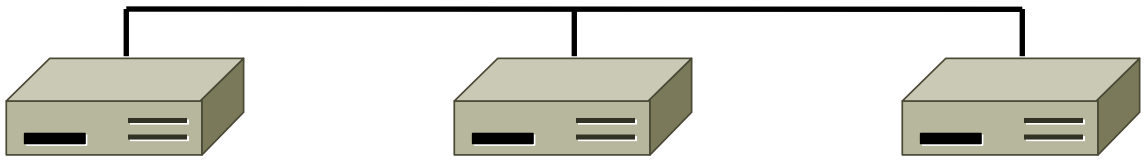
How to control?

Two classes of multiple access algorithms: Centralized and distributed

- Centralized: Use a privileged “Scheduler” to pick who gets to transmit and when.
  - Positives: Scales well, usually efficient.
  - Negatives: Requirements management, fairness
  - Examples: Cellular networks (tower coordinates)
- Distributed: Have all participants “figure it out” through some mechanism.
  - Positives: Operates well under low load, easy to set up, equality
  - Negatives: Scaling is really hard,
  - Examples: Wifi networks

Distributed (random) Access

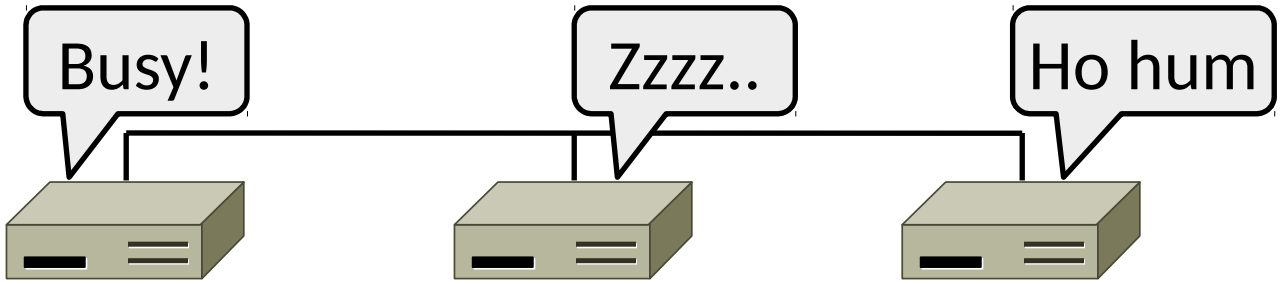
- How do nodes share a single link? Who sends when, e.g., in WiFI?
  - Explore with a simple model



- Assume no-one is in charge
  - Distributed system

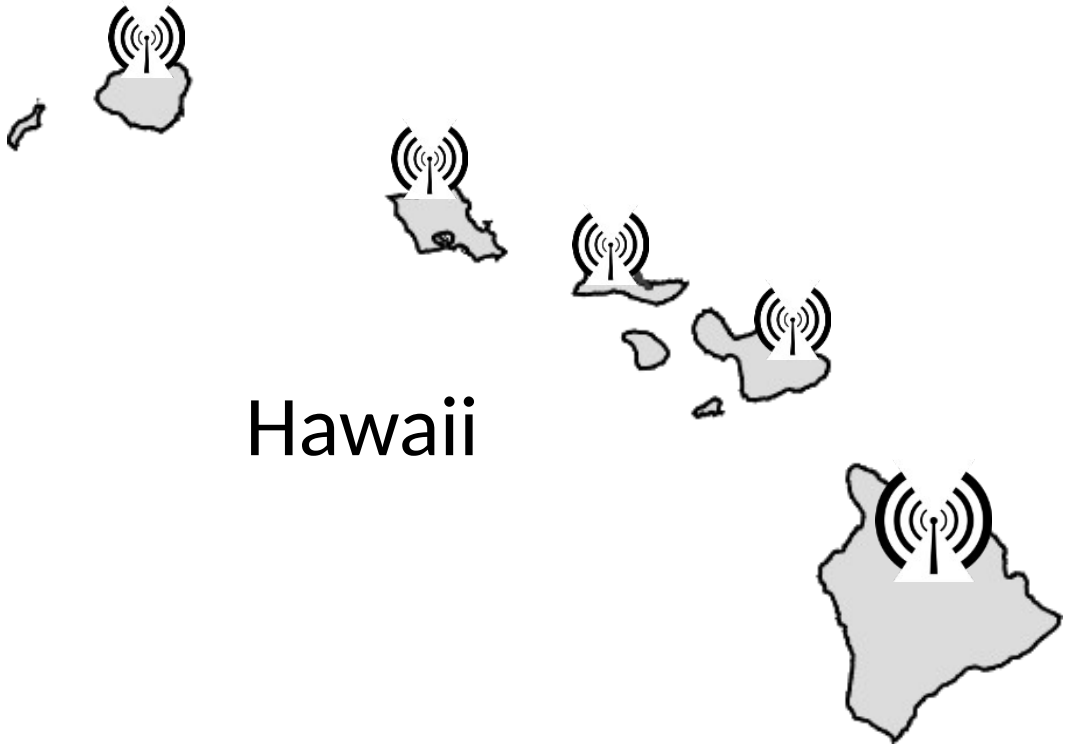
Distributed (random) Access (2)

- We will explore random multiple access control (MAC) protocols
  - This is the basis for classic Ethernet
  - Remember: data traffic is bursty



ALOHA Network

- Seminal computer network connecting the Hawaiian islands in the late 1960s
  - When should nodes send?
  - A new protocol was devised by Norm Abramson …

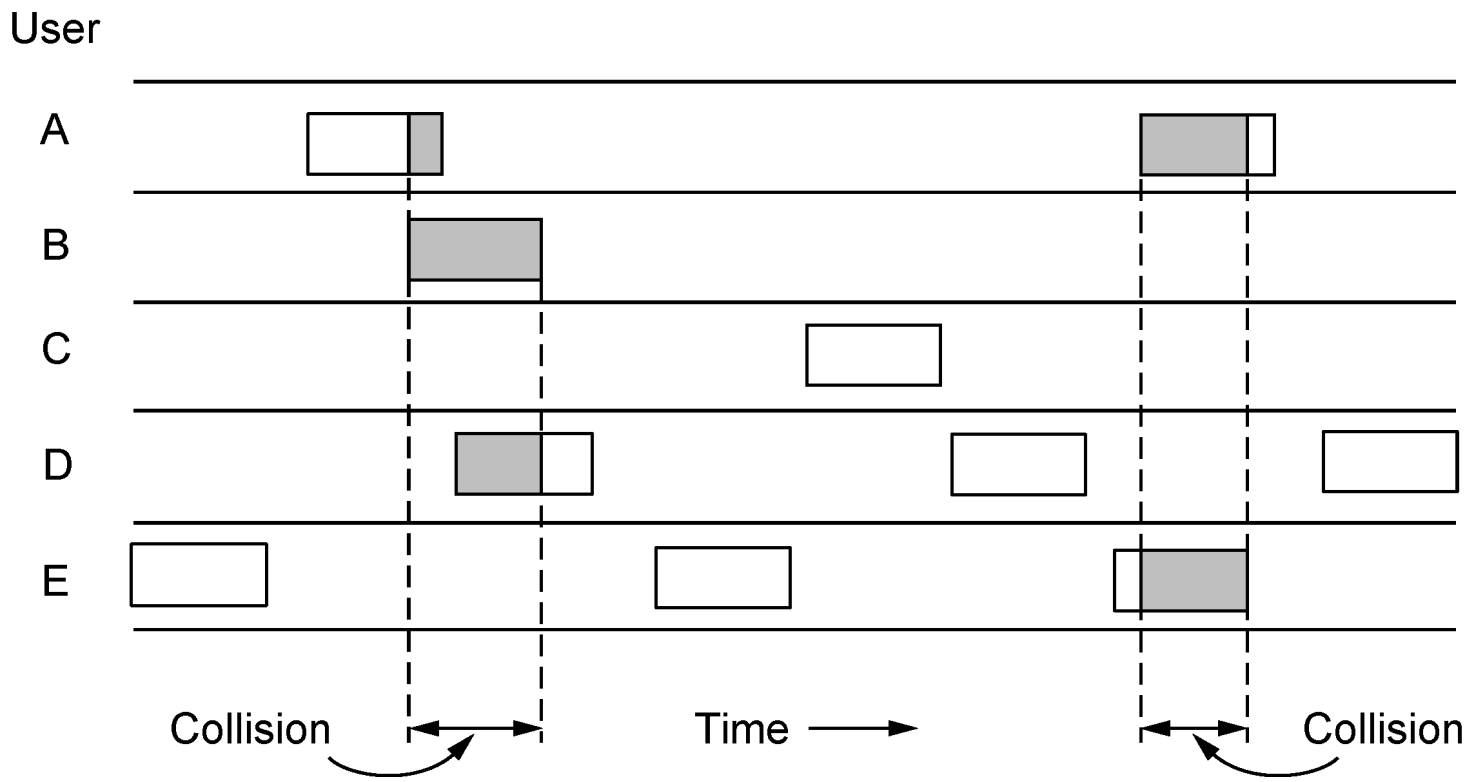


ALOHA Protocol

- Simple idea:
  - Node just sends when it has traffic.
  - If there was a collision (no ACK received) then wait a random time and resend
- That's it!

ALOHA Protocol (2)

- Some frames will be lost, but many may get through…
- Limitations?



ALOHA Protocol (3)

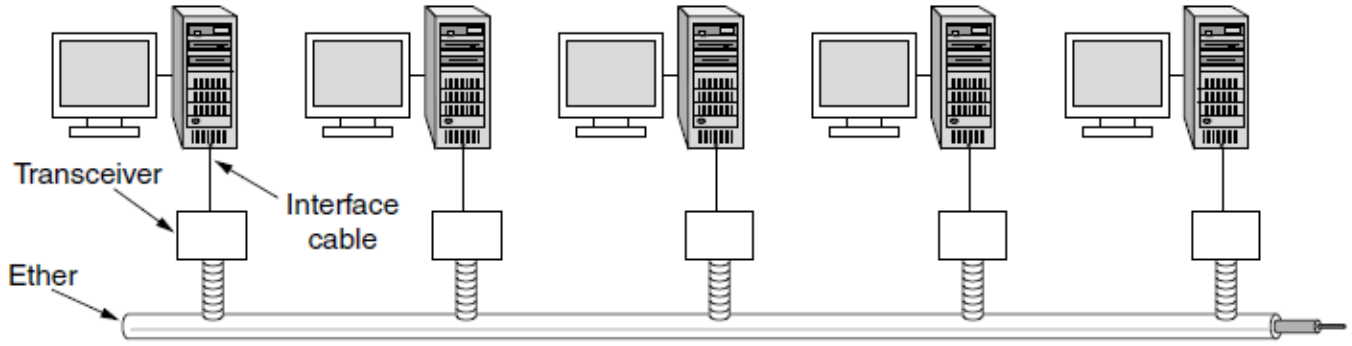
- Simple, decentralized protocol that works well under low load!

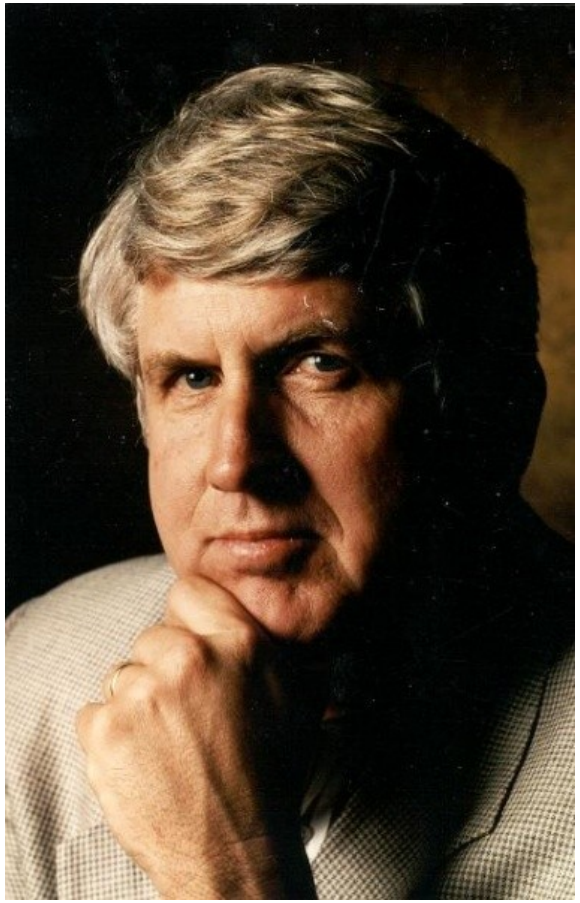
- Not efficient under high load
  - Analysis shows at most 18% efficiency
  - Improvement: divide time into slots and efficiency goes up to 36%

- We'll look at other improvements

Classic Ethernet

- ALOHA inspired Bob Metcalfe to invent Ethernet for LANs in 1973
  - Nodes share 10 Mbps coaxial cable
  - Hugely popular in 1980s, 1990s





: © 2009 IEEE

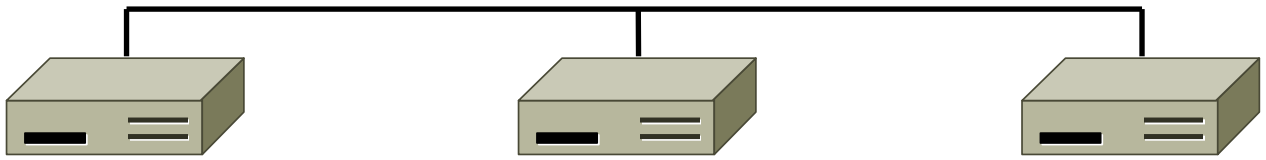
CSMA (Carrier Sense Multiple Access)

- Improve ALOHA by listening for activity before we send (Doh!)
  - Can do easily with wires, not wireless

- So does this eliminate collisions?
  - Why or why not?

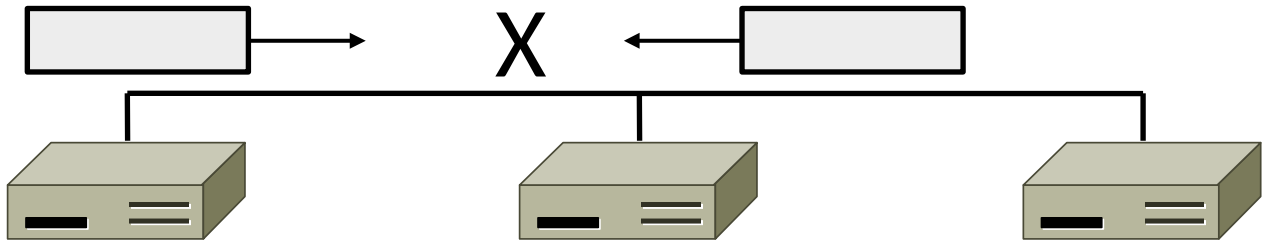
CSMA (2)

- Still possible to listen and hear nothing when another node is sending because of delay



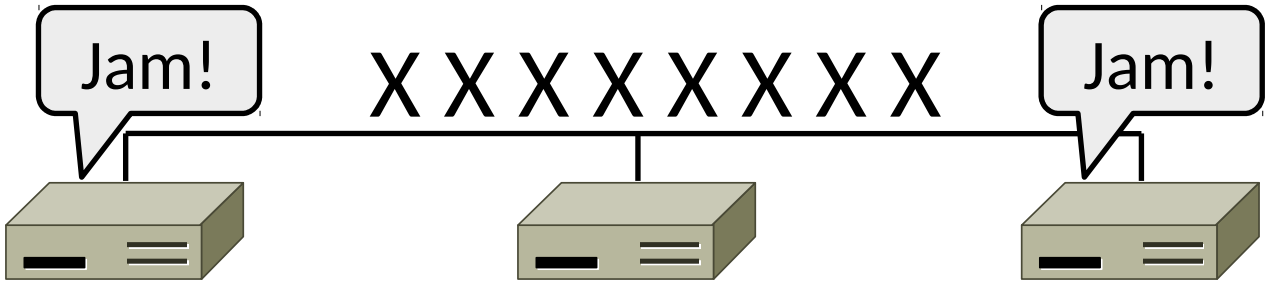
CSMA (3)

- CSMA is a good defense against collisions only when BD is small



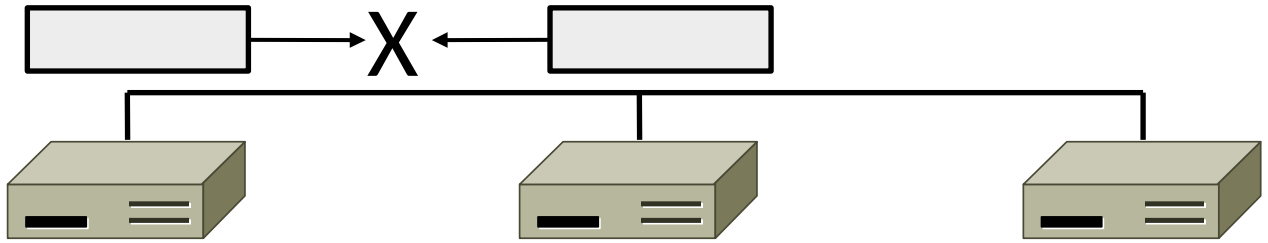
CSMA/CD (with Collision Detection)

- Can reduce the cost of collisions by detecting them and aborting (Jam) the rest of the frame time
  - Again, we can do this with wires



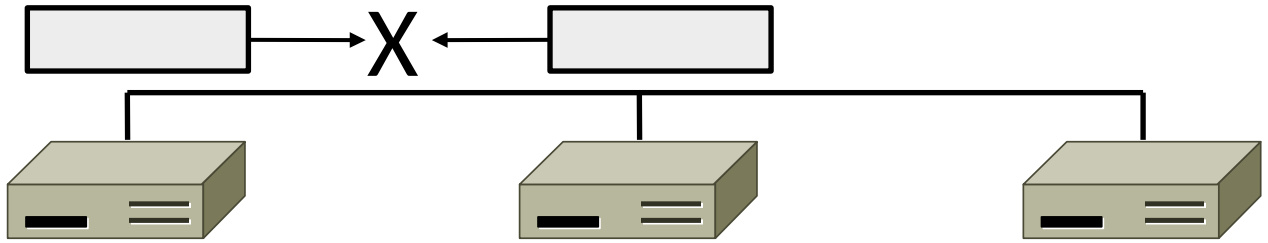
CSMA/CD Complications

- Everyone who collides needs to know it happened
  - How long do we need to wait to know there wasn't a JAM?



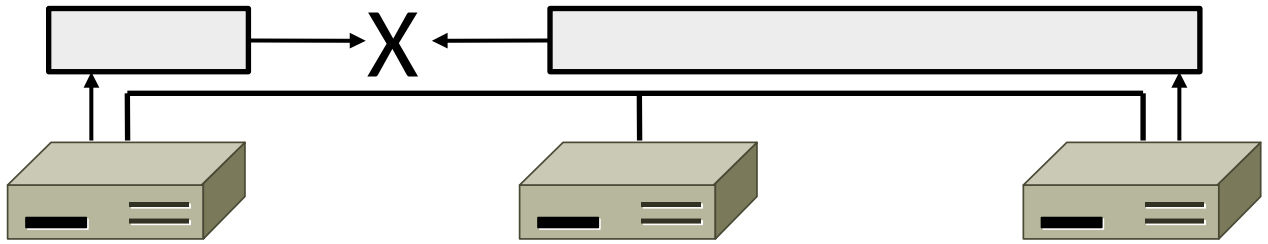
CSMA/CD Complications

- Everyone who collides needs to know it happened
  - How long do we need to wait to know there wasn't a JAM?
  - Time window in which a node may hear of a collision (transmission + jam) is 2D seconds



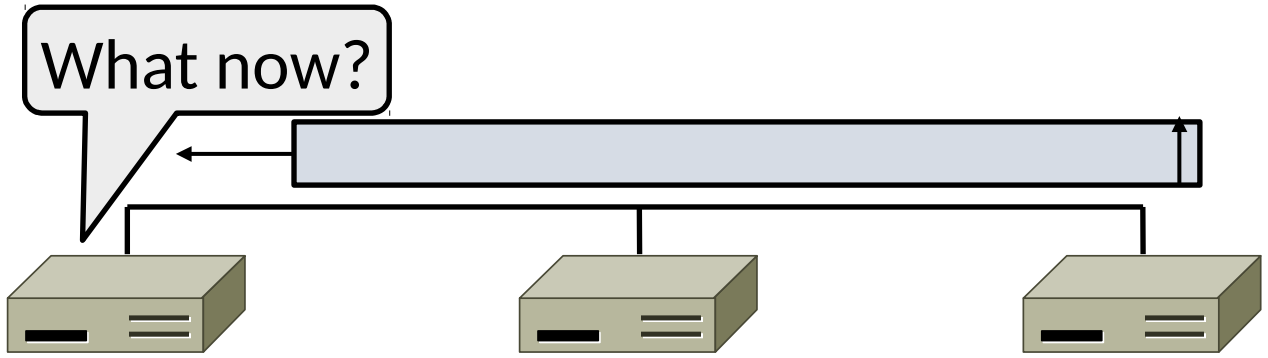
CSMA/CD Complications (2)

- Impose a minimum frame length of 2D seconds
  - So node can't finish before collision
  - Ethernet minimum frame is 64 bytes – Also sets maximum network length (500m w/ coax, 100m w/ Twisted Pair)



CSMA “Persistence”

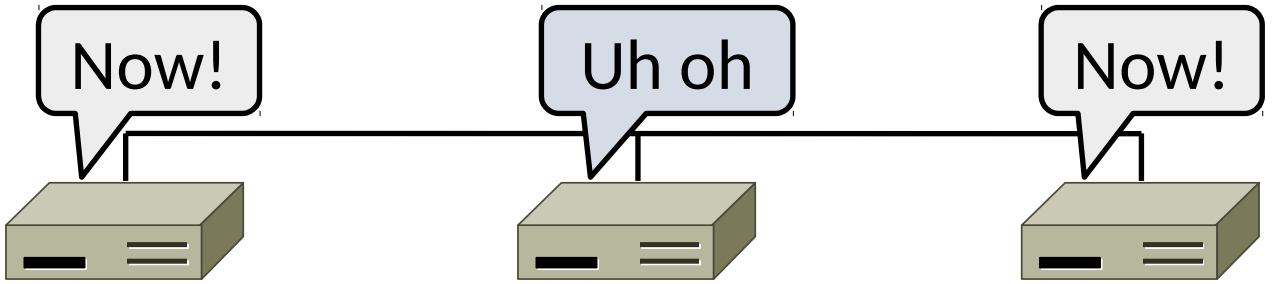
- What should a node do if another node is sending?



- Idea: Wait until it is done, and send

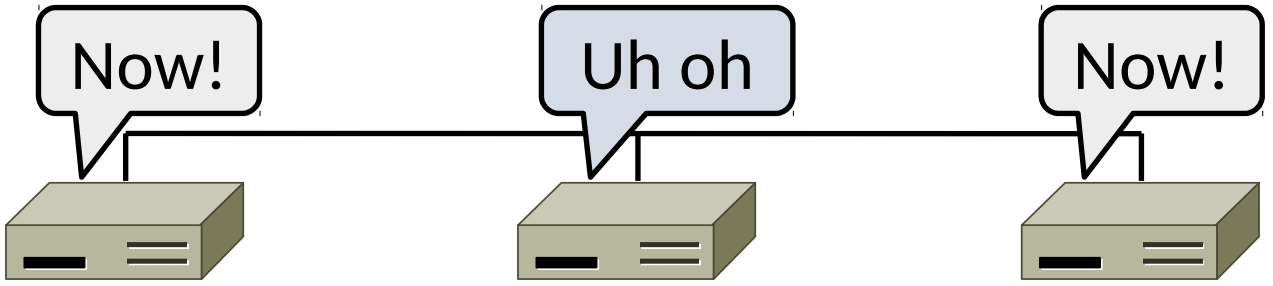
CSMA “Persistence” (2)

- Problem is that multiple waiting nodes will queue up then collide
  - More load, more of a problem



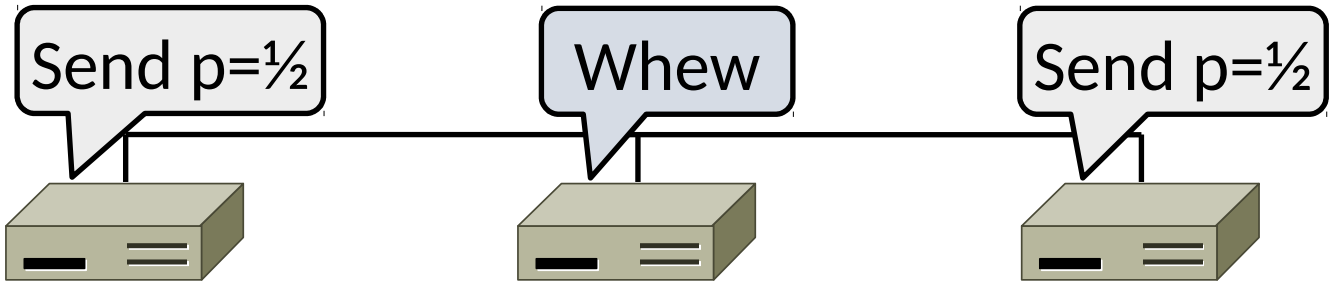
CSMA “Persistence” (2)

- Problem is that multiple waiting nodes will queue up then collide
  - Ideas?



CSMA "Persistence" (3)

- Intuition for a better solution
  - If there are N queued senders, we want each to send next with probability 1/N

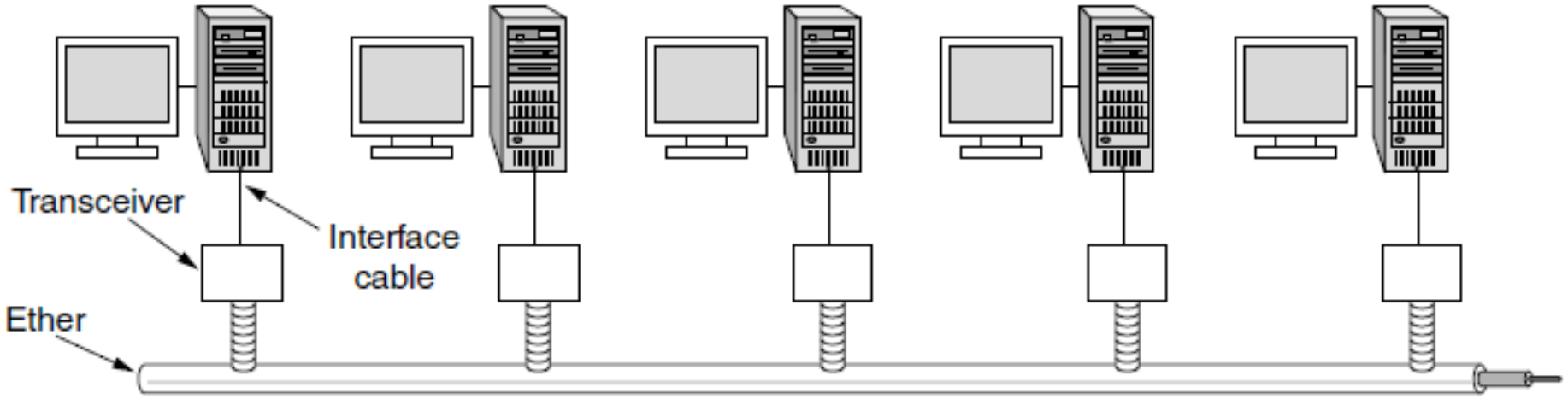


Binary Exponential Backoff (BEB)

- Cleverly estimates the probability
  - 1st collision, wait 0 or 1 frame times
  - 2nd collision, wait from 0 to 3 times
  - 3rd collision, wait from 0 to 7 times …
- BEB doubles interval for each successive collision
  - Quickly gets large enough to work
  - Very efficient in practice

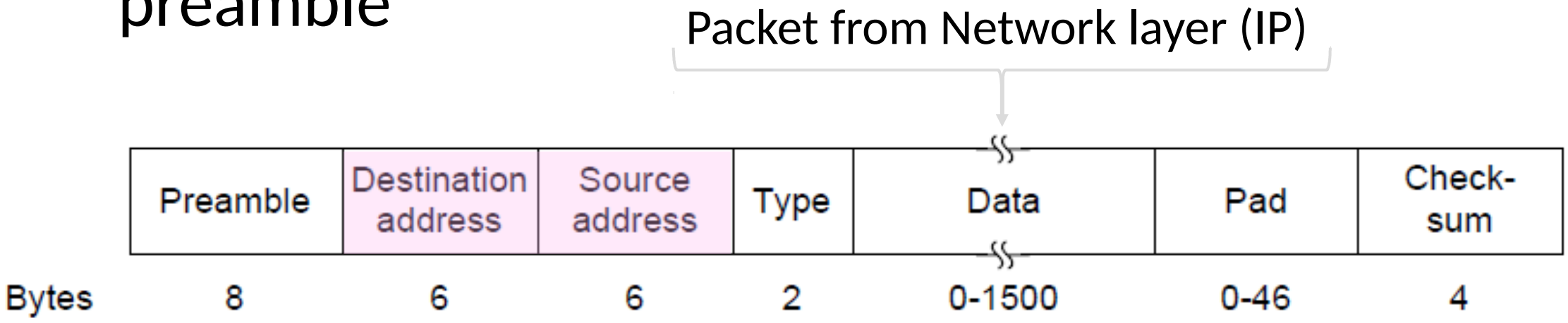
Classic Ethernet, or IEEE 802.3

- Most popular LAN of the 1980s, 1990s
  - 10 Mbps over shared coaxial cable, with baseband signals
  - Multiple access with "1-persistent CSMA/CD with BEB"



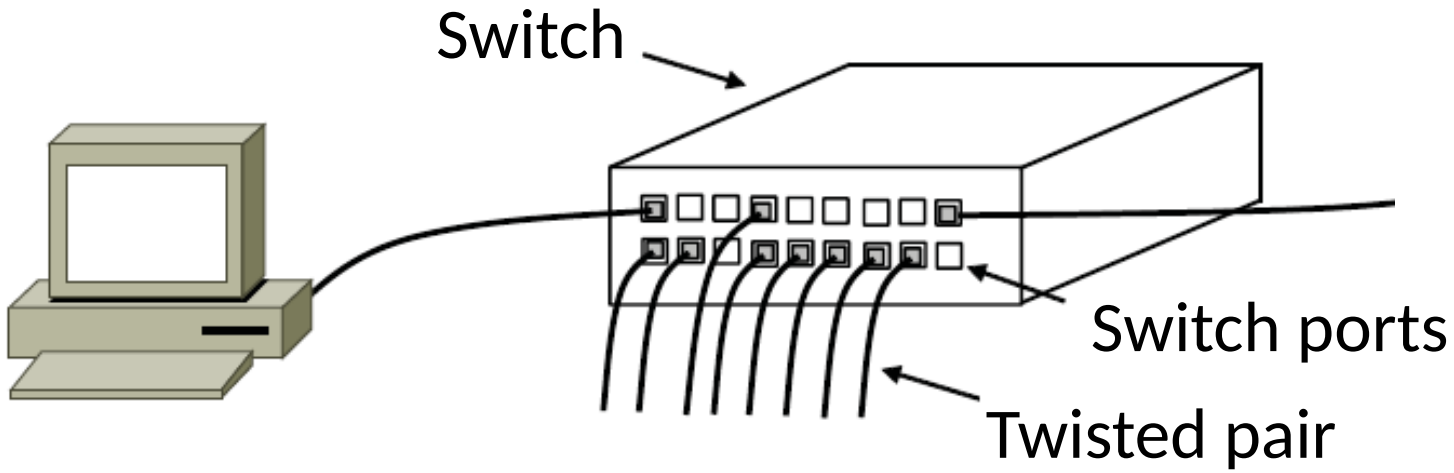
Ethernet Frame Format

- Has addresses to identify the sender and receiver
- CRC-32 for error detection; no ACKs or retransmission
- Start of frame identified with physical layer preamble



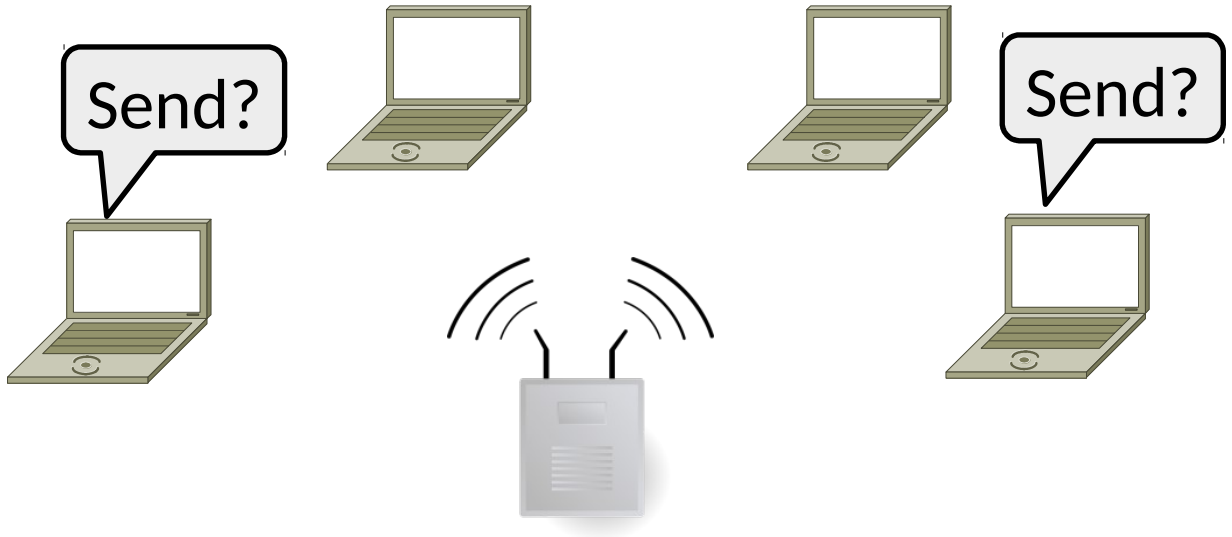
Modern Ethernet

- Based on switches, not multiple access, but still called Ethernet
  - We'll get to it in a later segment



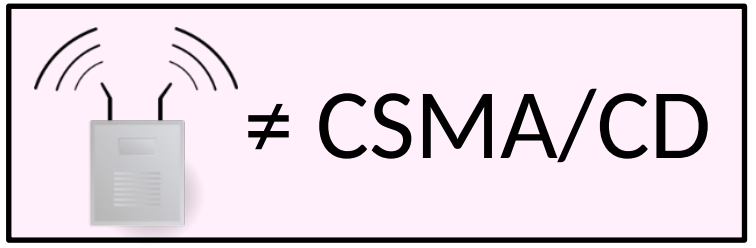
Topic

- How do wireless nodes share a single link? (Yes, this is WiFi!)
  - Build on our simple, wired model



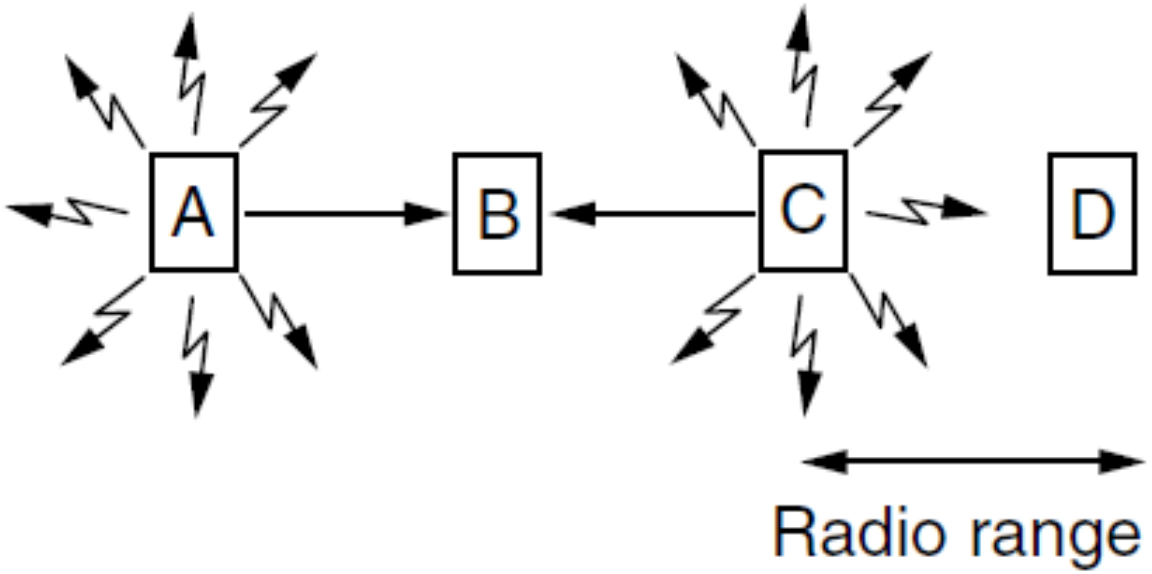
Wireless Complications

- Wireless is more complicated than the wired case (Surprise!)
  1. Media is infinite – can't Carrier Sense
  2. Nodes can't hear while sending – can't Collision Detect



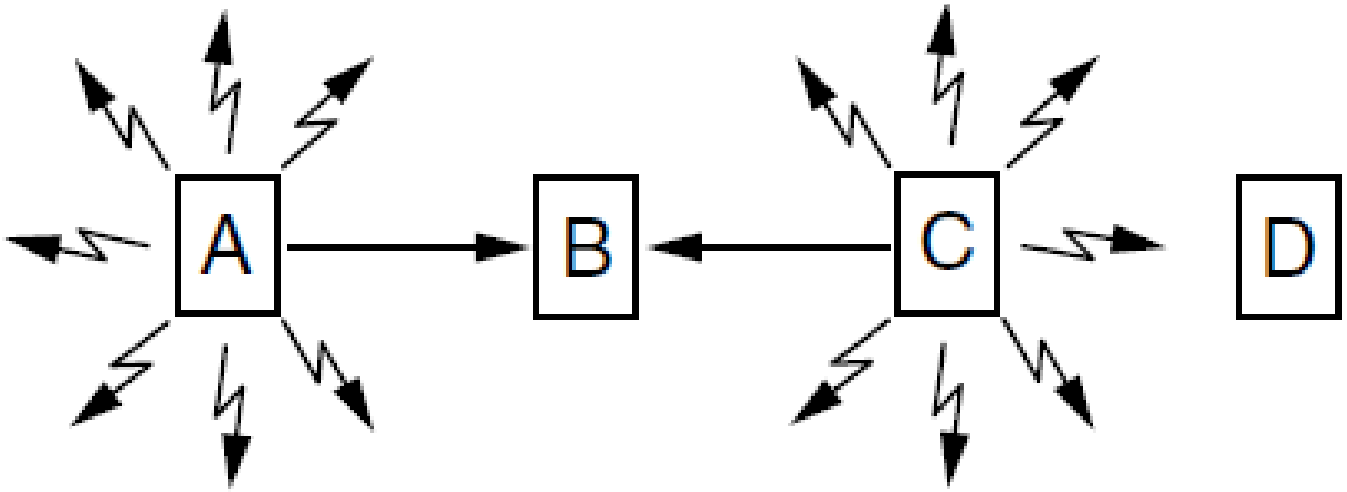
No CS: Different Coverage Areas

- Wireless signal is broadcast and received nearby, where there is sufficient SNR



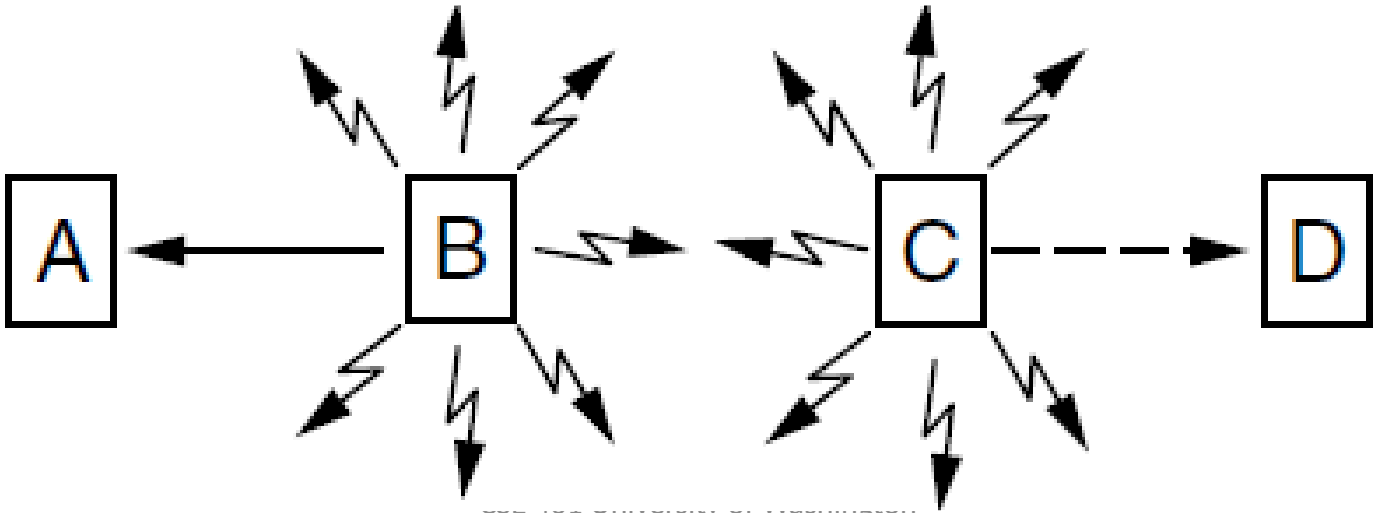
No CS: Hidden Terminals

- Nodes A and C are hidden terminals when sending to B
  - Can't hear each other (to coordinate) yet collide at B
  - We want to avoid the inefficiency of collisions



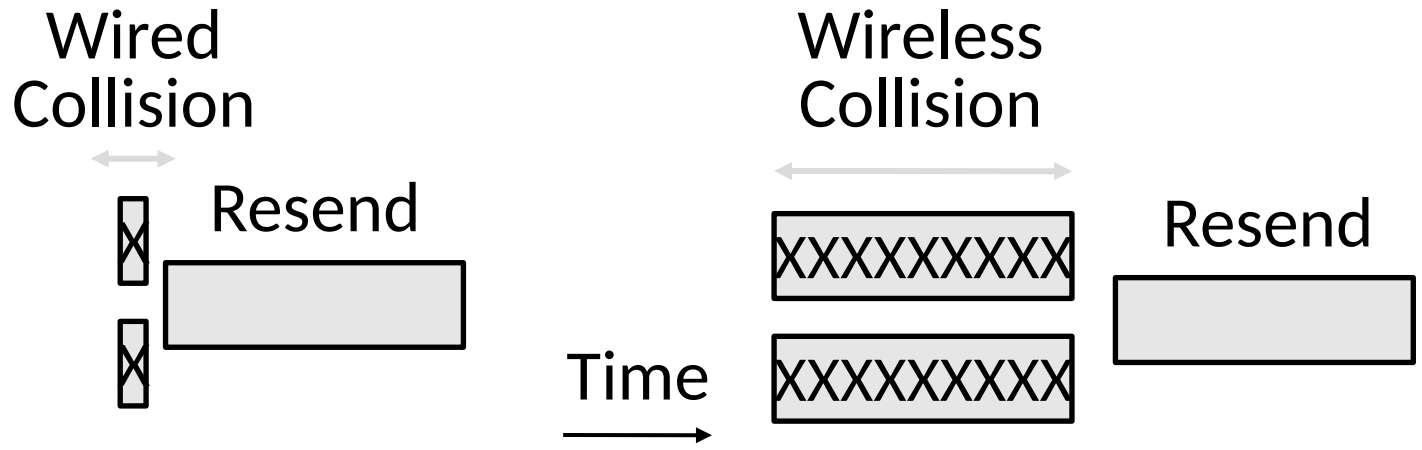
No CS: Exposed Terminals

- B and C are exposed terminals when sending to A and D
  - Can hear each other yet don't collide at receivers A and D
  - We want to send concurrently to increase performance



Nodes Can't Hear While Sending

- With wires, detecting collisions (and aborting) lowers their cost
- More wasted time with wireless



Wireless Problems:

- Ideas?

MACA (Multiple Access with Collision Avoidance)

- MACA uses a short handshake instead of CSMA (Karn, 1990)
  - 802.11 uses a refinement of MACA (later)

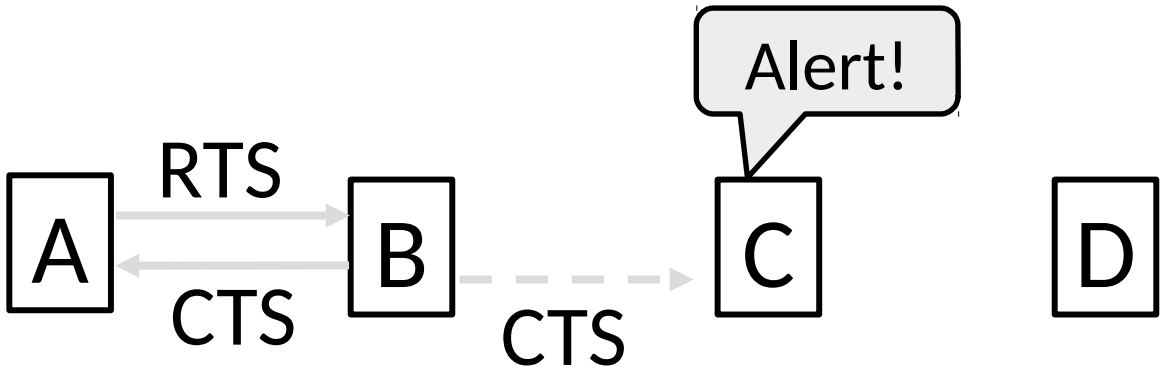
- Protocol rules:
  1. A sender node transmits a RTS (Request-To-Send, with frame length)
  2. The receiver replies with a CTS (Clear-To-Send, with frame length)
  3. Sender transmits the frame while nodes hearing the CTS stay silent
  - Collisions on the RTS/CTS are still possible, but less likely

MACA – Hidden Terminals

- A → B with hidden terminal C
  1. A sends RTS, to B

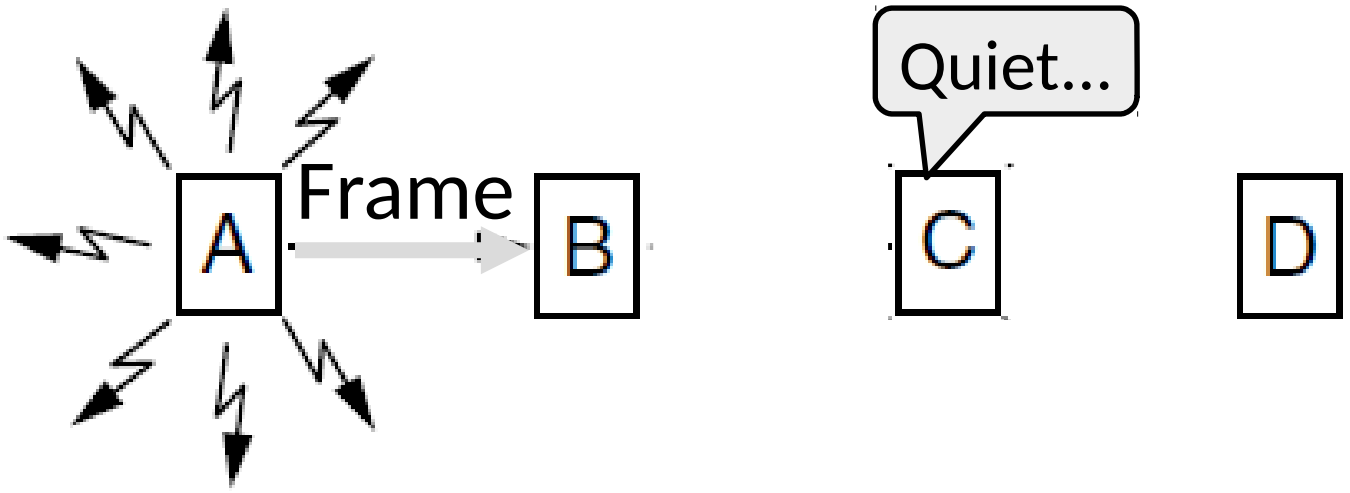
MACA – Hidden Terminals (2)

- A → B with hidden terminal C
  2. B sends CTS, to A, and C too



MACA – Hidden Terminals (3)

- A → B with hidden terminal C
  3. A sends frame while C defers

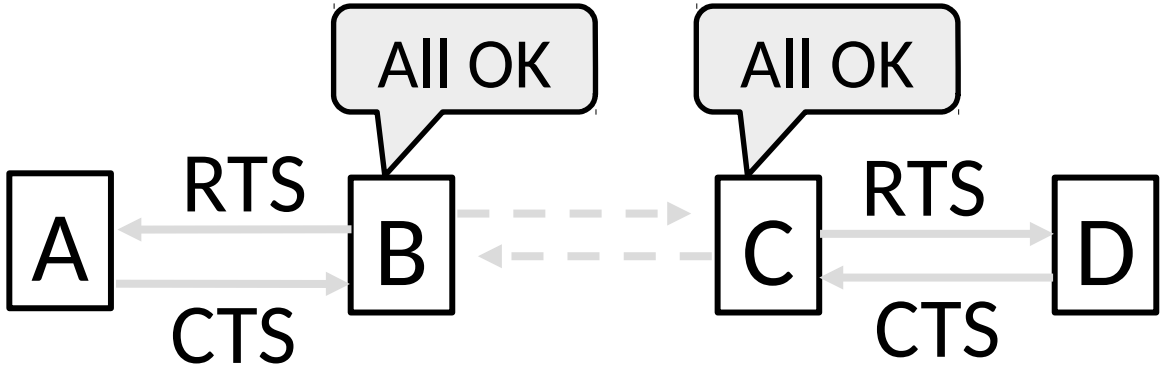


MACA – Exposed Terminals

- B → A, C → D as exposed terminals
  - B and C send RTS to A and D

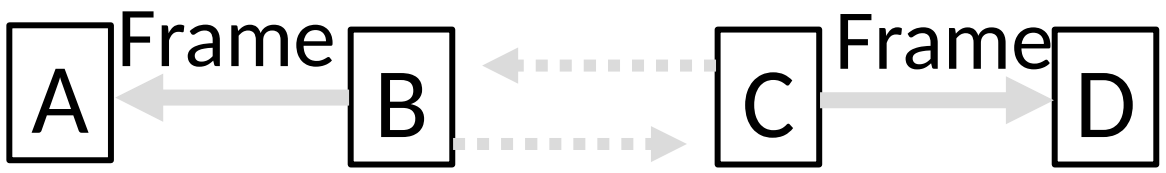
MACA – Exposed Terminals (2)

- B → A, C→ D as exposed terminals
  - A and D send CTS to B and C



MACA – Exposed Terminals (3)

- B → A, C → D as exposed terminals
  - A and D send CTS to B and C



MACA

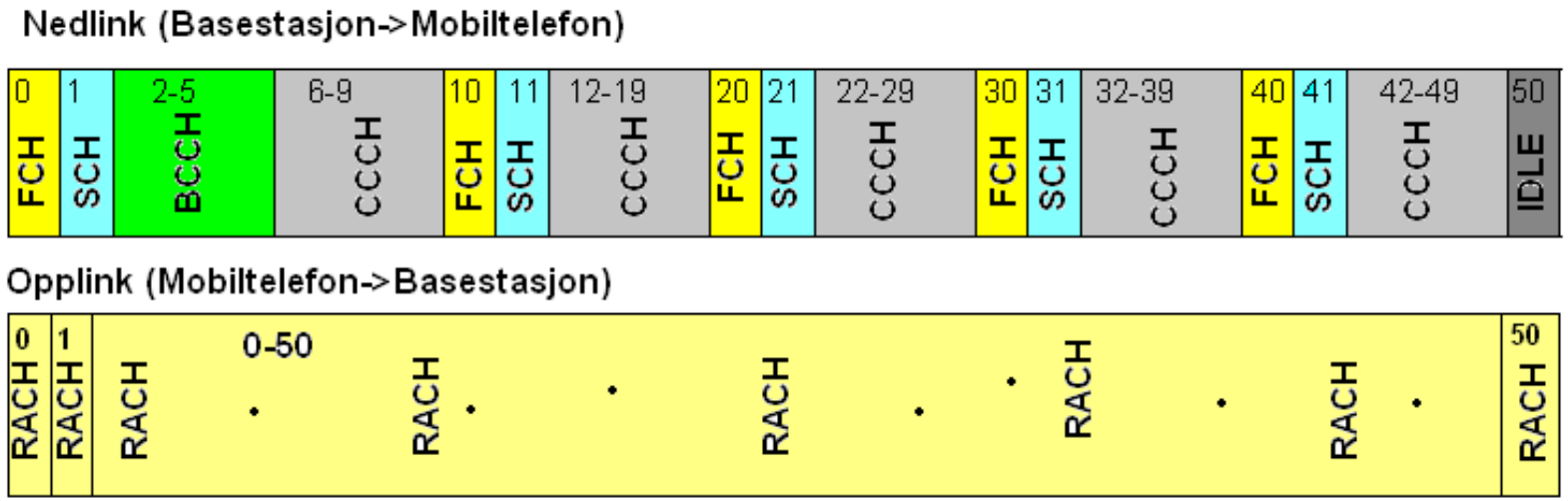
- Assumptions? Where does this break?

Centralized MAC: Cellular

- Spectrum suddenly very very scarce
  - We can’t waste all of it sending JAMs
- We have QoS requirements
  - Can’t be as loose with expectations
  - Can’t have traffic fail
- We also have client/server
  - Centralized control
  - Not peer-to-peer/decentralized

GSM MAC

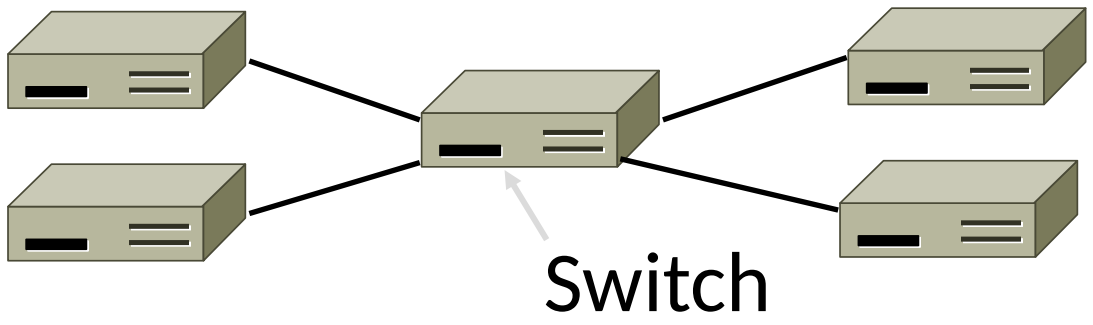
- FDMA/TDMA
- Use one channel for coordination – Random access w/BEB (no CSMA, can’t detect)
- Use other channels for traffic
  - Dedicated channel for QoS



Link Layer: Switching

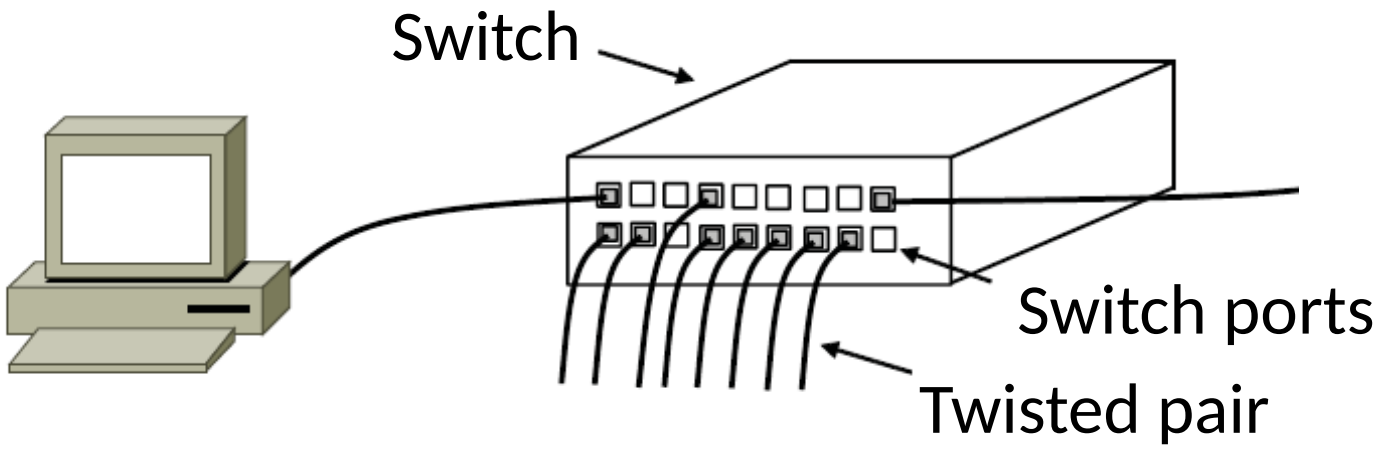
Topic

- How do we connect nodes with a switch instead of multiple access
  - Uses multiple links/wires
  - Basis of modern (switched) Ethernet



Switched Ethernet

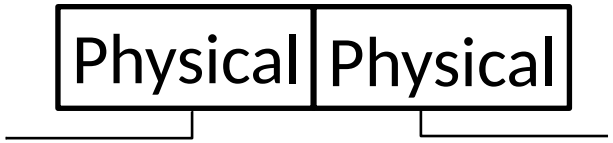
- Hosts are wired to Ethernet switches with twisted pair
  - Switch serves to connect the hosts
  - Wires usually run to a closet



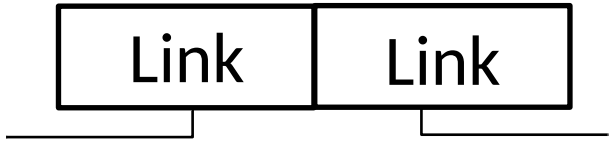
What's in the box?

- Remember from protocol layers:

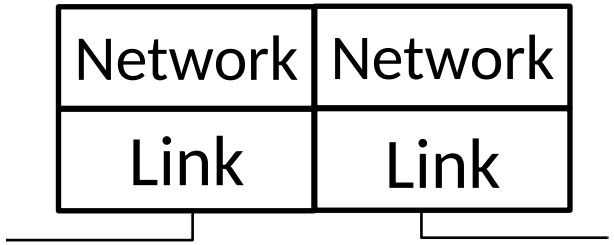
Hub, or repeater



Switch



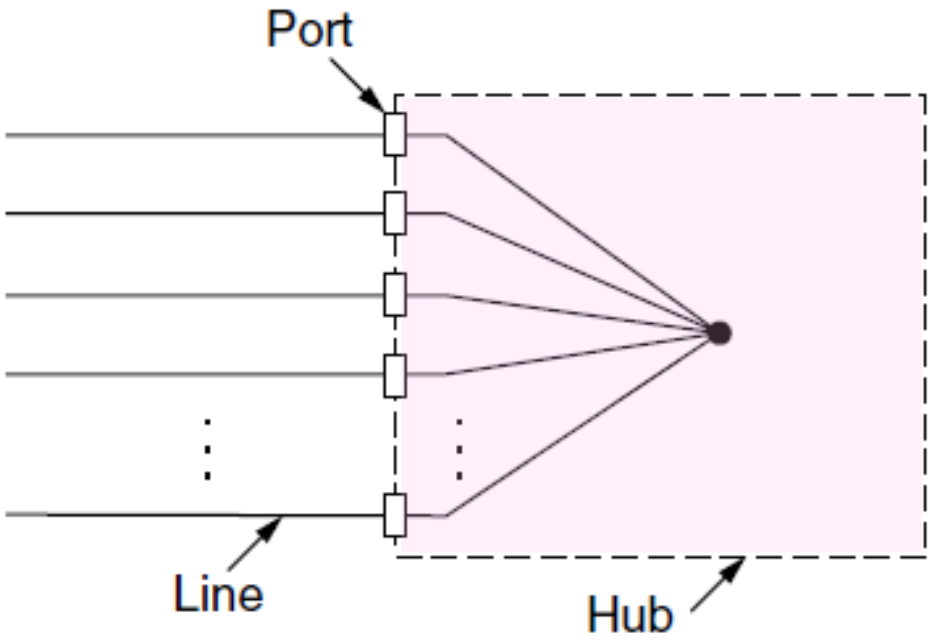
Router



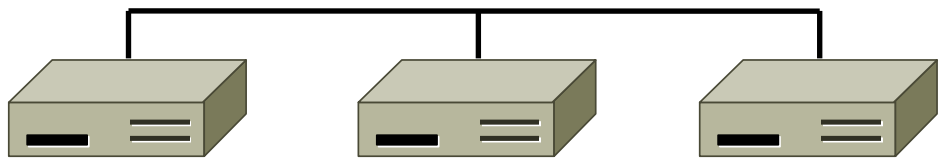
All look like this:

Inside a Hub

- All ports are wired together; more convenient and reliable than a single shared wire

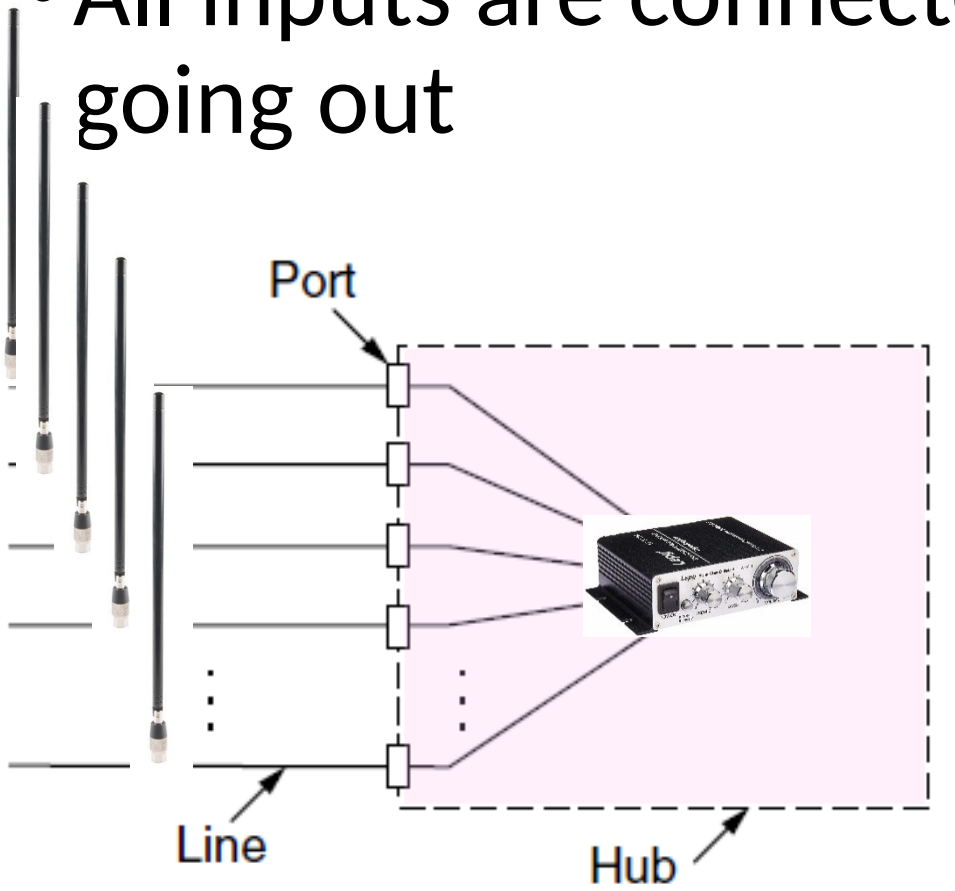


↔

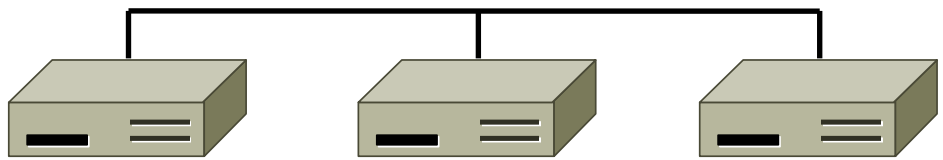


Inside a Repeater

- All inputs are connected; then amplified before going out

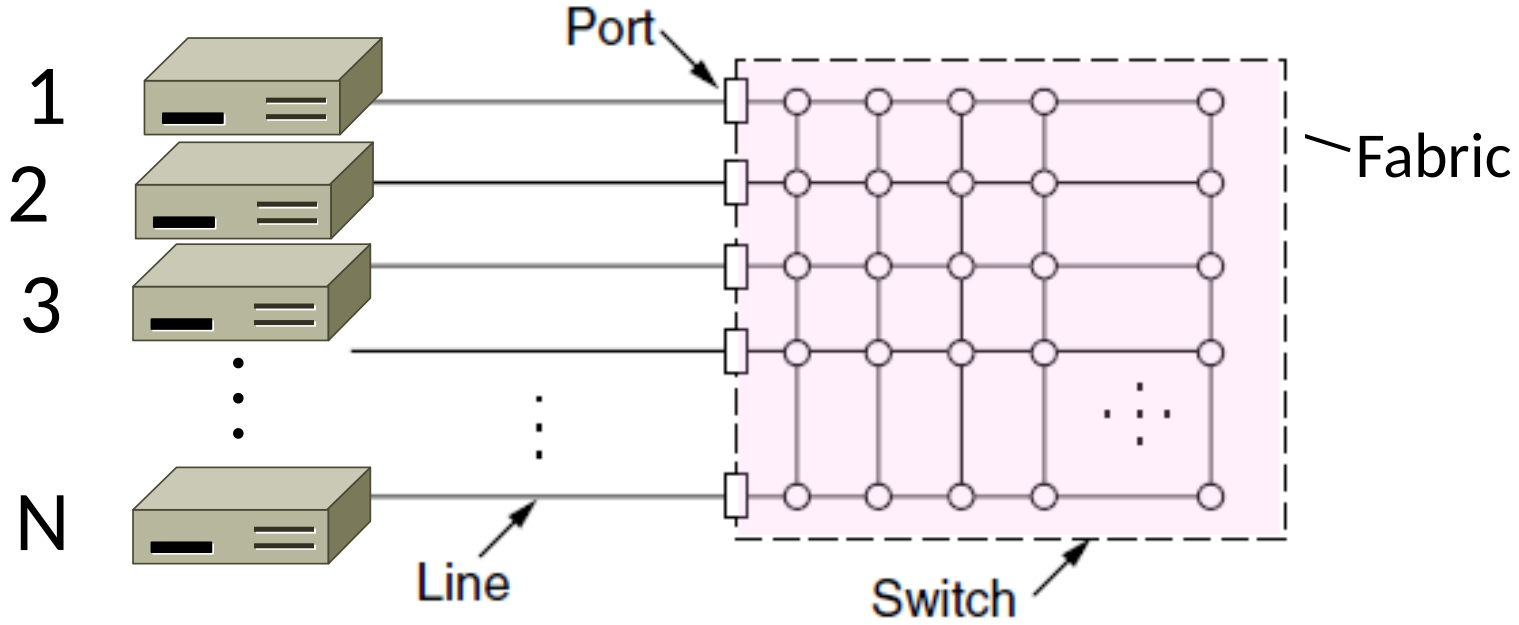


↔



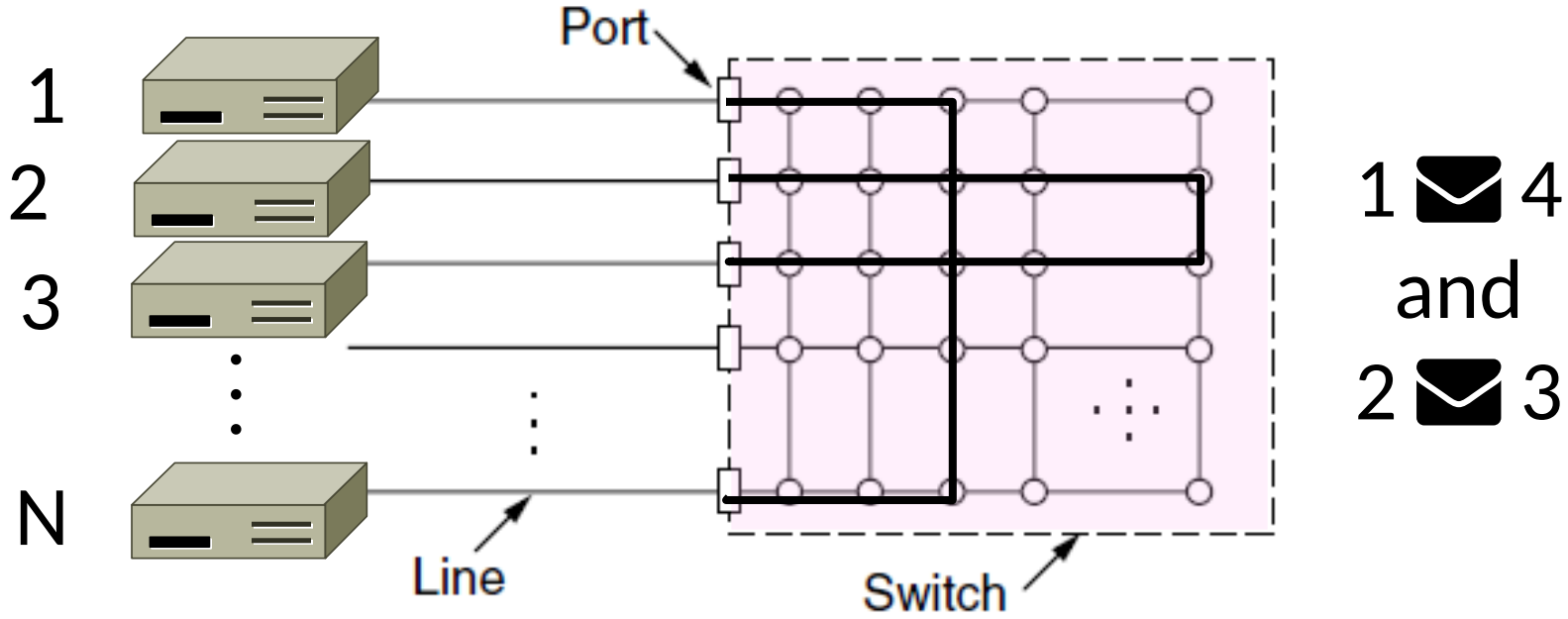
Inside a Switch

- Uses frame addresses (MAC addresses in Ethernet) to connect input port to the right output port; multiple frames may be switched in parallel



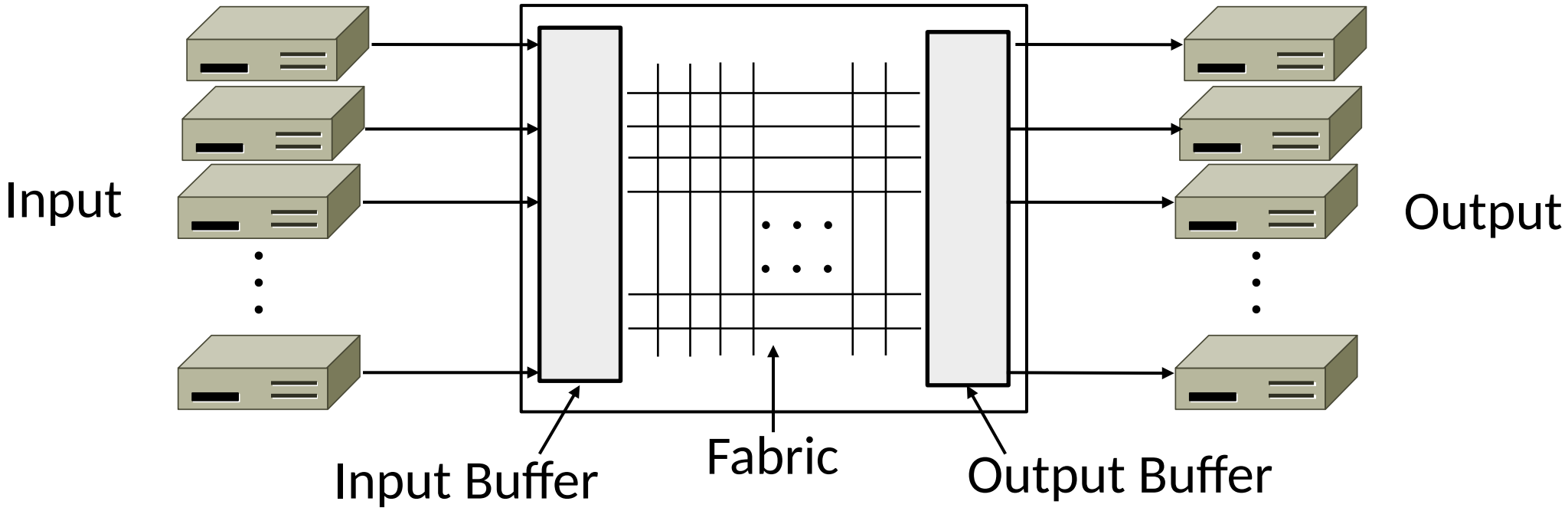
Inside a Switch (2)

- Port may be used for both input and output (full-duplex)
  - Just send, no multiple access protocol



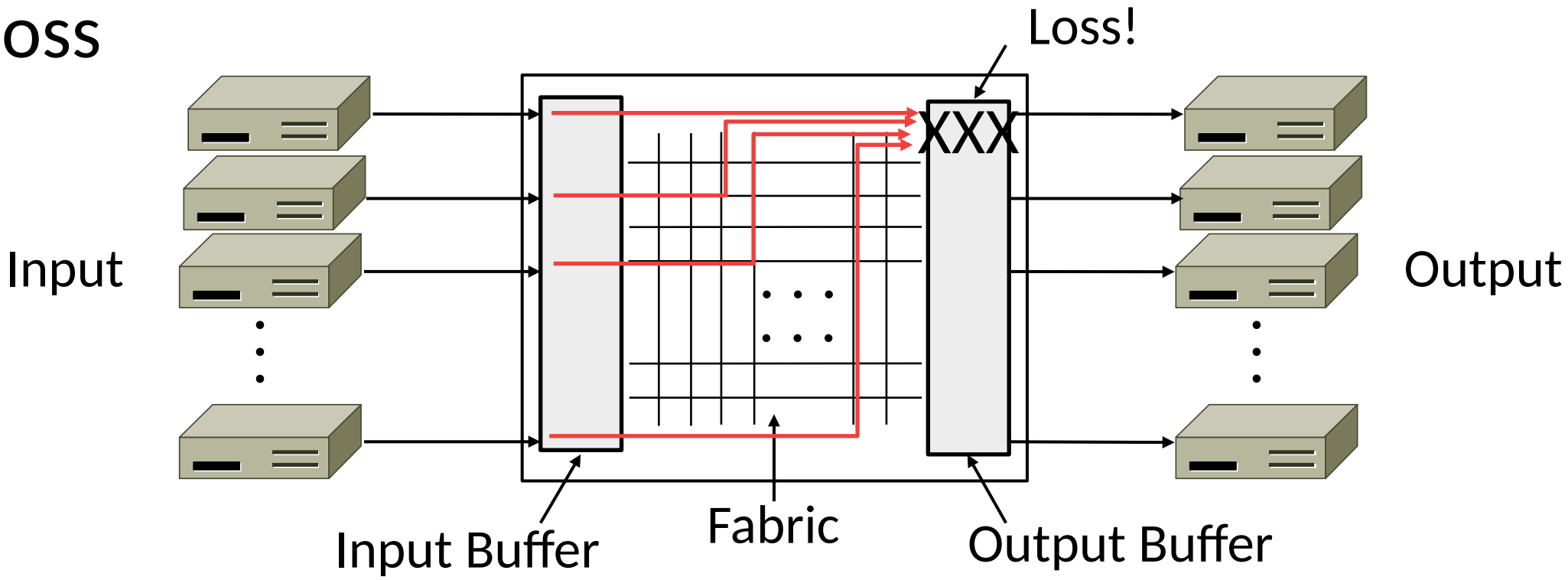
Inside a Switch (3)

- Need buffers for multiple inputs to send to one output



Inside a Switch (4)

- Sustained overload will fill buffer and lead to frame loss



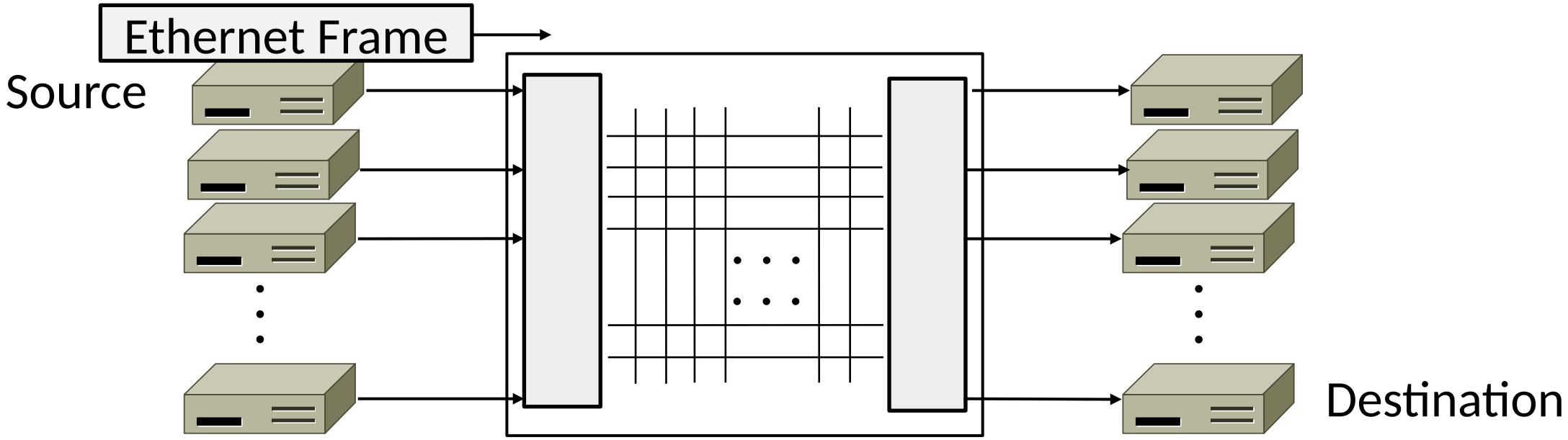
Advantages of Switches

- Switches and hubs (mostly switches) have replaced the shared cable of classic Ethernet
  - Convenient to run wires to one location
  - More reliable; wire cut is not a single point of failure that is hard to find

- Switches offer scalable performance
  - E.g., 100 Mbps per port instead of 100 Mbps for all nodes of shared cable / hub

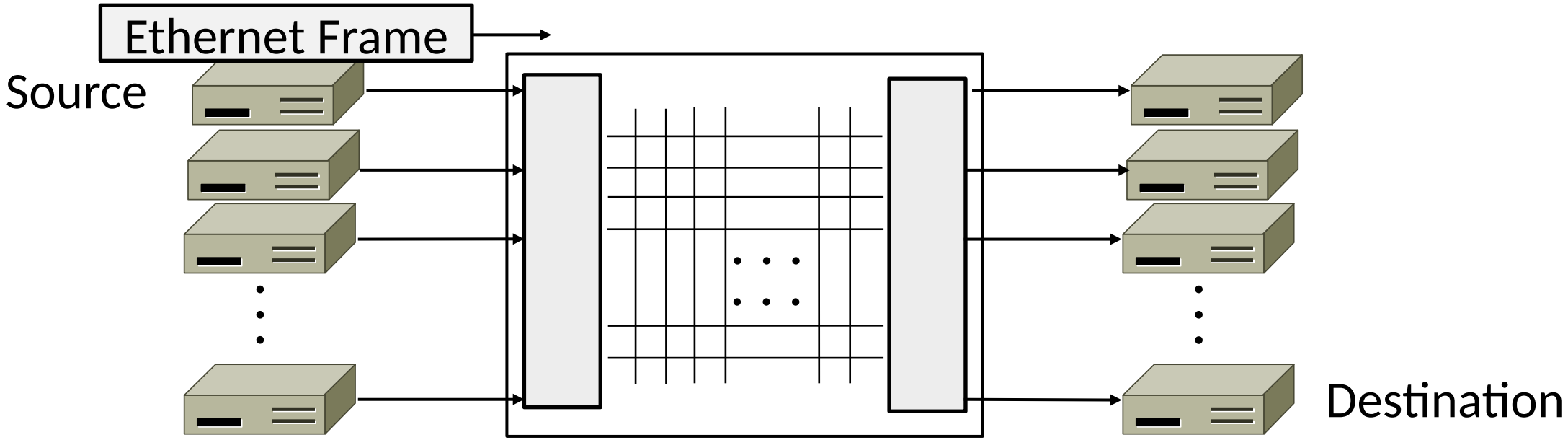
Switch Forwarding

- Switch needs to find the right output port for the destination address in the Ethernet frame. How?
  - Link-level, don't look at IP



Switch Forwarding

- Ideas?

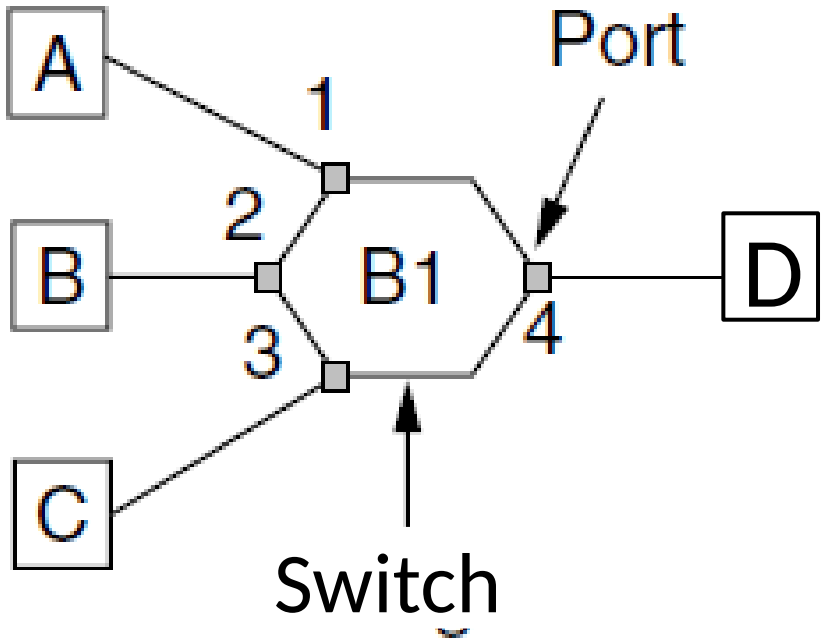


Backward Learning

- Switch forwards frames with a port/address table as follows:
  1. To fill the table, it looks at the source address of input frames
  2. To forward, it sends to the port, or else broadcasts to all ports

Backward Learning (2)

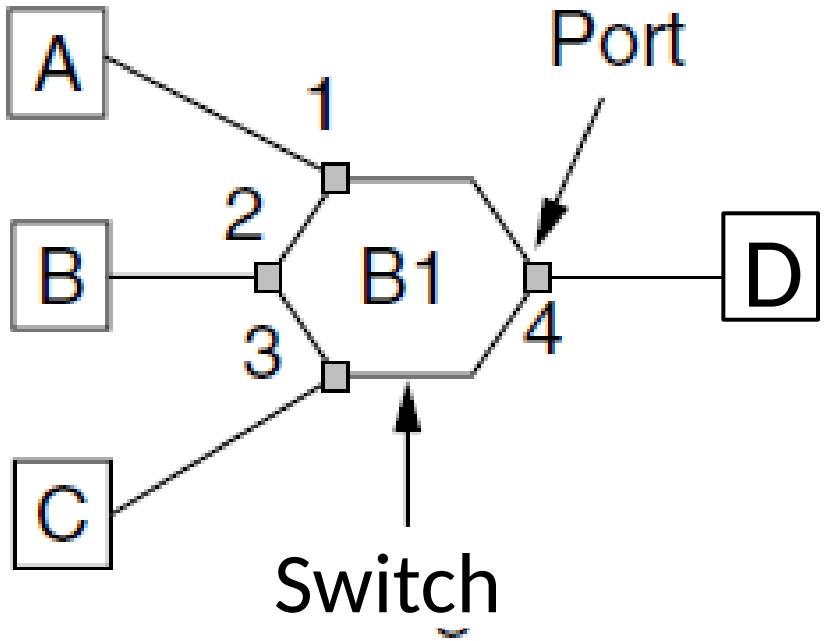
- 1: A sends to D



Address	Port
A	
B	
C	
D	

Backward Learning (3)

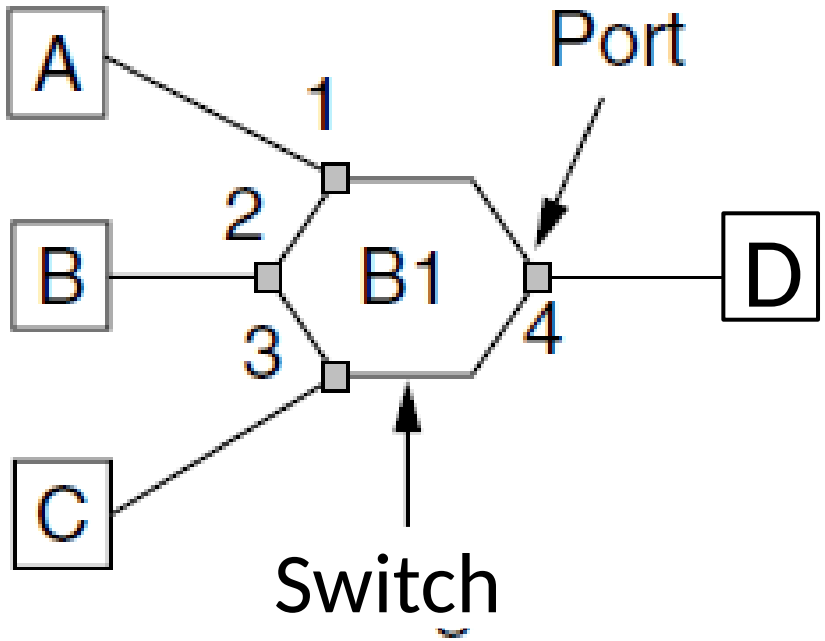
- 2: D sends to A



Address	Port
A	1
B	
C	
D	

Backward Learning (4)

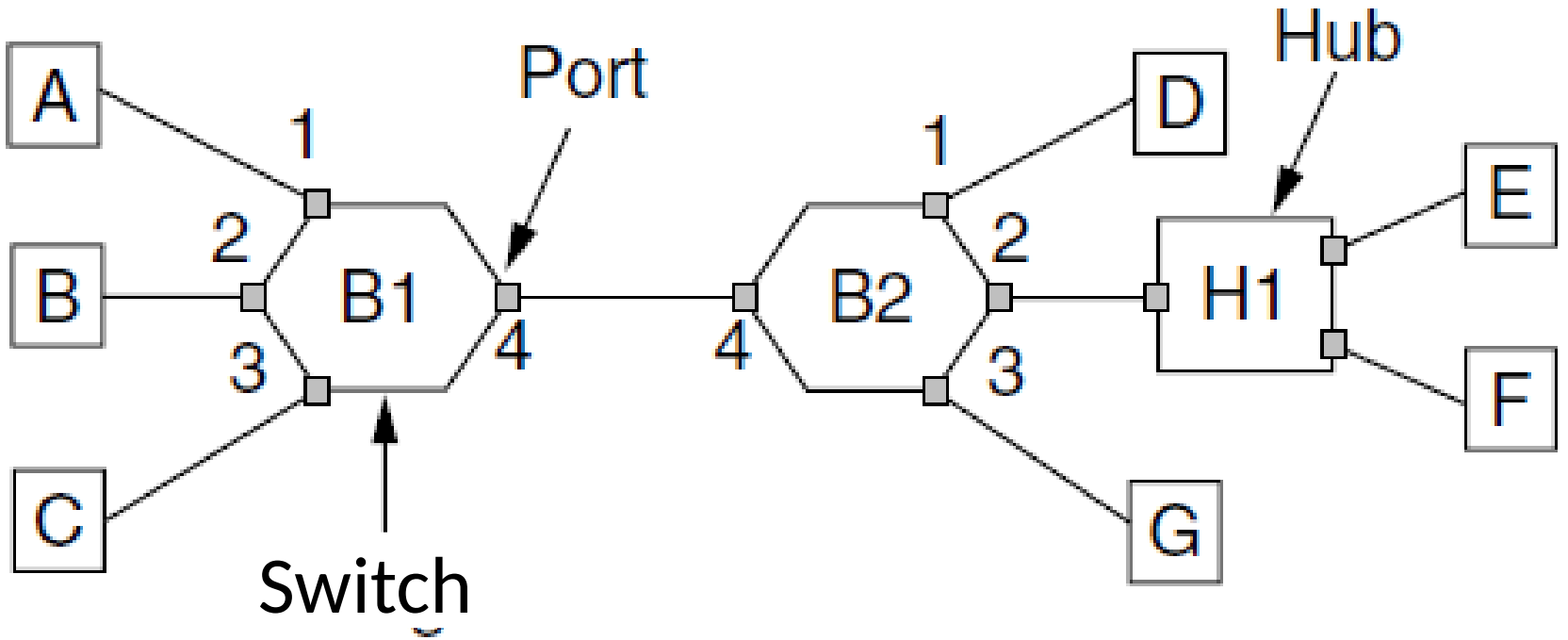
- 3: A sends to D



Address	Port
A	1
B	
C	
D	4

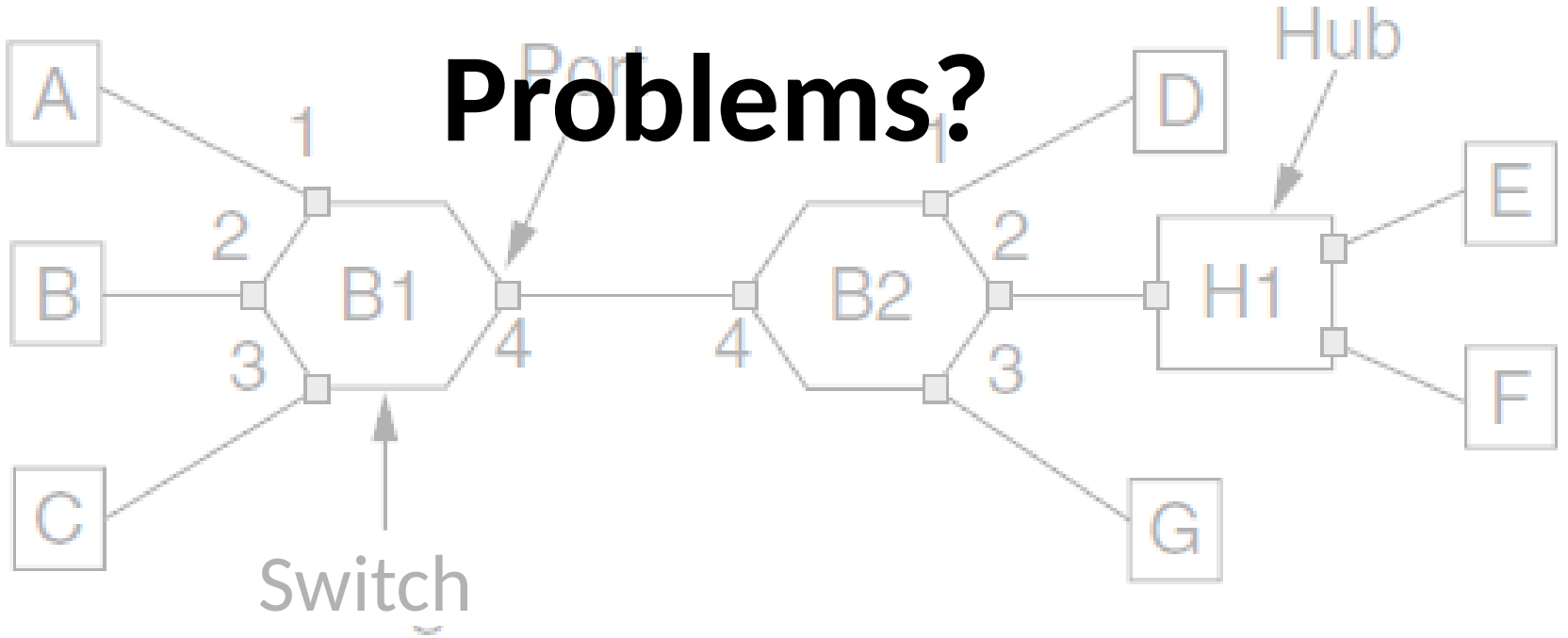
Learning with Multiple Switches

- Just works with multiple switches and a mix of hubs, e.g., A -> D then D -> A



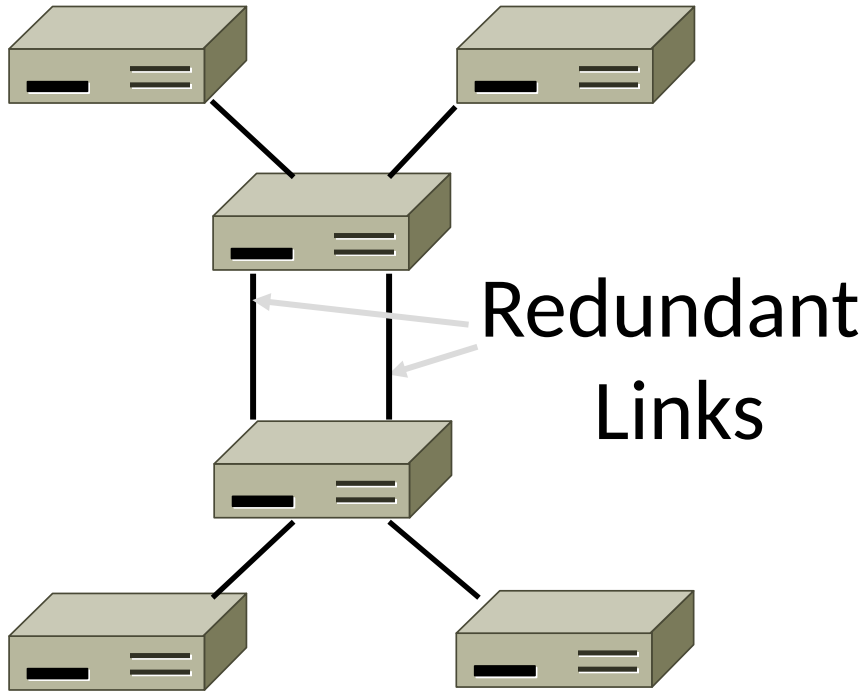
Learning with Multiple Switches

- Just works with multiple switches and a mix of hubs, e.g., A -> D then D -> A



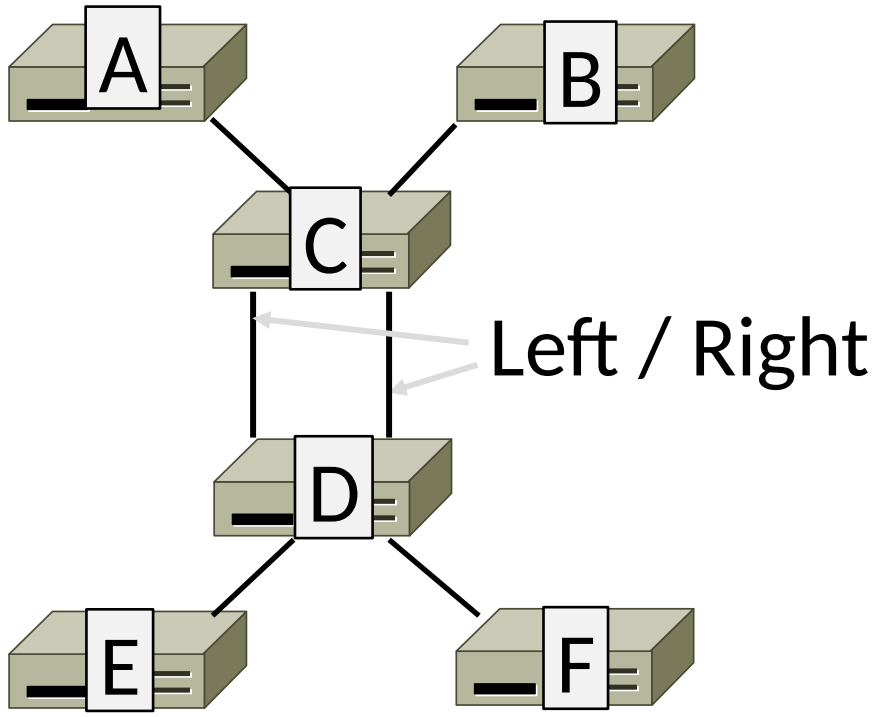
Problem – Forwarding Loops

- May have a loop in the topology
  - Redundancy in case of failures
  - Or a simple mistake
- Want LAN switches to "just work"
  - Plug-and-play, no changes to hosts
  - But loops cause a problem …



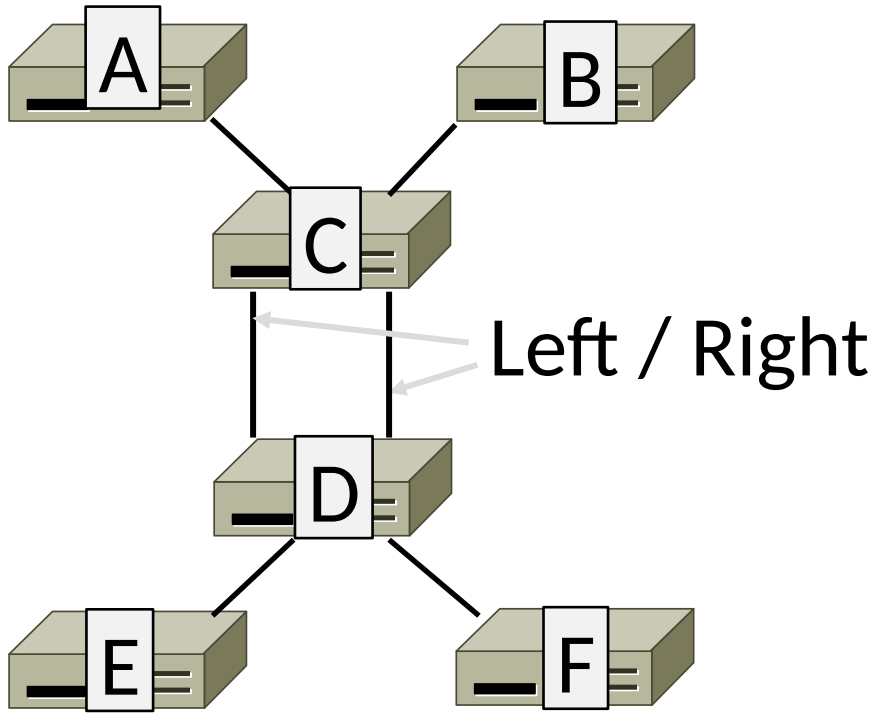
Forwarding Loops (2)

- Suppose the network is started and A sends to F. What happens?



Forwarding Loops (3)

- Suppose the network is started and A sends to F. What happens?
  - A → C → B, D-left, D-right
  - D-left → C-right, E, F
  - D-right → C-left, E, F
  - C-right → D-left, A, B
  - C-left → D-right, A, B
  - D-left → …
  - D-right → …

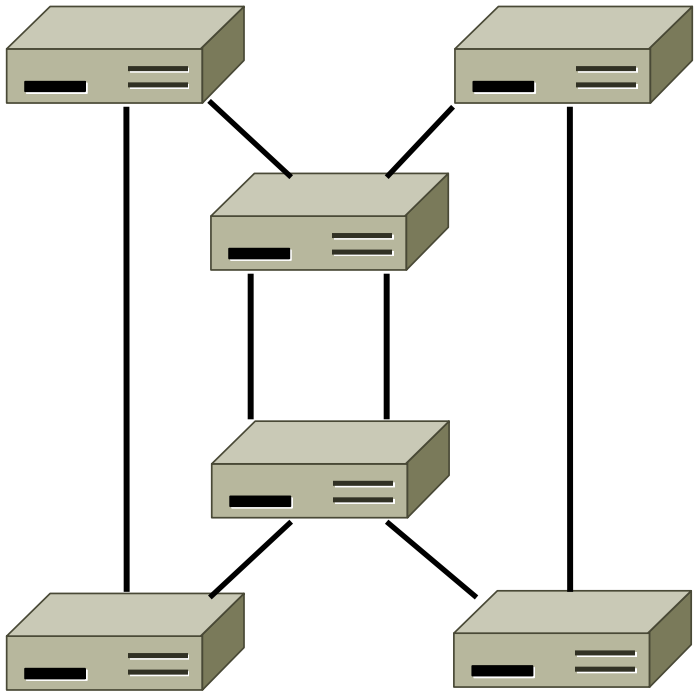


Spanning Tree Solution

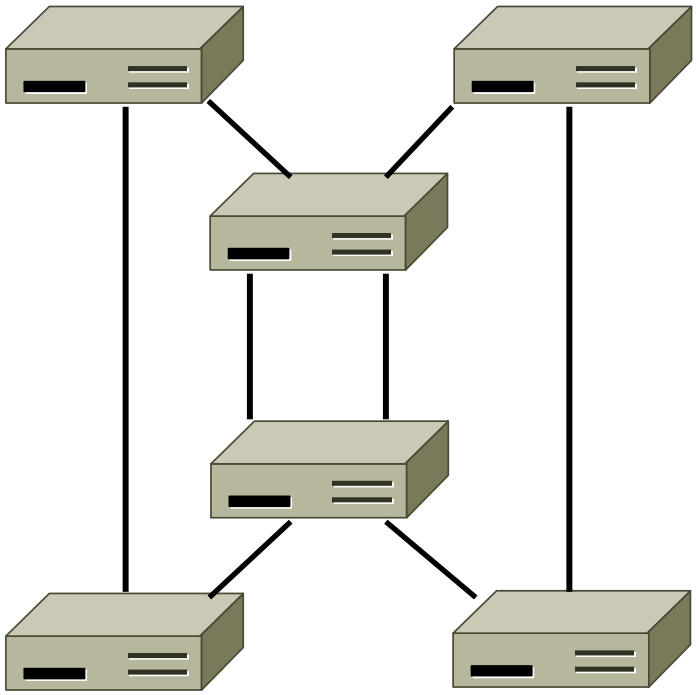
- Switches collectively find a spanning tree for the topology
  - A subset of links that is a tree (no loops) and reaches all switches
  - They switches forward as normal on the spanning tree
  - Broadcasts will go up to the root of the tree and down all the branches

Spanning Tree (2)

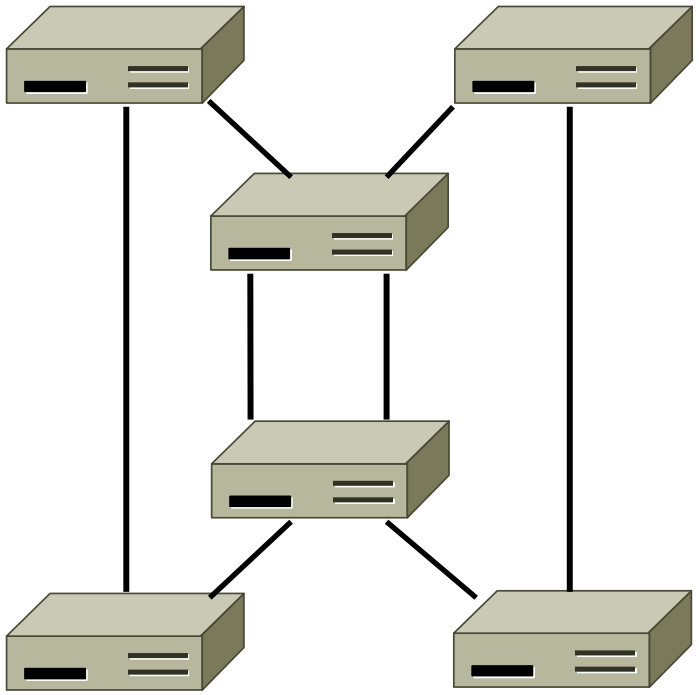
Topology



One ST

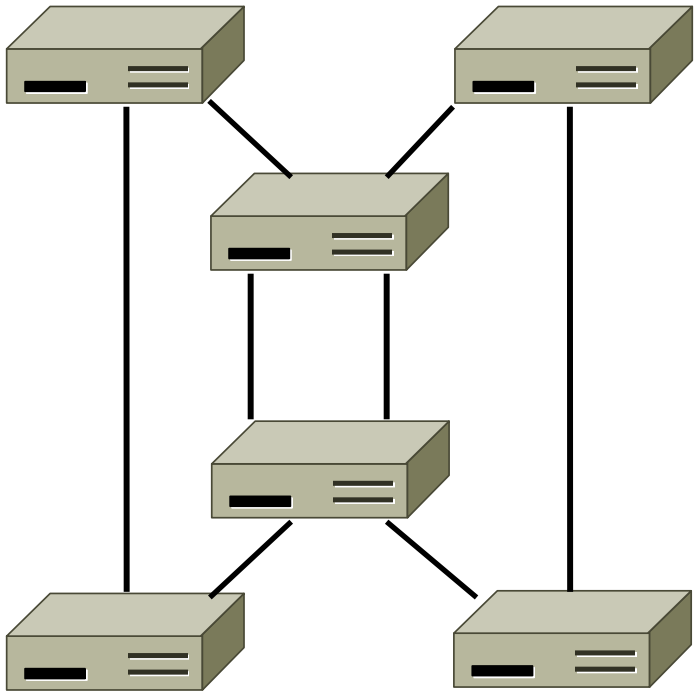


Another ST

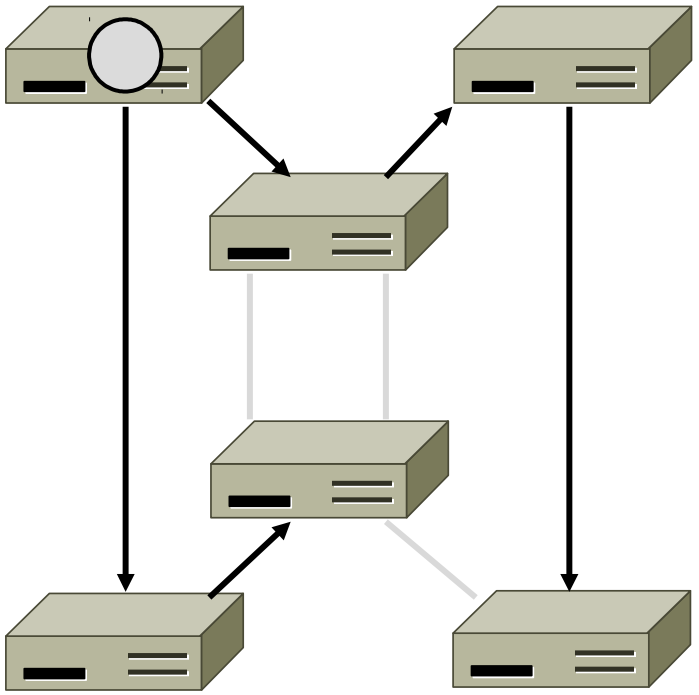


Spanning Tree (3)

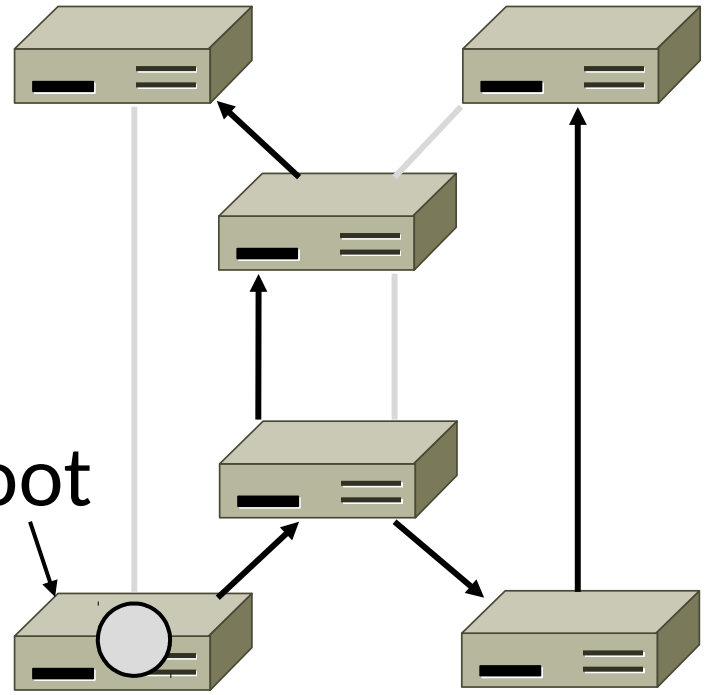
Topology



One ST



Another ST



Spanning Tree Algorithm

- Rules of the distributed game:
  - All switches run the same algorithm
  - They start with no information
  - Operate in parallel and send messages
  - Always search for the best solution

- Ensures a highly robust solution
  - Any topology, with no configuration
  - Adapts to link/switch failures, …

Radia Perlman (1952–)

- Key early work on routing protocols
  - Routing in the ARPANET
  - Spanning Tree for switches (next)
  - Link-state routing (later)
  - Worked at Digital Equipment Corp (DEC)

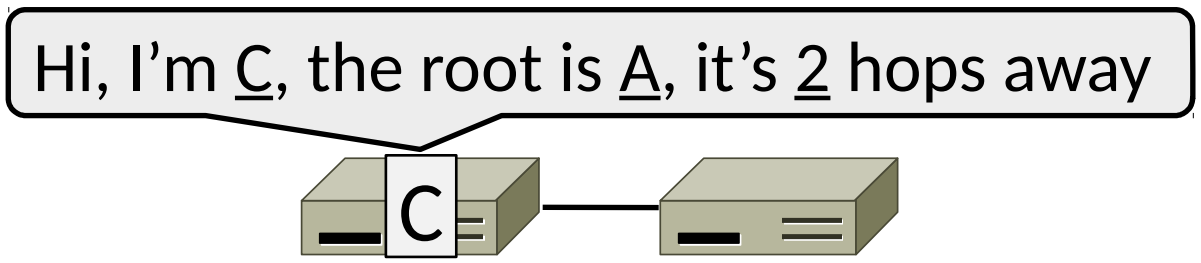
- Now focused on network security

Spanning Tree Algorithm (2)

- Outline:
  1. Elect a root node of the tree (switch with the lowest address)
  2. Grow tree as shortest distances from the root (using lowest address to break distance ties)
  3. Turn off ports for forwarding if they aren't on the spanning tree

Spanning Tree Algorithm (3)

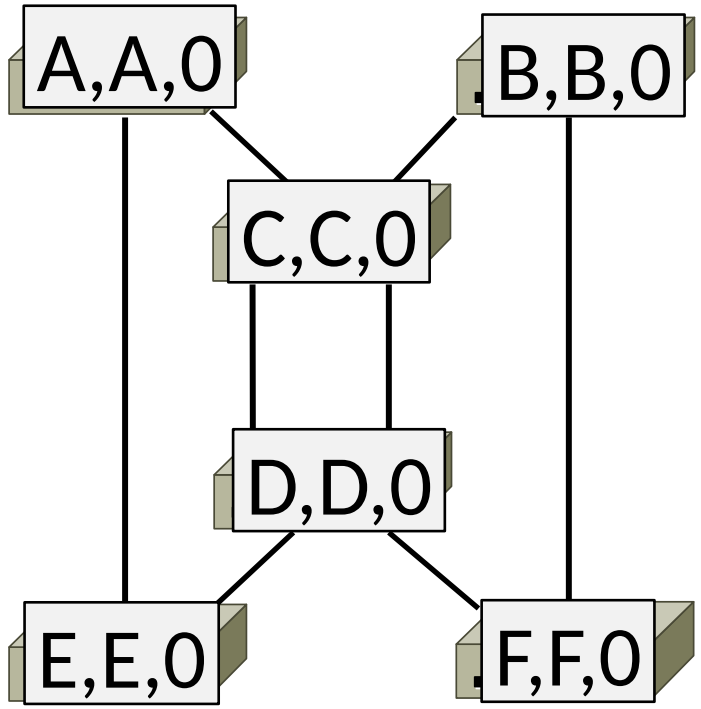
- Details:
  - Each switch initially believes it is the root of the tree
  - Each switch sends periodic updates to neighbors with:
    - Its address, address of the root, and distance (in hops) to root
    - Short-circuit when topology changes
  - Switches favors ports with shorter distances to lowest root
    - Uses lowest address as a tie for distances



or (C, A, 2)

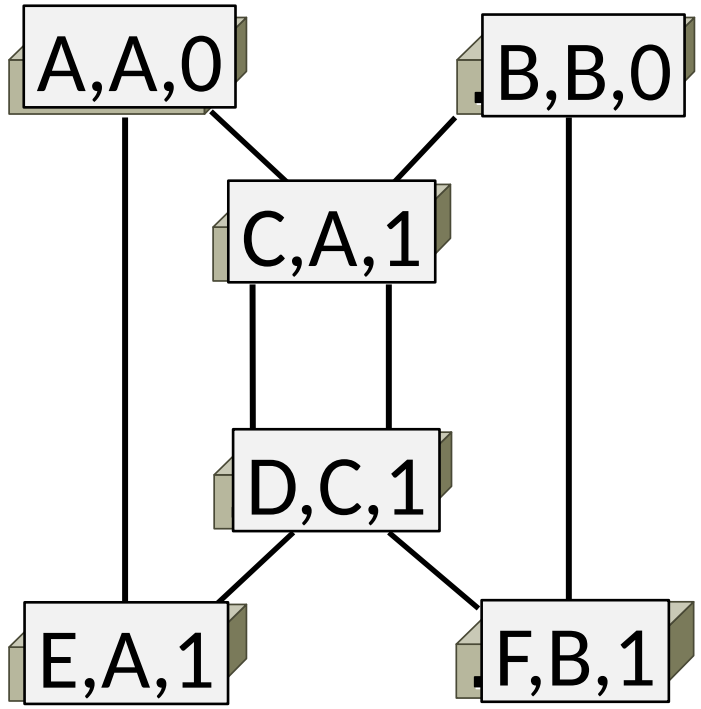
Spanning Tree Example

- 1st round, sending:
  - A sends (A, A, 0) to say it is root
  - B, C, D, E, and F do likewise
- 1st round, receiving:
  - A still thinks is it (A, A, 0)
  - B still thinks (B, B, 0)
  - C updates to (C, A, 1)
  - D updates to (D, C, 1)
  - E updates to (E, A, 1)
  - F updates to (F, B, 1)



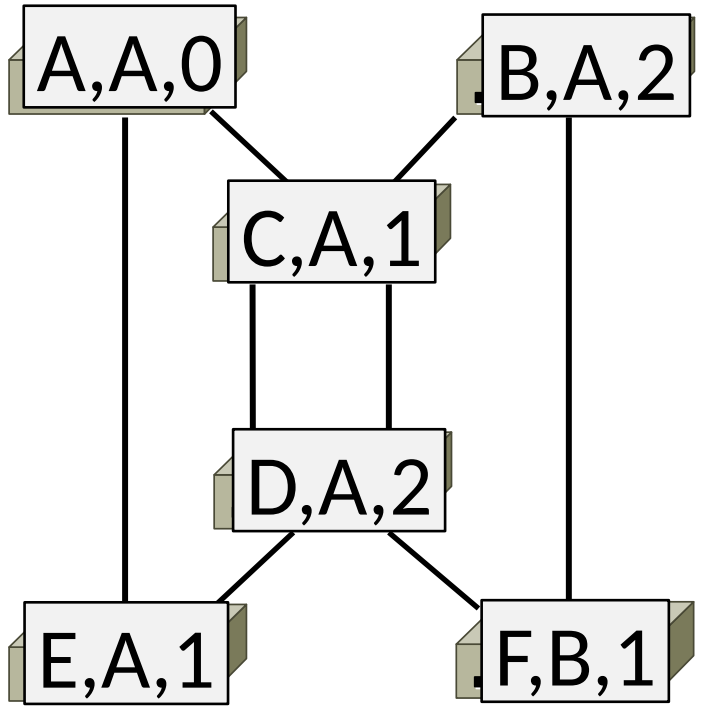
Spanning Tree Example (2)

- 2nd round, sending
  - Nodes send their updated state
- 2nd round receiving:
  - A remains (A, A, 0)
  - B updates to (B, A, 2) via C
  - C remains (C, A, 1)
  - D updates to (D, A, 2) via C
  - E remains (E, A, 1)
  - F remains (F, B, 1)



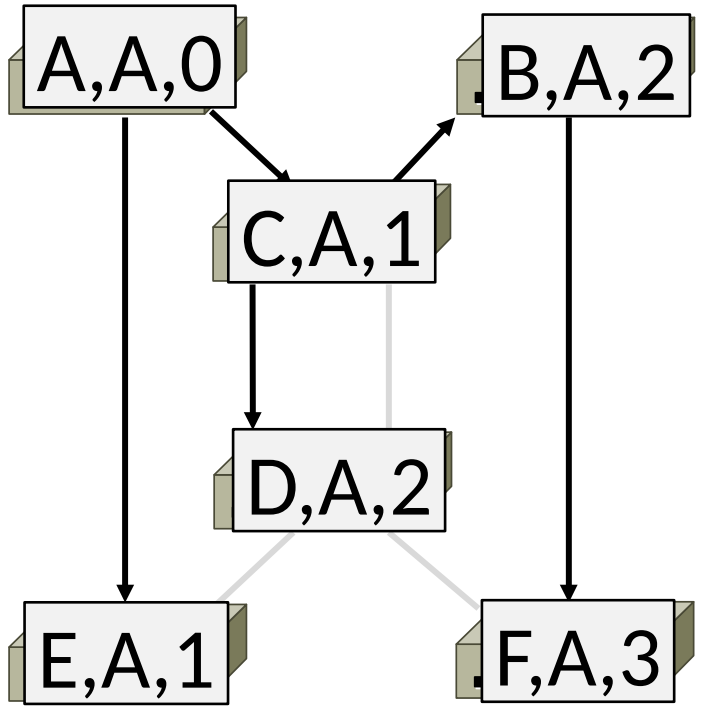
Spanning Tree Example (3)

- 3rd round, sending
  - Nodes send their updated state
- 3rd round receiving:
  - A remains (A, A, 0)
  - B remains (B, A, 2) via C
  - C remains (C, A, 1)
  - D remains (D, A, 2) via C-left
  - E remains (E, A, 1)
  - F updates to (F, A, 3) via B



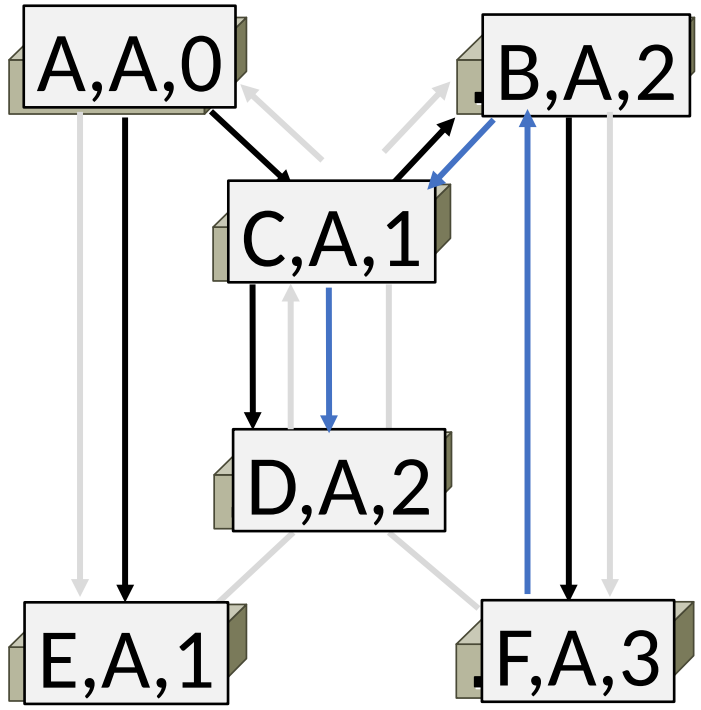
Spanning Tree Example (4)

- 4th round
  - Steady-state has been reached
  - Nodes turn off forwarding that is not on the spanning tree
- Algorithm continues to run
  - Adapts by timing out information
  - E.g., if A fails, other nodes forget it, and B will become the new root



Spanning Tree Example (5)

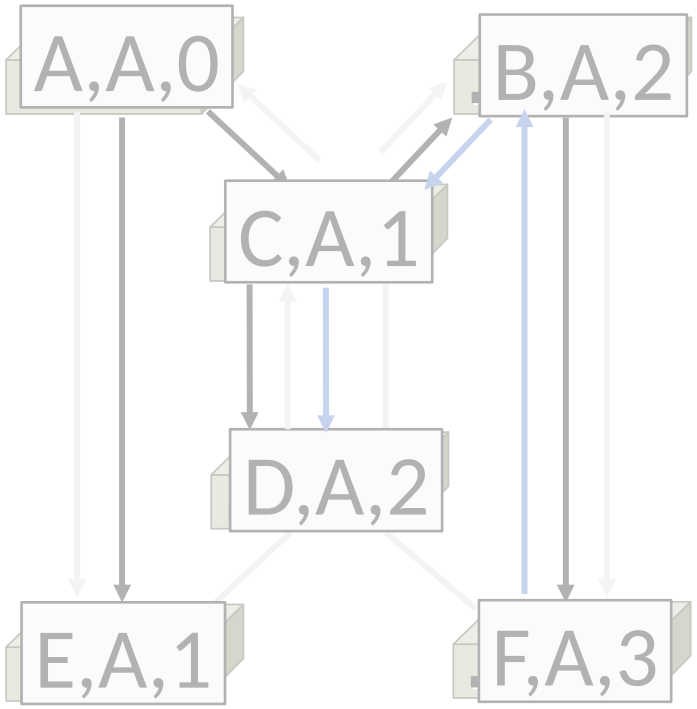
- Forwarding proceeds as usual on the ST
- Initially D sends to F:
  - D → C-left
  - C → A, B
  - A → E
  - B → F
- And F sends back to D:
  - F → B
  - B → C
  - C → D



Spanning Tree Example (6)

- Forwarding proceeds as usual on the ST
- Initially D sends to F:
  - D → C-left
  - C → A, B
  - A → E
  - B → F
- And F sends back to D:
  - F → B
  - B → C
  - C → D

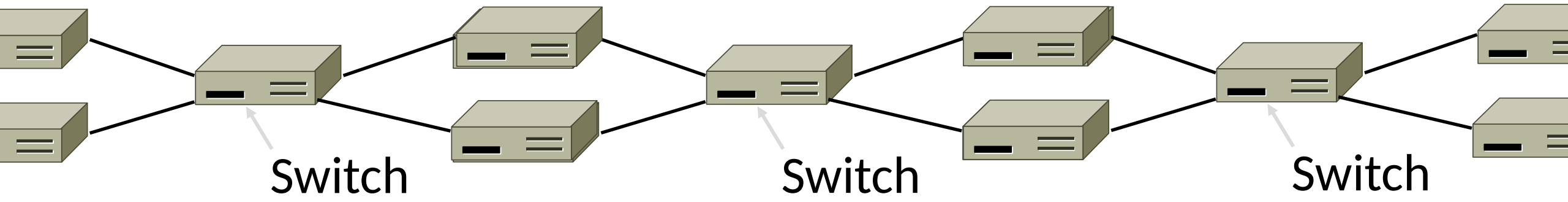
Problems?



Link Layer: Software Defined Networking

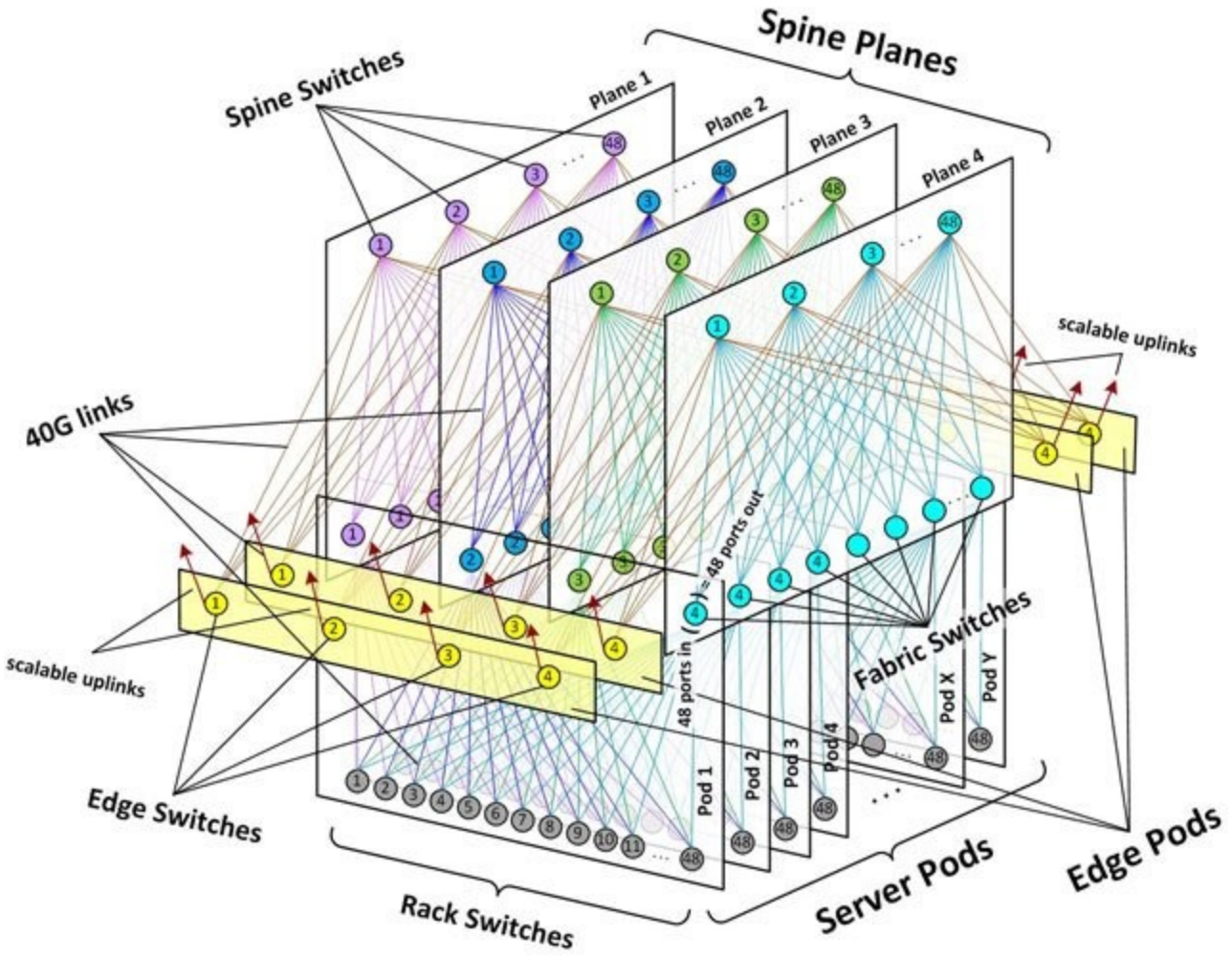
Topic

- How do we scale these networks up?
  - Answer 1: Network of networks, a.k.a. The Internet
  - Answer 2: Ah, just kinda hope spanning tree works?



Rise of the Datacenter

Datacenter Networking



Scaling the Link Layer

- Fundamentally, it's hard to scale distributed algorithms
  - Exacerbated when failures become common
  - Nodes go down, gotta run spanning tree again…
    - If nodes go down faster than spanning tree resolves, we get race conditions
    - If they don't, we may still be losing paths and wasting resources

- Ideas?

Software Defined Networking (SDN)

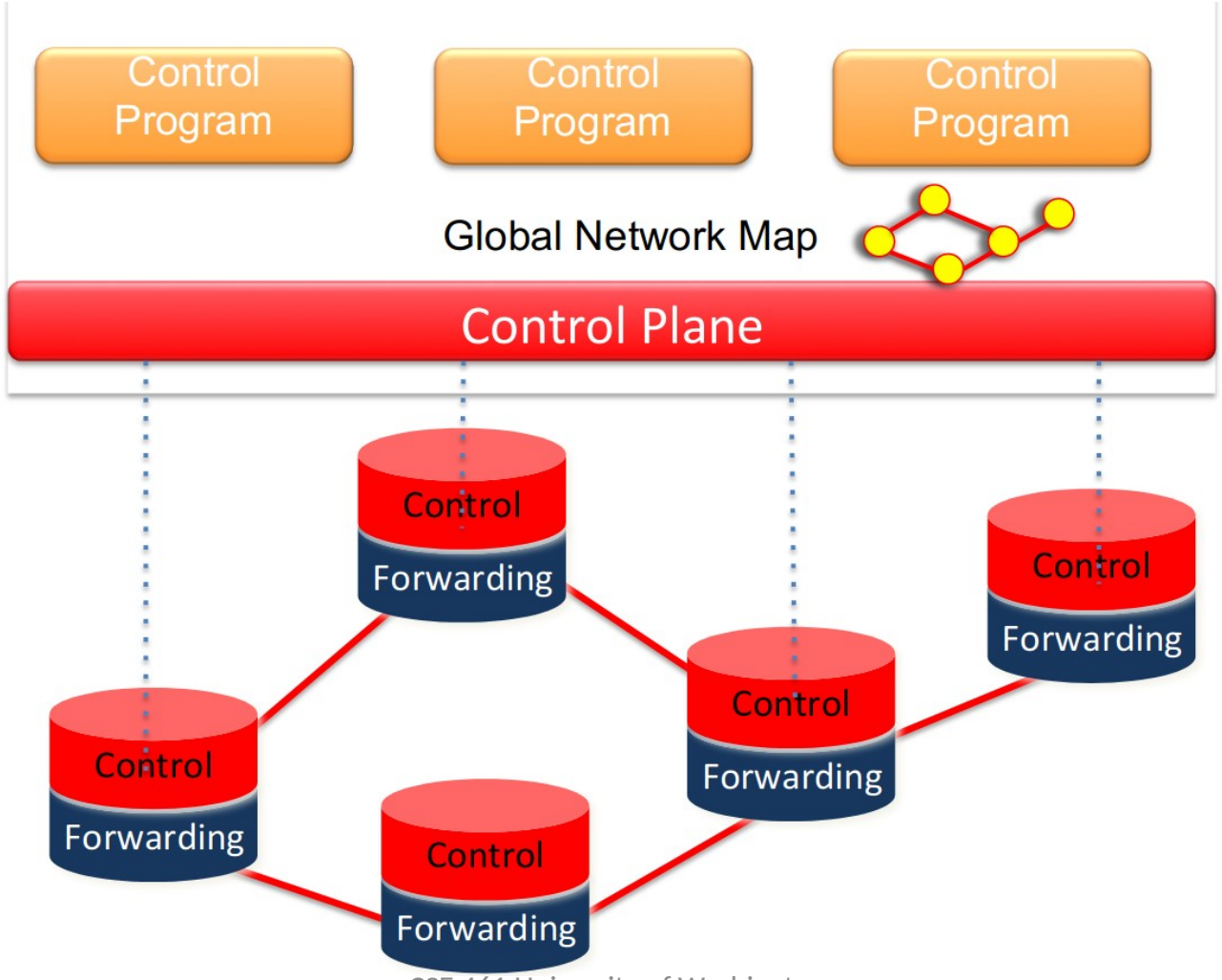
- Core idea: **stop being a distributed system**
  - Centralize the operation of the network
    - Create a “controller” that manages the network
  - Push new code, state, and configuration from “controller” to switches
    - Run link state with a global view of the network rather than in a distributed fashion.
    - Allows for “global” policies to be enforced.
    - Can resolve failures in more robust, faster manners
    - **Problems?**

SDN – Problem 1

- Problem: How do we talk to the switches if there's no network?
  - Seems a little chicken-and-egg
  - Nodes go down, gotta run spanning tree again…
    - If nodes go down faster than spanning tree resolves, we get race conditions
    - If they don't, we may still be losing paths and wasting resources

- Ideas?

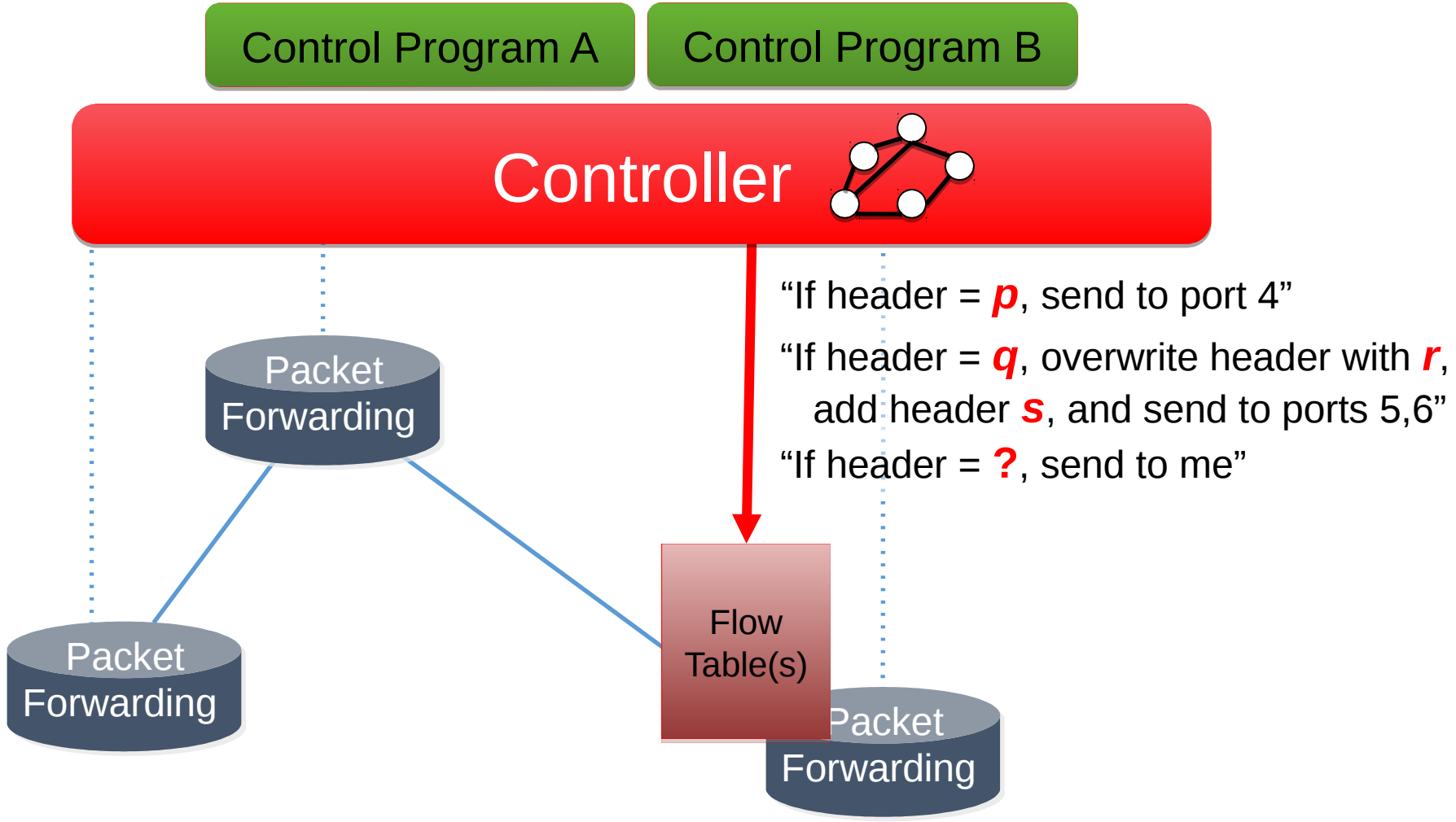
SDN – Control and Data Planes



SDN – Problem 2

- Problem: How do we efficiently run algorithms on switches?
  - These are extremely time-sensitive boxes
    - Gotta move the packets!
    - Need to be able to support
      - Fast packet handling
      - Quick route changes
      - Long-term policy updates
- Ideas?

SDN – OpenFlow



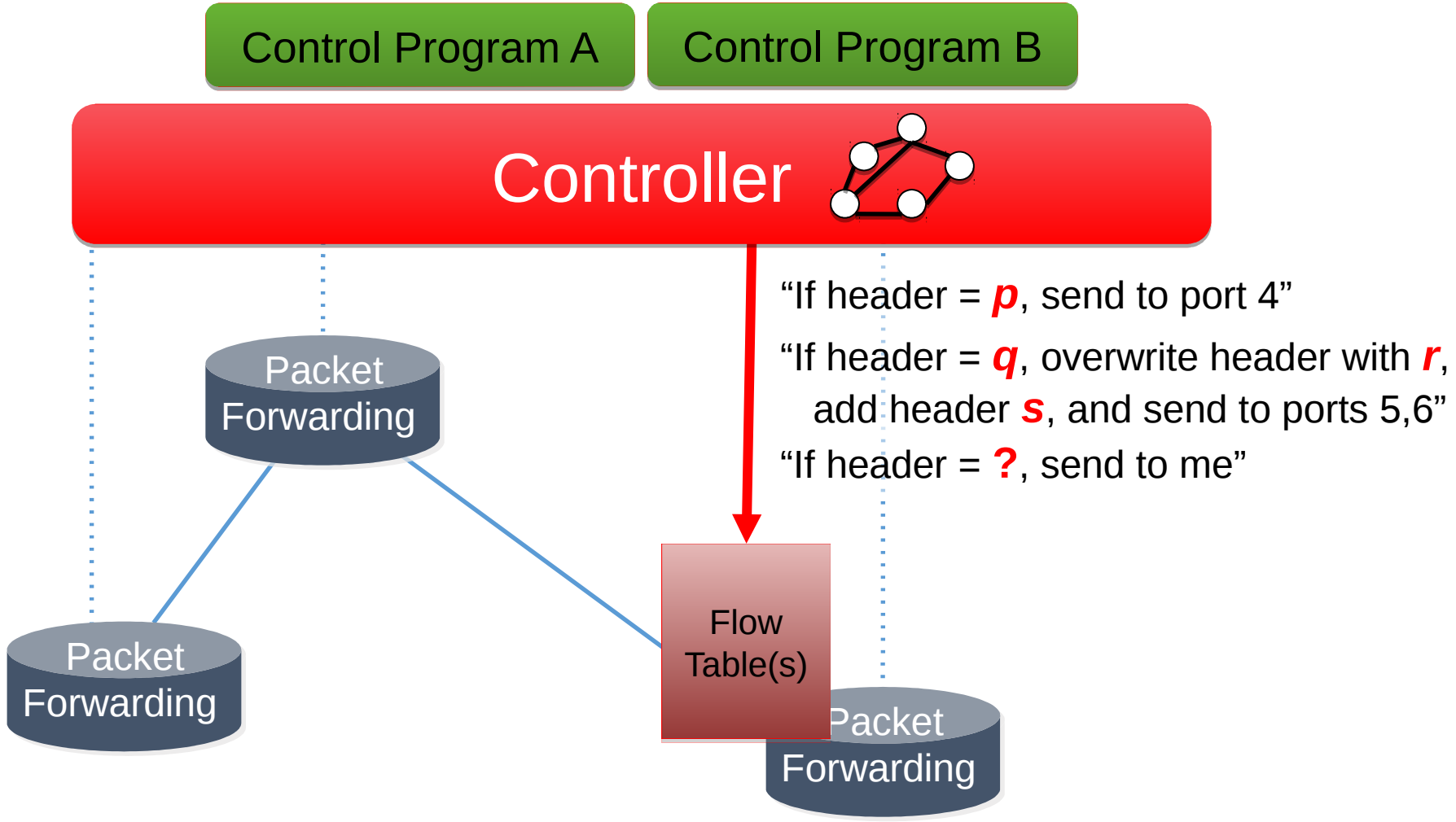
SDN – OpenFlow

- Two different classes of programmability
- At Controller
  - Can be heavy processing algorithms
  - Results in messages that update switch flow table
- At switch
  - Local flow table
  - Built from basic set of networking primitives
  - Allows for fast operation

SDN – Timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Location	Linecard hardware	Router software	Humans or scripts

SDN – OpenFlow



SDN – Key outputs

- Simplify network design and implementation?
  - Sorta. Kinda pushed the complexity around if anything
- However…
  - Does enable code reuse and libraries
  - Does standardize and simplify deployment of rules to switches
  - Allows for fast operation