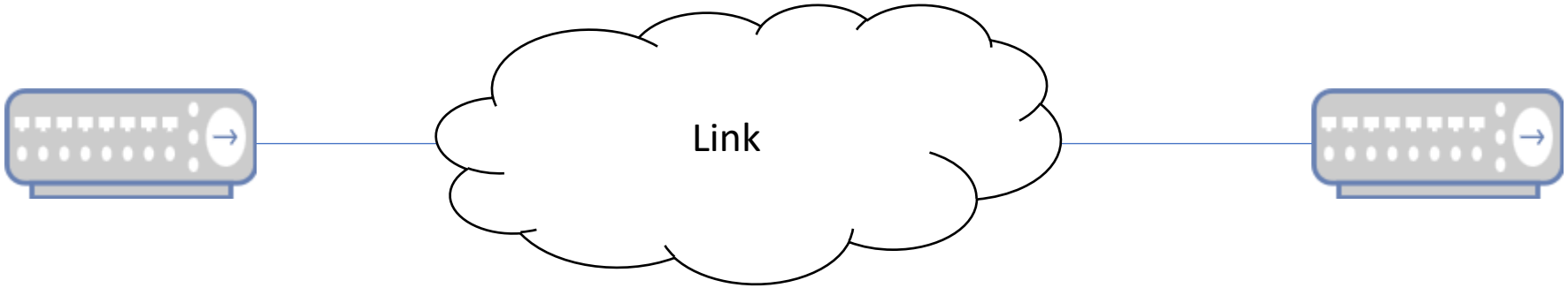
Problem / Overview

Course: Networking Fundamentals
Module: Link

University of Colorado **Boulder**

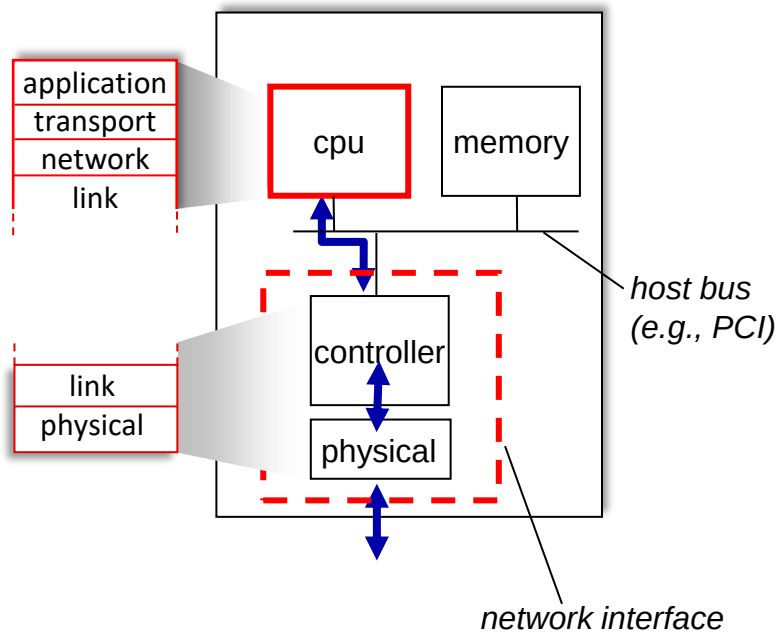
Introducing a Link



Networks consist of links interconnecting nodes (through wires or wirelessly)

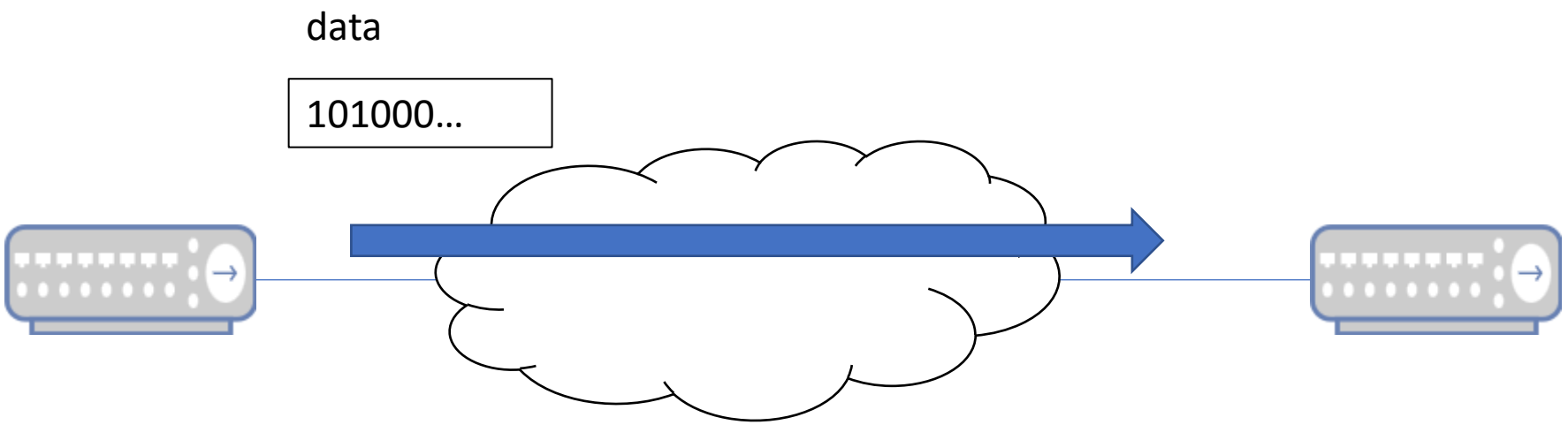
Where is the Link Layer implemented

- Typically, in hardware on network interface card (NIC)



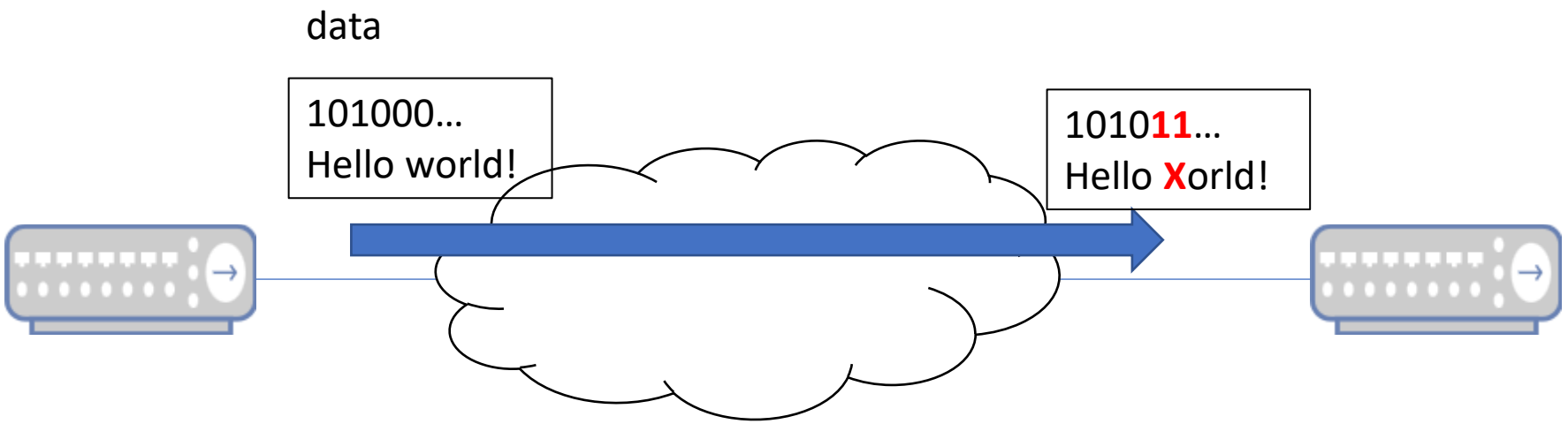
Problems solved by link layer

1. How do we encode the data such that each side can understand who it's destined for, and how to interpret the data.



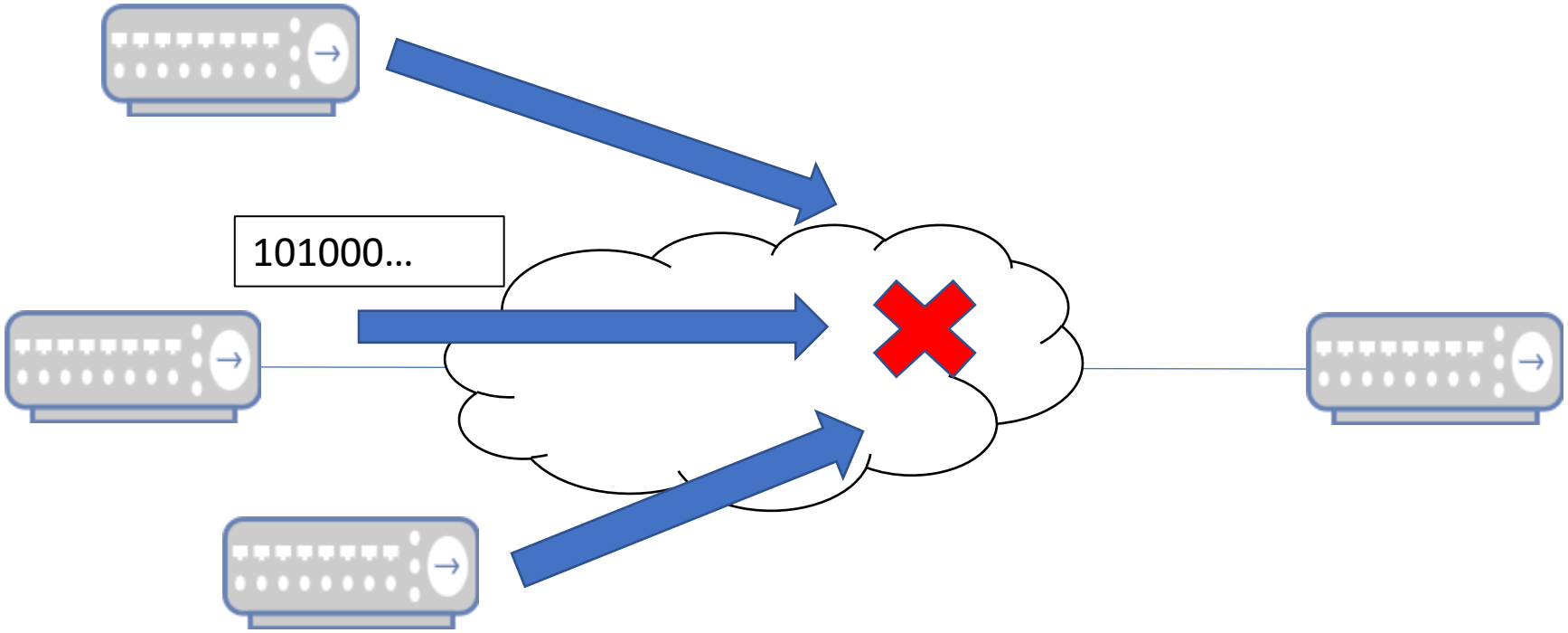
Problems solved by link layer

2. How do we detect if an error has occurred, and better yet, correct it.



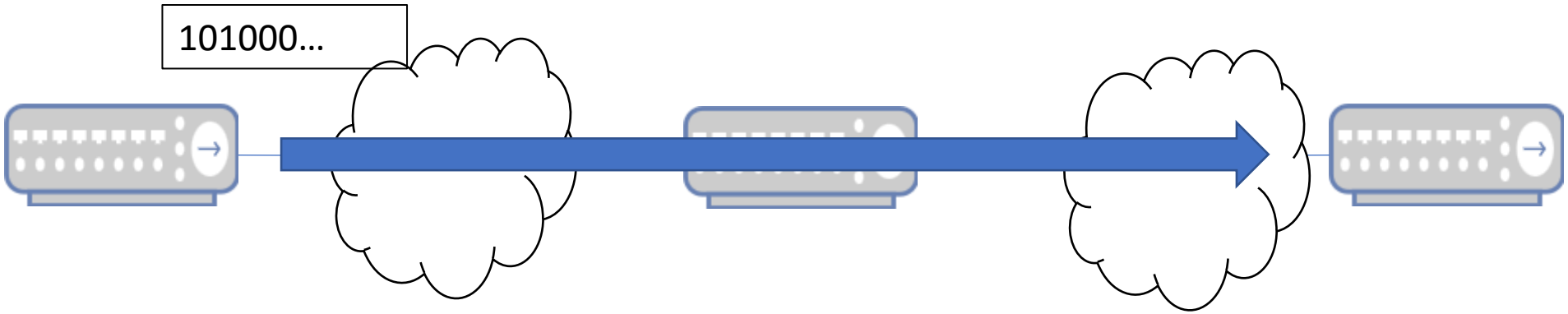
Problems solved by link layer

3. For shared access mediums, how do we coordinate between nodes to regulate access to the medium



Problems solved by link layer

4. How to extend into a local area network (to extend beyond the physical limits of the transmission medium)



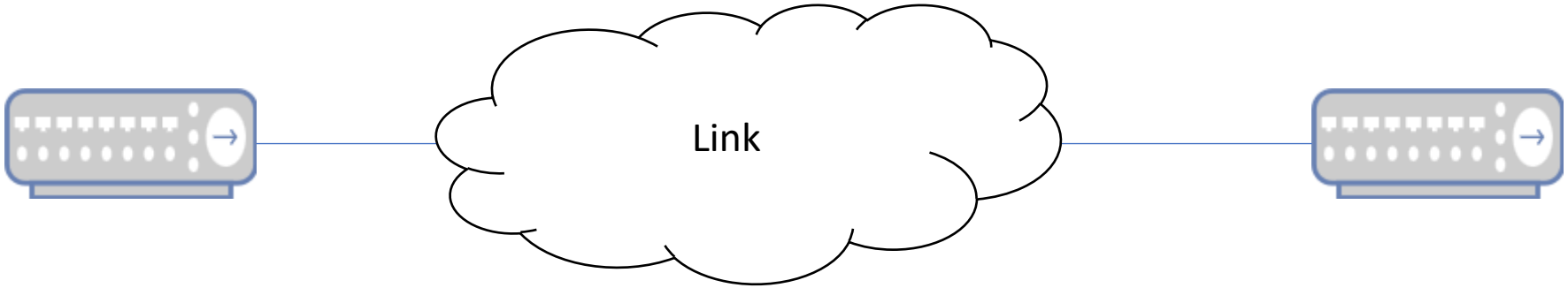
University of Colorado **Boulder**

Encoding into Frames

Course: Networking Fundamentals
Module: Link

University of Colorado **Boulder**

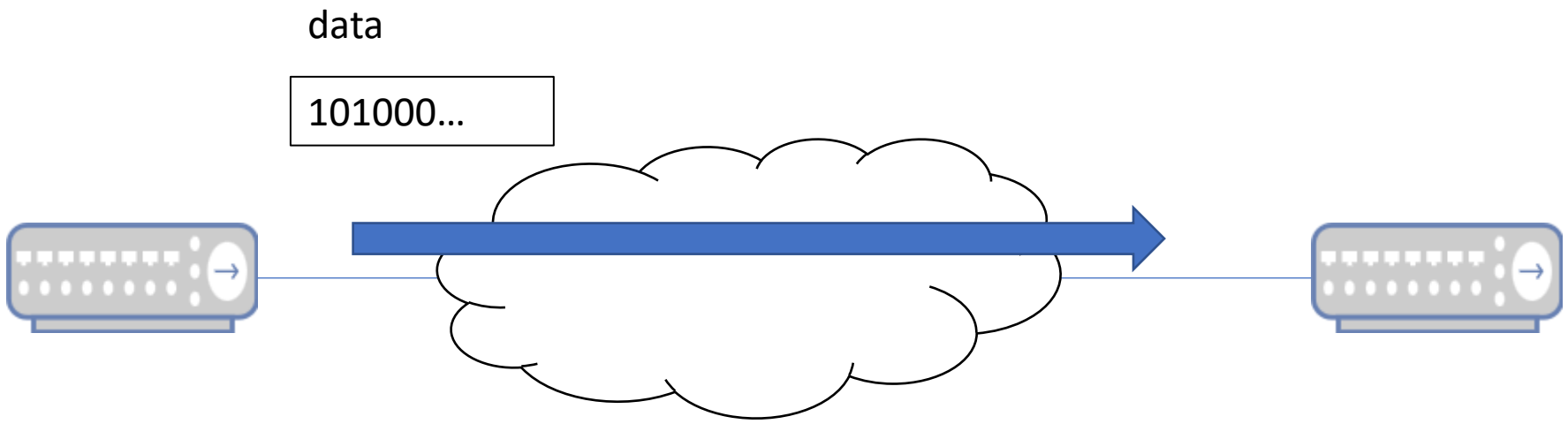
Introducing a Link



Networks consist of links interconnecting nodes (through wires or wirelessly)

Problems solved by link layer

1. How do we encode the data such that each side can understand who it's destined for, and how to interpret the data.



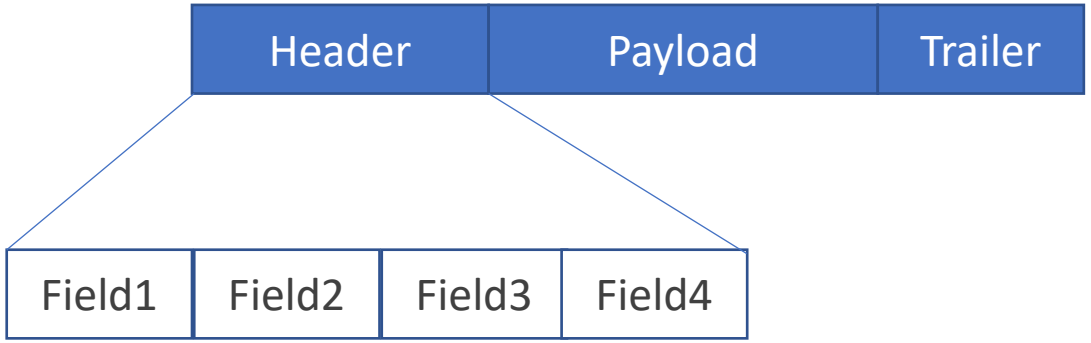
Structuring the data

- Assume you can transmit 1s and 0s through whatever medium
- Making sense of the series of 1s and 0s needs structure

➔ Frame

Header / Trailer / Payload

- The payload is the data that is to be transmitted
- The header and trailer are extra information to help understand how to handle it
- Each is broken down into fields



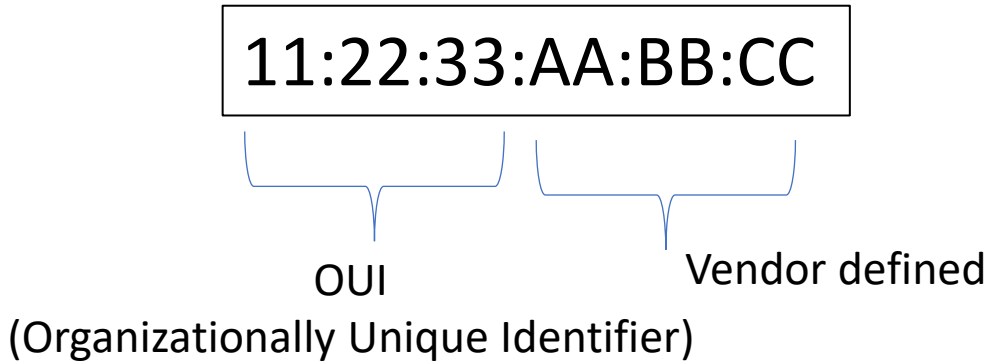
Ethernet

- Dest is for filtering messages (is it for this node)
- Src is useful to know who sent the message (so node can reply)
- Type – tells what the data is
- Frame Check Sequence (FCS) – for error handling

EtherType	Protocol
0x0800	Internet Protocol version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)
0x0842	Wake-on-LAN[8]
0x22F0	Audio Video Transport Protocol (AVTP)
0x22F3	IETF TRILL Protocol
0x22EA	Stream Reservation Protocol
…	

Media Access Control (MAC) address

- 48 bits, commonly displayed in hex with colons
- Historically assigned as part of the manufacturer and globally unique (with virtualization, not the case)



Broadcast

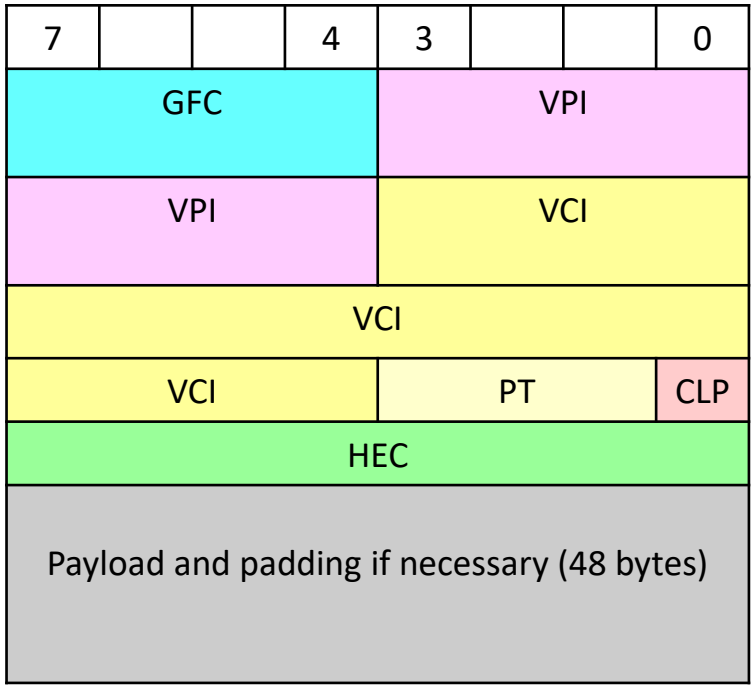
Special destination address which says this is for everybody

FF:FF:FF:FF:FF:FF

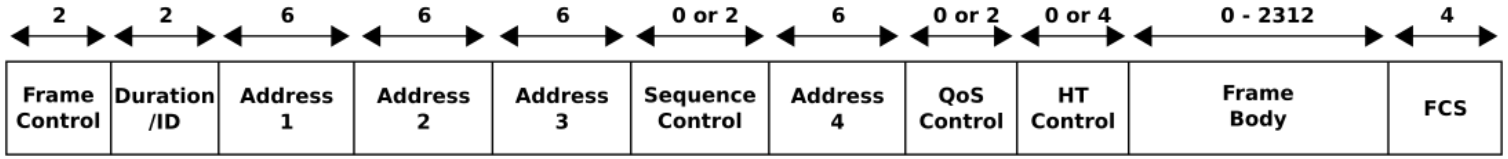
Different Frame Formats

Key take away is that some standard defines the format,
so everyone using it can know how to speak to one another

ATM



802.11



University of Colorado **Boulder**

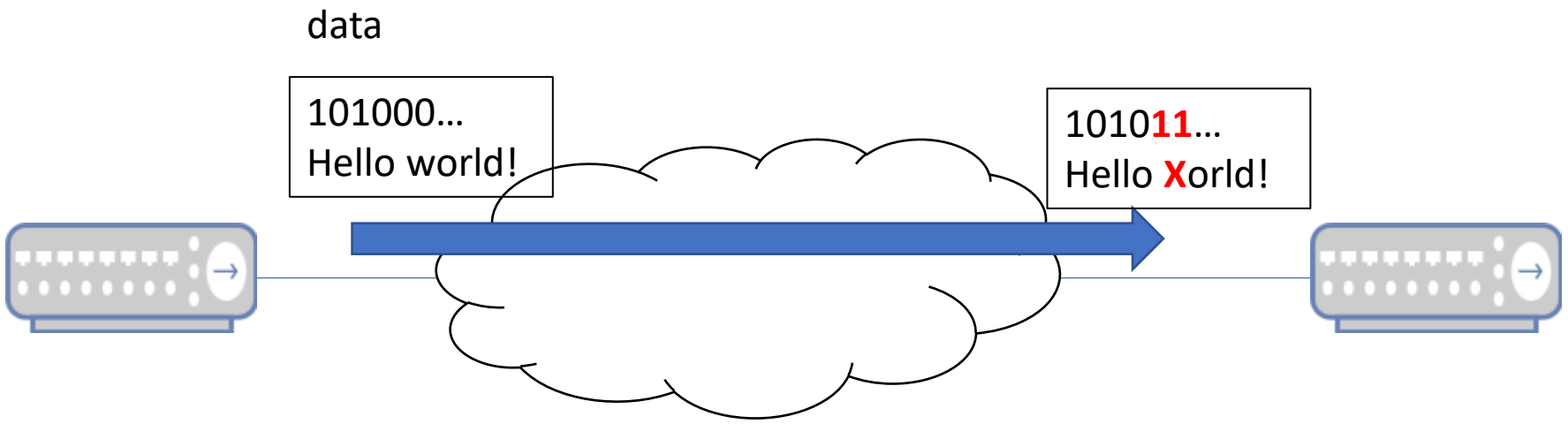
Error Handling

Course: Networking Fundamentals
Module: Link

University of Colorado **Boulder**

Problems solved by link layer

2. How do we detect if an error has occurred, and better yet, correct it.



Detecting an Error With Parity

Send		Receive	
1010 0011	→	1010 **1**011	**ERROR**
1010 0011	→	1010 0011	**CORRECT**

Goal:

Detect if there was an error in transmission

Parity:

Force every transmission to have an even (or odd) number of 1s by adding 1 extra bit set to 0 or 1 based on number of 1s in data

If a single bit gets flipped, it would make the transmission odd (or even)

Aside: xor

	0	1
0	0	1
1	1	0

$$\begin{array}{rl} & 0101 \\ \oplus & 0011 \\ \hline & 0110 \end{array}$$

Counting 1s – use xor

1010 => 1 ⊕ 0 ⊕ 1 ⊕ 0 = 0 (even)
1110 => 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1 (odd)

Parity Example

Even parity (make transmission have an even number of 1s)

Data: 11000000 => $1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$
(it's already even, so add a 0 to the transmission)
Transmit: 110000000

Data: 01010111 => $0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$
(it's odd, so add a 1 to the transmission)
Transmit: 010101111

Parity Example

Received: 110000000

$1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$ CORRECT

Received: 010101111

$0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$ CORRECT

Received: 010101101

$0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus \mathbf{0} \oplus 1 = 1$ ERROR

Cyclic redundancy check (CRC)

CRC is a more powerful form of error detection

Cyclic codes are particularly suited to detect burst errors

=> Adopted for the FCS in Ethernet

CRC Calculation

Given:

D: Data to be transmitted

G: generator polynomial (value defined by Ethernet standard) (r+1 bits)

bit pattern

$<D,R> = D * 2^r$ XOR R — formula for bit pattern

Calculate: Find R (where <D,R> is what will be transmitted) such that:

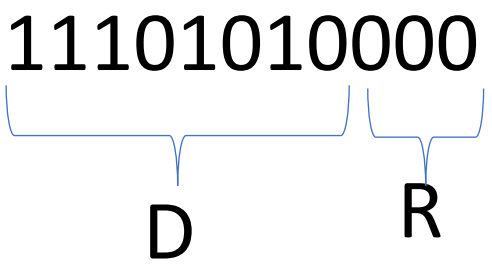
<D,R> is divisible by G (mod 2)

Check: Receiver divides <D,R> by G
If remainder is 0, no error. Otherwise, error

Example CRC calculation

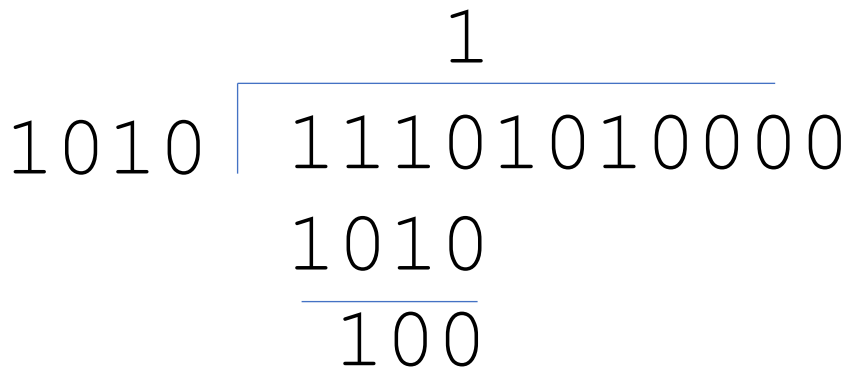
- r = 3
- G = 1010 (r+1=4 bits)
- D = 11101010
- Find: R (3 bits)

Step 1: Set R=0



Setting R = 0 allows us to determine what we need to transmit to make <D,R> exactly divisible by G

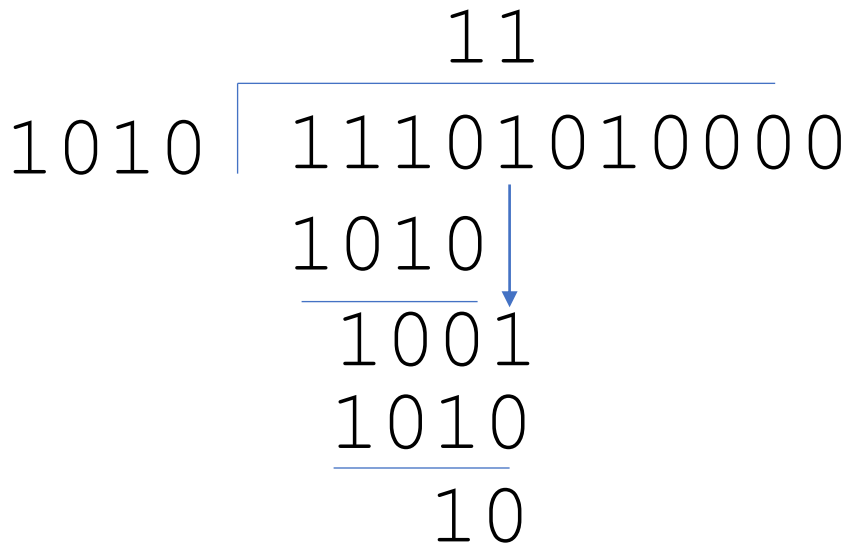
Step 2: Divide by G



Quotient bit = 1 if divisor can “fit”, otherwise 0

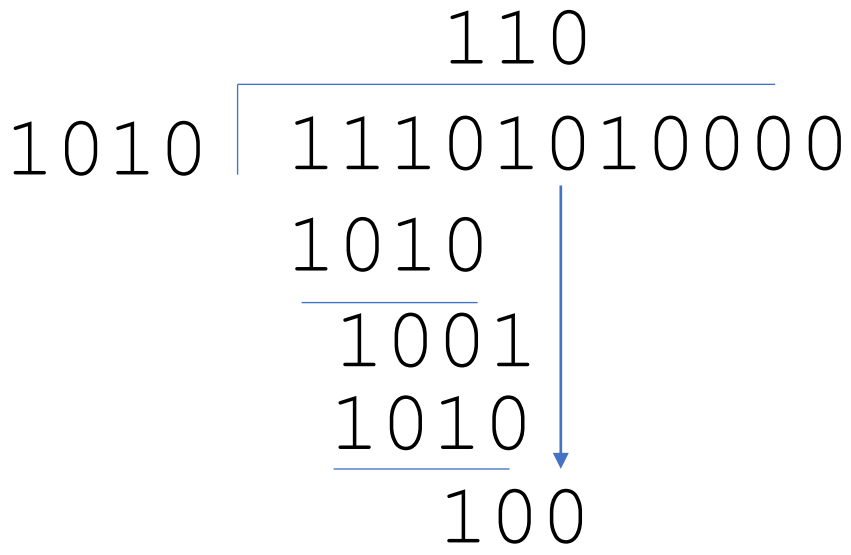
Modulo 2 Division simply involves XOR

Step 2: Divide by G



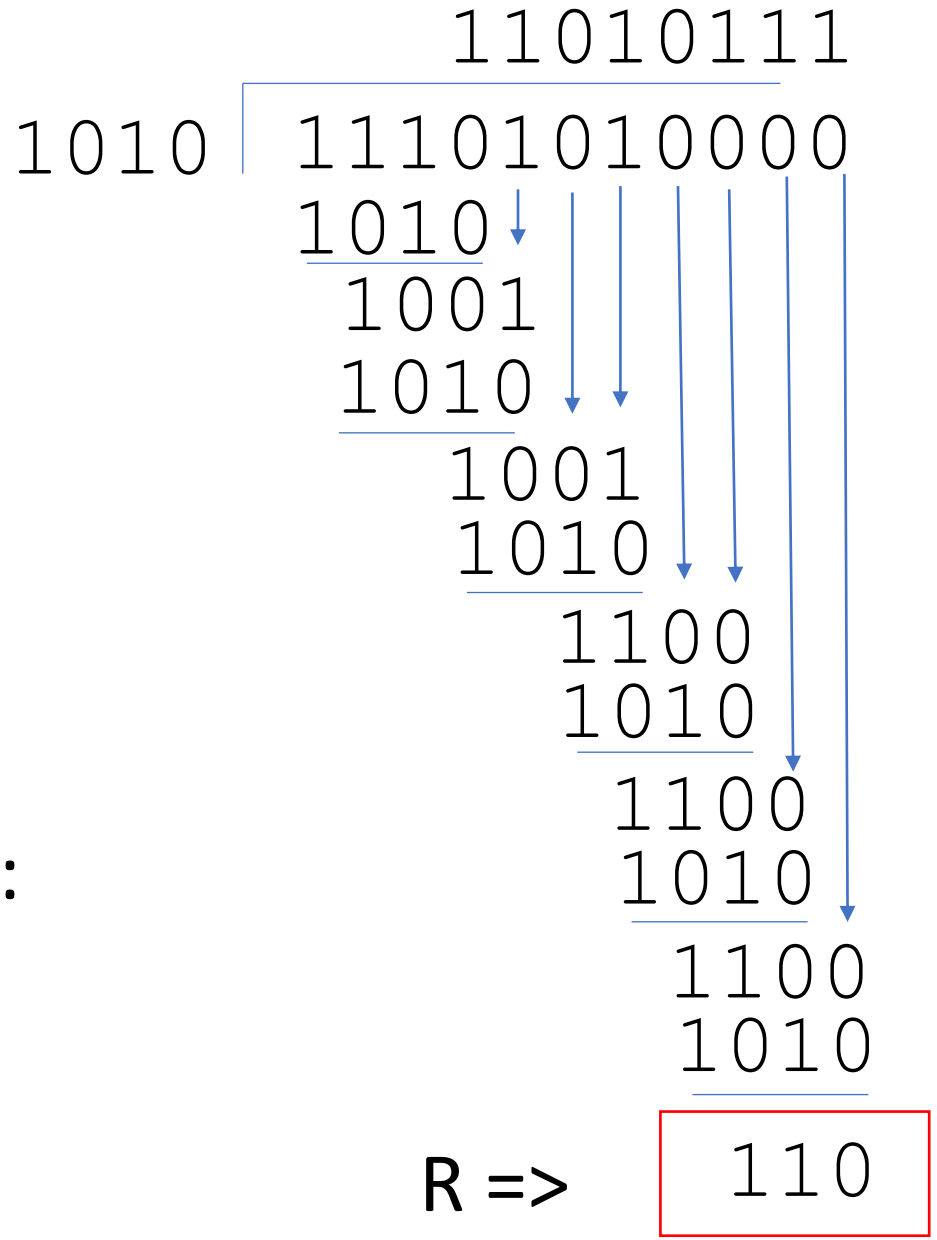
1010 can fit in 1001, so quotient bit is 1

Step 2: Divide by G



1010 can’t fit in 100, so quotient bit is 0

Step 2: Divide by G



R is 110, so would transmit (<D,R>) :

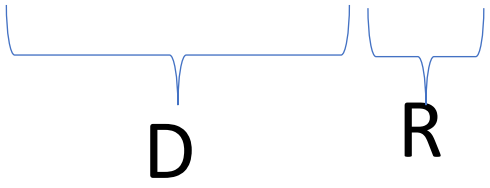
11101010110

D R

Step 3: Transmit

R is 110, so would transmit (<D,R>) :

11101010110



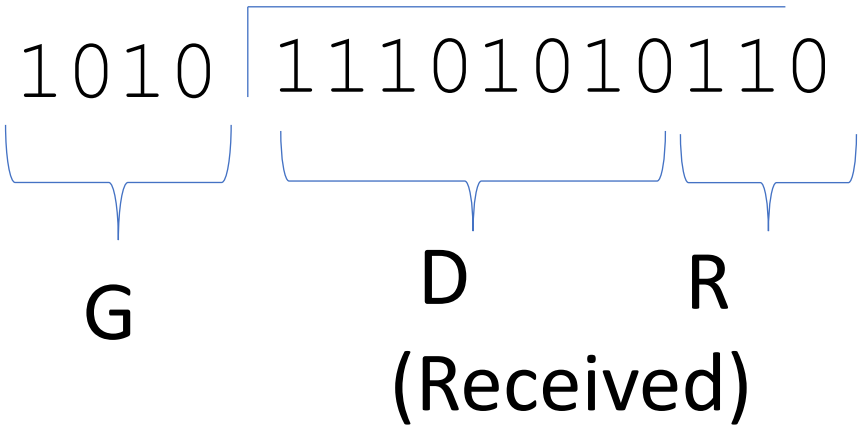
Check on Receive Side

Received <D,R>

Divide by G

If Remainder is 0 => correct

Else => Error



CRC in Ethernet

Detects:

- Any 1 bit error
- Any two adjacent 1 bit errors
- Any odd number of 1 bit errors
- Any burst of errors with a length of 32 or less

Can be calculated as bytes arrive off of the wire

- Simple calculation only involving xor operation

Ethernet FCS

3.2.9 Frame Check Sequence (FCS) field

A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The FCS field contains a 4-octet (32-bit) CRC value. This value is computed as a function of the contents of the protected fields of the MAC frame: the Destination Address, Source Address, Length/ Type field, MAC Client Data, and Pad (that is, all fields except FCS). The encoding is defined by the following generating polynomial.

G = 04C11DB7

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Implementation to deal with case of leading 0s

Mathematically, the CRC value corresponding to a given MAC frame is defined by the following procedure:

a) The first 32 bits of the frame are complemented.

b) The n bits of the protected fields are then considered to be the coefficients of a polynomial $M(x)$ of degree $n - 1$. (The first bit of the Destination Address field corresponds to the $x^{(n-1)}$ term and the last bit of the MAC Client Data field (or Pad field if present) corresponds to the x^0 term.)

D is frame data, M is <D,R> Divide by G and output is R

c) $M(x)$ is multiplied by x^{32} and divided by $G(x)$, producing a remainder $R(x)$ of degree ≤ 31.

d) The coefficients of $R(x)$ are considered to be a 32-bit sequence.

e) The bit sequence is complemented and the result is the CRC.

For further explanation and implementation

Understanding CRC32 by Joshua Davies

https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art008

University of Colorado **Boulder**

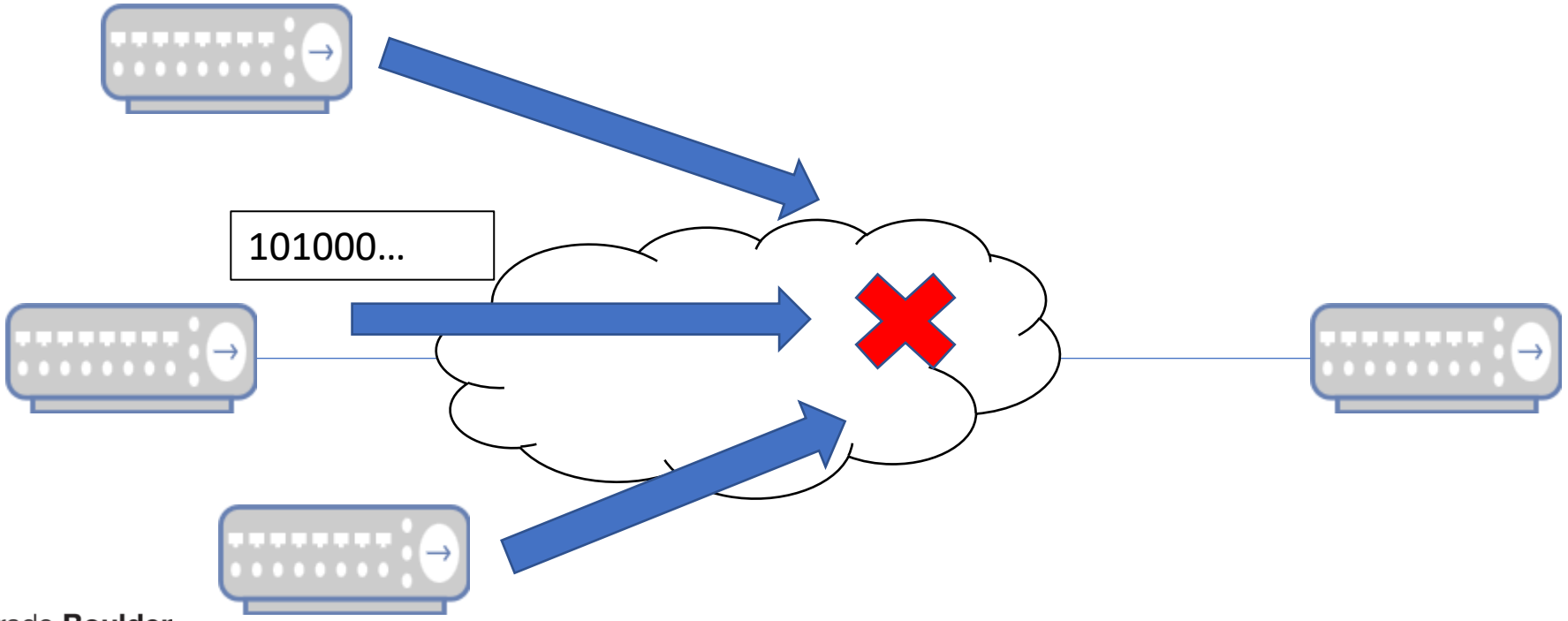
Sharing the Link

Course: Networking Fundamentals
Module: Link

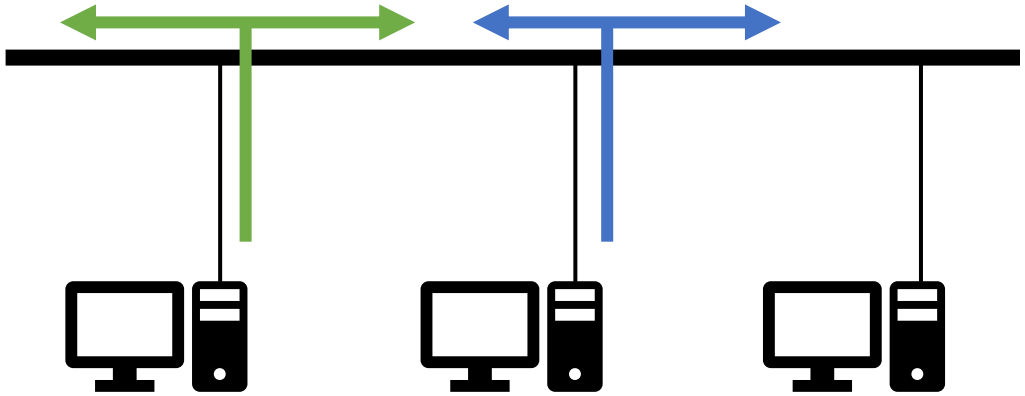
University of Colorado **Boulder**

Problems solved by link layer

3. For shared access mediums, how do we coordinate between nodes to regulate access to the medium



Interference

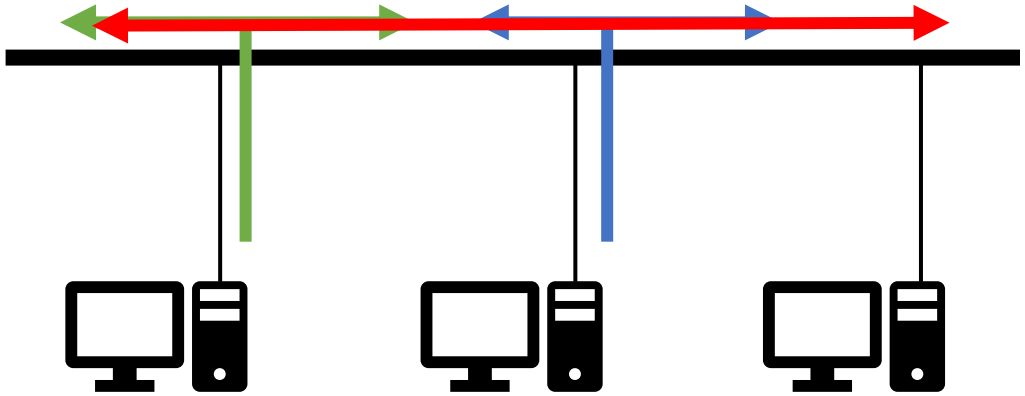


Wired

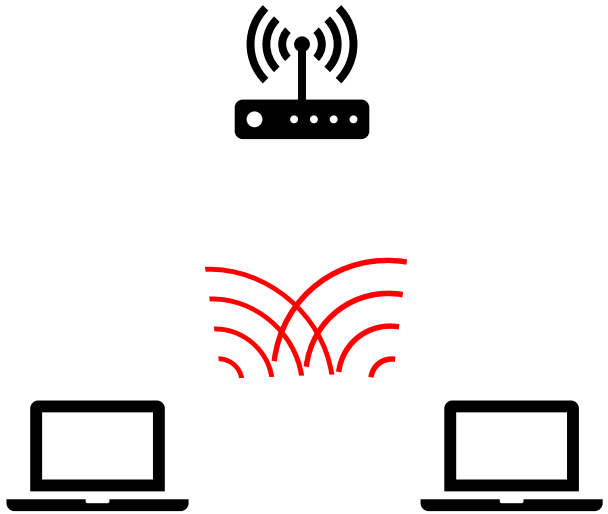
Wireless

Interference

When collisions happen – no data is usefully transmitted



Wired



Wireless

Multiple Access Protocols

How do we share the link fairly and efficiently?

Possible Approaches

Channel Partitioning:

- Divide the link into smaller pieces and assign to each node

Random

- Nodes can transmit in an uncoordinated manner
- Deal with collisions when they happen

Channel Partitioning - Time

Time Division Multiple Access (TDMA)

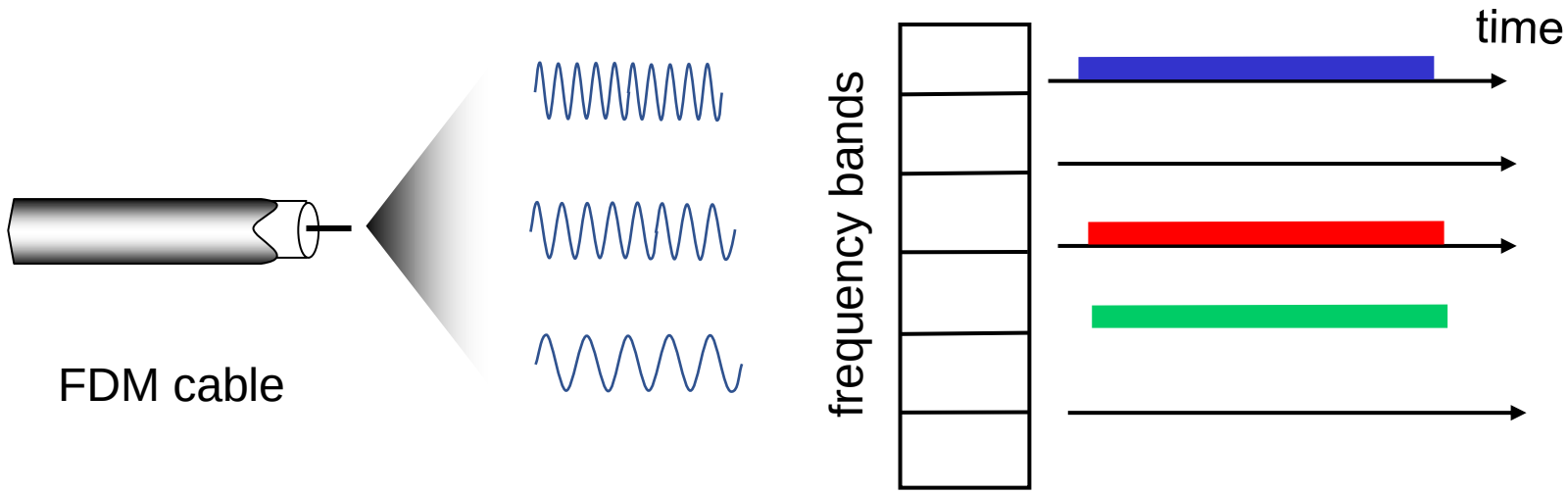
- Divide into equal sized time slots

- Assign slots to nodes

- Nodes can send during their slots only
  - Ex: Slots are 100 bytes. 1 has 10 bytes to send, 2 has 0, 3 has 200, 4 has 150

Partition Channel - Frequency

- Frequency Division Multiple Access (FDMA)
- Divide frequency spectrum and assign each node it's own frequency



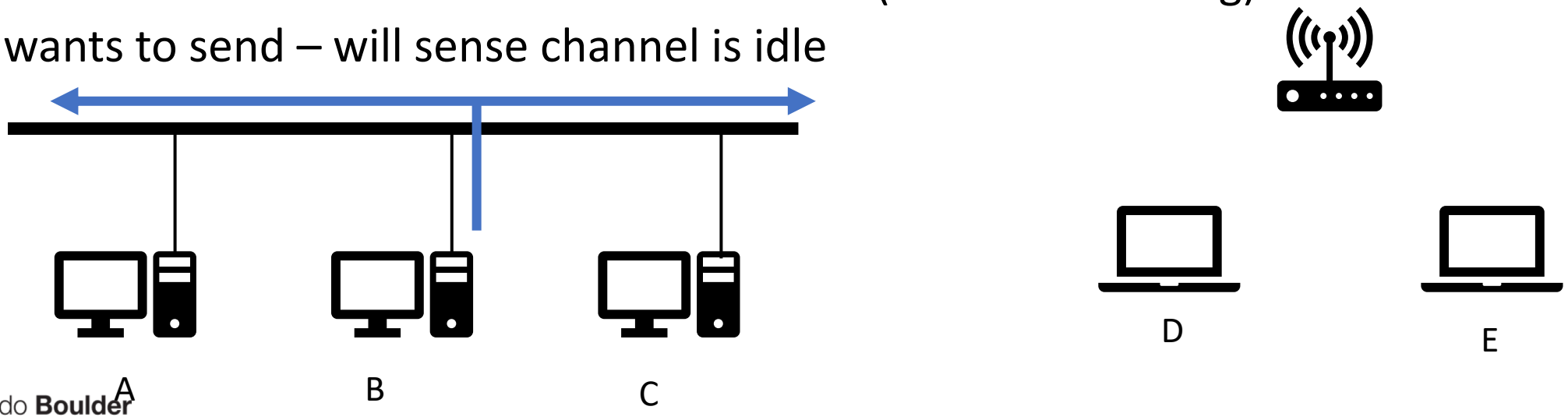
From: https://gaia.cs.umass.edu/kurose_ross

Random Multi Access Protocols

- When to send?
  - Anytime? Make an attempt to at least not interfere?
- How does a node know if transmission was successful?
  - Can a node detect this? Does the receiver have to acknowledge?
- What happens if transmission failed?
  - Resend immediately? Wait for some amount of time?

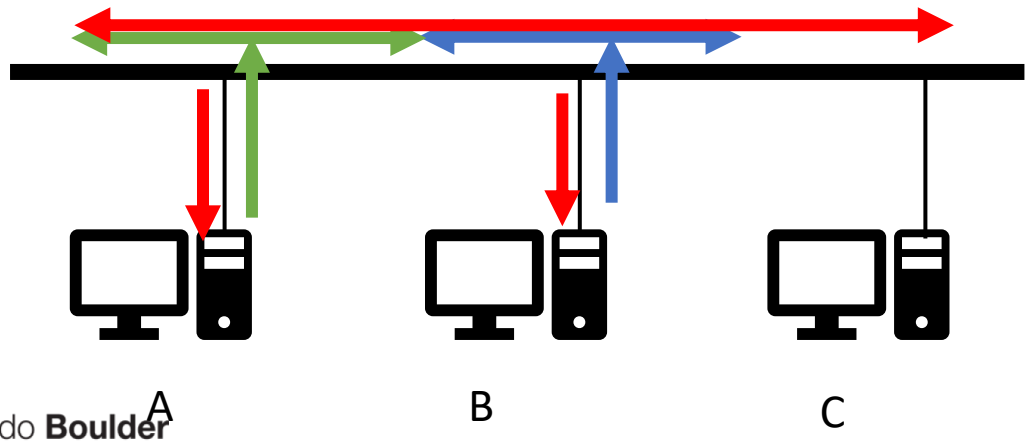
When to Send

- Carrier Sense Multiple Access (CSMA)
  - Used in both Ethernet (802.3) and Wi-Fi (802.11)
- Node will read from the channel to see if another node is sending and transmits when it senses idle
- Example:
  - A wants to send – will sense channel not idle (since B is sending)
  - D wants to send – will sense channel is idle



Knowing if a Transmission was Successful (CSMA / CD - Ethernet)

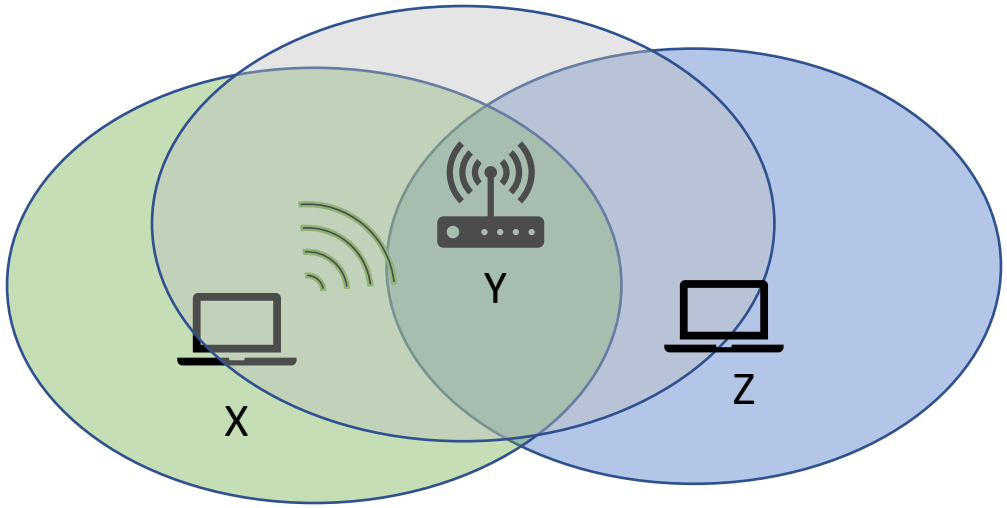
- CSMA / CD – collision detection (802.3)
- Ethernet NICs can read while transmitting
- If what it reads is different than what it's sending, it detects a collision
- Node will send a "jam" signal – transmit fixed bit pattern to ensure everyone detects the collision



Knowing if a Transmission was Successful (CSMA / CA – Wi-Fi)

- CSMA / CA – collision avoidance (802.11)
- 802.11 NICs cannot read while transmitting and may be out of range of some nodes (Z can't sense X is sending)
- It depends on an explicit acknowledgement from receiver
- If it doesn't receive an ACK in certain period of time, assumes collision

- Optional RTS / CTS:
  - Request to send
  - Clear to Send



When a Transmission Fails

Re-transmit the frame some time period (backoff)

CSMA/CA and CSMA/CD specify:

- Random backoff time – to avoid nodes repeatedly colliding
- Exponentially increasing – when multiple transmissions fail, wait longer each time

University of Colorado **Boulder**

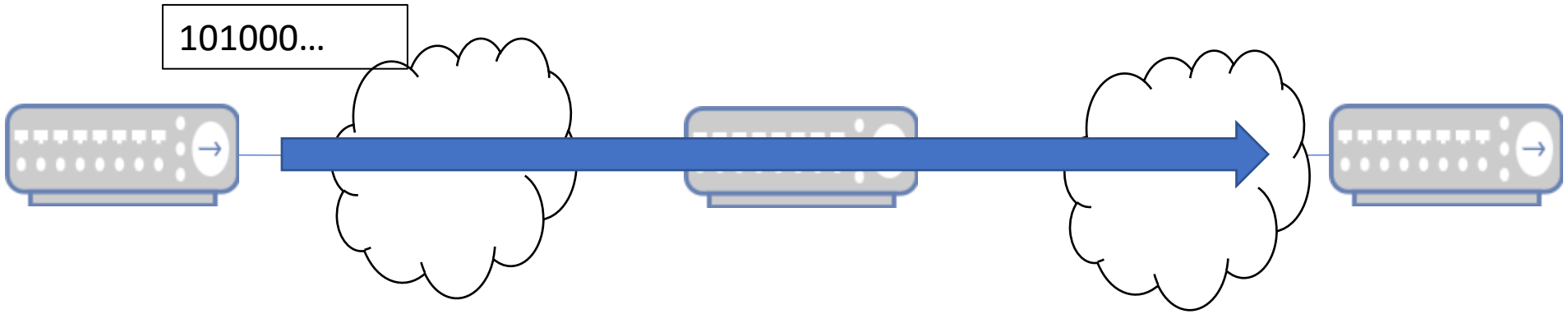
Local Area Network

Course: Networking Fundamentals
Module: Link

University of Colorado **Boulder**

Problems solved by link layer

4. How to extend into a local area network (to extend beyond the physical limits of the transmission medium)

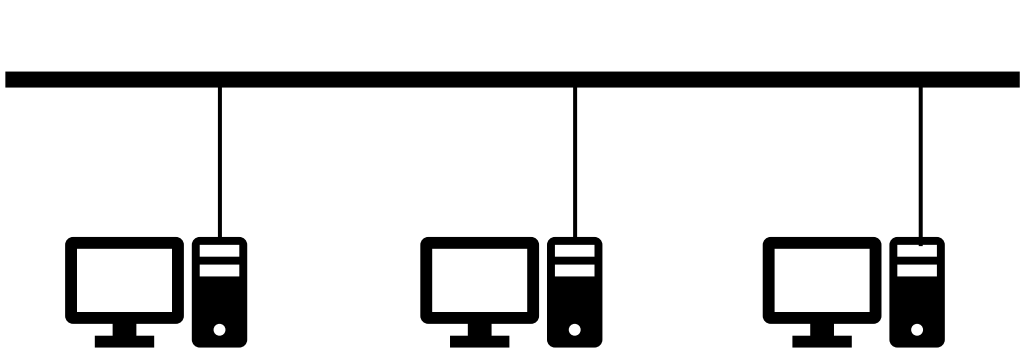


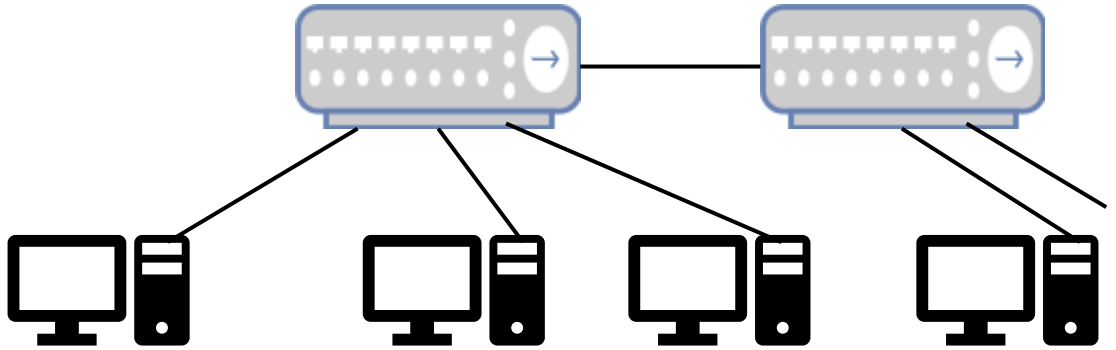
Topics Covered

- Ethernet switching
- Wireless access points

Ethernet Switching

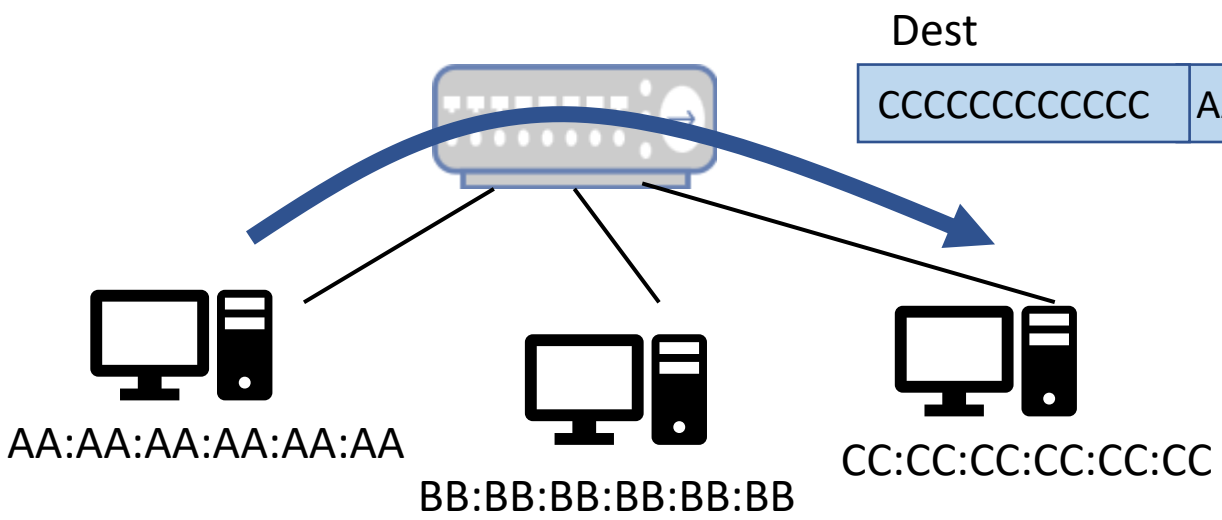
- Scalability challenges with shared cable (left)
- Switching solves this (right)
  - Uses point to point full-duplex links
  - Special nodes (switches) can direct traffic to destination





Ethernet Addressing

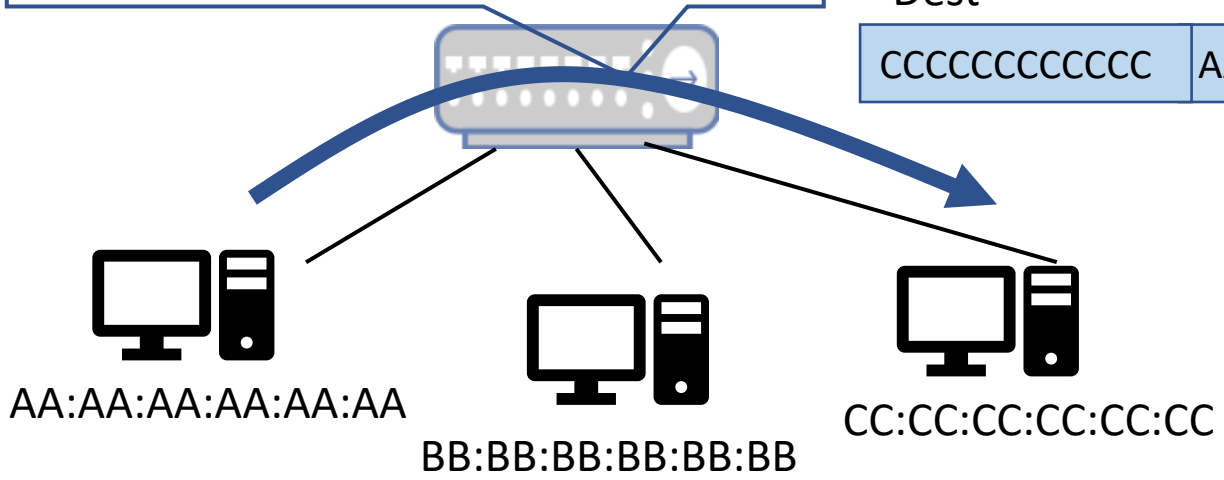
- Recall:
  - Each NIC has a MAC address
  - Each frame contains src and dest MAC address



Dest	Src	Type	Payload	FCS
CCCCCCCCCCCC	AAAAAAAAAAAA	0800	F10FE23…	102F

MAC Address Table

DEST MAC ADDRESS	OUT PORT
AA:AA:AA:AA:AA:AA	0
BB:BB:BB:BB:BB:BB	1
CC:CC:CC:CC:CC:CC	2

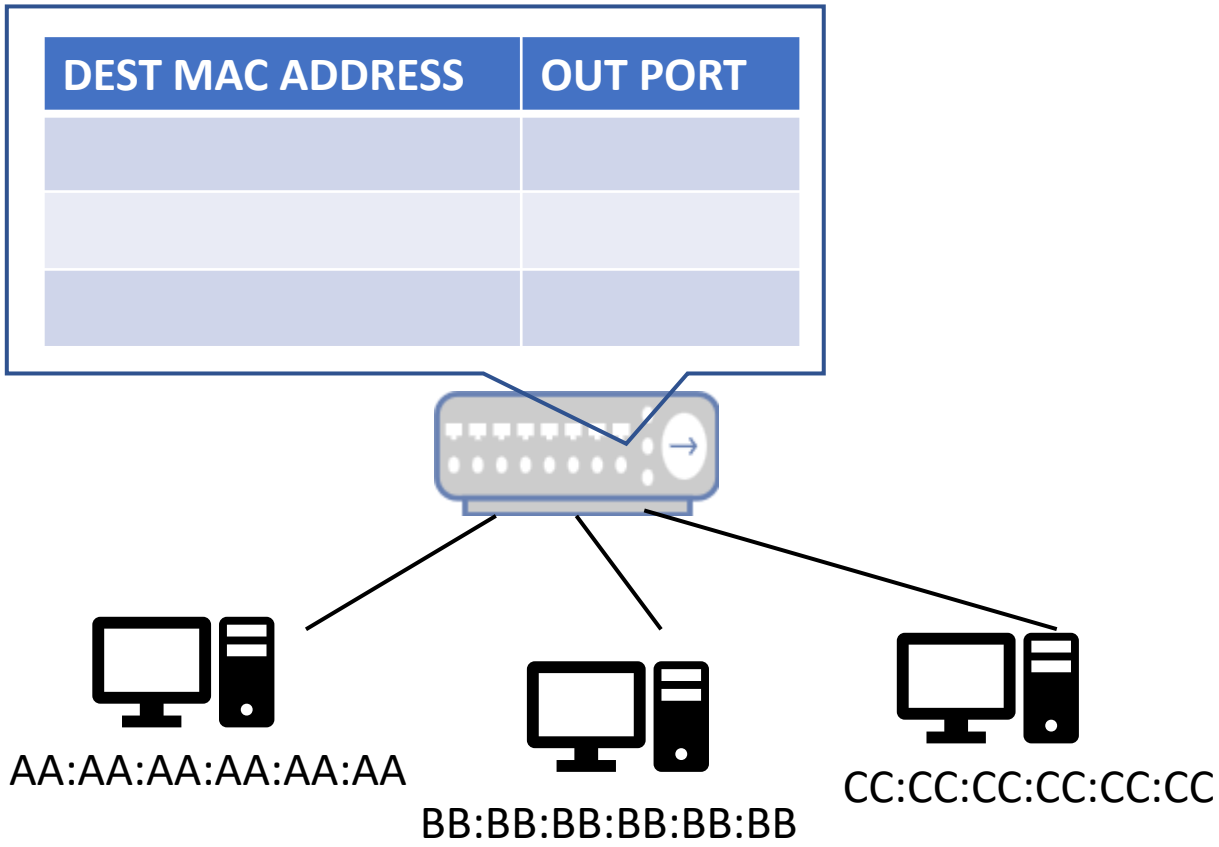


- Table in switch maps destination addresses to output ports
- Receive frame, lookup destination, send frame

Dest	Src	Type	Payload	FCS
CCCCCCCCCCCC	AAAAAAAAAAAA	0800	F10FE23…	102F

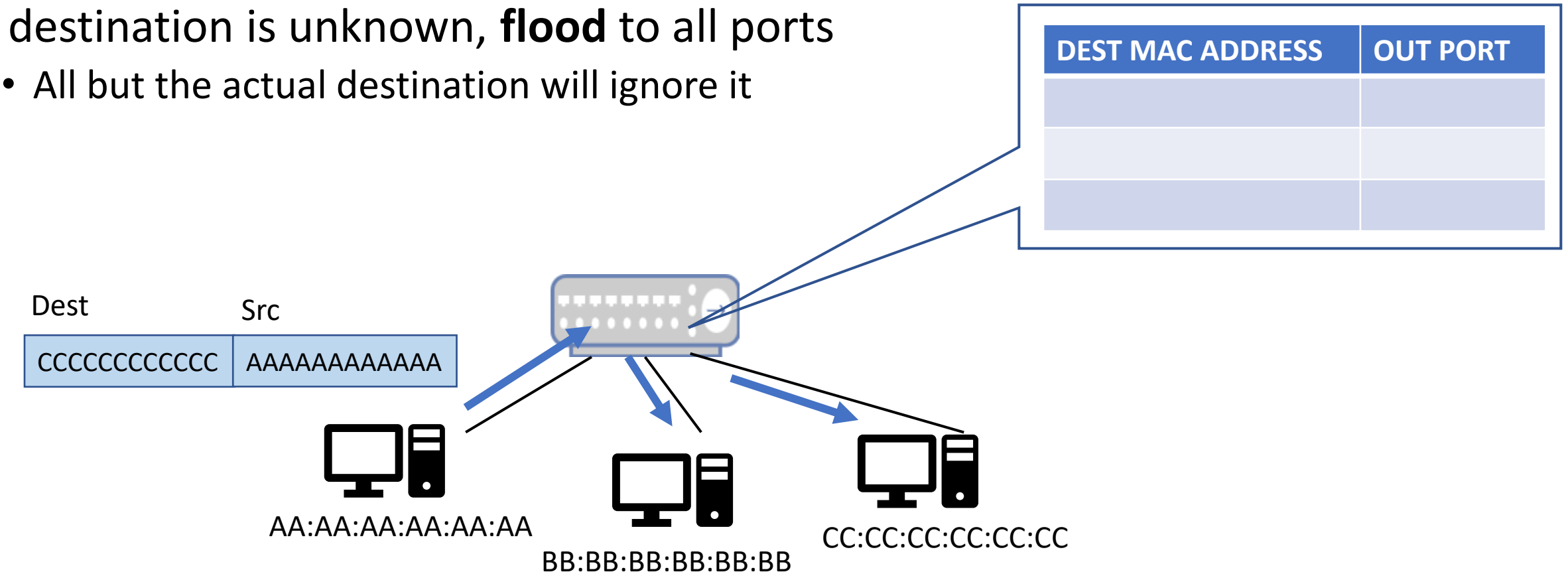
How do we Fill the MAC Address Table?

- Table starts empty



MAC Table Learning

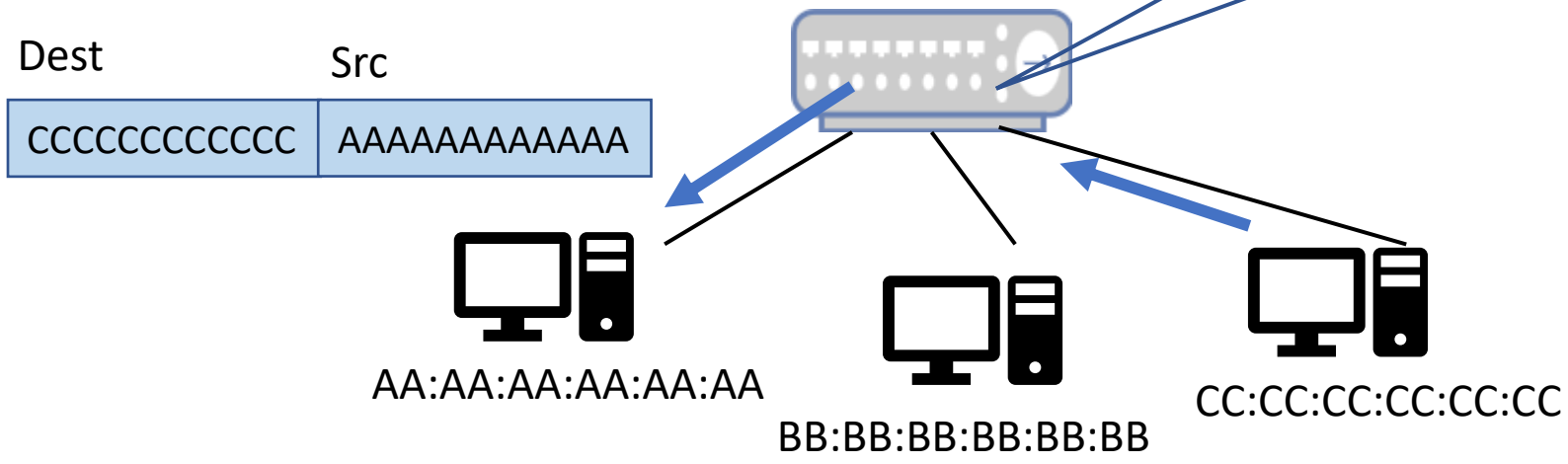
- When switch receives a frame it **learns** the source is at that port
- If destination is unknown, **flood** to all ports
  - All but the actual destination will ignore it



MAC Table Learning

- If destination replies, switch will learn the address on that port

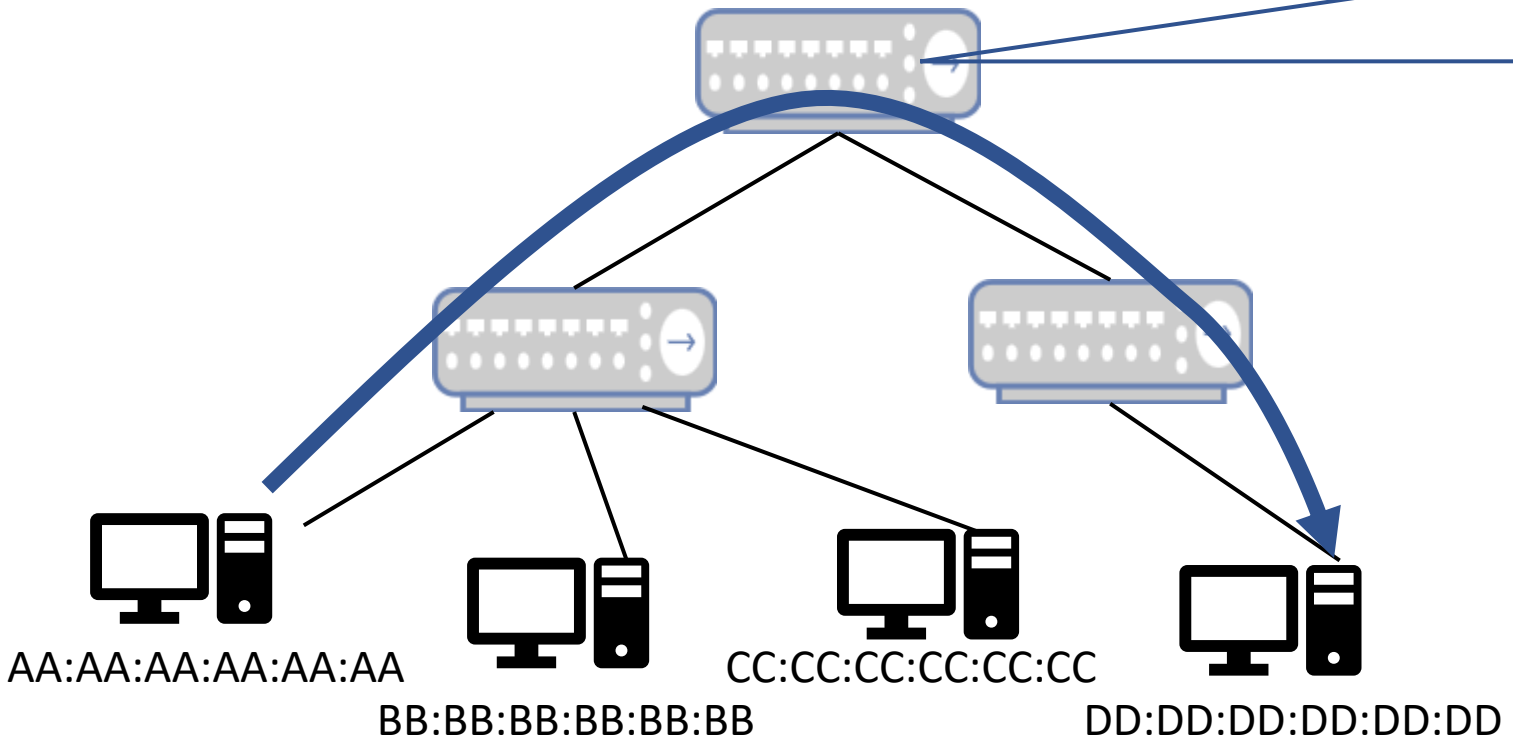
DEST MAC ADDRESS	OUT PORT
AA:AA:AA:AA:AA:AA	0



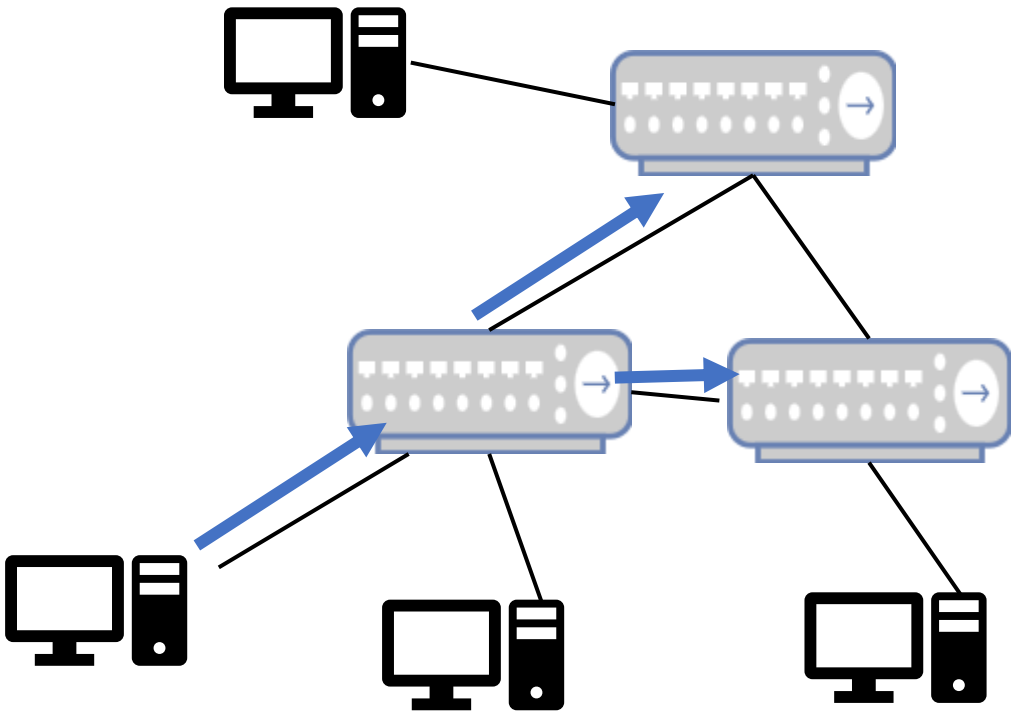
Topology of Switches

- Switches are limited in number of ports, so you can extend with a topology of switches

DEST MAC ADDRESS	OUT PORT
AA:AA:AA:AA:AA:AA	0
BB:BB:BB:BB:BB:BB	0
CC:CC:CC:CC:CC:CC	0
DD:DD:DD:DD:DD:DD	1

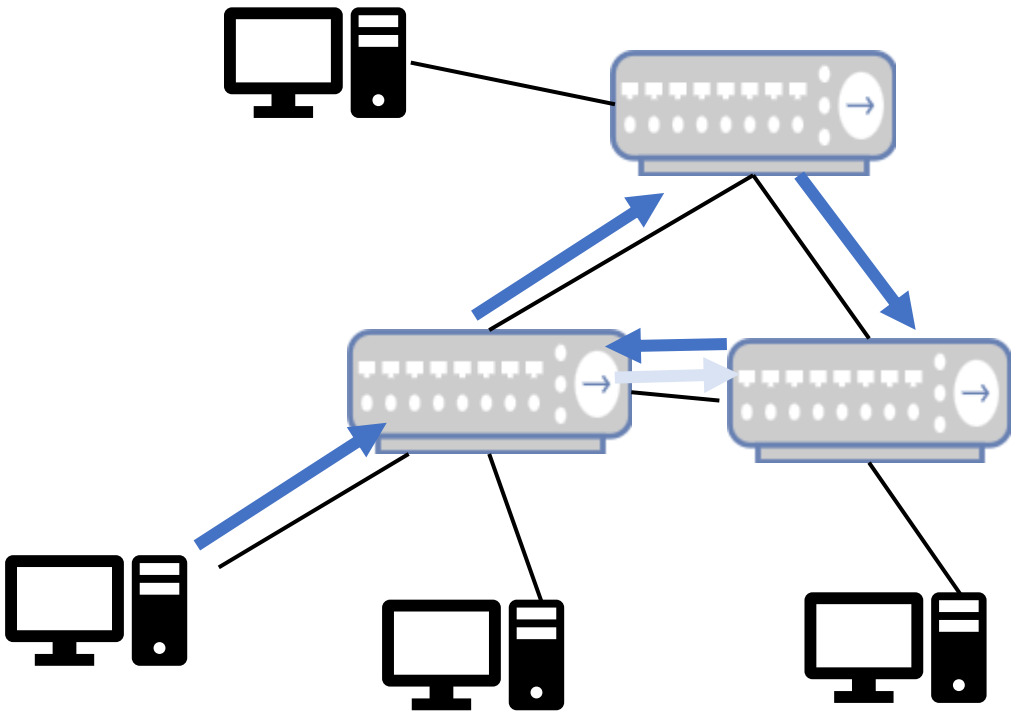


Problem: Loops



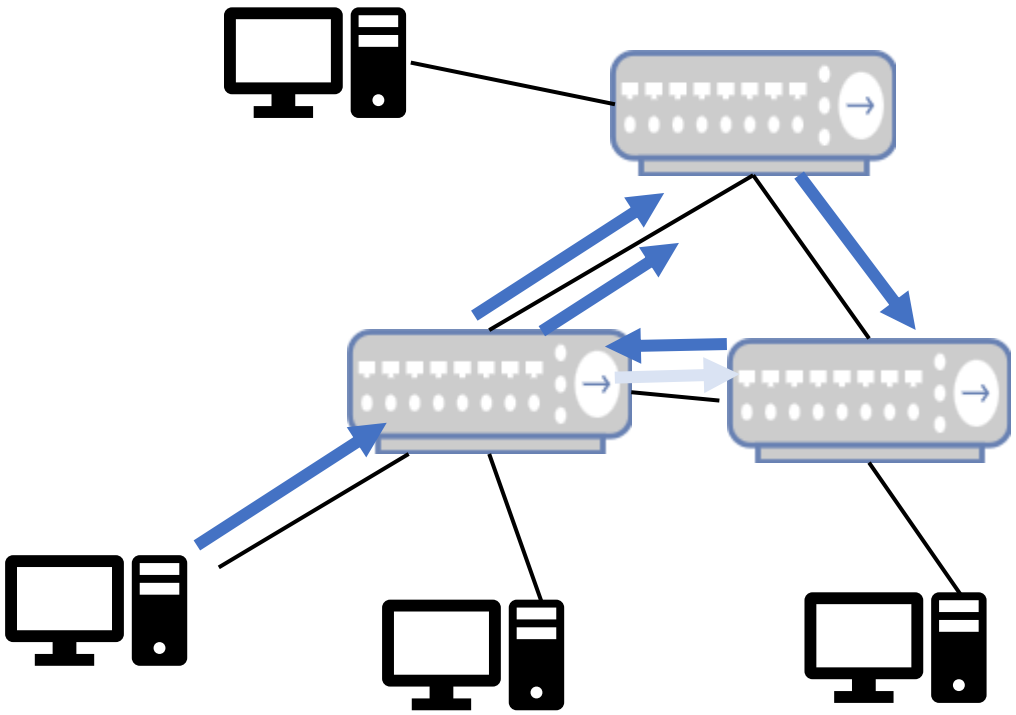
- Recall that there are broadcast MAC addresses
- Recall that we used flooding for learning

Problem: Loops



- Recall that there are broadcast MAC addresses
- Recall that we used flooding for learning

Problem: Loops



- Recall that there are broadcast MAC addresses
- Recall that we used flooding for learning

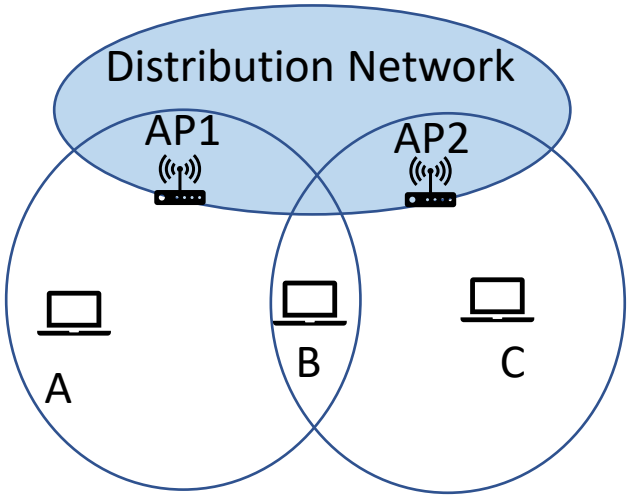
Solutions to loops

- Could keep track of messages seen
- Could limit the number of hops a given message takes
- Could turn off links that create the loop
  (with a protocol like Spanning Tree Protocol)
- Architect network such that looping won't occur
  (e.g., topology, or avoid broadcasts and flooding)

In Ethernet LANs, Spanning Tree Protocol is popular, as is architecting the network such that looping won't occur

Wireless LAN

- The special nodes in this case are access points
- Access points commonly connected to a wired network
- All traffic goes via the access points (even if nodes are in range)
- Key challenge here is associating with an access point



University of Colorado **Boulder**