CSCI 466: Networks

Link Layer: Error Detection, Multiple Access Protocols

Reese Pearsall

Fall 2023

Announcements

Wireshark Lab 3 Due tonight

PA3 due 11/8

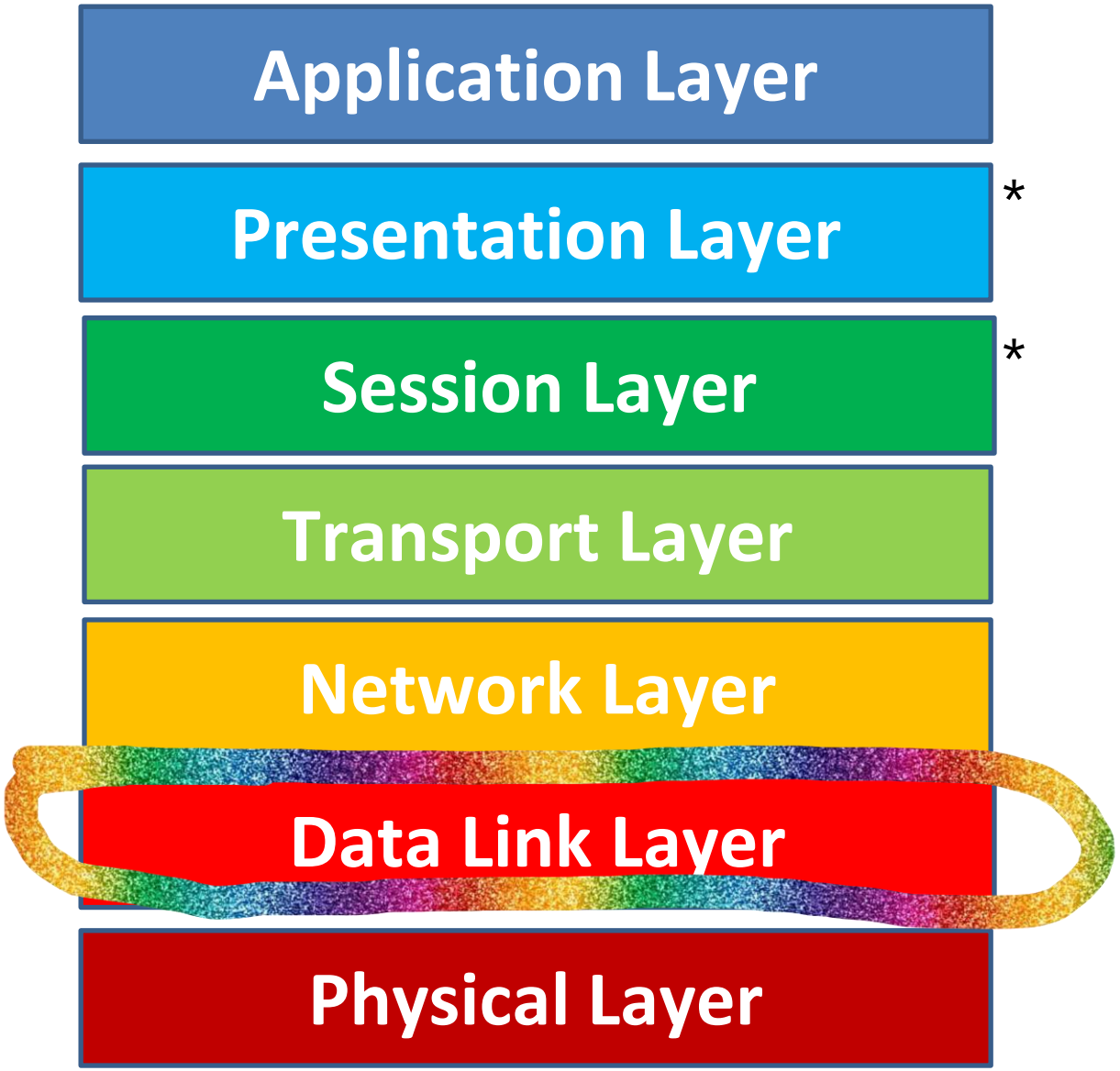
Error Correction

An IPv6 address is 128 bits in length (not 64)

IPv4	IPv6
Deployed 1981	Deployed 1998
32-bit IP address	128-bit IP address
4.3 billion addresses Addresses must be reused and masked	7.9×10^{28} addresses Every device can have a unique address
Numeric dot-decimal notation 192.168.5.18	Alphanumeric hexadecimal notation 50b2:6400:0000:0000:6c3a:b17d:0000:10a9 (Simplified - 50b2:6400::6c3a:b17d:0:10a9)
DHCP or manual configuration	Supports autoconfiguration

I am going to ask this question on the next Quiz ☺

OSI Model



Application Layer

Messages from Network Applications

Physical Layer

Bits being transmitted over a copper wire

**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

Data Link Layer

The link layer is responsible for the **actual node-to-node delivery** of data and ensure error-free transmission of information

terminology:

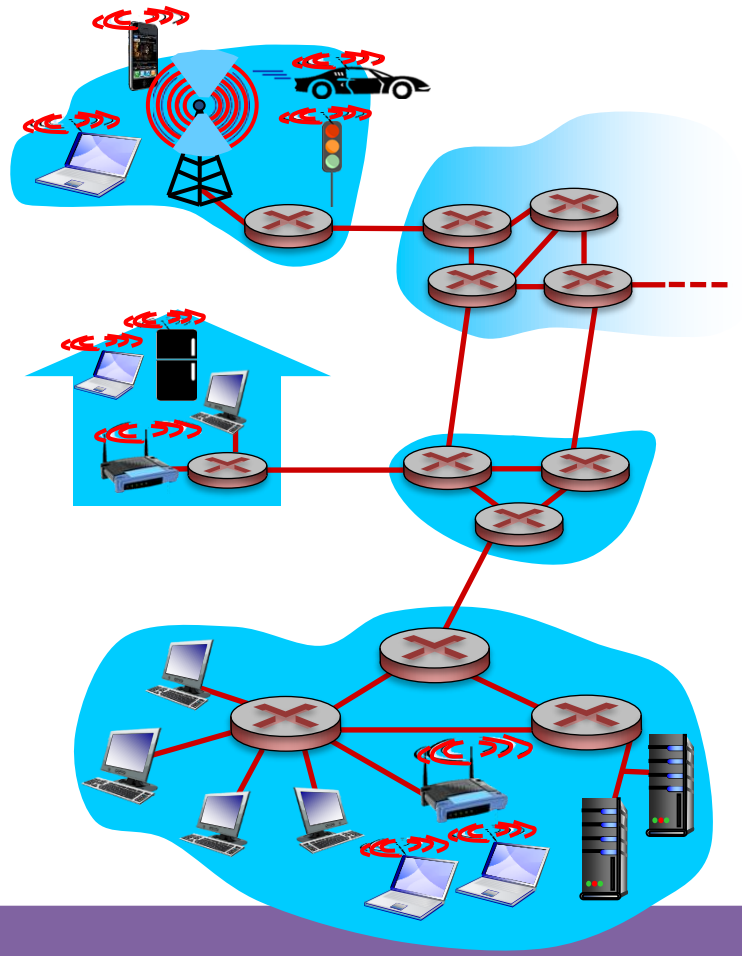
hosts and routers: **nodes**

communication channels that connect adjacent nodes along communication path: **links**

- wired links
- wireless links
- LANs

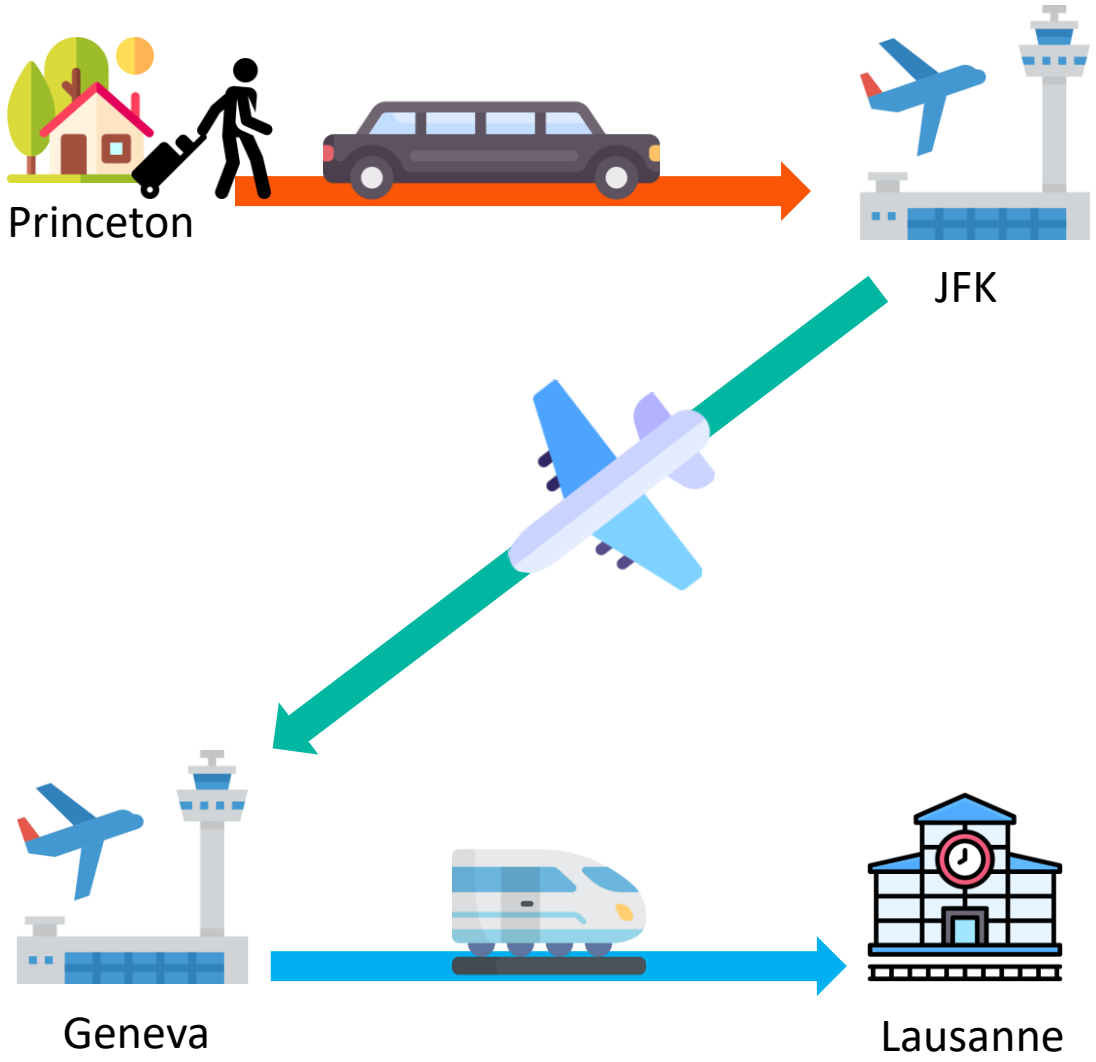
layer-2 packet: frame, encapsulates **datagram**

data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link



We have not addressed how we will overcome various transmission mediums!

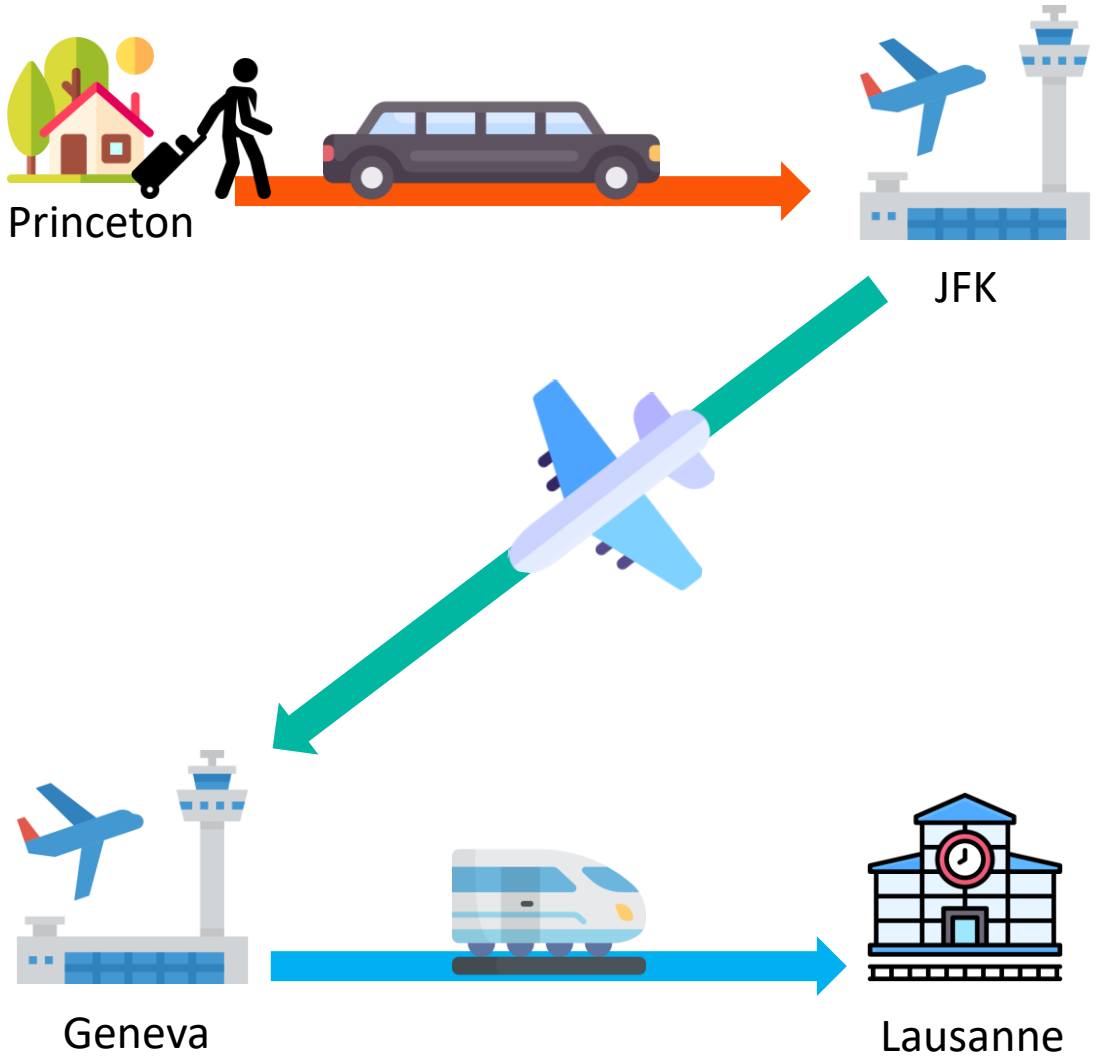
Data Link Layer



transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne

Data Link Layer



transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link-layer protocol
- travel agent = routing algorithm

Data Link Layer

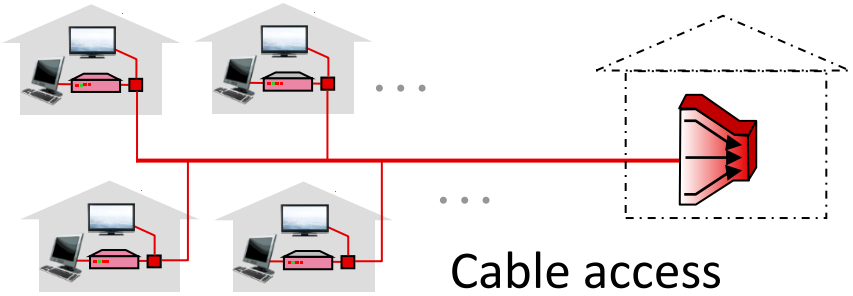
Services offered by the Link Layer

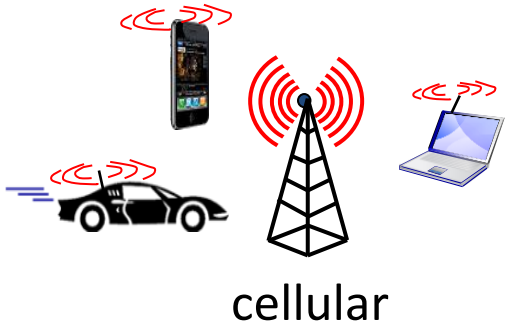
- Framing
  - → Encapsulate a network layer Datagram in *another* header
- Link access
  - → LL dictate the rules and process of transmitting a frame over a link
- Reliable Delivery
  - → For unreliable link, some reliable delivery mechanisms may need to be used
- Error Detection and Correction
  - → Bits can get messed up as the are transmitted through a medium

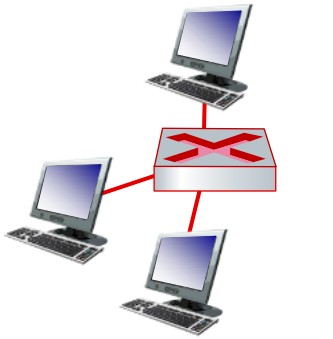
Why do we need RDT and error detection in the link layer when it is also offered in the transport layer?

Data Link Layer

- flow control:
  - pacing between adjacent sending and receiving nodes
- error detection:
  - errors caused by signal attenuation, noise.
  - receiver detects errors, signals retransmission, or drops frame
- error correction:
  - receiver identifies *and corrects* bit error(s) without retransmission
- half-duplex and full-duplex:
  - with half duplex, nodes at both ends of link can transmit, but not at same time







Data Link Layer

Services offered by the Link Layer

- Framing
  - → Encapsulate a network layer Datagram in *another* header
- Link access
  - → LL dictate the rules and process of transmitting a frame over a link
- Reliable Delivery
  - → For unreliable link, some reliable delivery mechanisms may need to be used
- Error Detection and Correction
  - → Bits can get messed up as the are transmitted through a medium

Why do we need RDT and error detection in the link layer when it is also offered in the transport layer?

Some packets of data don't even travel through the transport layer…

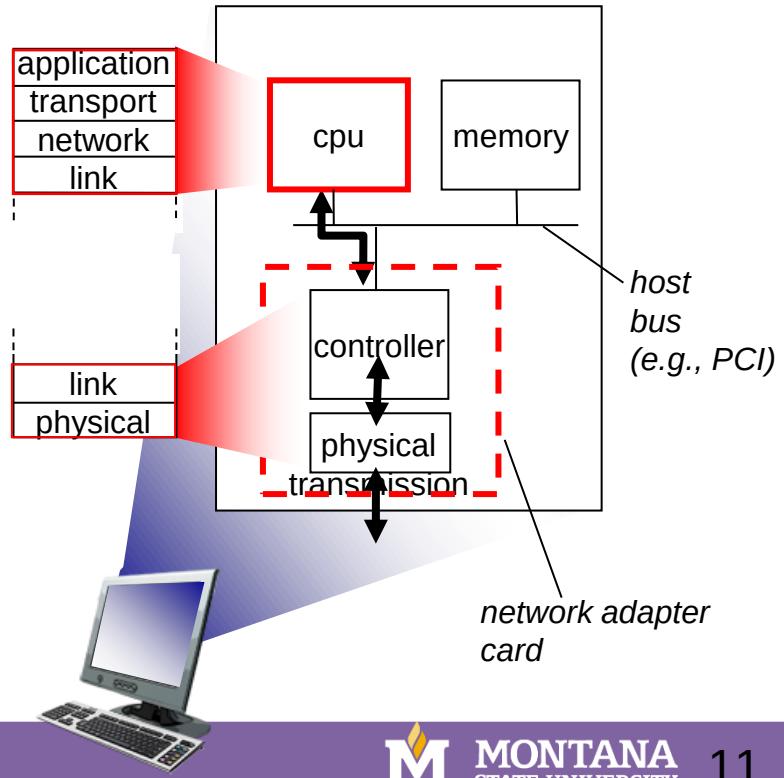
Data Link Layer

Implementation of Link Layer

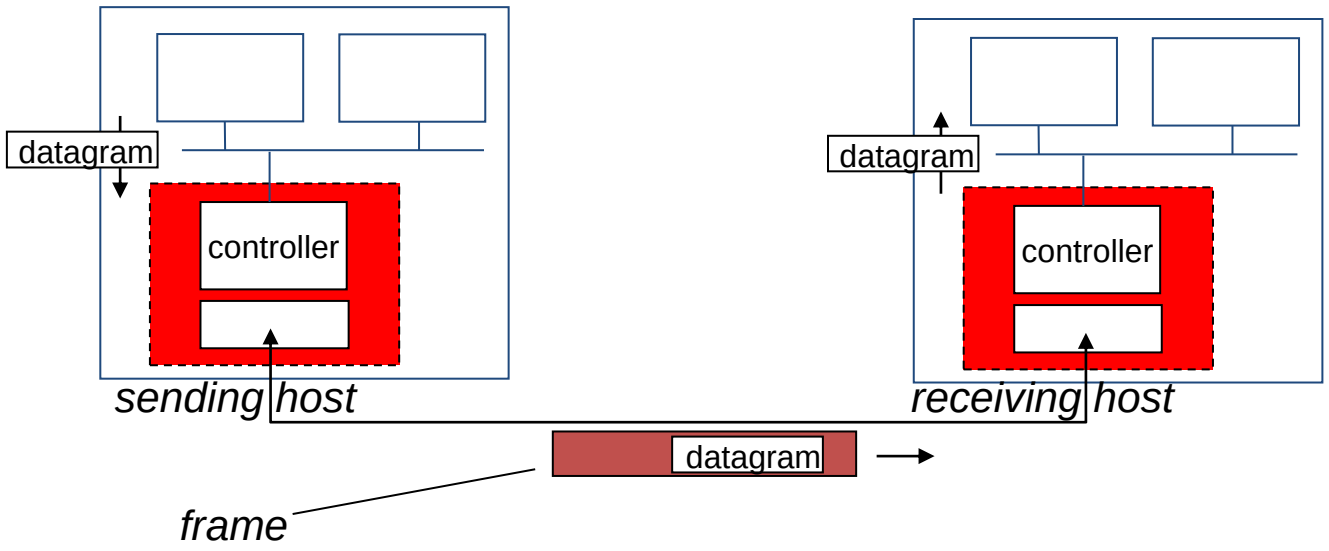
- Implemented within the hardware of your computer

NIC (Network Interface Controller)- Integrated into the motherboard and allows the machine to use LL services such as ethernet (combination of hardware, software, and some firmware)

Wireshark uses your NIC to determine which packets should be sniffed!



Data Link Layer



sending side:

- encapsulates datagram in frame
- adds error checking bits, rdt, flow control, etc.

receiving side

- looks for errors, rdt, flow control, etc.
- extracts datagram, passes to upper layer at receiving side

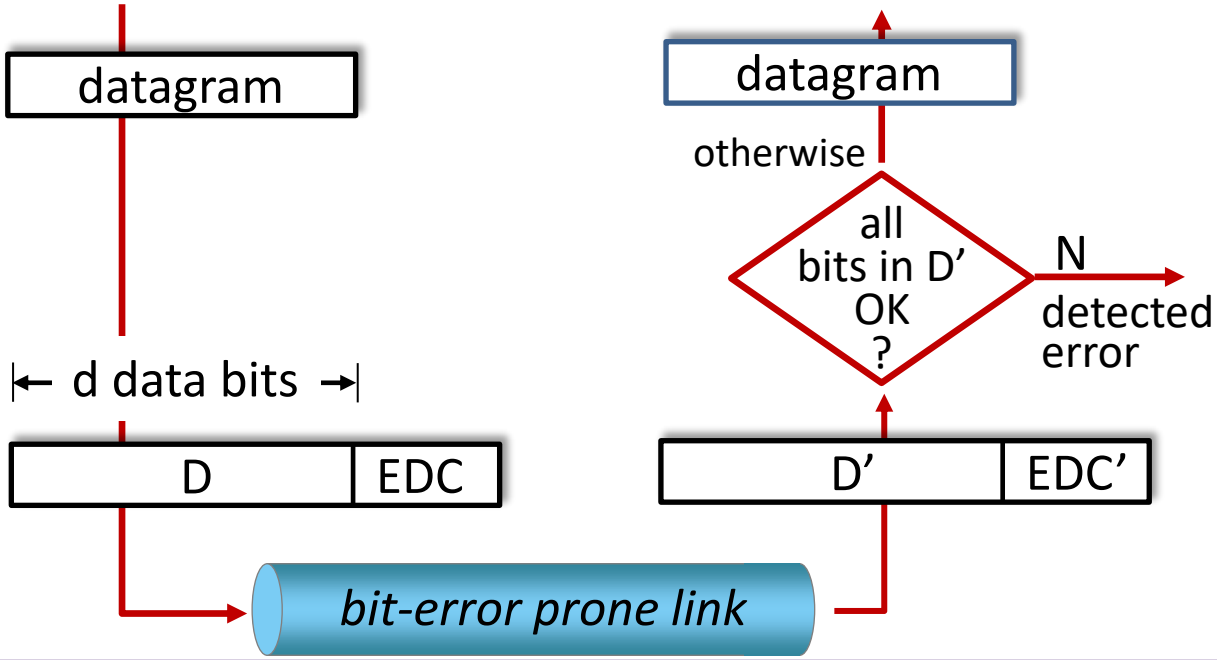
Data Link Layer

Bits can get messed during the physical layer and link layer

- Faulty wires
- NIC issues
- Unreliable mediums

EDC: error detection and correction bits (e.g., redundancy)

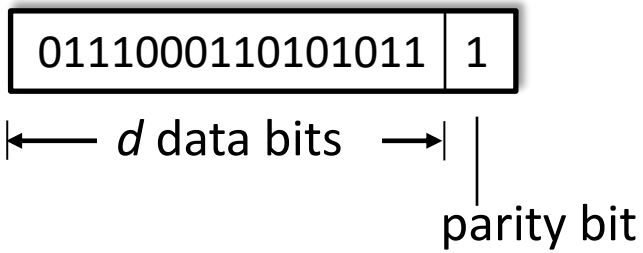
D: data protected by error checking, may include header fields



Error detection not 100% reliable!

- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction

Data Link Layer

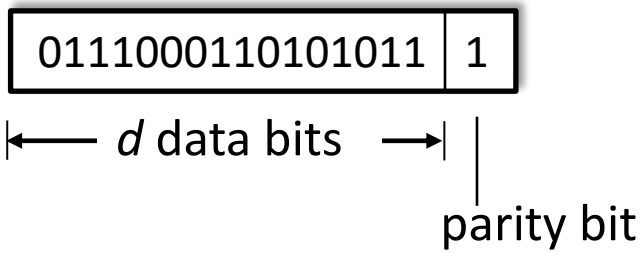


Even/odd parity: set parity bit so there is an even/odd number of 1's

At receiver:

- compute parity of *d* received bits
- compare with received parity bit – if different than error detected

Data Link Layer



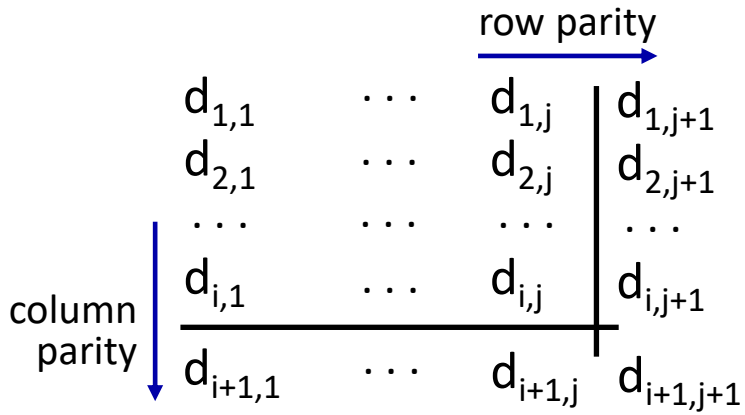
Even/odd parity: set parity bit so there is an even/odd number of 1's

At receiver:

- compute parity of *d* received bits
- compare with received parity bit – if different than error detected

Can detect *and* correct errors (without retransmission!)

- two-dimensional parity: detect *and correct* single bit errors



no errors:

1 0 1 0 1	1
1 1 1 1 0	0
0 1 1 1 0	1
1 0 1 0 1	0

detected and correctable single-bit error:

1 0 1 0 1	1
1 0 1 1 0	0
0 1 1 1 0	1
1 0 1 0 1	0

parity error

Data Link Layer

Checksum (Sender)

```
  0110011001100000
+ 0101010101010101
+ 1000111100001100
------------------
  0100101011000010
```

Binary sum of words

(one's complement)

1011010100111101 Checksum!

(Receiver)

```
  0110011001100000
+ 0101010101010101
+ 1000111100001100
+ 0100101011000010
------------------
```

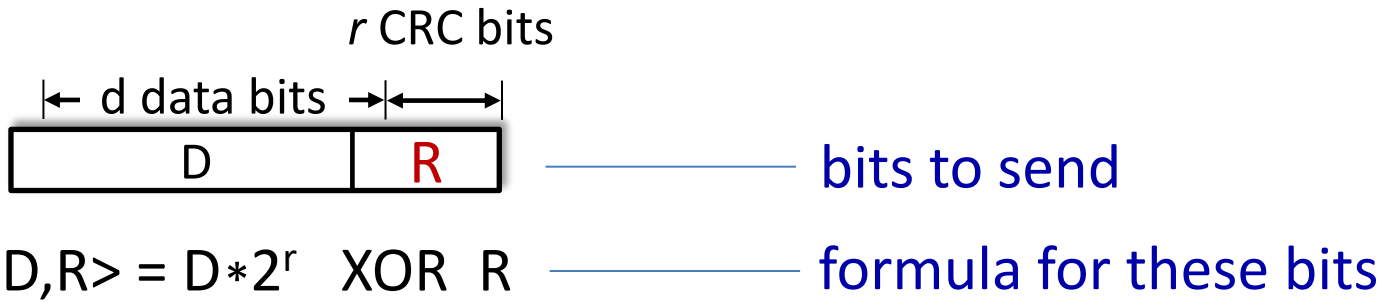
(Binary Sum → One's Complement)

= 1111111111111111

All 1s = No error!

Data Link Layer

- more powerful error-detection coding
- D: data bits (given, think of these as a binary number)
- G: bit pattern (generator), of *r+1* bits (given, specified in CRC standard)



sender: compute *r* CRC bits, R, such that <D,R> *exactly* divisible by G (mod 2)

- receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi)

Sender/Receiver has D and G.
Need to compute R

Data Link Layer

Sender wants to compute R such that:

$D \cdot 2^r \ XOR \ R = nG$

... or equivalently (XOR R both sides):

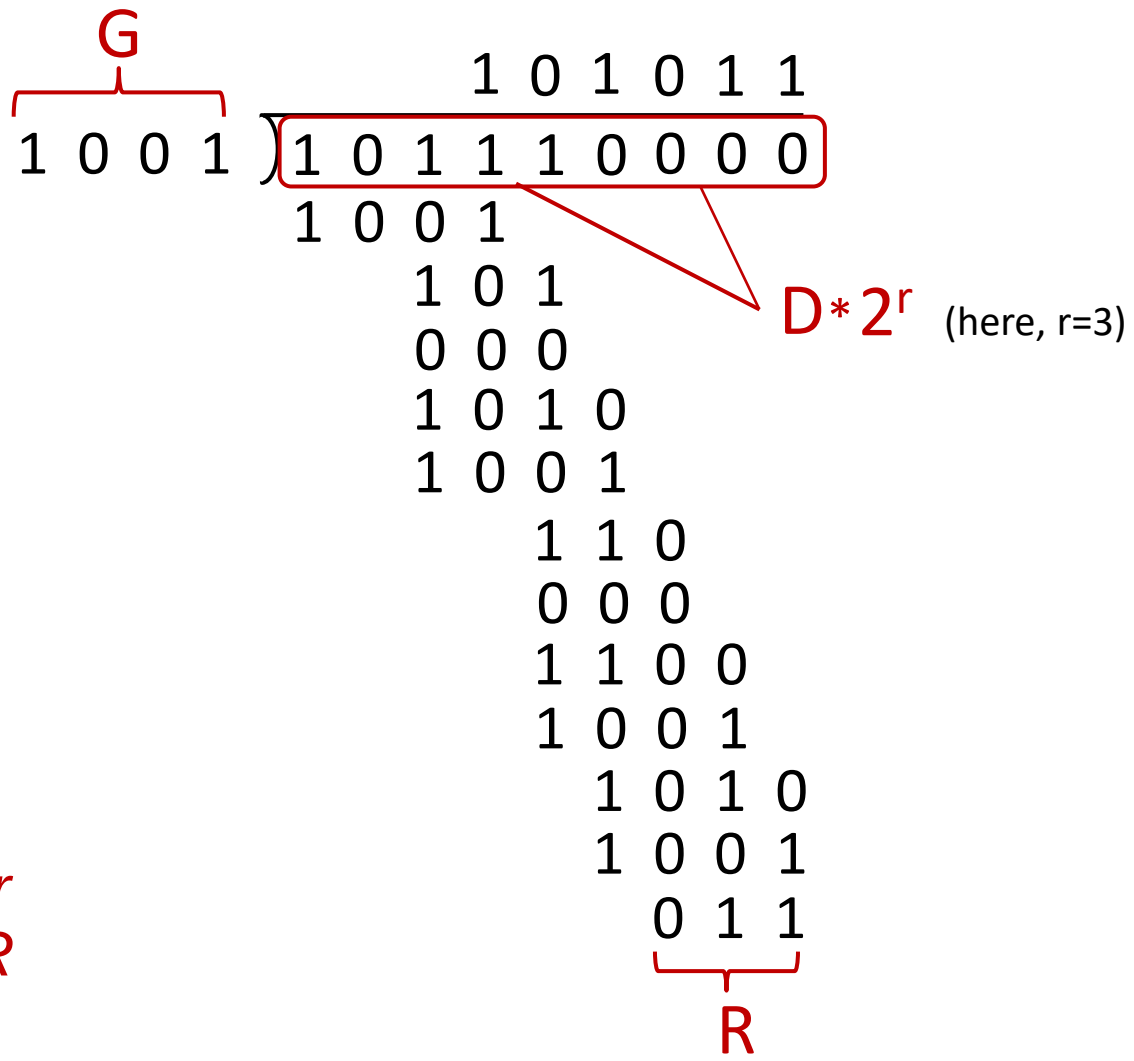
$D \cdot 2^r = nG \ XOR \ R$

... which says:

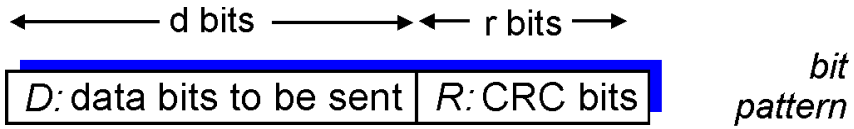
if we divide $D \cdot 2^r$ by G, we want remainder R to satisfy:

$$R = \text{remainder}\left[\frac{D \cdot 2^r}{G}\right]$$

algorithm for computing R



Data Link Layer



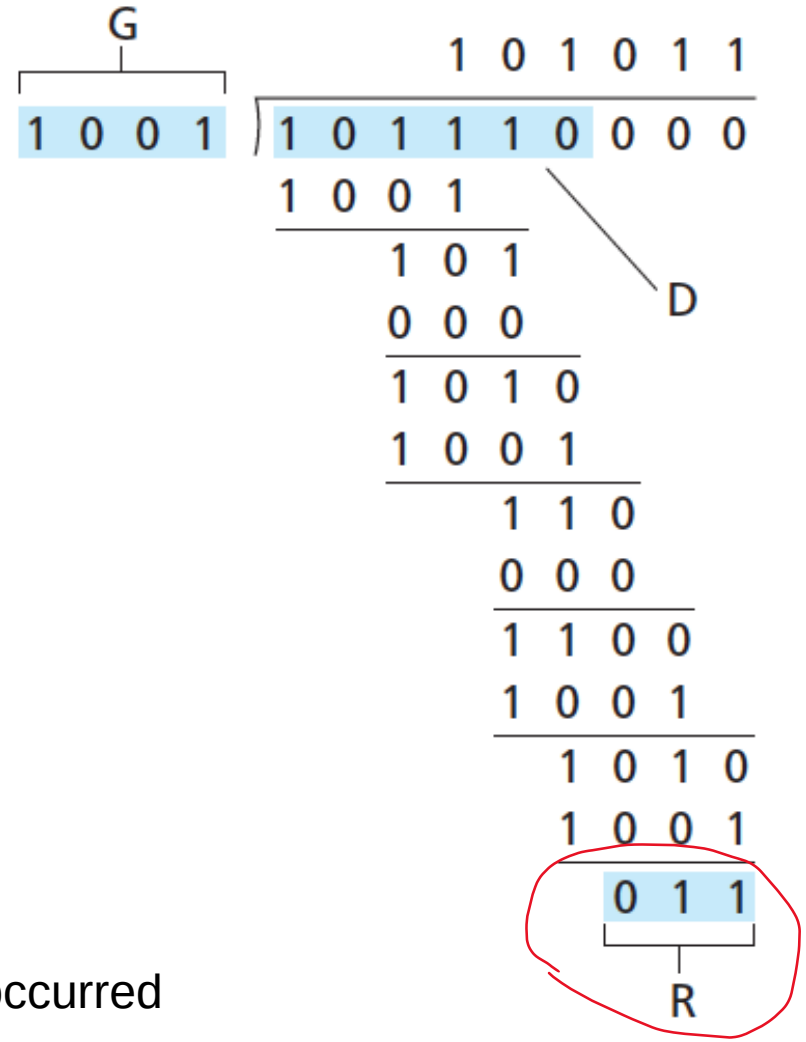
bit pattern

$D * 2^r$ XOR R

mathematical formula

(Do some algebra to find R)

$$R = remainder[\frac{D \cdot 2^r}{G}]$$



Sender sends D + R bits.

Receiver divides D + R bits by G. Result should always be Zero if no errors occurred

Data Link Layer

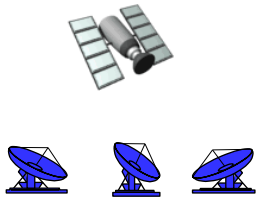
Access links

- Point to Point – Single sender, Single Receiver at each end of link

- Broadcast – shared medium

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

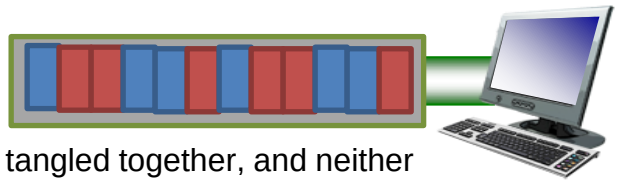
humans at a cocktail party (shared air, acoustical)

Multiple Access Links

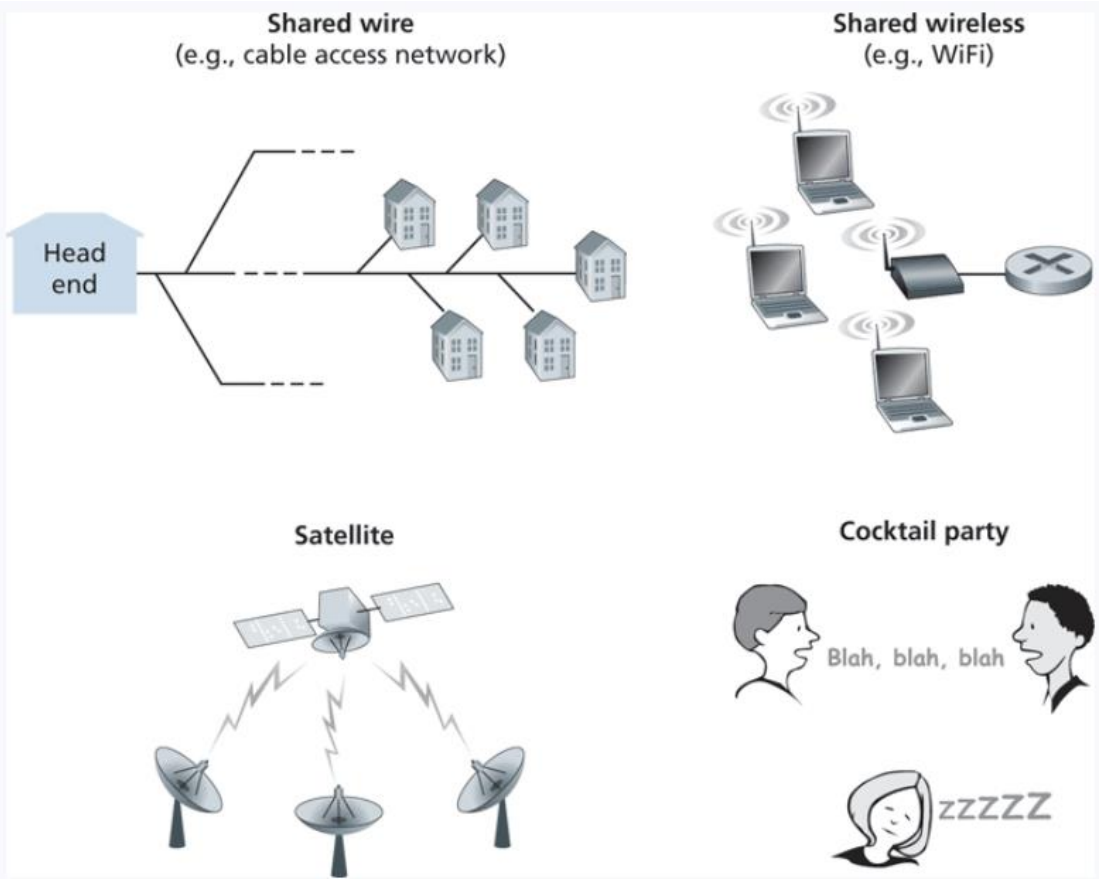
Shared medium = possibility for receivers to get two frame at the same time, AKA a **collision**

Frame X

Frame y



Frames get tangled together, and neither frame can be read clearly by receiver

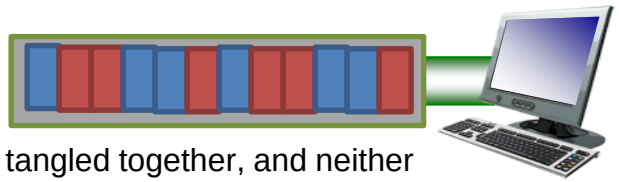


Multiple Access Links

Shared medium = possibility for receivers to get two frame at the same time, AKA a **collision**

Frame X

Frame y



Frames get tangled together, and neither frame can be read clearly by receiver

"Give everyone a chance to speak."

"Don't speak until you are spoken to."

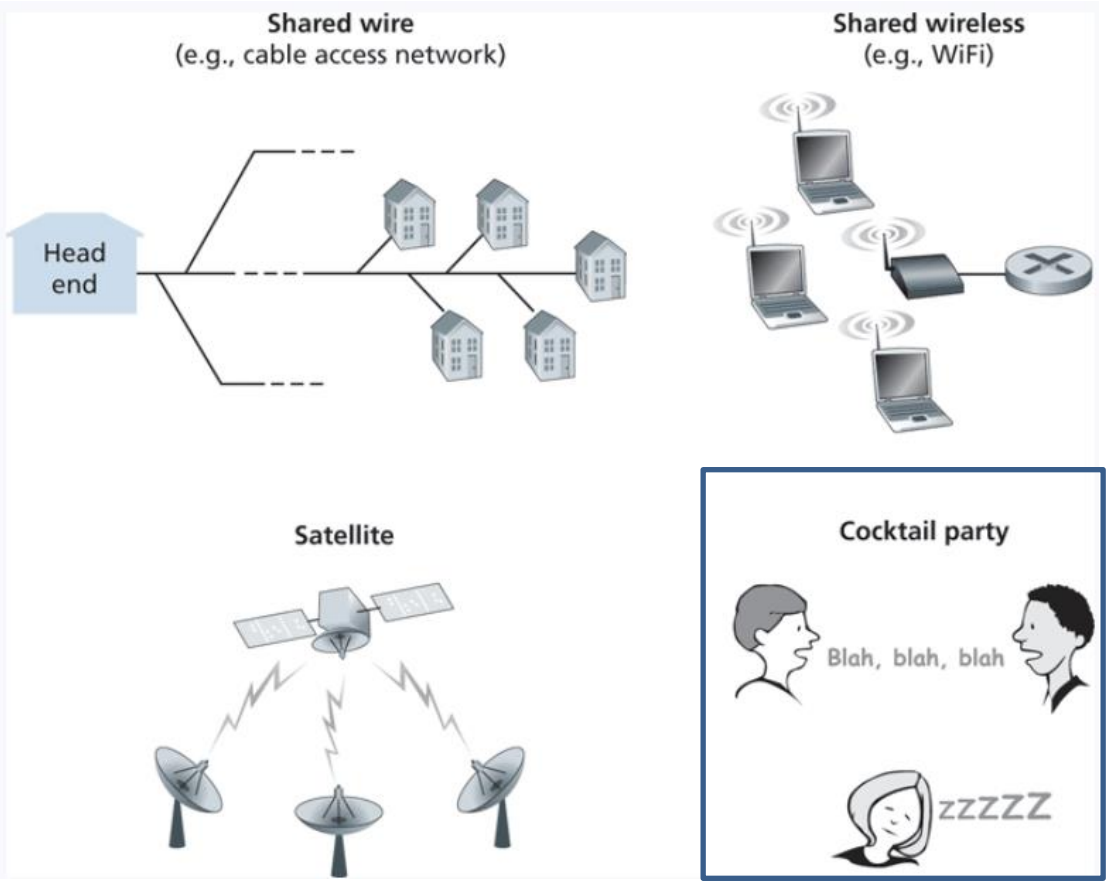
"Don't monopolize the conversation."

"Raise your hand if you have a question."

"Don't interrupt when someone is speaking."

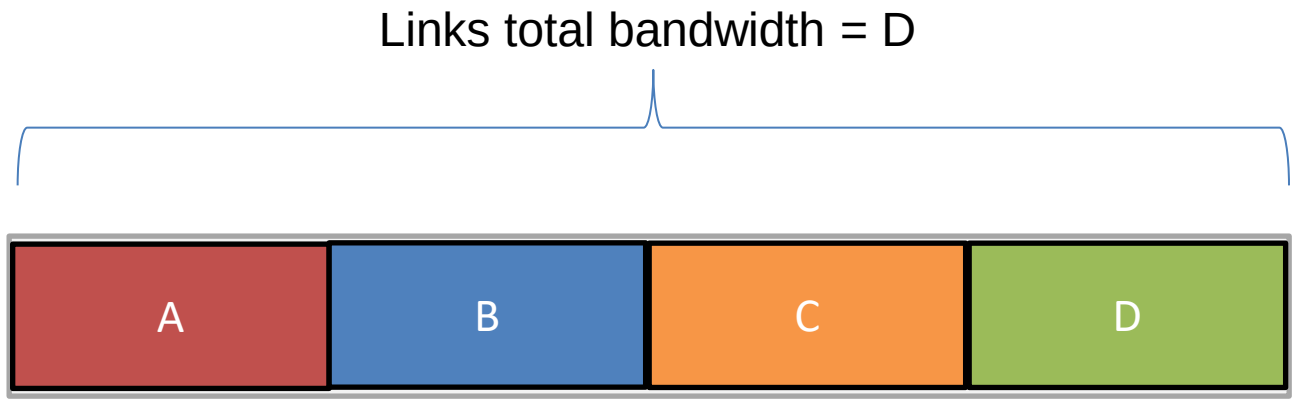
"Don't fall asleep when someone is talking."

In English, we have some rules to prevent collisions from happening

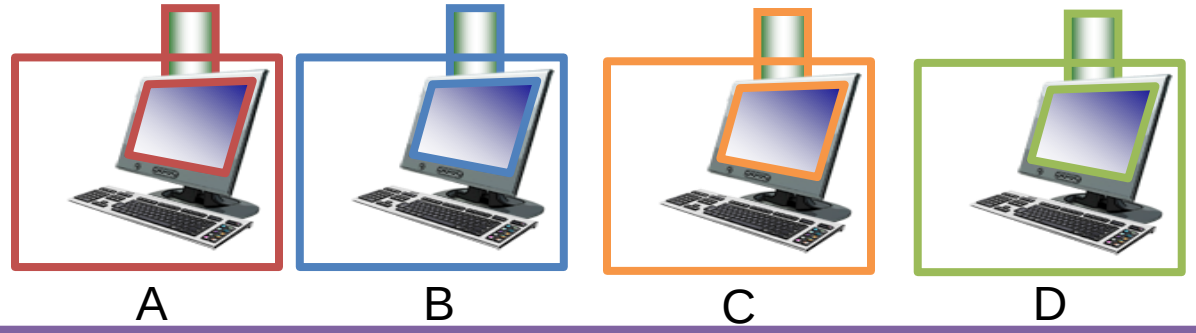


In the link layer, we will discuss 3 multiple access protocols:
Channel Partitioning, **Random Access**, and **Taking Turns**

Channel Partitioning

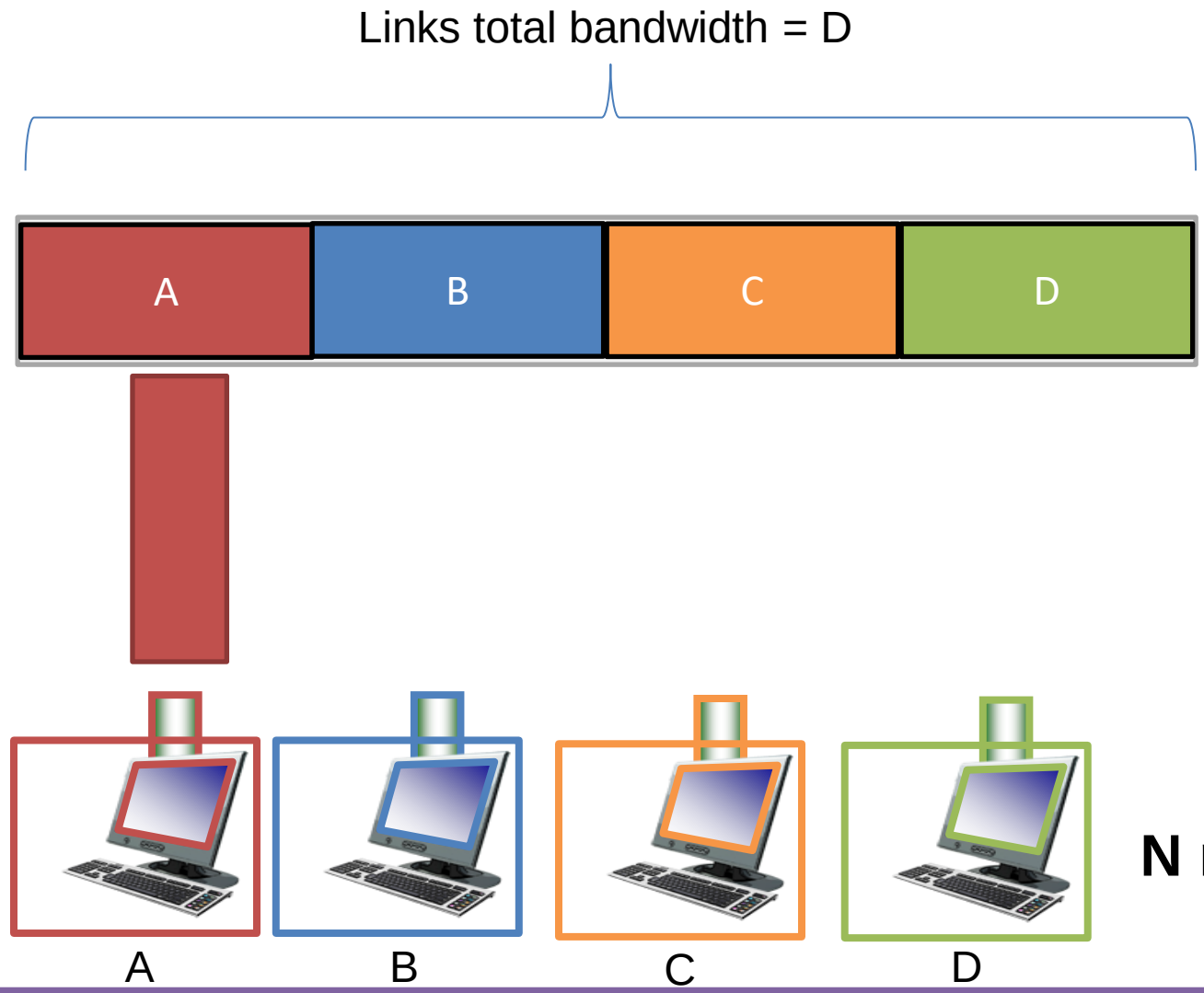


- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit



N nodes

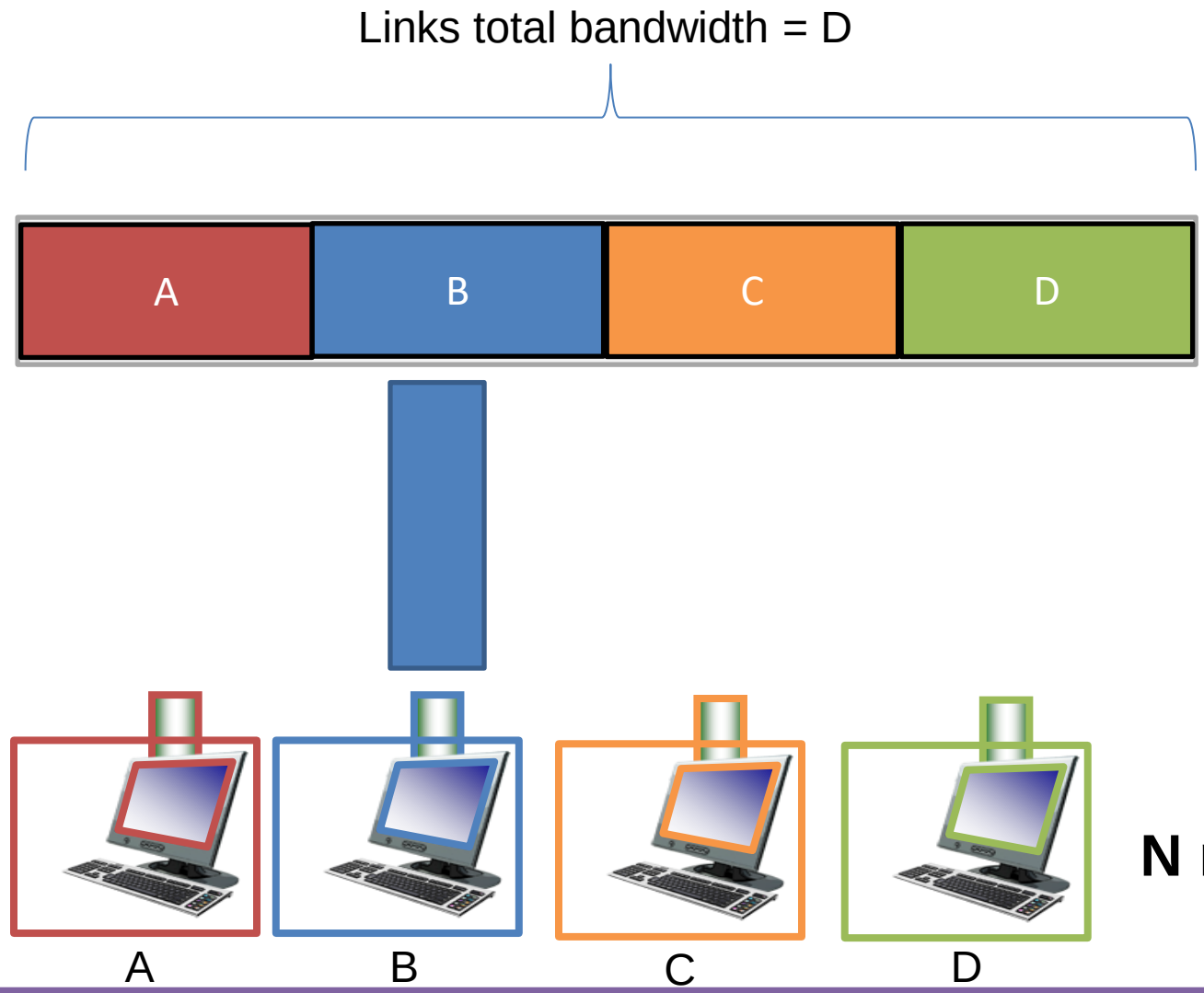
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

N nodes

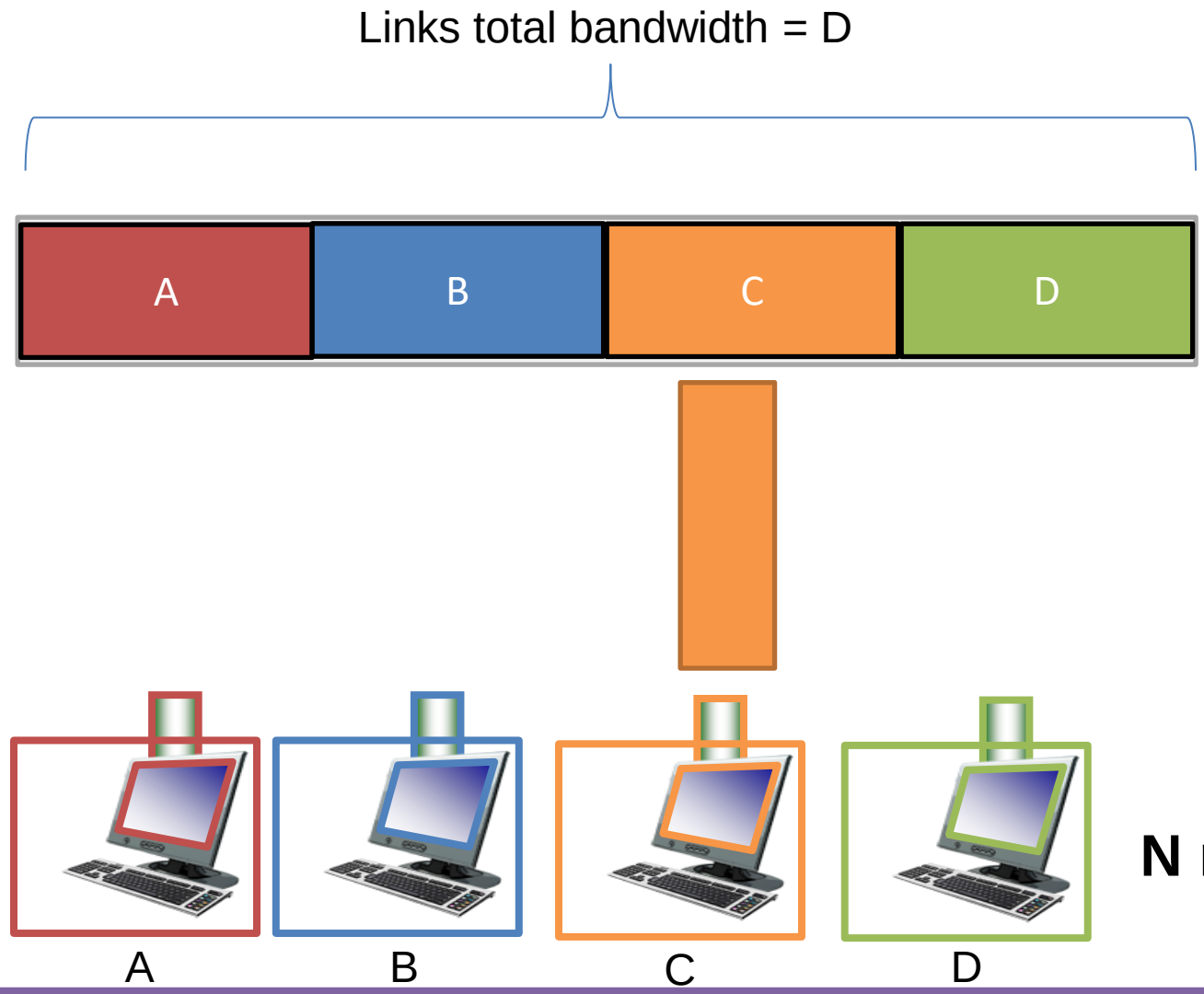
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

N nodes

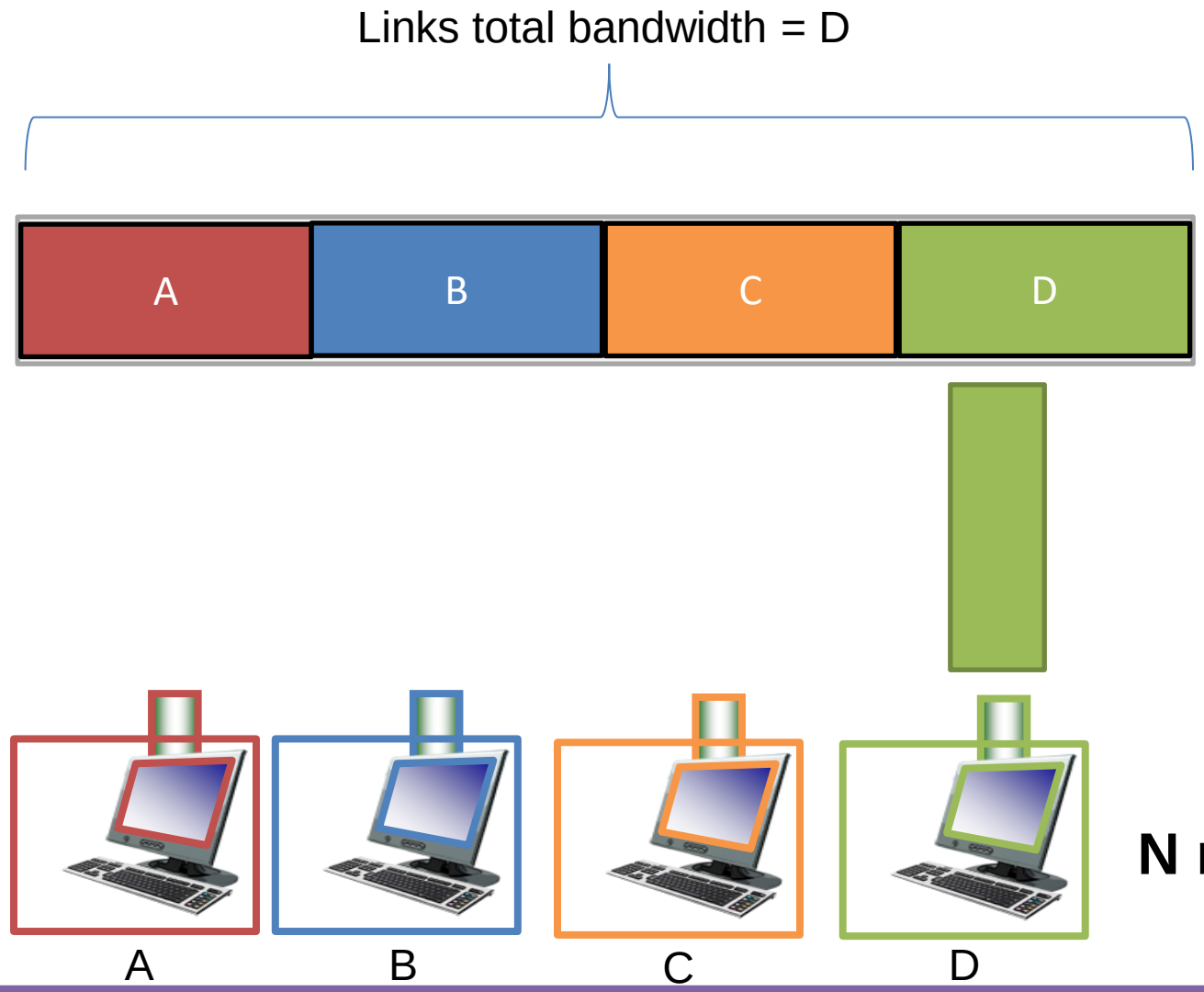
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

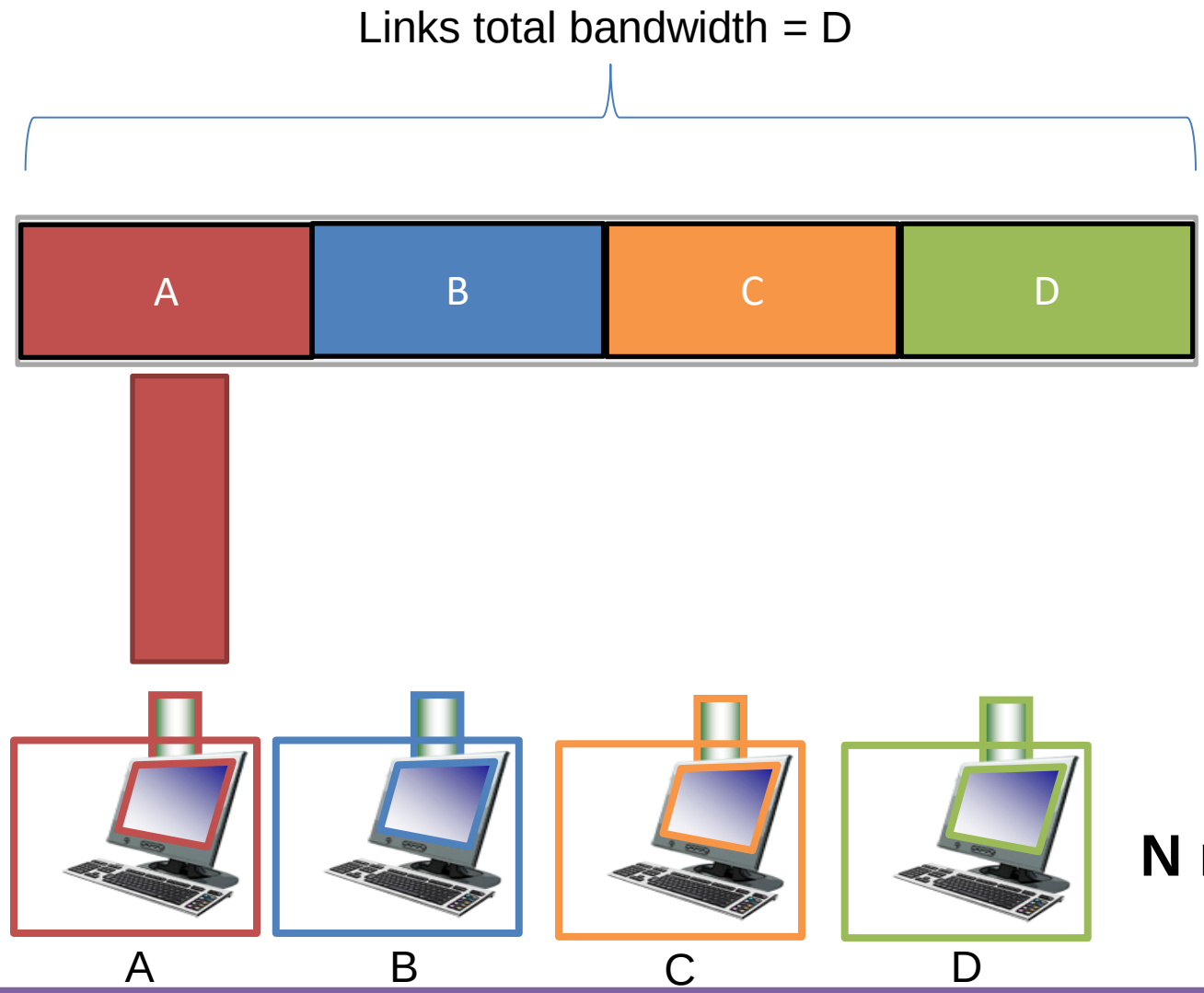
N nodes

Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

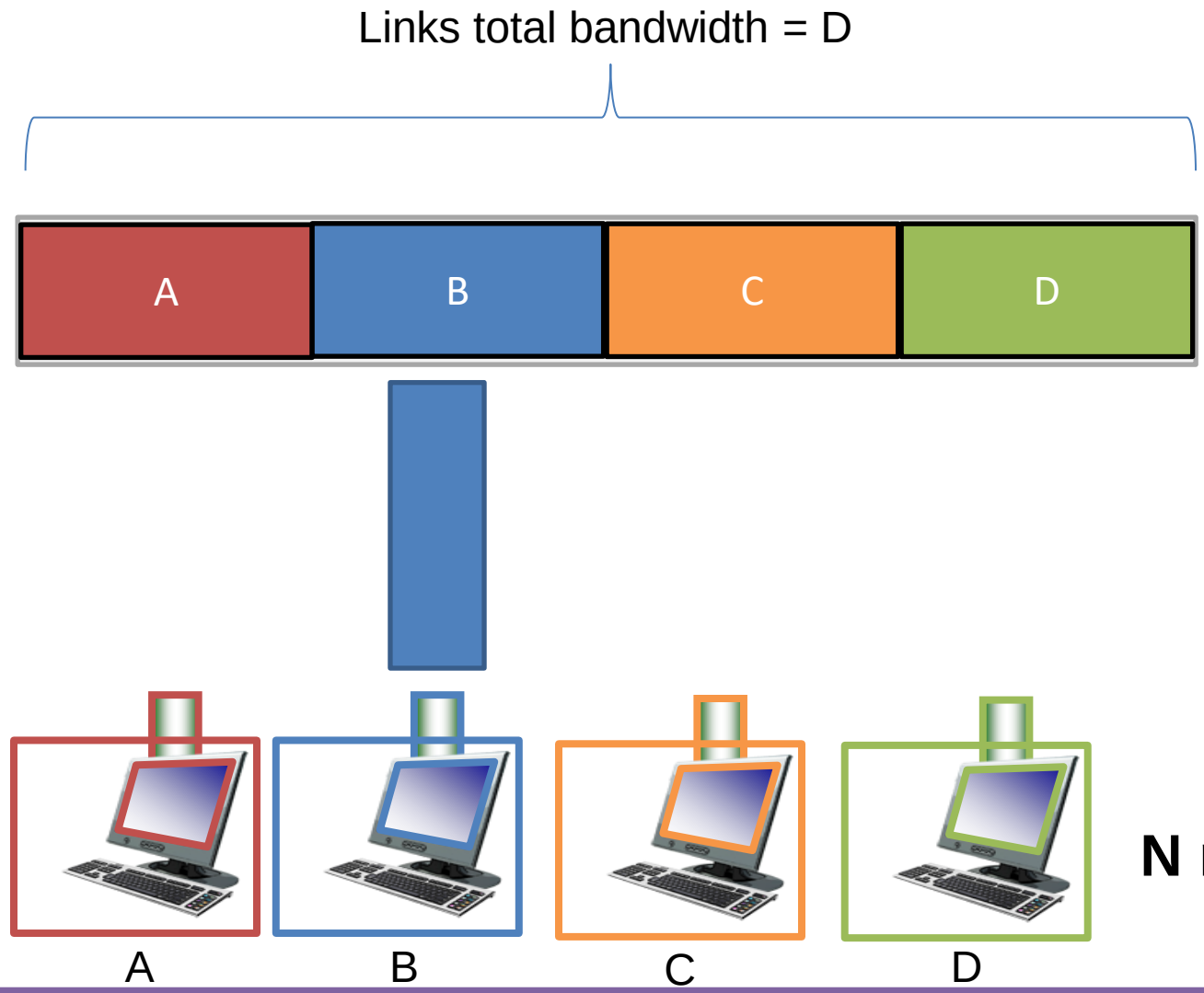
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

N nodes

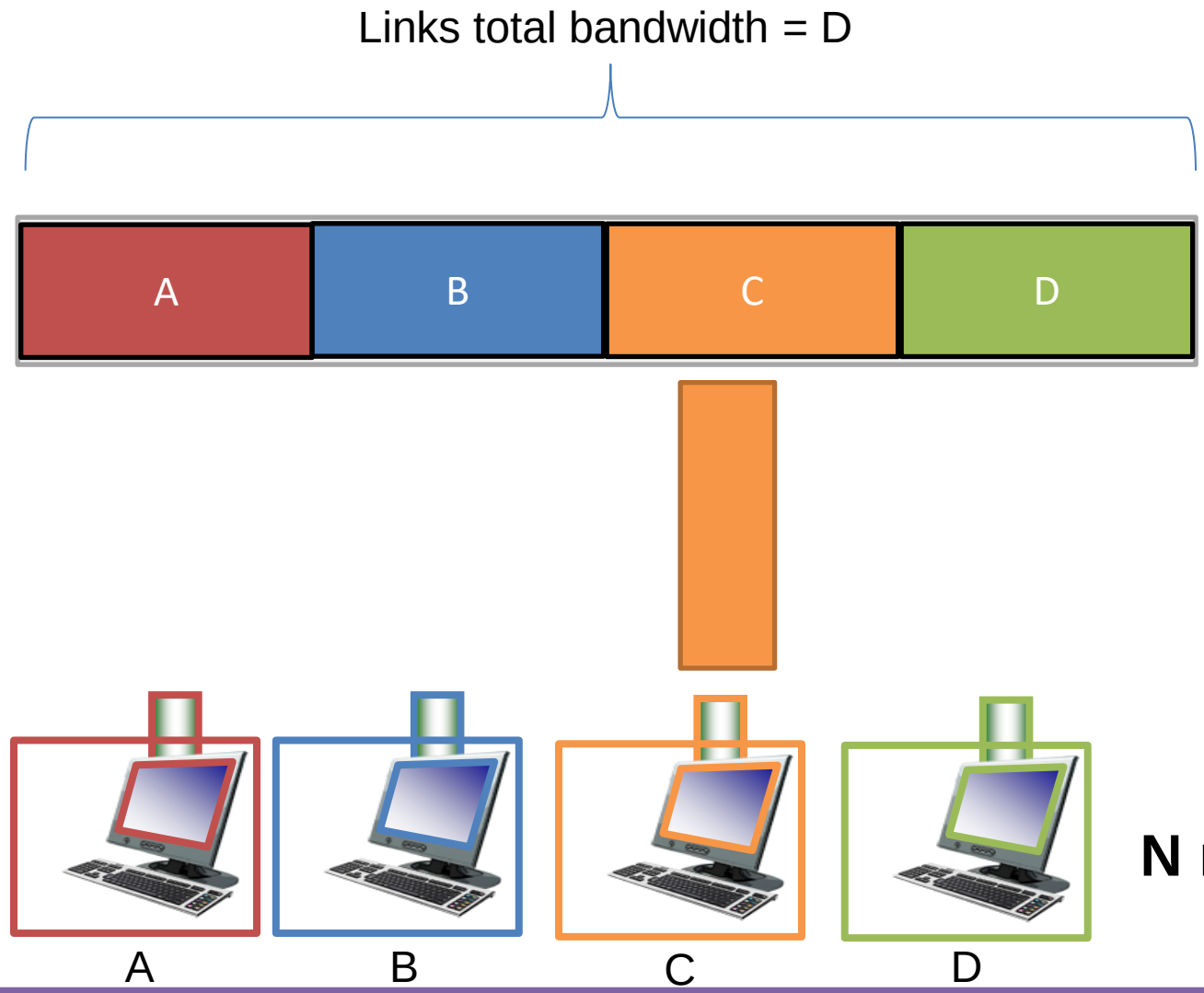
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

N nodes

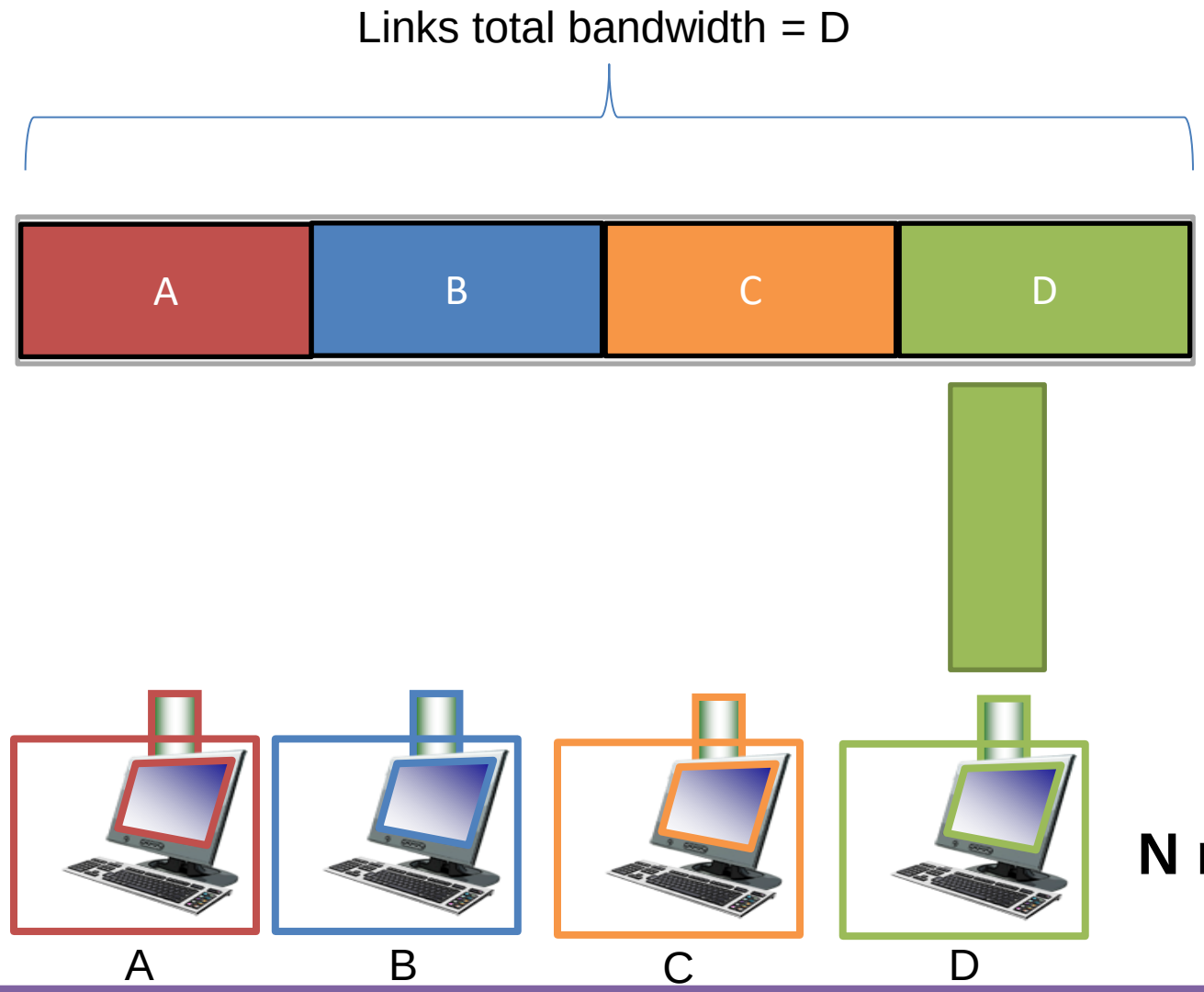
Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

N nodes

Channel Partitioning



- Divide channel into **N** slots
- Each node gets (on average) D/N bandwidth
- Get to transmit data for a fixed amount of time, and then next node gets to transmit

Random Access

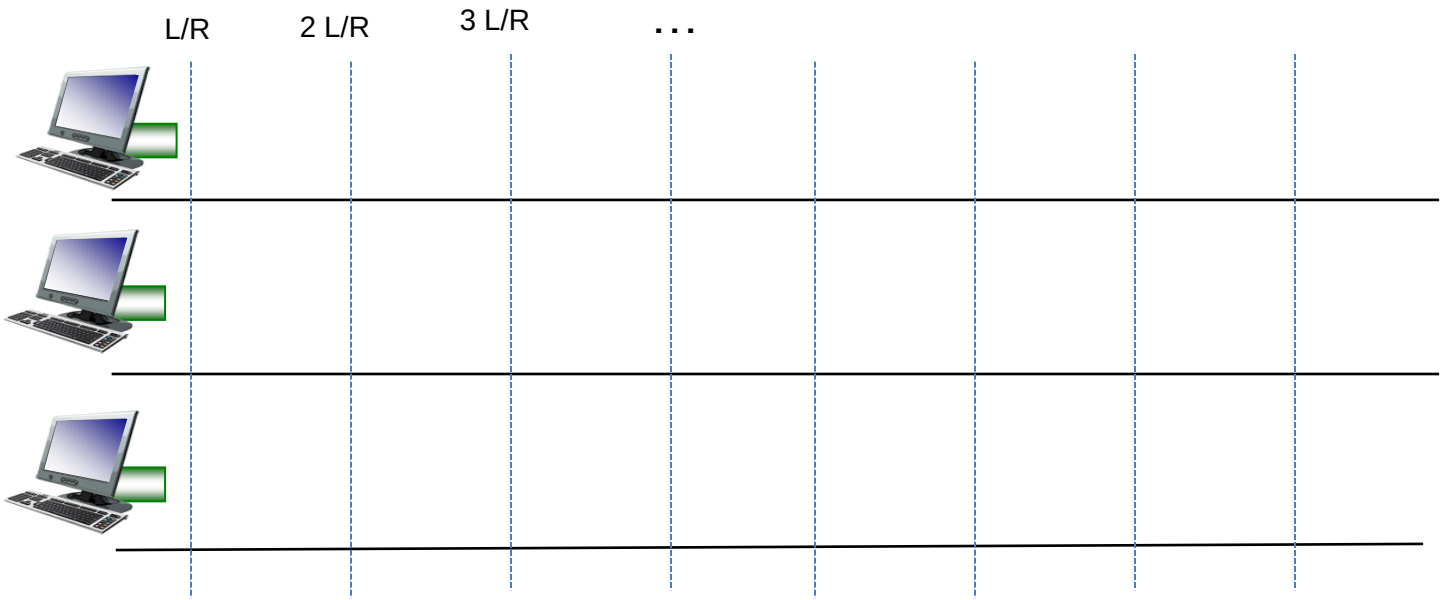
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Can only transmit frames at beginning of slots. If collision occurs, the nodes can detect collision before the slot ends

Random Access

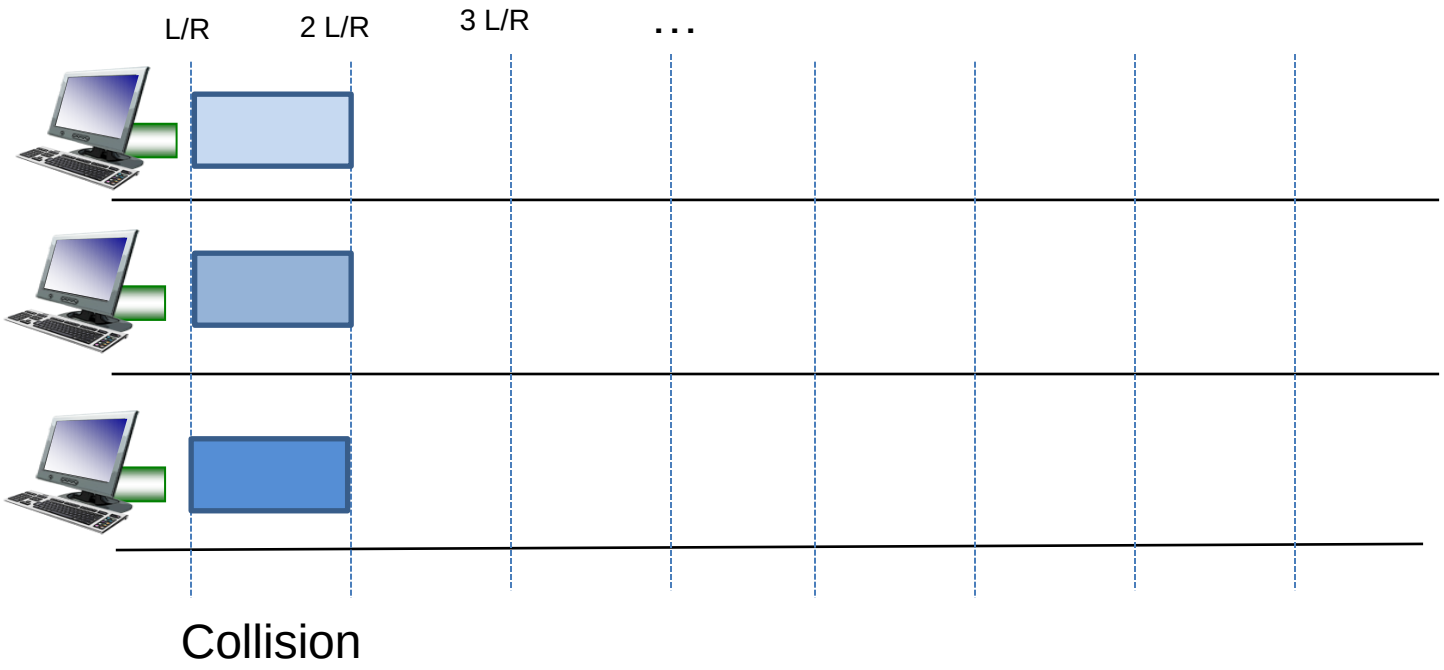
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Collision

Random Access

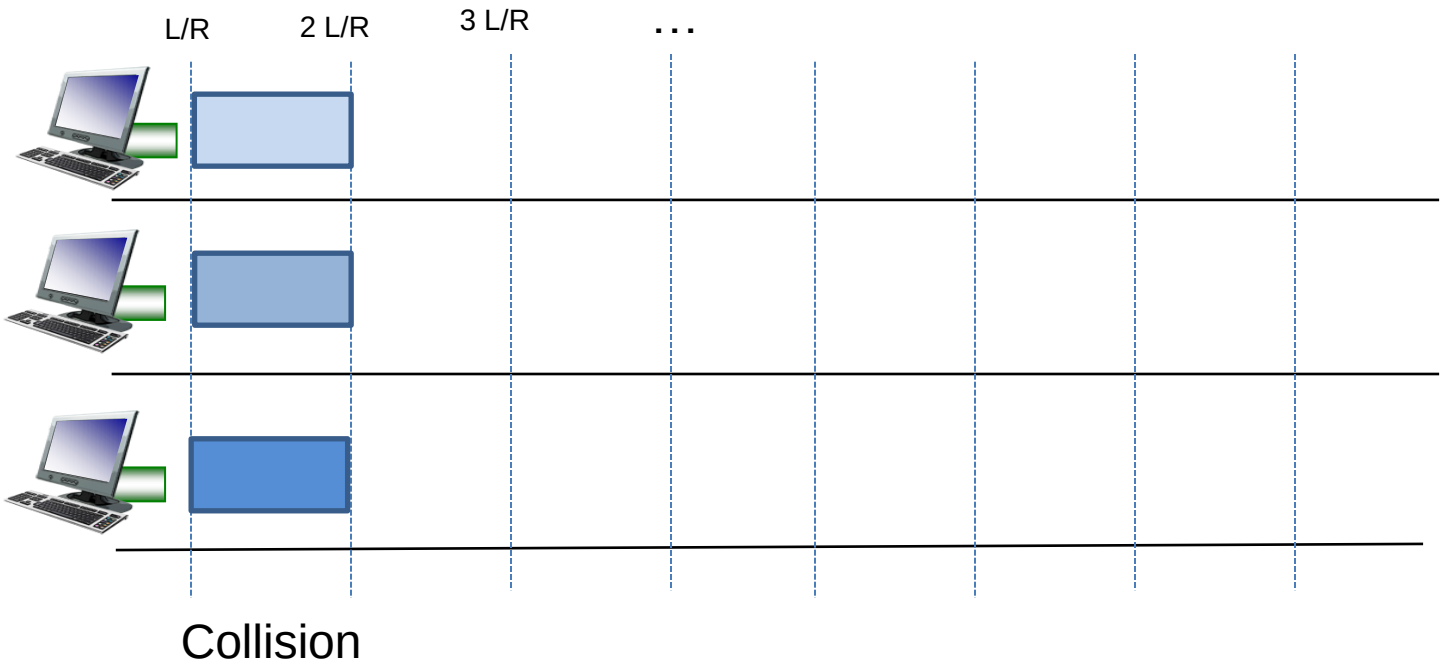
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Collision

Do some probability ***p*** and retransmit if needed

Random Access

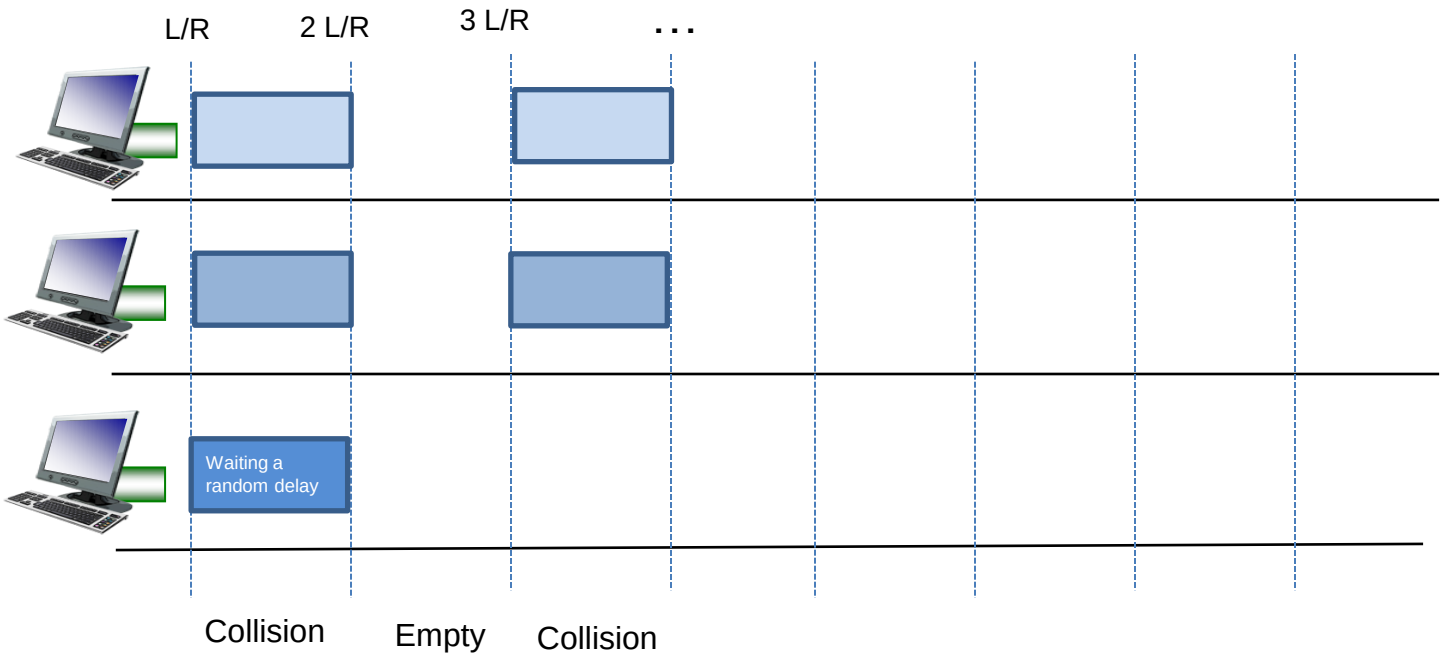
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

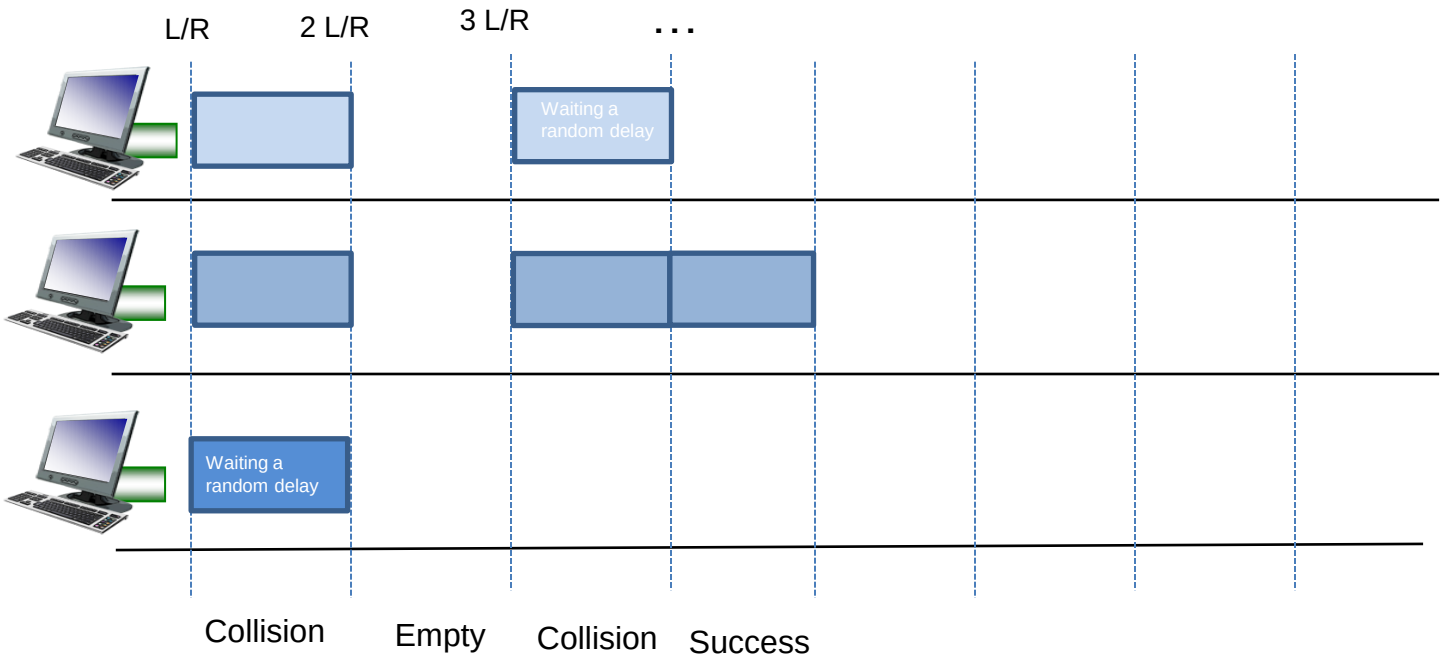
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

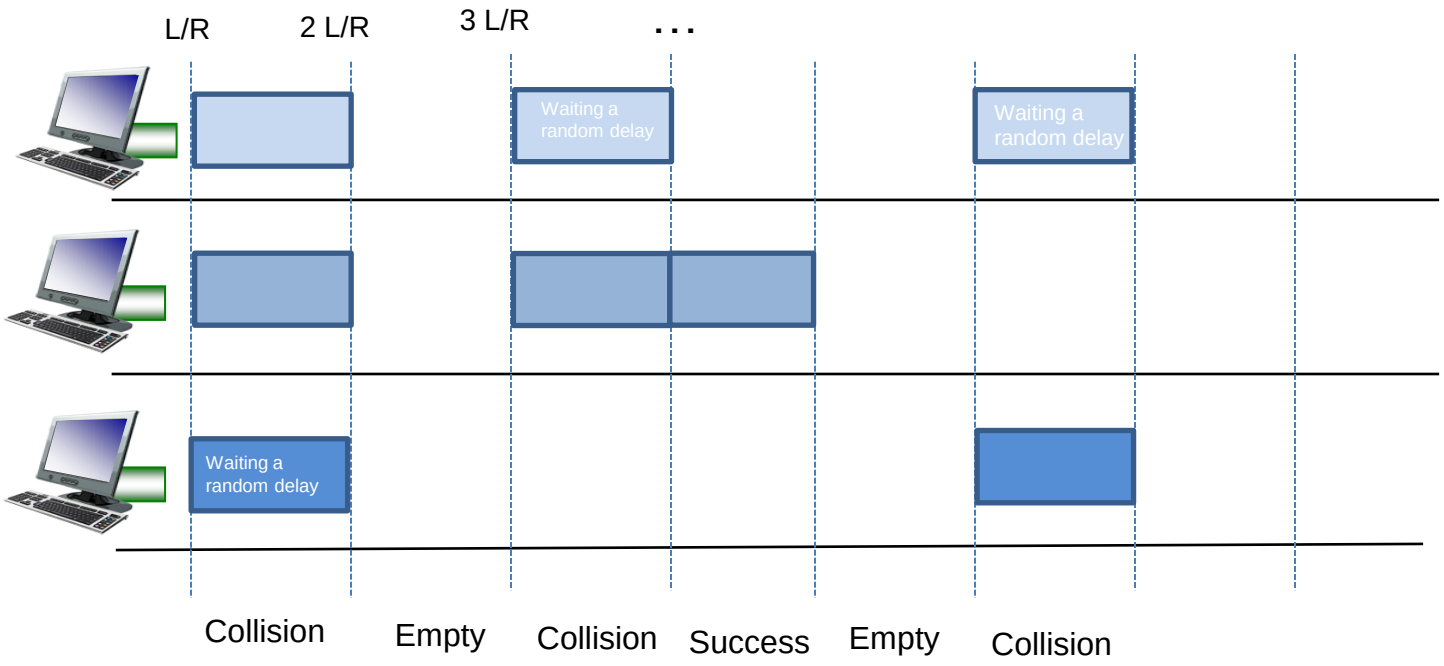
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

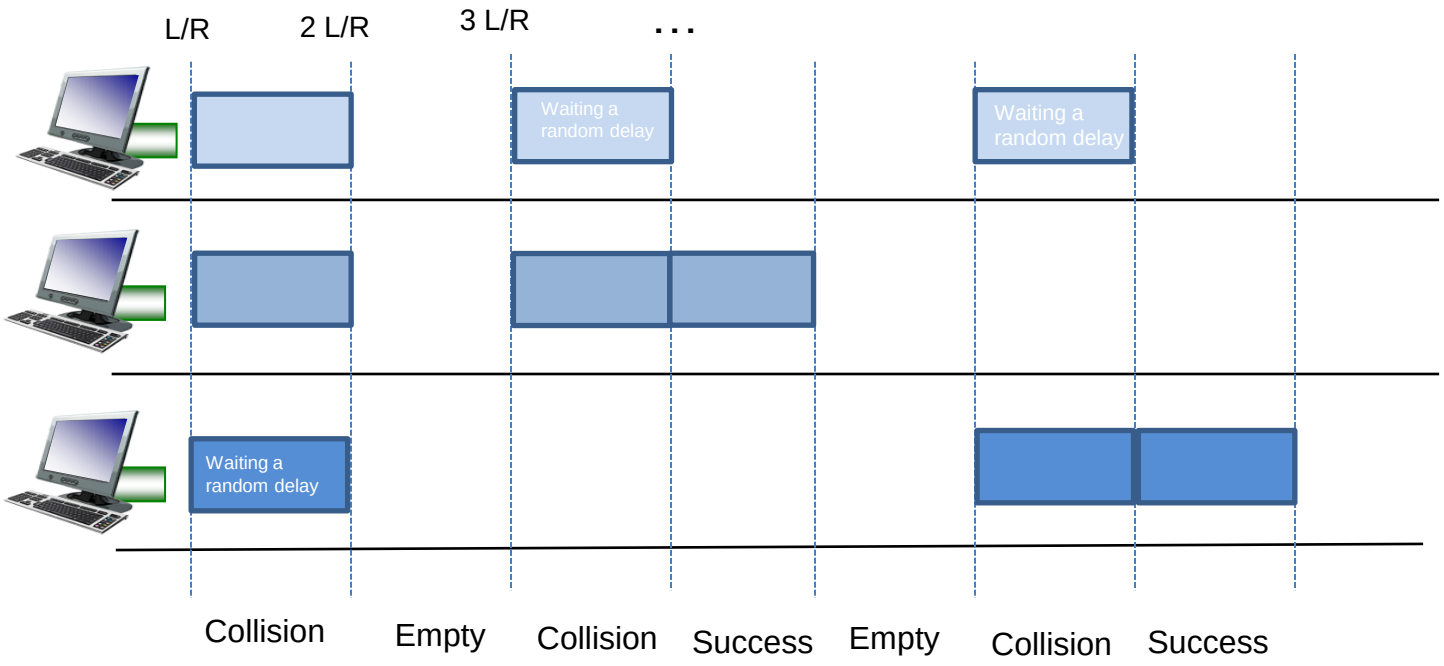
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



Random Access

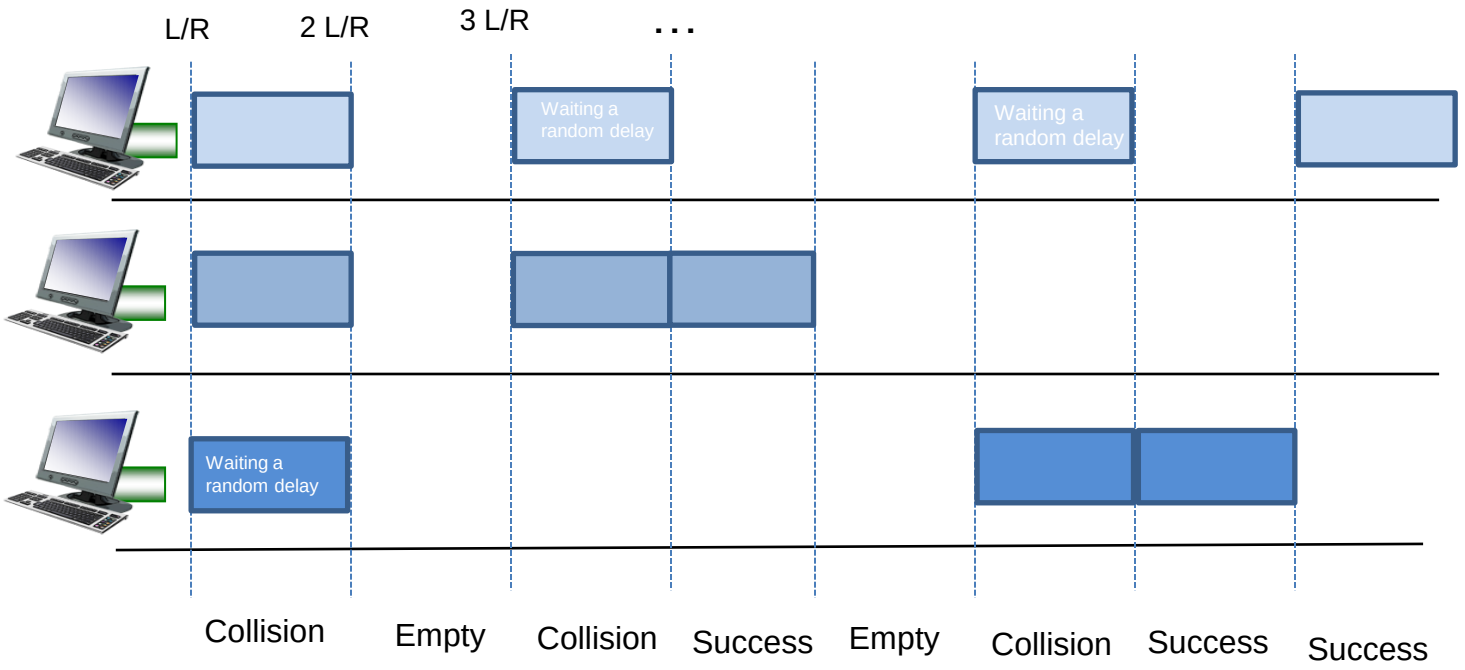
Collisions will occur, but we will try to *recover* from them

Slotted ALOHA: Divide up time into discrete L/R “slots”

If collisions occur, the colliding nodes will flip a coin to see who should retransmit

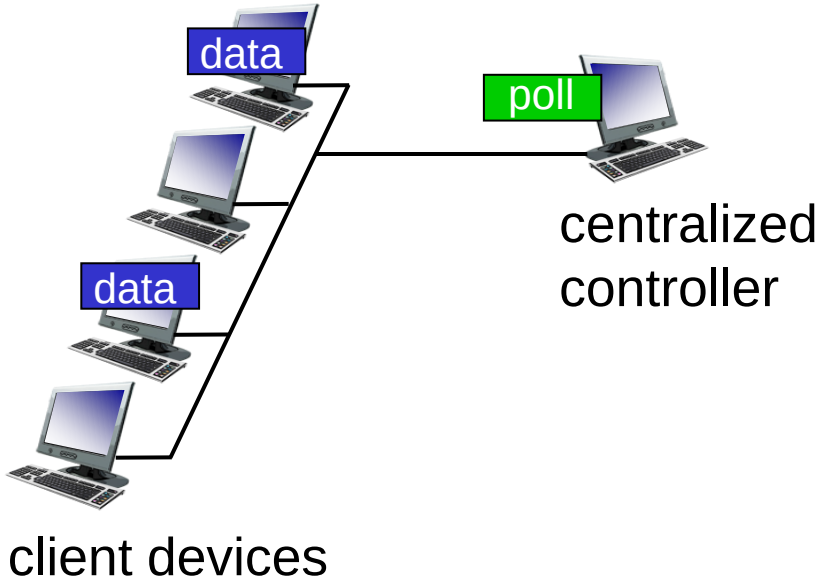
L = size of frame
R = Bandwidth

L/R = Time needed to transmit one frame



polling:

- centralized controller “invites” other nodes to transmit in turn
- typically used with “dumb” devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)
- Bluetooth uses polling



Taking Turns

Token Passing

Nodes are connected in a circular manner, and pass a special frame (token) between each other

Can only transmit messages if you have the token