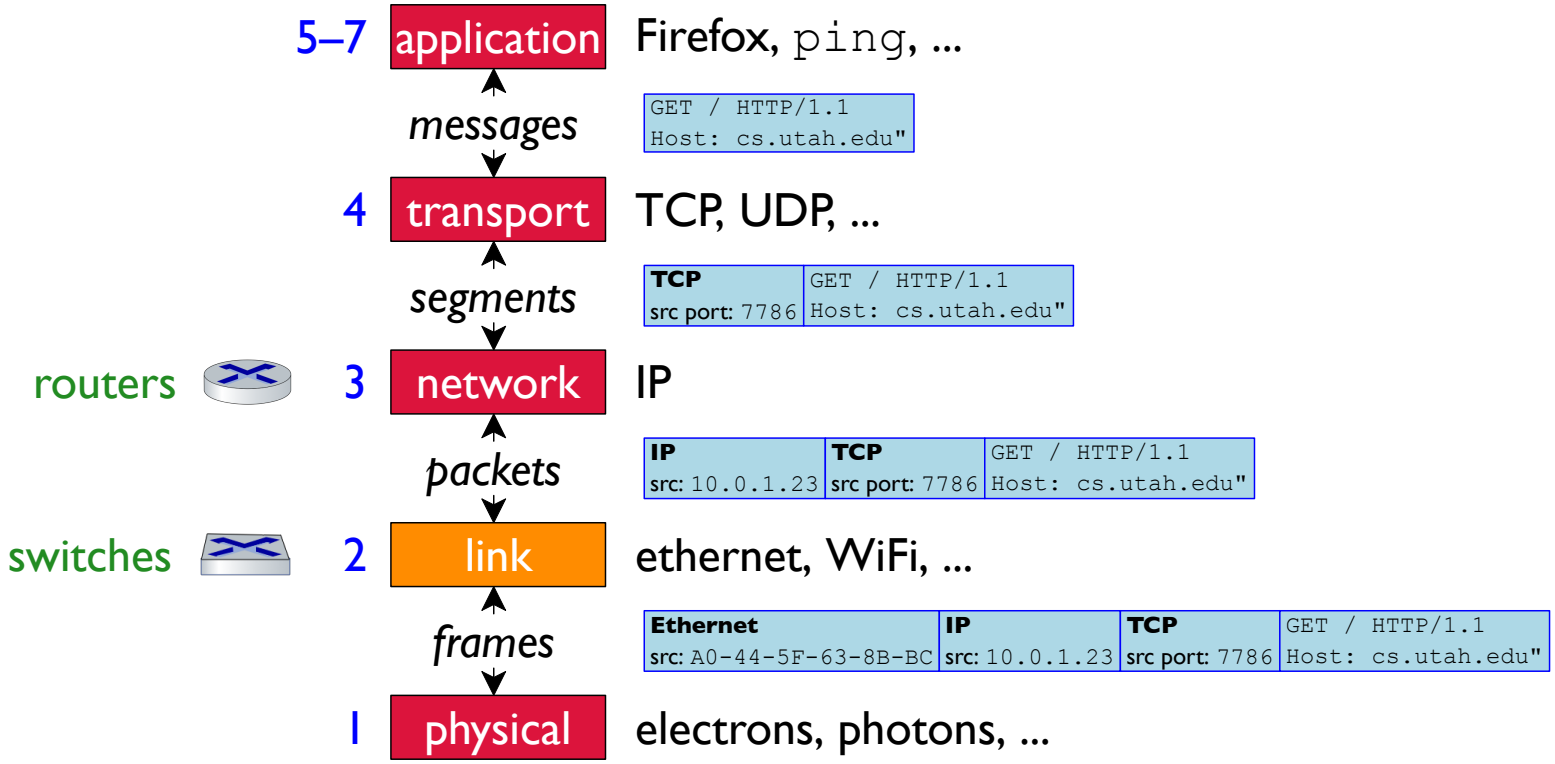
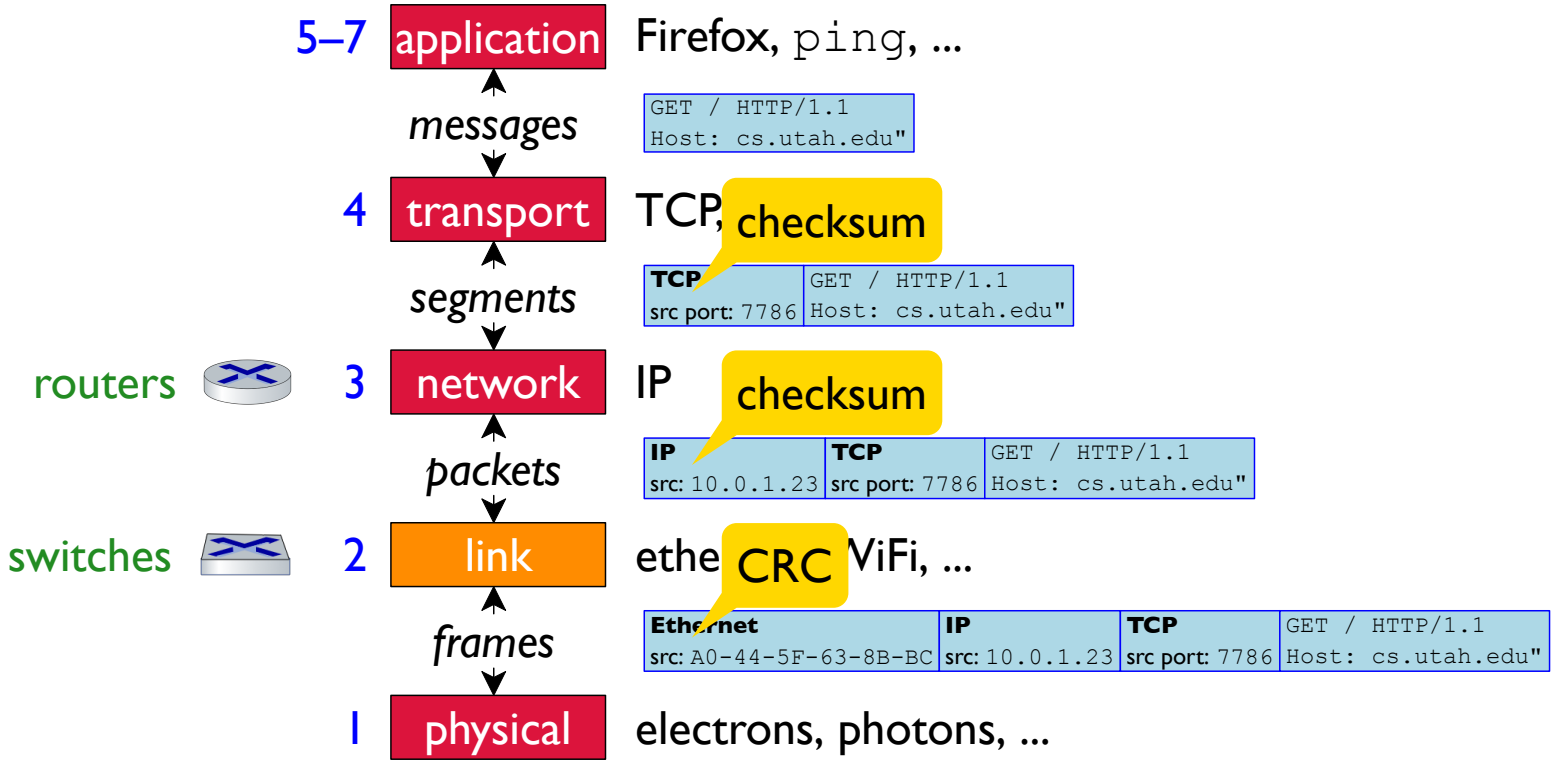
Layers


5–7
application
Firefox, ping, ...
messages
GET / HTTP/1.1
Host: cs.utah.edu"
4
transport
TCP, UDP, ...
segments
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
routers
3
network
IP
packets
IP
src: 10.0.1.23
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
switches
2
link
ethernet, WiFi, ...
frames
Ethernet
src: A0-44-5F-63-8B-BC
IP
src: 10.0.1.23
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
1
physical
electrons, photons, ...

Layers


5–7
application
Firefox, ping, ...
messages
GET / HTTP/1.1
Host: cs.utah.edu"
4
transport
TCP,
checksum
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
segments
routers
3
network
IP
checksum
IP
src: 10.0.1.23
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
packets
switches
2
link
ethe
ViFi, ...
CRC
Ethernet
src: A0-44-5F-63-8B-BC
IP
src: 10.0.1.23
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
frames
1
physical
electrons, photons, ...

Weak: Parity

A 1-bit checksum is a **parity** check

... since that 1 bit is either on/odd or off/even

```
10001010  →  1
10011010  →  0
10111010  →  1
```

Fast, but two corrupt bits cancel, which is especially bad when corruption is bursty

```
10001010  →  1
11101010  →  0
```

Better: Checksum

Adding numbers and keeping low bits is better

```
10001010        11111010
10011110        10011110
10111010        10111010
10111110        10111110
--------        --------
10100000   ≠    00010000
```

But it can miss many kinds of regular patterns

```
10001010        10001110
10011110        10011010
10111010        10111110
10111110        10111010
--------        --------
10100000   =    10100000
```

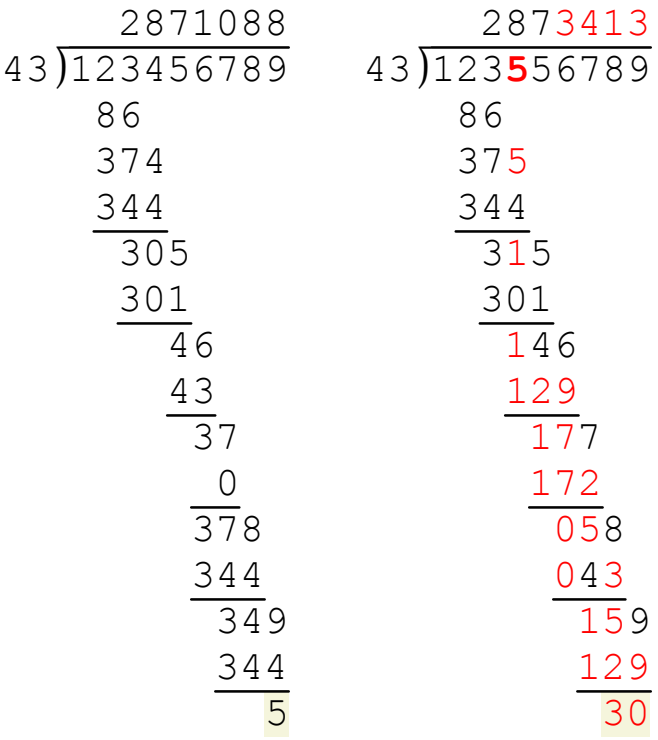
Strong: Cyclic Redundancy Check

Division is much better at mixing numbers

```
      2871088
43)123456789
   86
   374
   344
    305
    301
      46
      43
       37
        0
       378
       344
        349
        344
          5
```

Strong: Cyclic Redundancy Check

Division is much better at mixing numbers



Strong: Cyclic Redundancy Check

Division is much better at mixing numbers

```
     2871088
43)123456789
   86
   374
   344
    305
    301
     46
     43
      37
       0
      378
      344
       349
       344
         5
```

```
     2873413
43)123556789
   86
   375
   344
    315
    301
     146
     129
      177
      172
       058
       043
        159
        129
         30
```

General division is expensive on a CPU

Specialized division can be fast, especially in hardware

Strong: Cyclic Redundancy Check

A **cyclic redundancy check (CRC)** takes advantage of division:

$$R = \text{remainder of } \frac{D \times 2^r}{G}$$

d	=	number of bits to check
D	=	d bits of data
r	=	bits for hash result (typically 8, 12, 16, or 32)
G	=	a carefully chosen, agreed-on $r+1$-bit number
R	=	the result for D

For $r = 32$, IEEE standard is $G =$ `0x104C11DB7`

Strong: Cyclic Redundancy Check

A **cyclic redundancy check (CRC)** takes advantage of division:

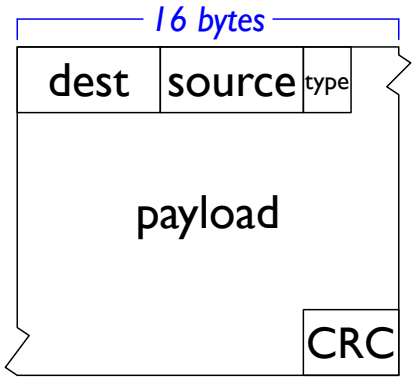
$$R = \text{remainder of } \frac{D \times 2^r}{G}$$

d	=	number of bits to check
D	=	d bits of data
r	=	bits for hash result (typically 8, 12, 16, or 32)
G	=	a carefully chosen, agreed-on $r+1$-bit number
R	=	the result for D

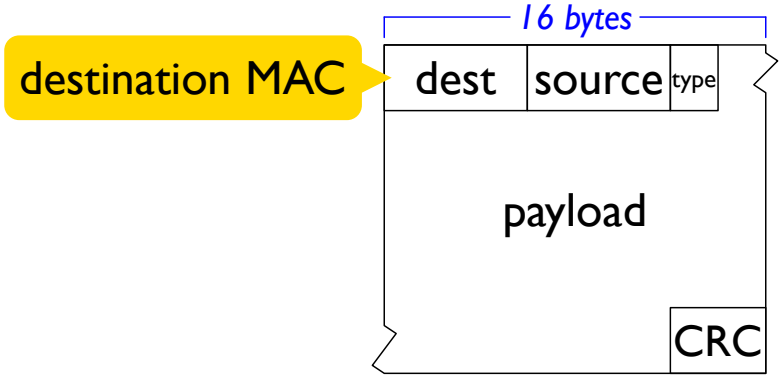
For $r = 32$, IEEE standard is $G =$ `0x104C11DB7`

Detects any r-bit error burst

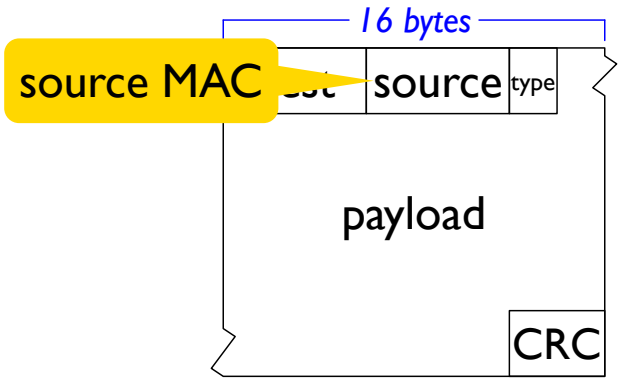
Ethernet Frame Layout



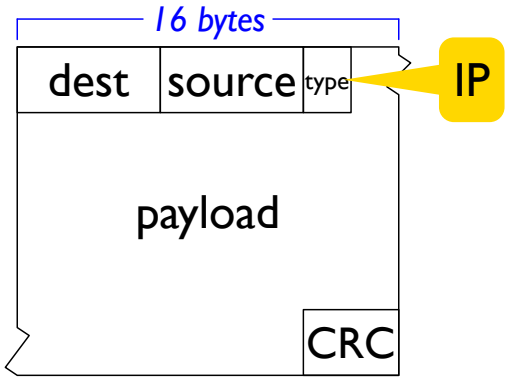
Ethernet Frame Layout



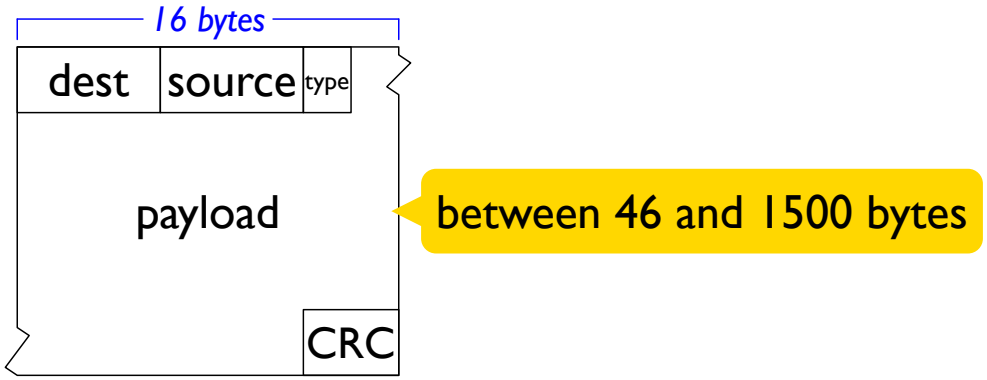
Ethernet Frame Layout



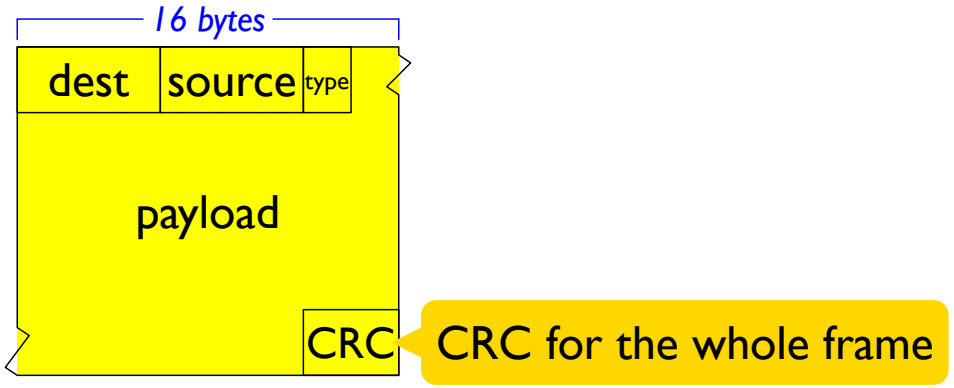
Ethernet Frame Layout



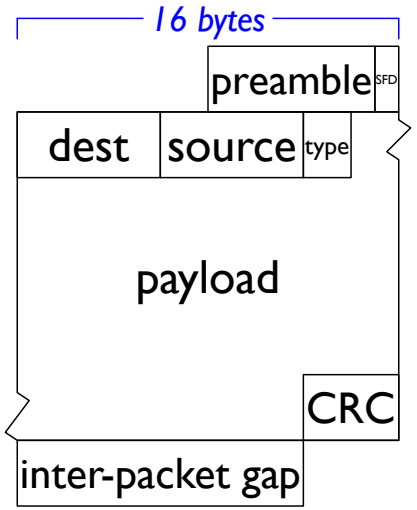
Ethernet Frame Layout



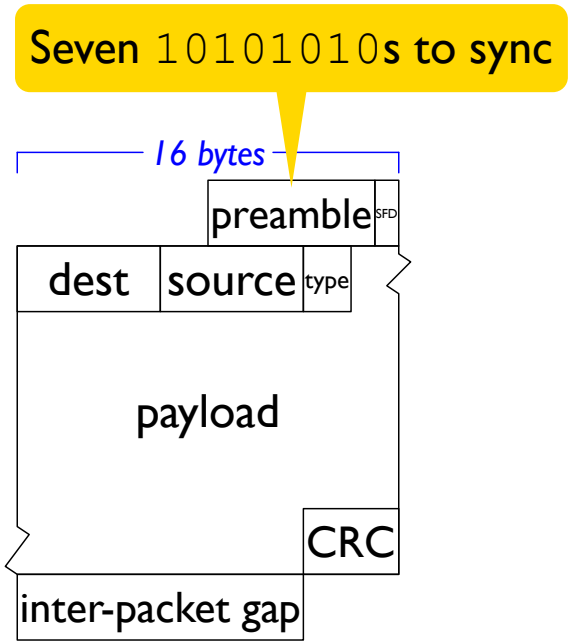
Ethernet Frame Layout



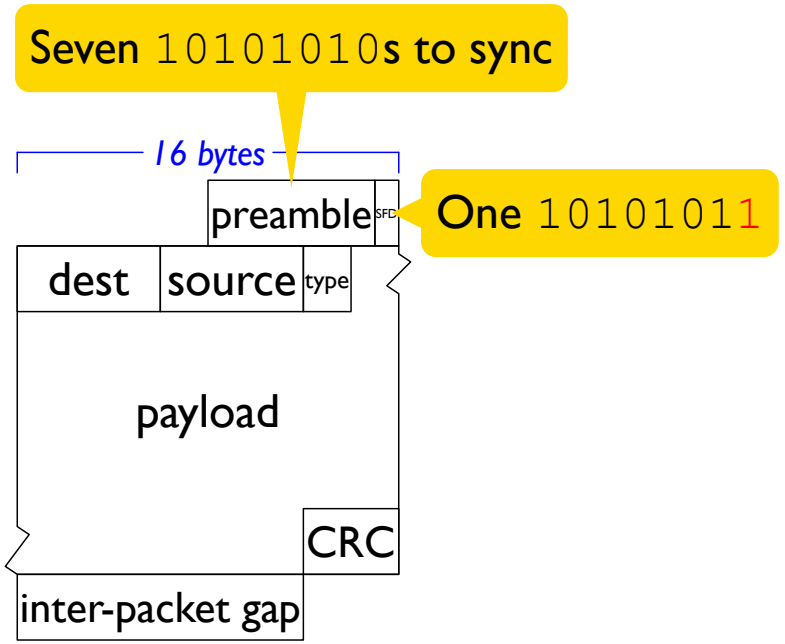
Ethernet Physical Layer Layout



Ethernet Physical Layer Layout



Ethernet Physical Layer Layout



Ethernet Physical Layer

Originally, machines on an ethernet LAN shared a wire

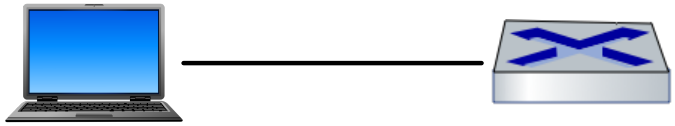
Modern ethernet is always **switched**: there's a dedicated wire from each machine to the switch

This makes MAC addresses somewhat redundant!

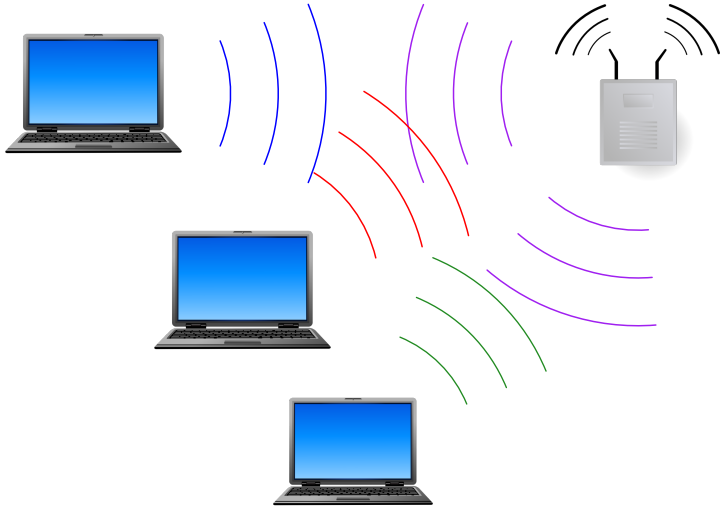
Many ethernet frames are not physically ethernet at all, because the easiest way to create a new physical layer is to emulate an ethernet device

Coordinating Communication

Easy mode: point-to-point communication

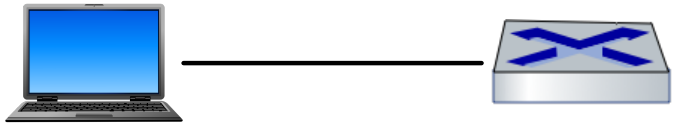


Hard mode: shared communication medium

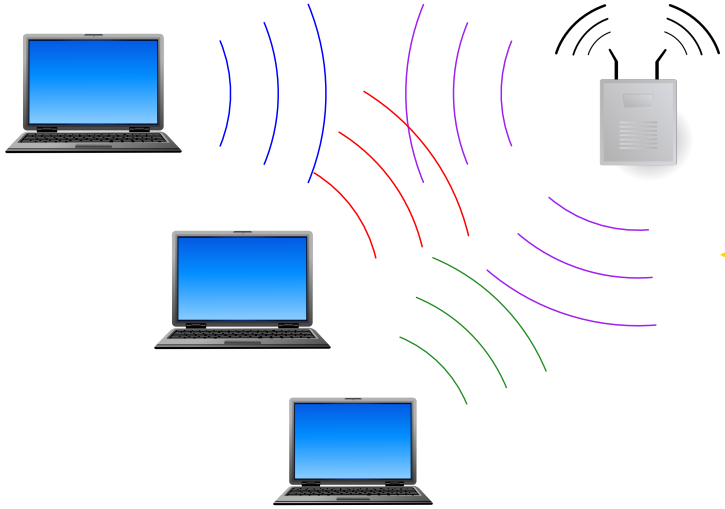


Coordinating Communication

Easy mode: point-to-point communication



Hard mode: shared communication medium

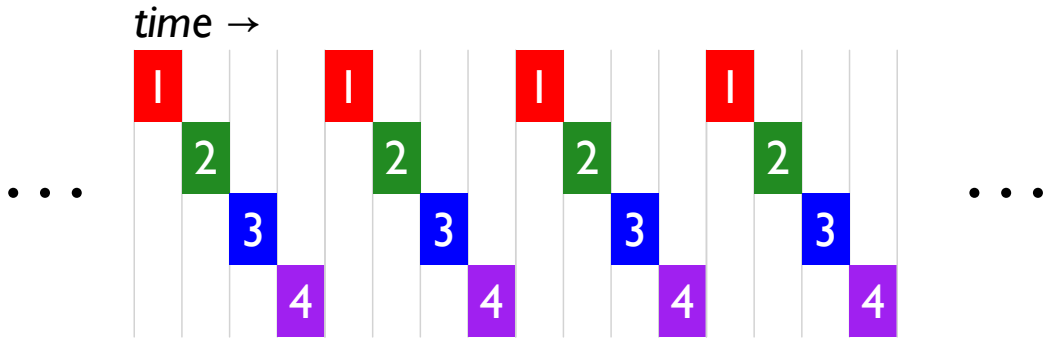


Goals:
- divide bandwidth fairly
- only one active

  ⇒ gets full bandwidth

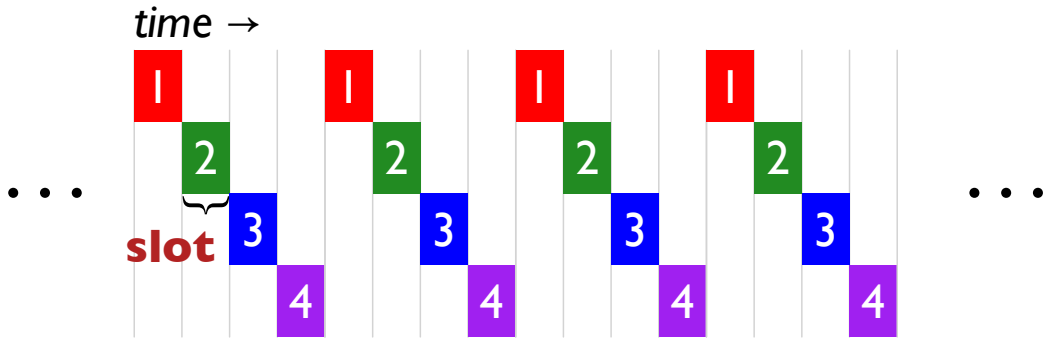
Shared-Medium Strategy 1: Channel Partitioning

Time-division multiplexing (TDM):



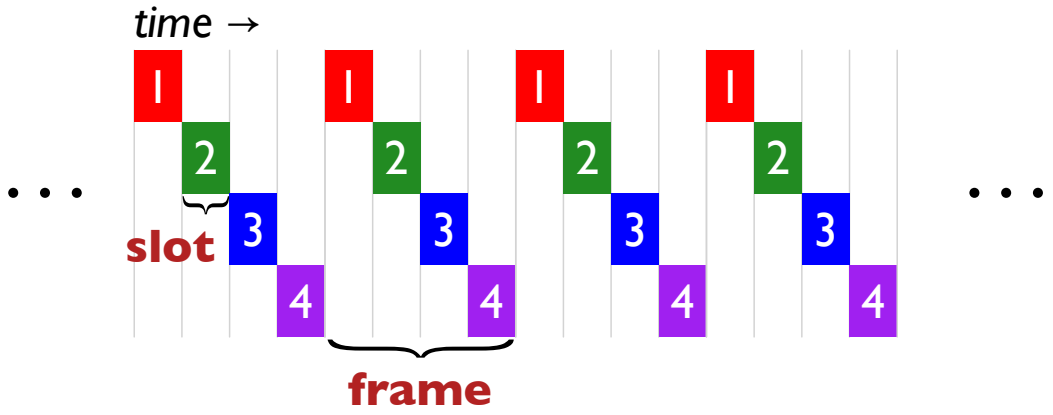
Shared-Medium Strategy 1: Channel Partitioning

Time-division multiplexing (TDM):



Shared-Medium Strategy 1: Channel Partitioning

Time-division multiplexing (TDM):

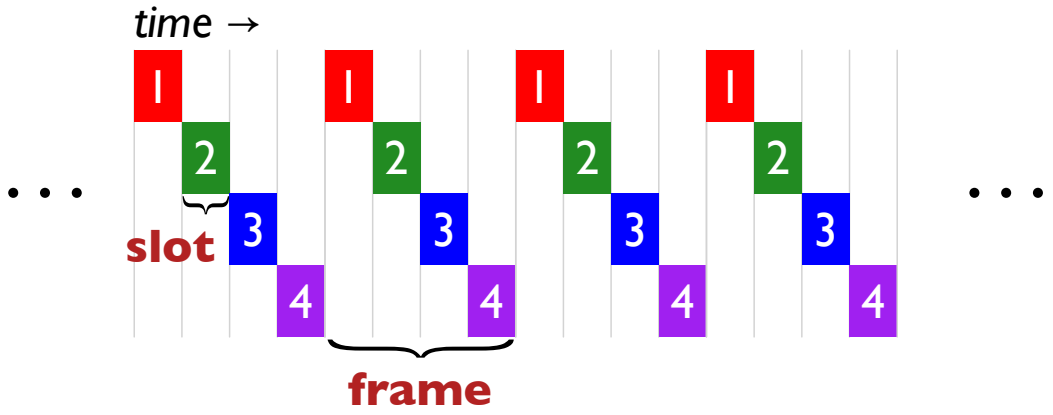


Frequency-division multiplexing (FDM):

same idea, but for simultaneous frequencies

Shared-Medium Strategy 1: Channel Partitioning

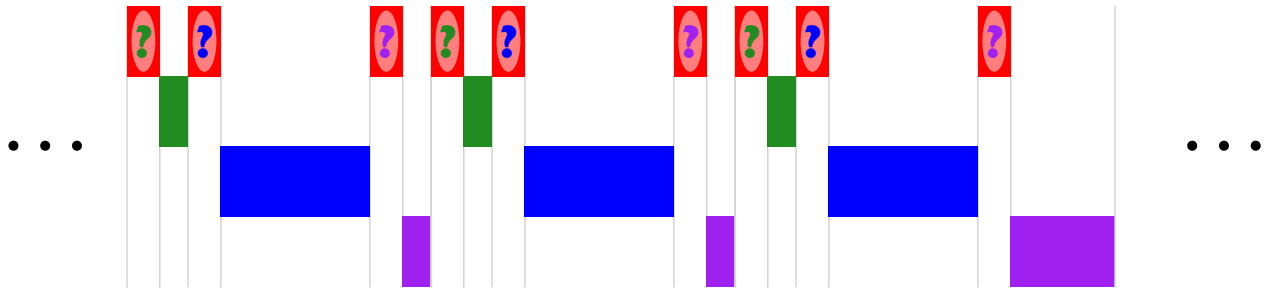
Time-division multiplexing (TDM):



+ No collisions
+ Perfectly fair
– Poor utilization when some are idle

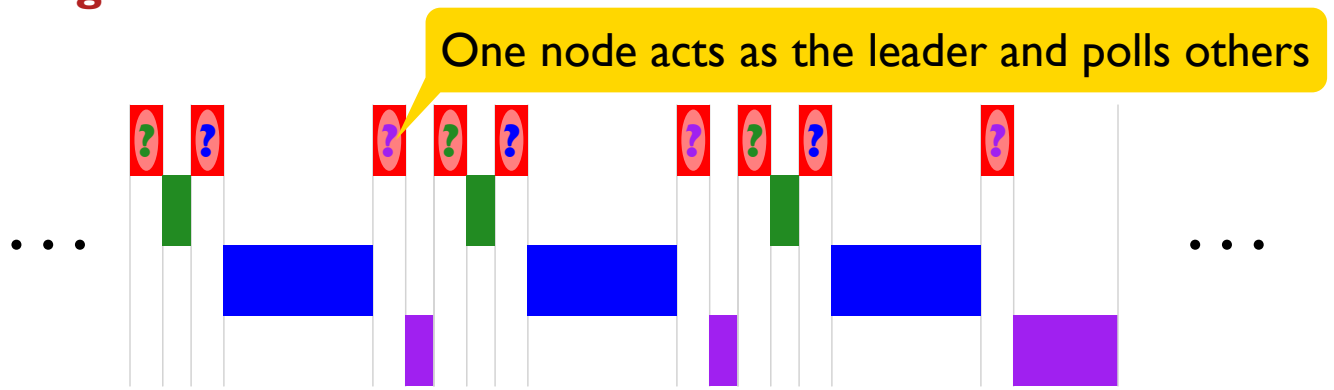
Shared-Medium Strategy 2: Turn Taking

Polling:



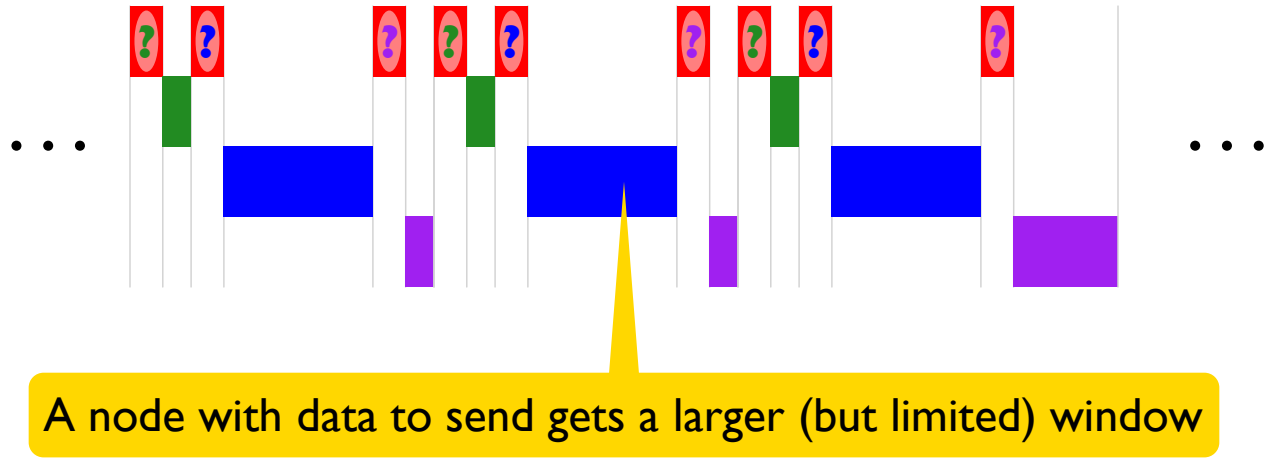
Shared-Medium Strategy 2: Turn Taking

Polling:



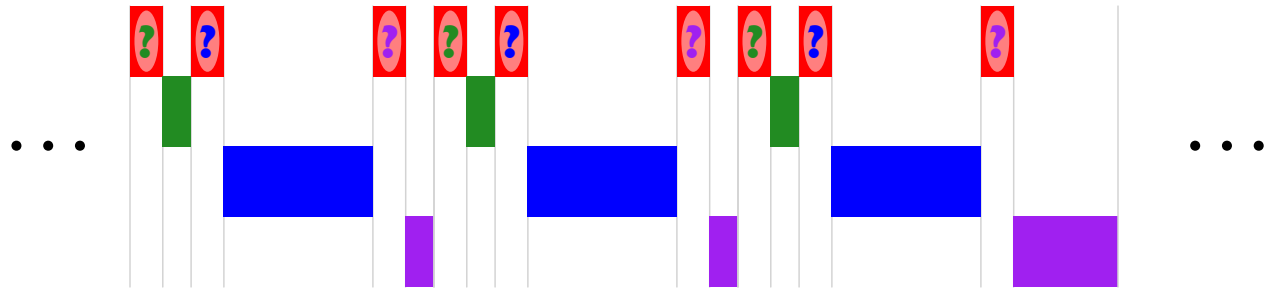
Shared-Medium Strategy 2: Turn Taking

Polling:



Shared-Medium Strategy 2: Turn Taking

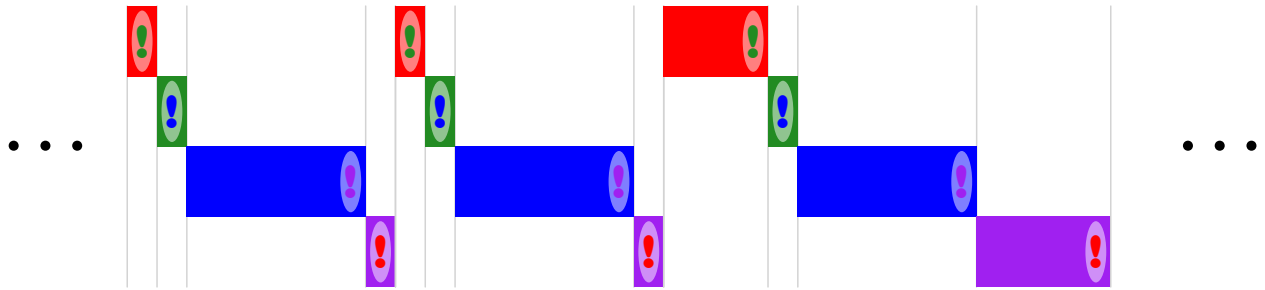
Polling:



+ Better utilization
– Polling causes delays
– Recovery needed if the leader fails

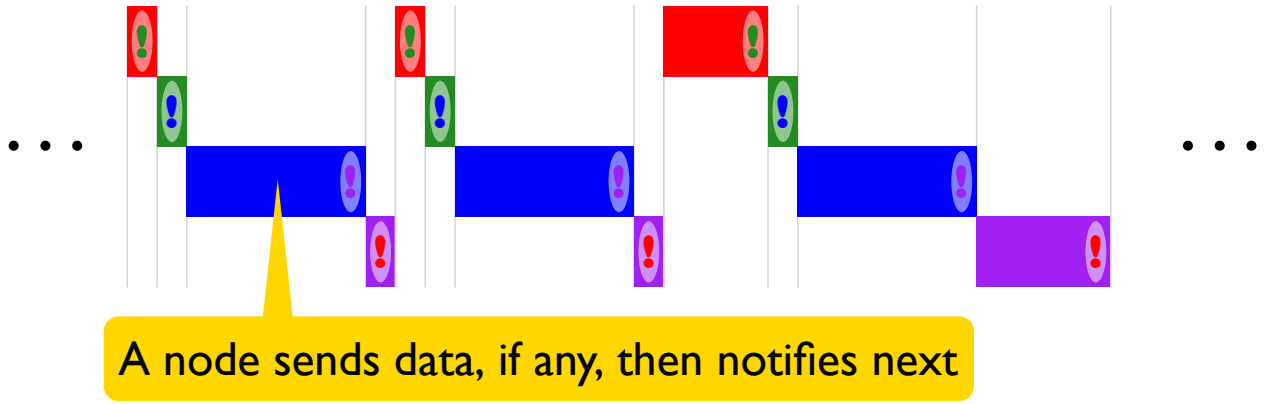
Shared-Medium Strategy 2: Turn Taking

Token passing:



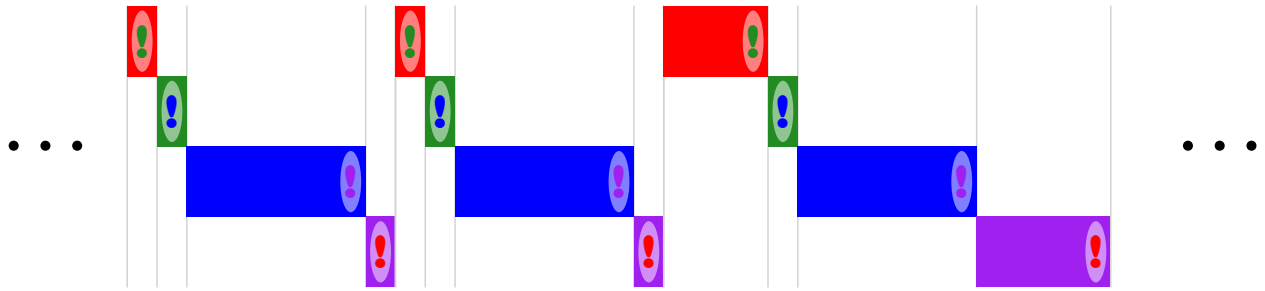
Shared-Medium Strategy 2: Turn Taking

Token passing:



Shared-Medium Strategy 2: Turn Taking

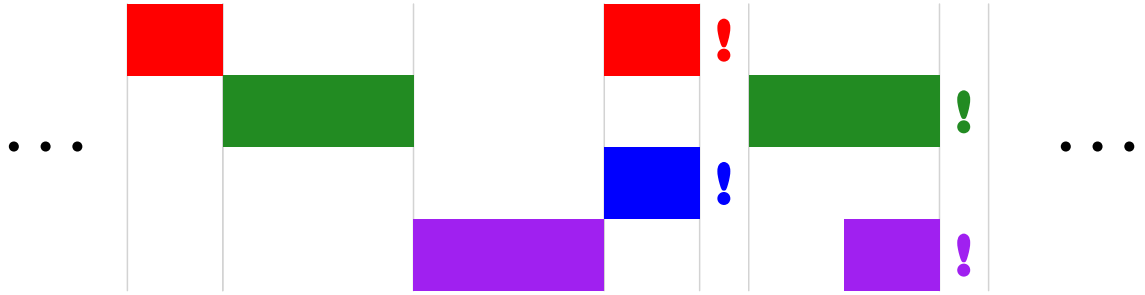
Token passing:



+ Better utilization
- Token-passing causes delays
- Recovery needed if any fails

Shared-Medium Strategy 3: Random Access

Random access:

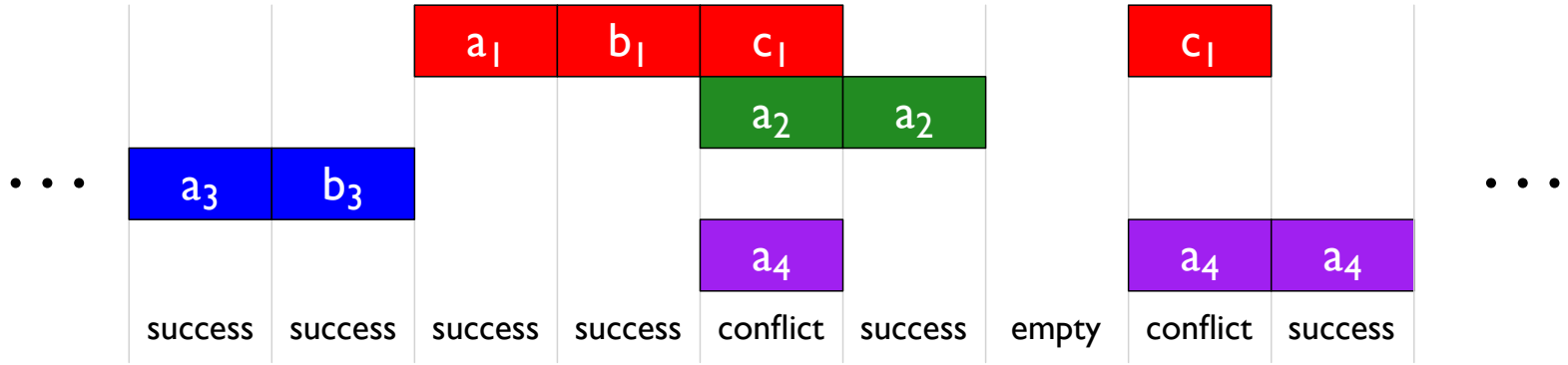


Need collision detection and/or **carrier sense**

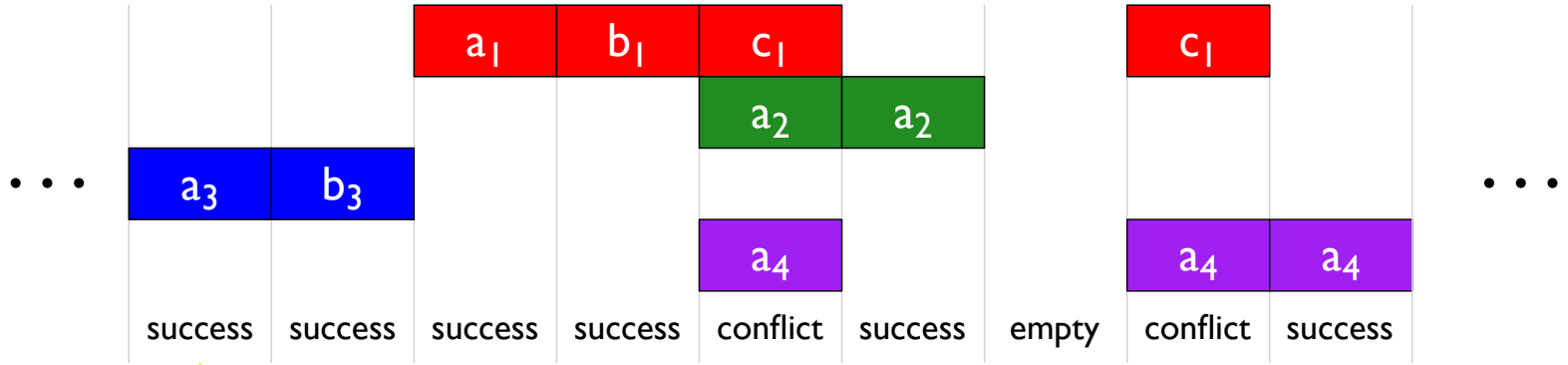
Random delay when collision is detected

\+ Potentially better utilization

Random Access: ALOHA

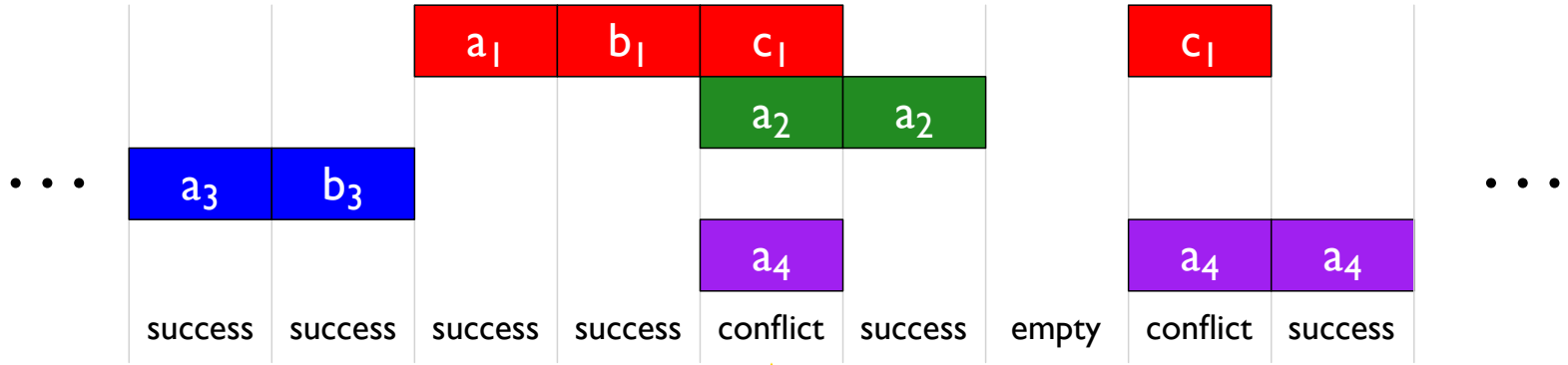


Random Access: ALOHA



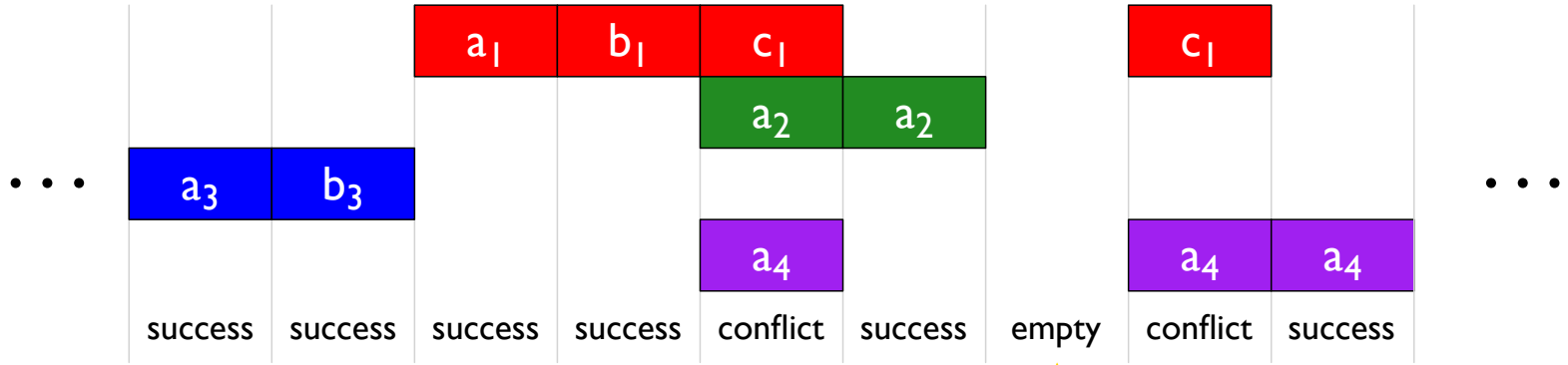
On success, a node can keep sending as long as it has data

Random Access: ALOHA

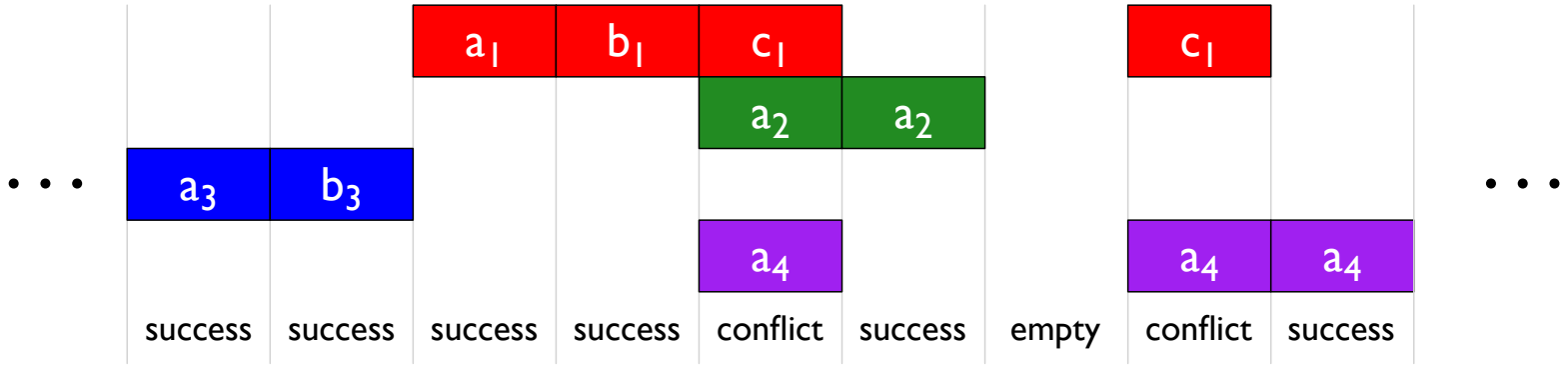


On conflict, each node retries on next slot probability P

Random Access: ALOHA



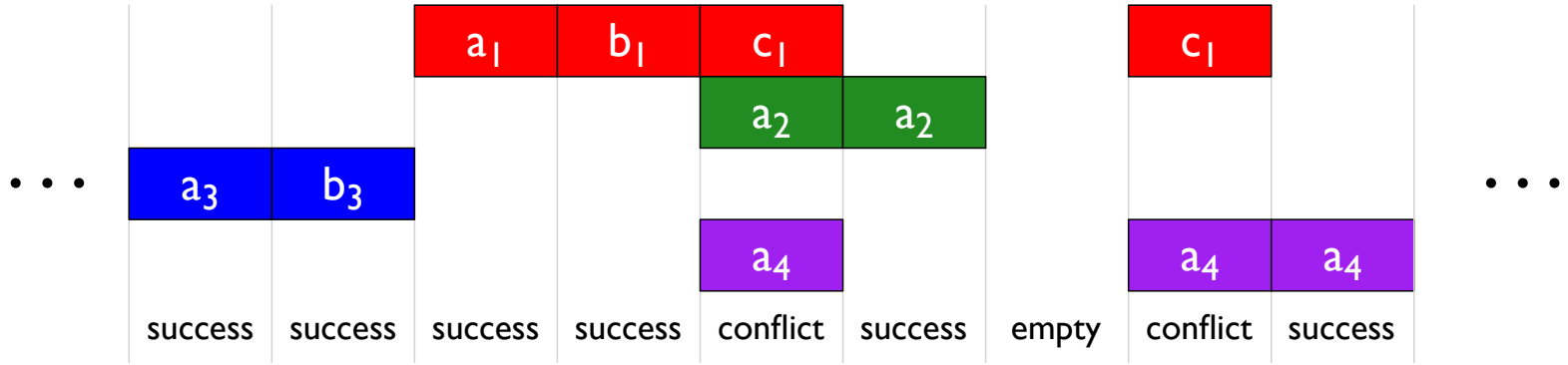
Random Access: ALOHA



Slotted ALOHA, which needs synchronization:

+ Sole active nodes can use full bandwidth
+ Multiple active nodes get fair share
– Even after optimizing P, likely to get only 37% success

Random Access: ALOHA



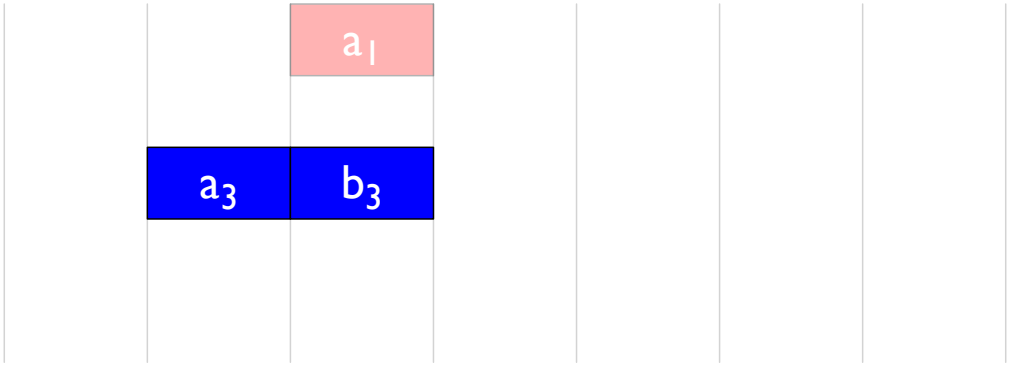
Original **unslotted ALOHA** avoided synchronization:

- Success drop drops by half

because each local slot likely overlaps two other peer slots

Random Access: Carrier Sense

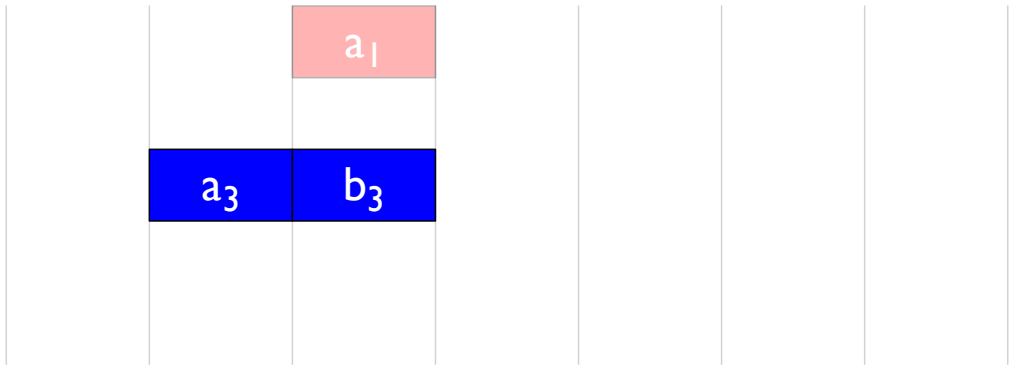
Carrier Sense Multiple Access (CSMA) means
"don't talk when someone else is talking"



Random Access: Carrier Sense

Carrier Sense Multiple Access (CSMA) means
"don't talk when someone else is talking"

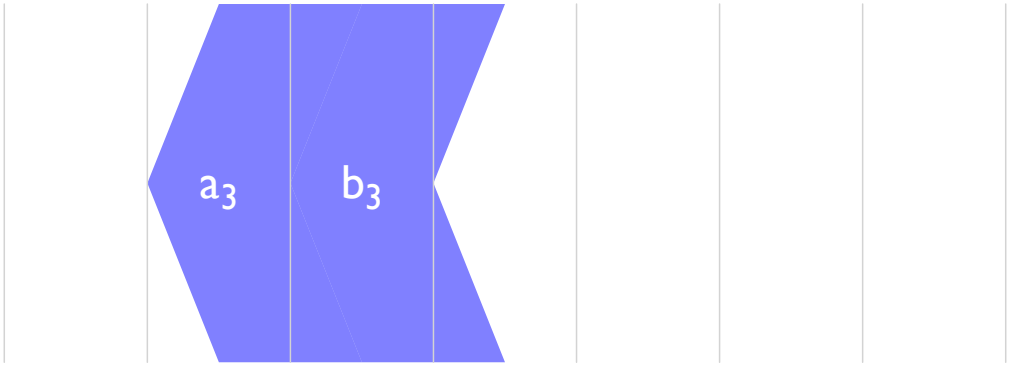
The catch: there's a delay between the time that one node sends and another node starts to sense it



Random Access: Carrier Sense

Carrier Sense Multiple Access (CSMA) means
"don't talk when someone else is talking"

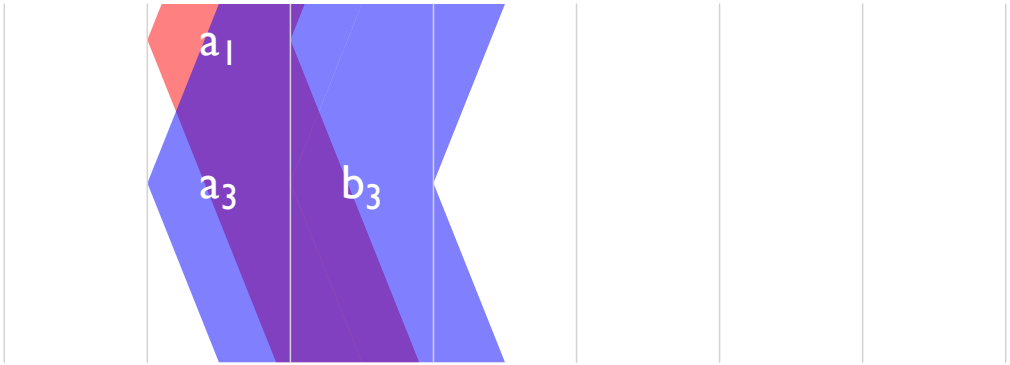
The catch: there's a delay between the time that one node sends and another node starts to sense it



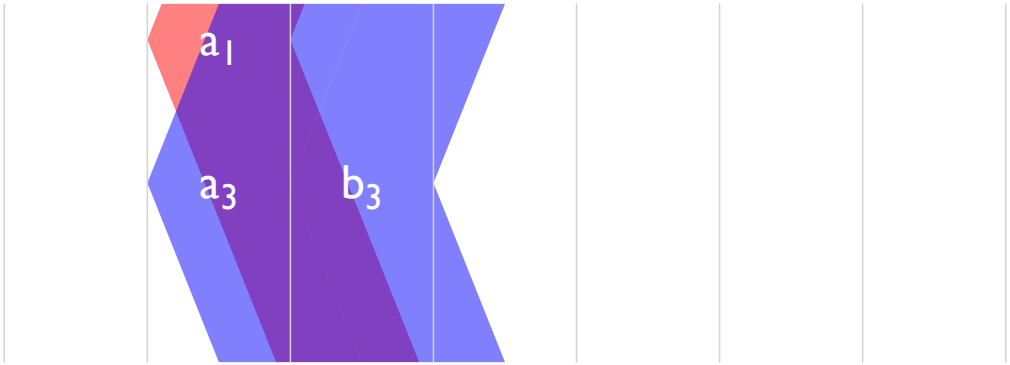
Random Access: Carrier Sense

Carrier Sense Multiple Access (CSMA) means
"don't talk when someone else is talking"

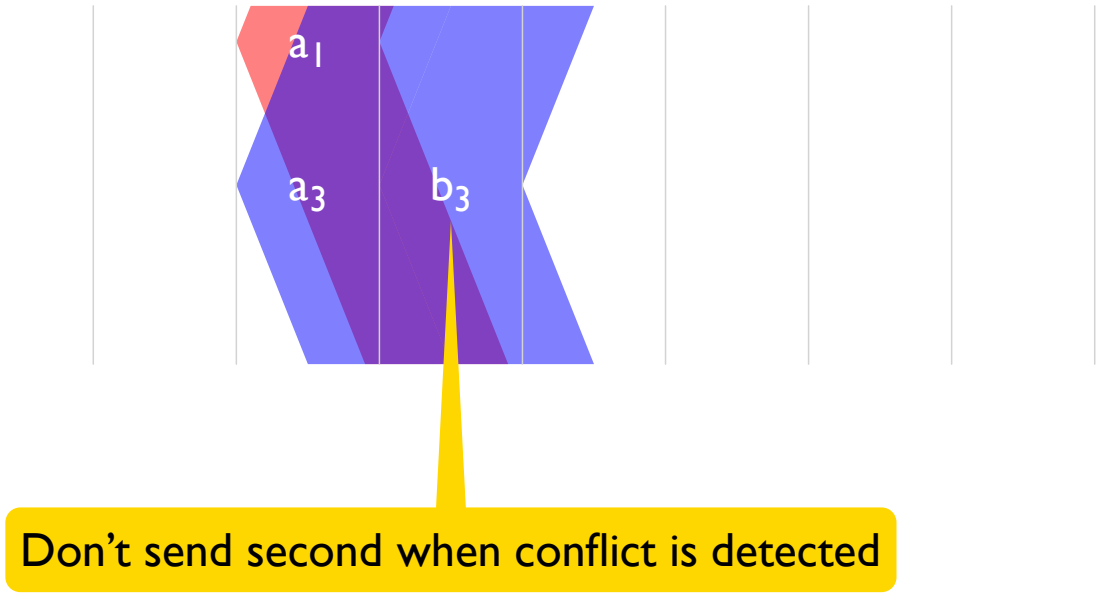
The catch: there's a delay between the time that one node sends and another node starts to sense it



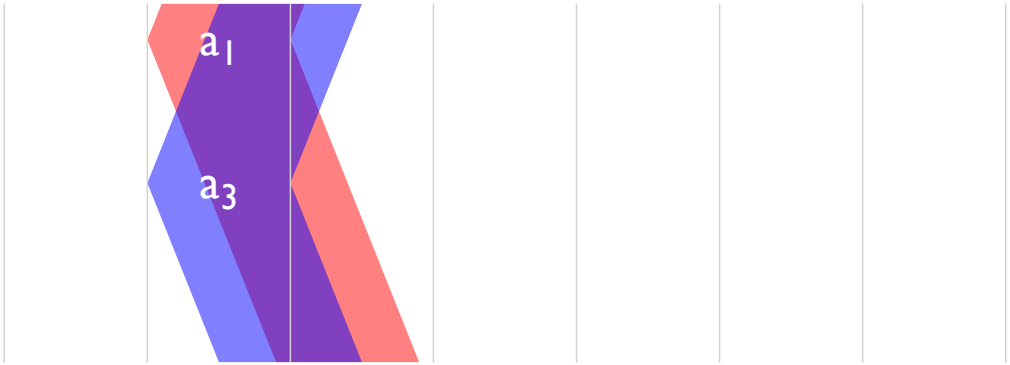
Handling Conflicts in CSMA/CD



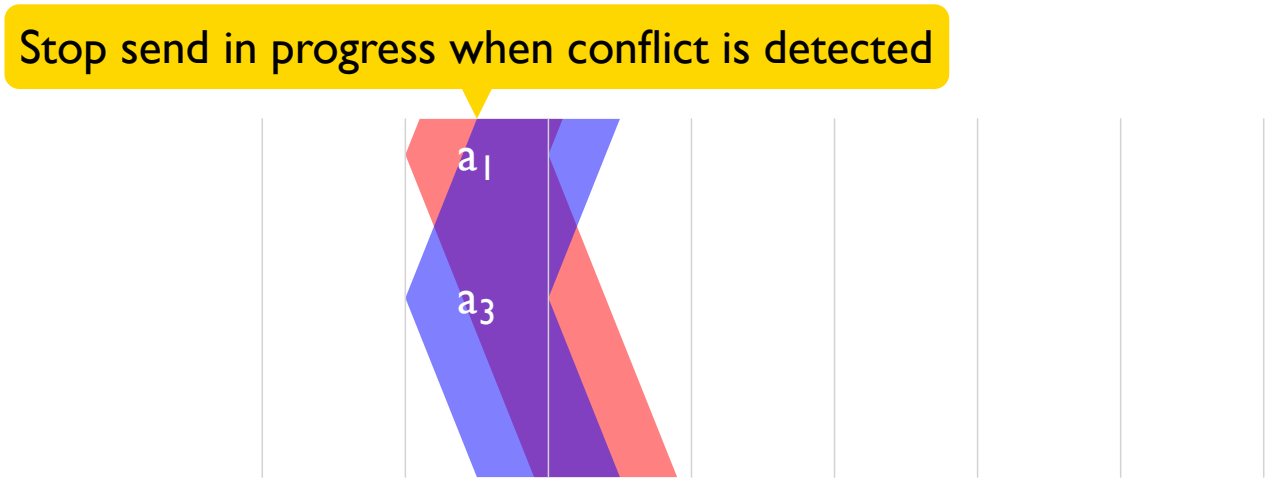
Handling Conflicts in CSMA/CD



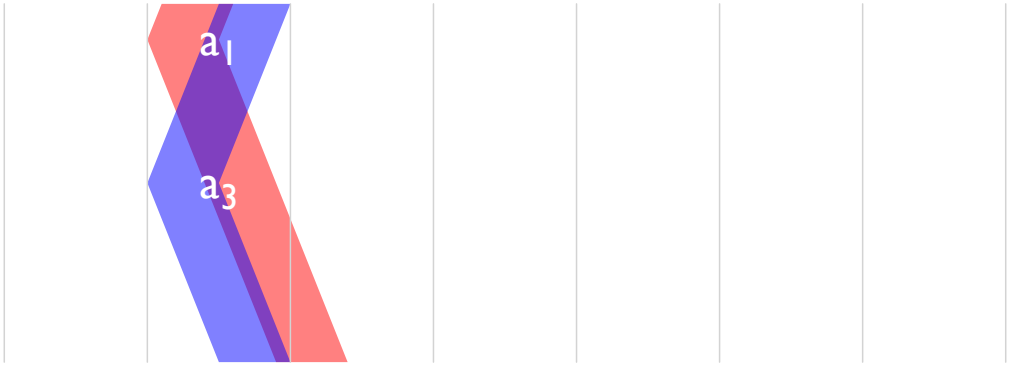
Handling Conflicts in CSMA/CD



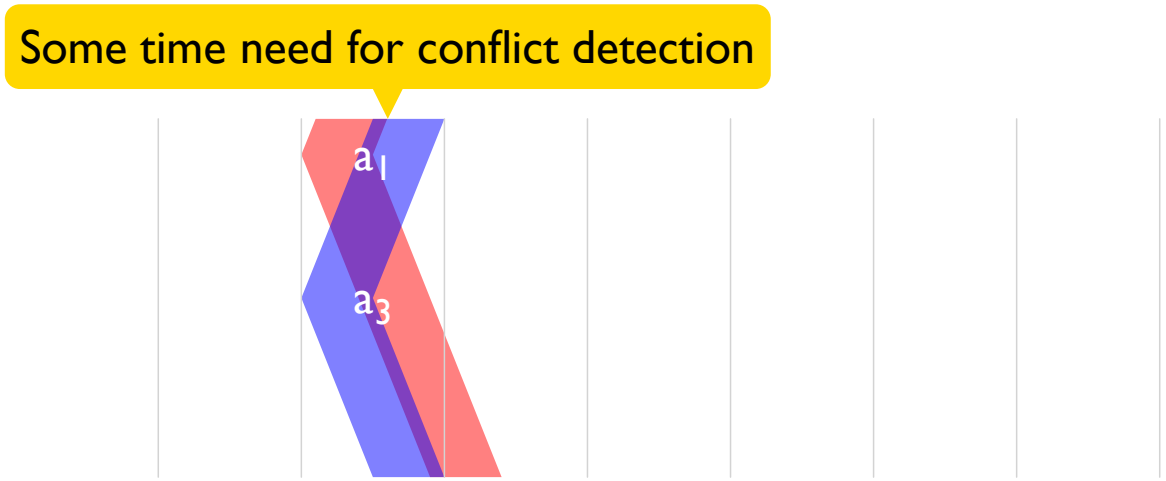
Handling Conflicts in CSMA/CD



Handling Conflicts in CSMA/CD

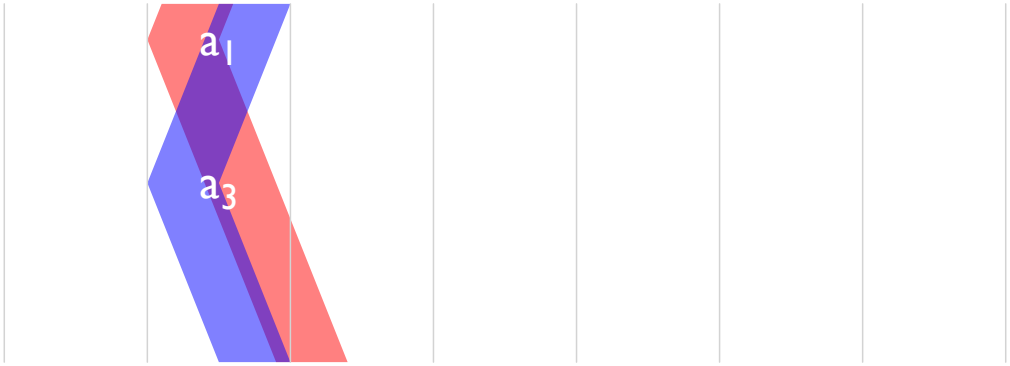


Handling Conflicts in CSMA/CD

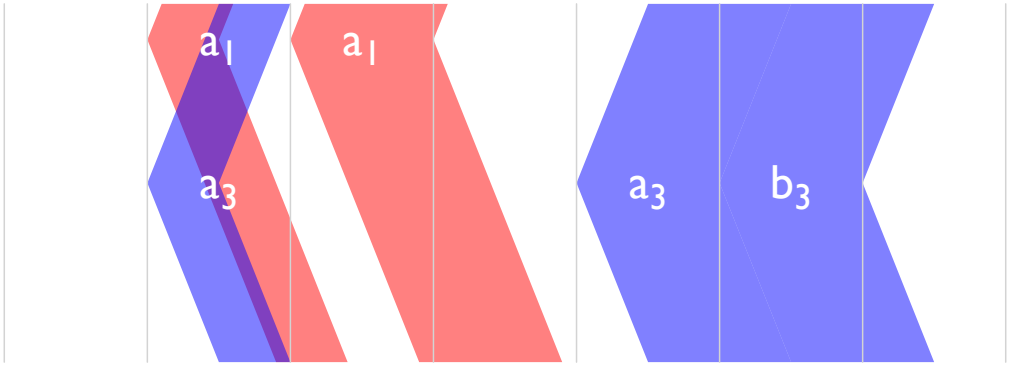


Handling Conflicts in CSMA/CD

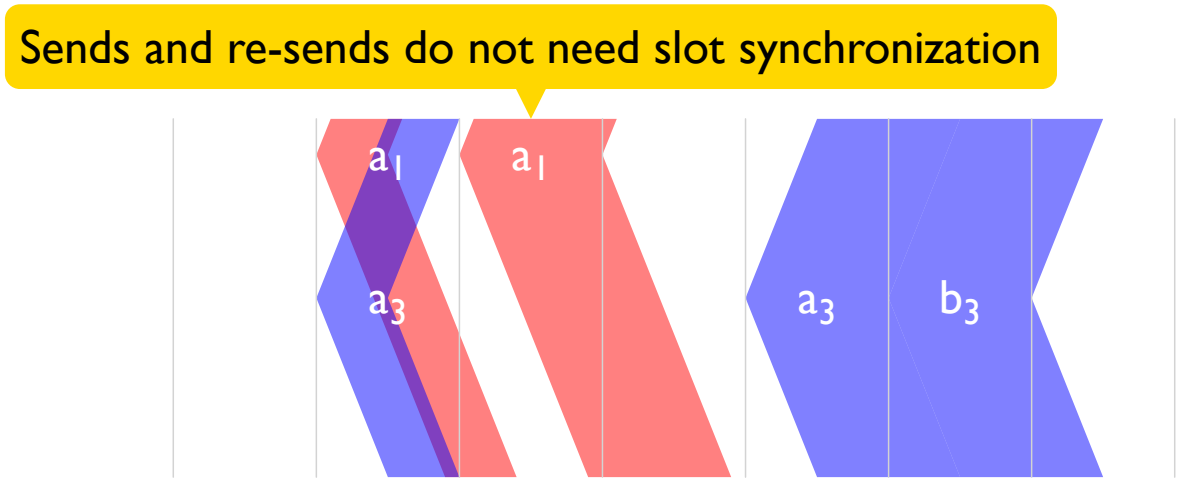
Random delay before retry



Handling Conflicts in CSMA/CD

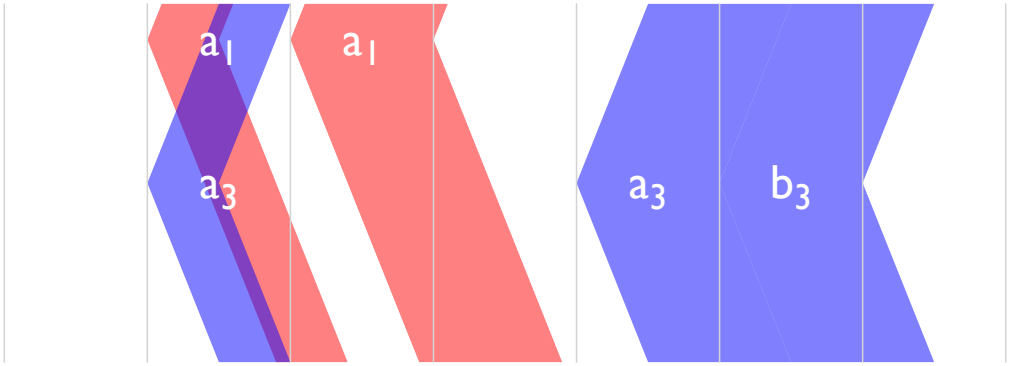


Handling Conflicts in CSMA/CD

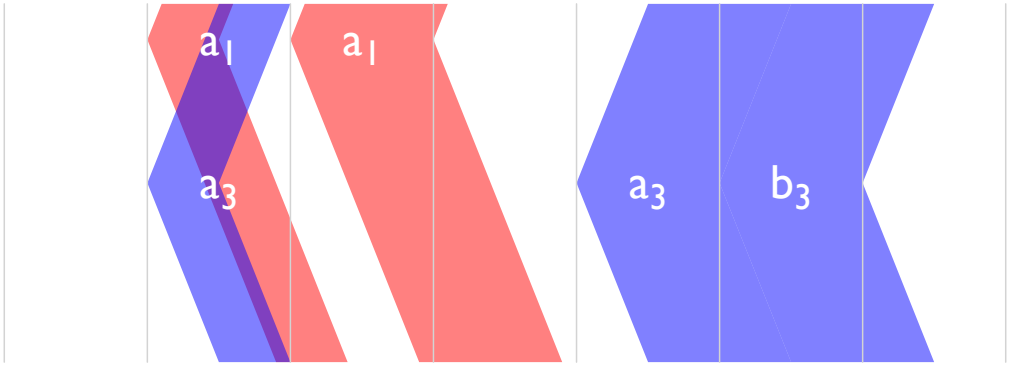


Handling Conflicts in CSMA/CD

Exponential back-off:
If another conflict, double average retry delay



Handling Conflicts in CSMA/CD

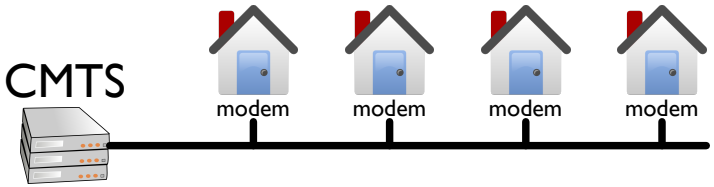


d_{prop} = max delay for signal

d_{trans} = max duration for frame

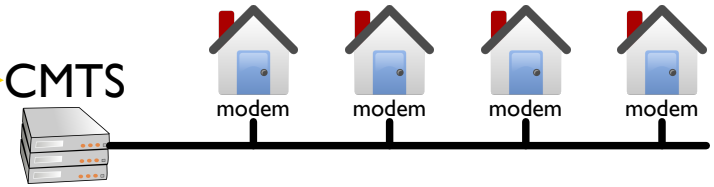
$$\text{efficiency} = \frac{1}{1 + 5\frac{d_{prop}}{d_{trans}}}$$

DOCSIS Cable Internet Protocol

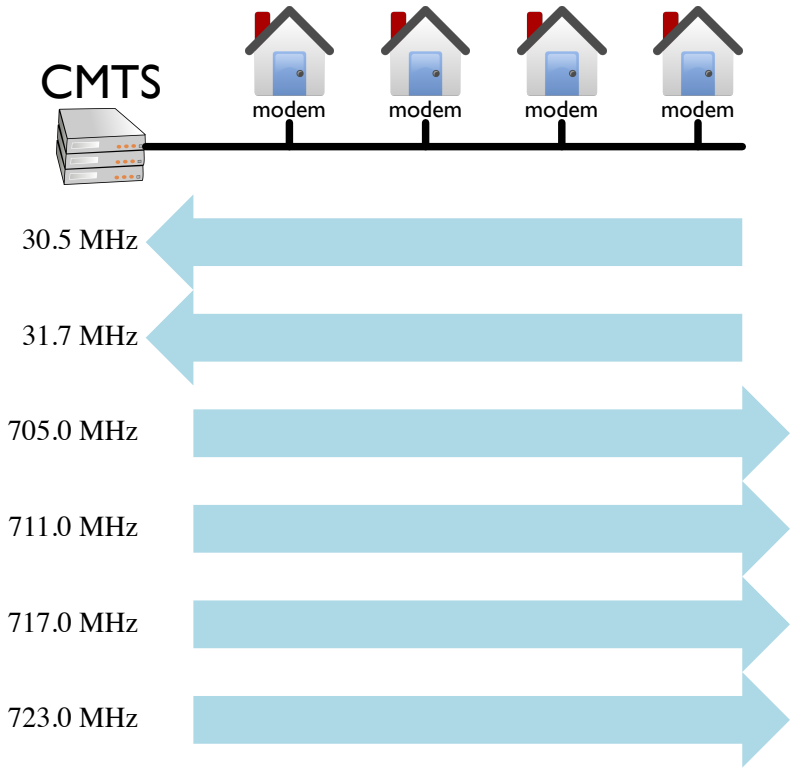


DOCSIS Cable Internet Protocol

Cable modem termination system (CMTS)
is in control



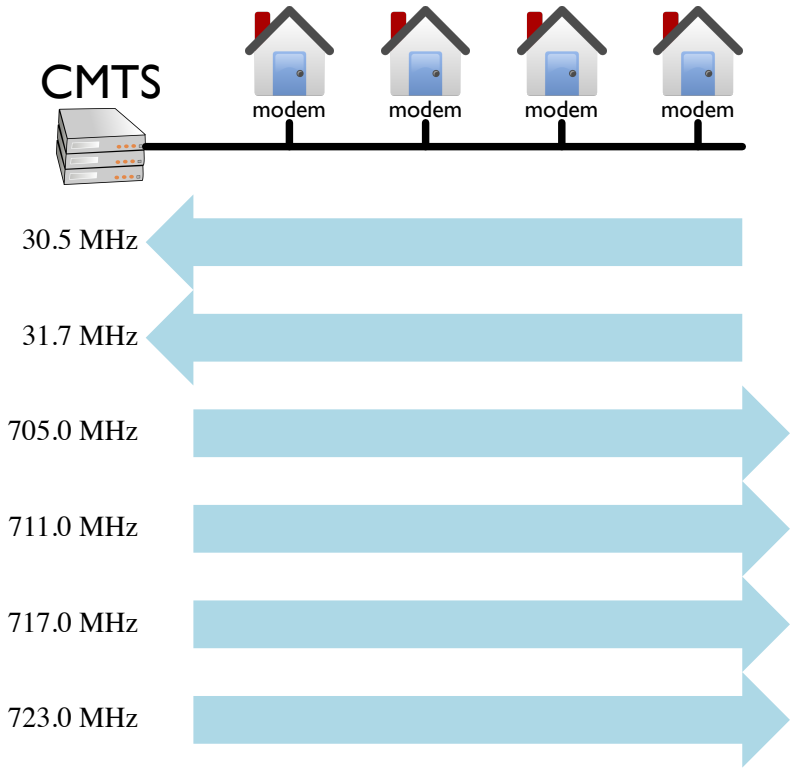
DOCSIS Cable Internet Protocol



Shared line is split into channels by frequency

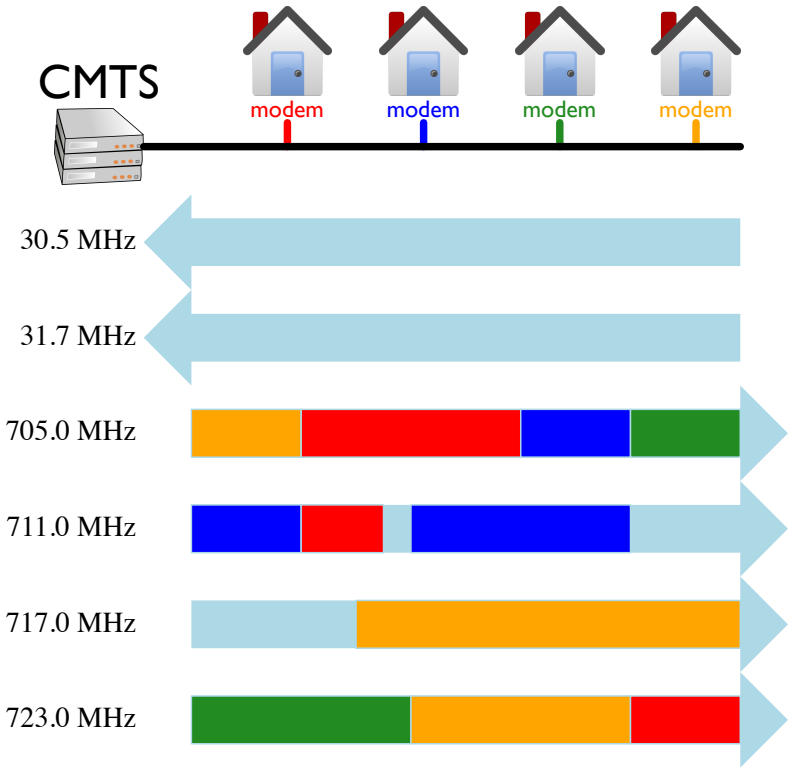
— which is an example of FDM

DOCSIS Cable Internet Protocol



More download channels than upload channels

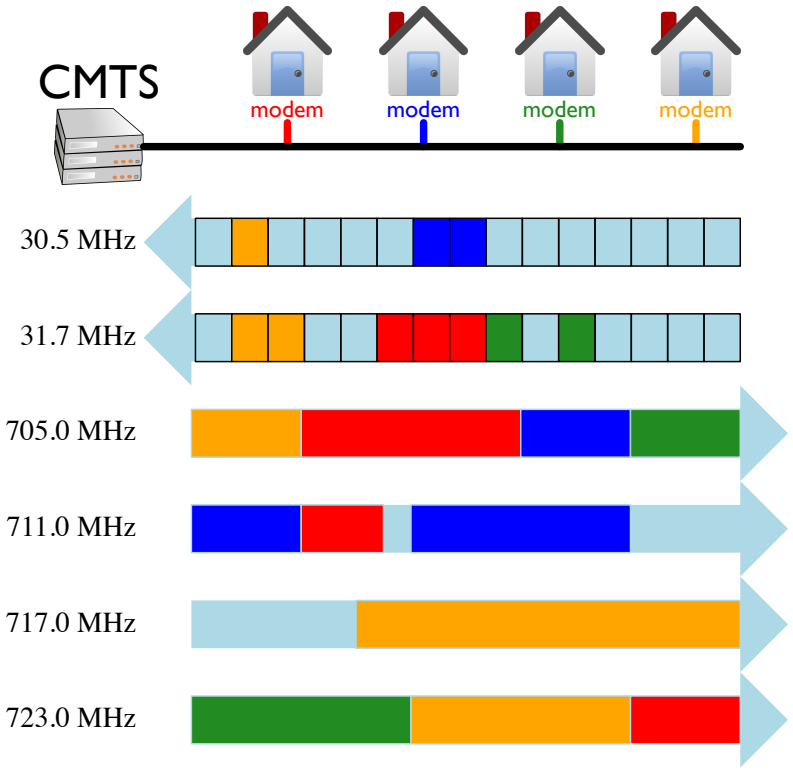
DOCSIS Cable Internet Protocol



All modems see all download channels

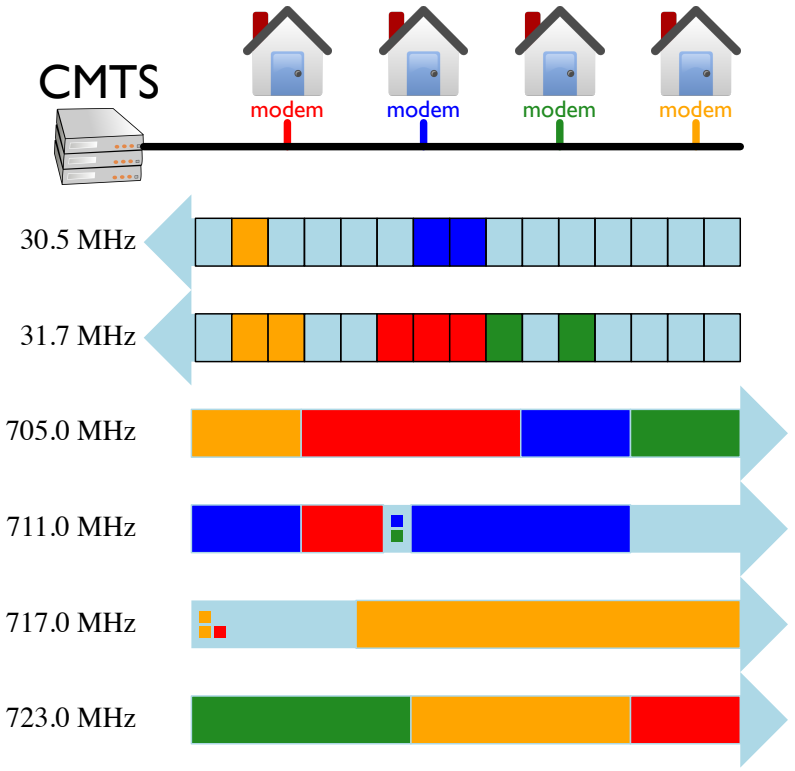
Since only CMTS writes, no need for slots or collision handling

DOCSIS Cable Internet Protocol



Shared upload channel has slots

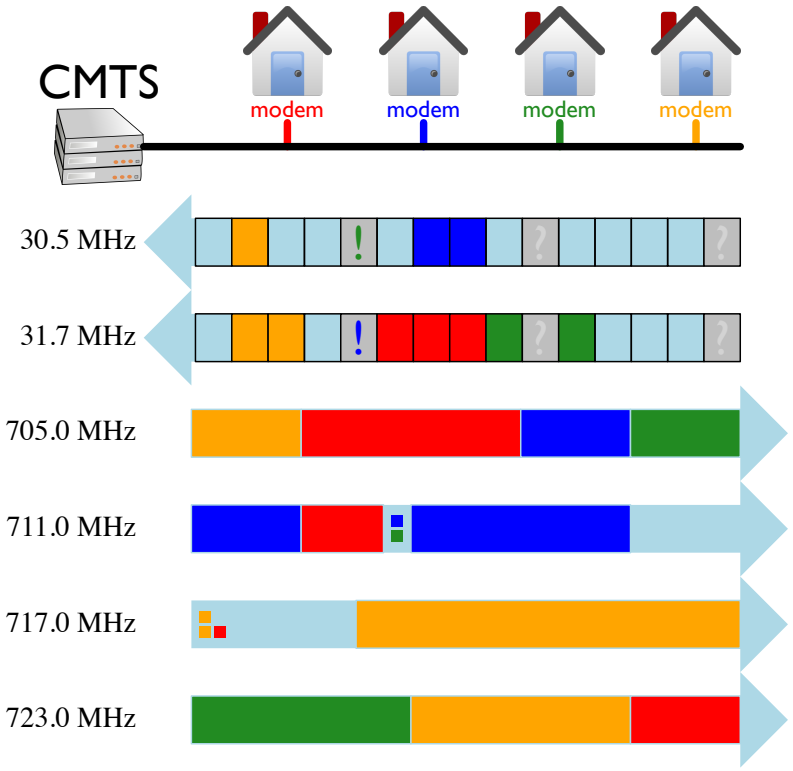
DOCSIS Cable Internet Protocol



Shared upload channel has slots

Slots are allocated by CMTS

DOCSIS Cable Internet Protocol



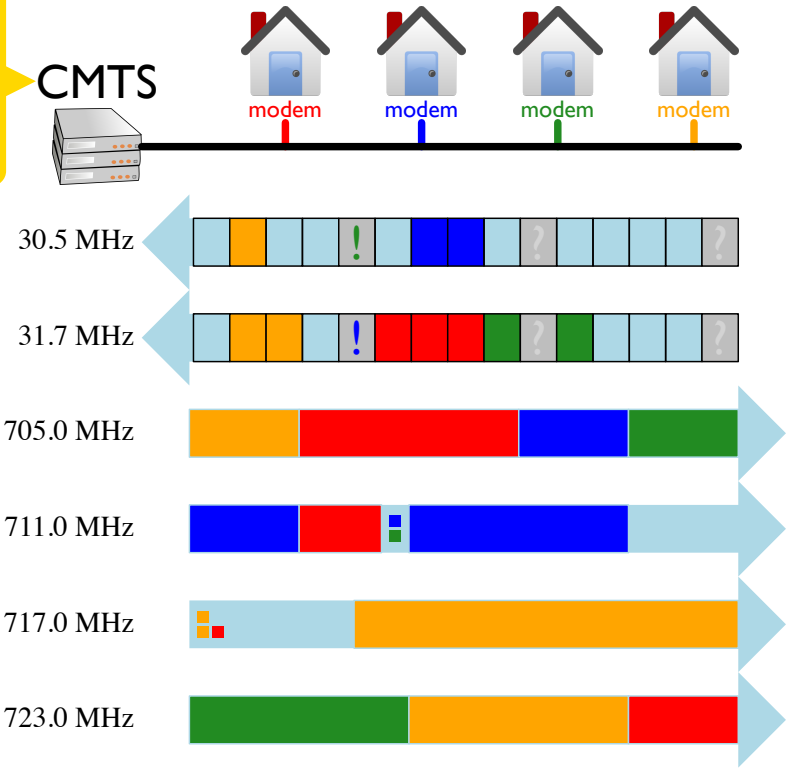
Shared upload channel has slots

Periodic polling uses designated slots

Slots are allocated by CMTS

DOCSIS Cable Internet Protocol

Centralized polling ok, since CMTS must always work

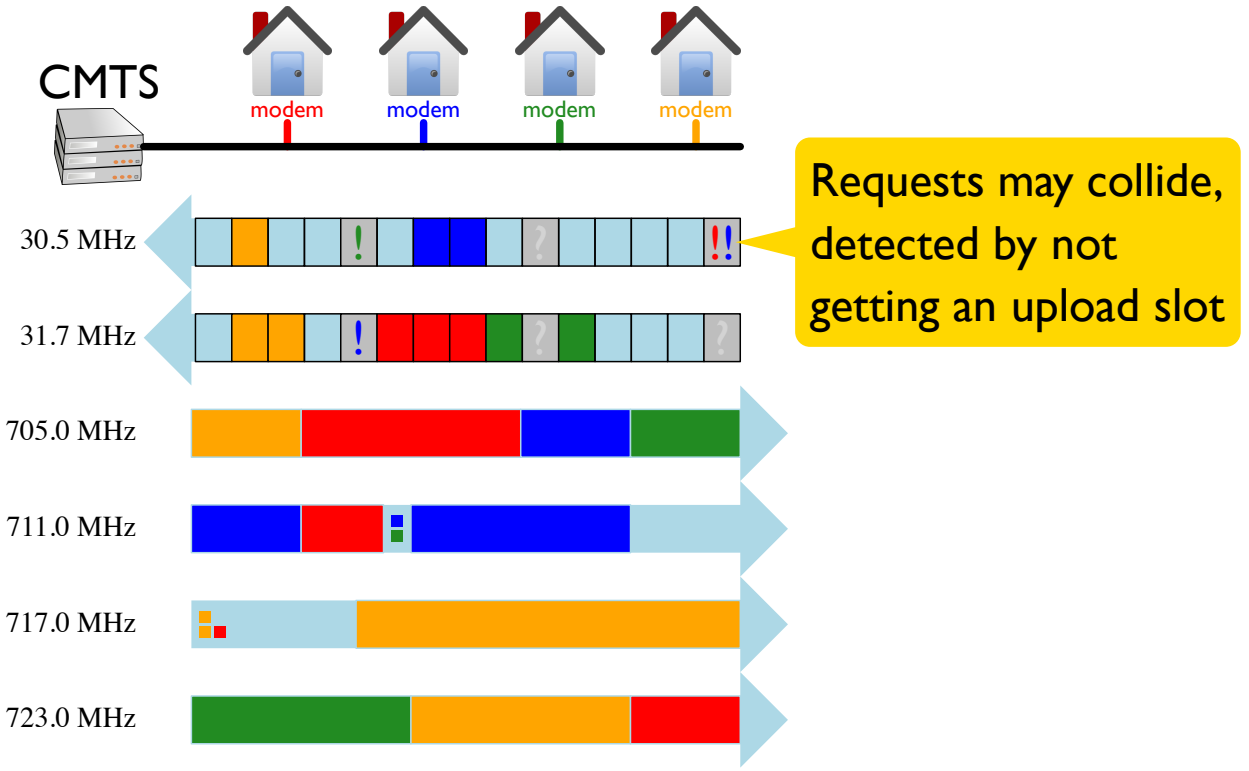


Shared upload channel has slots

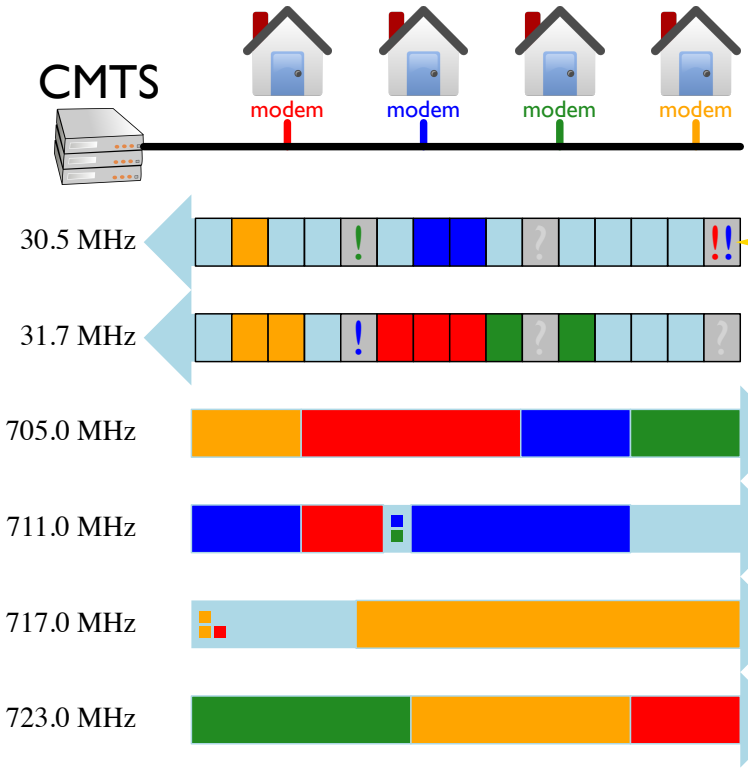
Periodic polling uses designated slots

Slots are allocated by CMTS

DOCSIS Cable Internet Protocol



DOCSIS Cable Internet Protocol



Requests may collide, detected by not getting an upload slot

Collision handled by random delay and then retry

Summary

Cyclic-redundancy check (CRC) commonly used at link layer

Link-to-physical transition often involves negotiating a shared medium

Two ways to share:

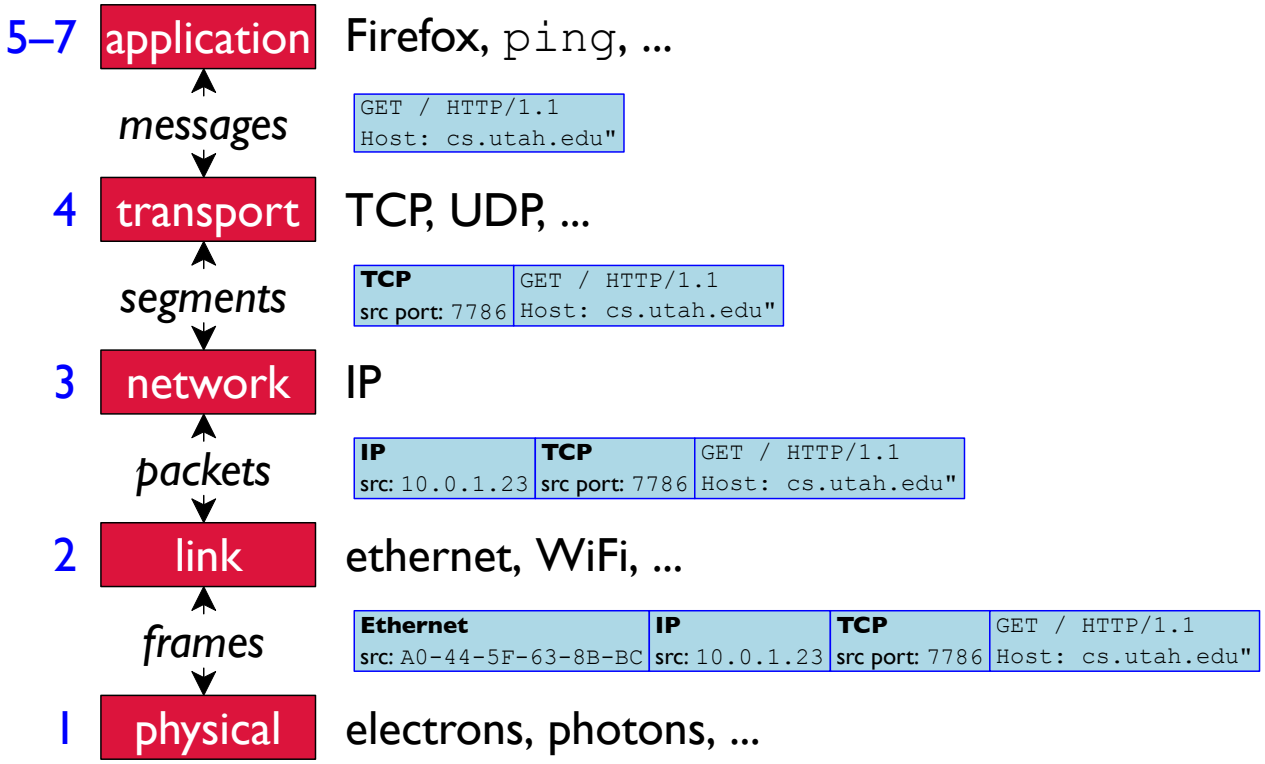
- **Time-division multiplexing (TDM)**
- **Frequency-division multiplexing (FDM)**

Three ways to allocate a division:

- **polling**
- **token-passing**
- **random access**

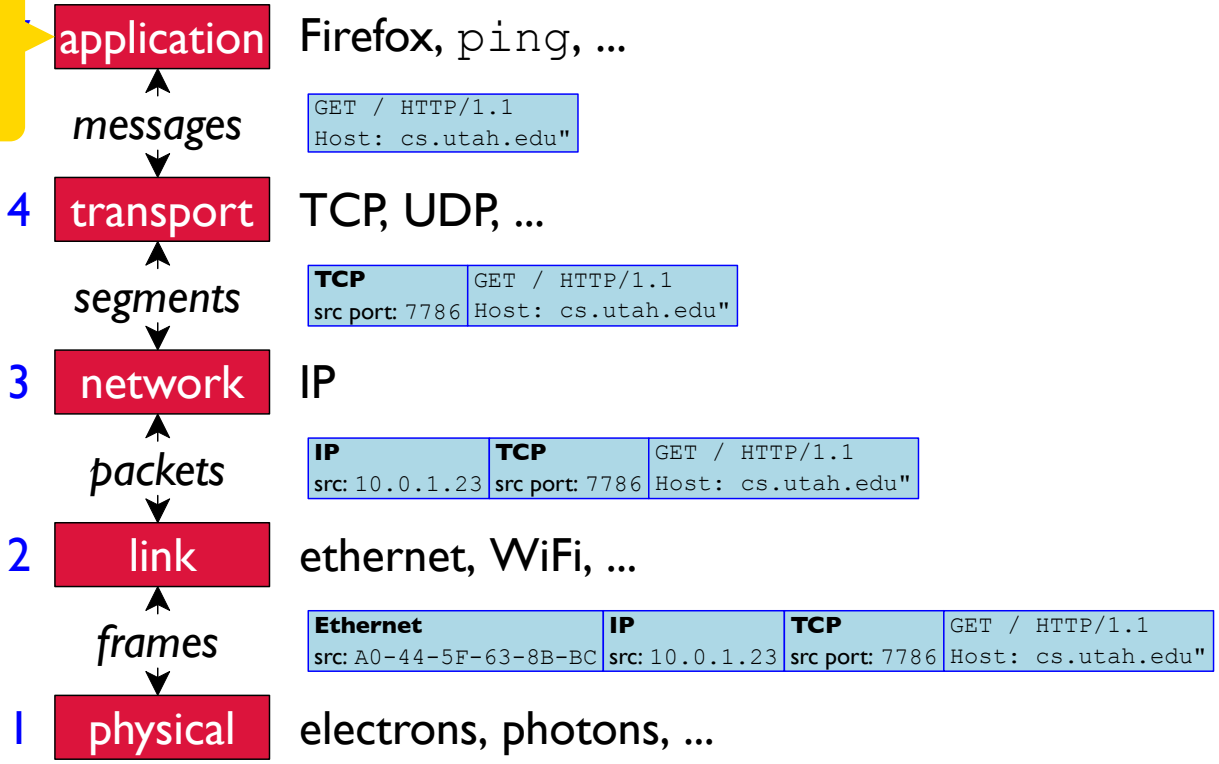
  ... with **carrier sense** and/or collision detection

Layers — *Done!*


5–7
application
Firefox, ping, ...
messages
GET / HTTP/1.1
Host: cs.utah.edu"
4
transport
TCP, UDP, ...
segments
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
3
network
IP
packets
IP
src: 10.0.1.23
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
2
link
ethernet, WiFi, ...
frames
Ethernet
src: A0-44-5F-63-8B-BC
IP
src: 10.0.1.23
TCP
src port: 7786
GET / HTTP/1.1
Host: cs.utah.edu"
1
physical
electrons, photons, ...

Layers — *Done!*

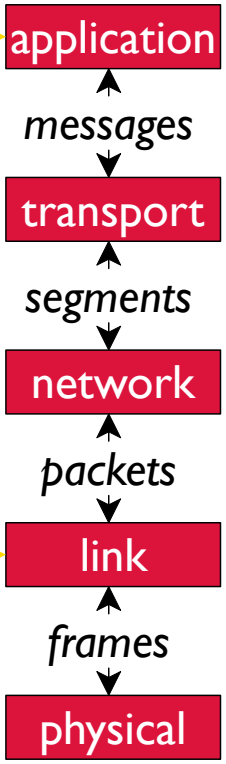
When building systems, we mostly get to work at this level of abstraction



Layers — *Done!*

When building systems, we mostly get to work at this level of abstraction

Attacks sometimes work by breaking a leaky abstraction



Firefox, `ping`, ...

```
GET / HTTP/1.1
Host: cs.utah.edu"
```

TCP, UDP, ...

TCP src port: 7786	GET / HTTP/1.1 Host: cs.utah.edu"

IP

IP src: 10.0.1.23	**TCP** src port: 7786	GET / HTTP/1.1 Host: cs.utah.edu"

ethernet, WiFi, ...

Ethernet src: A0-44-5F-63-8B-BC	**IP** src: 10.0.1.23	**TCP** src port: 7786	GET / HTTP/1.1 Host: cs.utah.edu"

electrons, photons, ...