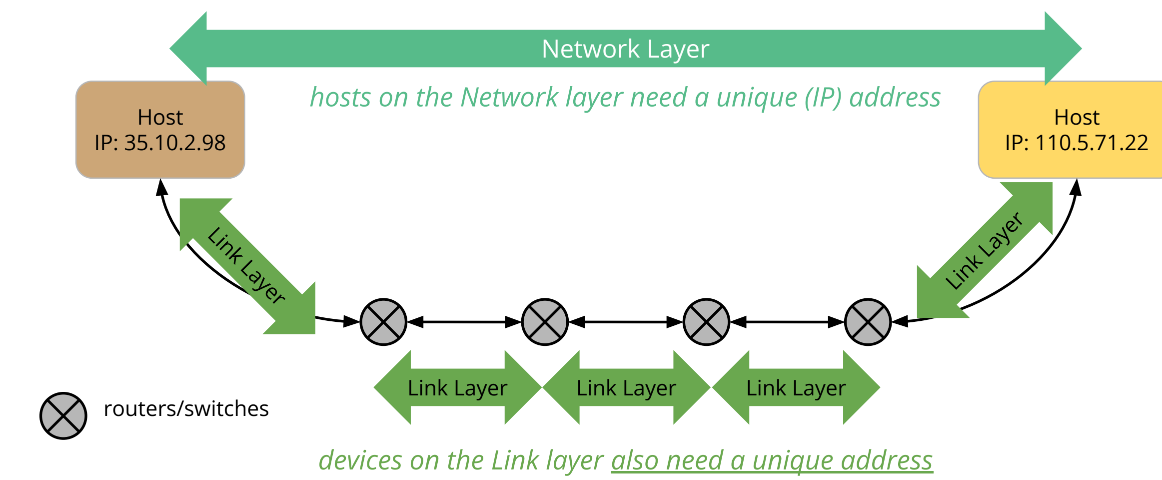
Chapter 06: Link Layer

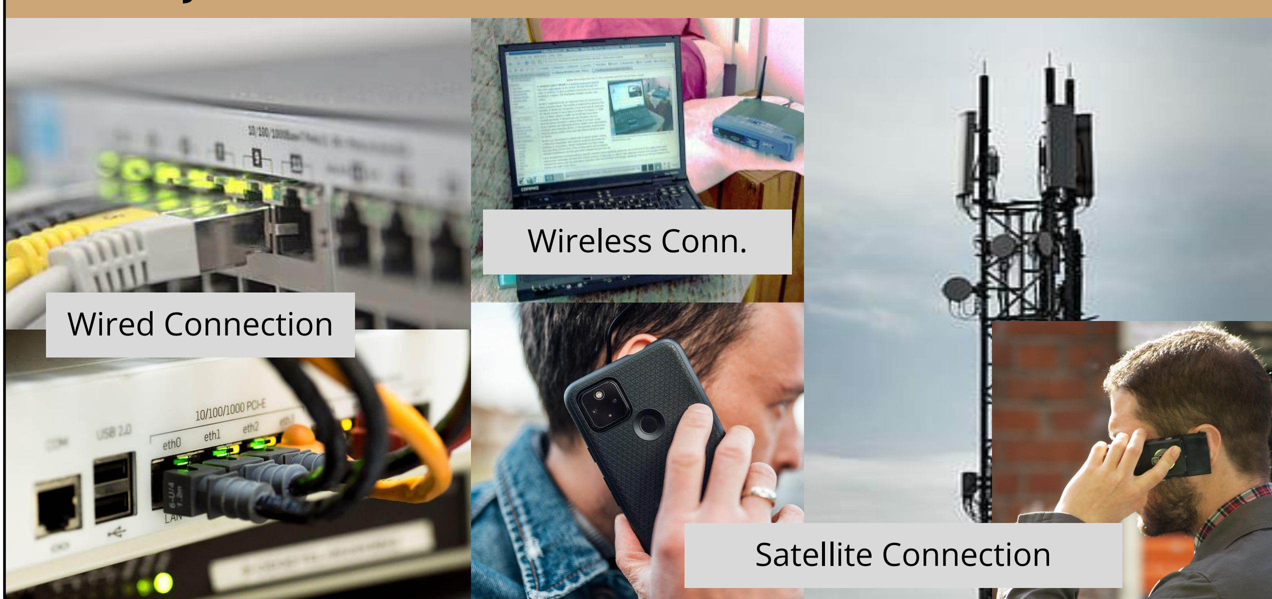
Internet Layers

Application	*Exchange messages between two applications*
Transport	*Data transfer between two processes*
Network	*Data transfer between two hosts*
Link	***Data (frame) transfer between two neighboring network elements*** ⬅
Physical	*Bit transfer on physical medium*

Network Layer vs. Link Layer



Link Layer: What You Can See



Main Job of Link Layer

- Data transfer between directly connected nodes
- "**Directly connected** nodes" NO ROUTER(s) in between
  - Wired connection: nodes connected to the same Ethernet switch box
  - Wireless connection: nodes connected to the same WiFi base station or Wireless Access Point
  - Satellite connection: i.e. StarLink

Sharing the Link

These laptops are connected to the same WiFi base station

Generalization: *network devices may be connected to the same link*

Issues with Link Sharing

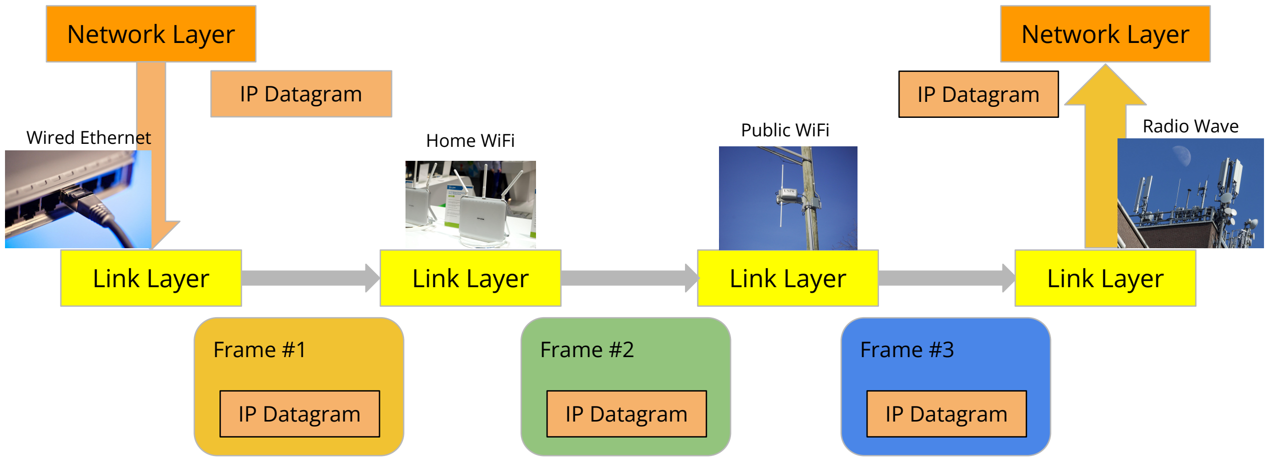
- **Who are you**
  - Uniquely identify a particular device on the shared link
  - Ensure the proper recipient node receives the data intended for it
- **Traffic Conflict**
  - How to control access to the shared link
  - Prevent multiple nodes to put data on the shared link at the same time
- **Error Detection**

IP Datagrams and Frames

- On the sender side
  - Encapsulate datagram (from the Network Layer) into a frame by adding a **header** and **trailer**
  - Access the media to pass the frame to the Physical Layer
- On the receiver side
  - Obtain frames from the Physical layer
  - Unpack the header/trailer, pass the datagram to the Network Layer

IP Datagrams and Link Frames

IP datagrams may be transported over a variety of link protocols

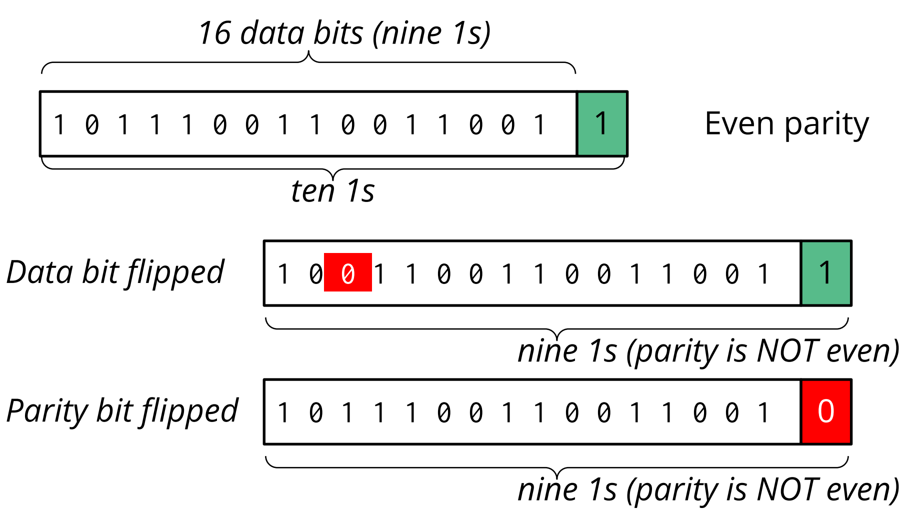


Indonesia Ham Radio Network

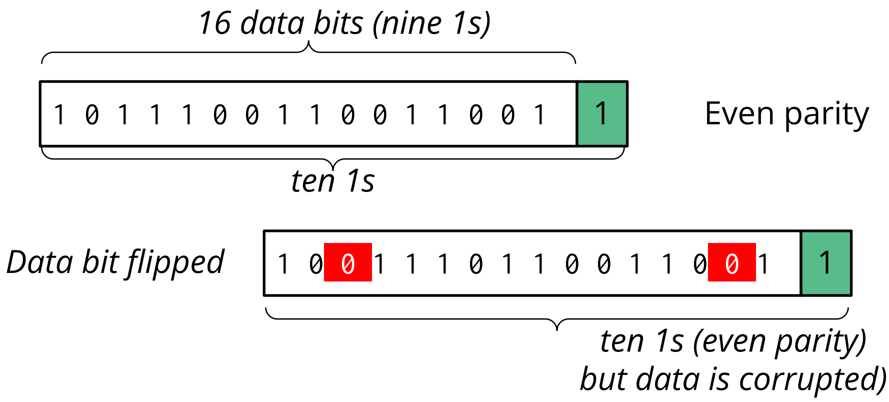
Error Detection & Correction

- One-dimensional Parity Bit:
    - Even parity: even number of 1s in the data and parity bit
    - Odd parity: odd number of 1s in the data and parity bit
    - 16-bit data + 1-bit parity ⇒ Total 17-bit bundle
- Two-dimensional Parity Bits
    - Arrange data bits in rows and columns
    - Compute parity bits: one for each row, one for each column
    - 16-bit data arranged into 4x4
        - Four 1-bit row parity bits and four 1-bit column parity bits
        - Total is 16 + 4 + 4 ⇒ 24-bit bundle
- Cyclic Redundancy Check

One-Dimensional Parity Bit



One-Dimensional Parity: False Positives

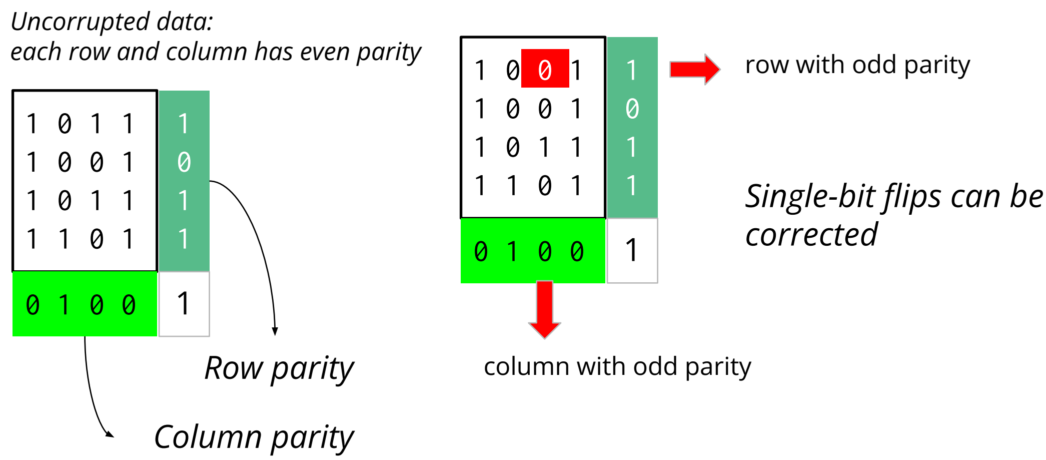


In general: even number of bit flips gives a false positive

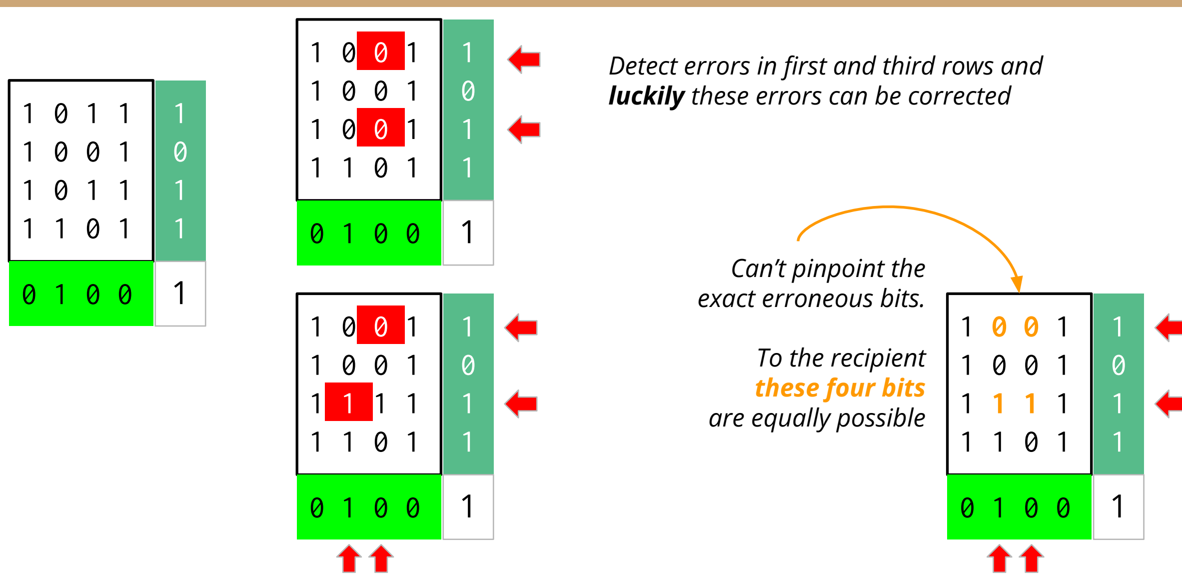
Limitations of one-dimensional parity

- Can detect errors caused by single bit flip, but cannot pinpoint which bit caused the error
- Errors caused by double bit flip are undetectable, the even parity check gives a false positive result
  - In general errors caused by 2N bit flips are undetected
- Better technique: 2D parity bits

Two-Dimensional Parity Bit(s)



2D Parity: Multi-bit Flips



Parity Bits vs. Check Sum vs. CRC

	CheckSum	CRC
Technique	• Treat **each byte** (or group of bytes) as an integer • Compute the sum • Include the carry bit(s) into the sum	• Treat the **entire data** as **a huge integer** • Compute the remainder of the value with a known divisor (generator)
Operation	Adding integers	Module 2, XOR logic
HW circuit	Expensive (N-bit full adders)	Simpler: XOR gates & shift register
False positive	Byte swaps	Resistant to byte swaps

Cyclic Redundancy Code

CRC: General Ideal

- Based on mathematical *theory of polynomials*
    - *Polynomial division and remainder*
- Treat the message M as a (*potentially huge*) integer (binary) value
- Use a generator number (G)
- Compute the check value R from M so that the "concatenated" values <M,R> is a multiple of G

Binary number	As a number	As a polynomial
1100101	$2^6 + 2^5 + 2^2 + 2^0$	$x^6 + x^5 + x^2 + x^0$

CRC (Non-Real) Example

Message A (its ASCII value is 65),   Generator number: 7

- Find the check value R (one-digit) such that
    - The 3-digit number (650 + R) is a multiple of 7?
- Answer R is 1, because 651 = 7 x 93

If the generator is 9, then the answer R is 7 ⇒ 657 is a multiple of 9

- How do you decide the value for the generator?
- How many digits needed for the check value?

How to compute R (with Generator 7)?

$$\begin{aligned} 650 + R &= 7k \\ (650 + R) \mod 7 &= 0 \\ (650 \mod 7) + (R \mod 7) &= 0 \\ (650 \mod 7) + R &= 7 \\ R = 7 - (650 &\mod 7) \\ R &= 1 \end{aligned}$$

- It turns out that when the generator is 7, then R must be 0, 1, 2, ..., 6
- For larger generators R requires more digits

More Examples

Message: "A" (ASCII value 65)

Generator	7	17
Range of R	0, 1, 2, ..., 6	0, 1, 2, ..., 16
R	7 - (650 mod 7) = 1	17 - (6500 mod 17) = 11
Message + Code	651	6511
Validation	651 is 7 x 93	6511 is 17 x 383

CRC for longer messages (simplified example)

```
Message    W  H  O
ASCII:    87 72 79
```

```
Value: 877279 (Eight hundred seventy-seven thousand two hundred seventy-nine)
Generator 113 (possible check values: 0, 1, 2, ...., 112)
Must use 3-digit check value
Check Value: 113 - (877279000 mod 113) = 53

Transmitted data: 877279053   (the leading zero is required to make 3-digit CRC)
```

```
Received data error check: 877279228 % 917 ⇒ should be zero
(both the sender and receiver must use the same generator 917)
When erroneous data received (WHO => RHO)
827279228 % 917 ⇒ 812    (not zero)
```

Bad choice of generator

Message "HI" (ASCII value **72 73**)   Generator = 3

R = 3 - (72730 mod 3) = **2**

Transmitted message is **72732**

Unfortunately these numbers are also multiple of 3 (*false positive*)

- **7372**2 (encoded message "**IH**")
- **3277**2 (encoded message " M")
- **7272**3 (encoded message "HH")

CRC in binary

message M (d bits)	check code R (r bits)

- Check code R is derived/computed from the message M using division and remainder
- Generator must be (r+1) bits
- The above pair (M,R) can also be viewed as
  - Shifting the message M left by r positions (and appending zeros)
  - XOR the check code with shifted M

message M (d bits)	0000000000000000000
	check code R (r bits)

Refresher: Long-Division & XOR

```
       16706
43 ) 718395
     43          ┐
     288         │
     258         │
      303        ├ subtraction
      301        │
        295      │
        258      ┘
         37  ⇒ remainder
```

Module 2 Arithmetic		
Addition	**Subtraction**	**XOR**
0 + 0 = 0	0 - 0 = 0	0 ⊕ 0 = 0
0 + 1 = 1	0 - 1 = 1	0 ⊕ 1 = 1
1 + 0 = 1	1 - 0 = 1	1 ⊕ 0 = 1
1 + 1 = 0	1 - 1 = 0	1 ⊕ 1 = 0

CRC Generalization to Binary Data

- Generator G ⇒ r digits Check value (R)
- Message M

$$R = G - (M \times 10^r) \bmod G$$

$M\times10^r$ ⇒ append r zeros to M

In binary:

- r-bit Check bits (R)
- (r+1) bit generator (G)
- d-bit message (M)

$$R = G \text{ xor } (M \times 2^r) \bmod G$$

$M\times2^r$ ⇒ append r zeros to M

CRC in Binary

```
          #          3      .
Binary: 00100011   00110011

Using
6-bit generator 101001
Hence, 5-bit check value

General rule:
(r+1)-bit generator
r-bit check value
```

```
                        1 0 1 0 0 0 0 1 1 0 1 0 1
                   ___________________________________
101001 )   1 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0
           1 0 1 0 0 1       |   |   |   |   |   |
               1 0 1 0 0 0   |   |   |   |   |   |
               1 0 1 0 0 1   |   |   |   |   |   |
                         1 1 1 0 0 1     |   |   |
                         1 0 1 0 0 1     |   |   |
                           1 0 0 0 0 1   |   |   |
                           1 0 1 0 0 1   |   |   |
   bitwise XOR                 1 0 0 0 0 0   |   |
                               1 0 1 0 0 1   |   |
                                   1 0 0 1 0 0   |
                                   1 0 1 0 0 1   |
                  Remainder          1 1 0 1 0

Transmitted Data   0 0 1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 0
```

CRC Validation

3 .

Binary: 00100011 **00110011**

Using
5-bit check bits
6-bit generator **101001**

```
                             1 0 1 0 0 0 0 1 1 0 1 0 1
101001 )  1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 0
          1 0 1 0 0 1     |   |   |   |   |   |
            1 0 1 0 0 0   |   |   |   |   |   |
            1 0 1 0 0 1   |   |   |   |   |   |
                      1 1 1 0 0 1     |   |   |
                      1 0 1 0 0 1     |   |   |
                        1 0 0 0 0 1   |   |   |
                        1 0 1 0 0 1   |   |   |
                            1 0 0 0 1 1   |   |
                            1 0 1 0 0 1   |   |
                                1 0 1 0 0 1   |
                                1 0 1 0 0 1   |
                       Remainder  ⇒       0 0
```

(Media|Link) Access Control

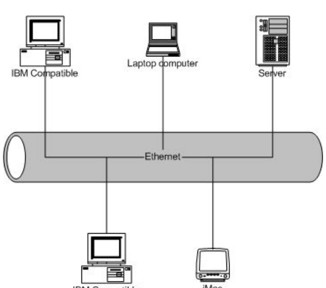
Access to Link: Point-to-Point

Modem

Phone Line

Modem

Access to Link: Broadcast on Shared Medium



shared wired (Ethernet)

shared radio (3G/4G/5G)

shared Wifi station

Issue: signal interference and collision when two (or more) sender are broadcasting to the shared medium at the same time

Multiple Access to Links

Preventing Collision & Fairness

- **Fully** distributed algorithm to share channel, decide who can transmit
- Communication about channel sharing must be exchanged using the channel itself
- **Fairness**: When a channel with capacity R bits/seconds is shared among M nodes, each node should be allowed to send at average rate R/M

Three Protocols for Multiple Access (to Shared Links)

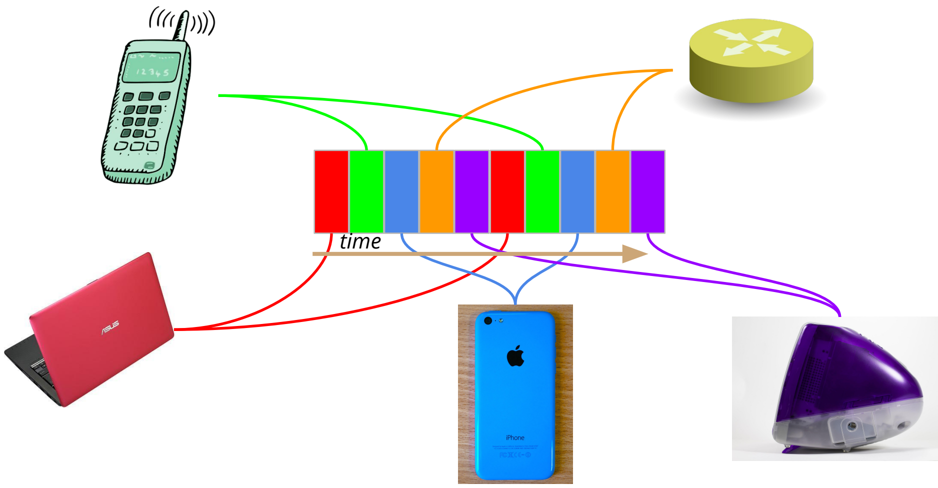
- Channel partition
    - TDMA (Time Division Multiple Access): divide the channel into N equal time slots
    - FDMA (Frequency Division Multiple Access): divide the channel into smaller frequency bands
    - Combining both FDMA & TDMA
    - CDMA (Code Division Multiple Access)
- Taking turns (avoid collisions)
- Random access (allow collisions but then recover from collisions)

Channel Partition

- TDMA: Time Division Multiple Access
    - Divide the link/channel into time frames
    - Divide each time frame into N slots (one per node)
    - Node-k can only use the shared channel during its assigned slot-k
- FDMA: Frequency Division Multiple Access
    - Divide the channel into N sub frequency
    - Each node use the assigned frequency to access the link
- CDMA: Code Division Multiple Access
    - Each node is assigned a unique code that will be used for encoding its data
    - (Further details in Chapter 7)

TDMA: Time Division Multiple Access

Each device can use the (shared) link only during its assigned time slots

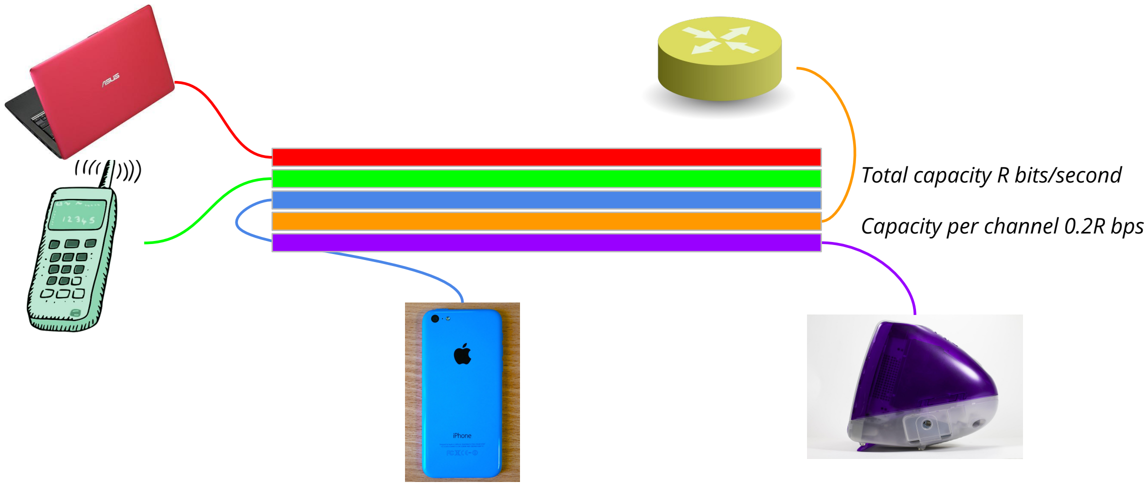


TDMA Examples: Cellular Communication

	Frequency	Time Frame Width
D-AMPS (Digital Advanced Mobile Phone System) “1G”	800MHz, 1900 MHz	6.67 milliseconds
Global System for Mobile Comm (GSM) “2G” “3G”	US, Canada: 850 MHz, 1900 MHz Europe: 900 MHz, 1800 MHz Others: 400 MHz, 450MHz	4.6 milliseconds

FDMA: Frequency Division Multiple Access

Each device must use the assigned sub frequency



FDMA: Examples

- Satellite Communication System
- AM Radio
- FM Radio
- WiFi 2.4 GHz and 5 GHz

WiFi 2.4 GHz

Channel	Center Freq
1	2.412 GHz
2	2.417 GHz
3	2.422 GHz
4	2.427 GHz
12	2.467 GHz.
13	2.472 GHz

Live Demo: WiFi Analyzer (Android App)

TDMA and FDMA Performance Comparison

Assume link capacity is R bits/second

- Each node in TDMA can use the link only 1/N of the entire available time
- Effective usable rate per node is only R/N bits per second
- Each node in FDMA is assigned a sub-channel who capacity is R/N bits/second
- In both TDMA/FDMA if the other nodes are not active, the active node cannot use the other time slots nor the other frequencies. Max rate of each node is capped to R/N bits/second

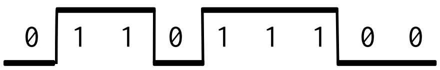
Taking Turns

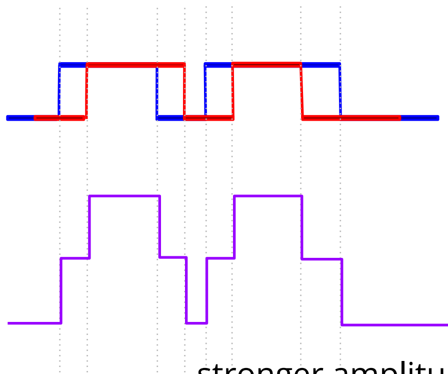
- Polling
  - One of the nodes becomes the “manager” which will polls the other node for data
  - Issue: when the manager node dies, communication will stop
- Token Passing
  - Use a special packet (“token”) which is passed from one node to another
  - A node is allowed to push data to the shared link only when it is currently holding the token
  - Issue: the the current node holding the token is dead, communication will stop

Random Access

- RA removes TDMA restriction, each RA node may *push data anytime*
- RA removes FDMA restriction, each RA node can used the *entire available link bandwidth*
- Issue: nodes may push anytime ⇒ **collisions are possible**
- How to (detect and) recover from collisions?
- Techniques
  - Slotted ALOHA:
  - CSMA: Carrier Sense Multiple Access
  - CSMA/CD: CSMA with Collision Detection

Detecting Signal Collisions: Physics of Waves





Red + **Blue**:

combined signals
show stronger amplitude

stronger amplitude implies collisions

Slotted ALOHA

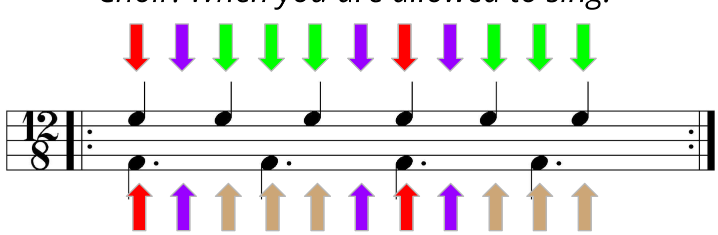
- Like TDMA, but each node may push data during ANY time slots (not just its assigned time slot)
- Data are sent in fixed frame size (L bits)
  - With link capacity R bits/sec frame duration is L/R seconds
- All the nodes must be synchronized (so each knows when is the beginning of the slot)
- If a node detects a collision, it retransmit with a probability of p

Slotted ALOHA

Everyone sings the notes at the *same tempo*

Slotted ALOHA ("Sings to the beats")

Choir: When you are allowed to sing!



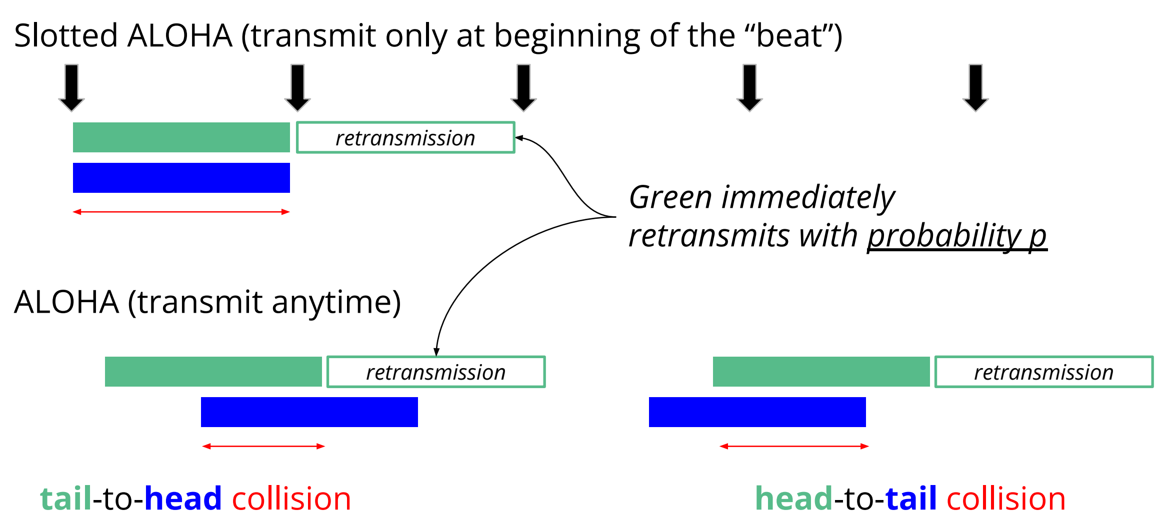
Link Layer nodes: The time you are allowed to push data to the (shared) link

Collisions ↑ ↑ *Unused slots*

Slotted Aloha Performance

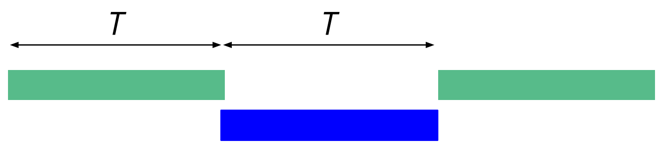
Probability of	**Value**
A single node pushing data	p
A single node NOT pushing data	$1 - p$
A given node successfully pushing data (only that node pushes data AND the other N-1 do not)	$p\,(1-p)^{N-1}$
Any given node successfully pushing data	$Np\,(1-p)^{N-1}$

ALOHA vs. Slotted ALOHA: Action AFTER a collision

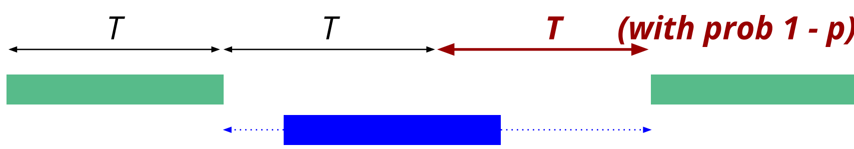


ALOHA: Action AFTER NO collision

- Green Action after successful transmission: always wait for T seconds



- In addition, with a probability of *1-p* wait for another T seconds. *Rationale: give more space for other nodes ("blue") to use the link*



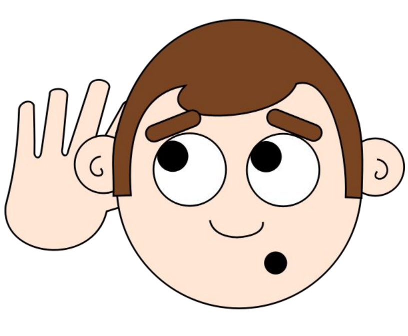
Aloha Performance

Probability of	**Value**
A single node pushing data	p
A single node NOT pushing data	$1 - p$
N-1 nodes NOT pushing data	$(1-p)^{N-1}$
A node successfully pushing data (neither H2T nor T2H collisions)	$p\ (1-p)^{N-1}(1-p)^{N-1}$
Any given node successfully pushing data	$N\ p\ (1-p)^{2(N-1)}$

Carrier Sense Multiple Access (CSMA)

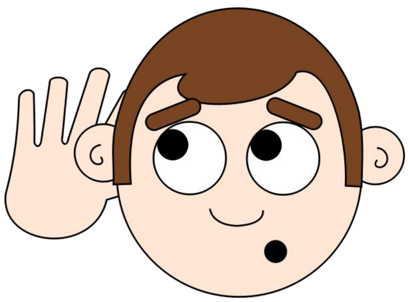
CSMA: Listen Before You Talk

CS: Carrier Sense ("Listen")

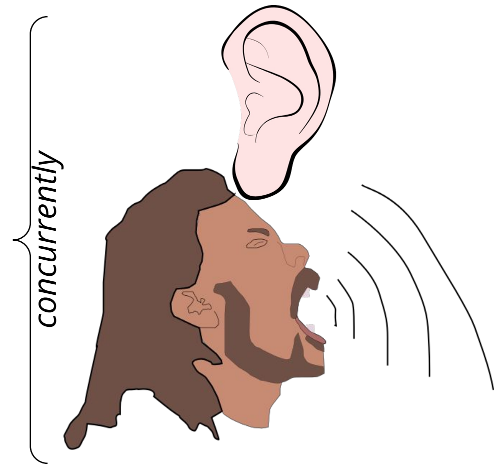


THEN

CSMA/CD: Listen Before You (Talk while Listening)



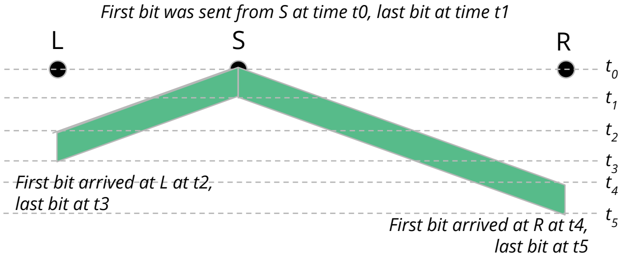
THEN

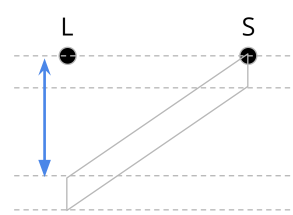


CD: Collision Detection ("Listen for Cross Talk & Stop Talking")

Signal Space-Time Diagram

First bit was sent from S at time t0, last bit at time t1

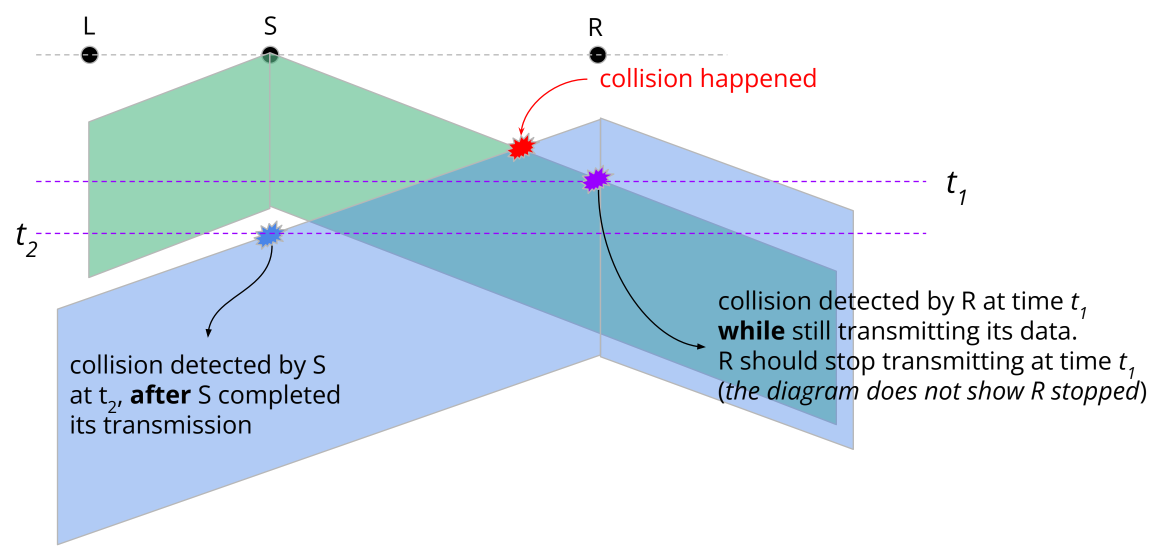




Slower propagation speed (Longer ***propagation time****)*

Within the physical medium, signal propagates in both directions

Signal Space-Time Diagram: Collisions



CSMA/CD Animation

CSMA/CD: When to Retransmit (After collision)

- Wait too short ⇒ Risk another collision
- Wait too long ⇒ Underutilized link capacity
- Wait a random amount of time
- Wait for a “controlled” random ⇒ Binary Exponential Backup
  - Used by Ethernet CSMA/CD

Binary Exponential Backoff

After N collision	Choice of random wait multiplier (K)
1	0,1
2	0,1,2,3
3	0,1,2,3,4,5,6,7
4	0,1, 2, 3, 4, ..., 15
5	0,1,2,3, ...,31
n	0, 1, 2, 3, ..., 2^n-1

Actual wait time = K x T

- *T is a predefined duration*
- *For Ethernet T = time to transmit 512 bits (less than the size of a frame)*
- *In practice n is capped to a fixed value*

Taking Turns

- Polling (*centralized* by a master node)
  - The master node is simply controlling the turn and not involved in transmission
  - Example: Bluetooth
- Token-passing (*distributed*)
  - A token is circulated among the nodes
  - Only the node who currently holds the token is allowed to transmit
  - Example: Fiber Distributed Data Interface (FDDI)
- Common issues for both:
  - Master node died
  - Lost token (one of the nodes failed to pass it to the next node)

Cable Access Network

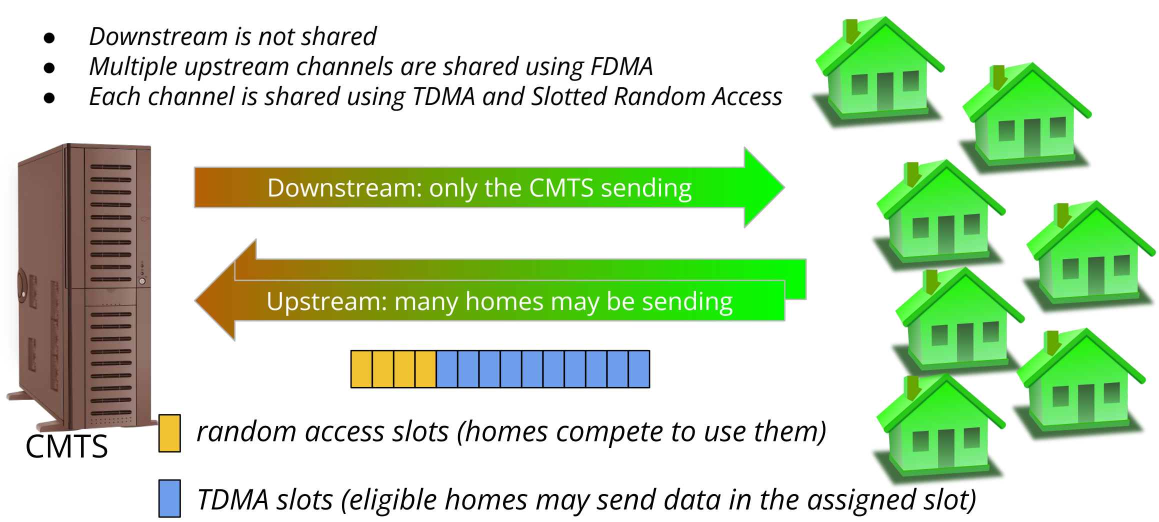
Cable Modem: FDMA + TDMA + Random Access

- DOCSIS: Data Over Cable Service Interface Specification
- CMTS = Cable Modem Termination System

	Upstream	**Downstream**
Transmission Direction	Homes ⇒ CMTS	CMTS ⇒ Homes
Multiple Access	Yes	No
Bandwidth per channel	6.4 - 96 MHz	24 - 192 MHz
Max Throughput	1 G bits/second	1.6 G bits/second

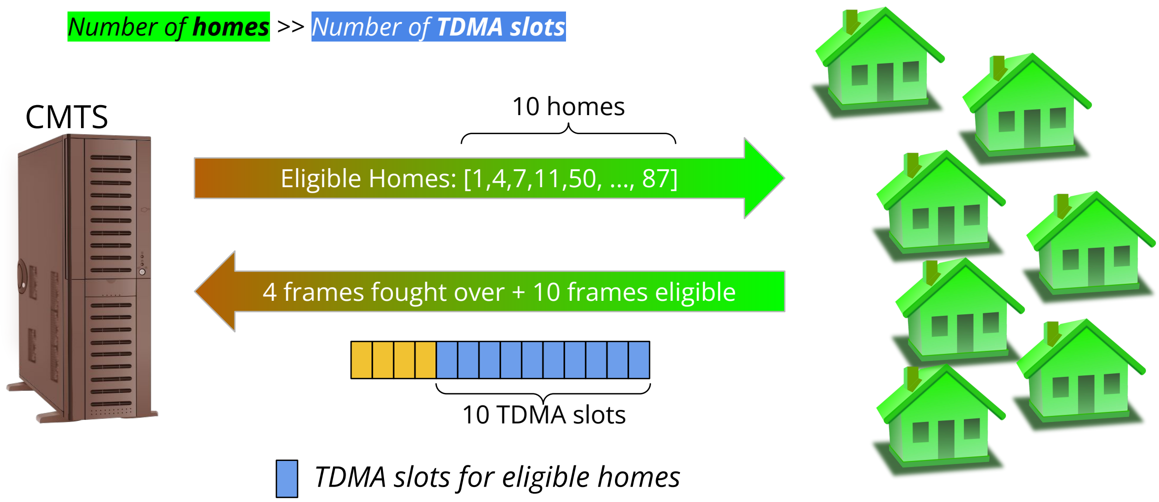
Cable Access Network

- *Downstream is not shared*
- *Multiple upstream channels are shared using FDMA*
- *Each channel is shared using TDMA and Slotted Random Access*



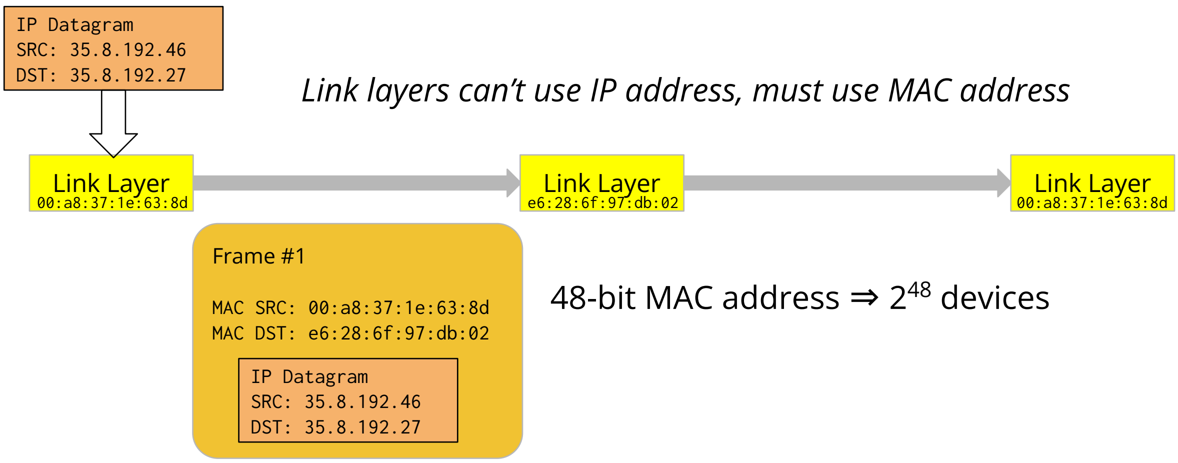
Cable Access Network: “Eligible Homes”

*Number of **homes*** >> *Number of **TDMA slots***



MAC Addresses & ARP

Link Addresses/MAC Addresses (Media Access Control)



Link layers can't use IP address, must use MAC address

48-bit MAC address ⇒ 2^{48} devices

Live Demo

MacOS ⇒ System Settings Network

Linux: `ipconfig` or `ipmaddr`

MAC-48 vs. EUI-64

	MAC-48	EUI-64
Standard Publication Date	1980	2018
Expected Lifetime	100 years (until 2080)	2^{16} x 100 years = 6.4 million yrs
Address length	48 bits	64 bits
Acronym	Media Access Control	Extended Unique Identifier

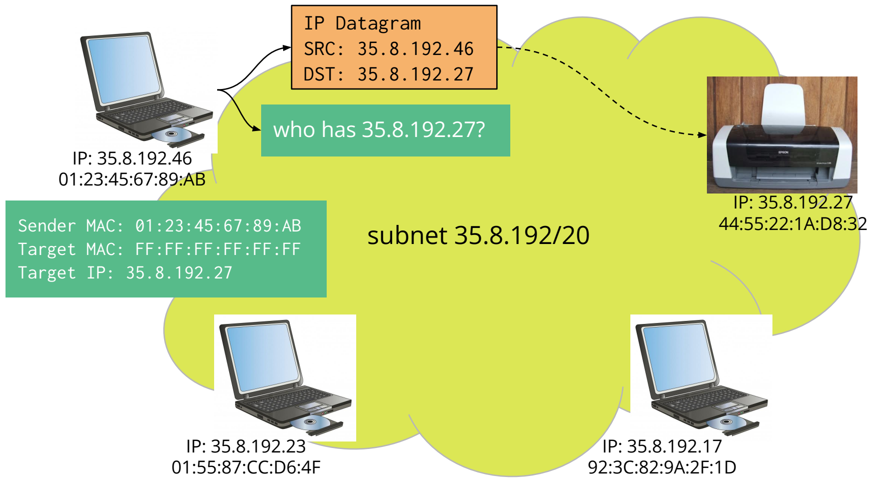
$$\text{Usage rate} = \frac{2^{48}\ \text{devices}}{100\ \text{year}}$$

$$\text{Life expectancy} = \frac{2^{64}}{\text{usage rate}} = \frac{2^{64}}{2^{48}/100} = 2^{16} \times 100$$

ARP (RFC826, RFC1180)

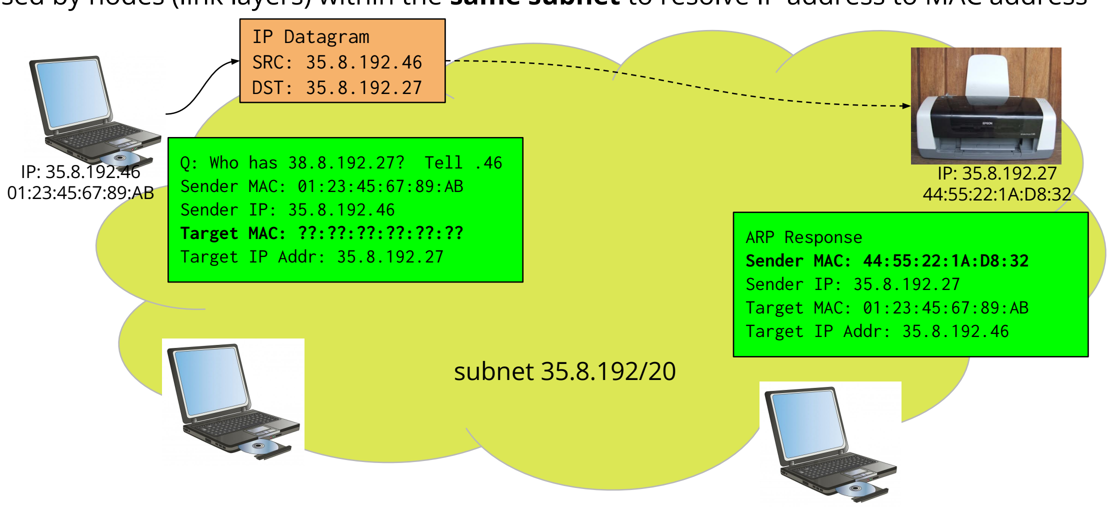
ARP: [MAC] Address Resolution Protocol

Used by nodes (link layers) within the **same subnet** to resolve IP address to MAC address



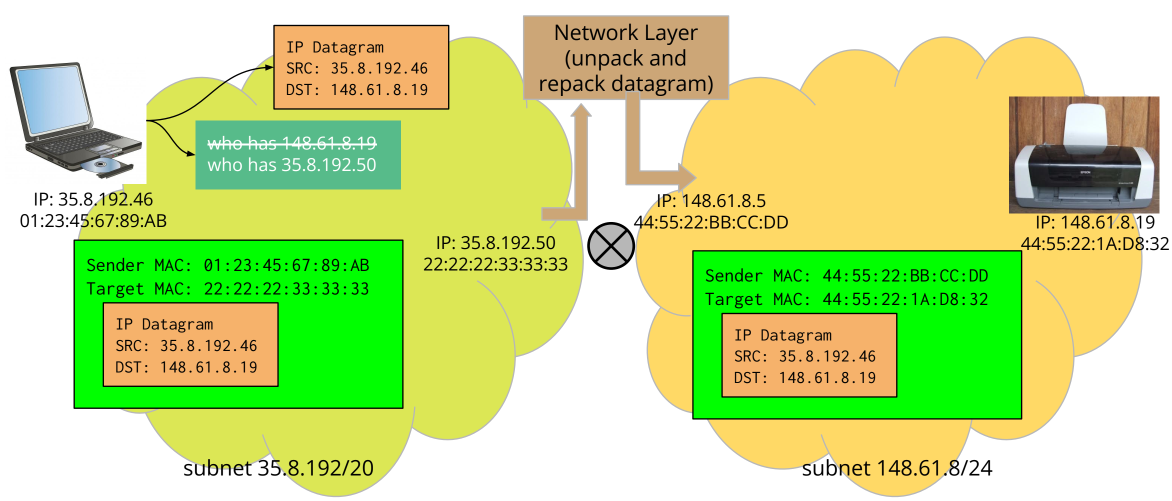
ARP: Query and Response

Used by nodes (link layers) within the **same subnet** to resolve IP address to MAC address



Wireshark ARP Demo: dns-trace-3-2

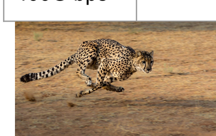
No ARP Across Subnets



Ethernet

Ethernet Standards

	Coaxial	Optical Fiber	Twisted Pair	Shielded Copper
10M bps	~~10 Base-2~~	~~10 Base-F~~	10 Base-T	
100M bps		100 Base-F	100 Base-T	
1G bps		1000 Base-SX (1998) 1000 Base-LX (1998) 1000 Base-BX (2004)	1000 Base-T	1000 Base-CX
40G bps		40G Base-FR	40G Base-T	40G Base-CR
100G bps		100G Base-SR		
400G bps		400GBe		



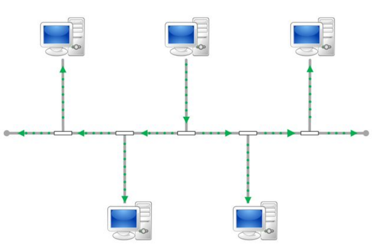
Latest product: 800G bps router by Nokia (2024)

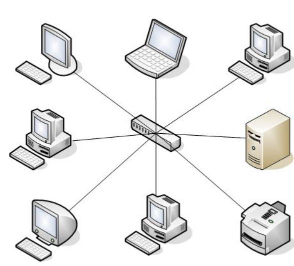
Related Standards (IEEE 802.x)

Standard	Description	Practical Examples
IEEE 802.3	Ethernet (wired connections)	
IEEE 802.11	WiFi (wireless connections	
IEEE 802.15	Wireless Personal Area Network (PAN)	Wireless USB, InfraRed
IEEE 802.15.4	Low-Rate Wireless PAN	Zigbee
IEEE 802.15.6	Body Area Network	

Ethernet

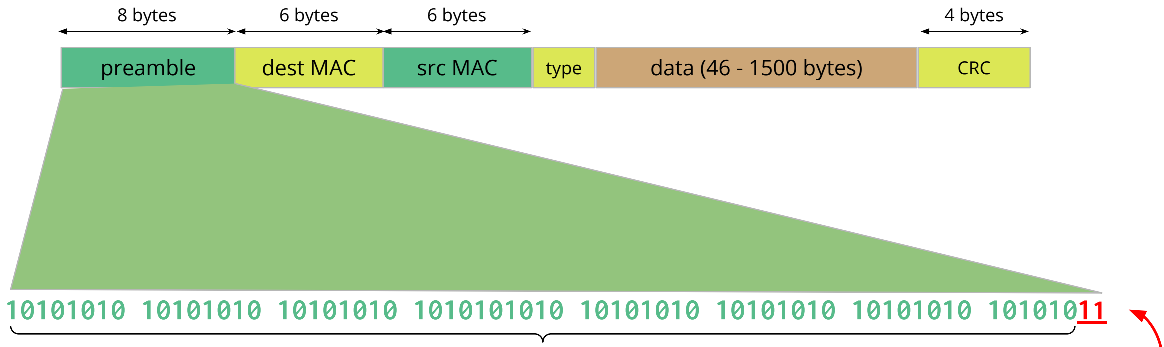
- Invented in 1970s by Bob Metcalfe and David Boggs
- Physical connections
  - Until mid 1990: bus topology (nodes can collide)
  - Hub-based Star topology (nodes do not collide)





Bob Metcalfe

Ethernet Frame Structure

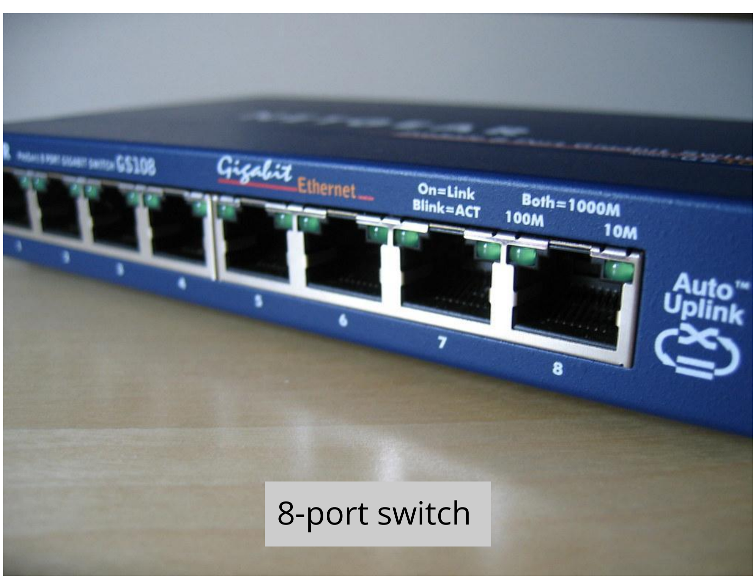


*bit pattern to **synchronize clock of both** the sender and recipient*

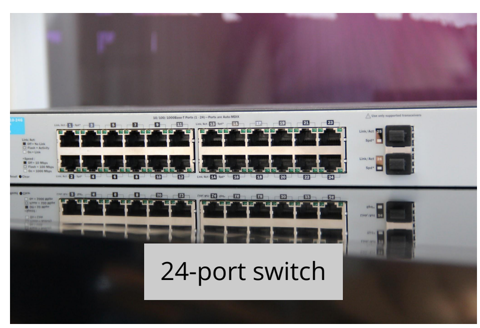
"beat count-in before your start playing instrument"

end of clock sync

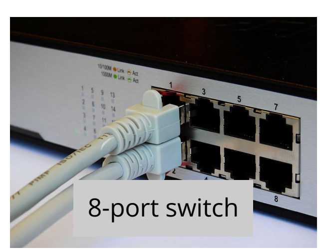
Ethernet Switches



8-port switch



24-port switch



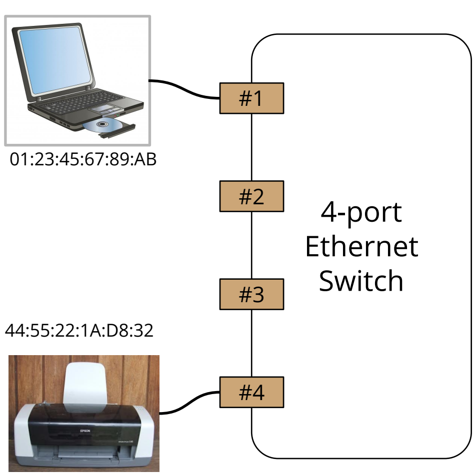
8-port switch

Ethernet Switches

- Each port is both INPUT & OUTPUT
- Frames from an INPUT port is stored, analyzed, and forwarded to an output port (based on destination MAC addr)
- Hosts are connected to a switch via a **dedicated port** in a star topology
    - No collision among hosts

What is the MAC address of the device connected to Port #1?

Ethernet Switches: Populate the Forwarding Table



Port	Device MAC	TTL
1	??:??:??:??:??:??	?
2	None	None
3	None	None
4	??:??:??:??:??:??	?

- Use **SOURCE MAC** address in **incoming** frames from the laptop/printer to fill-in the table above
- What if incoming frames use unregistered **DESTINATION MAC**?

Ethernet Switches: Forwarding "Algorithm"

Port	Device MAC	TTL
1	01:23:34:67:89:AB	300
2	None	None
3	23:A4:51:FD:00:07	300
4	44:55:22:1A:D8:32	300

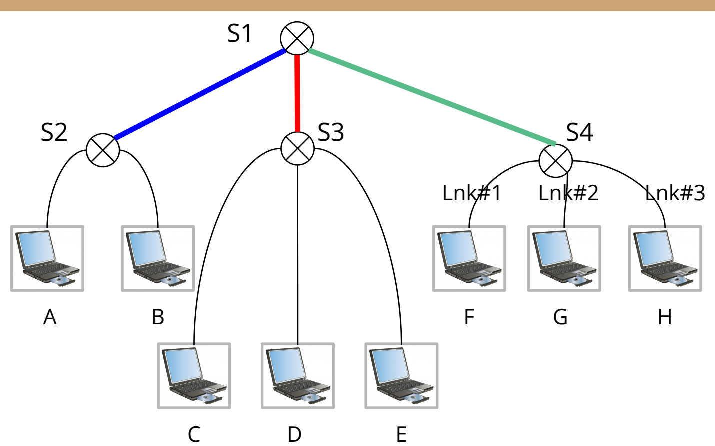
```
Sender MAC: 01:23:45:67:89:AB
Target MAC: 44:55:22:1A:D8:32
```

Target MAC is **registered** in the table.
Forward the incoming frame to Port #4

```
Sender MAC: 01:23:45:67:89:AB
Target MAC: 00:3F:5B:1A:00:02
```

Incoming Frame from Port #1.
Target MAC is not registered.
Broadcast to Ports #3 and #4

Hierarchy of Ethernet Switches



- Frames received by S1 from S4 with destination C,D,E will be forwarded
- Frames received by S1 from S3 with destination C, D, E will be dropped (erroneous packets?)

Forwarding Table @S1

MAC Addr	Port	TTL
A, B	Blue	
C, D, E	Red	
F, G	Green	

Forwarding Table @S4

MAC Addr	Port	TTL
A,B,C,D,E,	Green	
F	1	
G	2	

Virtual LANs

LANs: Physical vs. Virtual

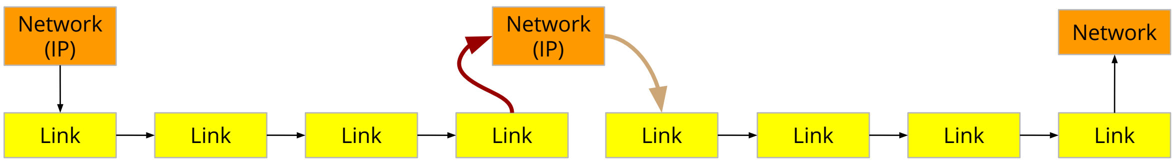
- In a physical LAN, hosts connected to the same switch (or group of switches) share the same broadcast traffic
- Smart **(programmable) switches** can be configured to partition *broadcast traffic* into one or more “islands” / ”logical boundaries”
    - Adjust the forwarding table to make ***broadcast frame traffic*** not to spill out from these “logical boundaries”
    - For instance on a 16-port switch
        - Assign ports 1-6 to the “Accounting” partition (broadcast from ports 1-6 will stay within ports 1-6)
        - Assign ports 7-14 to the “Marketing:” partition
        - Assign ports 15-16 to the “HR” partition
    - To handle bigger size partitions, these switches can be interconnected with one another

Partition Physical LAN into Virtual LANs

....and additional configuration by software

Can We Do Better than Traditional IP Routing?

IP Routing (or Not)



- Routing algorithm is performed by the Network Layer (IP Layer)
- The Link layer & switches perform *frame forwarding*
- Frame forwarding requires use of MAC addresses
- IP routing requires occasional frame **unpacking** (and **repacking**) by the IP/Network Layer

Better technique(s) than IP Routing?

Few options to improve IP routing

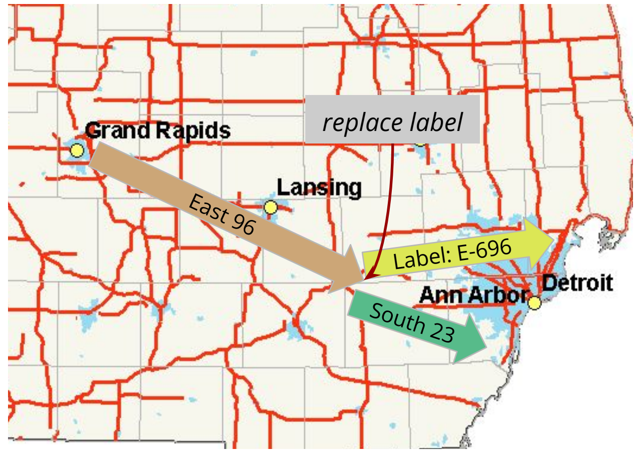
- Avoid frame unpacking and repacking by the Network Layer
- Create virtual Network layers (as opposed to physical Network Layer)
- Replace the Longest Prefix Matching with a faster technique
- Perform routing in the Link layer

MPLS

MPLS: Multi-Protocol Label Switching

- *The poor choice of color in the title is intentional*
- MP**LS** is a switching algorithm based on using labels (instead of IP dest)
  - "Multi-Protocol" is to emphasize that the technique can be implemented on top of other protocols (other than IP)
- Goal: packet routing/forwarding **only** by the Link layer (without involvement of the Network layer)
- General ideal
  - At the ingress router IP datagrams are assigned a label
  - En route to the destination, the label *may get replaced* with a new label by network switches
  - At the egress router, the label is removed and IP datagrams are passed to the Network Layer

By Destination vs By “Label”



Upstream node: Grand Rapids (**Source**)
Downstream nodes: Ann Arbor or Detroit (**Destination**)

Relevant RFCs

- RFC3031: Multiprotocol Switching Architecture
- RFC3032: MPLS Label Stack Encoding
- RFC3107: Carrying Label Information in BGP-4
- RFC3209: RSVP
- RFC5036: LDP Specification

Terminologies

	IP-based Routing	Label-based Switching
Routing Protocol	OSPF, BGP	Label Distribution Protocol
Routing Table	Routing Information Base*	Label Information Base
Forwarding Table	Forwarding Information base	Label Forwarding Information Base

Information Base or Database

Labels: Assignment & Distribution/Advertisement

- Labels are a fixed-length (32-bit) identifier which "encode" virtual path to destinations
- Labels are assigned locally by (and significant only on) a router
  - If a destination D is reachable from two routers R1 and R2, each router may assign a unique ID different from the other router
- Labels can be shared/distributed/advertised among router
  - Pushed by a downstream router to an upstream router
  - Pulled by an upstream router from a downstream router
  - Available Protocols
    - Label Distribution Protocol
    - Or piggyback on eBGP route announcement/advertisement

Initial Label Assignment (adjacent networks)

label range: 10-19 range: 20-29 range: 30-39 range: 40-49

Net Prefix	Label
1.1.0.0/16	11
2.2.0.0/16	12

Net Prefix	Label
2.2.0.0/16	21
3.3.0.0/16	22

Net Prefix	Label
3.3.0.0/16	31
4.4.0.0/16	32

Net Prefix	Label
4.4.0.0/16	41
5.5.0.0/16	42

Labels are significant only locally

Label Advertisement (From R1 to R2)

labels: 10-19

At R1

Net	Label
1.1.0.0/16	11
2.2.0.0/16	12

labels: 20-29

At R2

Net	Label
2.2.0.0/16	21
3.3.0.0/16	22

R1 advertised (1.1.0.0/16, 11) to R2

- R2 creates a new entry
- Assign a new local label (23)
- Associate the local label with the remote label (11)
- For R2, the label 11 is the outgoing label to go to 1.1.0.0/16

Update at R2

Net	Label	
	Local	Remote
2.2.0.0/16	21	
3.3.0.0/16	22	
1.1.0.0/16	23	11

Label Advertisement (R1 to R4)

At R1

Net	Label
1.1.0.0/16	11
2.2.0.0/16	12

At R2

Net	Label
2.2.0.0/16	21
3.3.0.0/16	22

At R3

Net	Label
3.3.0.0/16	31
4.4.0.0/16	32

At R4

Net	Label
4.4.0.0/16	41
5.5.0.0/16	42

R1 advertised (1.1.0.0/16, 11) to R2

Updated R2

Net	Label	
	Local	Rem
2.2.0.0/16	21	
3.3.0.0/16	22	
1.1.0.0/16	23	11

R2 advertised (1.1.0.0/16, 23) to R3

Updated R3

Net	Label	
	Local	Rem
3.3.0.0/16	31	
4.4.0.0/16	32	
1.1.0.0/16	33	23

R3 advertised (1.1.0.0/16, 33) to R4

Updated R4

Net	Label	
	Local	Rem
4.4.0.0/16	41	
5.5.0.0/16	42	
1.1.0.0/16	43	33

Label Advertisement (R4 to R1)

At R1

Net	Label
1.1.0.0/16	11
2.2.0.0/16	12

At R2

Net	Label
2.2.0.0/16	21
3.3.0.0/16	22

At R3

Net	Label
3.3.0.0/16	31
4.4.0.0/16	32

At R4

Net	Label
4.4.0.0/16	41
5.5.0.0/16	42

R4 advertised (5.5.0.0/16, 42) to R3

Updated R3

Net	Label	
	Local	Rem
4.4.0.0/16	41	
5.5.0.0/16	42	

R3 advertised (5.5.0.0/16, 33) to R2

Updated R2

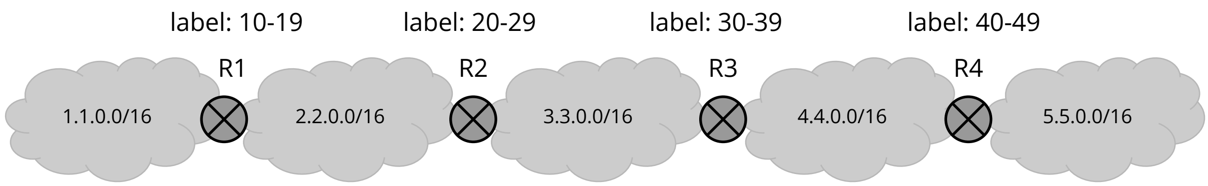
Net	Label	
	Local	Rem
3.3.0.0/16	31	
4.4.0.0/16	32	
5.5.0.0/16	33	42

R2 advertised (1.1.0.0/16, 33) to R1

Updated R1

Net	Label	
	Local	Rem
2.2.0.0/16	21	
3.3.0.0/16	22	
5.5.0.0/16	23	11

After Label Distribution



Net	Label @R1	
	Loc	Rem
1.1.0.0/16	11	-
2.2.0.0/16	12	-
3.3.0.0.16	13	22
4.4.0.0/16	14	24
5.5.0.0/16	15	25

Net	Label @R2	
	Loc	Rem
1.1.0.0/16	23	11
2.2.0.0/16	21	-
3.3.0.0.16	22	-
4.4.0.0/16	24	32
5.5.0.0/16	25	35

Net	Label @R3	
	Loc	Rem
1.1.0.0/16	33	23
2.2.0.0/16	34	21
3.3.0.0.16	31	-
4.4.0.0/16	32	-
5.5.0.0/16	35	42

Net	Label @R4	
	Loc	Rem
1.1.0.0/16	43	33
2.2.0.0/16	44	34
3.3.0.0.16	45	31
4.4.0.0/16	41	-
5.5.0.0/16	42	-

Local Forwarding Table

Net	Label @R1	
	Loc	Rem
3.3.0.0.16	13	22
4.4.0.0/16	14	24
5.5.0.0/16	15	25

Net	Label @R2	
	Loc	Rem
1.1.0.0/16	23	11
4.4.0.0/16	24	32
5.5.0.0/16	25	35

Net	Label @R3	
	Loc	Rem
1.1.0.0/16	33	23
2.2.0.0/16	34	21
5.5.0.0/16	35	42

Net	Label @R4	
	Loc	Rem
1.1.0.0/16	43	33
2.2.0.0/16	44	34
3.3.0.0.16	45	31
5.5.0.0/16	42	-

(Forward|Rout)ing Example: From 1.1.0.14 to 5.5.0.23

Net	Label @R1	
	Loc	Rem
3.3.0.0.16	13	22
4.4.0.0/16	14	24
5.5.0.0/16	15	25

Net	Label @R2	
	Loc	Rem
1.1.0.0/16	23	11
4.4.0.0/16	24	32
5.5.0.0/16	**25**	**35**

Net	Label @R3	
	Loc	Rem
1.1.0.0/16	33	23
2.2.0.0/16	34	21
5.5.0.0/16	35	42

Net	Label @R4	
	Loc	Rem
1.1.0.0/16	43	33
2.2.0.0/16	44	34
3.3.0.0.16	45	31
5.5.0.0/16	42	-

```
DST IP: 5.5.0.23
LABEL: 25 (set by R1)
```
payload

```
LABEL: 35 (swapped by R2)
```
payload

```
LABEL: 42 (swapped by R3)
```
payload

```
Label is removed by R4)
```
payload