

# Topic 3: The Data Link Layer

## Our goals:

- understand principles behind data link layer services:  
(these are methods & mechanisms in your networking toolbox)
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control
- instantiation and implementation of various link layer technologies
  - Wired Ethernet (aka 802.3)
  - Wireless Ethernet (aka 802.11 WiFi)
- Algorithms
  - Binary Exponential Backoff
  - Spanning Tree

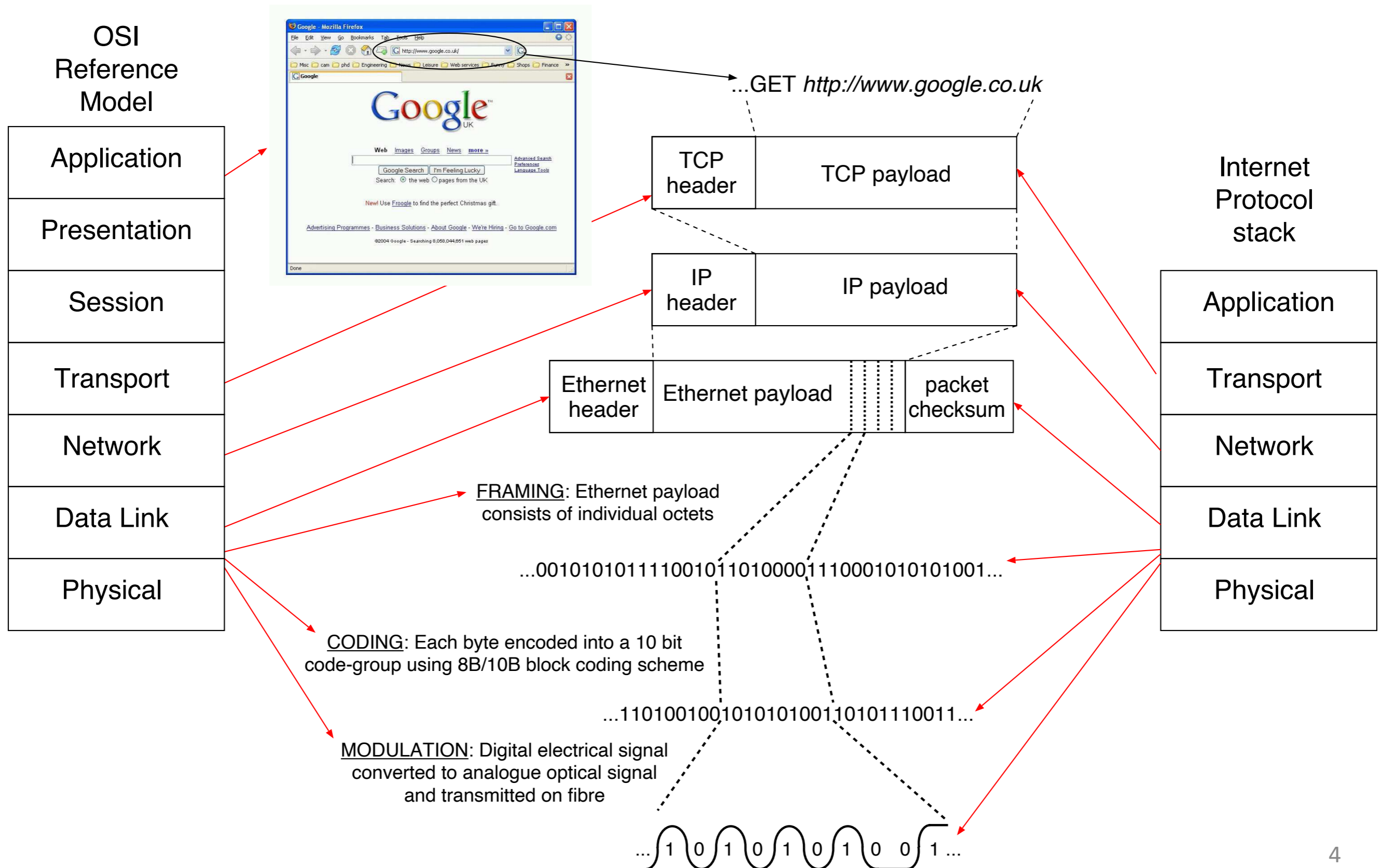
# Topic 3: The Data Link Layer

## Our goals:

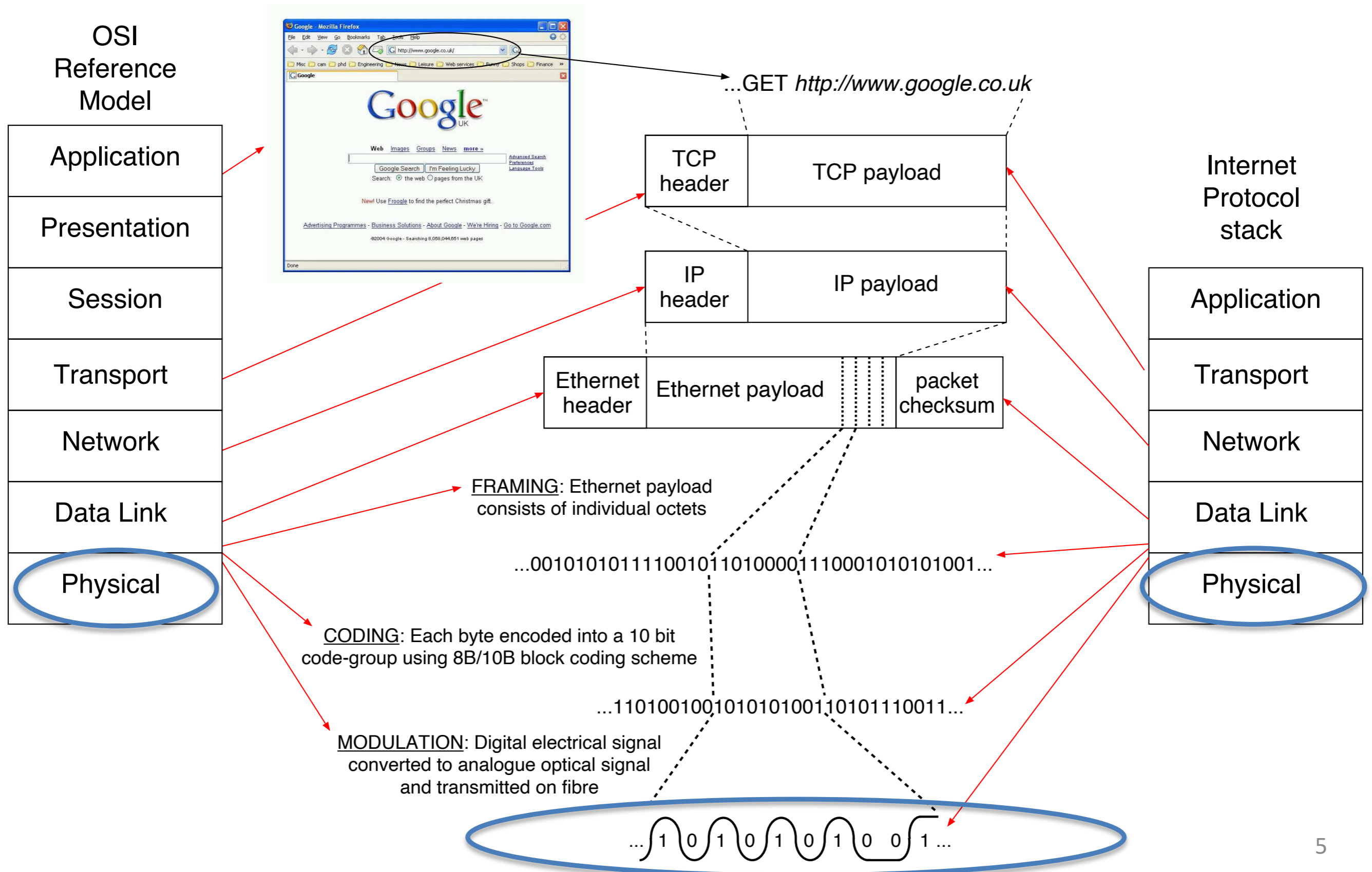
- understand principles behind data link layer protocols (these are methods & mechanisms in your network stack)
  - error detection, correction
  - sharing a broadcast channel
  - link layer addressing
  - reliable data transfer
  - flow control
- installation and implementation of various link layer technologies
  - Wired Ethernet (aka 802.3)
  - Wireless Ethernet (aka 802.11 WiFi)
- Algorithms
  - Binary Exponential Backoff
  - Spanning Tree

**But first a word or two about the Physical layer**

# Internet protocol stack *versus* OSI Reference Model



# Internet protocol stack *versus* OSI Reference Model





# Physical Channels / The Physical Layer

these example physical channels are also known as *Physical Media*

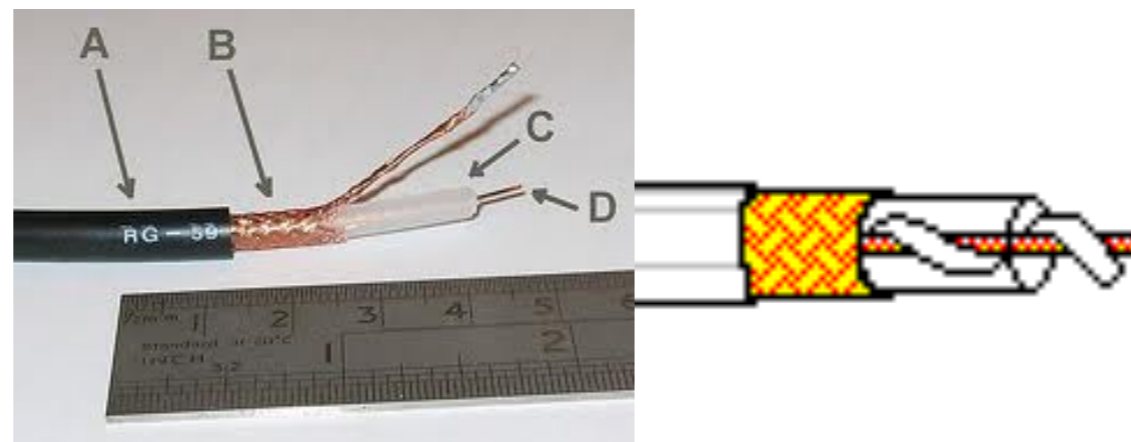
## Twisted Pair (TP)

- two insulated copper wires
  - Category 3: traditional phone wires, 10 Mbps Ethernet
  - Category 8: 25Gbps Ethernet
- Shielded (STP)
- Unshielded (UTP)



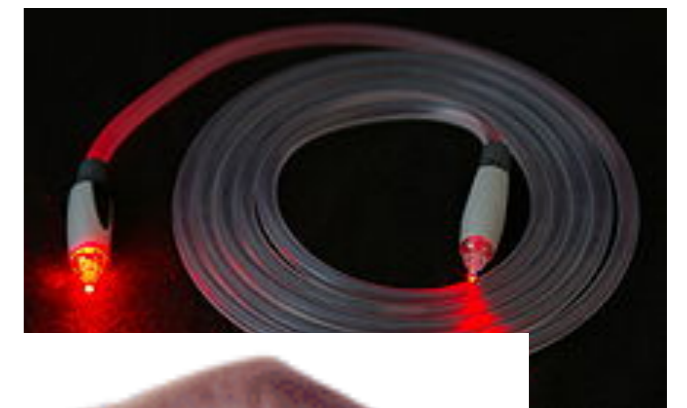
## Coaxial cable:

- two concentric copper conductors
- bidirectional
- baseband:
  - single channel on cable
  - legacy Ethernet
- broadband:
  - multiple channels on cable
  - HFC (Hybrid Fiber Coax)



## Fiber optic cable:

- high-speed operation
- point-to-point transmission
- (10' s-100' s Gbps)
- low error rate
- immune to electromagnetic noise



# More Physical media: **Radio**

- Bidirectional and multiple access
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

## **Radio link types:**

- ❑ **terrestrial microwave**
  - ❖ e.g. 90 Mbps channels
- ❑ **LAN** (e.g., Wifi)
  - ❖ 11Mbps, 54 Mbps, 600 Mbps
- ❑ **wide-area** (e.g., cellular)
  - ❖ 5G cellular: ~ 40 Mbps - 10Gbps
- ❑ **satellite**
  - ❖ 27-50MHz typical bandwidth
  - ❖ geosynchronous versus low altitude
  - ❖ 270 msec end-end delay to orbit



# Topic 3: The Data Link Layer

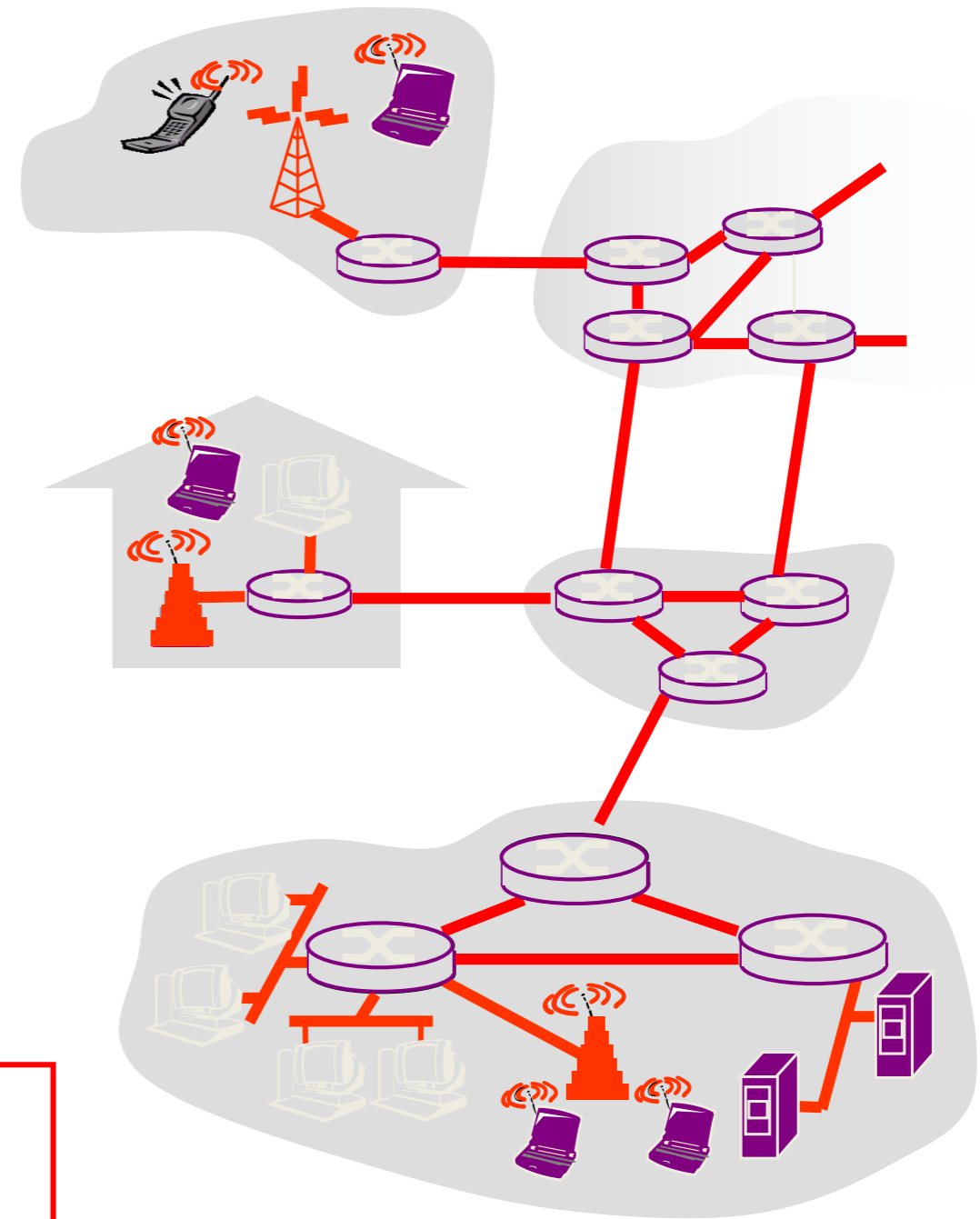
## Our goals:

- understand principles behind data link layer services:  
(these are methods & mechanisms in your networking toolbox)
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control
- instantiation and implementation of various link layer technologies
  - Wired Ethernet (aka 802.3)
  - Wireless Ethernet (aka 802.11 WiFi)
- Algorithms
  - Binary Exponential Backoff
  - Spanning Tree

# Link Layer: Introduction

## Some terminology:

- hosts and routers are **nodes**
- communication channels that connect adjacent nodes along communication path are **links**
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a **frame**, encapsulates datagram



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

# Link Layer (Channel) Services

- *framing, physical addressing:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we see some of this again in the Transport Topic
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates

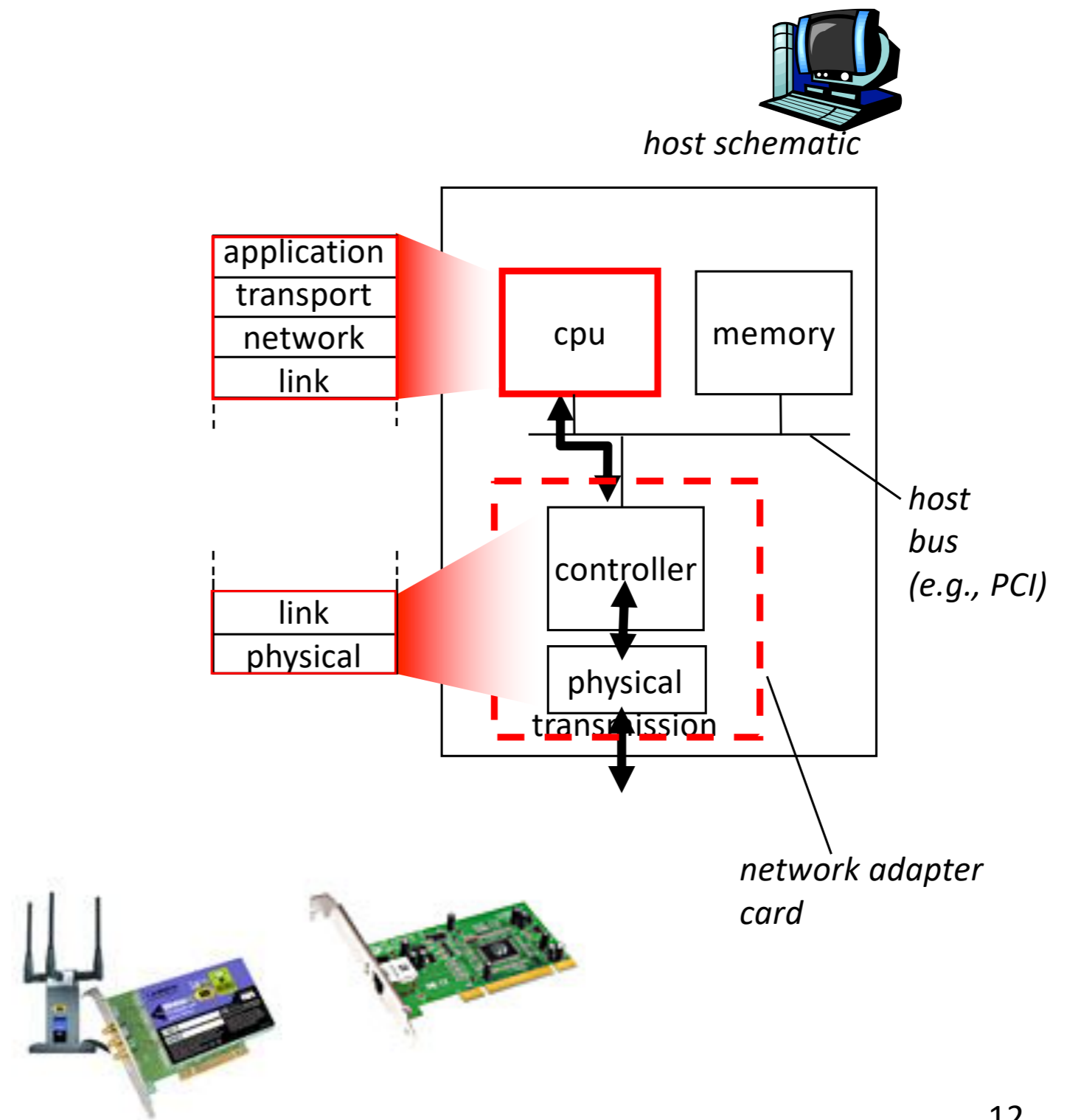
# Link Layer (Channel) Services - 2

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

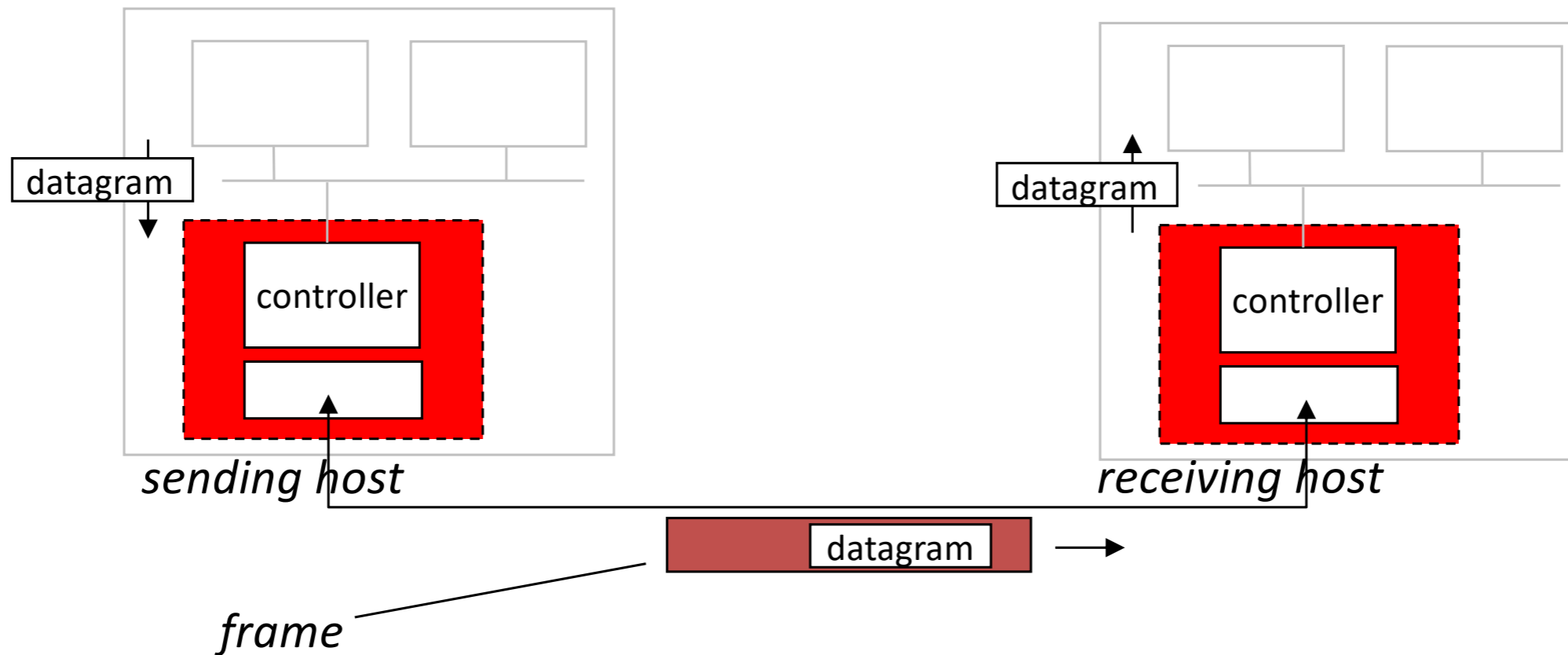


# Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka *network interface card* NIC)
  - Ethernet card, PCMCIA card, 802.11 card
  - implements link, physical layer
- attaches into host’s system buses
- combination of hardware, software, firmware



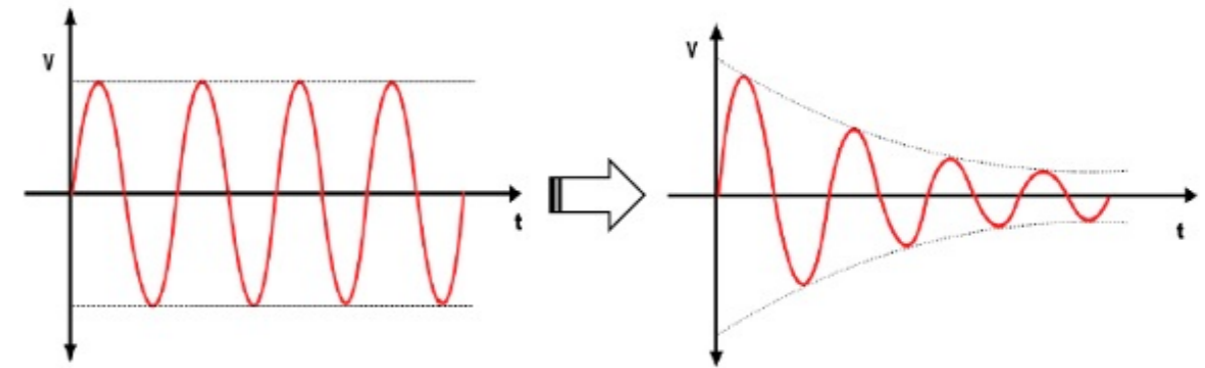
# Adaptors Communicating



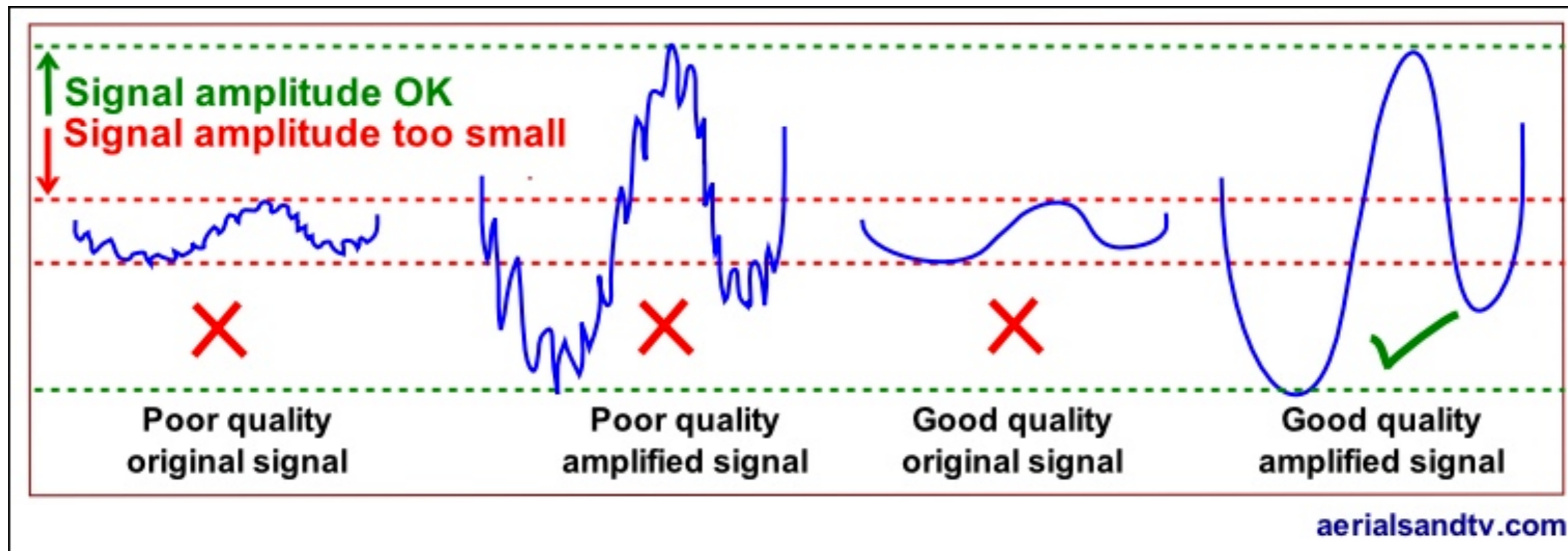
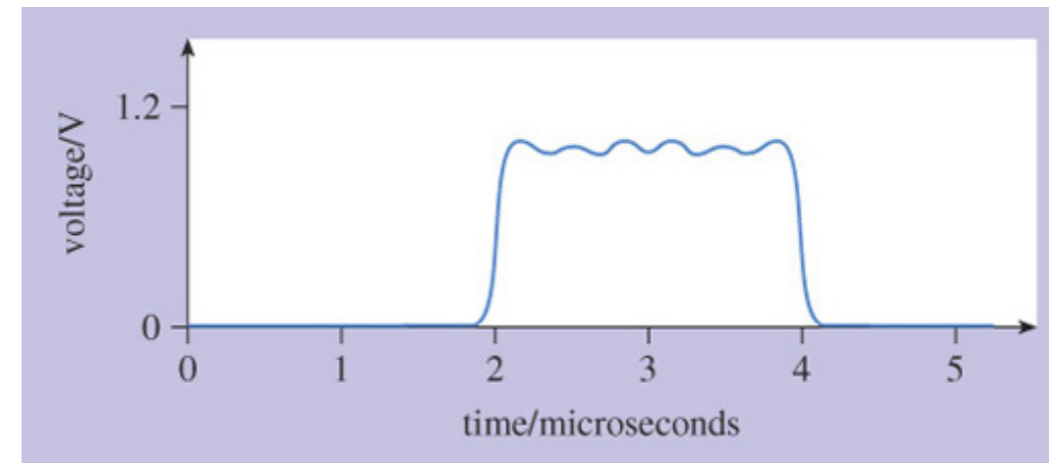
- sending side:
  - encapsulates datagram in frame
  - encodes data for the physical layer
  - adds error checking bits, provide reliability, flow control, etc.
- receiving side
  - decodes data from the physical layer
  - looks for errors, provide reliability, flow control, etc
  - extracts datagram, passes to upper layer at receiving side



# Enemies of Communications

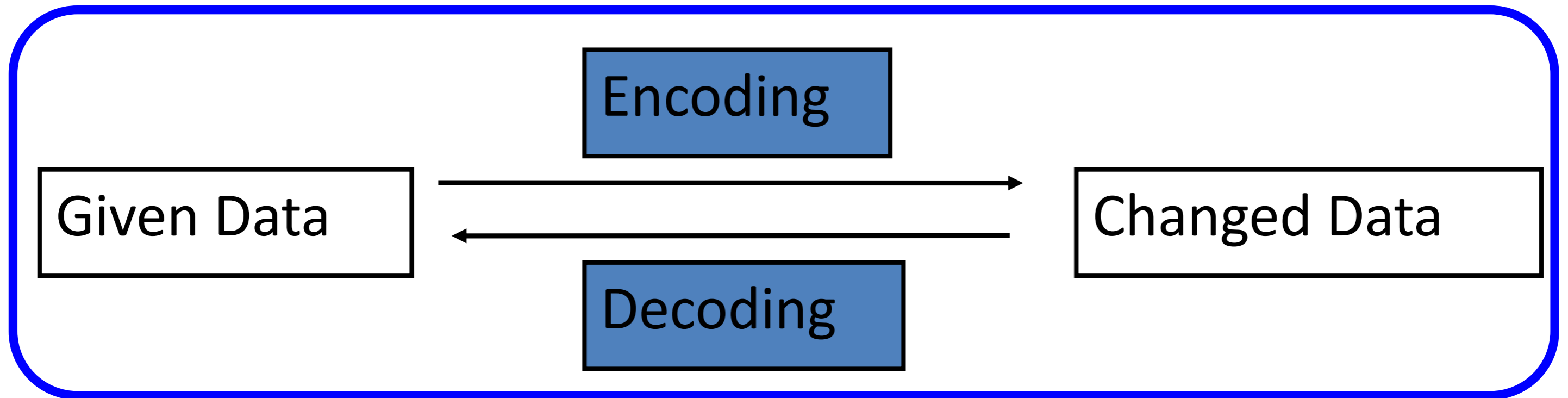


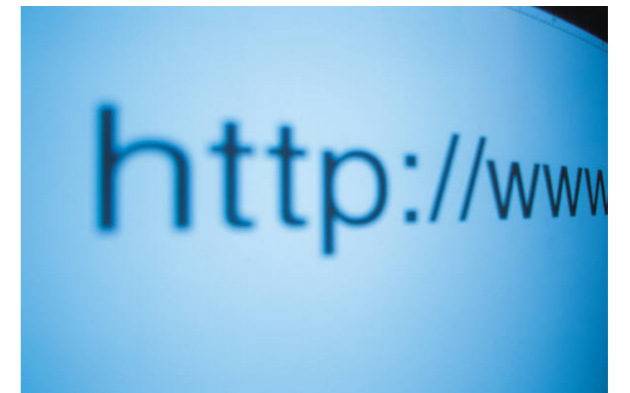
Attenuation, External Noise, Systematic, non-systematic, digitization, interference, ....



# Coding – a channel function

Change the representation of data.





MyPasswd

MyPasswd



AA\$\$\$\$ff

AA\$\$\$\$ff



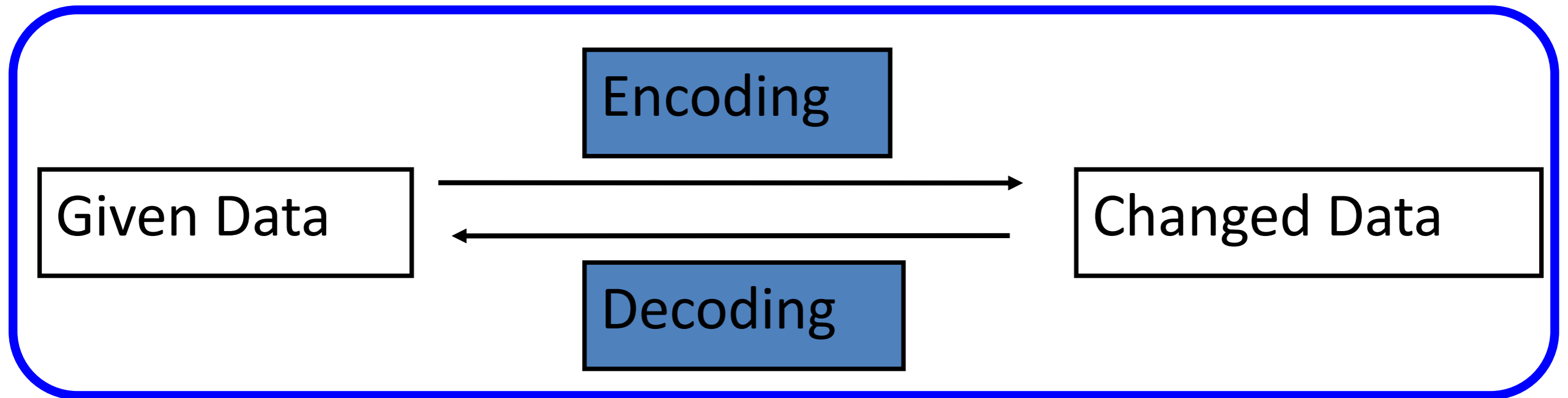
AA\$\$\$\$ffff


AA\$\$\$\$ffff



# Coding

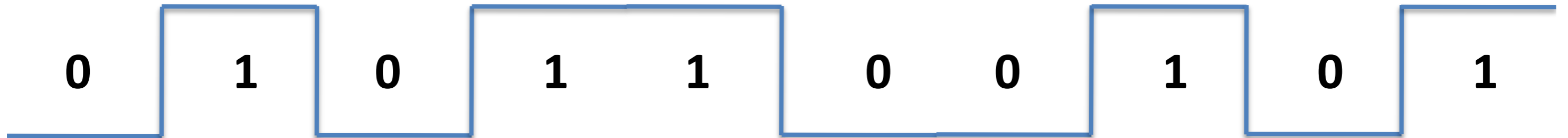
Change the representation of data.



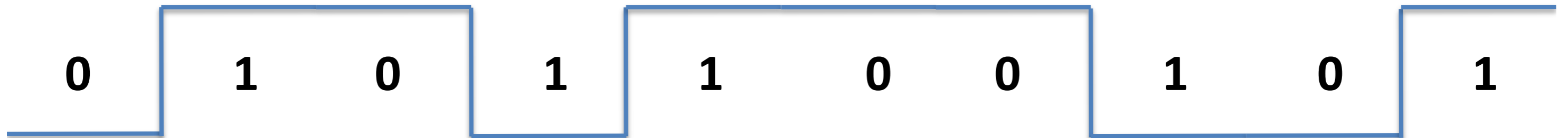
1. **Encryption:** MyPasswd  $\leftrightarrow$  AA\$\$\$\$ff
2. **Error Detection:** AA\$\$\$\$ff  $\leftrightarrow$  AA\$\$\$\$ffff
3. **Compression:** AA\$\$\$\$ffff  $\leftrightarrow$  A2\$4f4
4. **Analog:** A2\$4f4  $\leftrightarrow$  

# Line Coding Examples where Baud=bit-rate

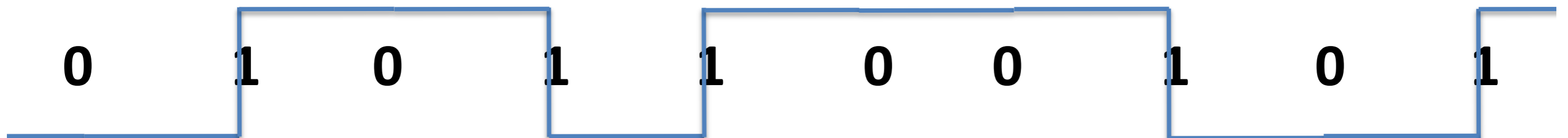
Non-Return-to-Zero (NRZ)



Non-Return-to-Zero-Mark (NRZM) 1 = transition 0 = no transition

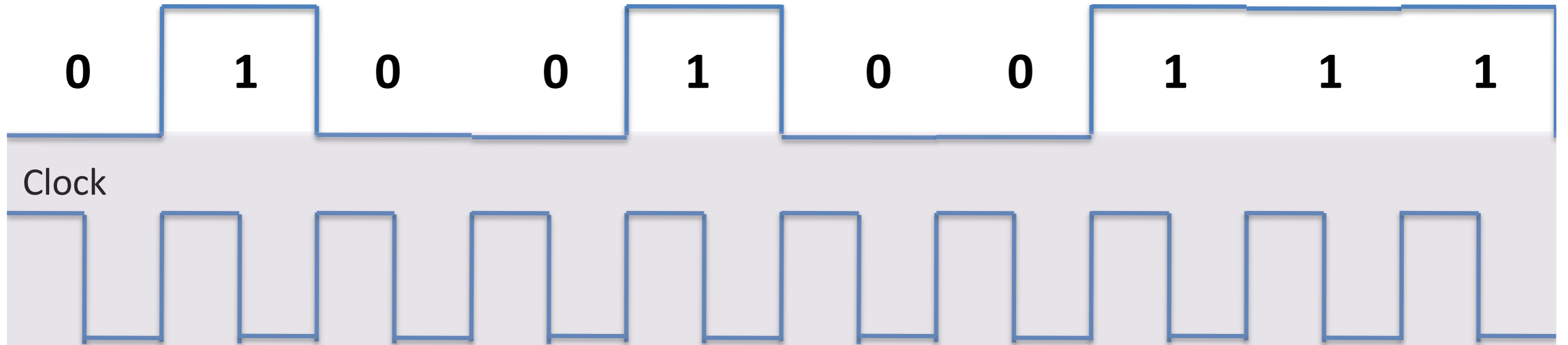


Non-Return-to-Zero Inverted (NRZI) (note transitions on the 1)

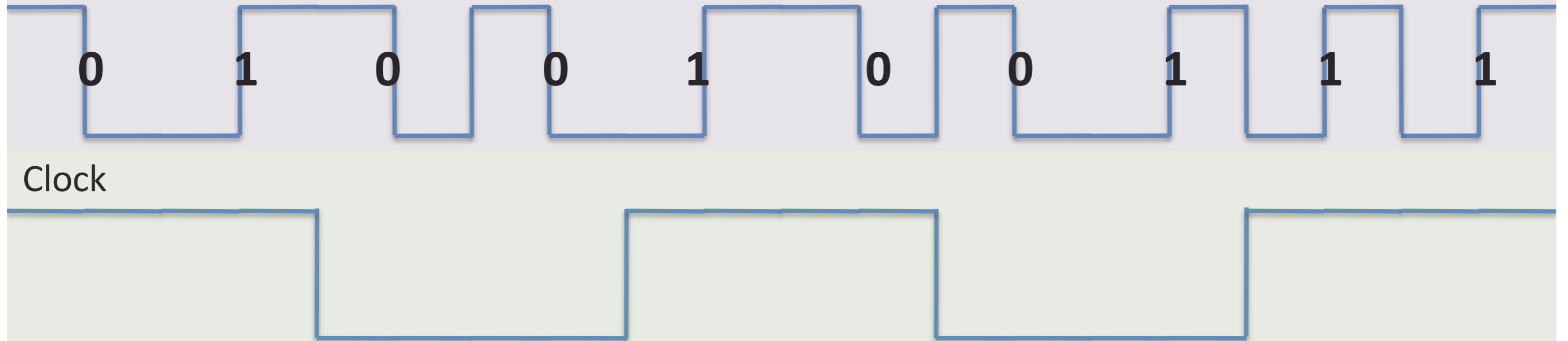


# Line Coding Examples

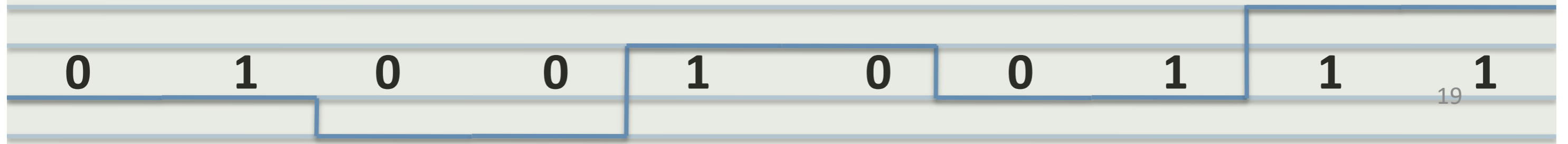
Non-Return-to-Zero (NRZ) (Baud = bit-rate)



Manchester example (Baud = 2 x bit-rate)



Quad-level code (2 x Baud = bit-rate)



# Line Coding – Block Code example

Data to send



Line-(Wire) representation



Name	4b	5b	Description	Name	4b	5b	Description
0	0000	11110	hex data 0	Q	-NONE-	00000	Quiet
1	0001	01001	hex data 1	I	-NONE-	11111	Idle
2	0010	10100	hex data 2	J	-NONE-	11000	SSD #1
3	0011	10101	hex data 3	K	-NONE-	10001	SSD #2
4	0100	01010	hex data 4	T	-NONE-	01101	ESD #1
5	0101	01011	hex data 5	R	-NONE-	00111	ESD #2
6	0110	01110	hex data 6	H	-NONE-	00100	Halt
7	0111	01111	hex data 7				
8		1000	10010 hex data 8				
9		1001	10011 hex data 9				
A	1010	10110	hex data A				
B	1011	10111	hex data B				
C	1100	11010	hex data C				
D	1101	11011	hex data D				
E	1110	11100	hex data E				
F	1111	11101	hex data F				

Block coding transfers data with a fixed overhead: 20% less information per Baud in the case of 4B/5B

So to send data at 100Mbps; the line rate (the Baud rate) must be 125Mbps.

1Gbps uses an 8b/10b codec; encoding entire bytes at a time but with 25% overhead



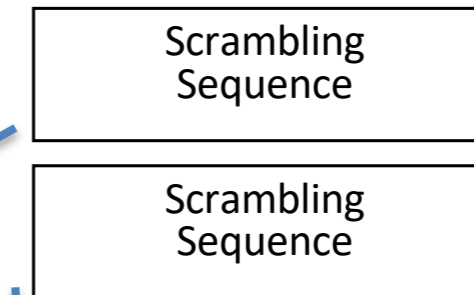
# Line Coding Scrambling – with secrecy

Step 1

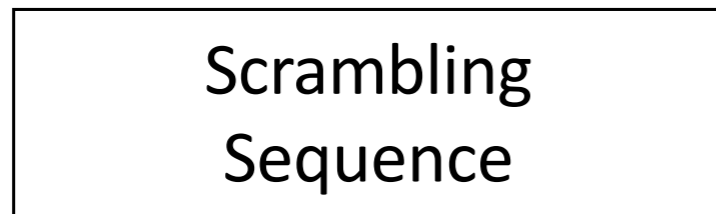


....G8wDFrB  
EAFDSWbzQ7  
BW2fbdTqeT  
ImrukTYwQY  
ndYdKb4....

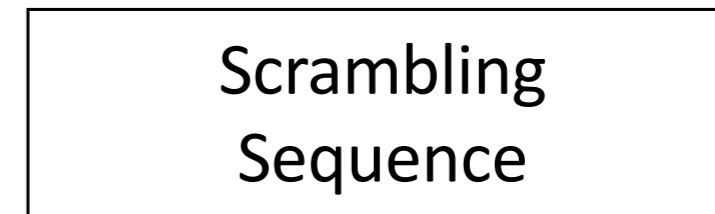
REPLICATE  
SECURELY



Step 2



DISTRIBUTE  
SECURELY



Message



Message



Message  
XOR  
Sequence



Message  
XOR  
Sequence



Step 3

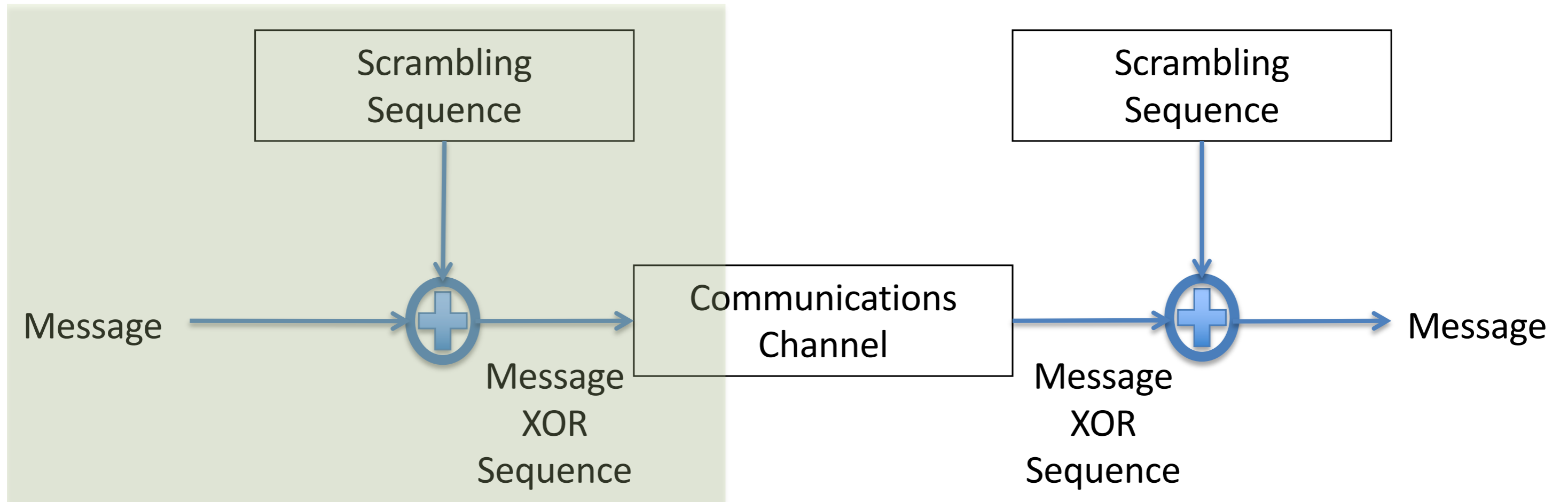
Don't ever reuse Scrambling sequence, ever. <<< **this is quite important**

Whitfield  
Diffie

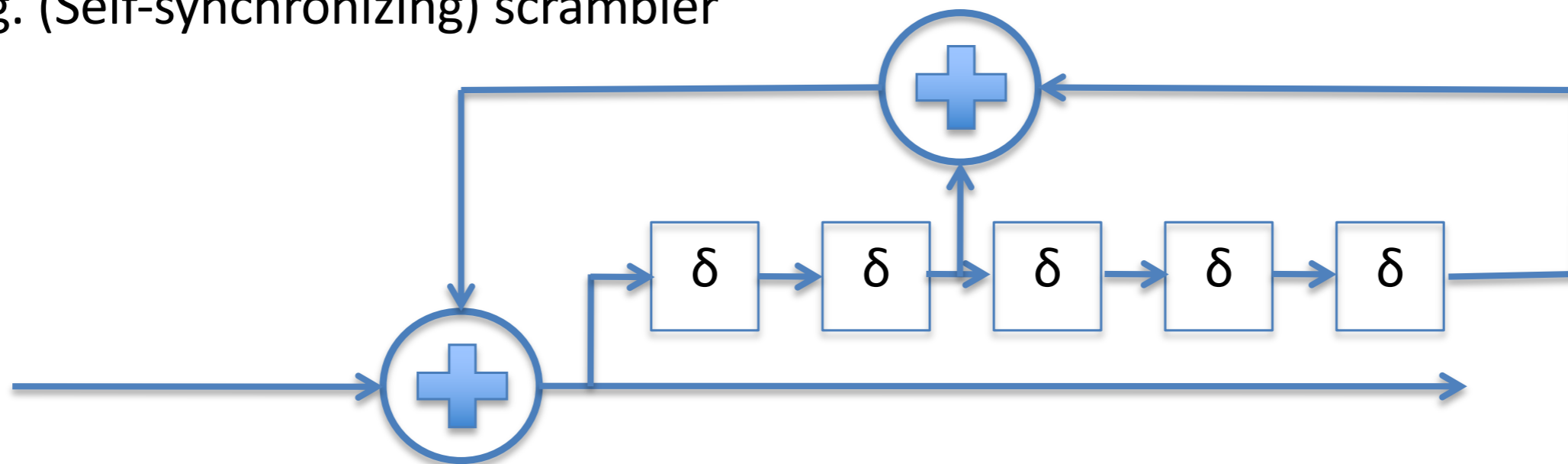
Martin  
Hellman



# Line Coding Scrambling– no secrecy



e.g. (Self-synchronizing) scrambler



# Line Coding Examples (Hybrid)

...100111101101010001000101100111010001010010110101001001110101110100...  
 ...10011110110101000101000101100111010001010010110101001001110101110100...



Inserted bits marking “start of frame/block/sequence”

Scramble / Transmit / Unscramble



...0100010110011101000101001011010100100111010111010010010111011101111000...

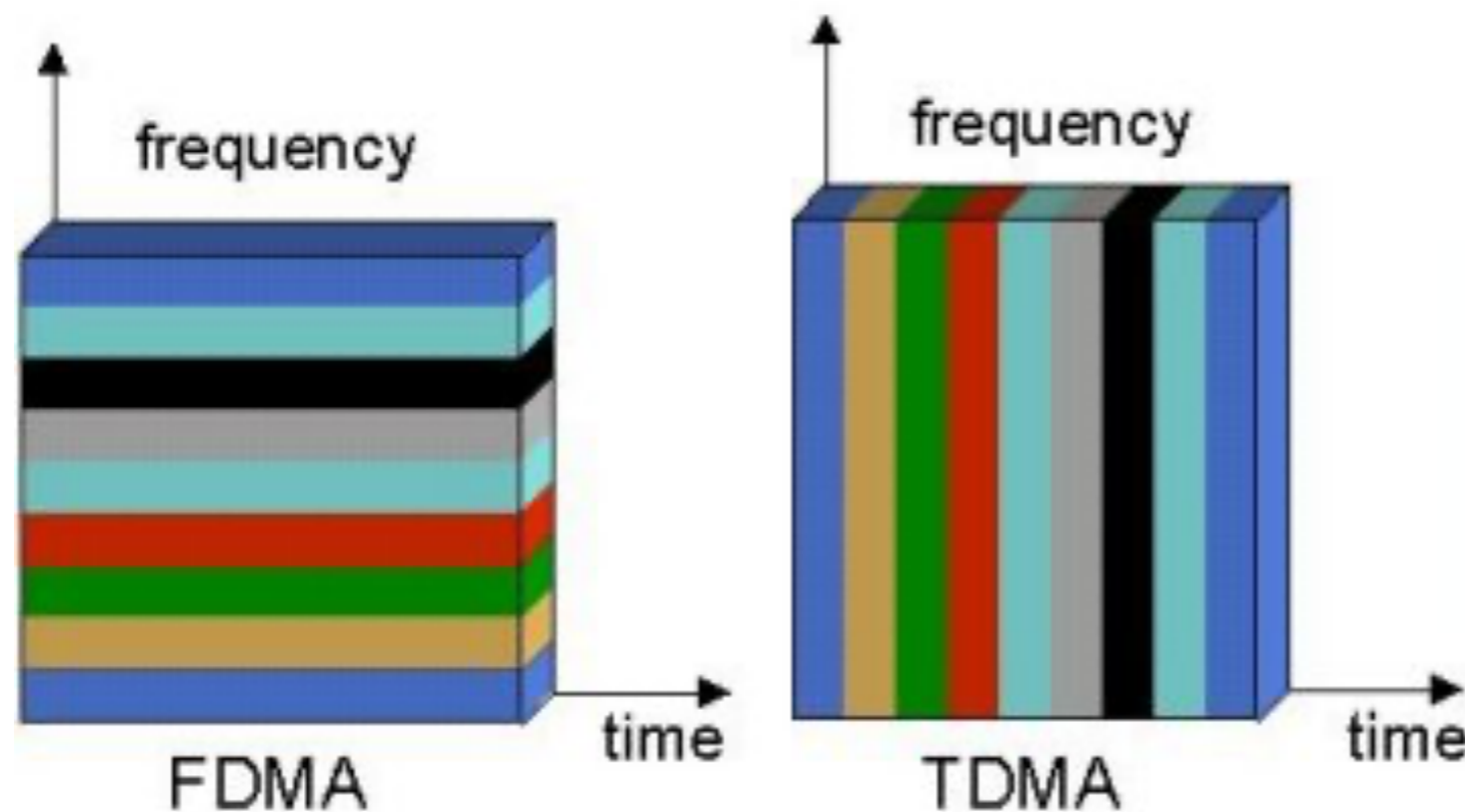


Identify (and remove) “start of frame/block/sequence”

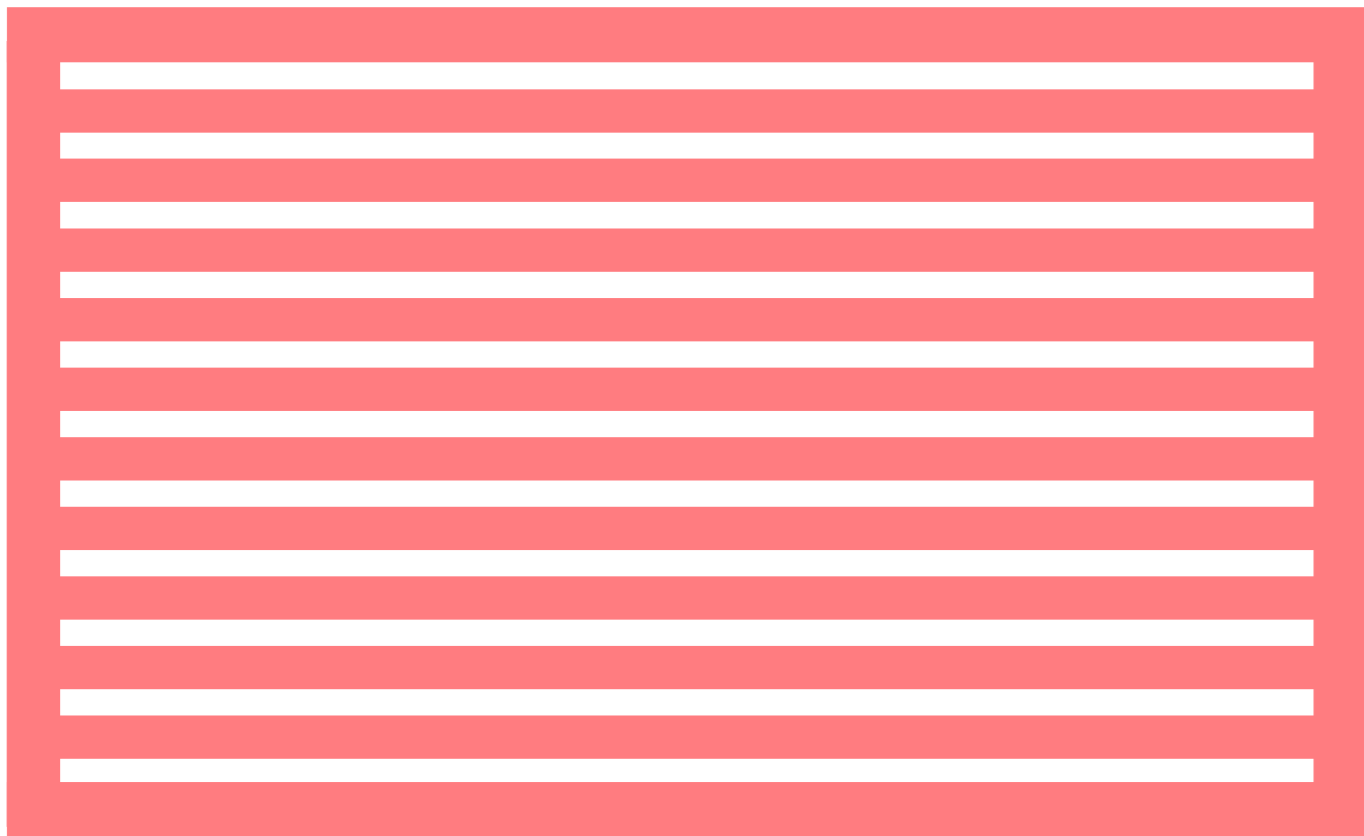
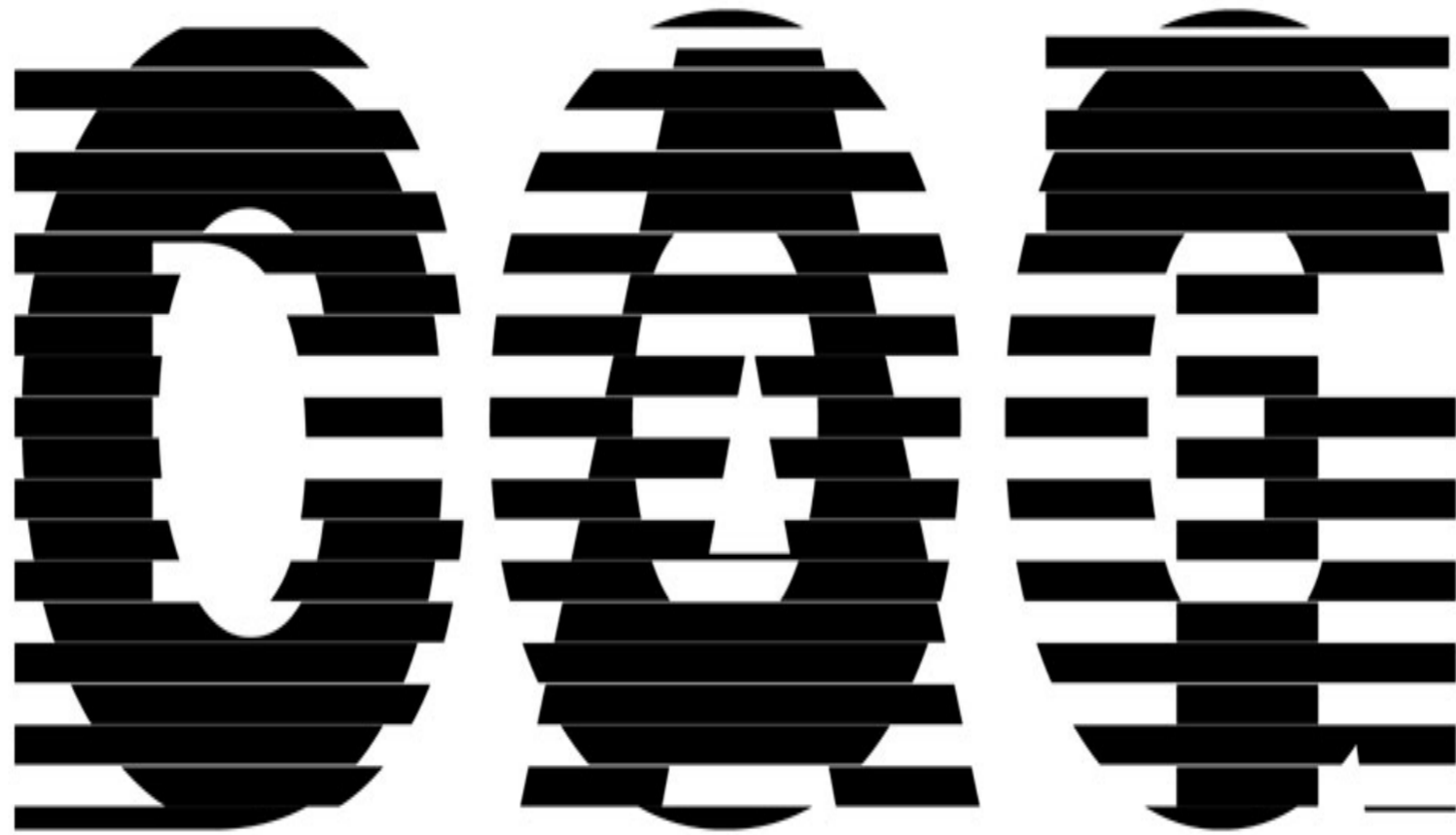
This gives you the Byte-delineations for *free*

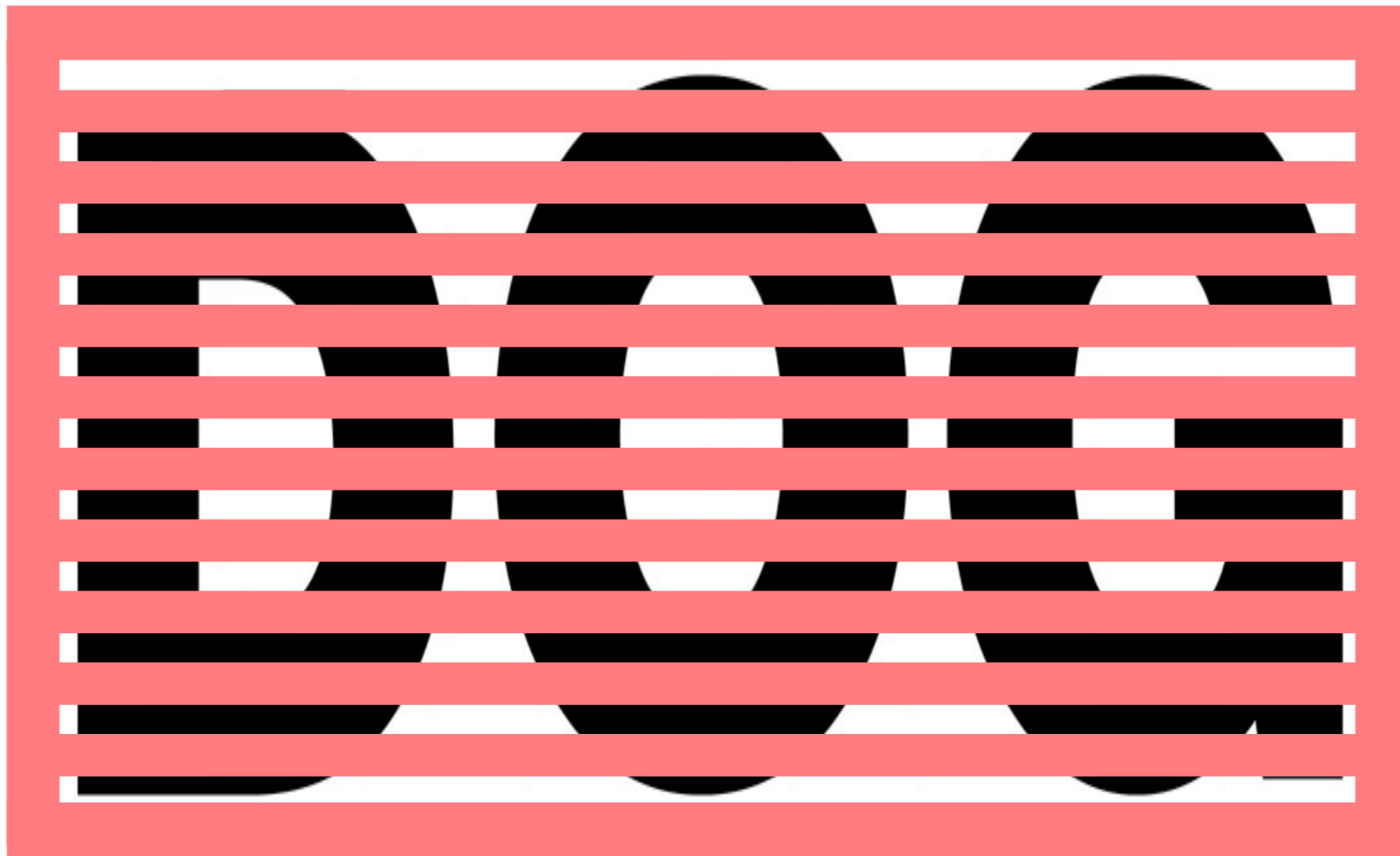
64b/66b combines a scrambler and a framer. The start of frame is a pair of bits 01 or 10: 01 means “this frame is data” 10 means “this frame contains data and control” – control could be configuration information, length of encoded data or simply “this line is idle” (no data at all)

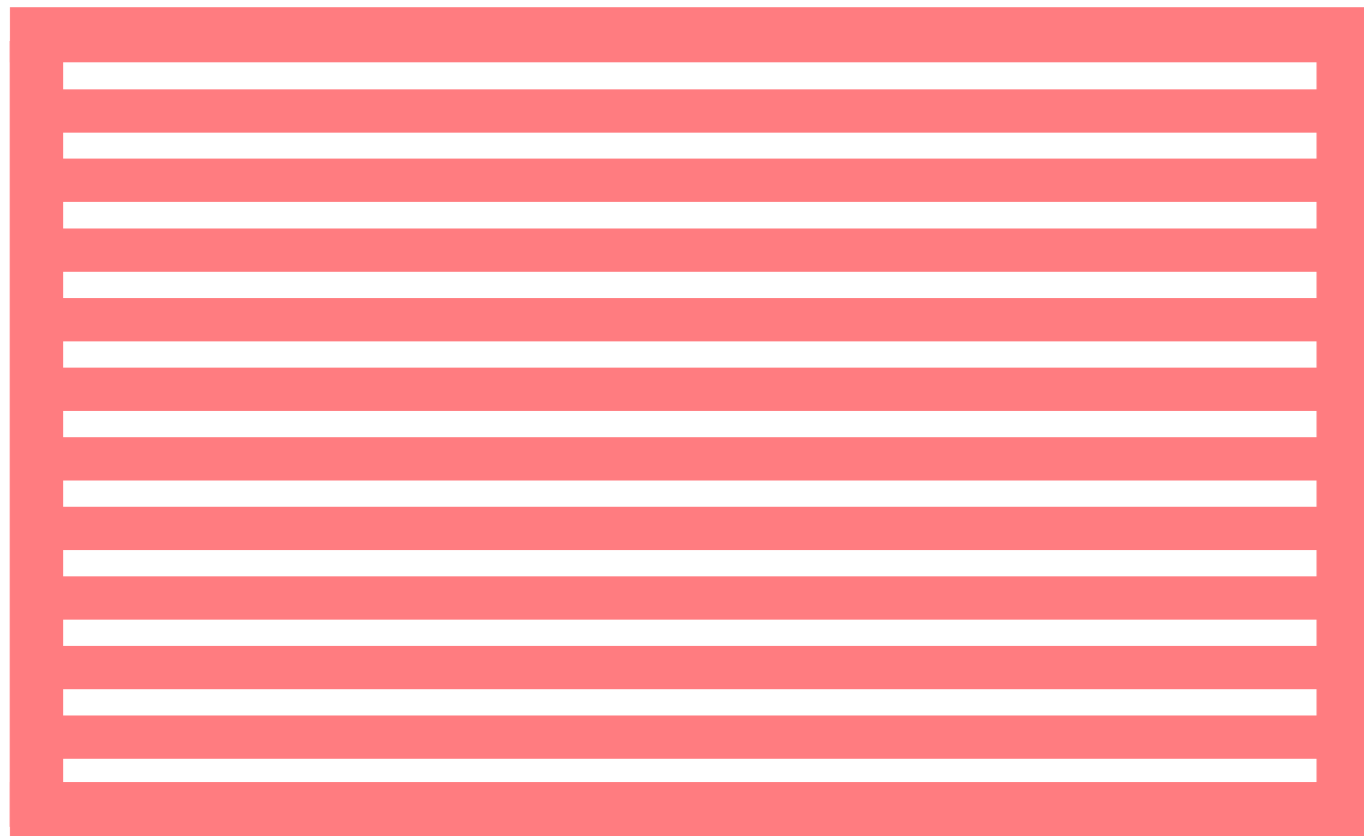
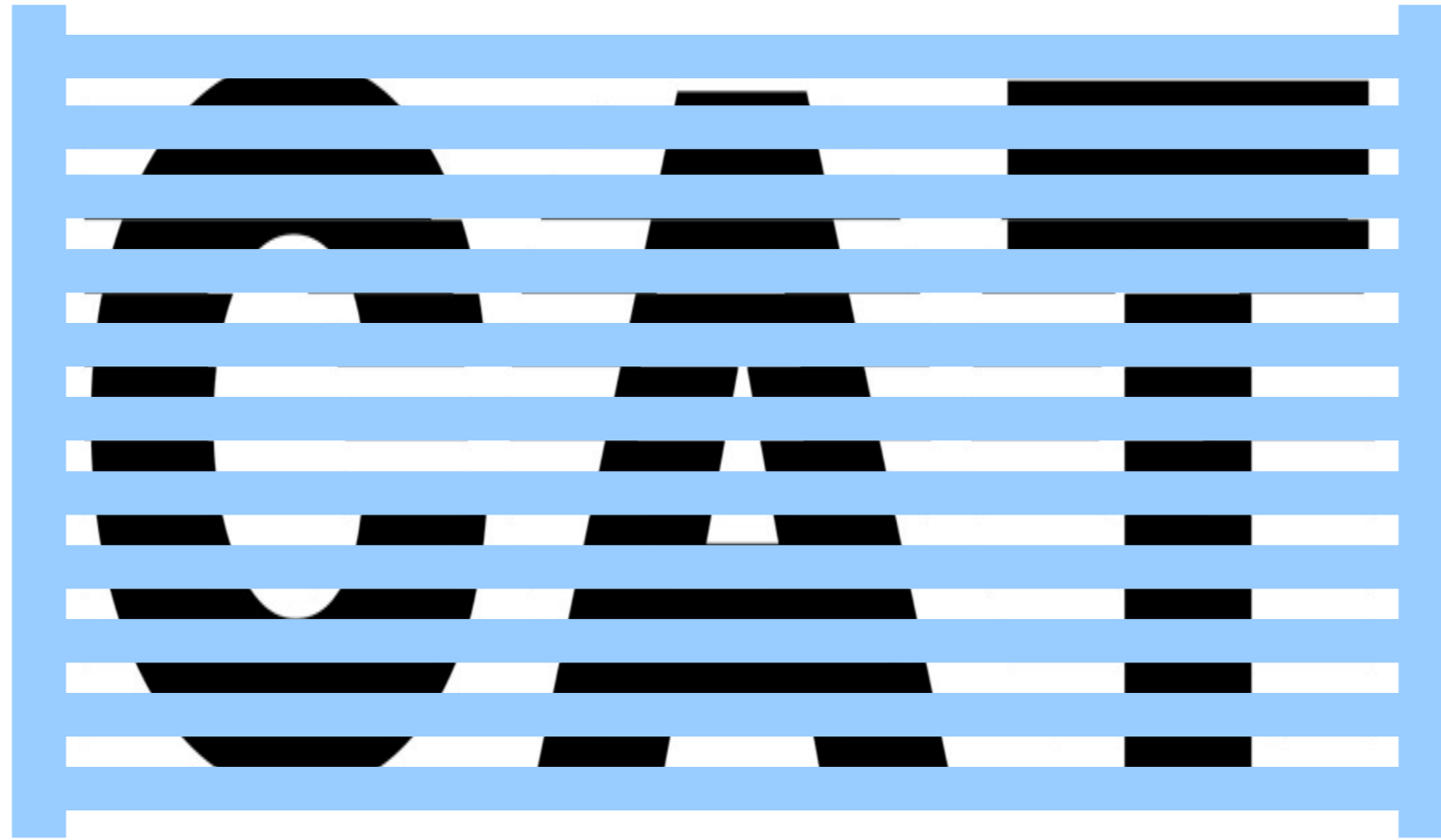
# Multiple Access Mechanisms



Each dimension is orthogonal (so may be trivially combined)  
Other dimensions may also be available...





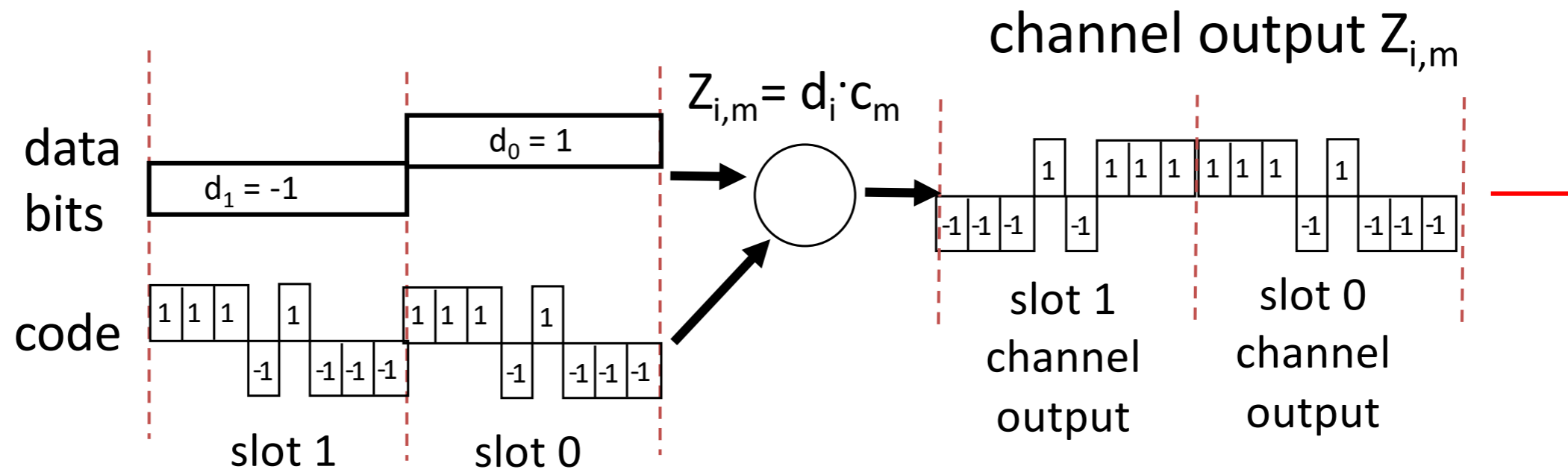


# Code Division Multiple Access (CDMA) (not to be confused with CSMA!)

- used in several wireless broadcast channels (cellular, satellite, etc) standards
- unique “code” assigned to each user; i.e., code set partitioning
- all users share same frequency, but each user has own “chipping” sequence (i.e., code) to encode data
- *encoded signal* = (original data) XOR (chipping sequence)
- *decoding*: inner-product of encoded signal and chipping sequence
- allows multiple users to “coexist” and transmit simultaneously with minimal interference (if codes are “orthogonal”)

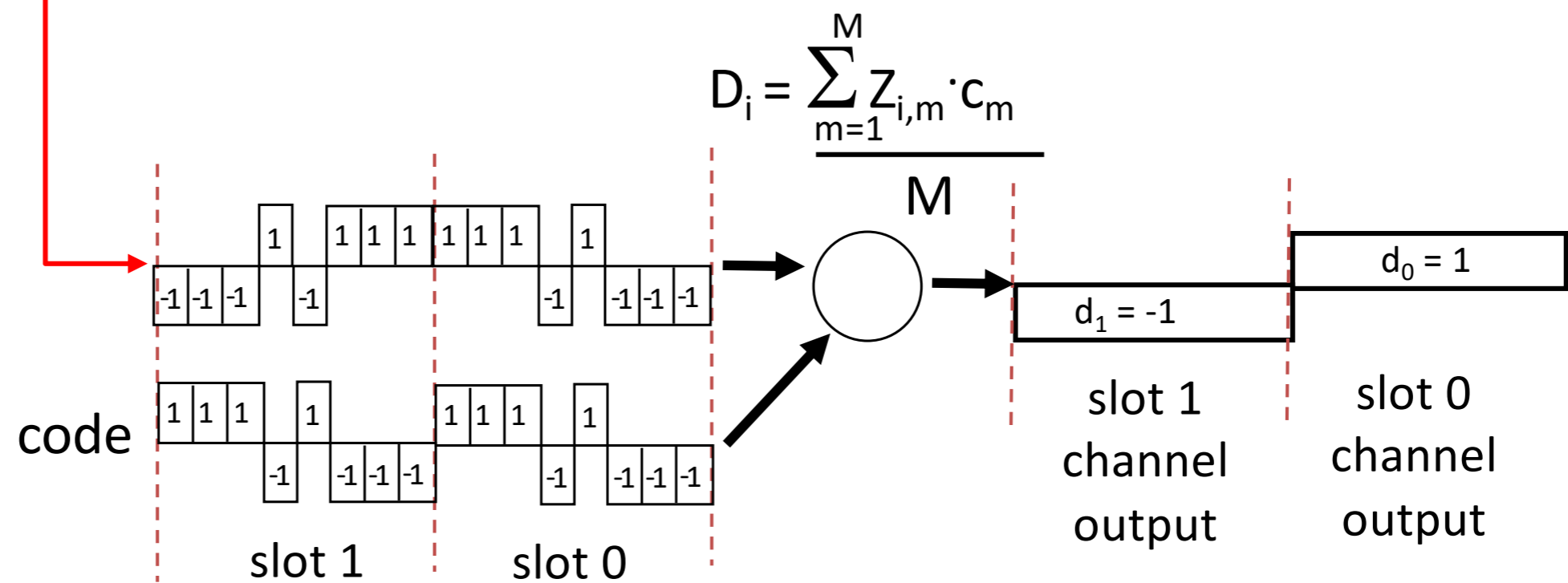
# CDMA Encode/Decode

sender  
adds code



received  
input

receiver  
removes code

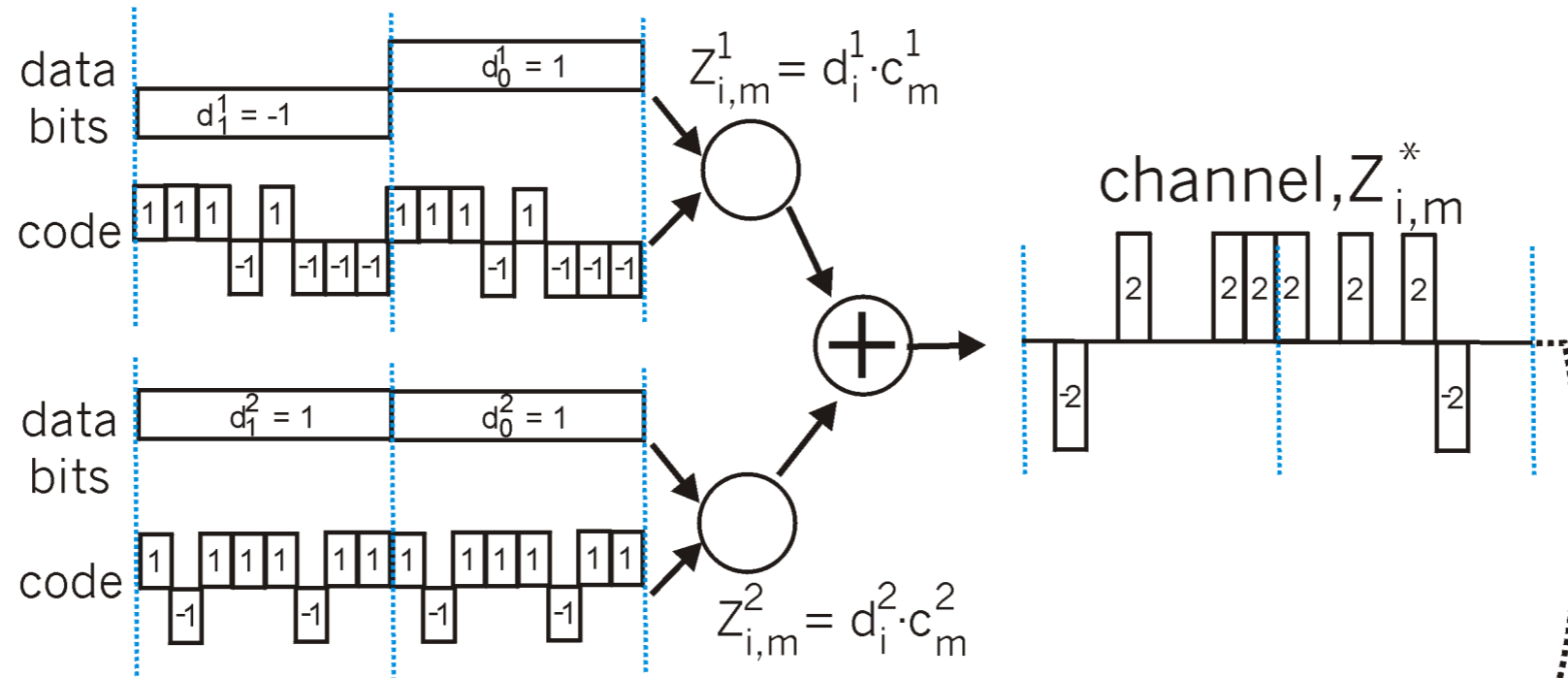




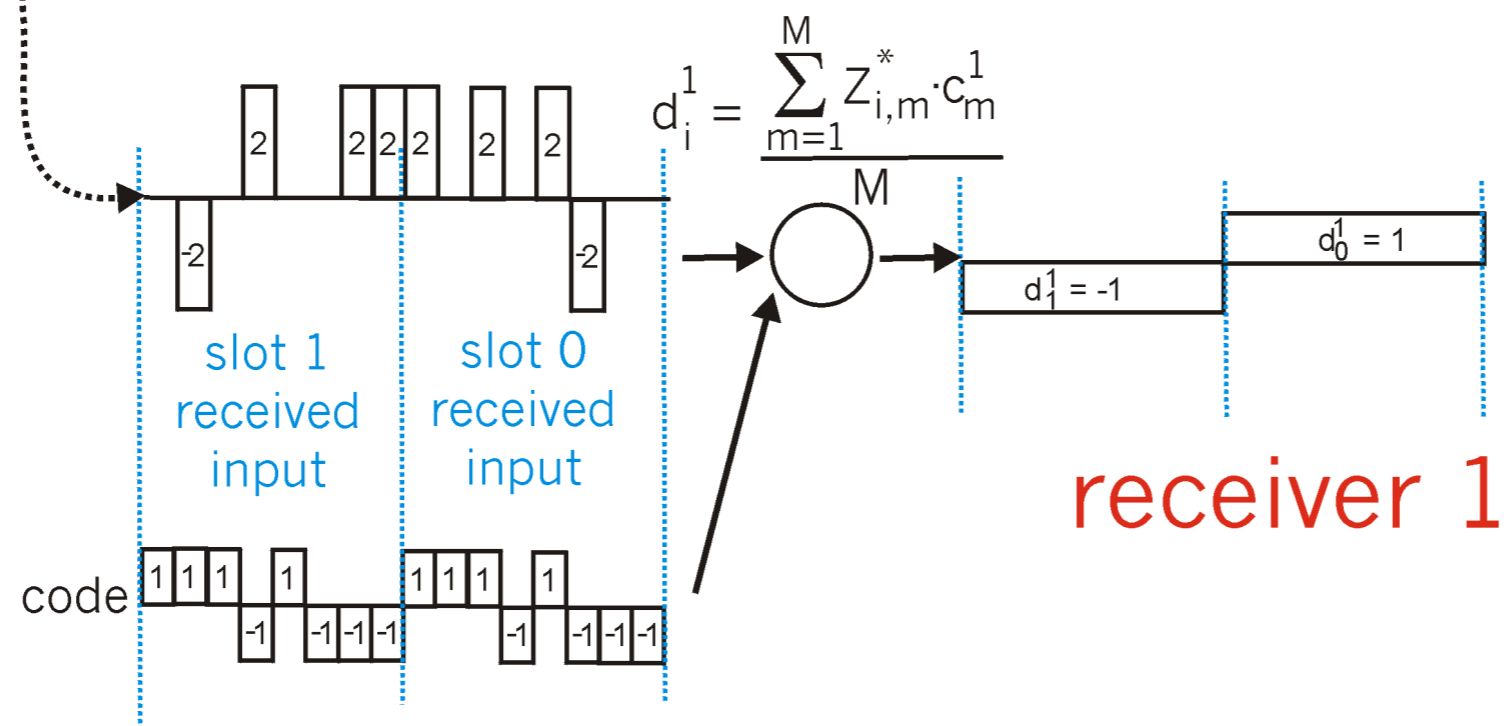
# CDMA: two-sender interference

Each sender adds a *unique* code

senders



Sender one removes its *unique* code



# Coding Examples summary

- Common Wired coding
  - Block codecs: table-lookups
    - fixed overhead, inline control signals
  - Scramblers: shift registers
    - overhead free

Like earlier coding schemes and error correction/detection; you can combine these

- e.g, 10Gb/s Ethernet may use a hybrid

CDMA (Code Division Multiple Access)

- coping intelligently with competing sources
- Mobile phones

# Error Detection and Correction

Transmission media are not perfect and cause signal impairments:

## 1. Attenuation

- Loss of energy to overcome medium's resistance

## 2. Distortion

- The signal changes its form or shape, caused in composite signals

## 3. Noise

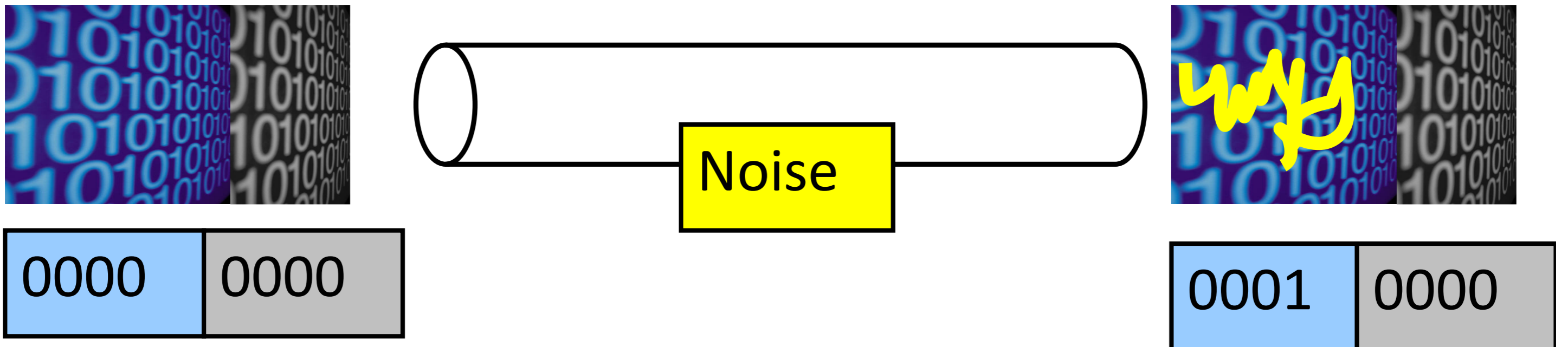
- Thermal noise, induced noise, crosstalk, impulse noise

Interference can change the shape or timing of a signal:

$0 \rightarrow 1$  or  $1 \rightarrow 0$

# Error Detection and Correction

How to use coding to deal with errors in data communication?



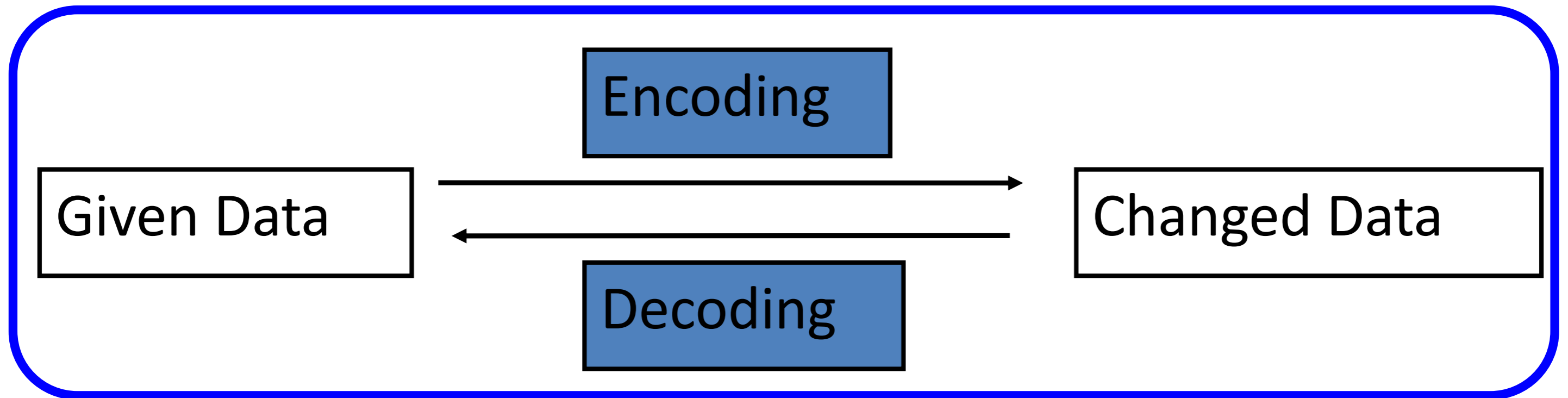
Basic Idea :

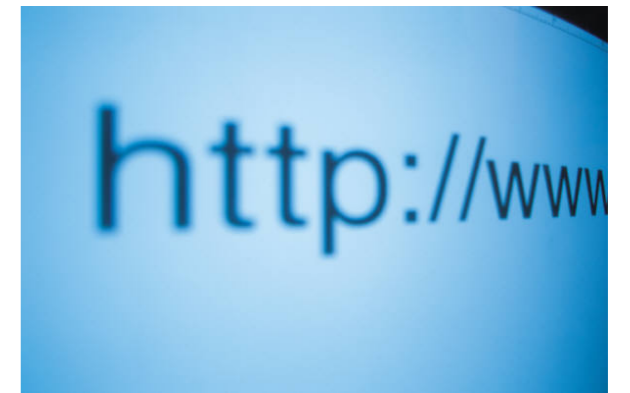
1. Add additional information (redundancy) to a message.
2. Detect an error and discard

Or, fix an error in the received message.

# Coding – a channel function

Change the representation of data.





MyPasswd



AA\$\$\$\$ff



AA\$\$\$\$ffff



MyPasswd



AA\$\$\$\$ff

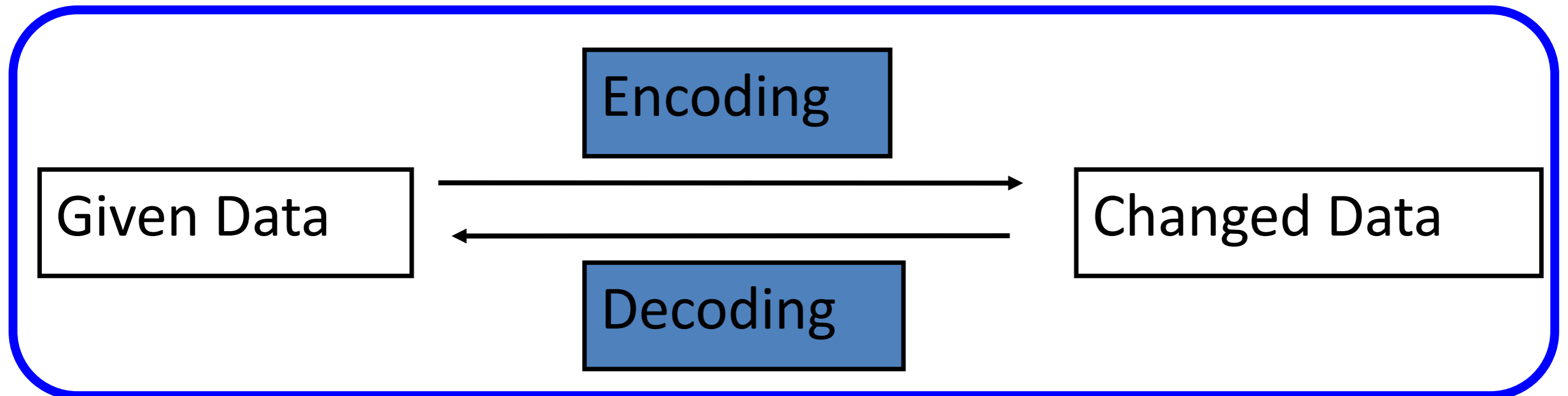


AA\$\$\$\$ffff



# Coding Examples

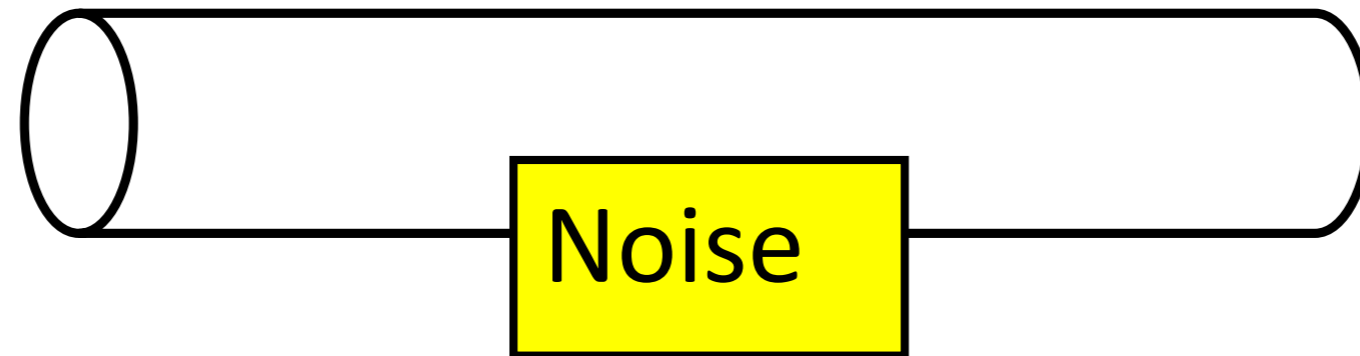
Changing the representation of data.



1. **Encryption:** MyPasswd  $\leftrightarrow$  AA\$\$\$\$ff
2. **Error Detection:** AA\$\$\$\$ff  $\leftrightarrow$  AA\$\$\$\$ffff
3. **Compression:** AA\$\$\$\$ffff  $\leftrightarrow$  A2\$4f4
4. **Analog:** A2\$4f4  $\leftrightarrow$

# Error Detection Code: Parity

Add one bit, such that the number of all 1's is even.



0000	0
------	---

0001	1
------	---

1001	0
------	---

X

0001	0
------	---

✓

0001	1
------	---

✓

1111	0
------	---

Problem: This simple parity cannot detect two-bit errors.



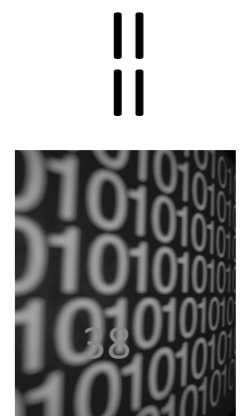
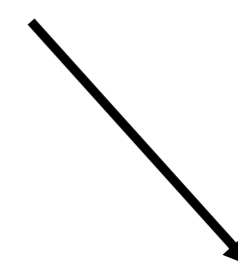
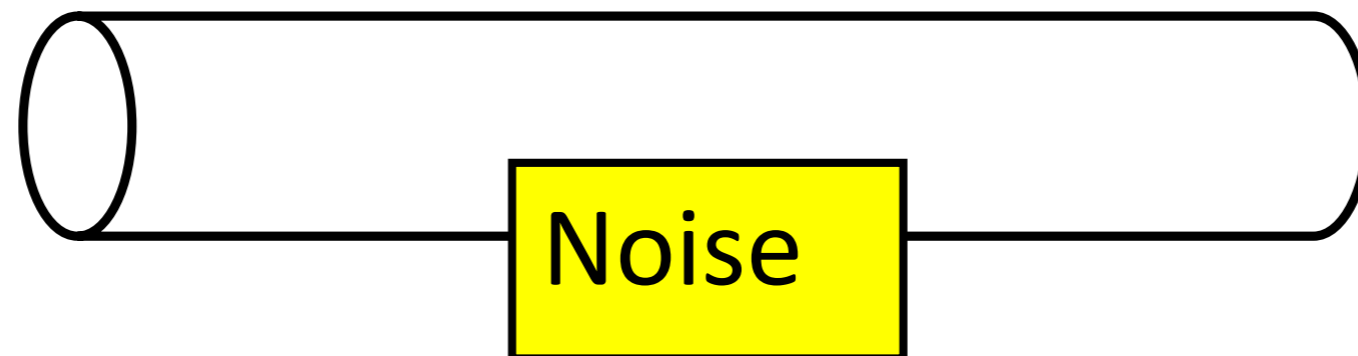
# Error Detection Code

Sender:

```
Y = generateCheckBit(X);  
send(XY);
```

Receiver:

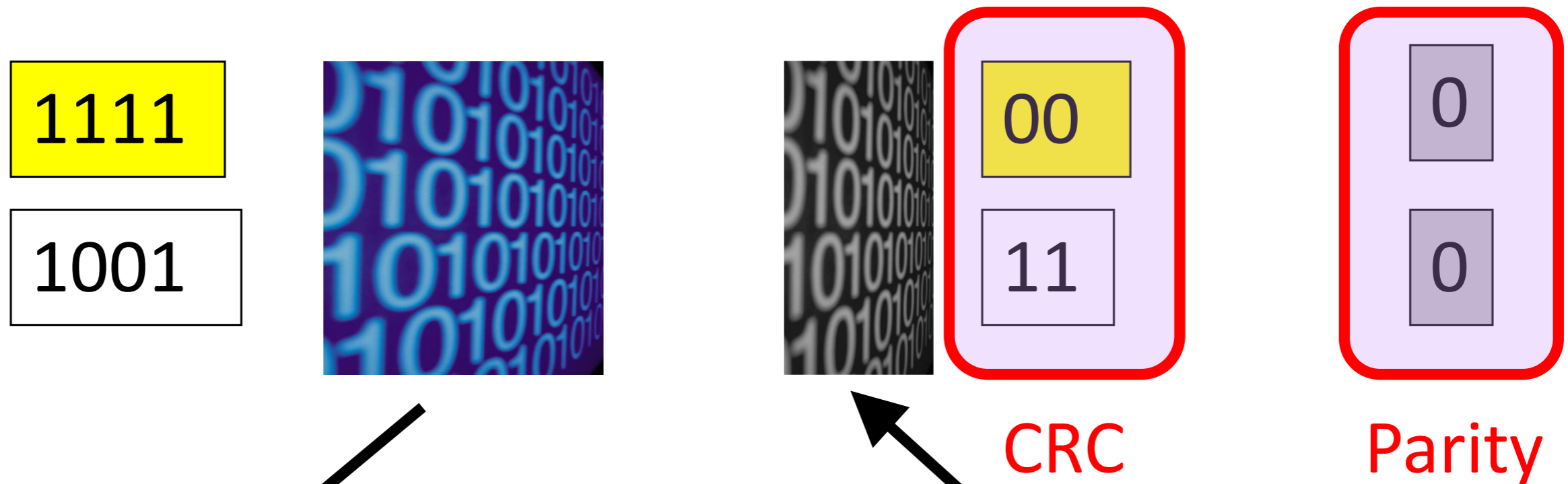
```
receive(X1Y1);  
Y2=generateCheckBit(X1);  
if (Y1 != Y2) ERROR;  
else NOERROR
```



# Error Detection Code: CRC

- CRC means “Cyclic Redundancy Check”.
- *“A sequence of redundant bits, called CRC, is appended to the end of data so that the resulting data becomes exactly divisible by a second, predetermined binary number.”*
- *CRC := remainder (data  $\div$  predetermined divisor)*
- More powerful than parity.
  - It can detect various kinds of errors, including 2-bit errors.
- More complex: multiplication, binary division.
- Parameterized by n-bit divisor P.
  - Example: 3-bit divisor 101.
  - Choosing good P is crucial.

# CRC with 3-bit Divisor 101



1111 same check bits from Parity,  
1001 but different ones from CRC

Multiplication by  $2^3$   
 $D2 = D * 2^3$

Binary Division by 101  
CheckBit = (D2) rem (101)

Add three 0's at the end

Kurose p478 §5.2.3  
Peterson [URL](#) §2.4

# Error Detection Code

Sender:

```
Y = generateCRC(X div P);
```

```
send(X);
```

```
send(Y);
```

Receiver:

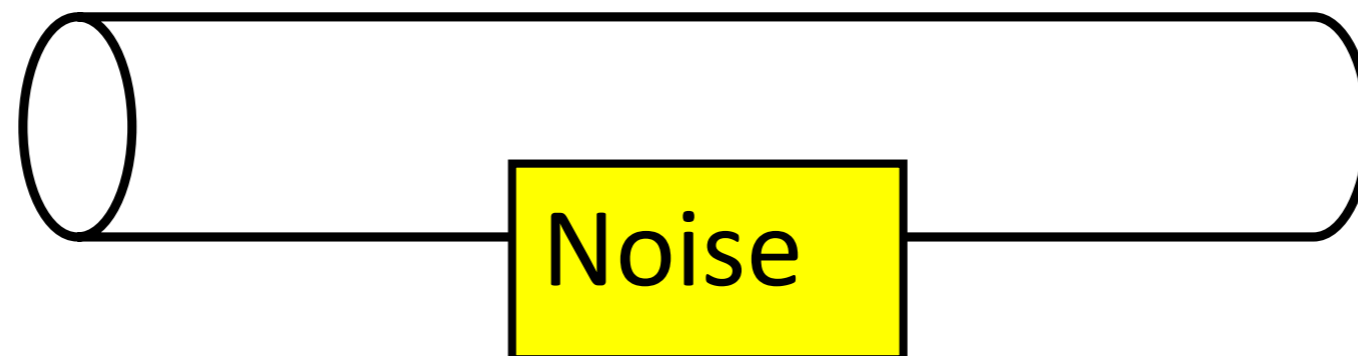
```
receive(X1);
```

```
receive(Y1);
```

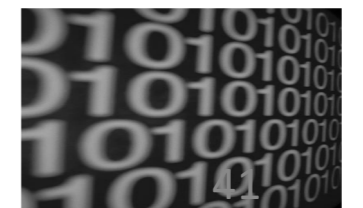
```
Y2=generateCRC(X1Y1 div P);
```

```
if (Y2 != 0s) ERROR;
```

```
else NOERROR
```



0s ==



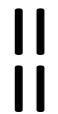
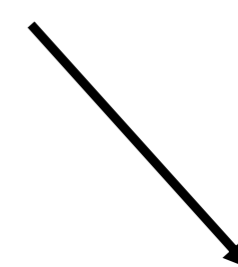
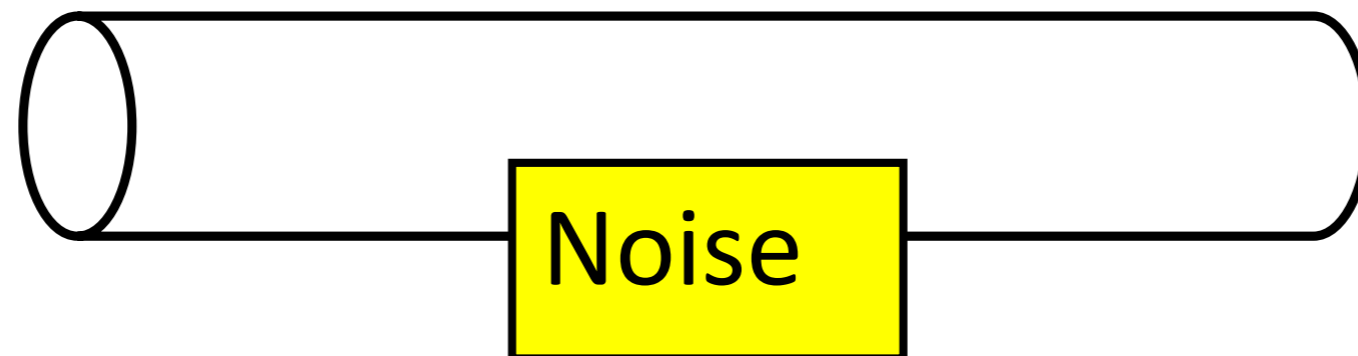
# Transforming Error Detection to...

Sender:

```
Y = generateCheckBit(X);  
send(XY);
```

Receiver:

```
receive(X1Y1);  
Y2=generateCheckBit(X1);  
if (Y1 != Y2) ERROR;  
else NOERROR
```



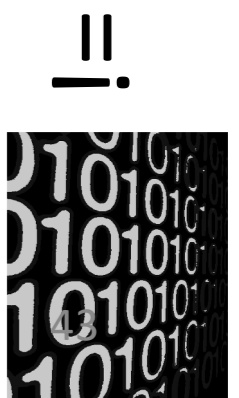
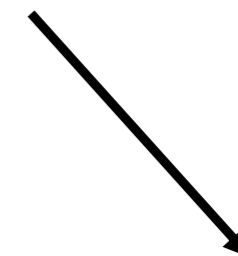
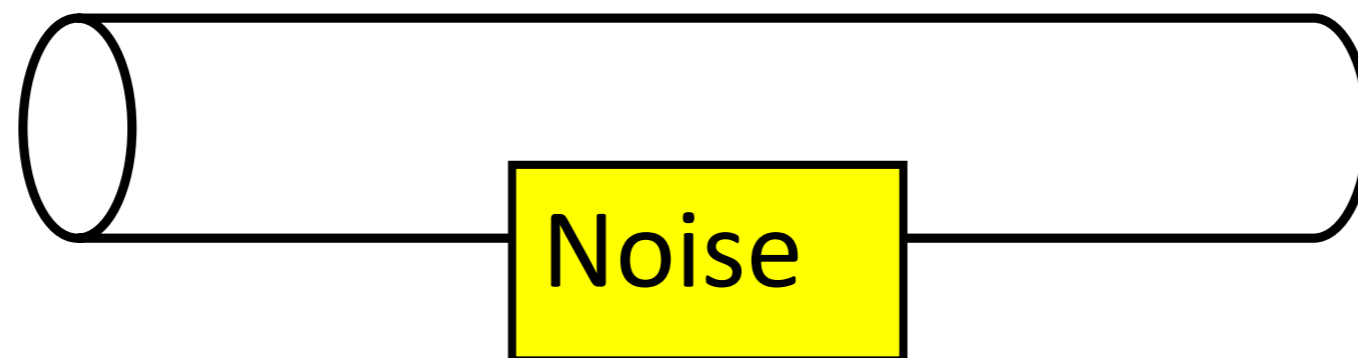
# Forward Error Correction (FEC)

Sender:

```
Y = generateCheckBit(X);  
send(XY);
```

Receiver:

```
receive(X1Y1);  
Y2=generateCheckBit(X1);  
if (Y1 != Y2) FIXERROR(X1Y1);  
else NOERROR
```





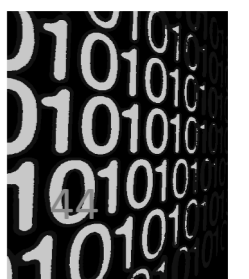
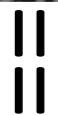
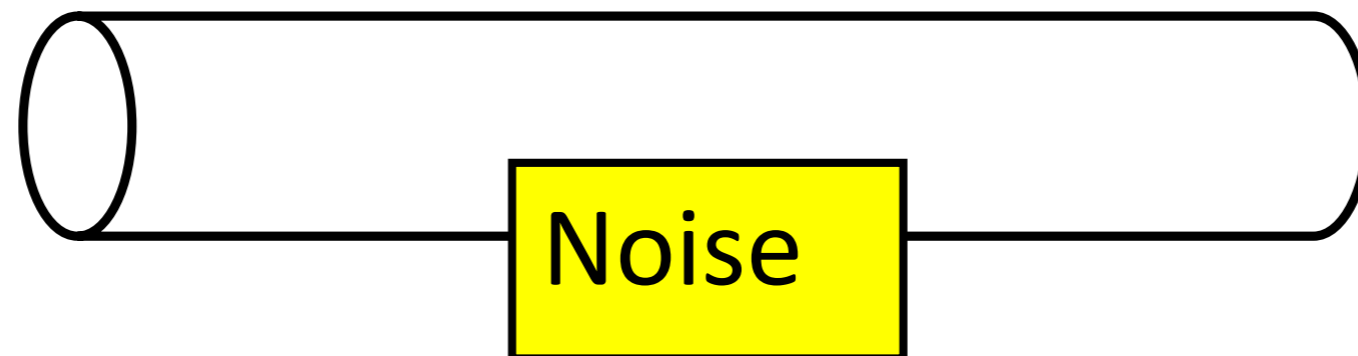
# Forward Error Correction (FEC)

Sender:

```
Y = generateCheckBit(X);  
send(XY);
```

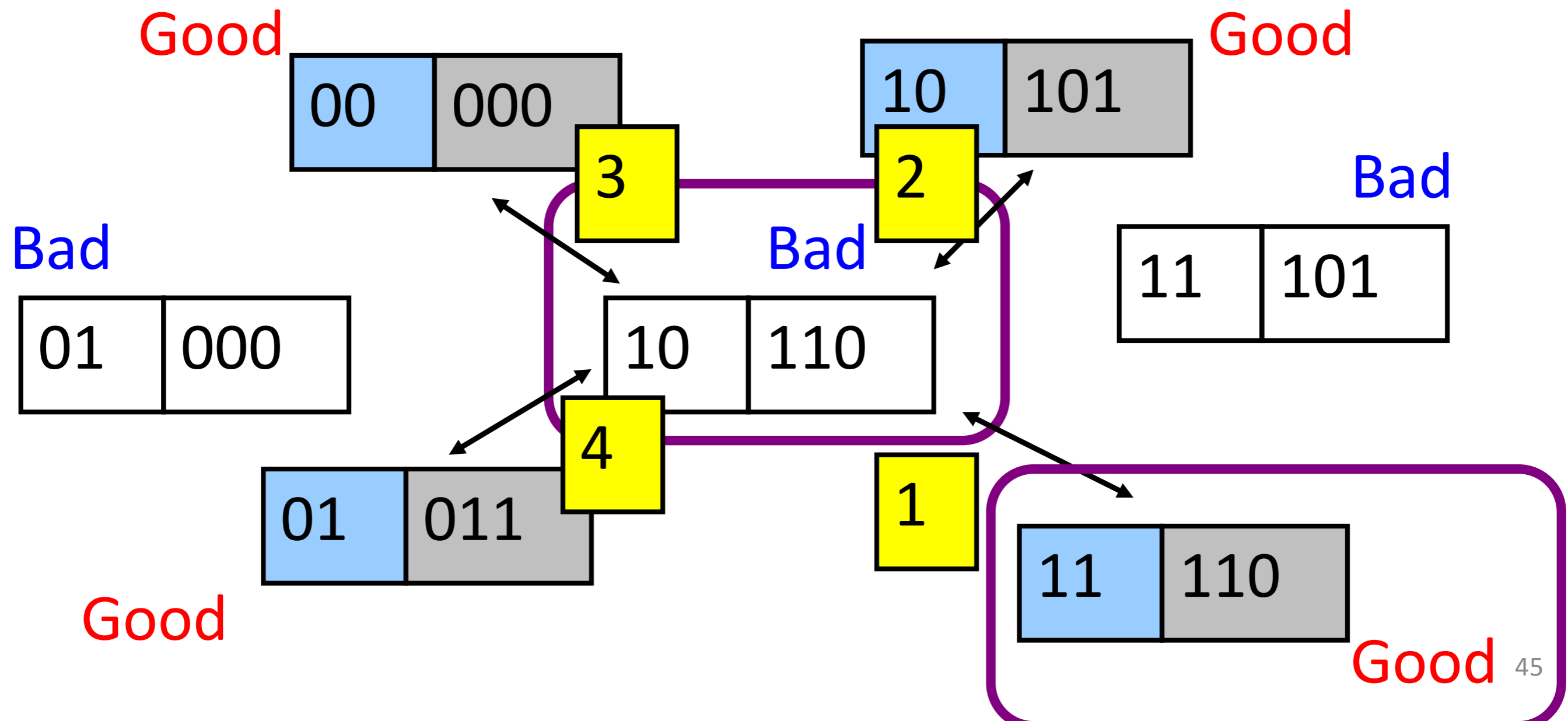
Receiver:

```
receive(X1Y1);  
Y2=generateCheckBit(X1);  
if (Y1 != Y2) FIXERROR(X1Y1);  
else NOERROR
```



# Basic Idea of Forward Error Correction

Replace erroneous data by its “closest” error-free data.





# Error Detection vs Correction

## Error Correction:

- Cons: More check bits. False recovery.
- Pros: No need to re-send.

## Error Detection:

- Cons: Need to re-send.
- Pros: Less check bits.

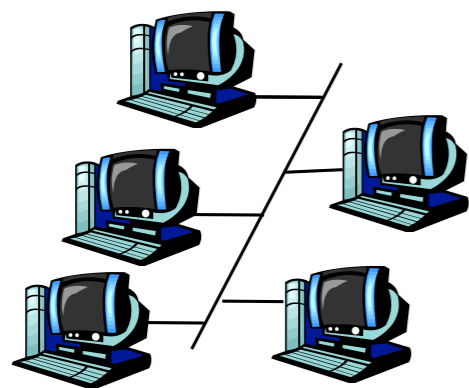
## Usage:

- Correction: A lot of noise. Expensive to re-send.
- Detection: Less noise. Easy to re-send.
- Can be used together.

# Multiple Access Links and Protocols

## Two types of “links”:

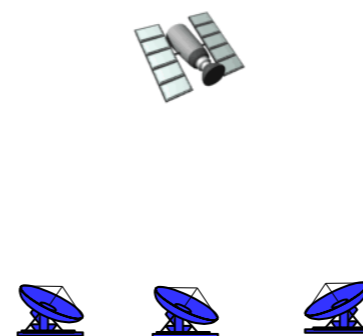
- point-to-point
  - point-to-point link between Ethernet switch and host
- **broadcast** (shared wire or medium)
  - old-fashioned wired Ethernet (*here be dinosaurs* – extinct)
  - upstream HFC (Hybrid Fiber-Coax – the Coax may be broadcast)
  - Home plug / Powerline networking
  - 802.11 wireless LAN



shared wire (e.g.,  
Coax cabled Ethernet)



shared RF  
(e.g., 802.11 WiFi)



shared RF  
(satellite)



humans at a  
cocktail party  
(shared air, acoustical)

# Multiple Access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes:  
interference
  - **collision** if node receives two or more signals at the same time

## multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

# Ideal Multiple Access Protocol

## Broadcast channel of rate $R$ bps

1. when one node wants to transmit, it can send at rate  $R$
2. when  $M$  nodes want to transmit,  
each can send at average rate  $R/M$
3. fully decentralized:
  - no special node to coordinate transmissions
  - no synchronization of clocks, slots
4. simple

# MAC Protocols: a taxonomy

Three broad classes:

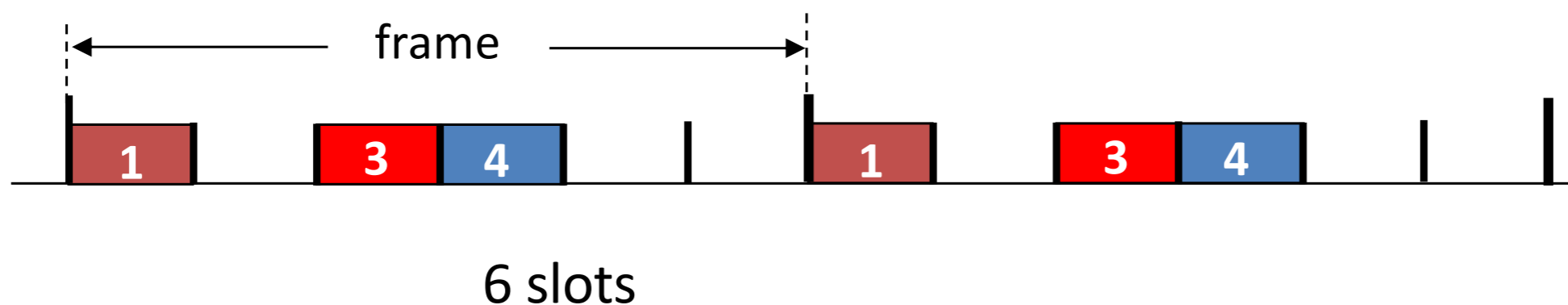
- **Channel Partitioning**
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- **Random Access**
  - channel not divided, allow collisions
  - “recover” from collisions
- **“Taking turns”**
  - nodes take turns, but nodes with more to send can take longer turns

# Channel Partitioning MAC protocols: TDMA

*(we discussed this earlier)*

## TDMA: time division multiple access

- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: station LAN, 1,3,4 have pkt, slots 2,5,6 idle

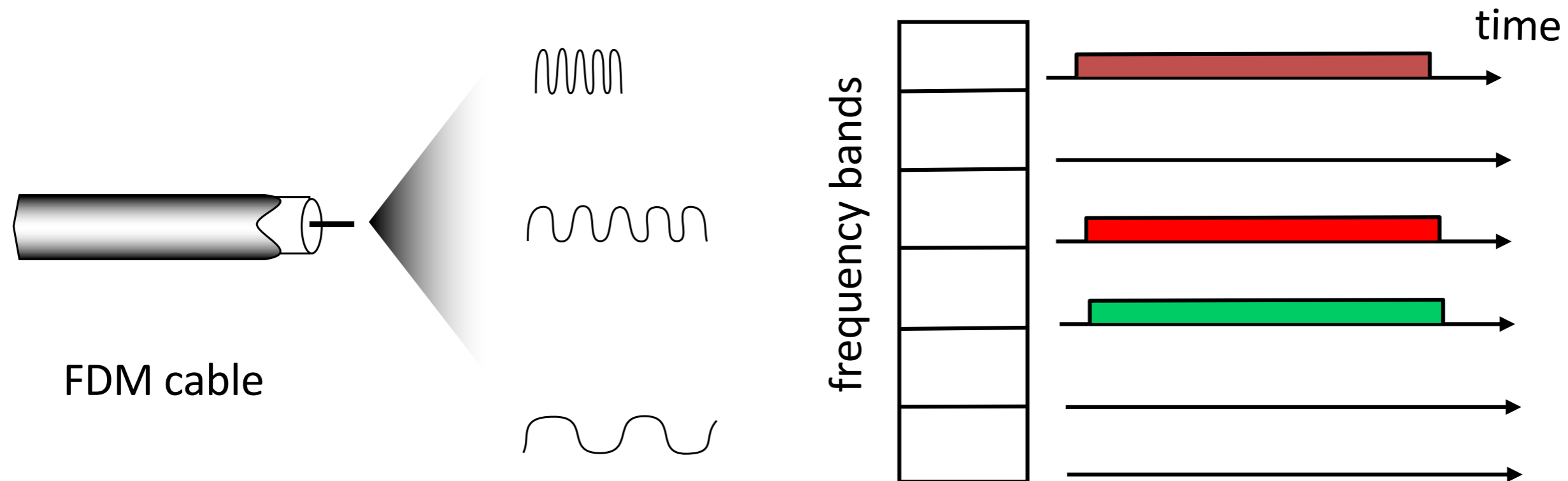


# Channel Partitioning MAC protocols: FDMA

*(we discussed this earlier)*

## FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



# “Taking Turns” MAC protocols

## channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access,  $1/N$  bandwidth allocated even if only 1 active node!

## random access MAC protocols:

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

## “taking turns” protocols:

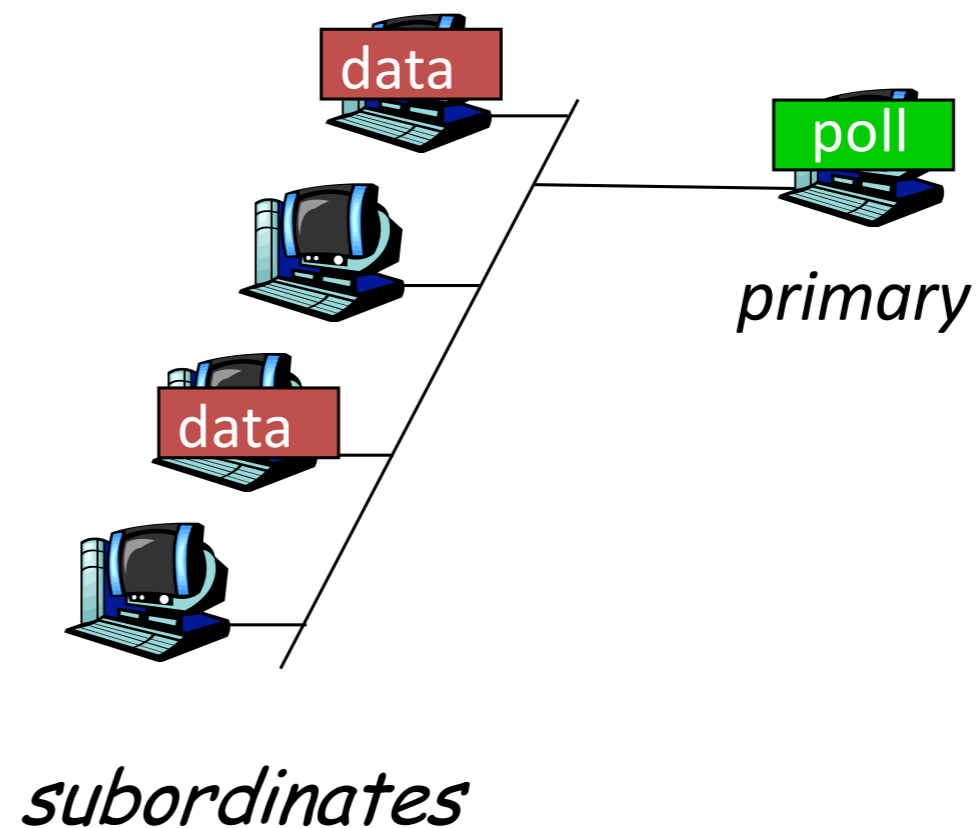
look for best of both worlds!



# “Taking Turns” MAC protocols

## Polling:

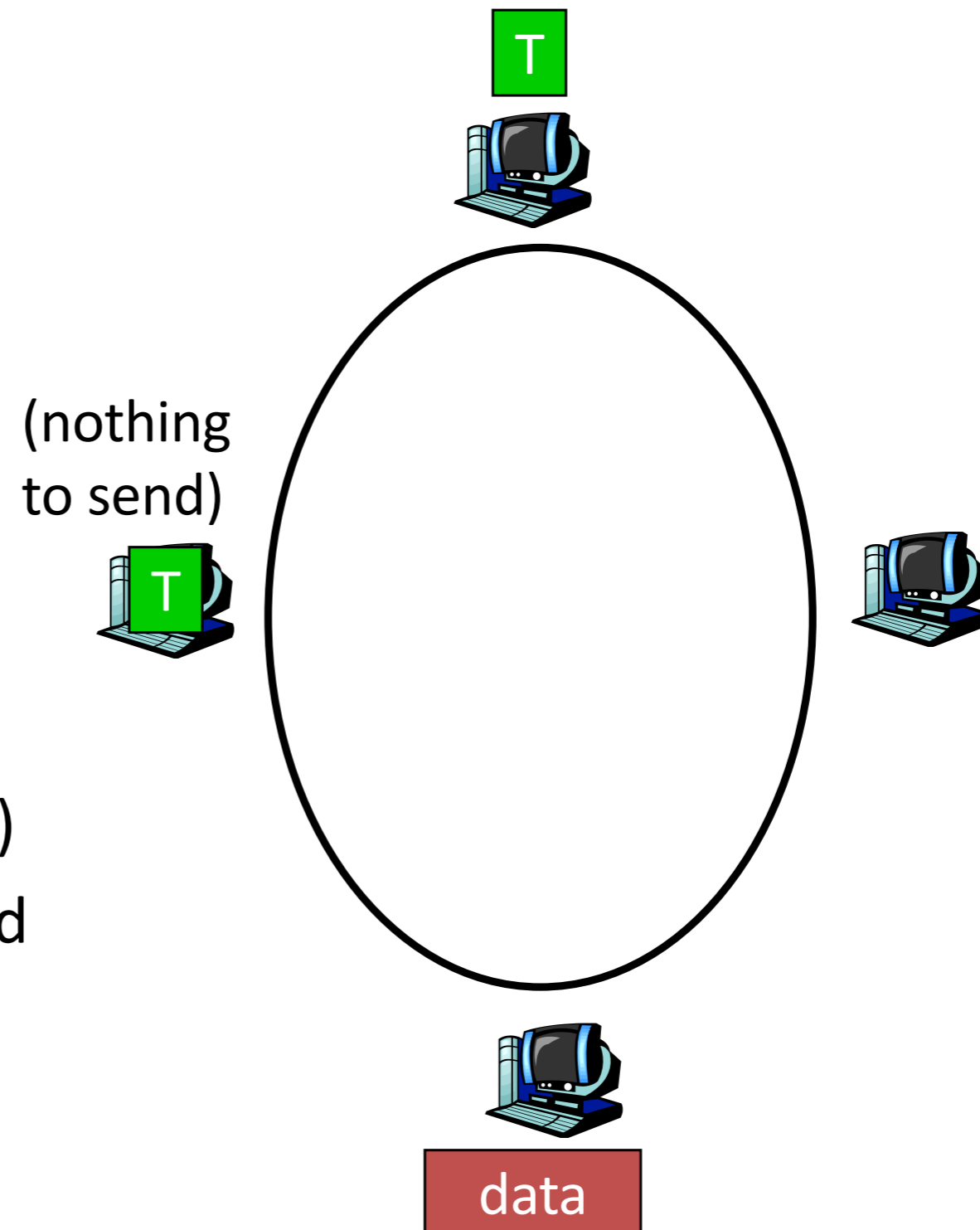
- Primary node “invites” subordinates nodes to transmit in turn
- typically used with simpler subordinate devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (primary)



# “Taking Turns” MAC protocols

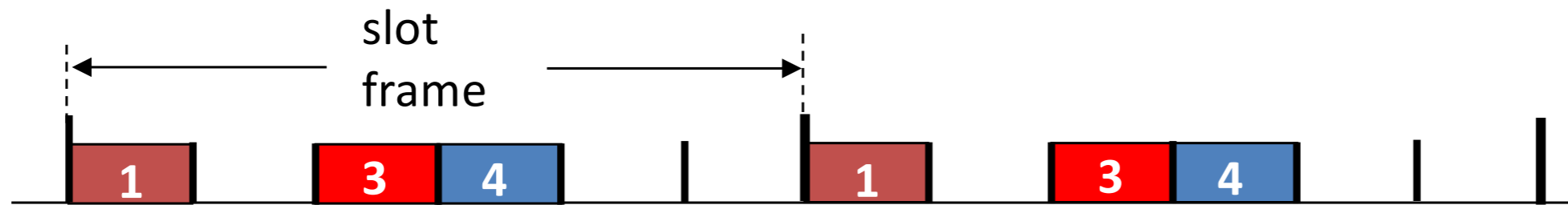
## Token passing:

- r control **token** passed from one node to next sequentially.
- r token message
- r concerns:
  - m token overhead
  - m latency
  - m single point of failure (token)
- m concerns fixed in part by a slotted ring (many simultaneous *tokens*)

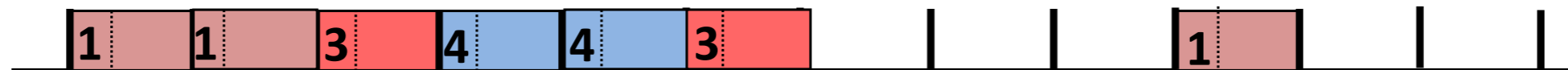


# ATM

In TDM a sender may only use a pre-allocated slot



In ATM a sender transmits labeled cells whenever necessary



ATM = Asynchronous Transfer Mode – an ugly expression  
think of it as ATDM – Asynchronous Time Division Multiplexing

That's a variant of **PACKET SWITCHING** to the rest of us – just like Ethernet  
but using fixed length slots/packets/cells

Use the media when you need it, but  
ATM had virtual circuits and these needed setup....

# Random Access MAC Protocols

- When node has packet to send
  - Transmit at full channel data rate
  - No *a priori* coordination among nodes
- Two or more transmitting nodes  $\Rightarrow$  collision
  - Data lost
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions
- Examples
  - ALOHA and Slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA (wireless)

# Key Ideas of Random Access

- **Carrier sense**
  - *Listen before speaking, and don't interrupt*
  - Checking if someone else is already sending data
  - ... and waiting till the other node is done
- **Collision detection**
  - *If someone else starts talking at the same time, stop*
  - Realizing when two nodes are transmitting at once
  - ...by detecting that the data on the wire is garbled
- **Randomness**
  - *Don't start talking again right away*
  - Waiting for a random time before trying again

# CSMA (Carrier Sense Multiple Access)

- CSMA: **listen** before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy, defer transmission
- Human analogy: don't interrupt others!
- Does this eliminate all collisions?
  - No, because of nonzero propagation delay

# CSMA Collisions

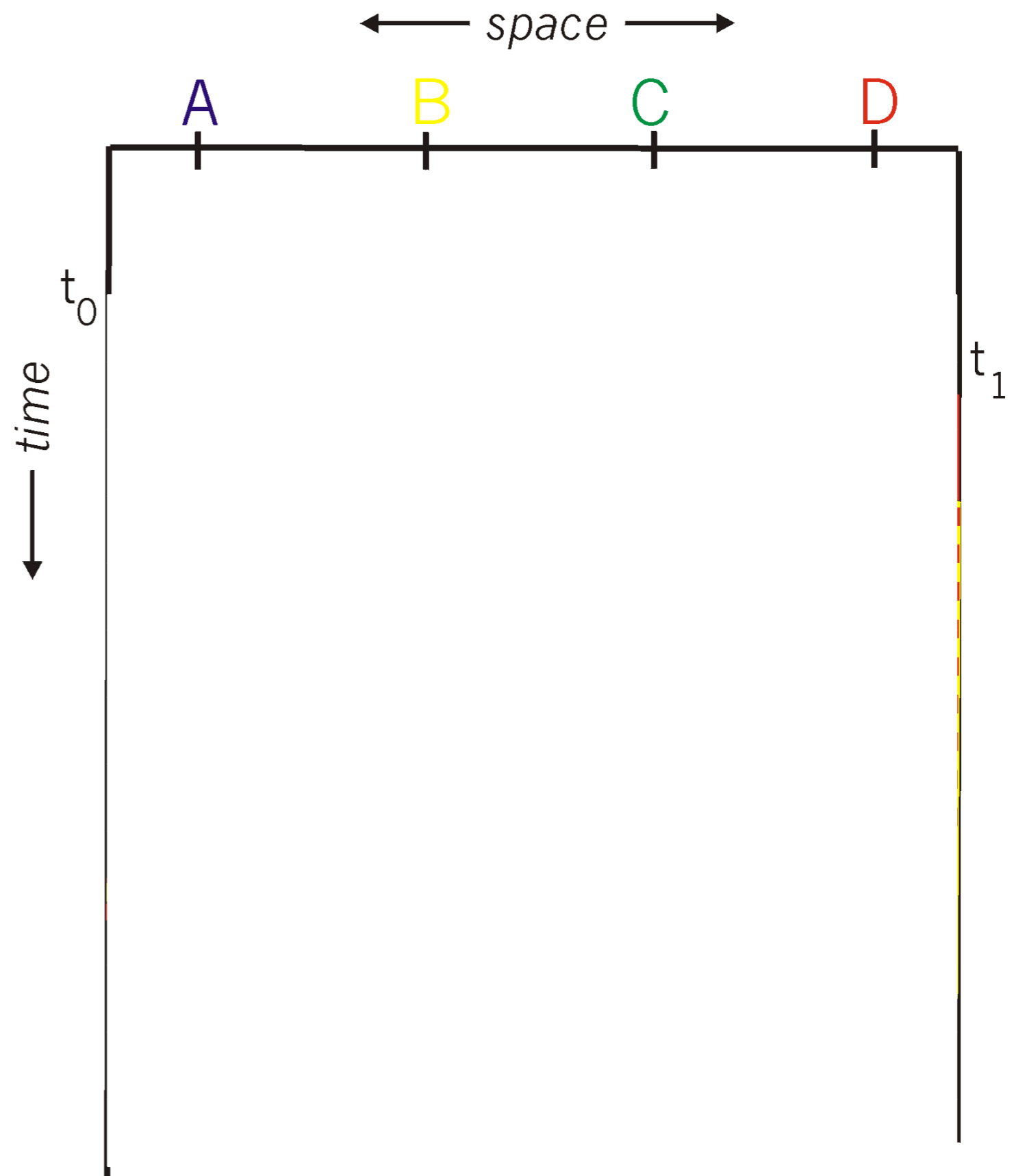
Propagation delay: two nodes may not hear each other's before sending.

*Would slots hurt or help?*

CSMA reduces but does not eliminate collisions

*Biggest remaining problem?*

Collisions still take full slot!  
How do you fix that?



# CSMA/CD (Collision Detection)

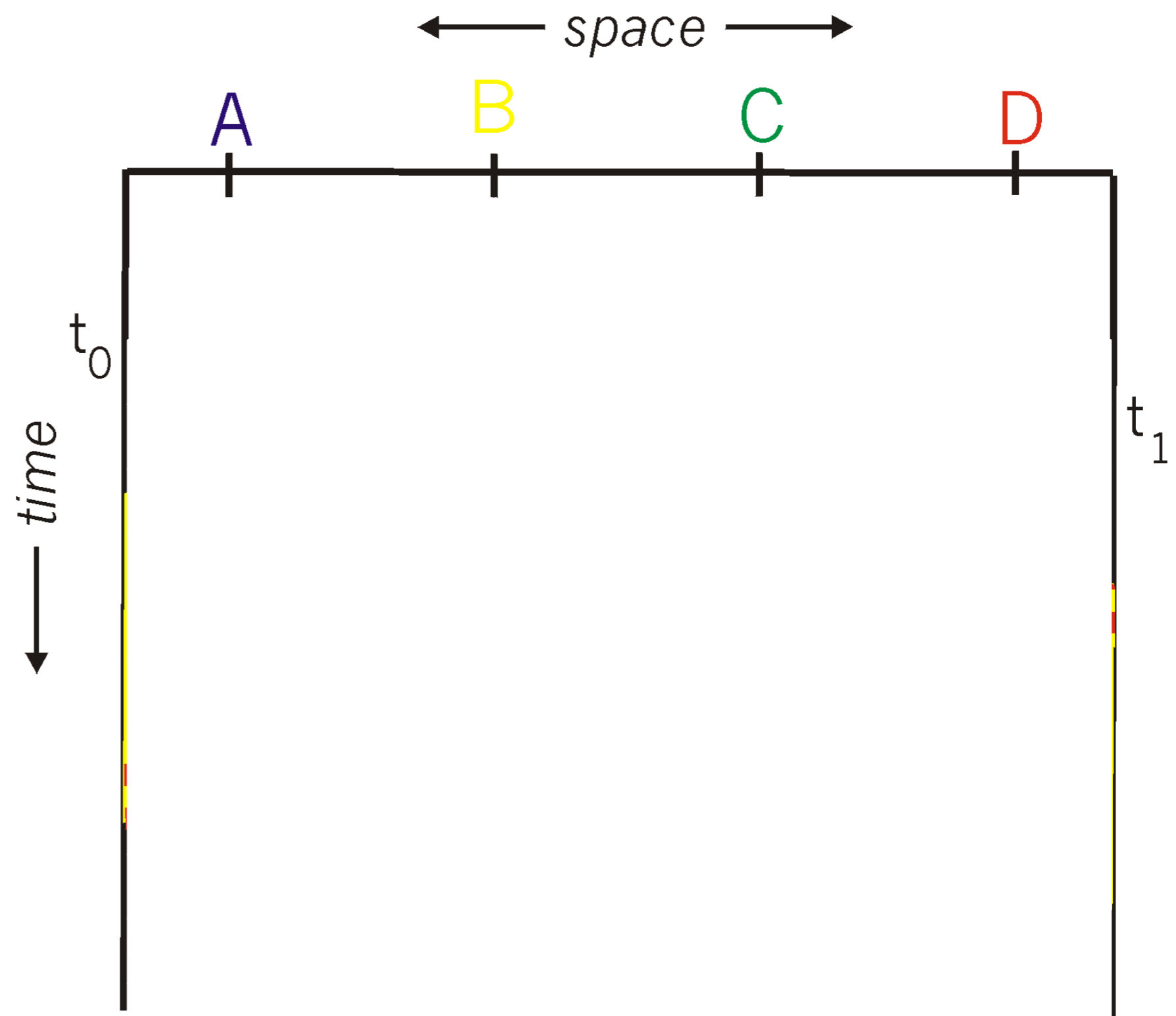
- CSMA/CD: carrier sensing, deferral as in CSMA
  - **Collisions detected within short time**
  - Colliding transmissions aborted, reducing wastage
- Collision detection easy in wired LANs:
  - Compare transmitted, received signals
- Collision detection difficult in wireless LANs:
  - Reception shut off while transmitting (well, perhaps not)
  - Not perfect broadcast (limited range) so collisions local
  - Leads to use of *collision avoidance* instead (later)



# CSMA/CD Collision Detection

B and D can tell that collision occurred.

Note: for this to work, need restrictions on minimum frame size and maximum distance. Why?



# Limits on CSMA/CD Network

## Length



- Latency depends on physical length of link
  - Time to propagate a packet from one end to the other
- Suppose *A* sends a packet at time  $t$ 
  - And *B* sees an idle line at a time just before  $t+d$
  - ... so *B* happily starts transmitting a packet
- *B* detects a collision, and sends **jamming signal**
  - But *A* can't see collision until  $t+2d$

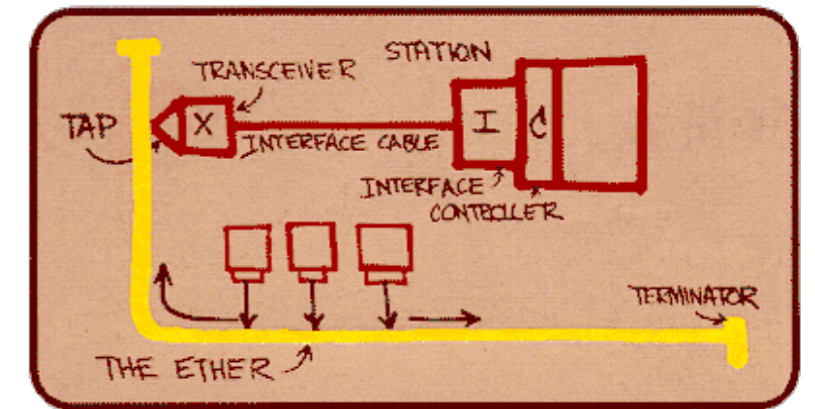
# Performance of CSMA/CD

- Time wasted in collisions
  - Proportional to distance  $d$
- Time spend transmitting a packet
  - Packet length  $p$  divided by bandwidth  $b$
- Rough estimate for efficiency (K some constant)

$$E \sim \frac{\frac{p}{b}}{\frac{p}{b} + Kd}$$

- Note:
  - For large packets, small distances,  $E \sim 1$
  - As bandwidth increases,  $E$  decreases
  - That is why high-speed LANs are all switched aka packets are sent via a switch - (any  $d$  is bad)

# Ethernet: CSMA/CD Protocol



- **Carrier sense:** wait for link to be idle
- **Collision detection:** listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission & send **jam** signal
- **Random access:** **binary exponential back-off**
  - After collision, wait a random time before trying again
  - After  $m^{\text{th}}$  collision, choose  $K$  randomly from  $\{0, \dots, 2^m - 1\}$
  - ... and wait for  $K * 512$  bit times before trying again
    - Using min packet size as “slot”
    - **If transmission occurring when ready to send, wait until end of transmission (CSMA)**

# Benefits of Ethernet

- Easy to administer and maintain
- Inexpensive
- Increasingly higher speed
- Evolvable!

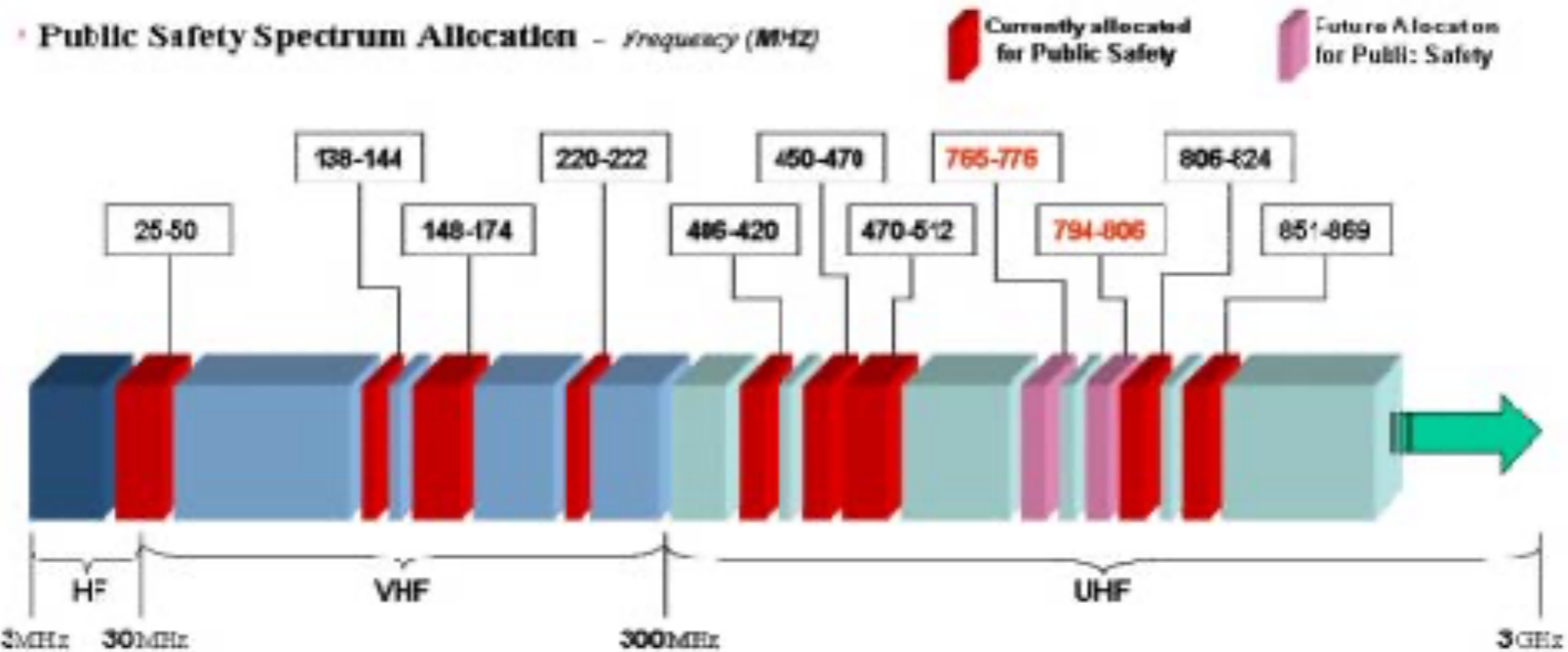
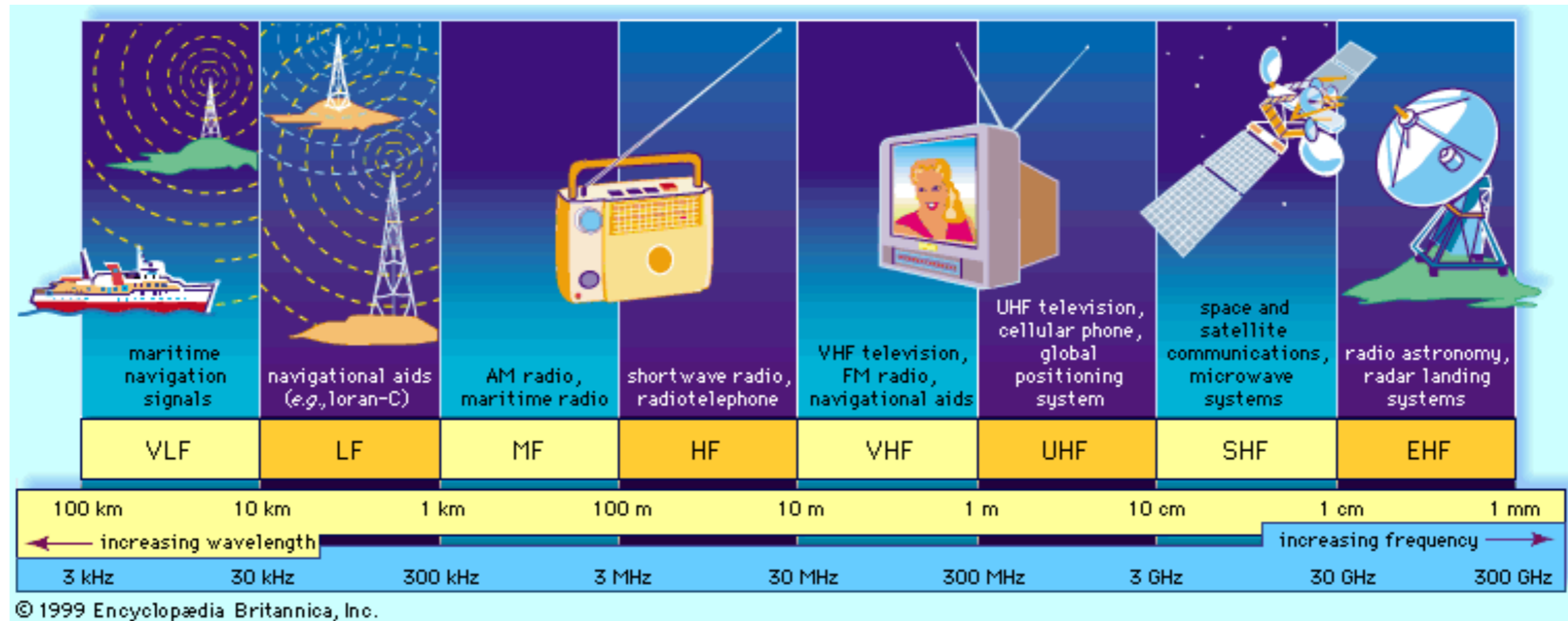
# Evolution of Ethernet

- Changed **everything** except the frame **format**
  - From single coaxial cable to hub-based star
  - From shared media to **switches**
  - From electrical signaling to optical
- **Lesson #1**
  - The right **interface** can accommodate many **changes**
  - Implementation is hidden behind interface
- **Lesson #2**
  - Really hard to displace the dominant technology
  - Slight performance improvements are not enough





# The Wireless Spectrum





# Metrics for evaluation / comparison of wireless technologies

- Bitrate or Bandwidth
- Range - PAN, LAN, MAN, WAN
- Two-way / One-way
- Multi-Access / Point-to-Point
- Digital / Analog
- Applications and industries
- Frequency – Affects most physical properties:
  - Distance (free-space loss)
  - Penetration, Reflection, Absorption
  - Energy proportionality
  - Policy: Licensed / Deregulated
  - Line of Sight (Fresnel zone)
  - Size of antenna
- Determined by wavelength –  $\lambda = \frac{v}{f}$ .)

# Wireless Communication Standards

- Cellular (**800/900/1700/1800/1900Mhz**):
  - 2G: GSM / CDMA / GPRS /EDGE
  - 3G: CDMA2000/UMTS/HSDPA/EVDO
  - 4G: LTE, WiMax
- IEEE 802.11 (aka WiFi): (some examples)
  - b: **2.4Ghz** band, 11Mbps (*~4.5 Mbps operating rate*)
  - g: **2.4Ghz**, 54-108Mbps (*~19 Mbps operating rate*)
  - a: **5.0Ghz** band, 54-108Mbps (*~25 Mbps operating rate*)
  - n: **2.4/5Ghz**, 150-600Mbps (4x4 mimo)
  - ac: **2.4/5Ghz**, 433-1300Mbps (improved coding 256-QAM)
  - ad: **60Ghz**, 7Gbps
  - af: **54/790Mhz**, 26-35Mbps (TV whitespace)
- IEEE 802.15 – lower power wireless:
  - 802.15.1: **2.4Ghz**, 2.1 Mbps (Bluetooth)
  - 802.15.4: **2.4Ghz**, 250 Kbps (Sensor Networks)

# What Makes Wireless Different?

- Broadcast and multi-access medium...
  - err, so....
- BUT, Signals sent by sender don't always end up at receiver intact
  - Complicated physics involved, which we won't discuss
  - But what can go wrong?

# Lets focus on 802.11

aka - WiFi ...

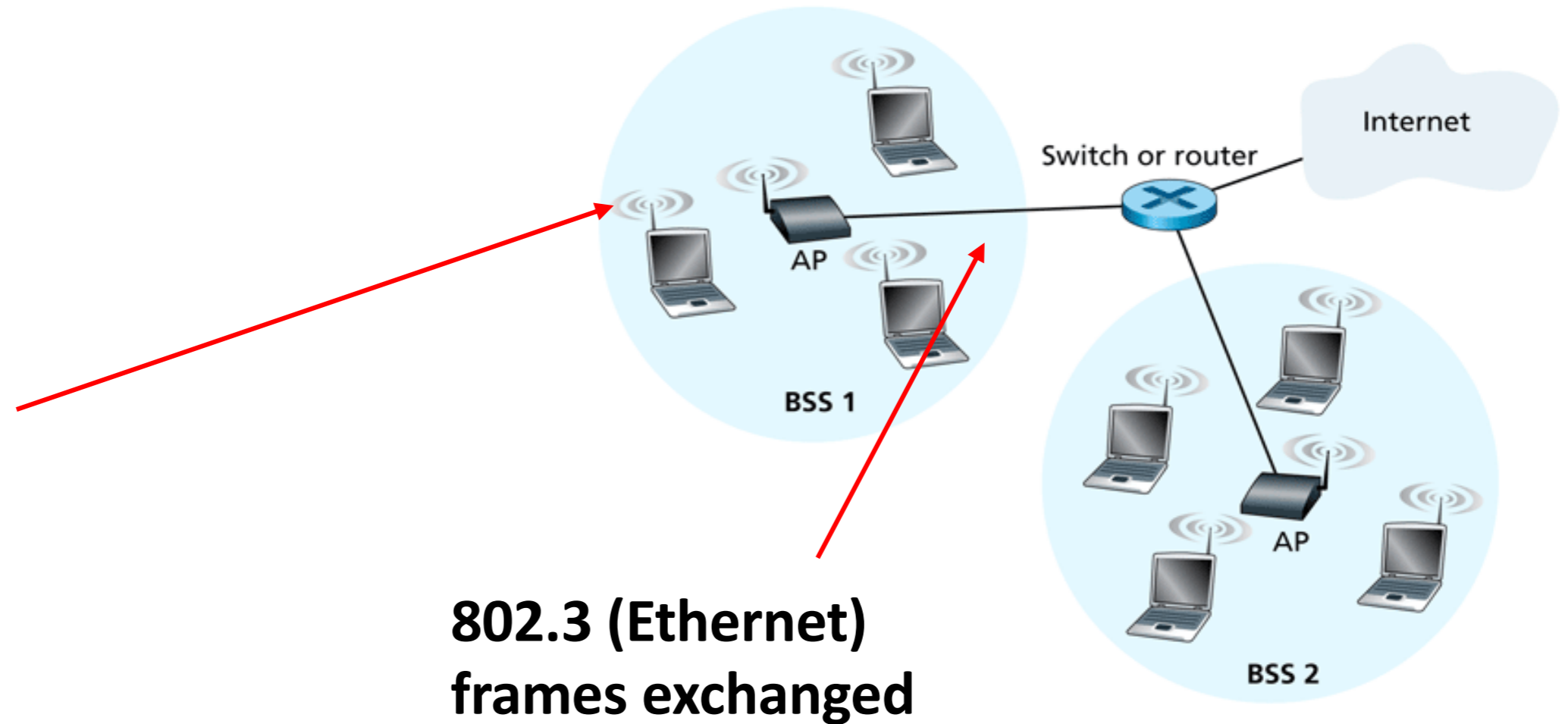
What makes it special?

**Deregulation** > Innovation > Adoption > Lower cost = Ubiquitous technology

JUST LIKE ETHERNET – not lovely but sufficient

# 802.11 Architecture

802.11 frames exchanges



802.3 (Ethernet) frames exchanged

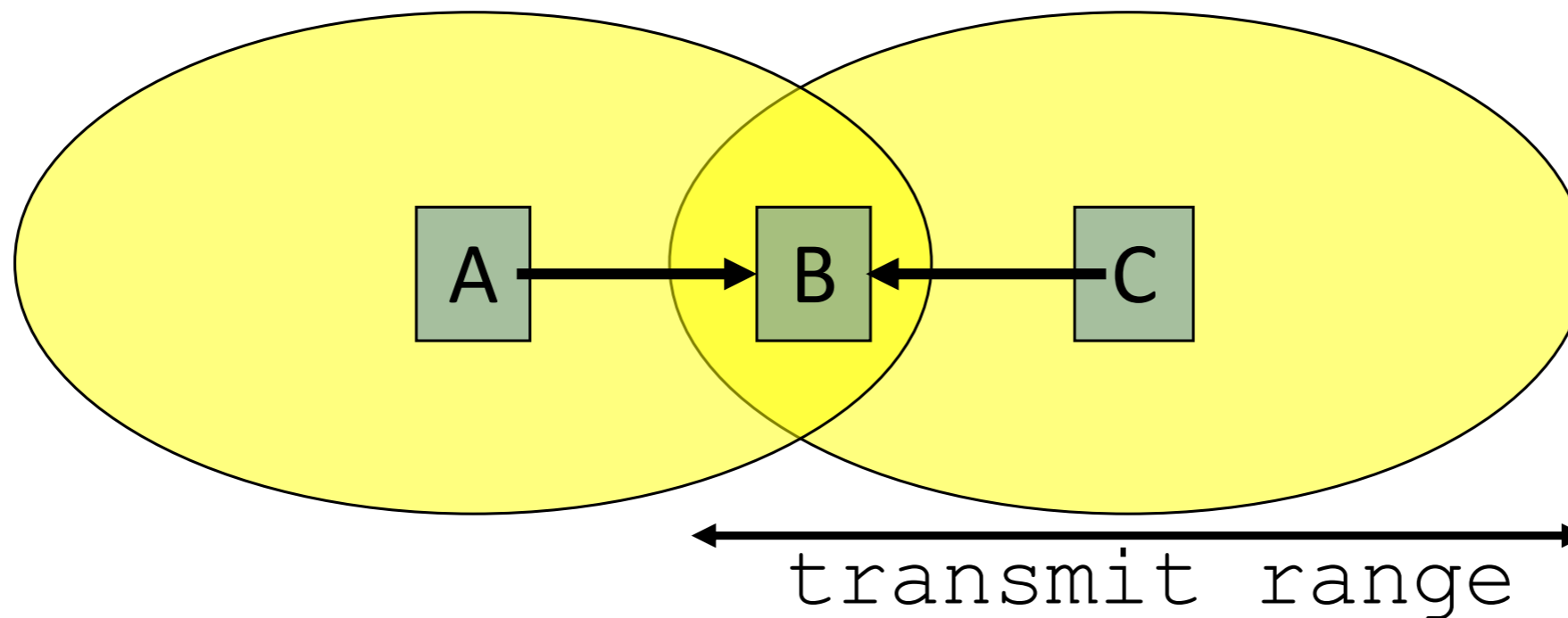
Figure 6.7 ♦ IEEE 802.11 LAN architecture

- Designed for limited area
- AP' s (Access Points) set to specific channel
- Broadcast beacon messages with SSID (Service Set Identifier) and MAC Address periodically
- Hosts scan all the channels to discover the AP' s
  - Host associates with AP

# Wireless Multiple Access Technique?

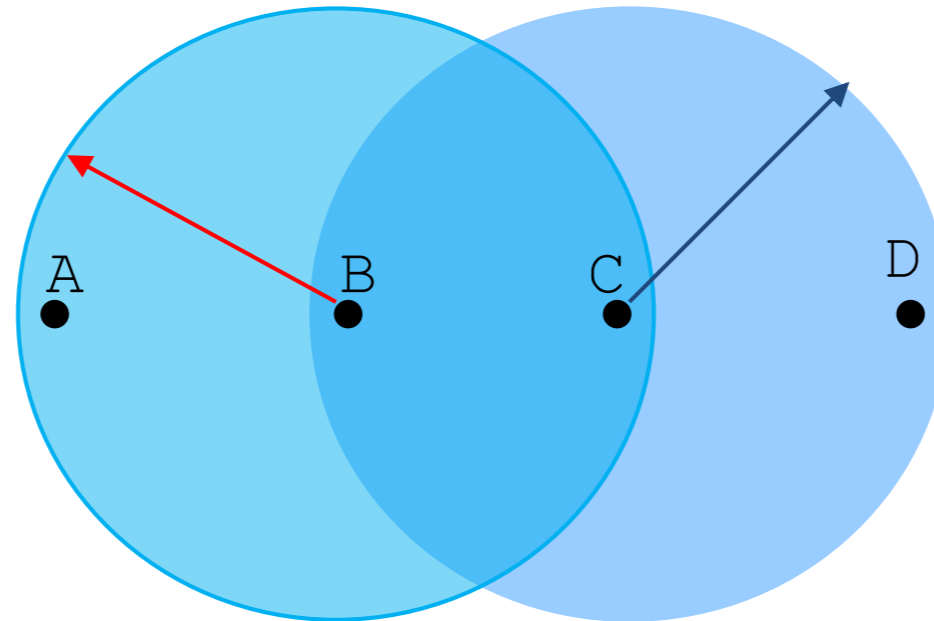
- Carrier Sense?
  - Sender can listen before sending
  - What does that tell the sender?
- Collision Detection?
  - Where do collisions occur?
  - How can you detect them?

# Hidden Terminals



- A and C can both send to B but **can't hear each other**
  - A is a *hidden terminal* for C and vice versa
- Carrier Sense will be **ineffective**

# Exposed Terminals



- **Exposed node:** B sends a packet to A; C hears this and decides not to send a packet to D (despite the fact that this will not cause interference)!
- Carrier sense would prevent a successful transmission.



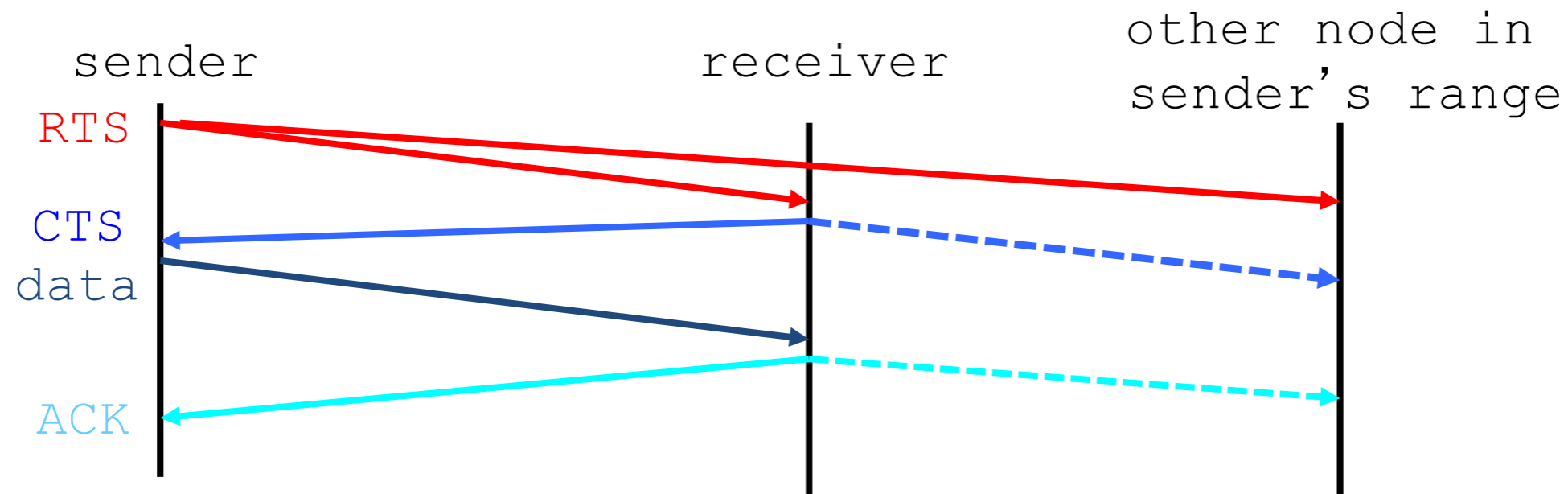
# Key Points

- No concept of a global collision
  - Different receivers hear different signals
  - Different senders reach different receivers
- Collisions are at receiver, not sender
  - Only care if receiver can hear the sender clearly
  - It does not matter if sender can hear someone else
  - As long as that signal does not interfere with receiver
- Goal of protocol:
  - Detect if receiver can hear sender
  - Tell senders who might interfere with receiver to shut up

# Basic Collision Avoidance

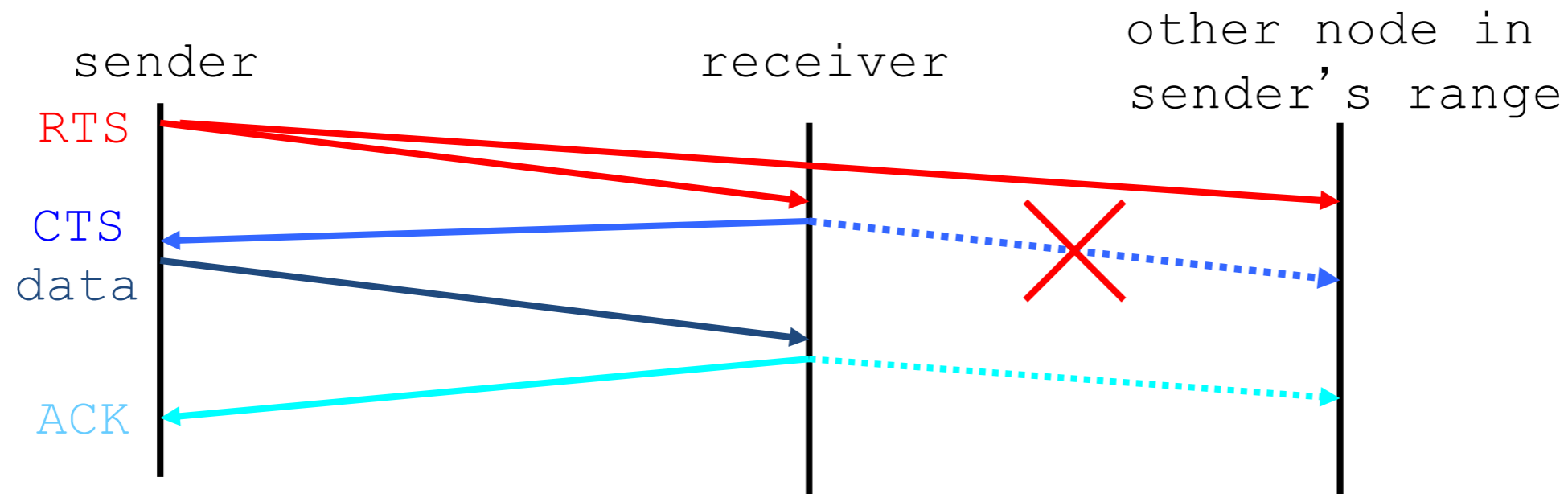
- Since can't detect collisions, we try to *avoid* them
- Carrier sense:
  - When medium busy, choose random interval
  - Wait that many **idle** timeslots to pass before sending
- When a collision is inferred, retransmit with binary exponential backoff (like Ethernet)
  - Use **ACK** from receiver to infer “no collision”
  - Use exponential backoff to adapt contention window

# CSMA/CA -MA with Collision Avoidance



- Before every data transmission
  - Sender sends a Request to Send (RTS) frame containing the length of the transmission
  - Receiver respond with a Clear to Send (CTS) frame
  - Sender sends data
  - Receiver sends an ACK; now another sender can send data
- When sender doesn't get a CTS back, it assumes collision

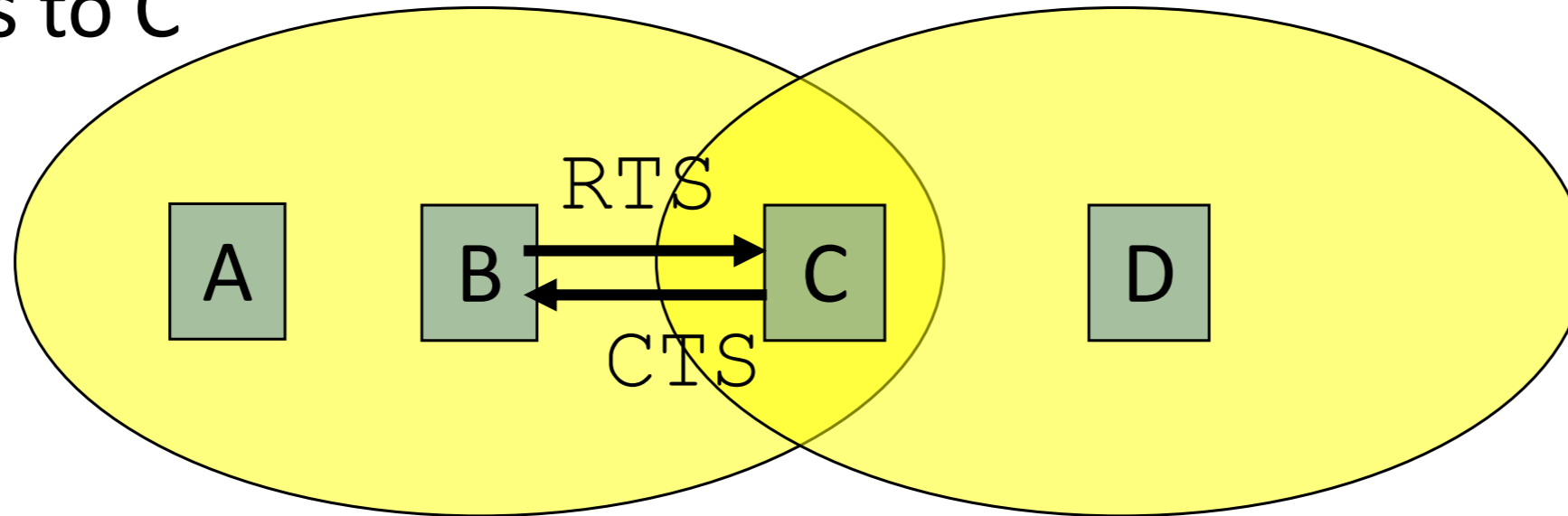
# CSMA/CA, con't



- If other nodes hear RTS, but not CTS: **send**
  - Presumably, destination for first sender is out of node's range ...
  - ... Can cause problems when a CTS is **lost**
- When you hear a CTS, you keep quiet until scheduled transmission is over (hear ACK)

# RTS / CTS Protocols (CSMA/CA)

B sends to C

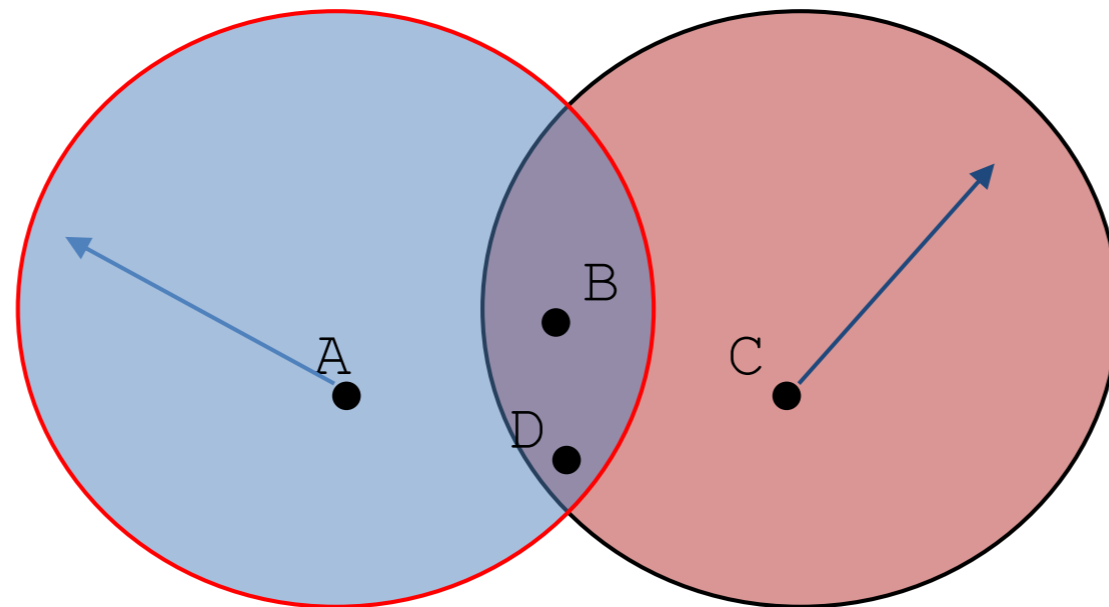


Overcome hidden terminal problems with contention-free protocol

1. B sends to C **Request To Send** (RTS)
2. A hears RTS and defers (to allow C to answer)
3. C replies to B with **Clear To Send** (CTS)
4. D hears CTS and defers to allow the data
5. B sends to C

# Preventing Collisions Altogether

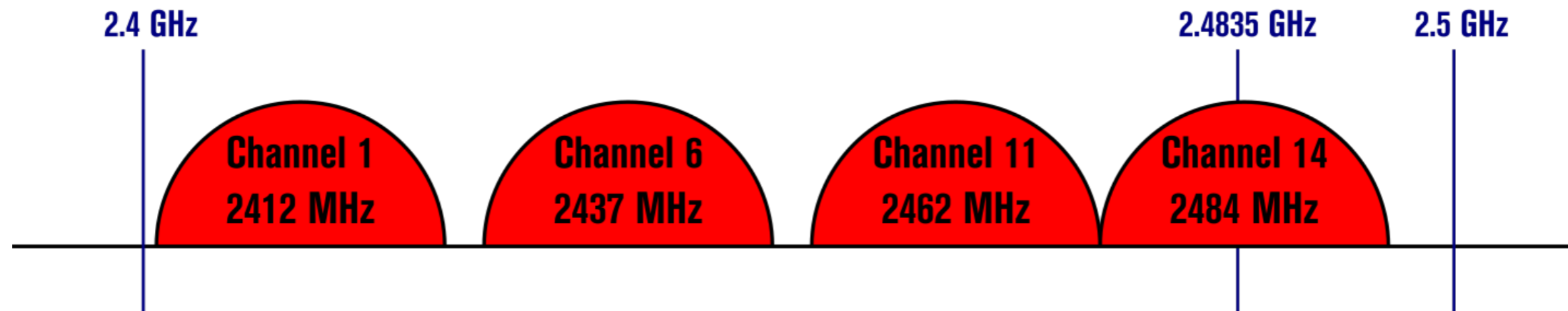
- Frequency Spectrum partitioned into several channels
  - Nodes within interference range can use separate channels



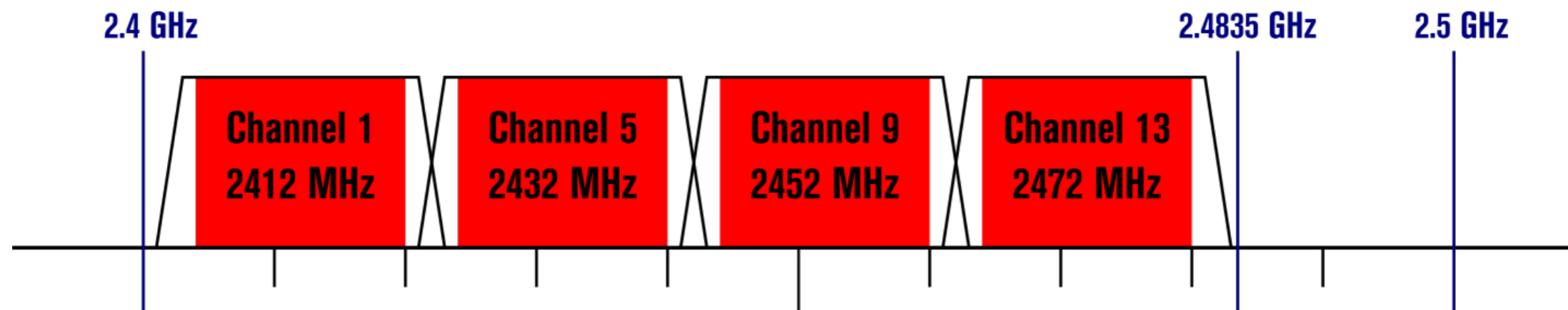
- Now A and C can send without any interference!
- Most cards have only 1 transceiver
  - **Not Full Duplex: Cannot send and receive at the same time**
  - Aggregate Network throughput doubles

# Non-Overlapping Channels for 2.4 GHz WLAN

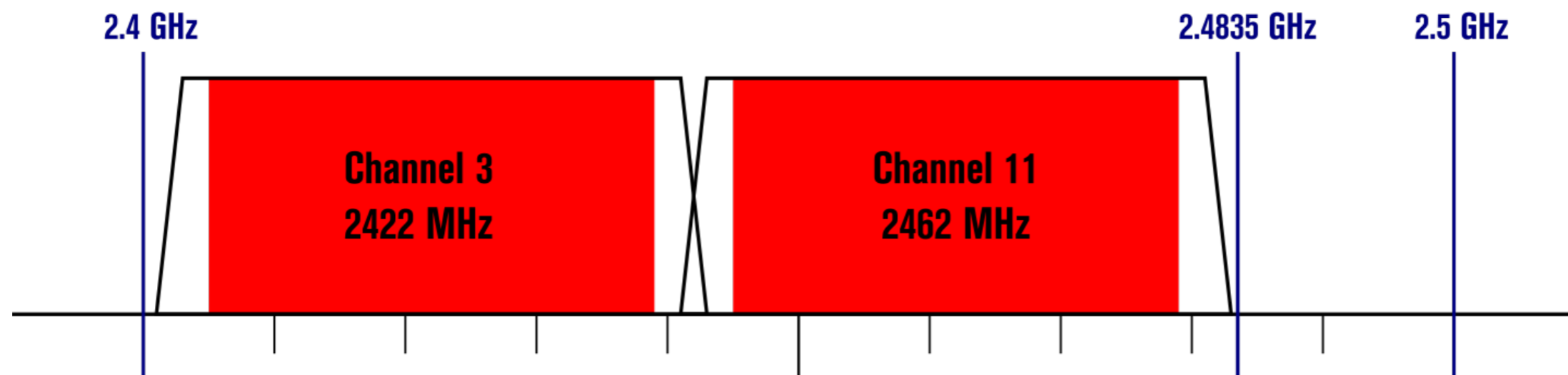
802.11b (DSSS) channel width 22 MHz

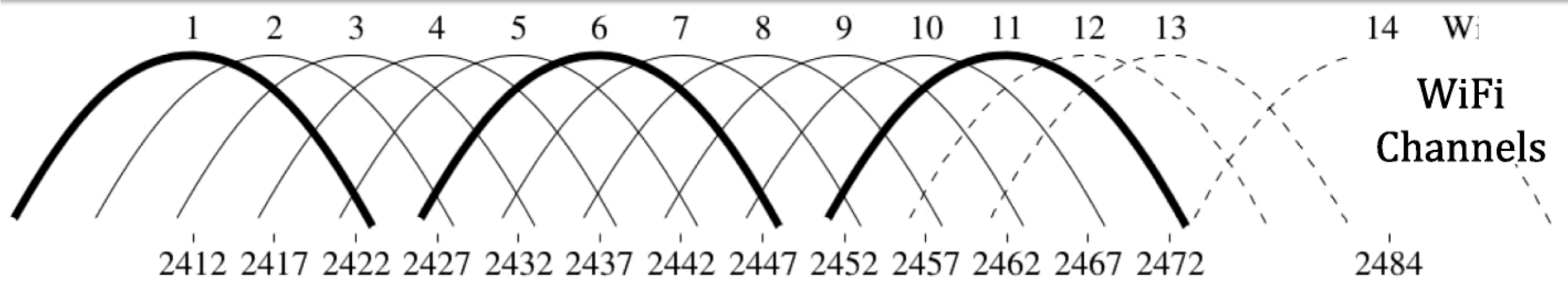


802.11g/n (OFDM) 20 MHz ch. width – 16.25 MHz used by sub-carriers



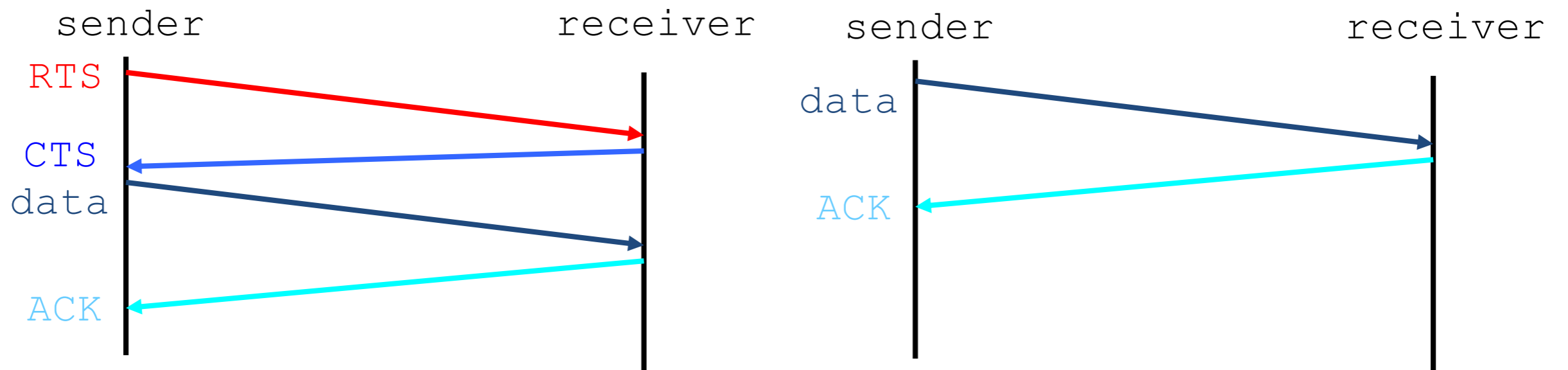
802.11n (OFDM) 40 MHz ch. width – 33.75 MHz used by sub-carriers







# CSMA/CA and RTS/CTS



## RTS/CTS

- helps with hidden terminal
- good for high-traffic Access Points
- often turned on/off dynamically

## Without RTS/CTS

- lower latency -> faster!
- reduces wasted b/w
  - if the  $Pr(\text{collision})$  is low
- good for when net is small and not *weird*
  - eg no hidden/exposed terminals

# CSMA/CD vs CSMA/CA (without RTS/CTS)

## CD Collision Detect

wired – listen and talk

1. Listen for others
2. Busy? goto 1.
3. Send message (and listen)
4. Collision?
  - a. JAM
  - b. increase your BEB
  - c. sleep
  - d. goto 1.

## CA Collision Avoidance

wireless – talk OR listen

1. Listen for others
2. Busy? goto 1.
3. Send message
4. Wait for ACK (*MAC ACK*)
5. Got No ACK from MAC?
  - a. increase your BEB
  - b. sleep
  - c. goto 1.

# Summary of MAC protocols

- *channel partitioning*, by time, frequency or code
  - Time Division (TDMA), Frequency Division (FDMA), Code Division (CDMA)
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in (old-style, coax) Ethernet, and PowerLine
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - Bluetooth, FDDI, IBM Token Ring

# MAC Addresses

- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - *burned* in NIC ROM, nowadays usually software settable and set at boot time

```
awm22@rio:~$ ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:30:48:fe:c0:64
          inet addr:128.232.33.4 Bcast:128.232.47.255 Mask:255.255.240.0
          inet6 addr: fe80::230:48ff:fe:c064/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:215084512 errors:252 dropped:25 overruns:0 frame:123
          TX packets:146711866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:170815941033 (170.8 GB) TX bytes:86755864270 (86.7 GB)
          Memory:f0000000-f0020000
```

# LAN Address (more)

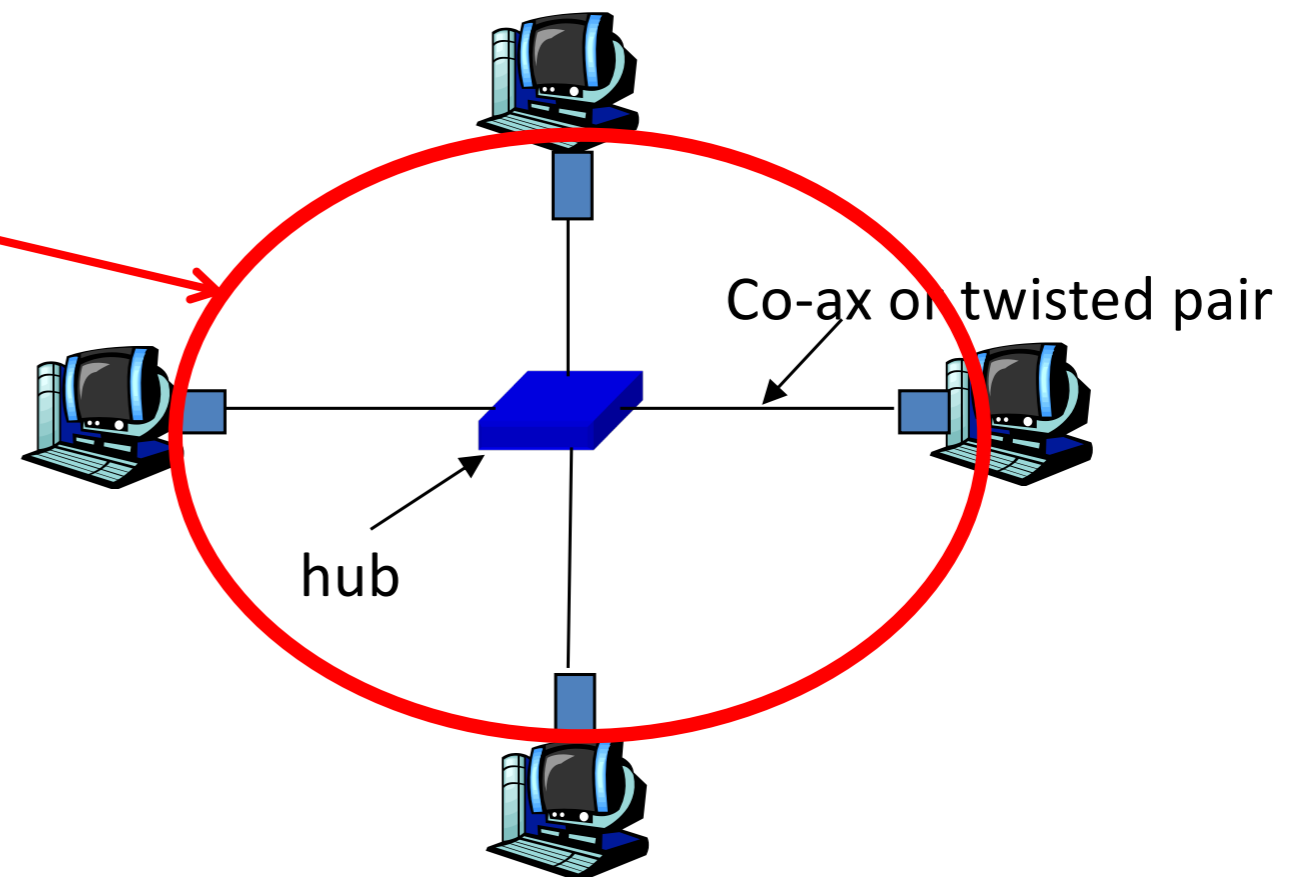
- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - (a) MAC address: like a National Insurance Number
  - (b) IP address: like a postal address
- MAC flat address → portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached

# Hubs

... physical-layer (“dumb”) repeaters:

- bits coming in one link go out *all* other links at same rate
- all nodes connected to hub can collide with one another
- no frame buffering
- no CSMA/CD at hub: host NICs detect collisions

Collision Domain  
in CSMA/CD *speak*

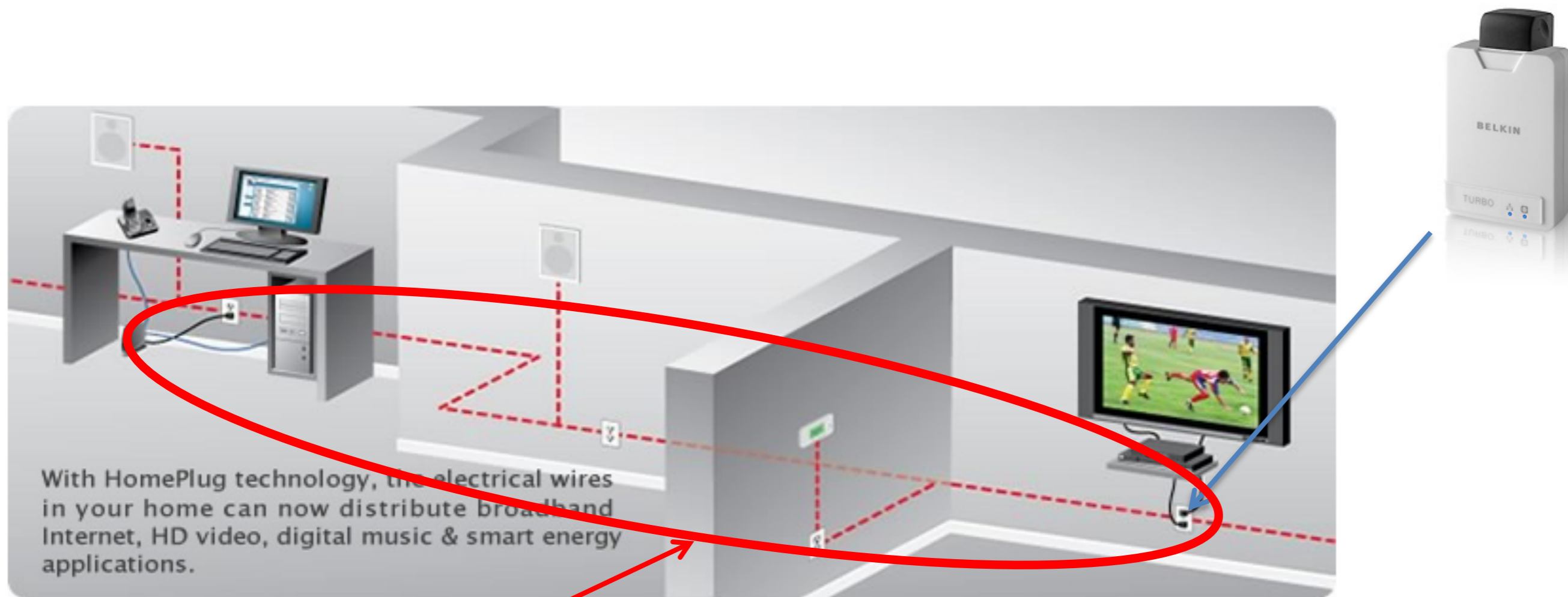


# CSMA in our home

## Home Plug Powerline Networking...



# Home Plug and similar Powerline Networking....



Collision Domain  
in CSMA speak

To secure network traffic on a specific HomePlug network, each set of adapters use an encryption key common to a specific HomePlug network



# Switch

*(like a Hub but smarter)*

- **link-layer device: smarter than hubs, take *active* role**
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, **selectively** forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- ***transparent***
  - hosts are unaware of presence of switches
- ***plug-and-play, self-learning***
  - switches do not need to be configured

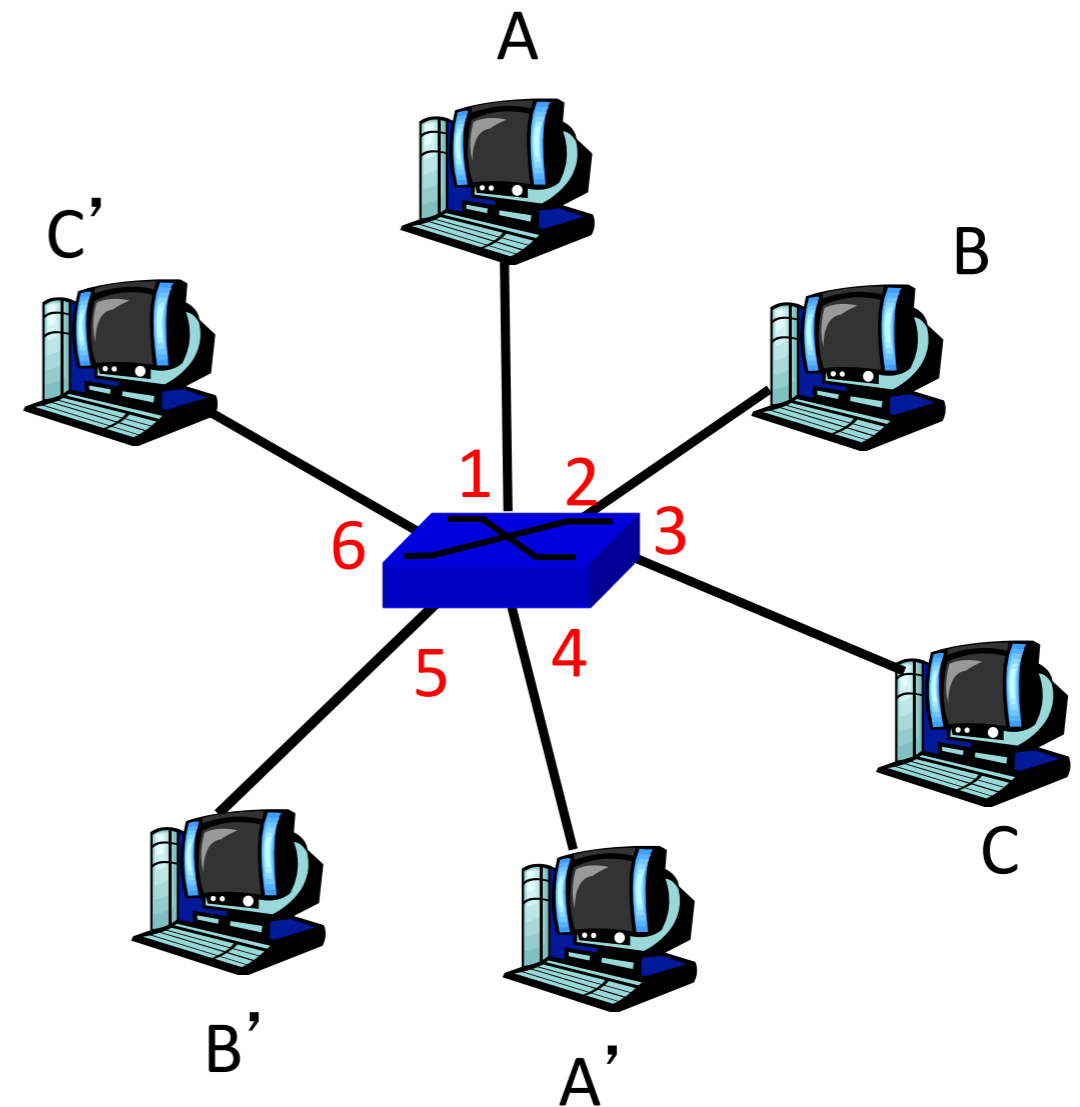
If you want to connect different physical media  
(optical – copper – coax – wireless - ....)

you **NEED** a switch.

Why? (Because each link, each media access protocol is specialised)

# Switch: allows *multiple* simultaneous transmissions

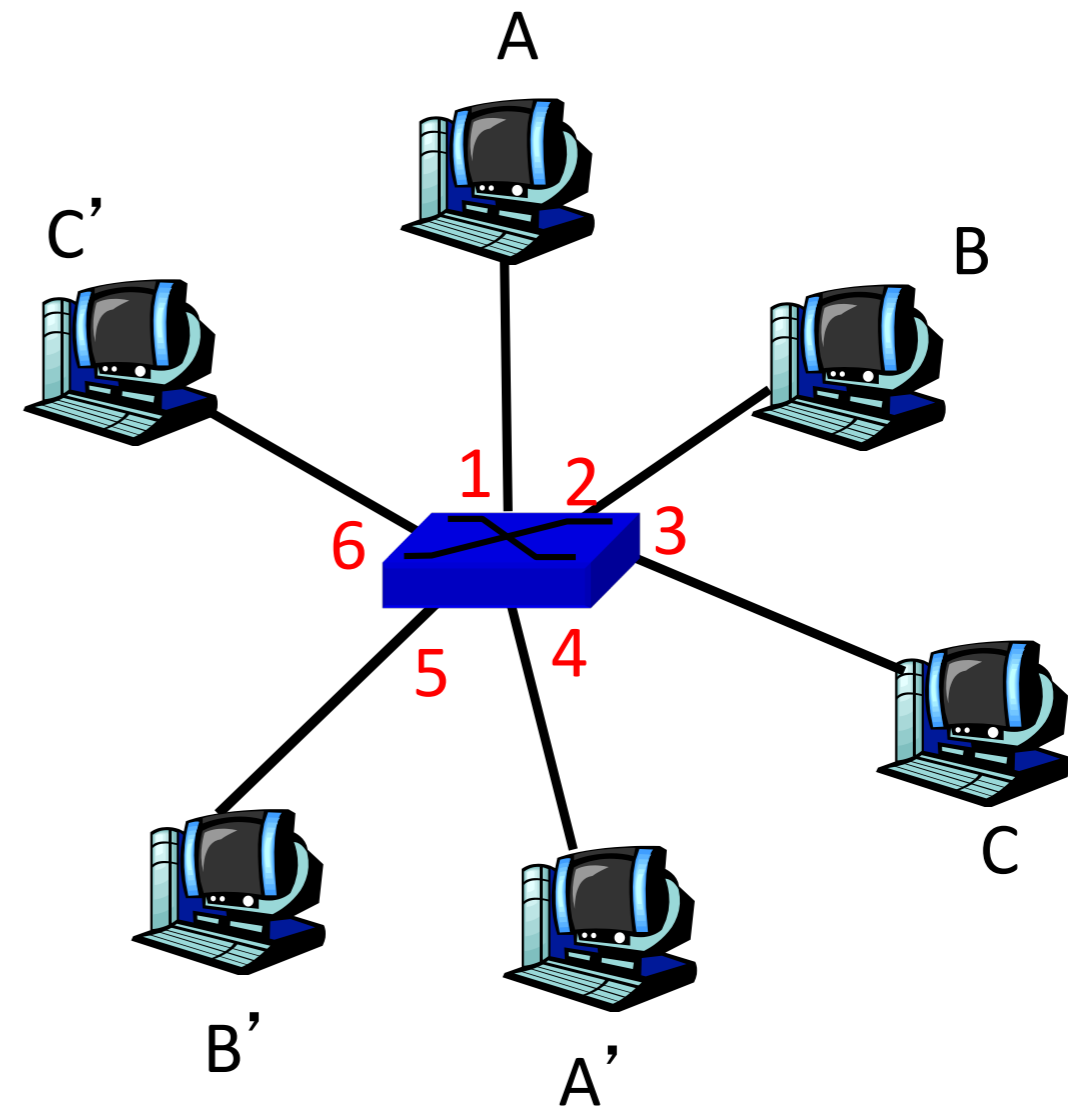
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- **switching**: A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



switch with six interfaces  
(1,2,3,4,5,6)

# Switch Table

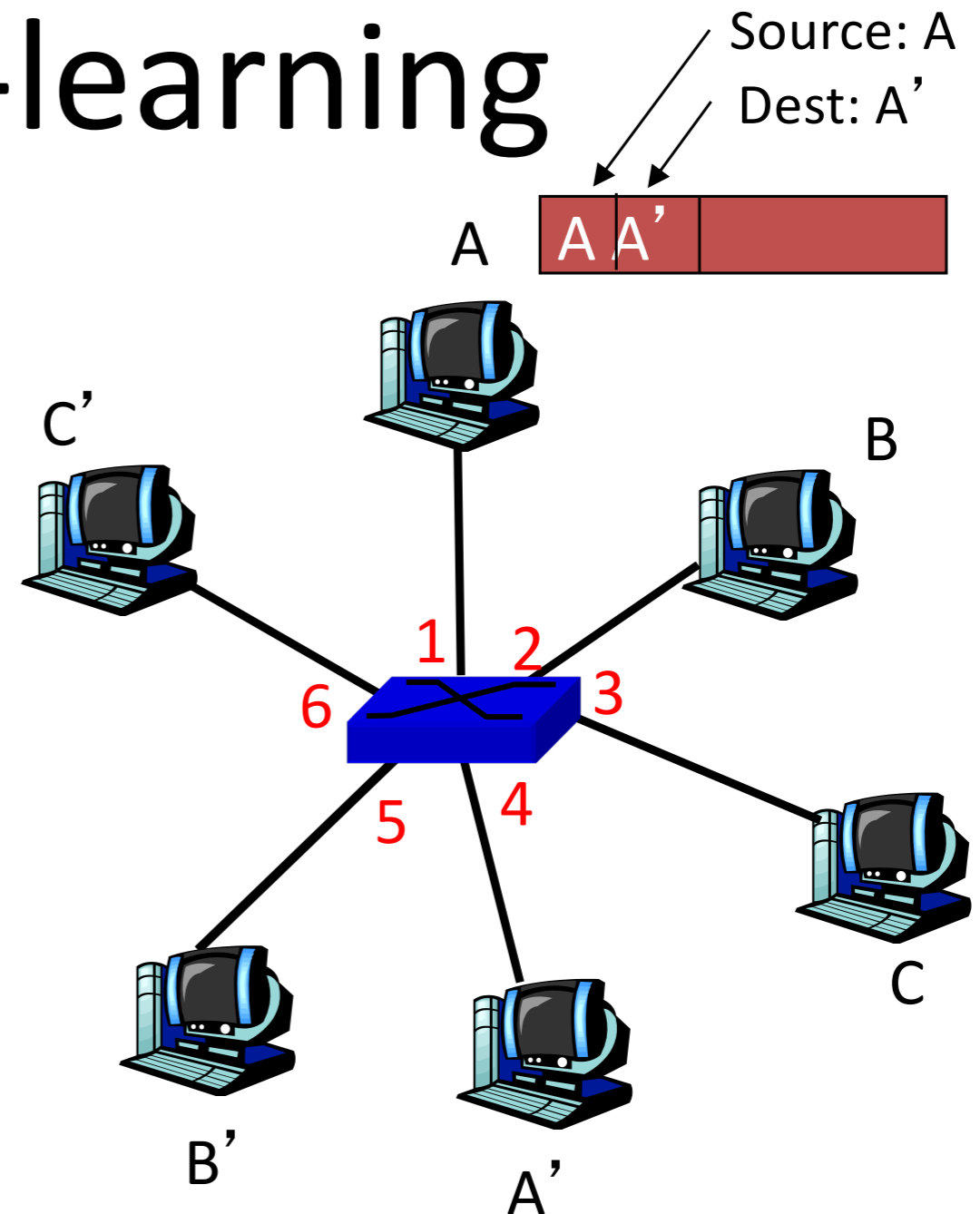
- Q: how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- A: each switch has a **switch table**, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- Q: how are entries created, maintained in switch table?
  - something like a routing protocol?



*switch with six interfaces  
(1,2,3,4,5,6)*

# Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch “learns” location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	1	60

*Switch table  
(initially empty)*

# Switch: frame filtering/forwarding

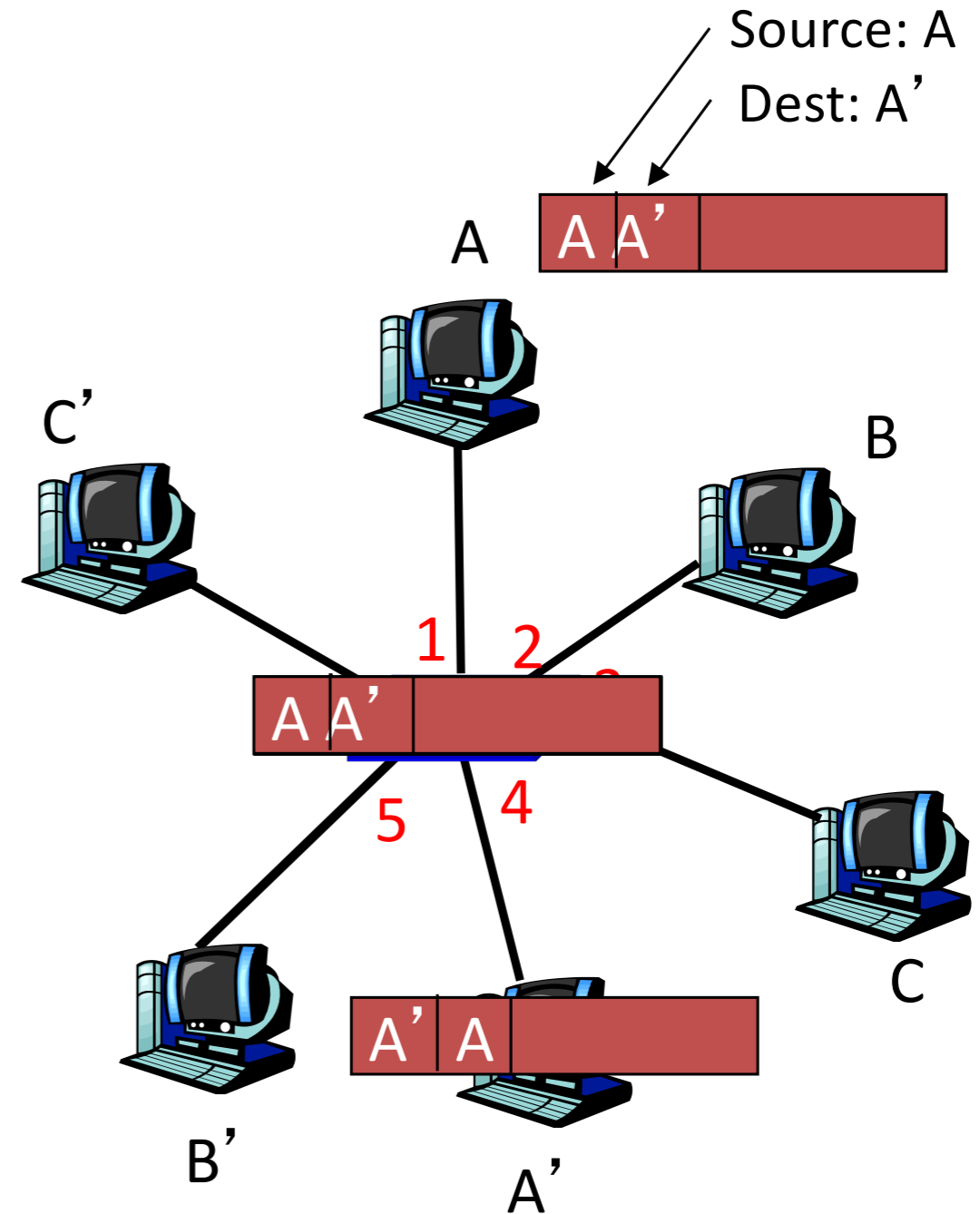
## When frame received:

1. record link associated with sending host
2. index switch table using MAC dest address
- 3. if** entry found for destination  
    **then {**  
        **if** dest on segment from which frame arrived  
            **then** drop the frame  
            **else** forward the frame on interface indicated  
        **}**  
    **else** flood

*forward on all but the interface  
on which the frame arrived*

# Self-learning, forwarding: example

- frame destination unknown: *flood*
- destination A location known: *selective send*

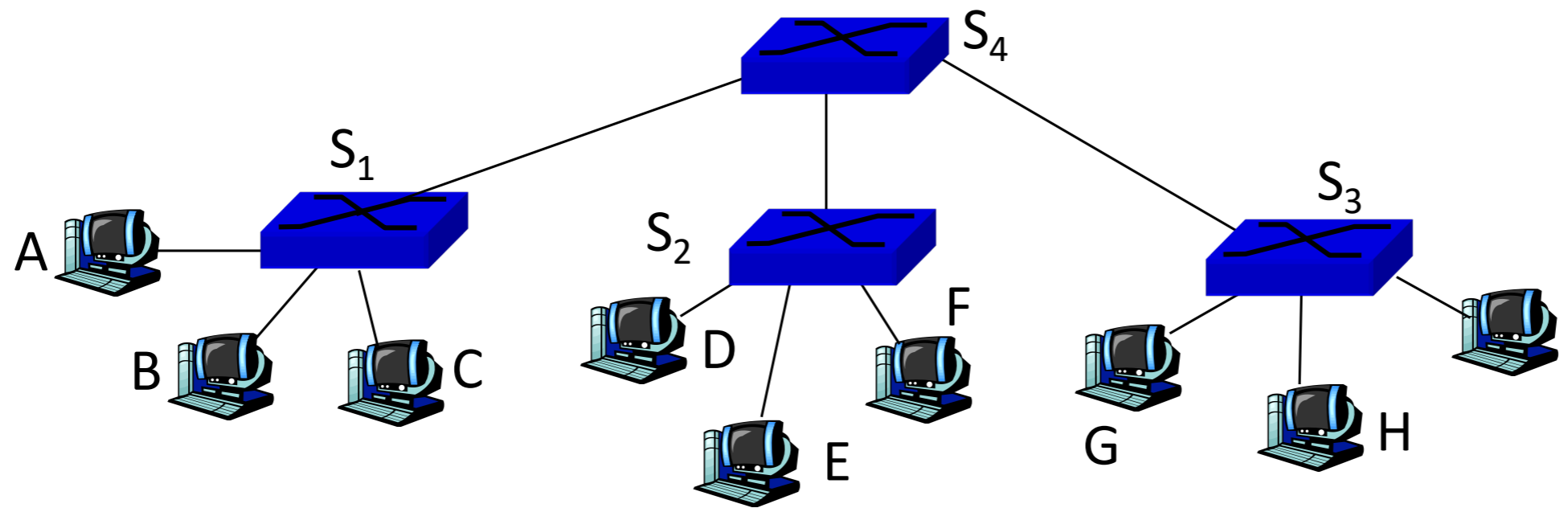


MAC addr	interface	TTL
A	1	60
A'	4	60

*Switch table  
(initially empty)*

# Interconnecting switches

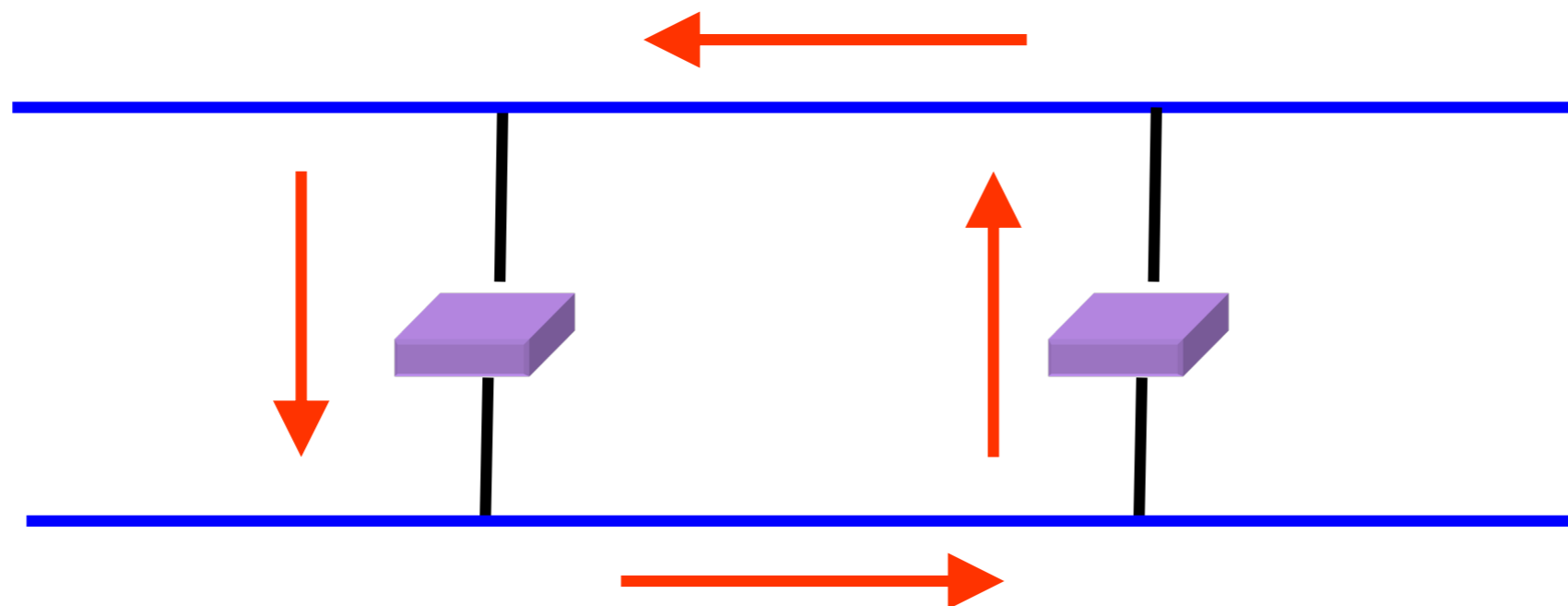
- switches can be connected together



- r **Q:** sending from A to G - how does S<sub>1</sub> know to forward frame destined to F via S<sub>4</sub> and S<sub>3</sub>?
- r **A:** self learning! (works exactly the same as in single-switch case – **flood/forward/drop**)

# Flooding Can Lead to Loops

- Flooding can lead to **forwarding loops**
  - E.g., if the network contains a cycle of switches
  - “Broadcast storm”

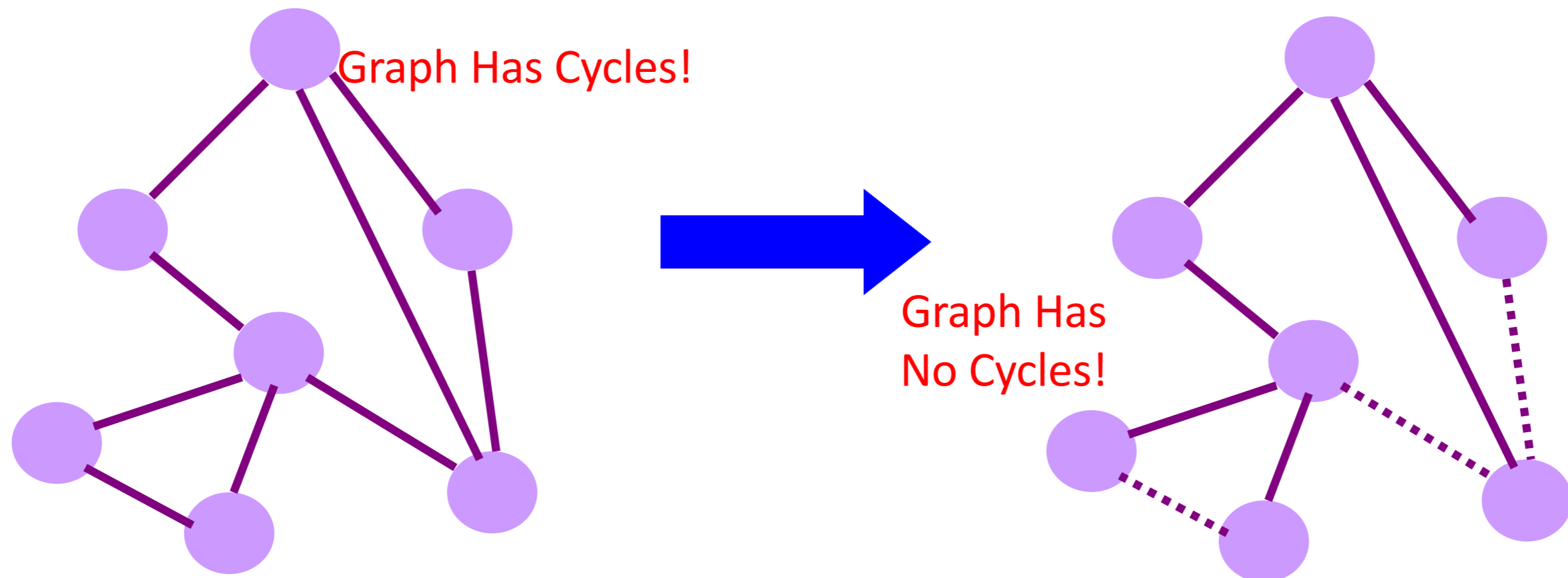






# Solution: Spanning Trees

- Ensure the forwarding **topology** has no loops
  - Avoid using some of the links when flooding
  - ... to prevent loop from forming
- **Spanning tree**
  - **Sub-graph** that covers all vertices but *contains no cycles*
  - Links not in the spanning tree do not forward frames

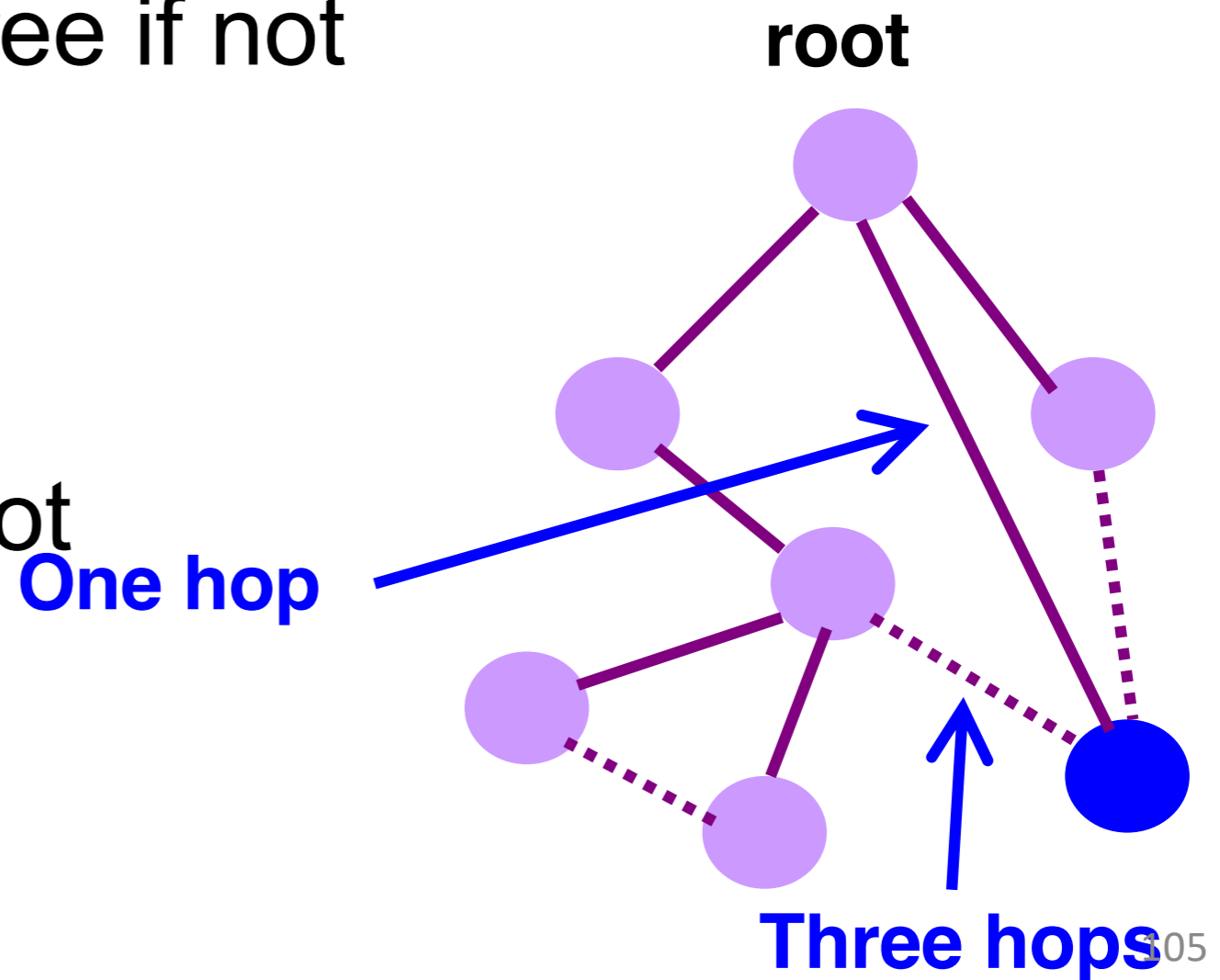


# What Do We Know?

- *“Spanning tree algorithm is an algorithm to create a tree out of a graph that includes all nodes with a minimum number of edges connecting to vertices.”*
- Shortest paths to (or from) a node form a tree
- So, algorithm has two aspects :
  - Pick a root
  - Compute shortest paths to it
- Only keep the links on shortest-path

# Constructing a Spanning Tree

- Switches need to **elect a root**
  - The switch w/ smallest identifier (MAC addr)
- Each switch determines if each interface is on the **shortest path** from the root
  - Excludes it from the tree if not
- Messages (Y, d, X)
  - From node X
  - Proposing Y as the root
  - And the distance is d

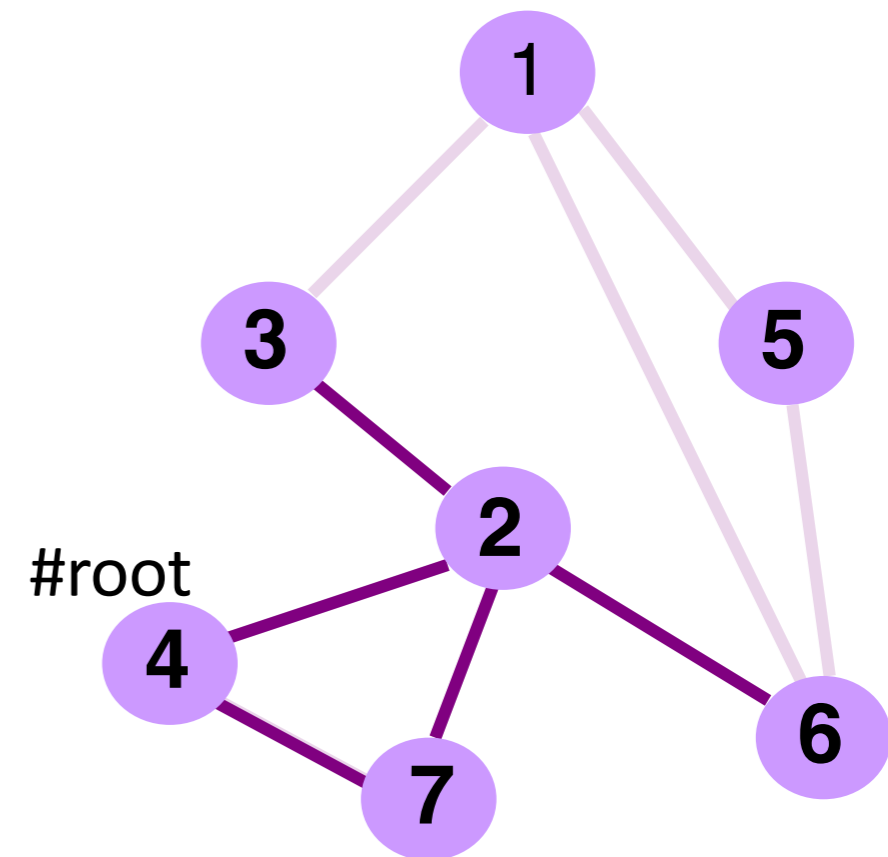


# Steps in Spanning Tree Algorithm

- Initially, each switch proposes itself as the root
  - Switch sends a message out every interface
  - ... proposing itself as the root with distance 0
  - Example: switch X announces (X, 0, X)
- Switches update their view of the root
  - Upon receiving message (Y, d, Z) from Z, check Y's id
  - If new id smaller, start viewing that switch as root
- Switches compute their distance from the root
  - Add 1 to the distance received from a neighbor
  - Identify interfaces not on shortest path to the root
  - ... and exclude them from the spanning tree
- If root or shortest distance to it **changed**, “flood” updated message (Y, d+1, X)

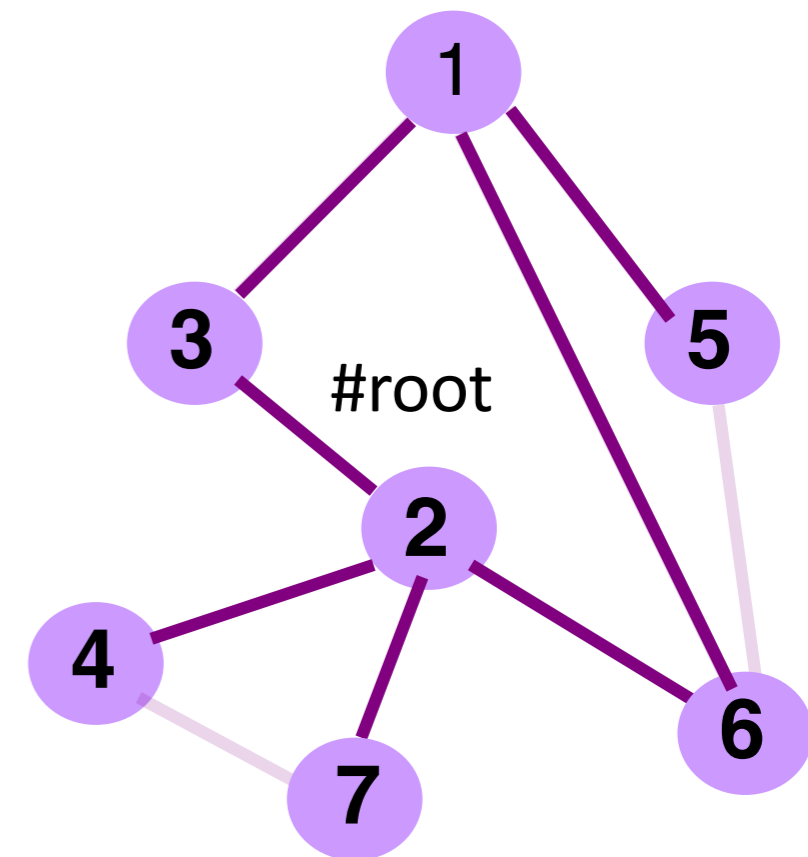
# Example From Switch #4' s Viewpoint

- Switch #4 thinks it is the root
  - Sends (4, 0, 4) message to 2 and 7
- Then, switch #4 hears from #2
  - Receives (2, 0, 2) message from 2
  - ... and thinks that #2 is the root
  - And realizes it is just one hop away
- Then, switch #4 hears from #7
  - Receives (2, 1, 7) from 7
  - And realizes this is a longer path
  - So, prefers its own one-hop path
  - And removes 4-7 link from the tree



# Example From Switch #4' s Viewpoint

- Switch #2 hears about switch #1
  - Switch 2 hears (1, 1, 3) from 3
  - Switch 2 starts treating 1 as root
  - And sends (1, 2, 2) to neighbors
- Switch #4 hears from switch #2
  - Switch 4 starts treating 1 as root
  - And sends (1, 3, 4) to neighbors
- Switch #4 hears from switch #7
  - Switch 4 receives (1, 3, 7) from 7
  - And realizes this is a longer path
  - So, prefers its own three-hop path
  - And removes 4-7 link from the tree



# Robust Spanning Tree Algorithm

- Algorithm must react to **failures**
  - Failure of the root node
    - Need to elect a new root, with the next lowest identifier
  - Failure of other switches and links
    - Need to recompute the spanning tree
- Root switch continues sending messages
  - Periodically reannouncing itself as the root (1, 0, 1)
  - Other switches continue forwarding messages
- Detecting failures through timeout (**soft state**)
  - If no word from root, times out and claims to be the root
  - Delay in reestablishing spanning tree is **major problem**
  - Work on rapid spanning tree algorithms...

Given a switch-tree of a given size, link length, speed of computation, ...

How long does a failure take to rectify?

# Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - Ethernet
  - switched LANS
  - WiFi
- algorithms
  - Binary Exponential Backoff
  - Spanning Tree