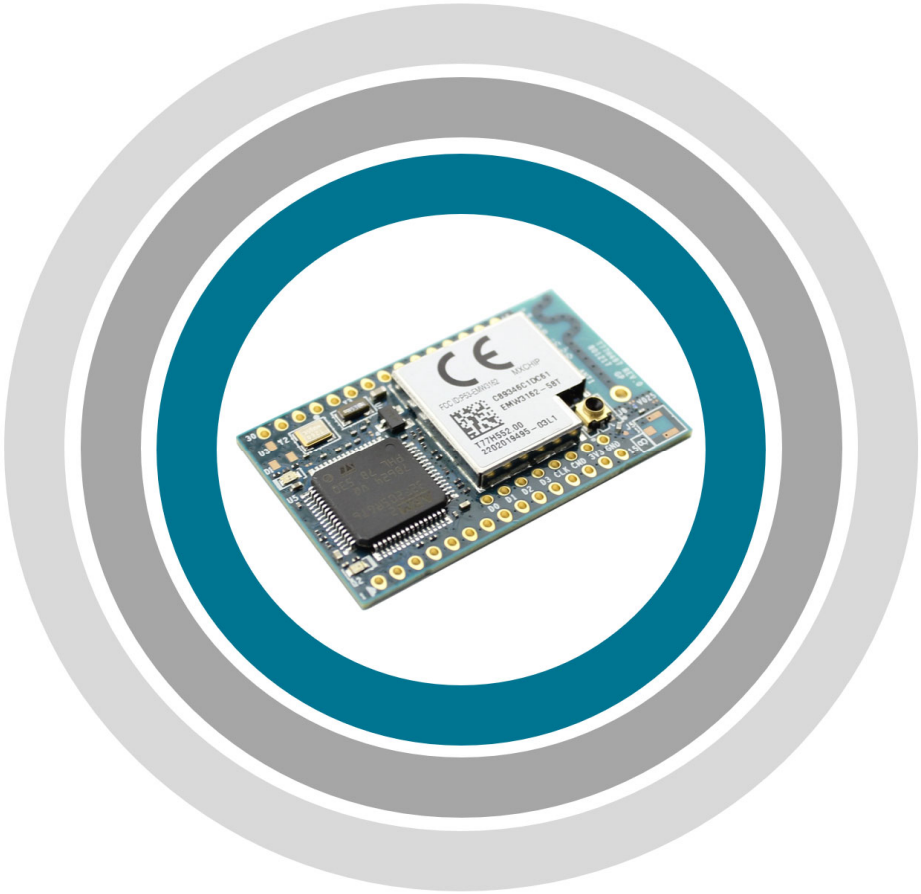
Internet of Things

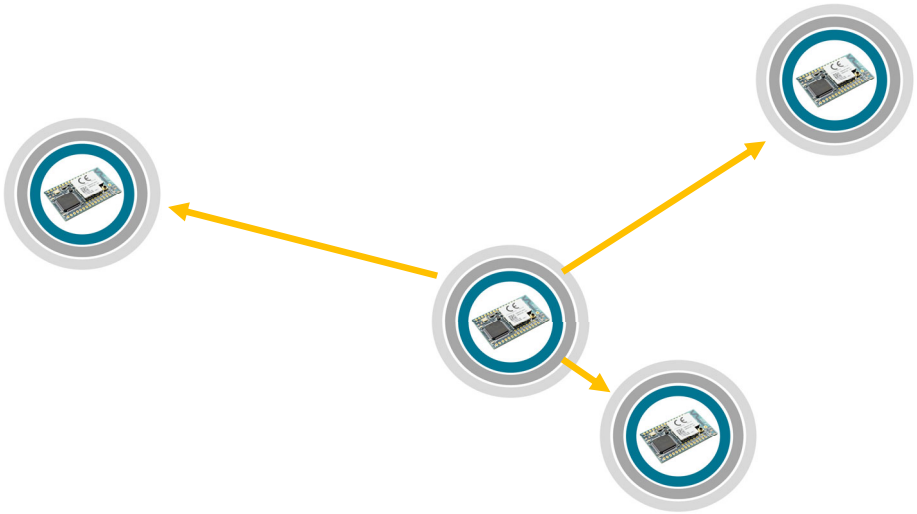
Cycle 3: IoT Protocols

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

What Are IoT Networks



What Are IoT Networks



Challenges We Need to Deal With

1. How to send information without wires?
   - *Solution: Radio Frequency Communications*

2. How to share the channel?
   - *Solution: Media Access Control (MAC)*

3. How to communicate across multiple hops?
   - *Solution: Mesh Routing and Service Discovery*

Radio Frequency Communications

Radio Frequency Communications

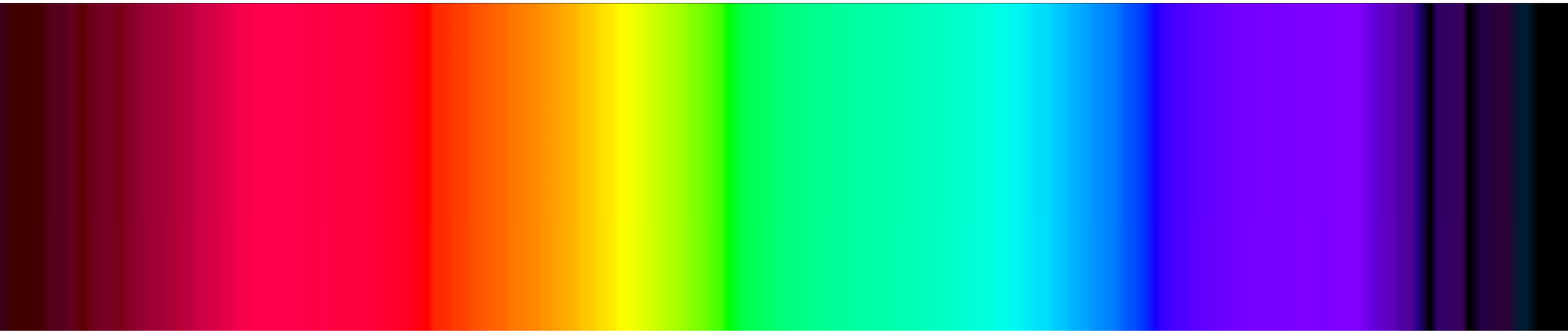
1. Electromagnetism theory
   → What is RF?
2. Antenna design
   → How can we send RF?
3. Signal propagation
   → How do wireless signals propagate?
4. Modulation
   → How do we encode data on RF?

What is light?

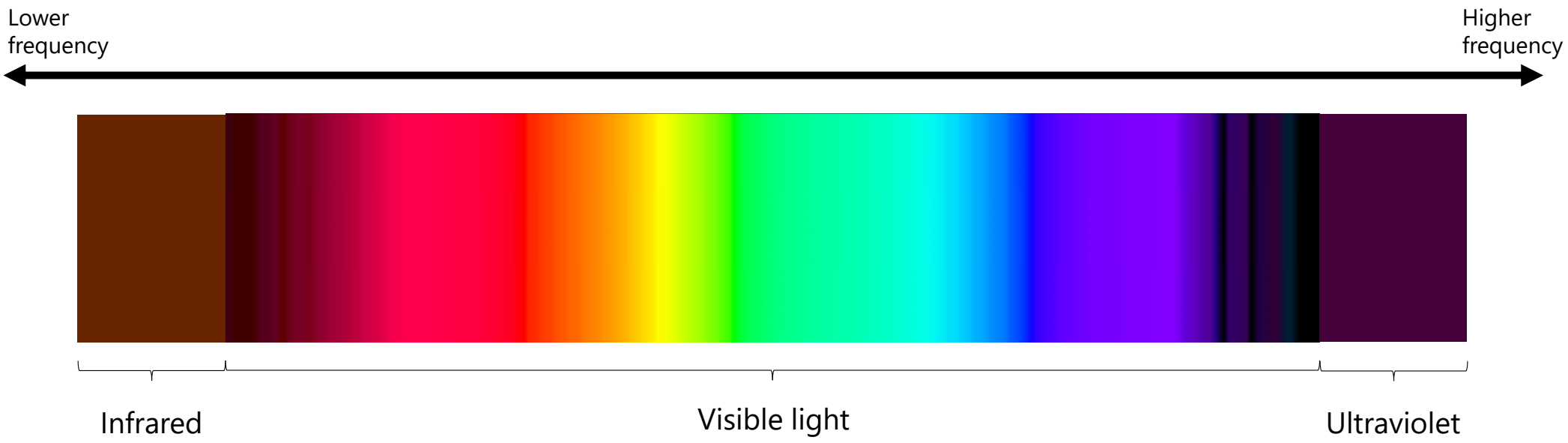
What is light?

Lower frequency

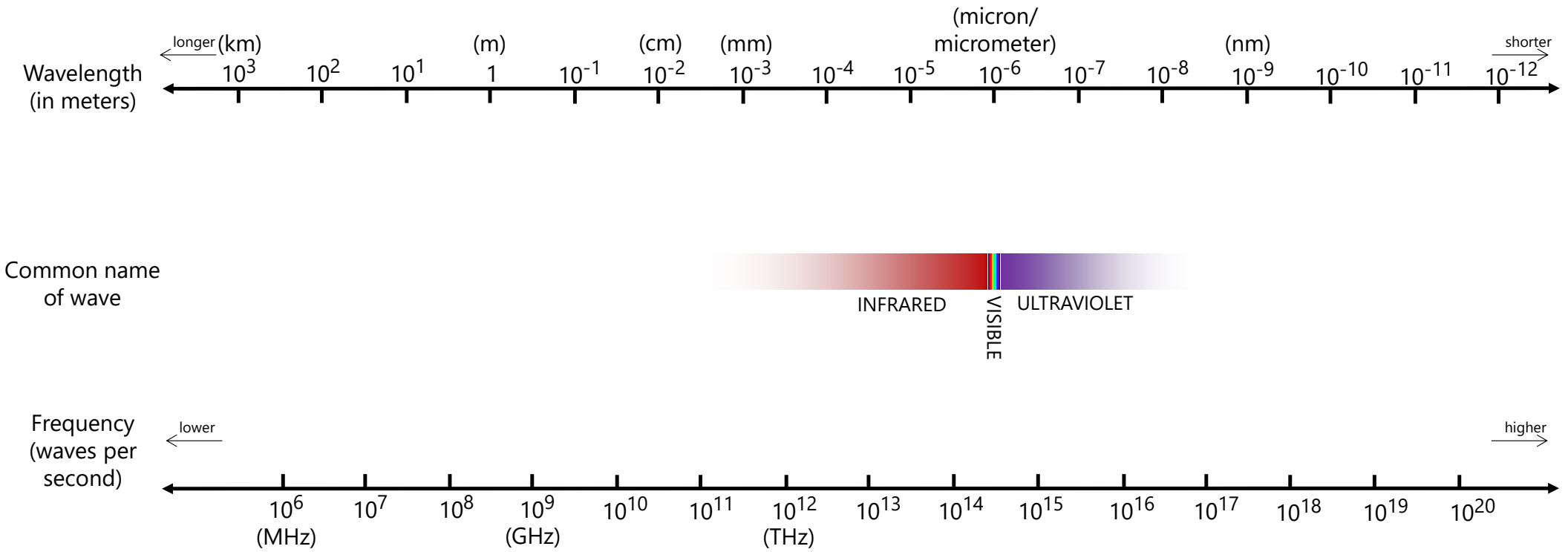
Higher frequency



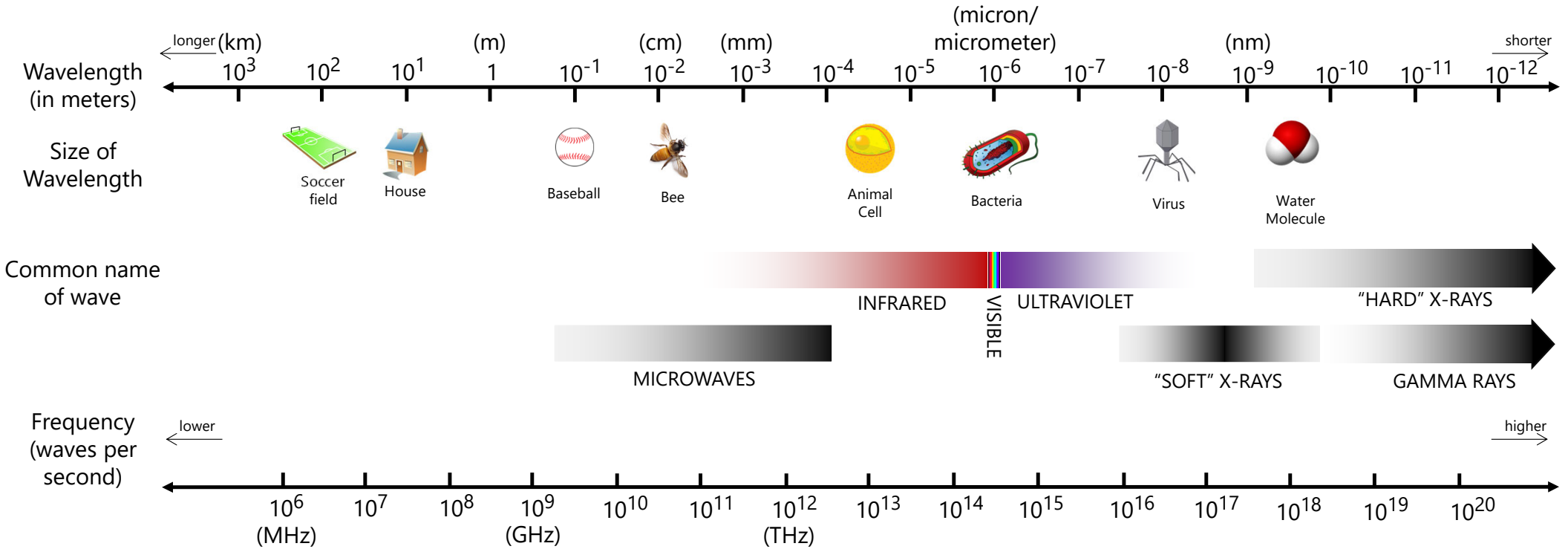
What is light?



The Electromagnetic Spectrum



The Electromagnetic Spectrum



Can we use EM waves to communicate?

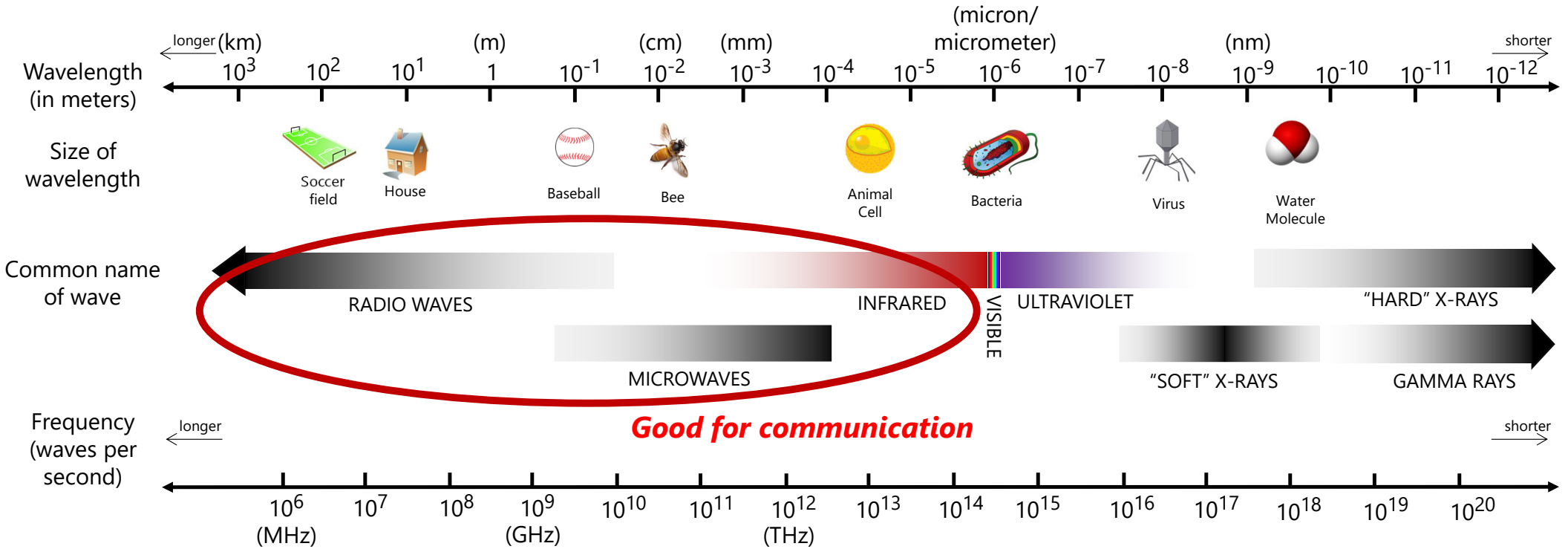
- Observation: We can “encode” information by sending an EM source and varying properties of it
  - Brightness, color, etc.

- Key question: Which frequency should we use?
  - EM radiation acts differently at different wavelengths

EM Radiation Acts Differently at Different Wavelengths

- Higher frequencies are more dangerous to humans
- High and low frequencies tend to go through objects
  - IR, optical, and UV tend to be dissipated by air and building materials
- Lower and middle frequencies are easier to make and precisely control with circuits

The Electromagnetic Spectrum

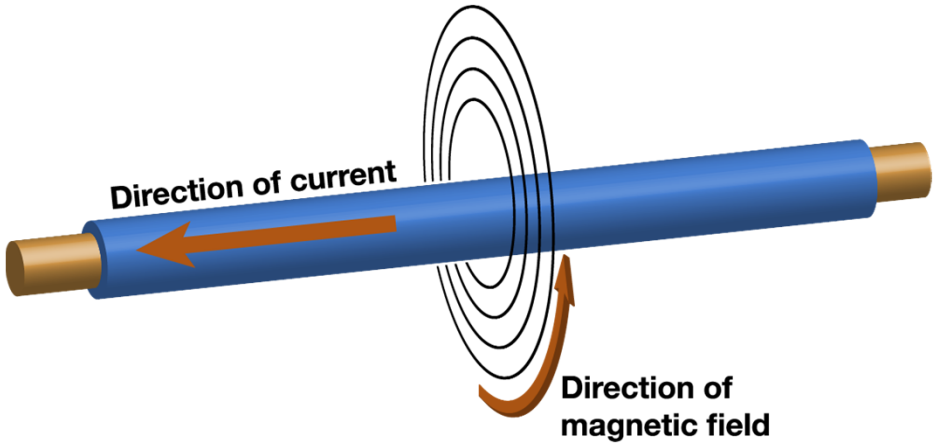


- Radio frequency: 3kHz to 300GHz
  - Easy to generate, good propagation characteristics, relatively safe for people

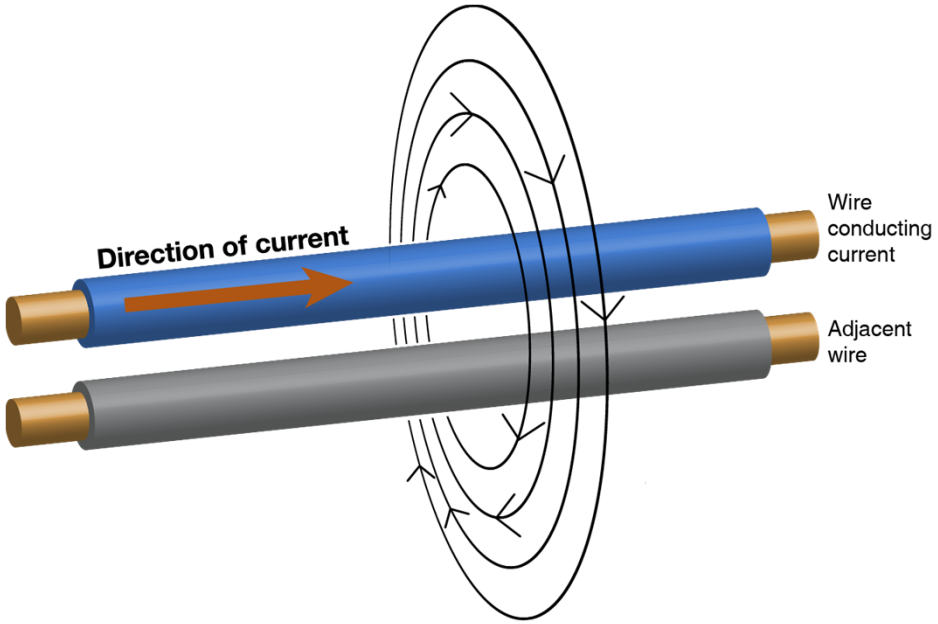
How to generate EM waves?

- Need some way to generate them, controllably
  - At specific frequencies, with certain patterns to encode data

- Observation: Electricity in a wire produces EM field

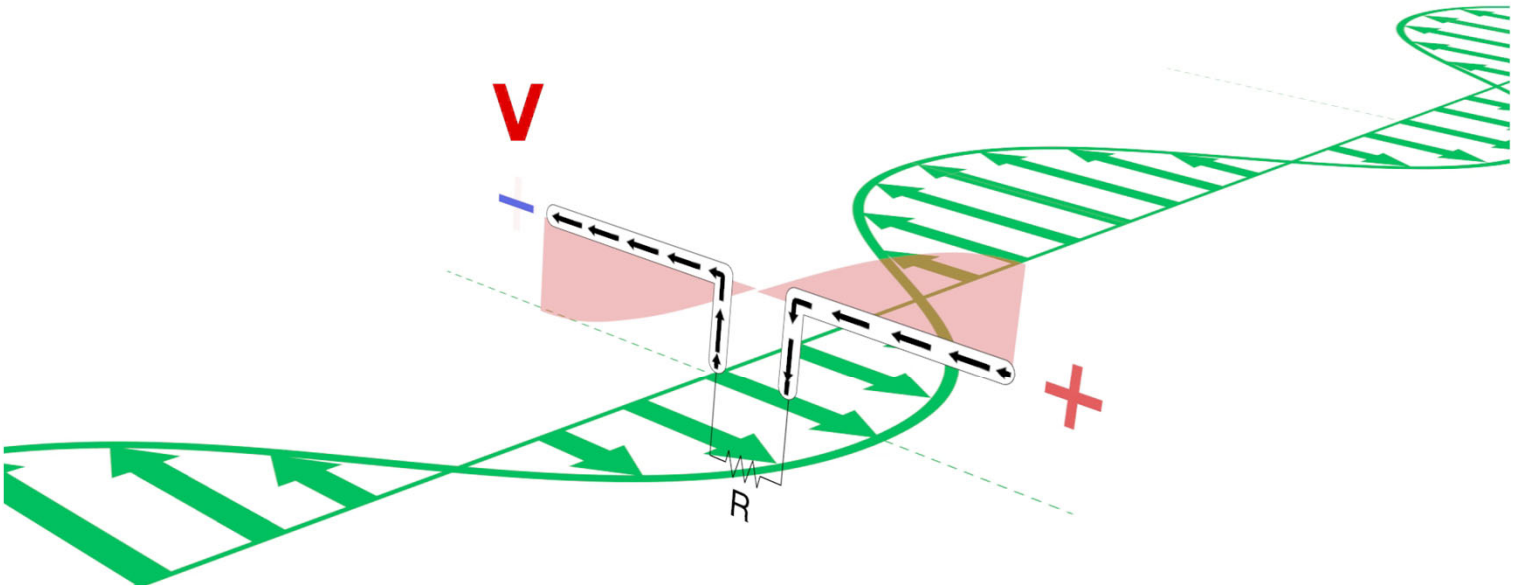


Magnetic Field Induces Current Flow



- This can be bad (crosstalk)
- This can be good (can transmit data)

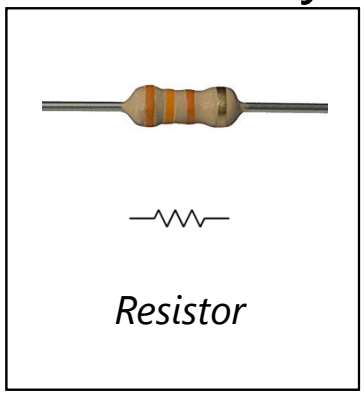
We Can Use EM Leakage to Transmit Data

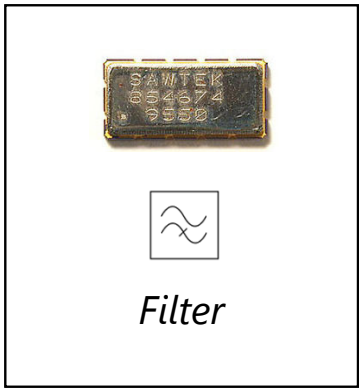


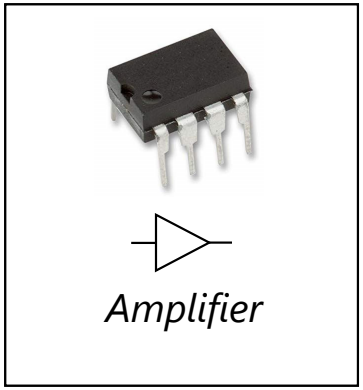
- Idea: Modulate electricity through wire
  - Change modulation to encode information
  - Bend wires to improve signal strength (dipole)

How to modulate electricity?

- We can make electrical components, which can manipulate electricity

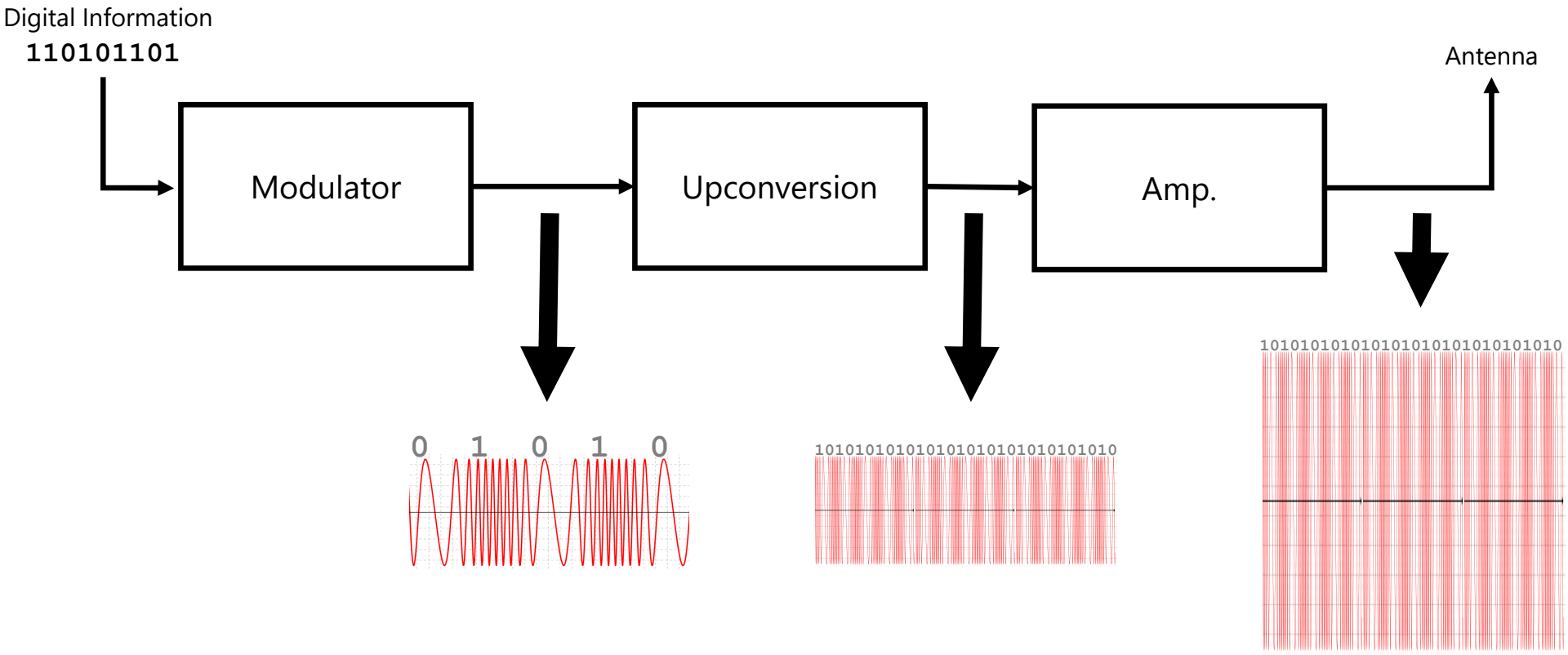




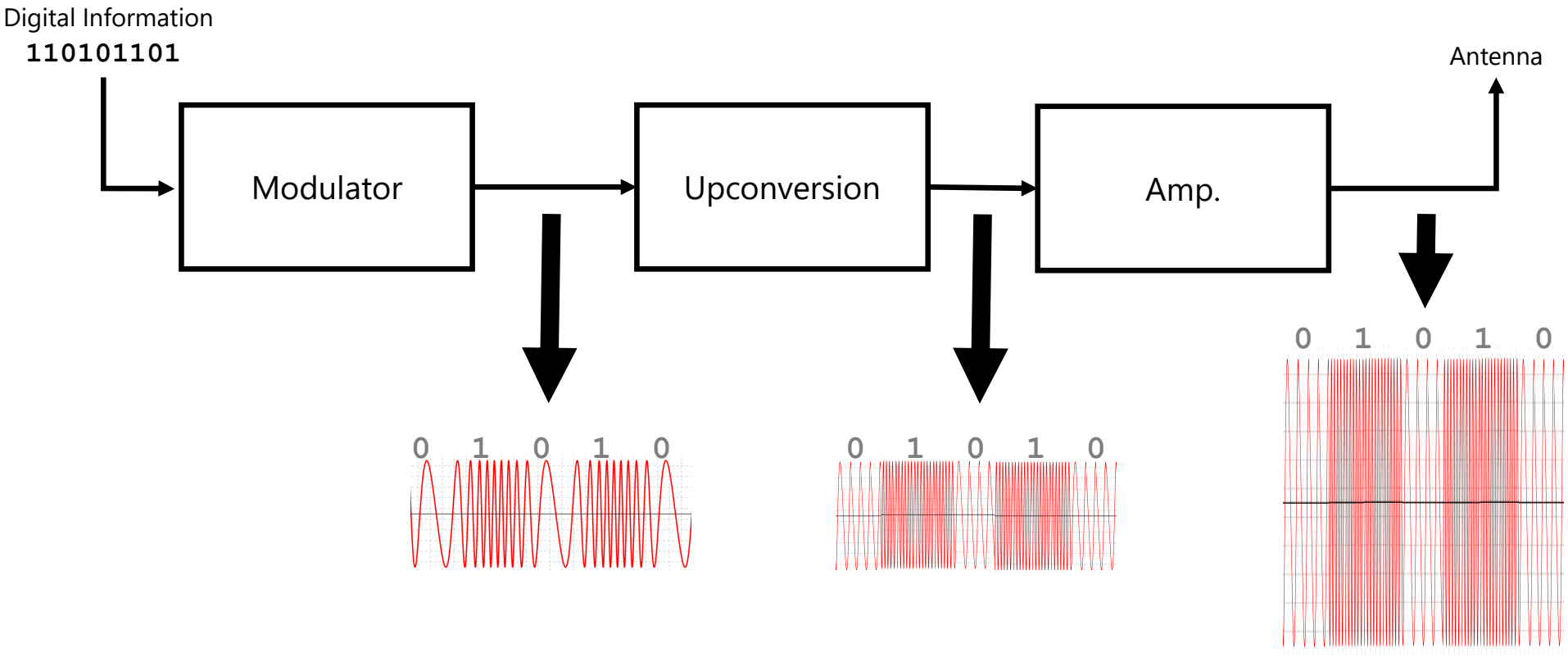


- We can combine these components into “circuits”
- Circuits that operate on radio frequencies are called RF circuits

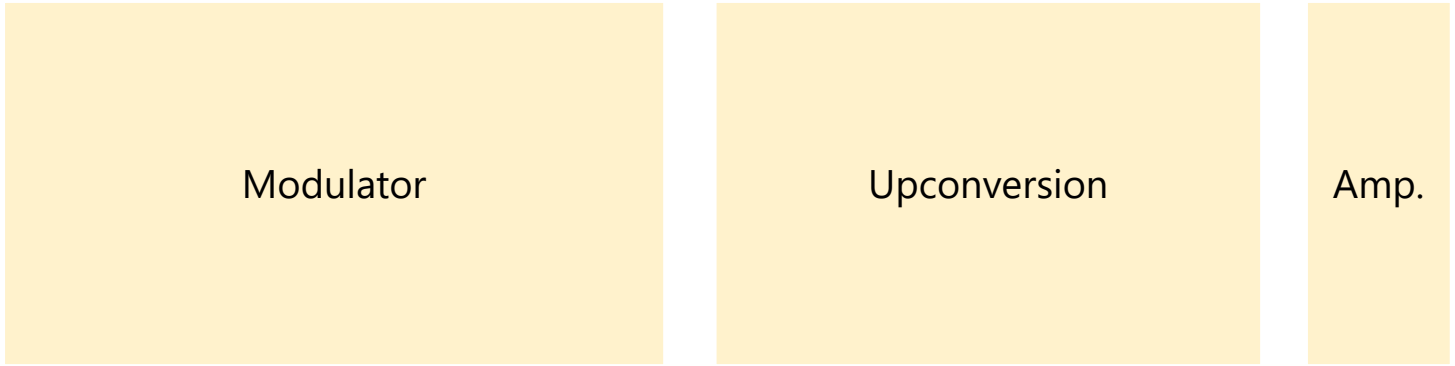
RF Transmitter Architecture



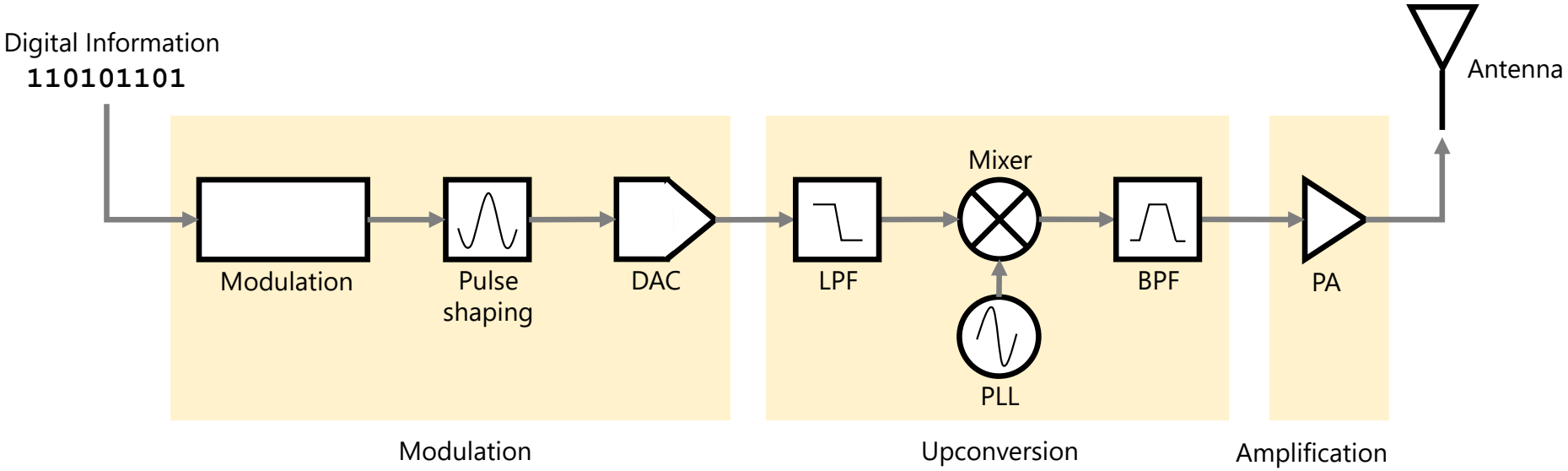
RF Transmitter Architecture



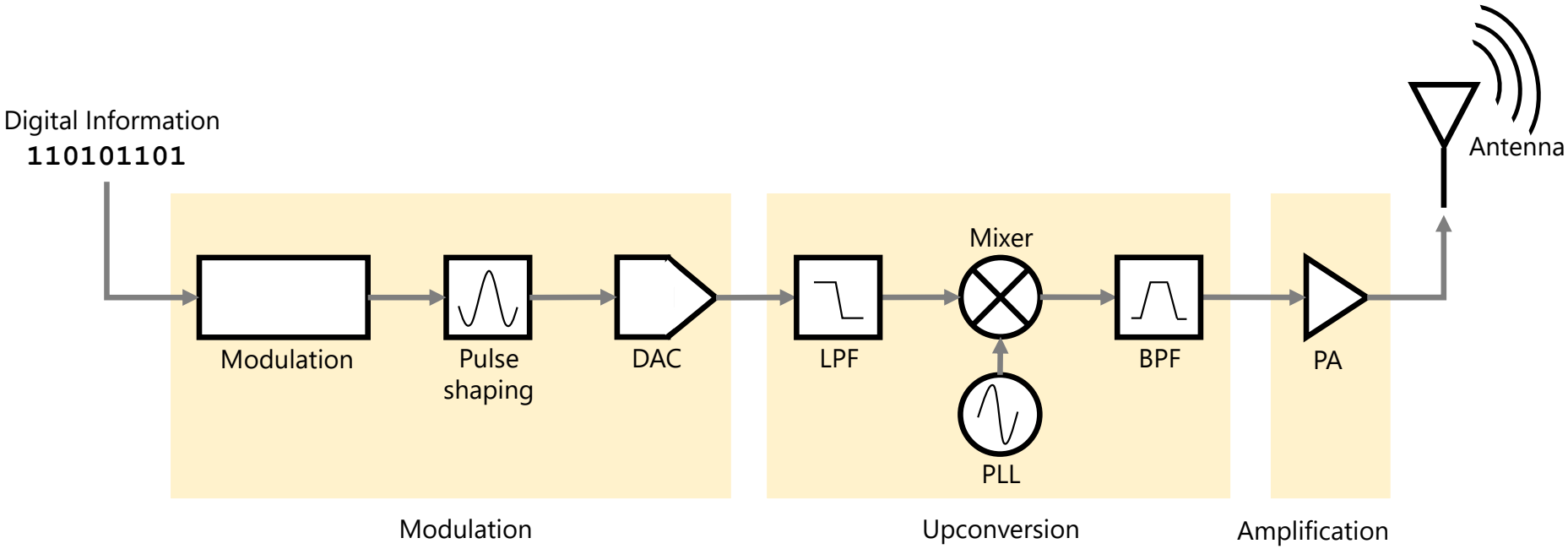
RF Transmitter Architecture



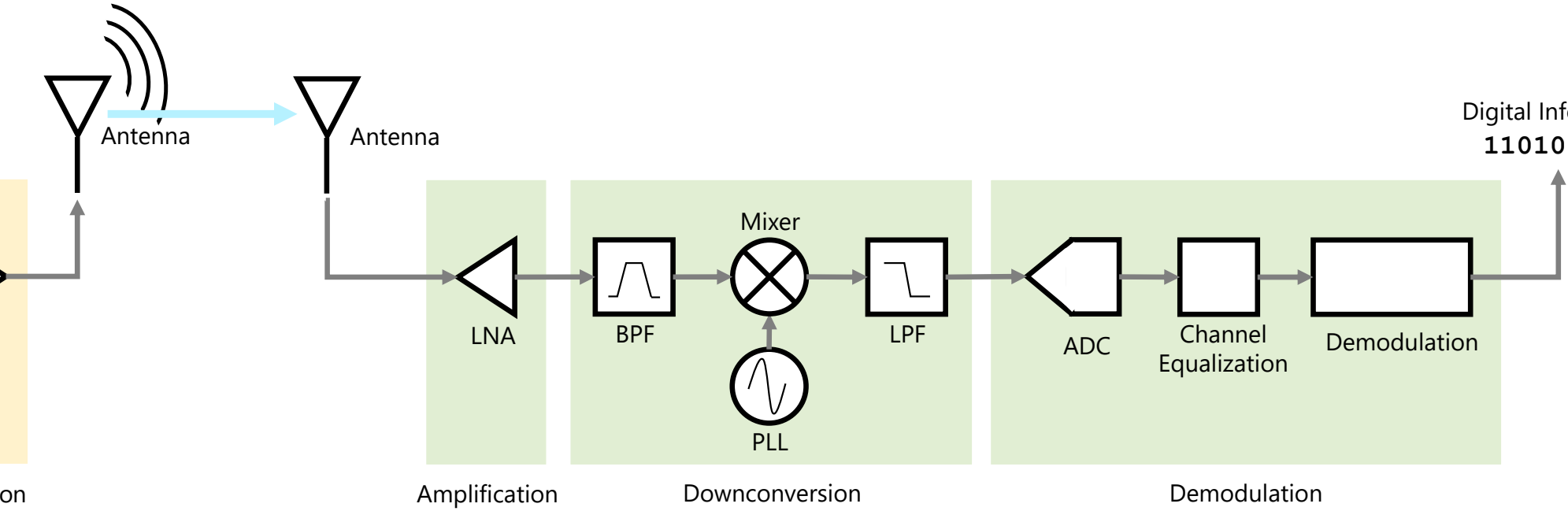
RF Transmitter Architecture



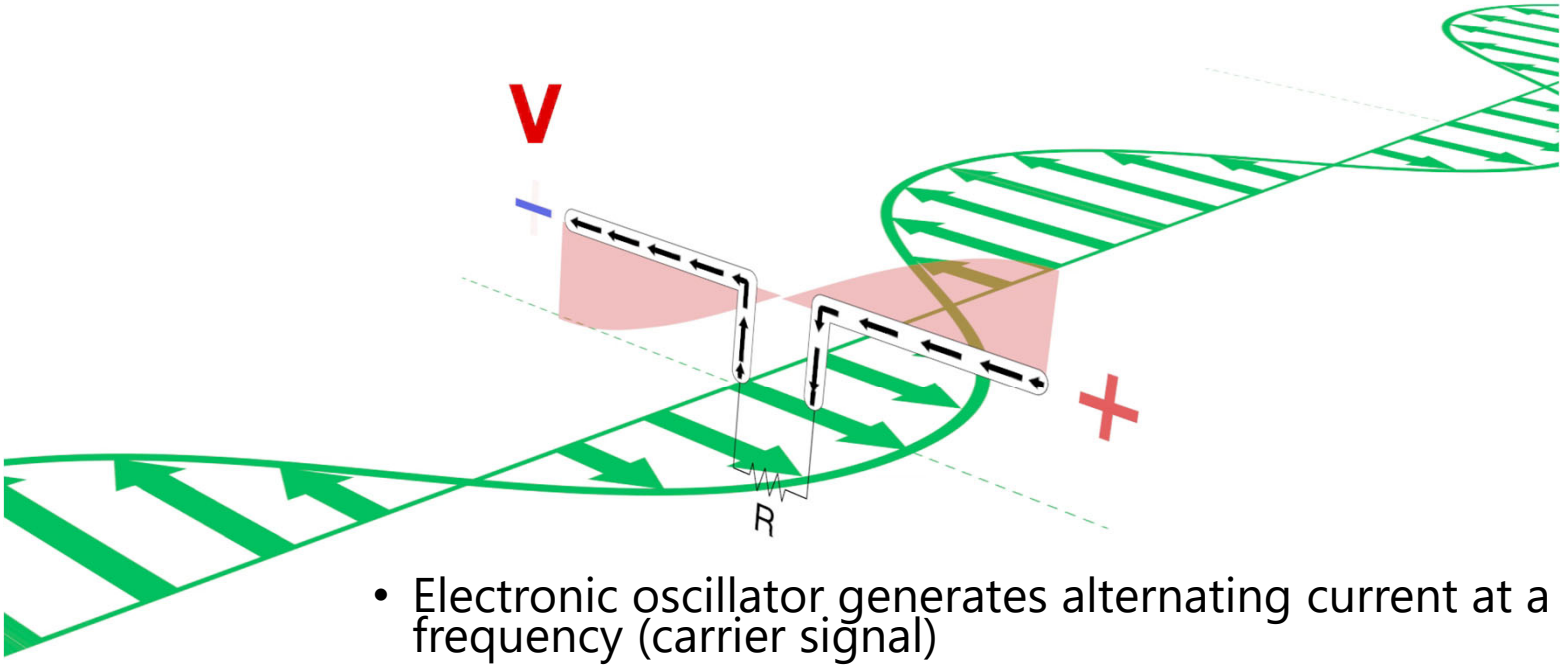
RF Transmitter Architecture



RF Transmitter Architecture



We Can Use EM Leakage to Transmit Data



- Electronic oscillator generates alternating current at a frequency (carrier signal)
  - Rate of oscillation: Wavelength of signal
  - Information is piggybacked by modulating carrier signal
- Bandpass filter used on receiver to extract information signal

Can We Use EM Waves to Communicate?

- Solution: Use low frequencies
  - Radio Frequency: 3 kHz to 300 GHz
  - Easy to generate, good propagation characteristics, relatively safe for people
- Challenge: Low bandwidth of available frequencies
  - Visible spectrum is 2 million GhZ wide!
  - Takes more time to send data
- → Need smart ways to pack information into that bandwidth
  - Modulation, channel sharing, frequency allocation

UNITED STATES FREQUENCY ALLOCATIONS

THE RADIO SPECTRUM

RADIO SERVICES COLOR LEGEND

- AERONAUTICAL MOBILE
- AERONAUTICAL MOBILE SATELLITE
- AERONAUTICAL RADIONAVIGATION
- AMATEUR
- AMATEUR SATELLITE
- BROADCASTING
- BROADCASTING SATELLITE
- EARTH EXPLORATION SATELLITE
- FIXED
- FIXED SATELLITE
- INTER-SATELLITE
- LAND MOBILE
- LAND MOBILE SATELLITE
- MARITIME MOBILE
- MARITIME MOBILE SATELLITE
- MARITIME RADIONAVIGATION
- METEOROLOGICAL AIDS
- METEOROLOGICAL SATELLITE
- MOBILE
- MOBILE SATELLITE
- RADIO ASTRONOMY
- RADIODETERMINATION SATELLITE
- RADIOLOCATION
- RADIOLOCATION SATELLITE
- RADIONAVIGATION
- RADIONAVIGATION SATELLITE
- SPACE OPERATION
- SPACE RESEARCH
- STANDARD FREQUENCY AND TIME SIGNAL
- STANDARD FREQUENCY AND TIME SIGNAL SATELLITE

ACTIVITY CODE

- GOVERNMENT EXCLUSIVE
- GOVERNMENT/NON-GOVERNMENT SHARED
- NON-GOVERNMENT EXCLUSIVE

ALLOCATION USAGE DESIGNATION

SERVICE	EXAMPLE	DESCRIPTION
Primary	FIXED	Capital Letters
Secondary	Mobile	1st Capital with lower case letters

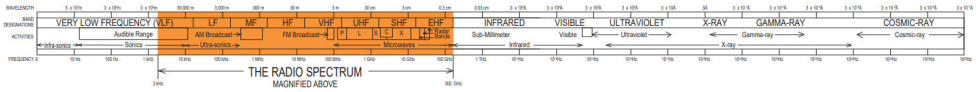
This chart is a graphic single-point-in-time portrayal of the Table of Frequency Allocations used by the FCC and NTIA. As such, it does not completely reflect all aspects, i.e., footnotes and recent changes made to the Table of Frequency Allocations. Therefore, for complete information, users should consult the Table to determine the current status of U.S. allocations.

U.S. DEPARTMENT OF COMMERCE
National Telecommunications and Information Administration
Office of Spectrum Management

October 2003

* EXCEPT AERO MOBILE (R)

** EXCEPT AERO MOBILE

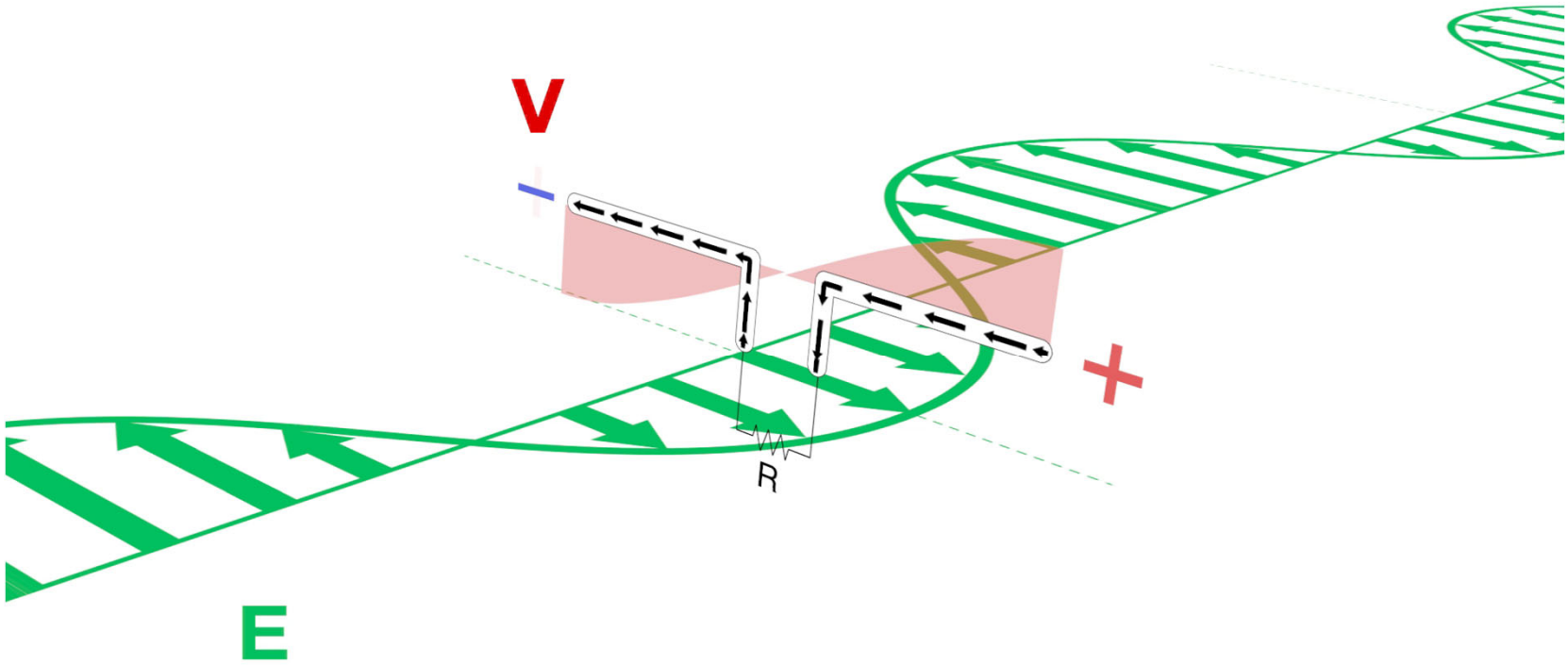


PLEASE NOTE: THE SPACING ALLOTTED THE SERVICES IN THE SPECTRUM SEGMENTS SHOWN IS NOT PROPORTIONAL TO THE ACTUAL AMOUNT OF SPECTRUM OCCUPIED.

Radio Frequency Communications

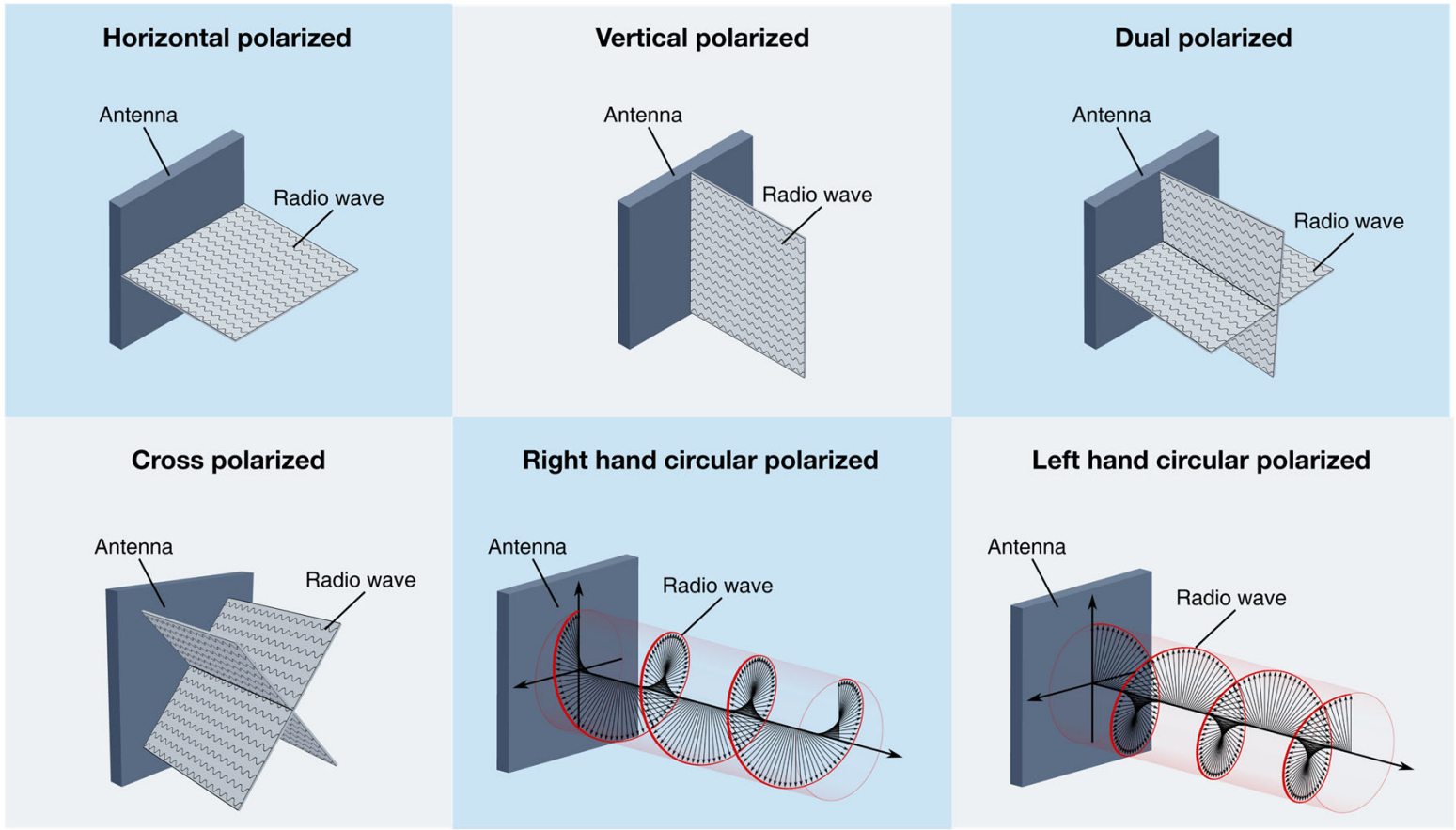
1. Electromagnetism theory
   → What is RF?
2. Antenna design
   → How can we send RF?
3. Signal propagation
   → How do wireless signals propagate?
4. Modulation
   → How do we encode data on RF?

“Polarization” of EM Waves

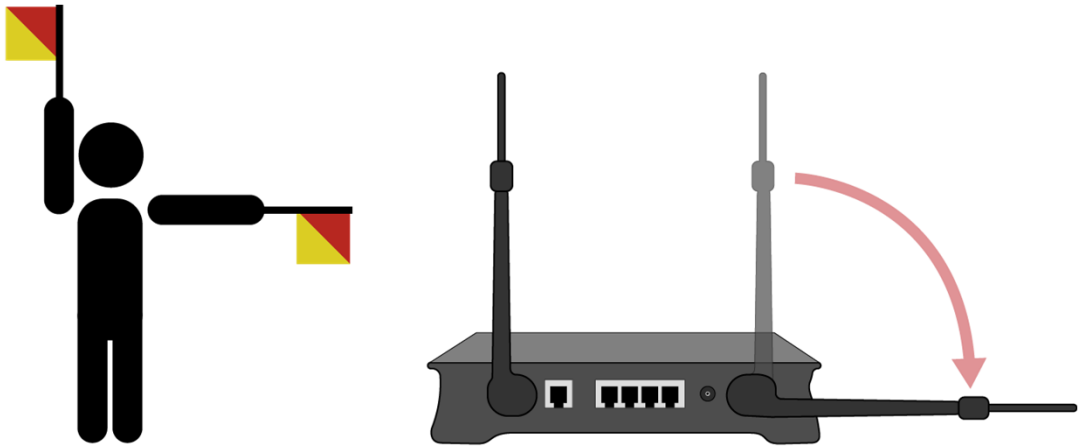


- Signals are “polarized” in direction of current flow
- Receiver antenna orientation must match that of transmitter

Polarization Options



Antenna Placement for Polarization



- You don’t know how your users are polarized → propagate signals for both
  - Some APs propagate two polarizations
  - Can manually configure antennas
- Also covers vertical and horizontal space

Radiation Patterns

Radiation Patterns

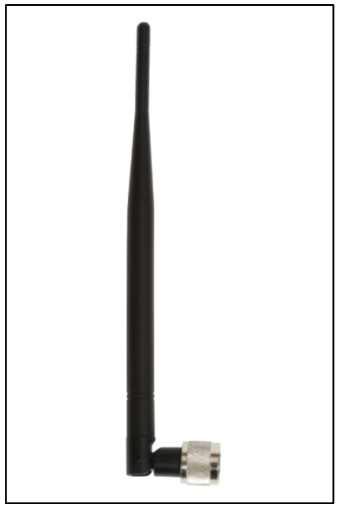
Omnidirectional

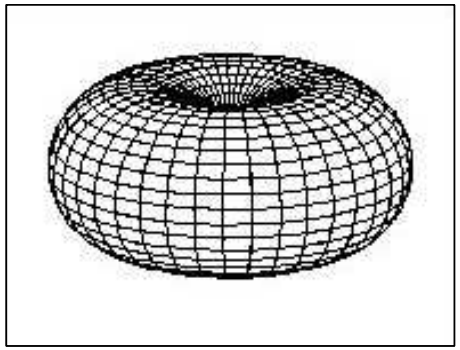
Multidirectional

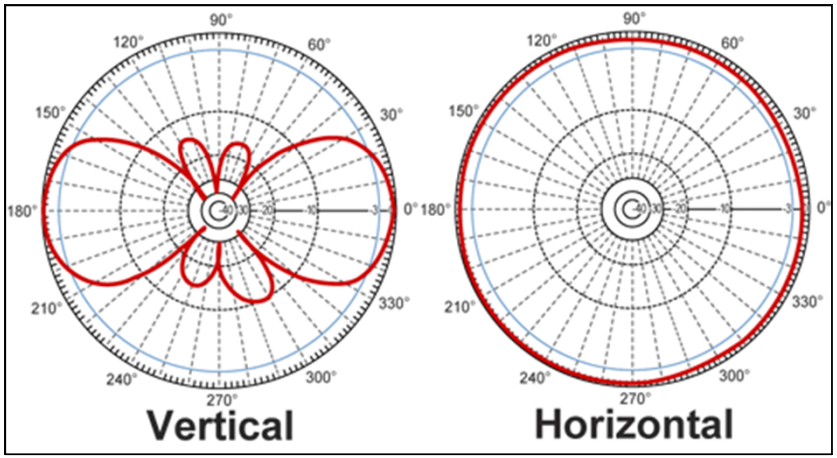
Directional

- Light sources can transmit different amounts of intensity in different directions
- Same can happen for antennas

Antenna Types: Omnidirectional

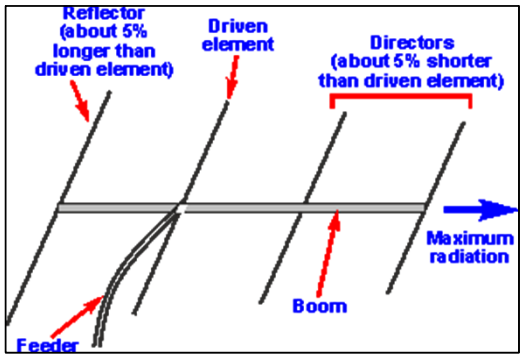


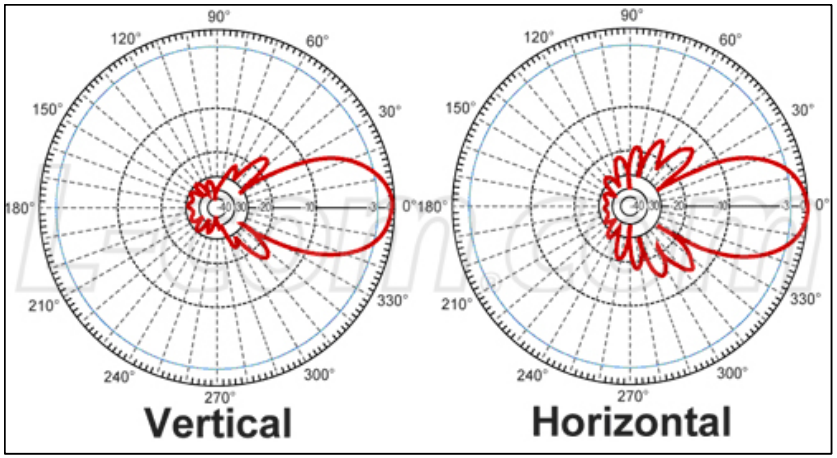




- Radiates equally in all directions in one plane
  - Power decreases above/below plane
- Widely used in APs, NICs

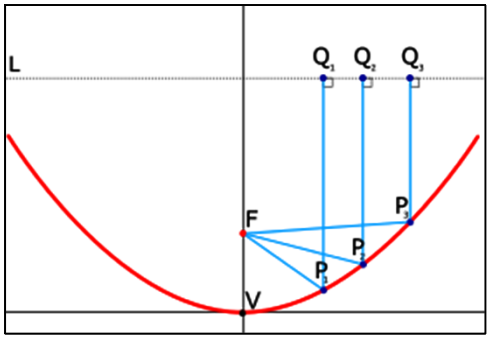
Antenna Types: Directional (Yagi)

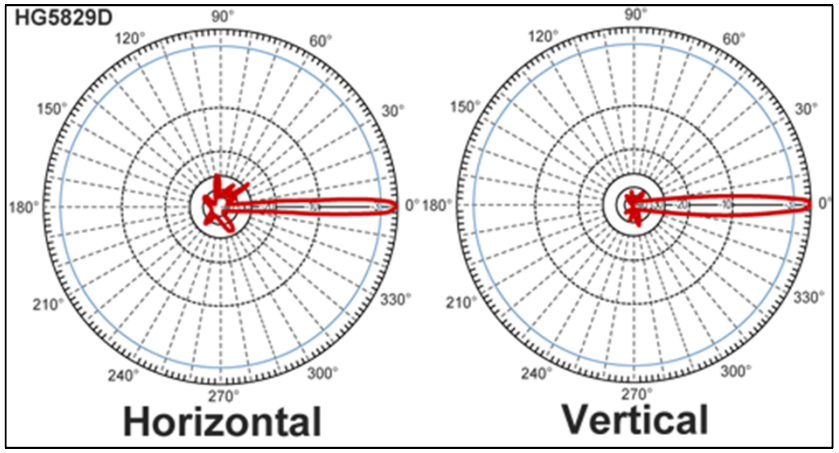




- Multiple parallel elements in a line
- Substantial increase in directionality and gain

Antenna Types: Directional (Parabolic)





- Reflector must be substantially larger than wavelength
- Metal screen reflects as well as solid dish if spaces < 1/10 wavelength

How to Choose an Antenna

- Directionality: Do you want to light up a particular area, or entire surrounding region?
- Gain: How much “reach” the antenna has compared to an omnidirectional antenna
- Bandwidth: Range of frequencies over which the antenna operates effectively

Radio Frequency Communications

1. Electromagnetism theory
   → What is RF?
2. Antenna design
   → How can we send RF?
3. Signal propagation
   → How do wireless signals propagate?
4. Modulation
   → How do we encode data on RF?

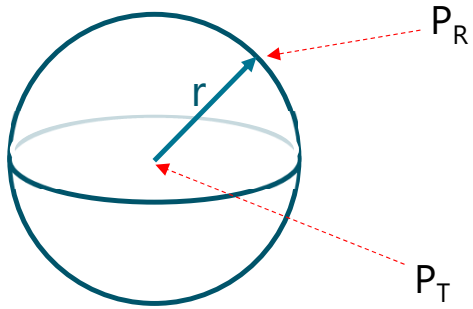
Signal Propagation

What happens to radio waves between antennas?

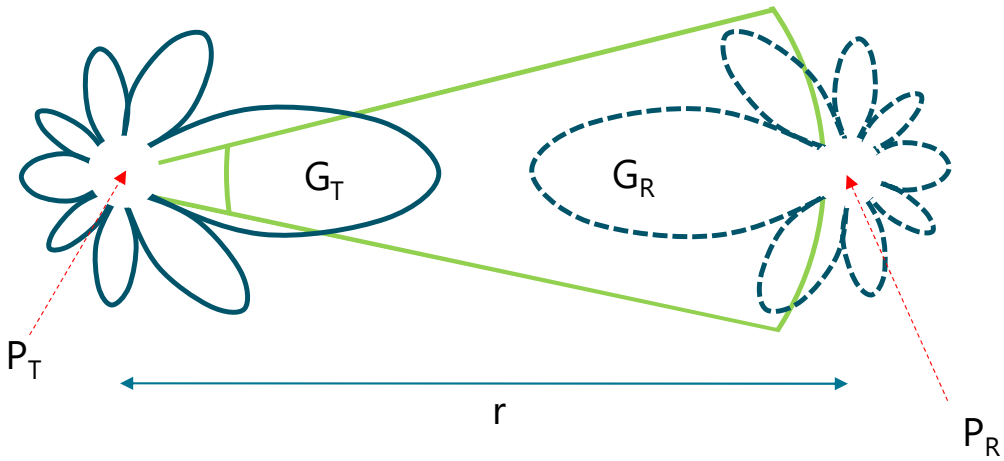
1. They lose energy as they propagate out (path loss)
2. They get deflected when they hit objects (reflection)
3. They bounce off things and then recombine (multipath/fading)

Free Space Propagation

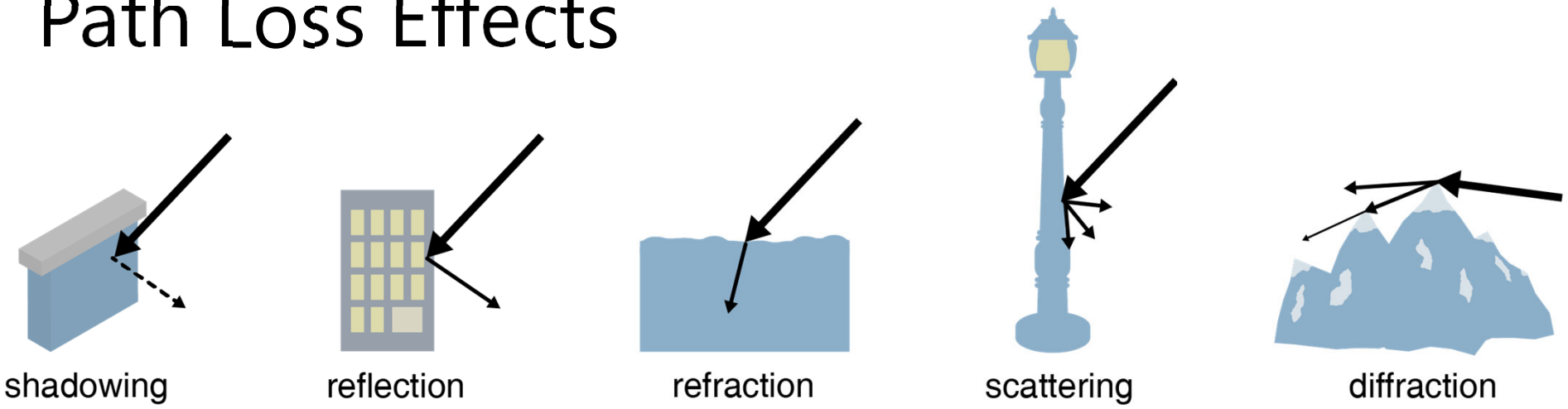
- Propagation from idealized (isotropic) antenna is a sphere
  - Power density (P_R is transmit power P_T divided by surface area)
  - $P_R = P_T / (4\pi r^2)$



- If transmitting antenna has gain G_T, receiving antenna has gain G_R
  - Then power density increases by the product of the gains
  - $P_R = G_T G_R P_T / (4\pi r^2)$
  - Known as the *Friis Transmission Equation*

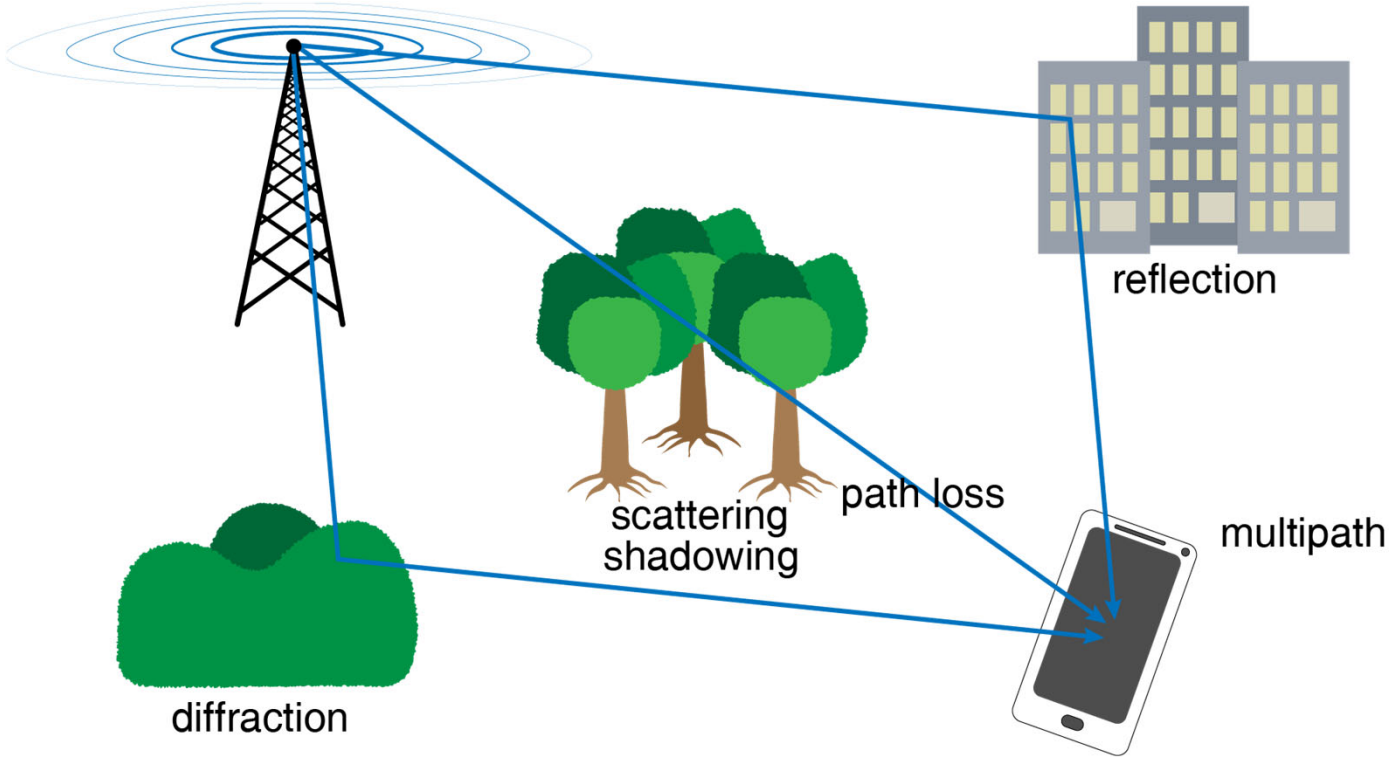


Path Loss Effects



- Wireless signals are like light
  - Just a different wavelength along EM spectrum
- They interact with physical objects in similar ways
  - Can be bad – can degrade signal
  - Can be good – can leverage this to improve signal strength/range

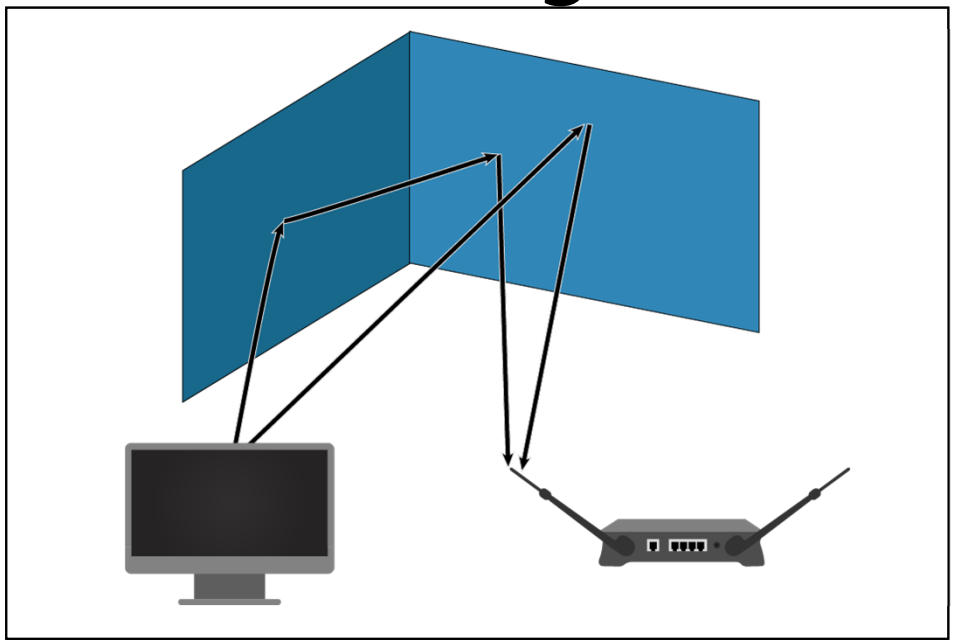
Path Loss Effects

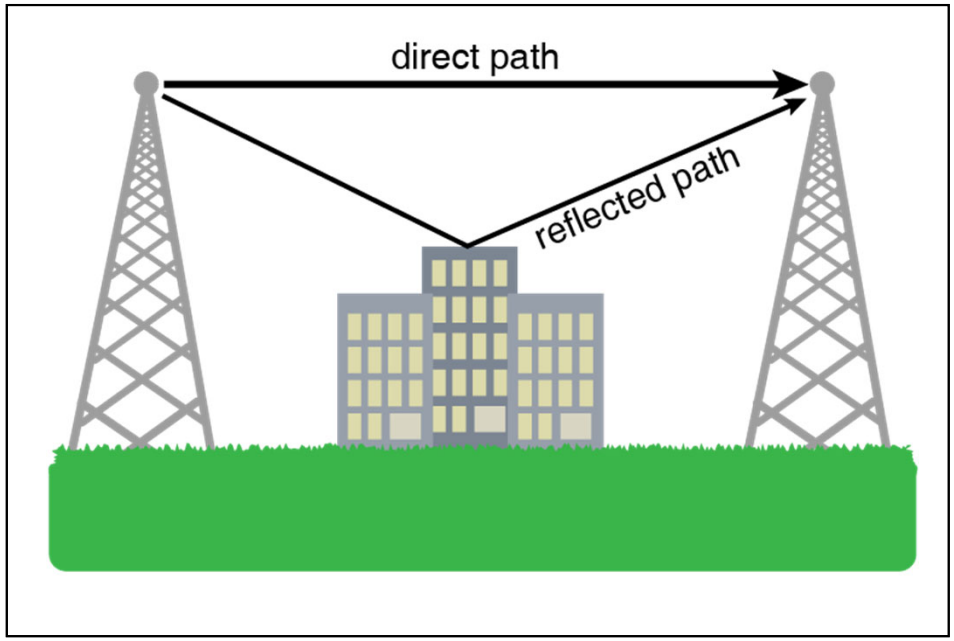


Wireless Signals Propagate Like Waves

- Propagate at speed of light in vacuum
  - Propagation speed independent of wavelength
- Matching wavelengths construct, opposing wavelengths destruct

Wireless Signals Can Self-Interfere



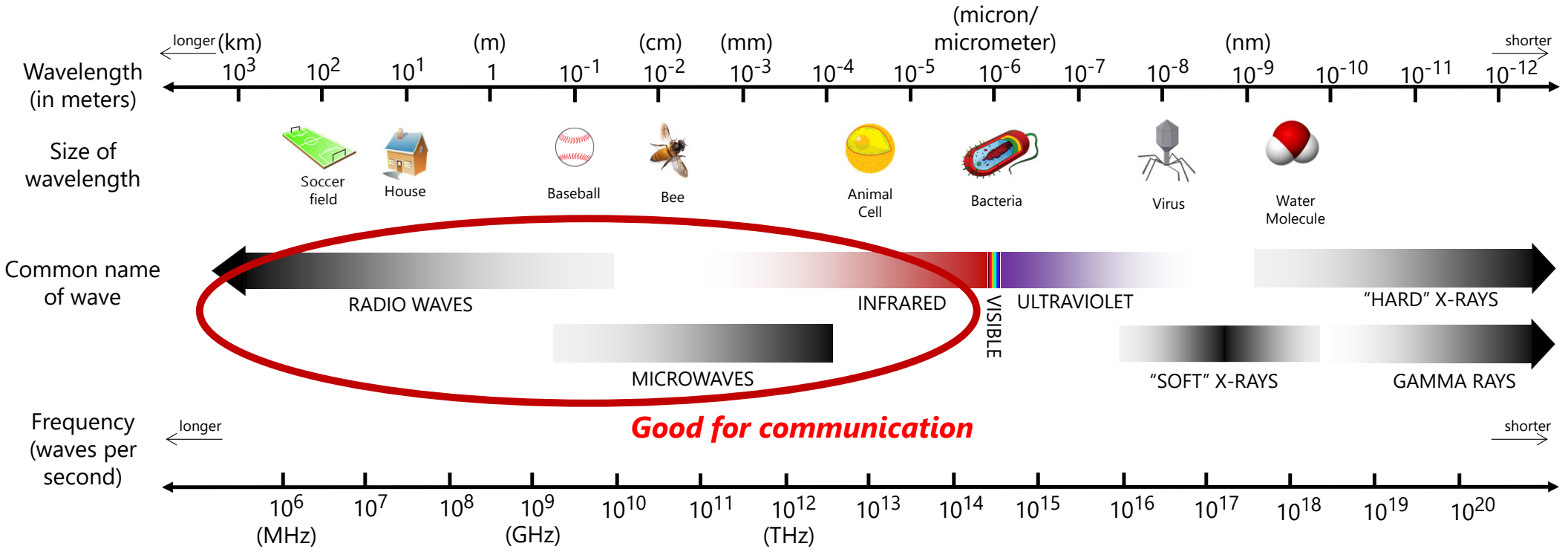


- Could be a good thing, if they construct
- Could be a bad thing, if they destruct

Frequency Affects Propagation

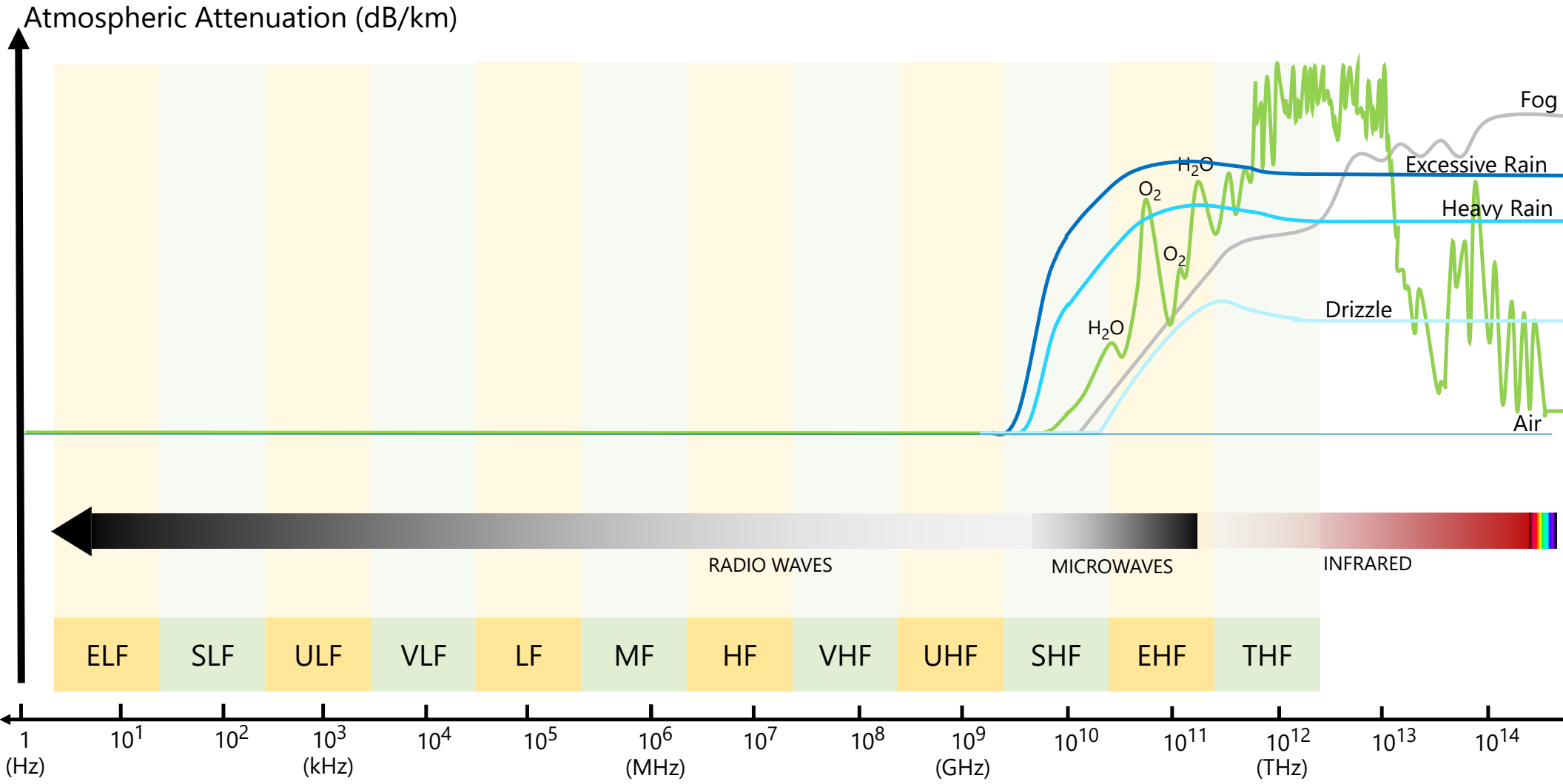
- **Low frequencies (long wavelengths):**
  - Diffract around objects, can follow curvature of earth
  - Can penetrate through objects (e.g., ground, water)
  - Require bigger antennas
- **High frequencies (short wavelengths):**
  - Act more like light
  - Can be “focused” (parabolic antennas)
  - Attenuate more in air; increased scattering by objects, rain

The Electromagnetic Spectrum

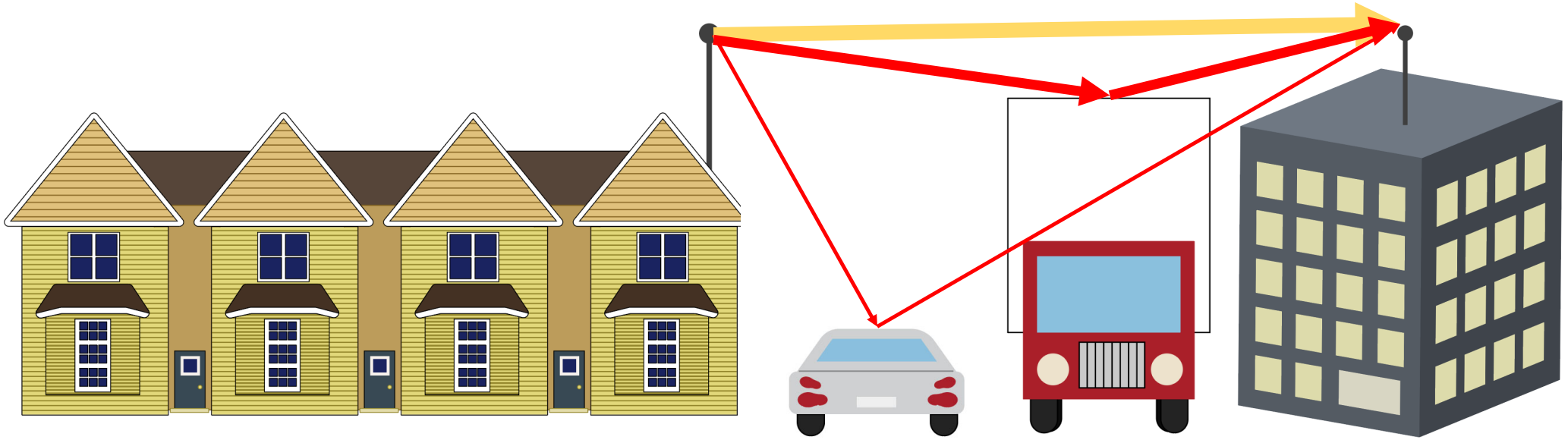


Good for communication

- Radio frequency: 3kHz to 300GHz
  - Easy to generate, good propagation characteristics, relatively safe for people

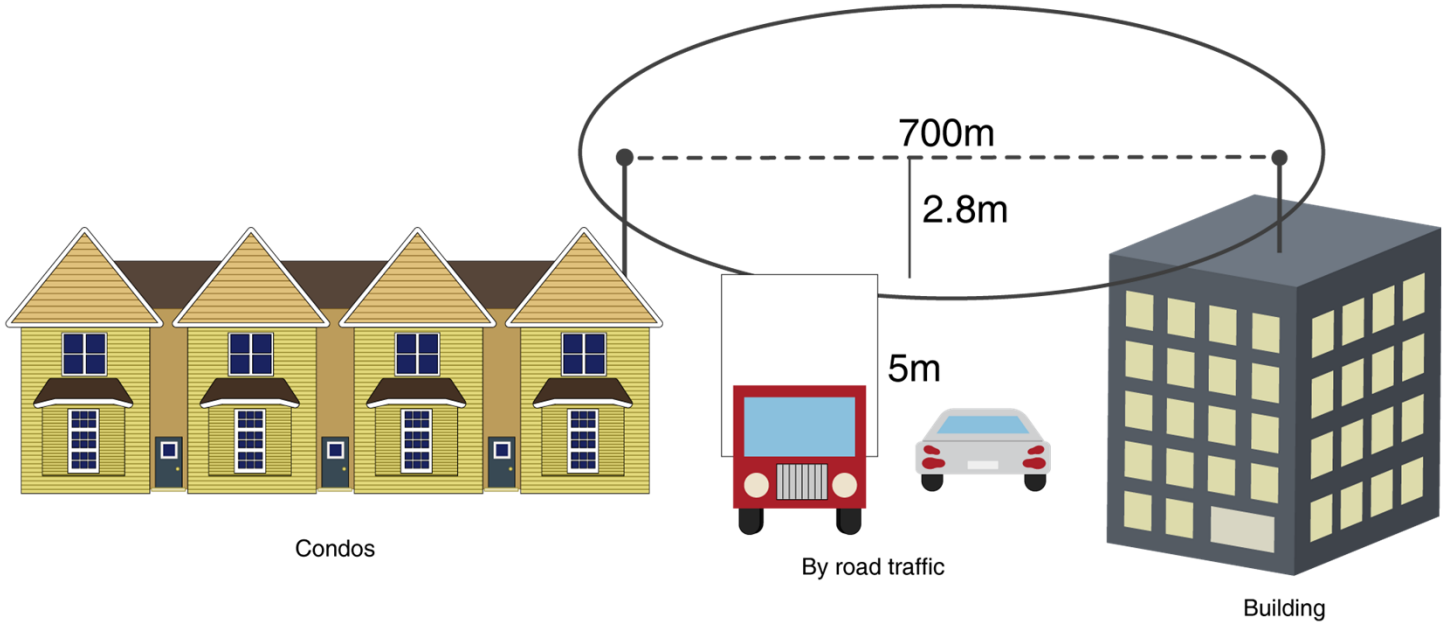

Atmospheric Attenuation (dB/km)
Fog
Excessive Rain
Heavy Rain
Drizzle
Air
H_2O
O_2
O_2
H_2O
RADIO WAVES
MICROWAVES
INFRARED
ELF
SLF
ULF
VLF
LF
MF
HF
VHF
UHF
SHF
EHF
THF
1 (Hz)
10^1
10^2
10^3 (kHz)
10^4
10^5
10^6 (MHz)
10^7
10^8
10^9 (GHz)
10^{10}
10^{11}
10^{12} (THz)
10^{13}
10^{14}

Interference Changes with Distance from Line-of-Sight Path



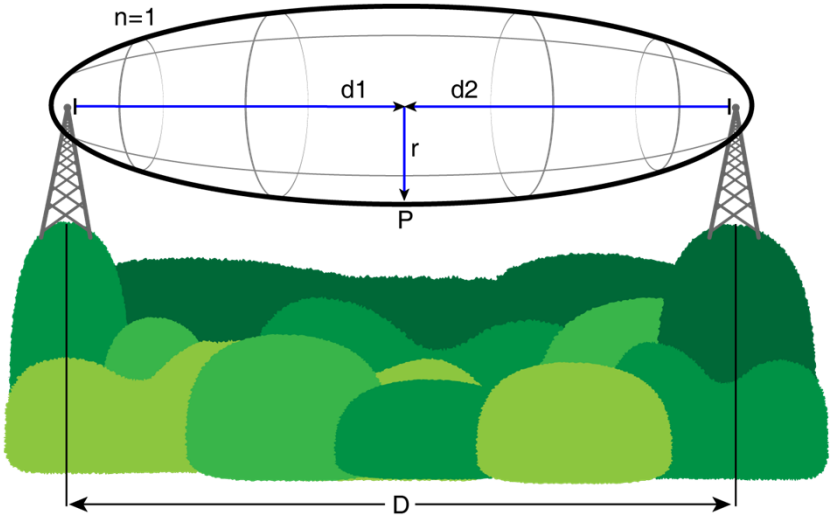
- Vertical distance decreases signal strength of secondary path
- Increasing vertical distance alternates between constructive and destructive interference

Fresnel Zone



- Ellipsoid drawn between transmitter and receiver should be kept clear from obstacles
  - Heuristic: Max obstruction <40%, recommended <20%

Fresnel Zone Radius Computation



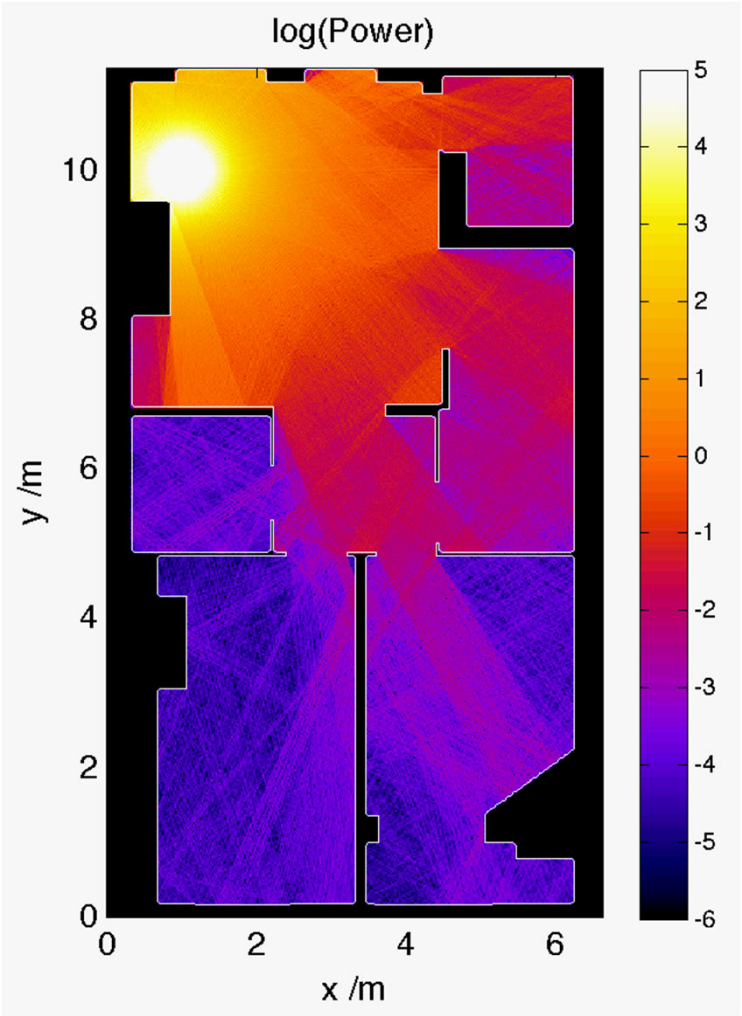
$$F_n = \sqrt{\frac{n\lambda d_1 d_2}{d_1 + d_2}}, \quad d_1, d_2 \gg n\lambda$$

F_n is the nth Fresnel zone radius,
d_1 is the distance of P from one end,
d_2 is the distance of P from the other end,
λ is the wavelength of the transmitted signal.

- Calculators available online
  - Google for “freznel zone calculator”

How Physical Objects Affect Wireless Signals

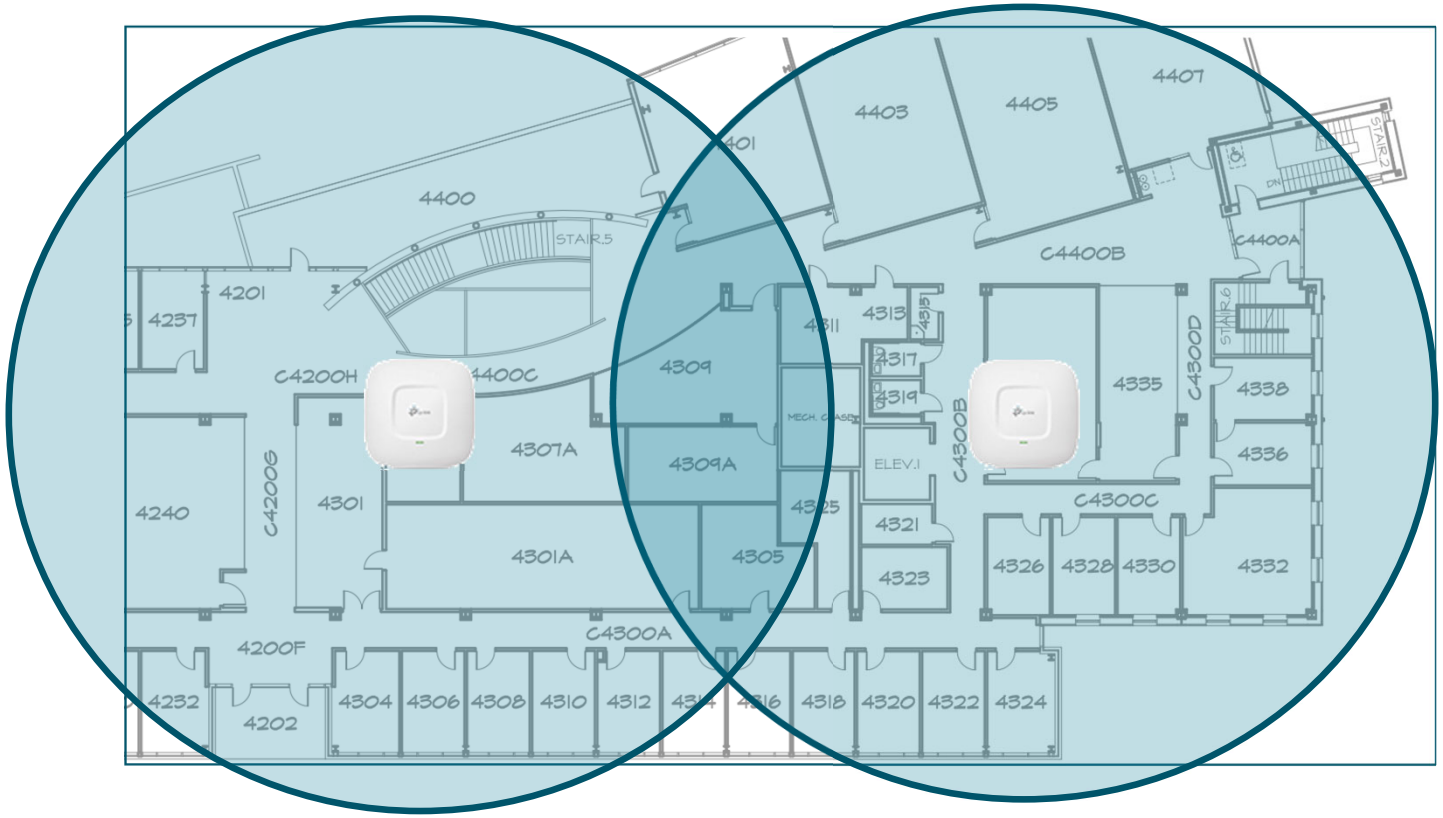
- Different materials affect reflection/absorption/ refraction in different ways
- Conduct wireless surveys to understand propagation in your environment



Is attenuation good or bad?

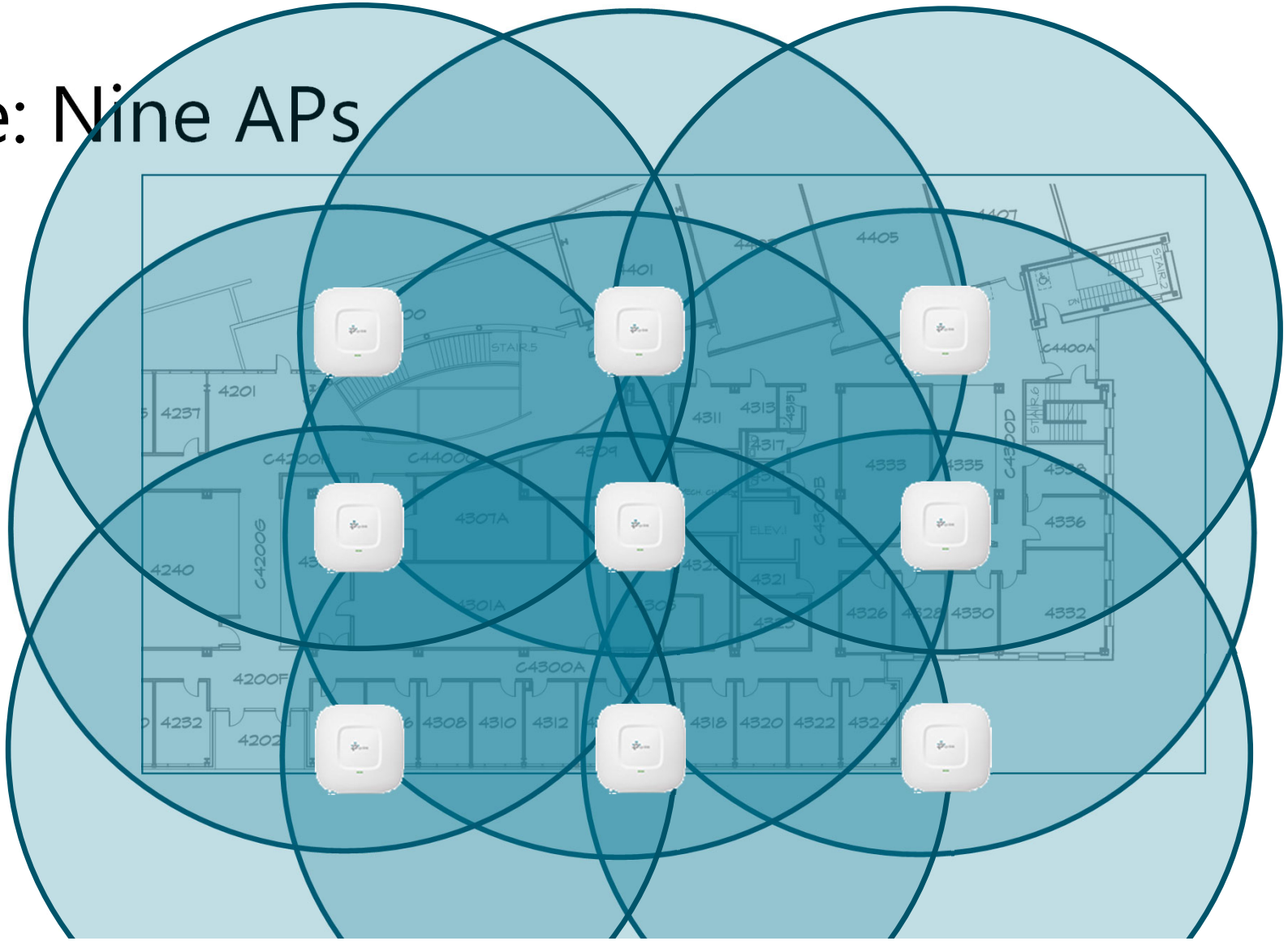
Example: Two APs

- What if not enough capacity?



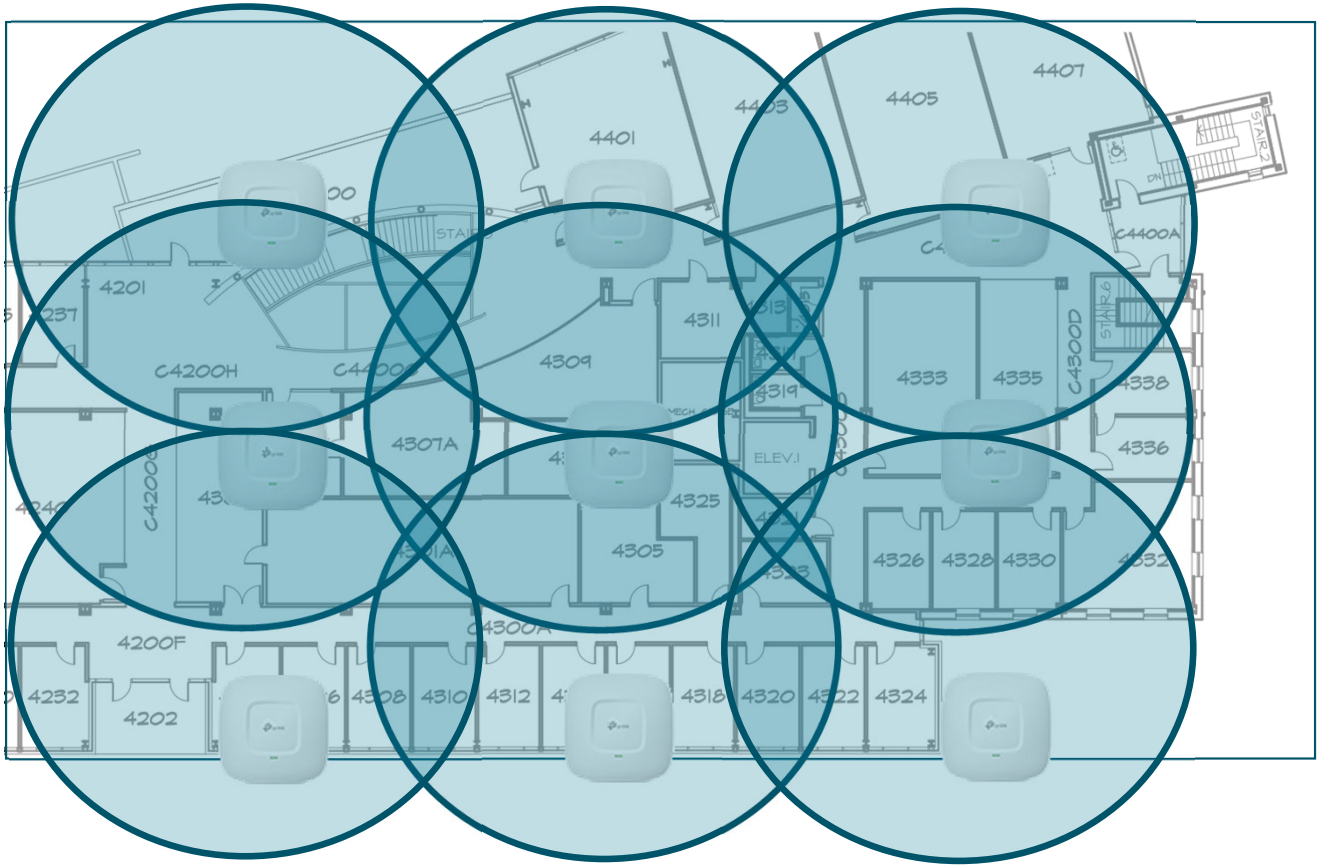
Example: Nine APs

- Problem: Increased transmitter density lowers available bandwidth

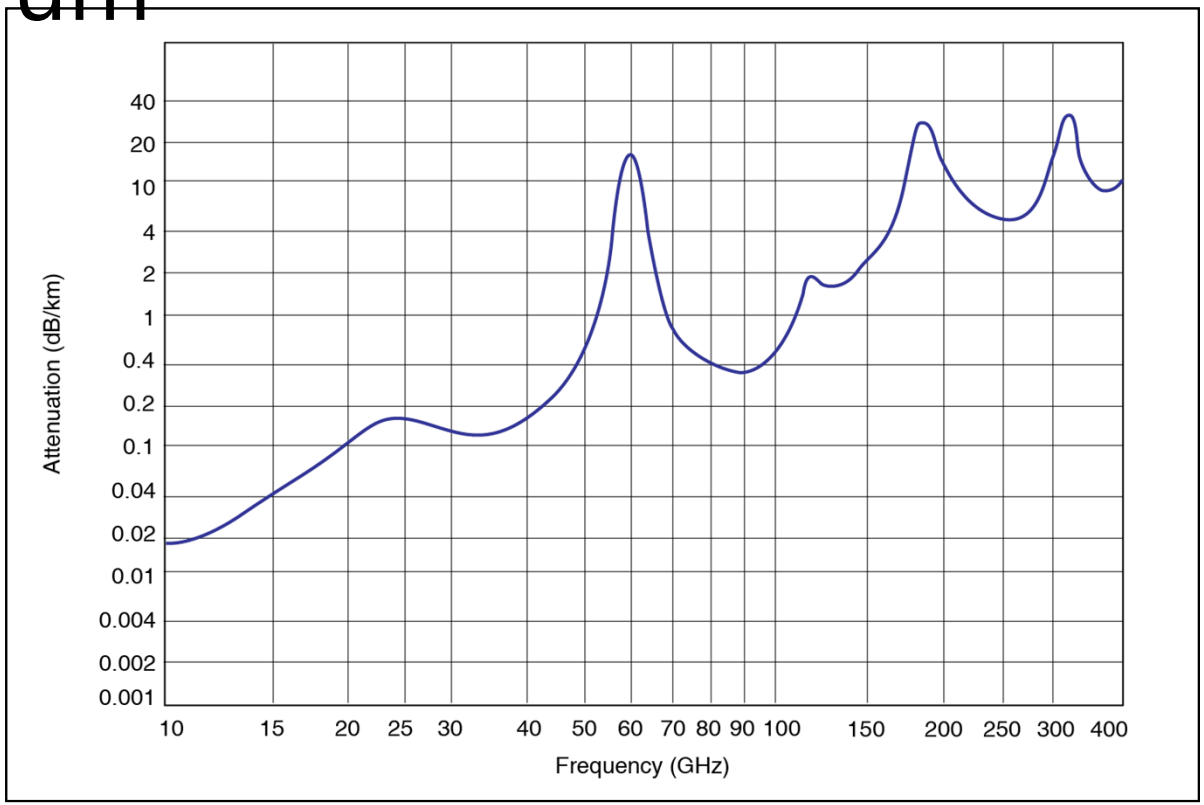


Example: Nine APs

- More attenuation better for dense deployments

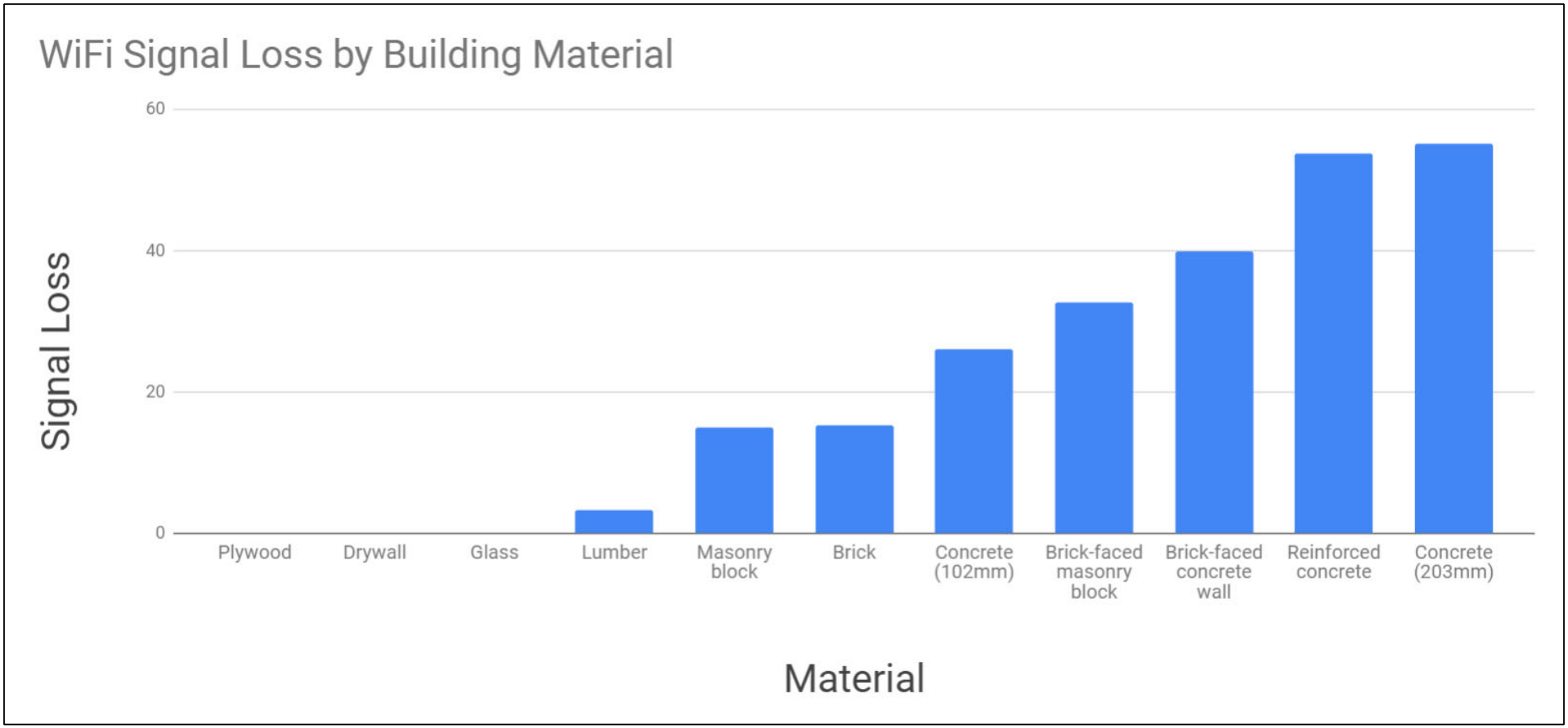


Atmospheric Attenuation in Wifi Spectrum



- 802.11ad: 60 GHz chosen because of increased atmospheric attenuation

How Physical Objects Affect Wireless Signals

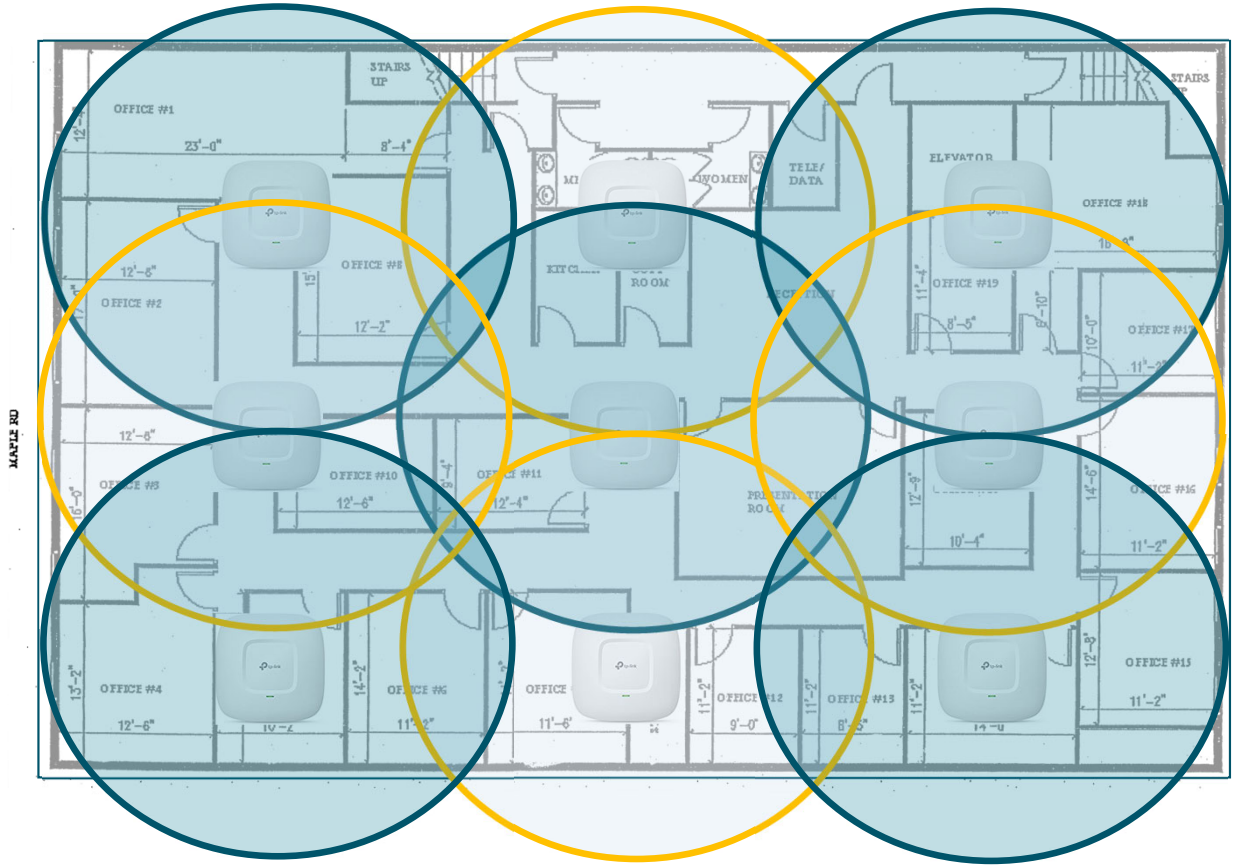


Attenuation from Water

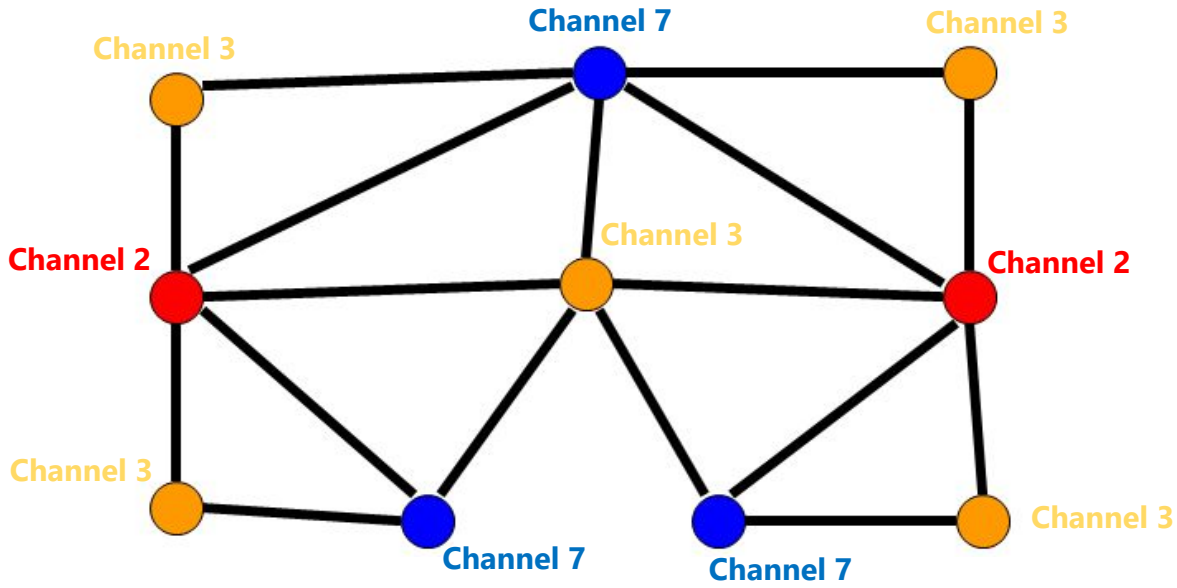
- Snow, ice, lakes, aquariums, plumbing, animals, etc.
- 2.4GHz attenuation in water to 1/3 signal strength:
  - Pure water: 8.01Km
  - Drinking water: 44.54m
  - Sea water: 8.91mm
- Humans similar to sea water
  - More fat→more attenuation
- Trees: Leaves and rain increase attenuation

Increased Capacity with Spectrum Division

- 802.11 supports different “channels”/ wavelengths
- Put nearby APs on different wavelengths



General Problem: Graph Coloring



- Vertices are APs, edges denote reachability
- Number of channels/colors needed a function of graph connectivity

Rules of Thumb for Transmitter Deployment

- Try to use the same SSID when possible, to enable automated roaming to the highest-quality AP
- APs that use the same channel should be installed as far away from each other as possible to minimize interference
- AP cell size should overlap by 15-25% to ensure that there are no gaps in coverage and to ensure a roaming client will always have a connection visible

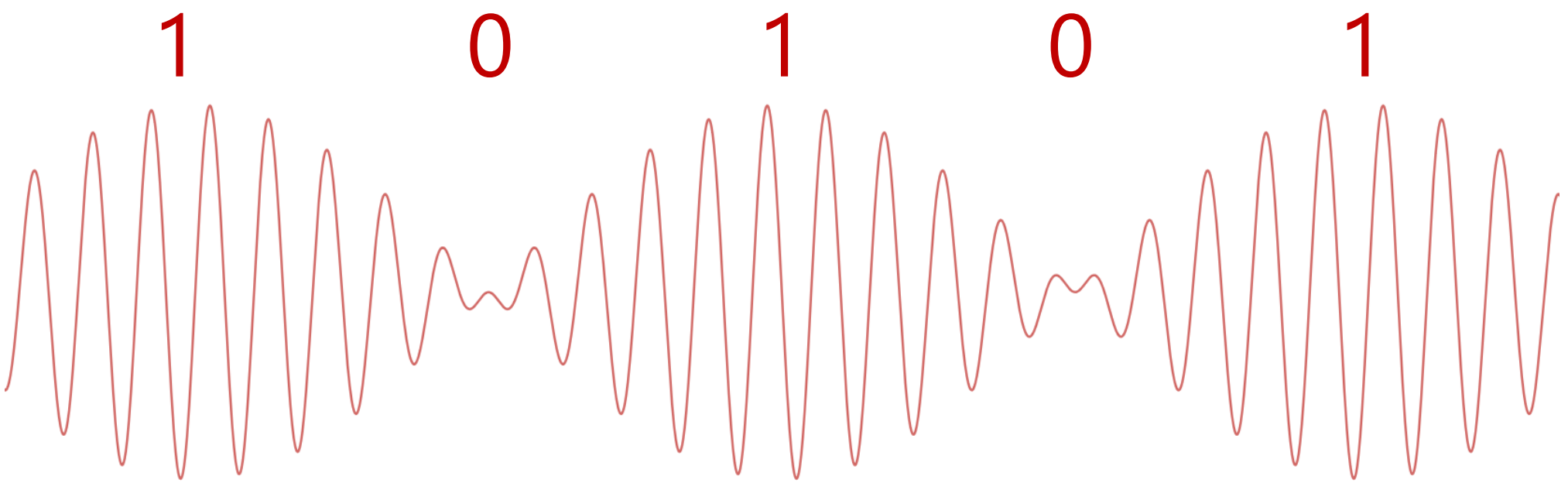
Radio Frequency Communications

1. Electromagnetism theory
   → What is RF?
2. Antenna design
   → How can we send RF?
3. Signal propagation
   → How do wireless signals propagate?
4. Modulation
   → How do we encode data on RF?

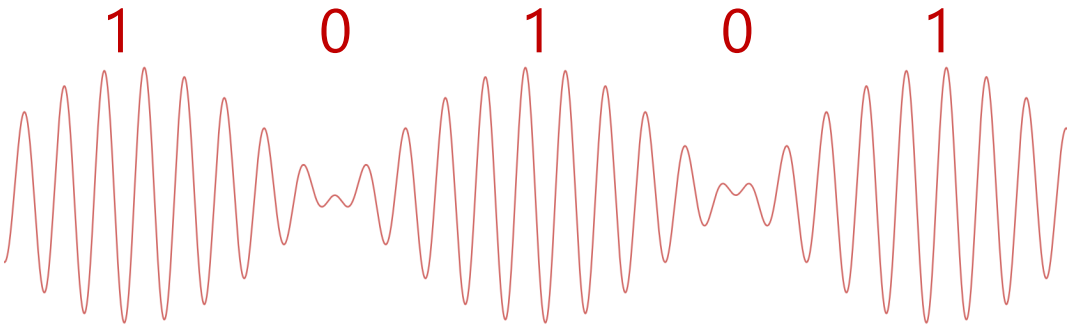
How do we encode data in wireless signals?

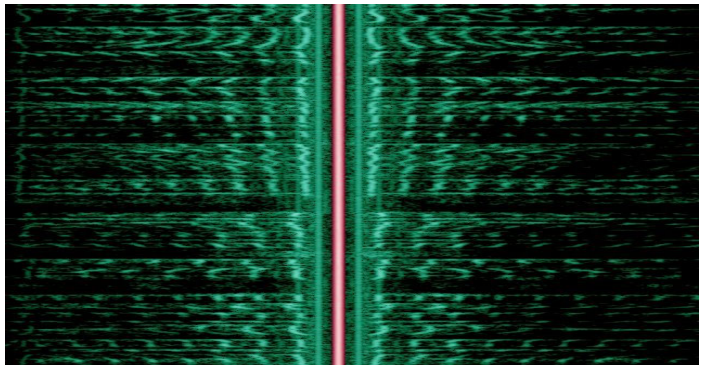
- Need to generate signals to transmit data
- Challenges:
  - Low bandwidth
  - Propagation channel impairments (noise, multipath, interference, etc.)
  - Simplicity/cost of hardware
- Remember that we send using a carrier signal
- Options for modulation: Amplitude, frequency, phase

Amplitude Shift Keying (ASK)



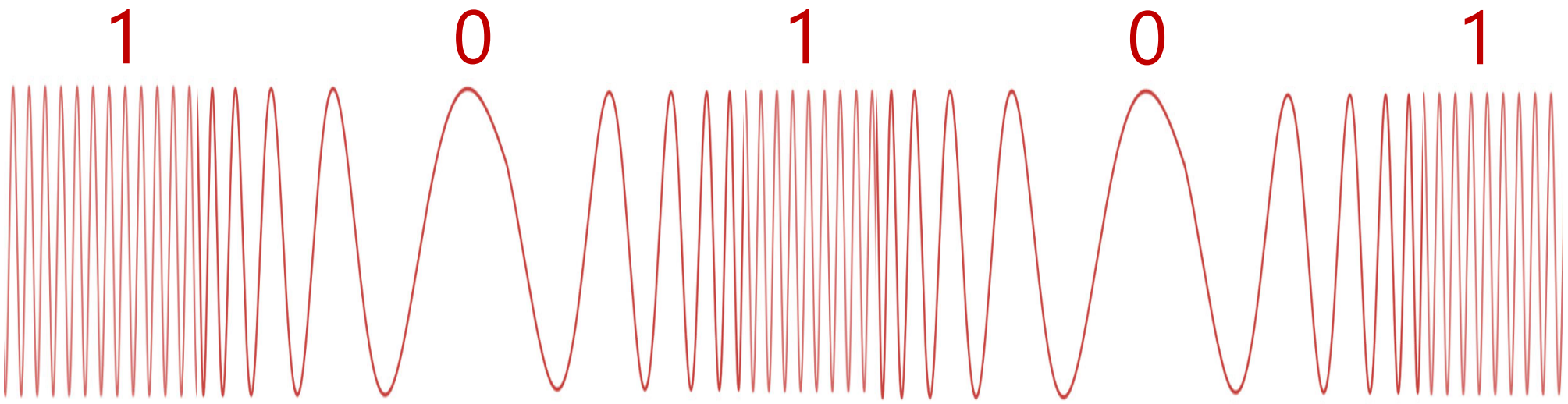
Amplitude Shift Keying (ASK)



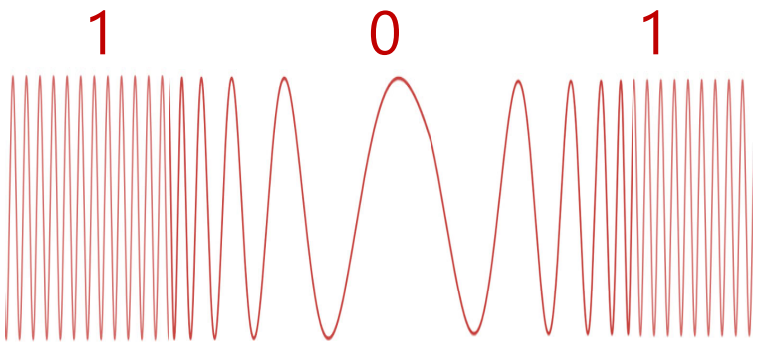


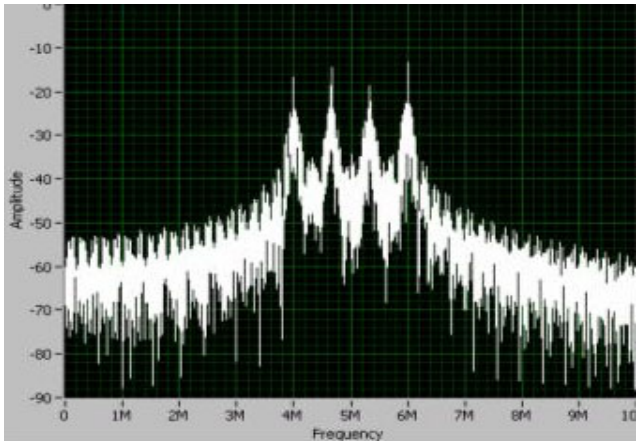
- Vary signal strength to transmit information
  - Ones/zeros represented by presence/absence of carrier
  - Modulating amplitude creates “sidebands”
  - Optimizations – carrier suppression, single-sideband suppression
- Pros: Cheap receivers (envelope detection)
- Cons: Amplifies noise, inefficient use of bandwidth and power usage

Frequency Shift Keying (FSK)



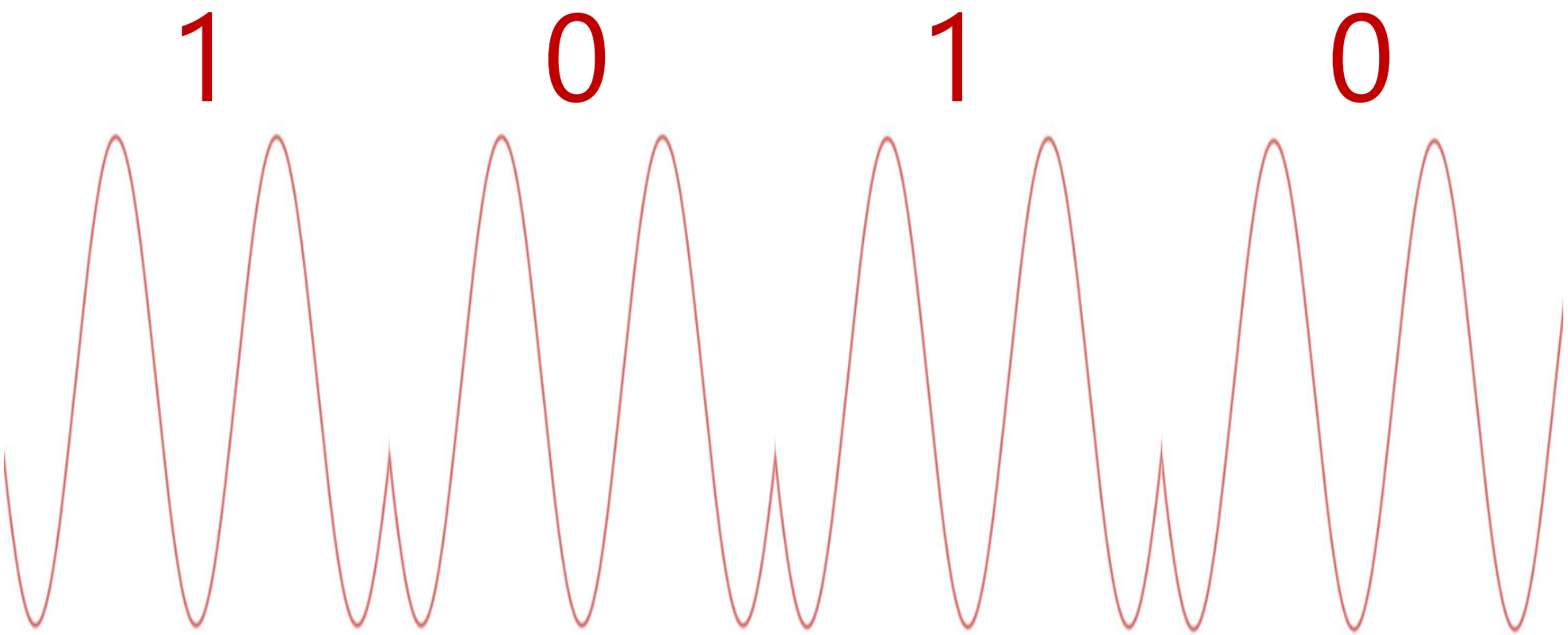
Frequency Shift Keying (FSK)



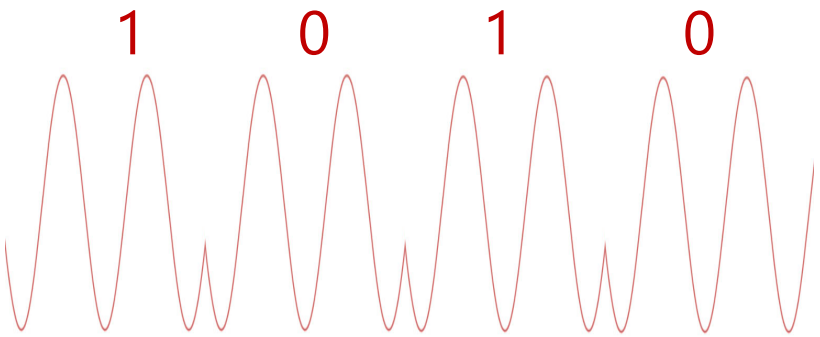


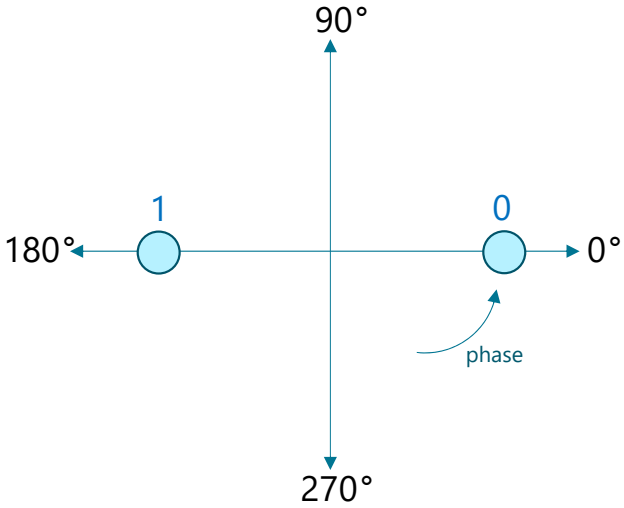
- Uses two carrier frequencies to represent binary 1 and 0
- Optimizations: Multiple FSK (groups of bits represented by multiple frequency levels)
- Pros: Higher immunity of noise, robust to variation in attenuation, simple implementation, “capture effect” completely suppresses weaker signals
- Cons: Larger bandwidth compared to ASK and PSK

Phase Shift Keying (PSK)



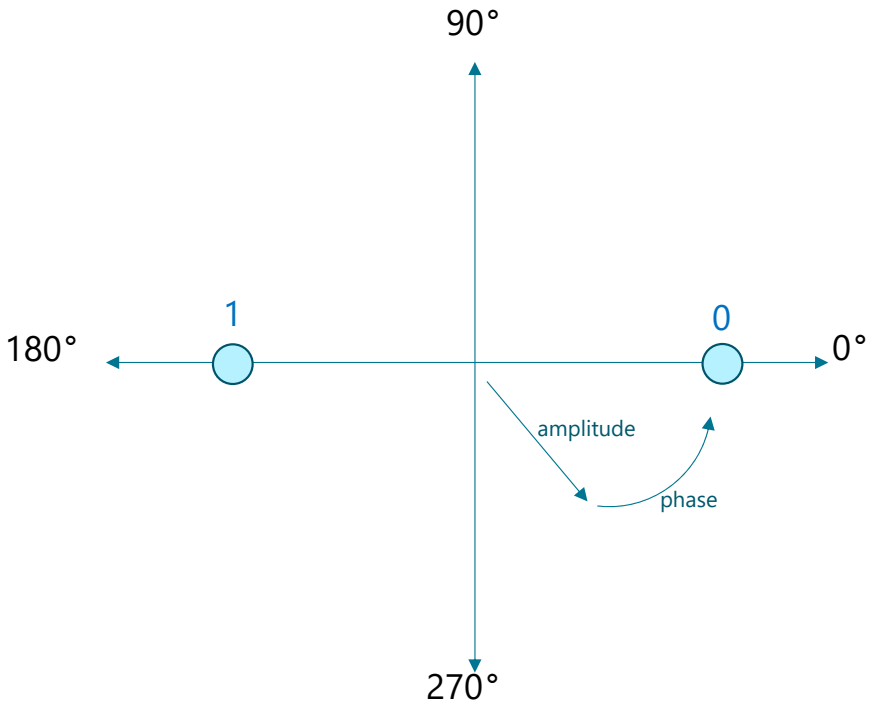
Phase Shift Keying (PSK)



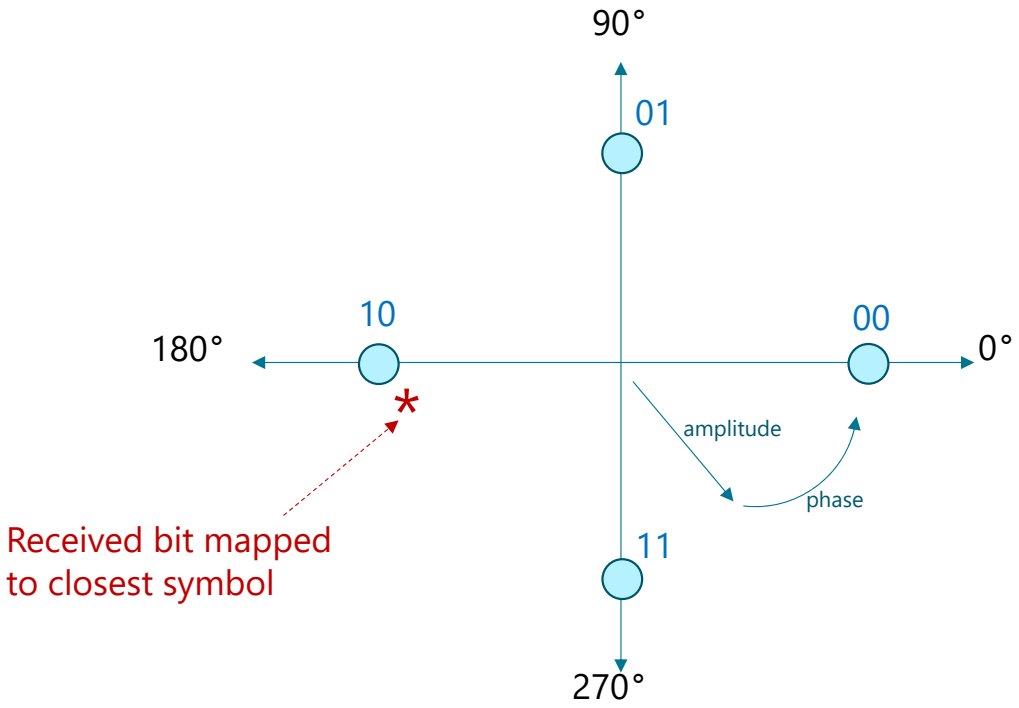


- Changes phase of carrier to convey data
- Pros: Power efficient, less susceptible to errors than ASK
- Cons: Can be complicated to implement, worse noise rejection than FSK

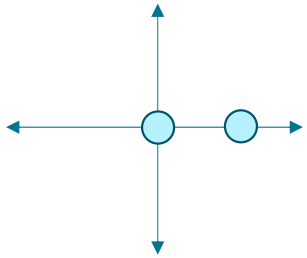
Constellation Diagram



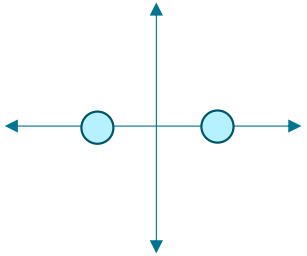
Constellation Diagram



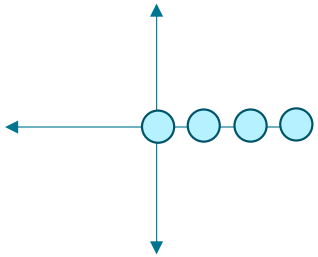
Common Phase-Amplitude Modulations



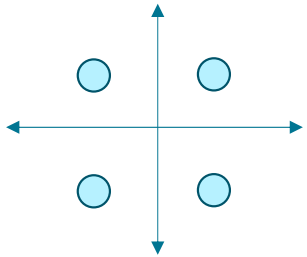
On-Off Keying
(OOK)



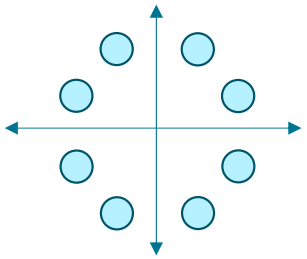
Binary Phase-Shift Keying
(BPSK)



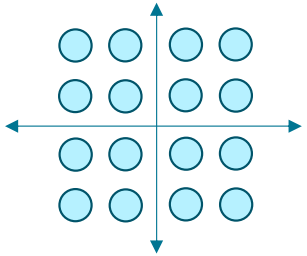
4-ASK



Quadrature Phase
Shift Keying
(QPSK)



8-PSK



16-Quadrature
Amplitude Modulation
(16-QAM)

What do real IoT protocols use?

				
	NFC	Bluetooth	ZigBee	802.11ac
Modulation	ASK	FSK	QPSK	BPSK, QPSK, QAM
Frequencies	13.56MHz	2.4GHz	2.4GHz	5GHz
Bandwidth	424 Kbps	1-3 Mbps	250 Kbps	1.3 Gbps

Wireless/IoT Protocols

HaLow

Modulation	BPSK, QPSK, QAM	FSK, PSK	QAM, QPSK	QPSK, 16QAM, 64QAM
Frequencies	900MHz	868MHz, 915MHz	750, 850, 1700, 1900, 2300 MHz	700, 1700, 1900 MHz
Bandwidth	347 Mbps	27-50Kbps	3.1 Mbps down, 1.8 Mbps up	5-12 Mbps down, 2-5 Mbps up

Wireless/IoT Protocols

	Near-Field Communication (NFC)	Bluetooth	Zigbee	802.11ac (WiFi)	CDMA (Verizon)	4G LTE (AT&T)
Modulation	ASK	FSK	QPSK	BPSK, QPSK, QAM	QAM and QPSK	QPSK, 16QAM, 64QAM
Frequency bands	13.56MHz	2.4GHz	2.4GHz	5GHz	750, 850, 1700, 1900, 2300 MHz	700, 1700, 1900 MHz
Bit rate	424 Kbps	1-3 Mbps	250 Kbps	1.3 Gbps	3.1 Mbps down, 1.8 Mbps up	5-12 Mbps down, 2-5 Mbps up

Wireless MAC

Wireless MAC

1. Collision detection and resolution
   → What if two nodes send at same time?

2. Power-saving algorithms
   → What if a node transmits but the destination is sleeping?

3. Association and neighbor discovery
   → Who is out there?

How can wireless hosts share the spectrum?

- What happens when two hosts transmit at the same time?
  - Collision
- What should we do about collisions?

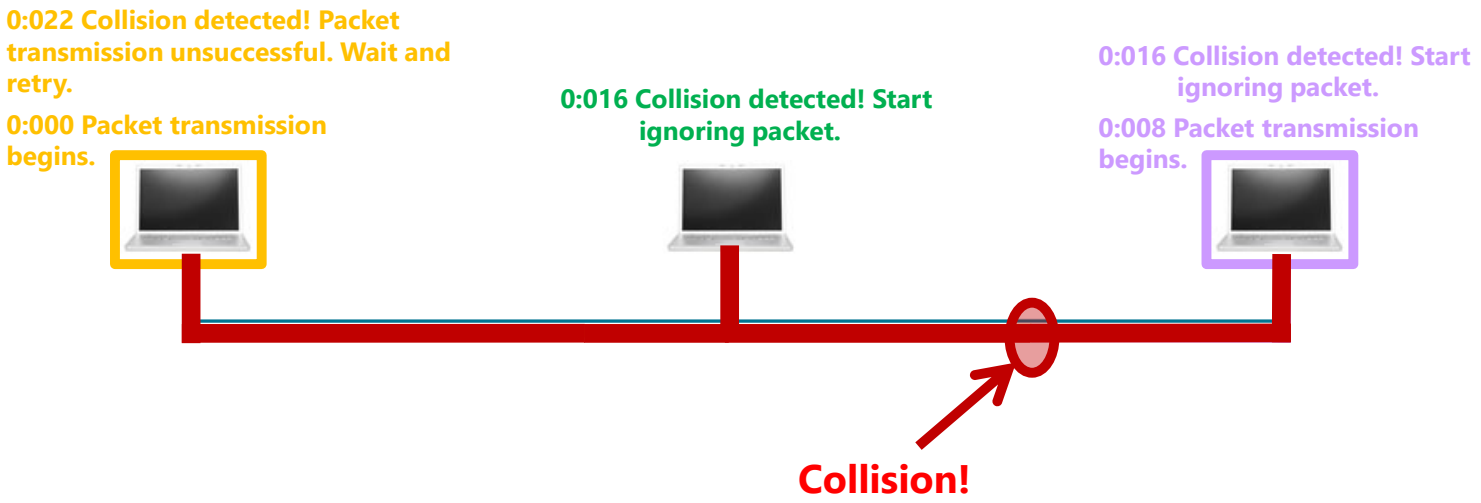
How to mitigate collisions?

1. **Just let transmissions collide**
   - Problem: Data gets corrupted

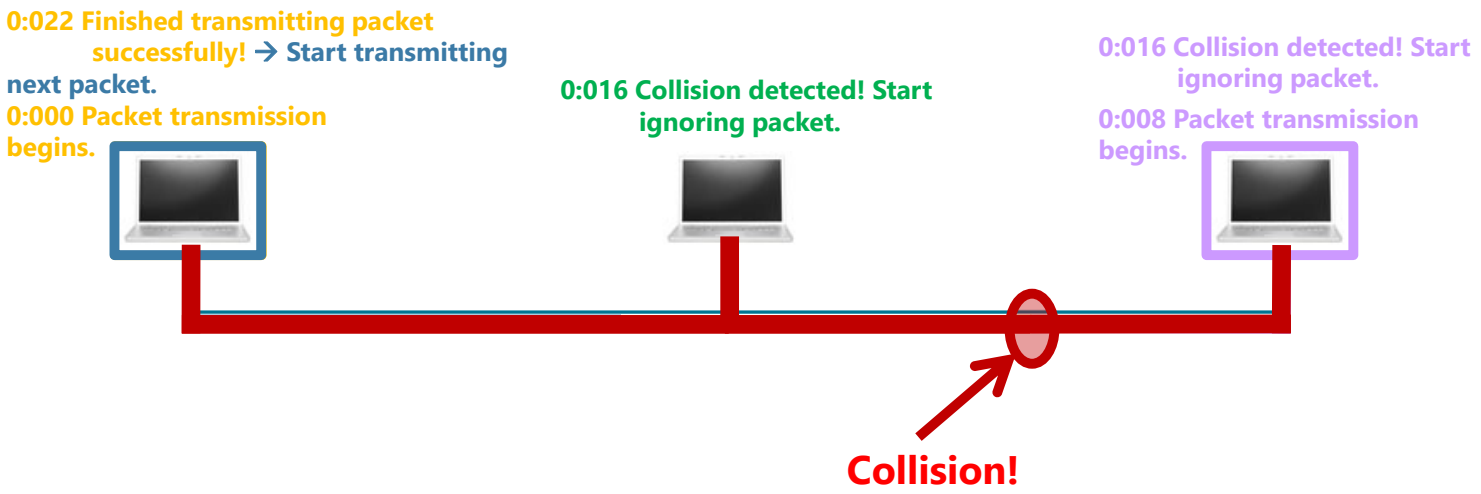
2. **Collision detection (CD)**
   - Let transmissions collide, but detect collisions
   - If sender knows their packet collided, they can retransmit

3. **Collision avoidance (CA)**
   - Try to prevent collisions

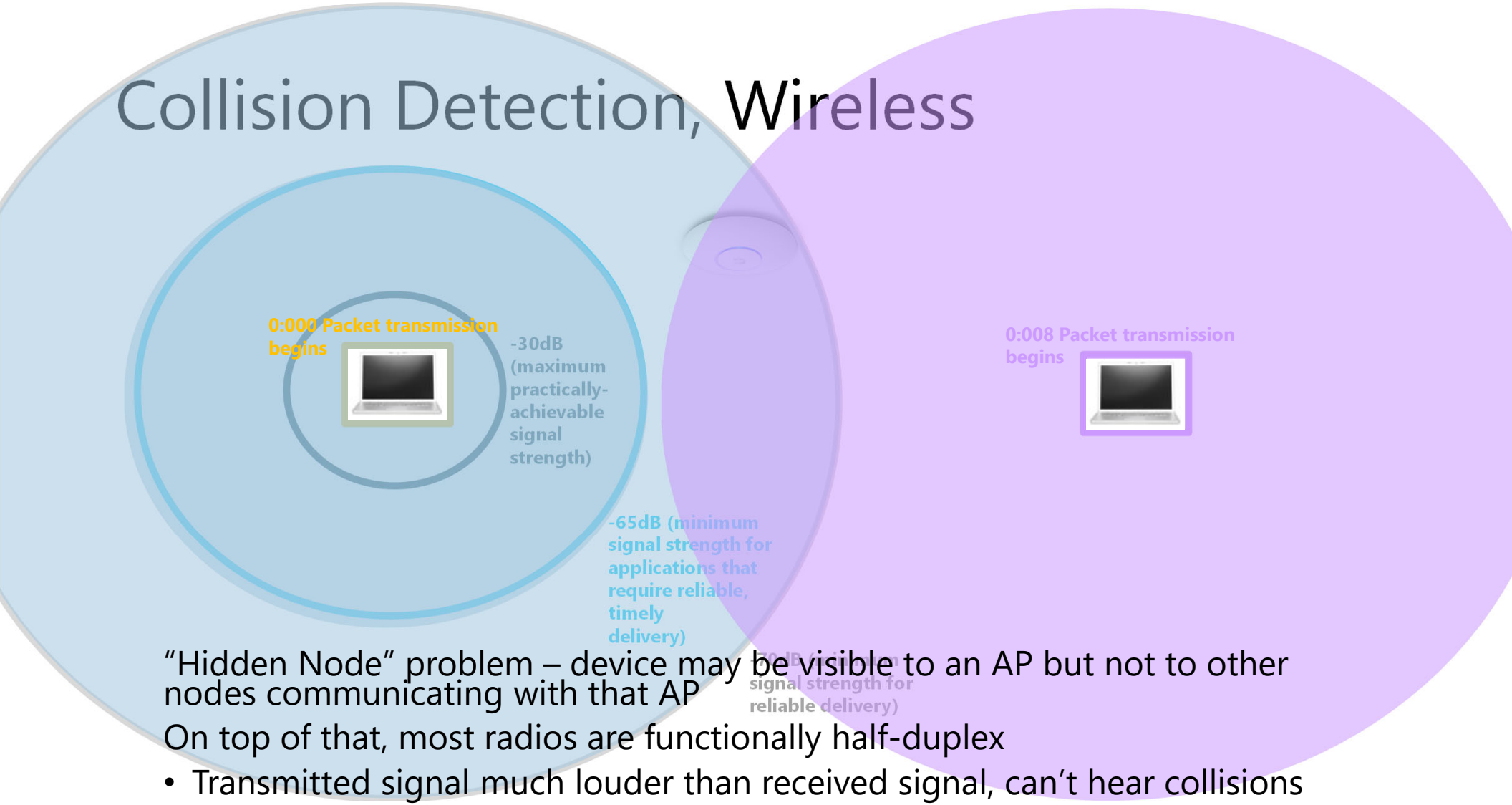
Collision Detection (CD): Wired Networks



Collision Detection (Wired, Small Packets)



Collision Detection, Wireless

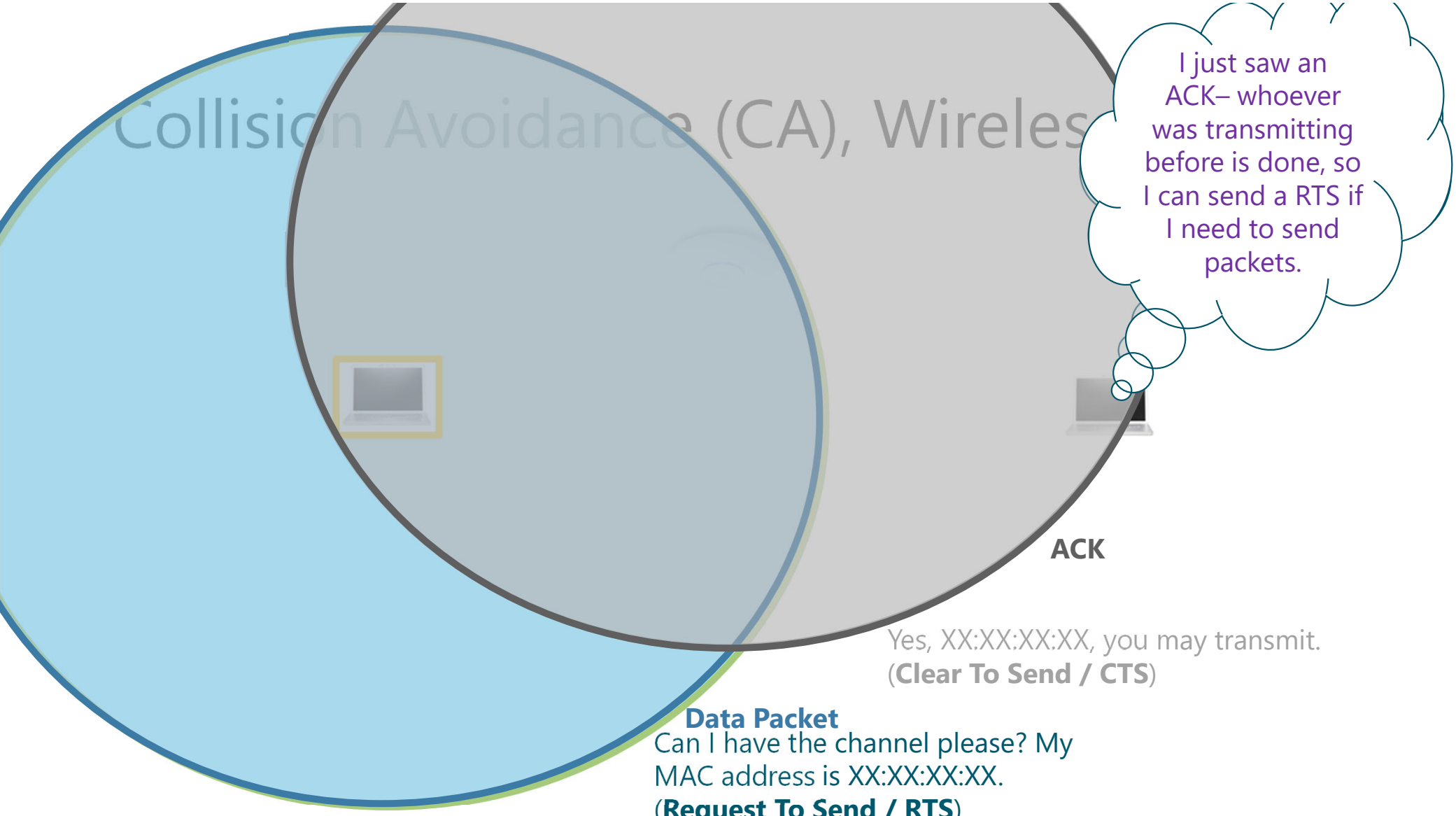


"Hidden Node" problem – device may be visible to an AP but not to other nodes communicating with that AP

On top of that, most radios are functionally half-duplex

- Transmitted signal much louder than received signal, can't hear collisions

Collision Avoidance (CA), Wireless

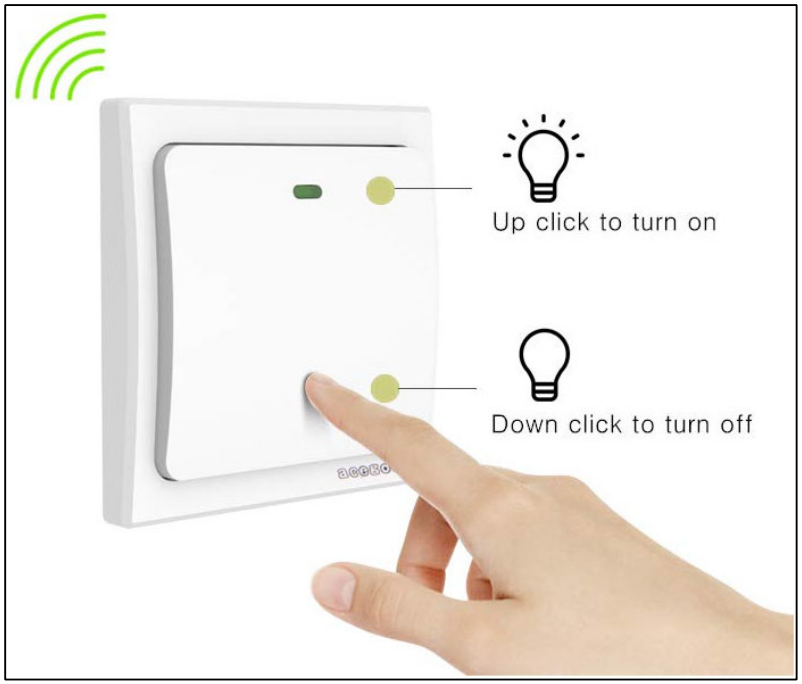


Wireless MAC

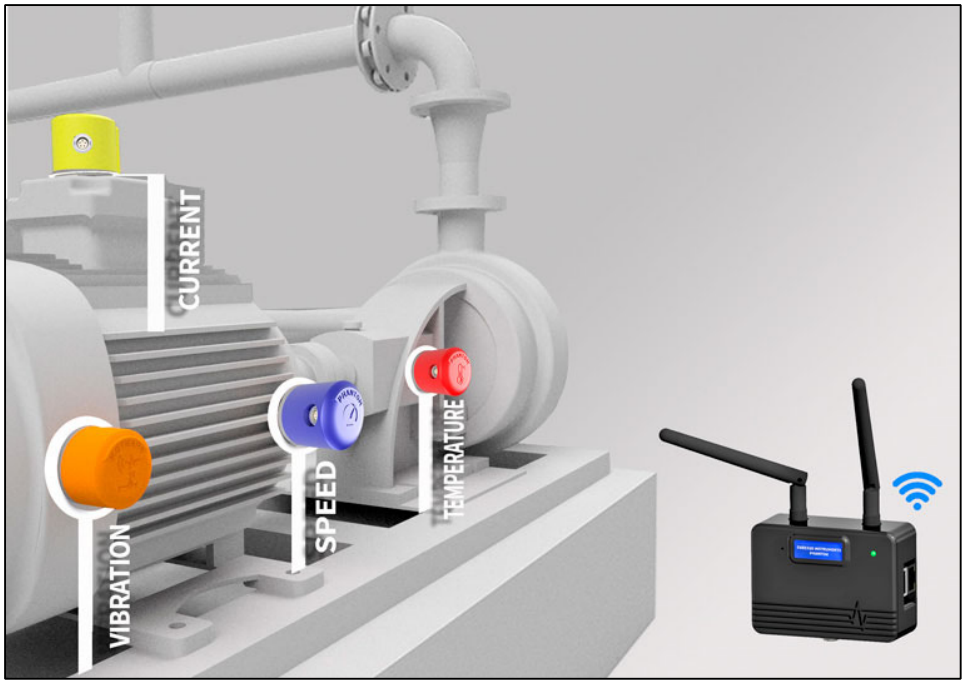
1. Collision detection and resolution
   → What if two nodes send at same time?

2. Power-saving algorithms
   → What if a node transmits but the destination is sleeping?

3. Association and neighbor discovery
   → Who is out there?



Batteryless Light Switch



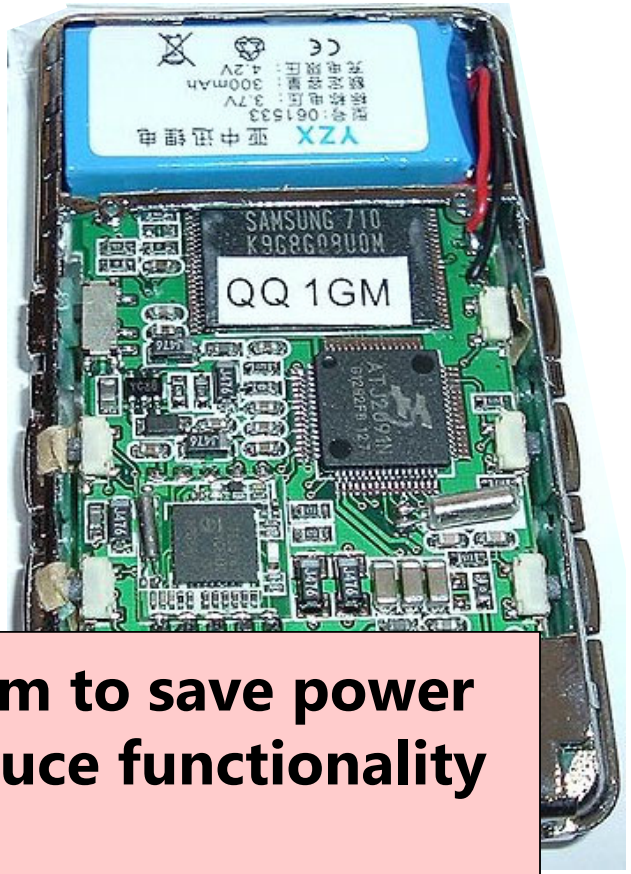
Vibration Monitoring Sensor

Smart Clothing

Grain Monitoring

Can we be smart about managing power?

Component
LTE Radio (transmitting at 1Mbps)
3G Radio (transmitting at 1Mbps)
WiFi Radio (transmitting at 1Mbps)
ARM CPU+RAM (100% CPU utilization)
ARM CPU+RAM (idle)
Smartphone Screen (100% brightness)
GPS (after lock is acquired)
Acc
Ima



- **<u>Idea</u>: Shut down or idle parts of system to save power**
- **<u>Challenge</u>: Low-power mode may reduce functionality or stop components from working**

Problem: Power Usage

Component
LTE Radio (transmitting at 1Mbps)
3G Radio (transmitting at 1Mbps)
WiFi Radio (transmitting at 1Mbps)
ARM CPU+RAM (100% CPU utilization)
ARM CPU+RAM (idle)
Smartphone Screen (100% brightness)
GPS (after lock is acquired)
Accelerometer (10Hz)
Image Sensor (1080p@30Hz)

- Many IoT devices operate in energy-constrained environments
- Idea: Shut down or idle parts of them to save power
  - Low-power mode may reduce functionality or stop components from working

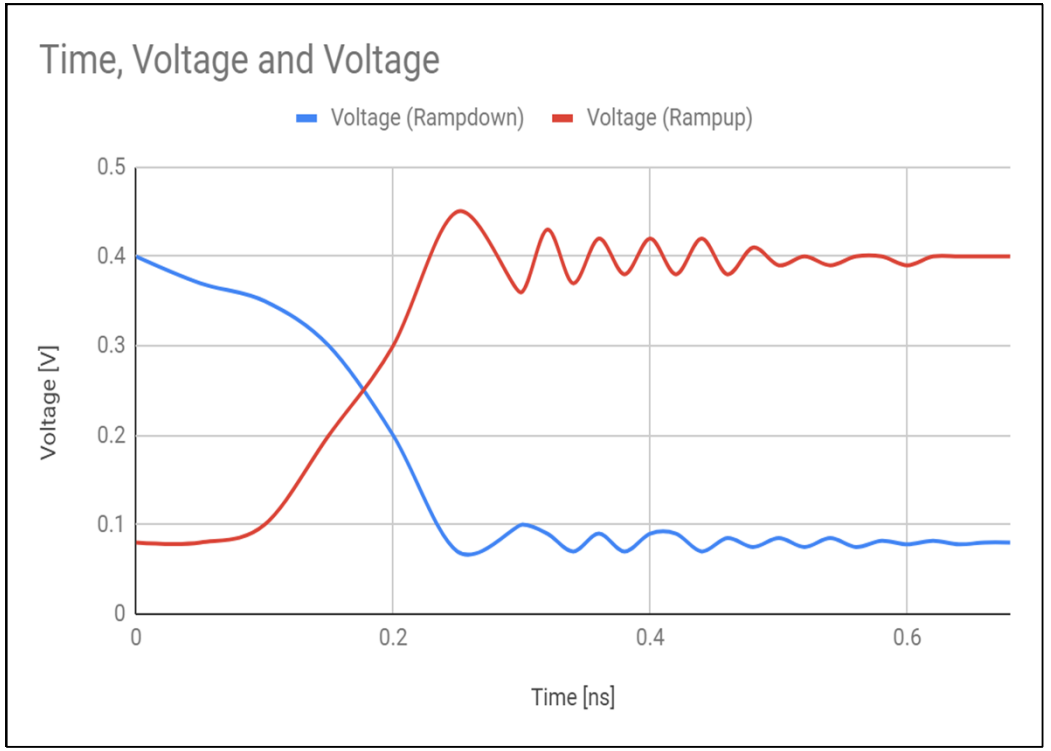
Saving Power: More Complex Than Just Not Sending

- High power consumption even when receiving and idling
- Sleep mode allows most of electronics to be turned off
  - Can't receive or send packets when sleeping
  - MUCH lower power usage
- Only protocols that use sleep mode a lot can save a significant fraction of power

Function	Power Usage (approx.)
Transmit	1.33 Watts
Receive	0.97 Watts
Idle	0.84 Watts
Sleep	0.07 Watts

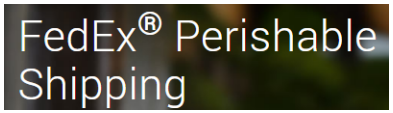
Saving Power: More Complex Than Just Not Sending

- Entering/leaving sleep mode takes time
  - Takes a little time to “wake up”
  - Circuits need some time to stabilize
- So, need to plan carefully about sleeping
  - Can’t just replace idling with sleeping



Example Use Case: Cold Supply Chain Auditing

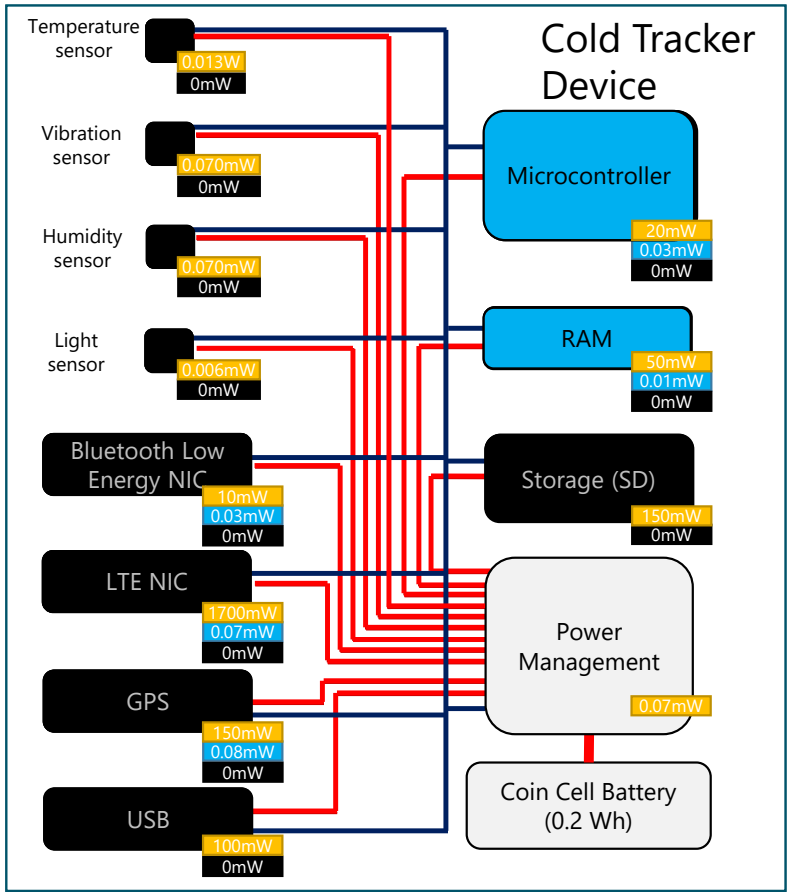
- Need to keep items cold during transport
  - Vaccines, blood, insulin, medicine
  - Food: Meat, vegetables, dairy, wine
  - Chemicals and sensitive electronics
- Cold Chain: Temperature-controlled supply chain
  - Ensures cold temperatures at every step of transport process
  - Critical for product effectiveness and consumer safety
  - $200B market



Example Use Case: Cold Supply Chain Auditing

- Suppose you want to build an IoT Cold Chain Auditor
- Implemented as tag attached to product
- Continually monitors temperature
  - Alarms when above threshold
  - Periodically records samples
- Records and communicates findings

Example Use Case: Cold Supply Chain Monitoring

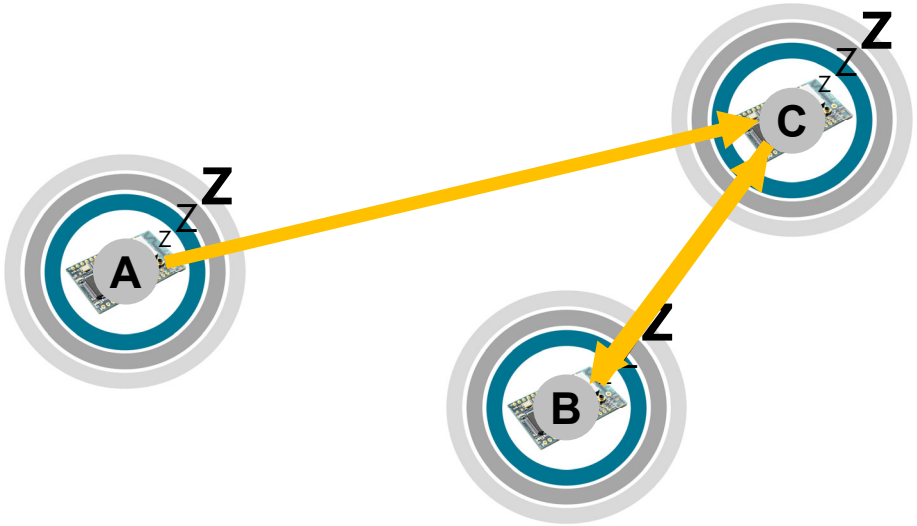


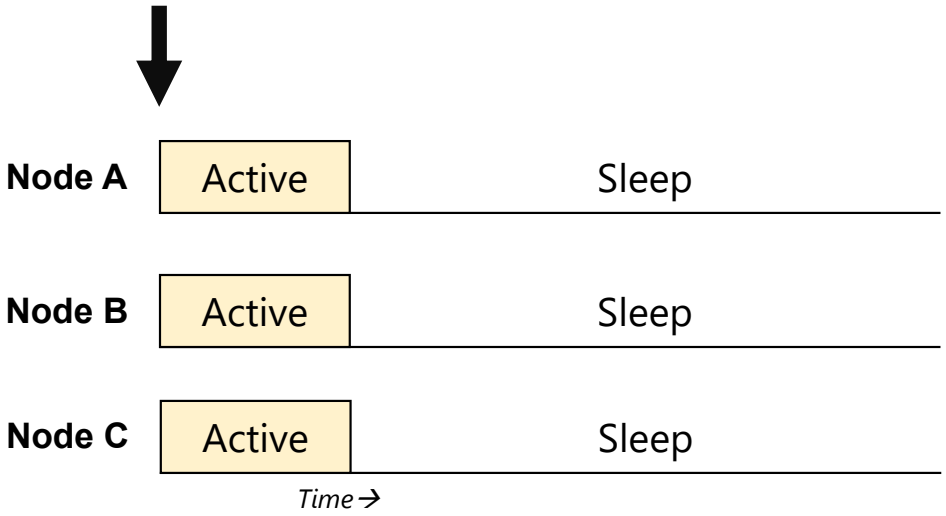
- **System on:**
  - 0.013+0.07+0.07+0.006+10+1700+150+100+20+50+150+0.07
  - = 2180.66mW → **5.5 minutes**
- **Deep sleep (emergency mode):**
  - 0+0+0+0+0+0+0+0+0+0+0+0.07
  - = 0.07mW → **battery lifetime (<7.8 years)**
- **Sensors-only mode:**
  - 0.013+0.07+0.07+0.006+0+0+0+0+0.03+0.01+0
  - = 0.199mW → **41.8 days**
- **Sensors+[BLE on 10ms per hour]:**
  - 0.013+0.07+0.07+0.006+[(10mW)*10/1000/60/60+(0.03mW)*(1-10/1000/60/60)]+0+0+0+0.03+0.01+0
  - = 0.229mW → **36.4 days**
- **Sensors+[BTE on 10ms per hour]+[GPS on 10 seconds per 3 hours]+[LTE on 50ms per 24 hours]:**
  - 0.013+0.07+0.07+0.006+[(10mW)*10/1000/60/60+(0.03mW)*(1-10/1000/60/60)]+[(150mW)*10/3/60/60+(0.03mW)*(1-10/3/60/60)]+[(1700mW)*50/1000/60/60/24+(0.03mW)*(1-50/1000/60/60/24)]+0+0+0+0.03+0.01+0
  - = 0.428mW → **19.4 days**

What happens if we send data to a device that's sleeping?

- It's ok to go to sleep, but need to be awake when data is sent to you
  - Need some way to coordinate when/how nodes sleep
- Sleeping also breaks CSMA/CD and CSMA/CA
  - Need some other way to coordinate channel acquisition

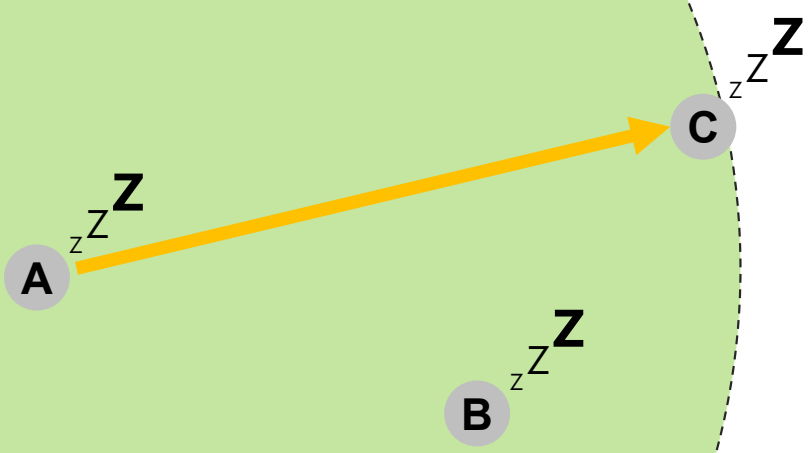
Idea: Synchronize Sleep/Wake Times

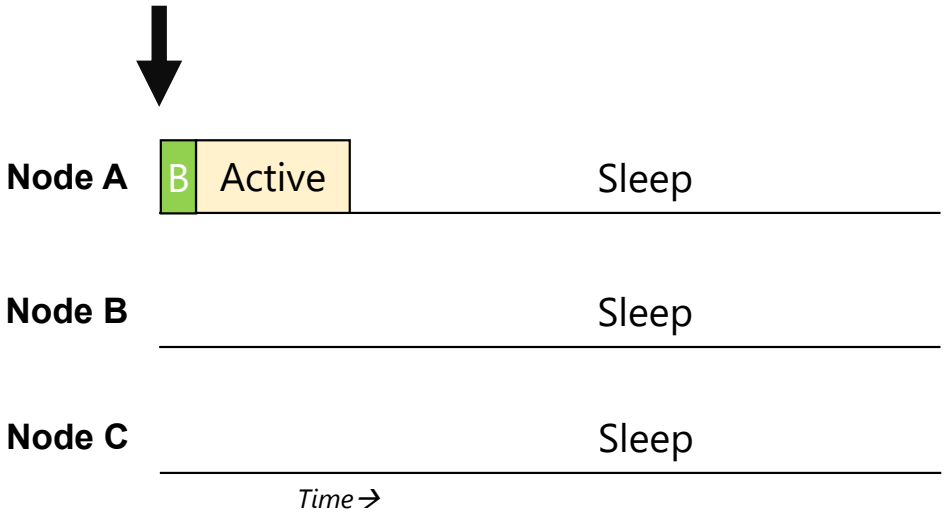




- All nodes wake up and go to sleep at the same time
  - Agree on "active" time when data can be exchanged, sleep time when they can sleep
  - Amount/ratio of active/sleep time may be statically configured
    - Goal may be to minimize active ratio
- Problem: How do nodes know when to wake up and go to sleep?

Idea: Tell Other Nodes When You Wake Up



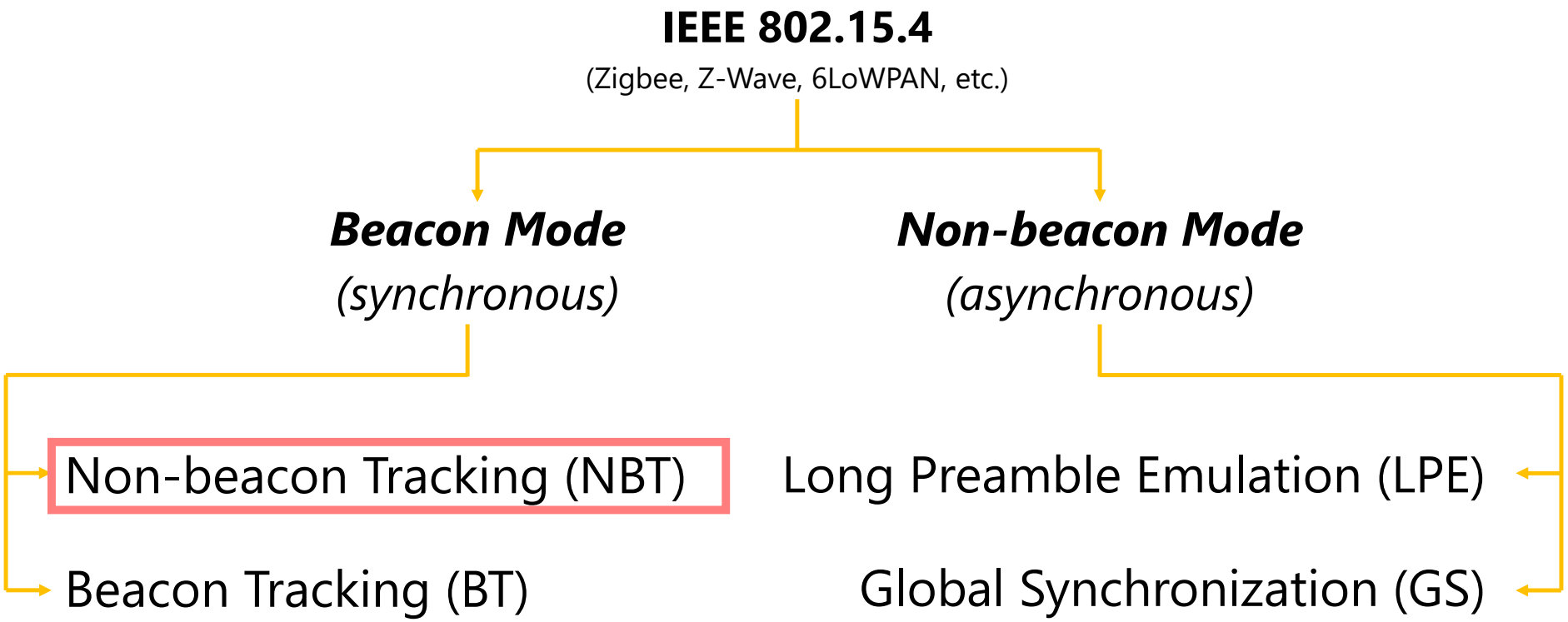


- Tell other nodes we are awake with “beacons”
  - Protocol: Stay awake for some duration after you send a beacon
- Problem: What if other nodes are asleep when beacon is sent?

MAC Power Saving Algorithms

- Low-energy wireless MACs (e.g., IEEE 802.15.4) have power-saving algorithms to deal with these challenges
- Two main kinds of algorithms:
- **Beacon mode (synchronous)**
  - Main idea: Keep nodes in sync with beacons
  - Examples: Beacon Tracking (BT), Non-beacon Tracking (NBT)
- **Non-beacon mode (asynchronous)**
  - Main idea: Let nodes autonomously discover each other's wake times on demand
  - Examples: Long Preamble Emulation (LPE/BMAC), Long Preamble Emulation with Ack (LPEA/XMAC), Non-beacon Tracking Emulation (NTE), Global Synchronization (GS/SMAC)

MAC Power Saving Algorithms

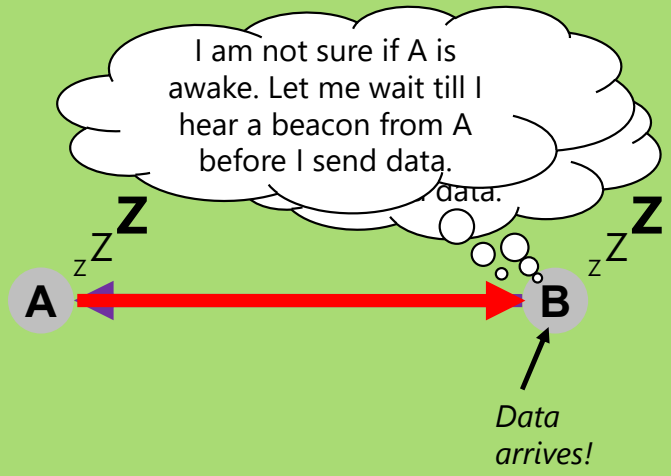


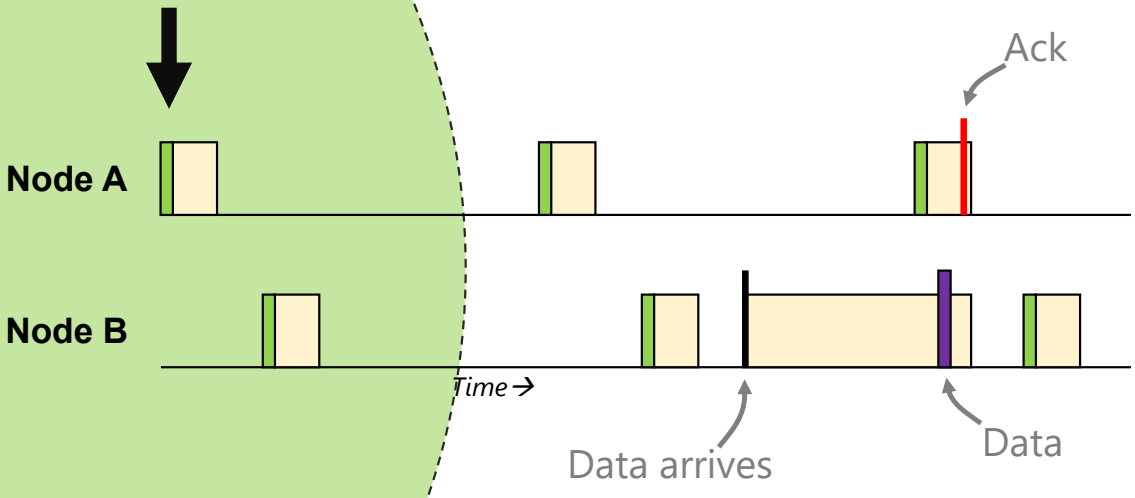
Non-beacon Tracking (NBT)

- Asynchronous wakeup algorithm:
  - Uses beacons, but doesn't force nodes to continually listen for them

- Each node periodically wakes up and beacons
  - Stays awake for a bit after the beacon
  - If another node has data to send, it stays awake till it hears the beacon of the destination, then transmits

Non-beacon Tracking (NBT)





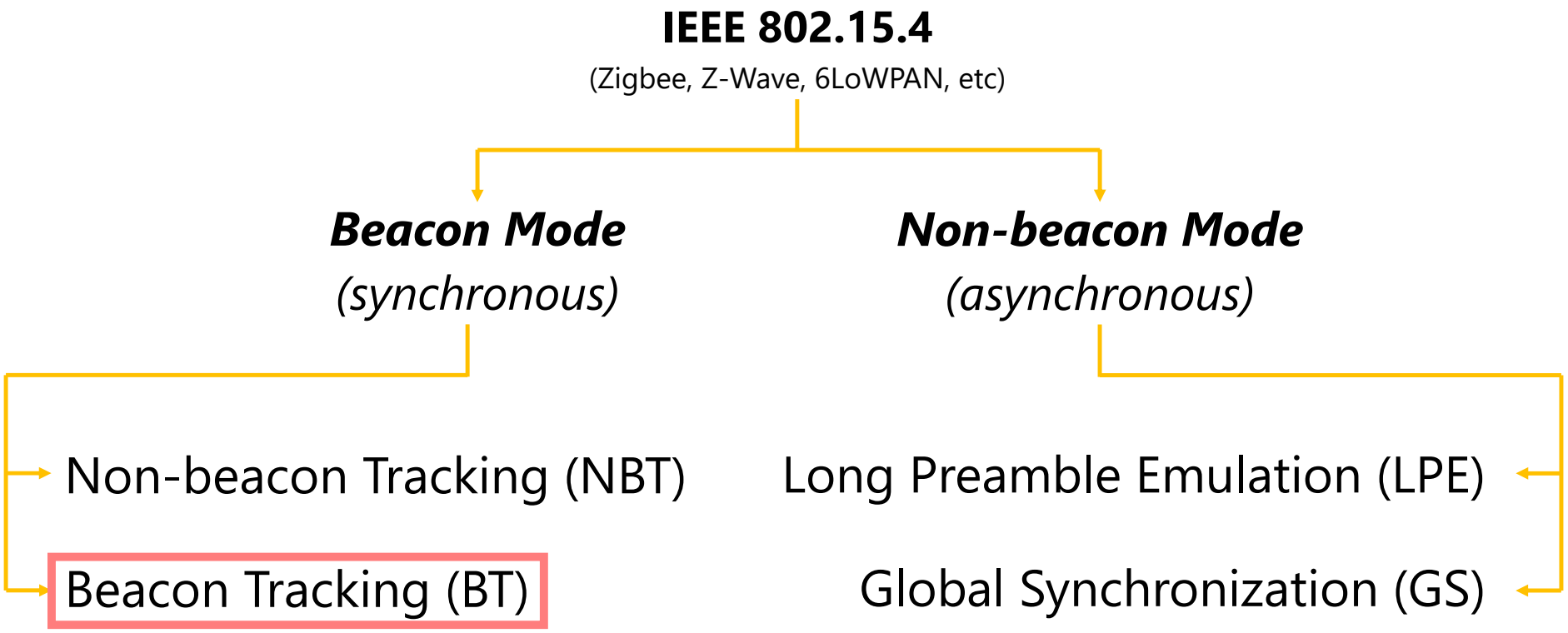
- Questions:
  - How long to make active, sleep durations?
  - How long does a node have to "wait" in active state?

Non-beacon Tracking (NBT): Downsides

- What if beacons collide?
  - Regular schedule → they'd continually collide
  - Idea: Restart beaconing after random delay, if detect another node's beacon during active window
  - Doesn't always work: Collision may happen outside radio range (hidden terminal)

- May have to wait a long time for other node to wake up
  - For beacon interarrival time t_{bi} , need to idle active on average $t_{bi}/2$ to receive destination beacon

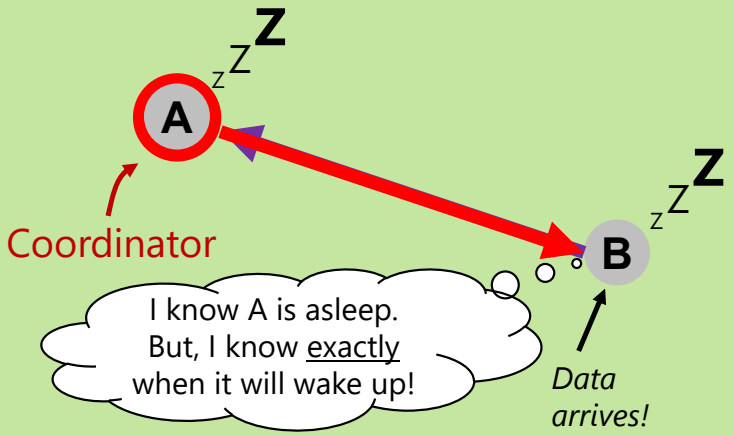
MAC Power Saving Algorithms

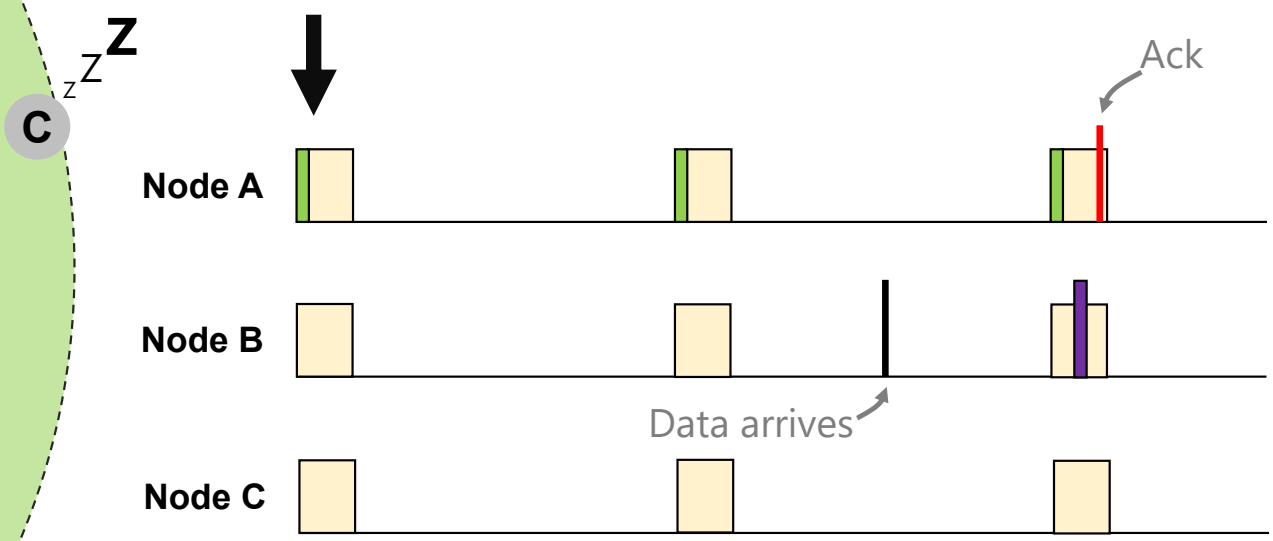


Beacon Tracking (BT)

- Synchronous wakeup algorithm using superframe structures
  - No long blocks of idle listening for beacons
- Coordinator broadcasts beacons, which contain the coordinator's schedule
  - Nodes store coordinator's schedule and sync clock to beacon arrival time
  - Nodes try to receive next beacon by turning on their receiver slightly before expected beacon transmission time

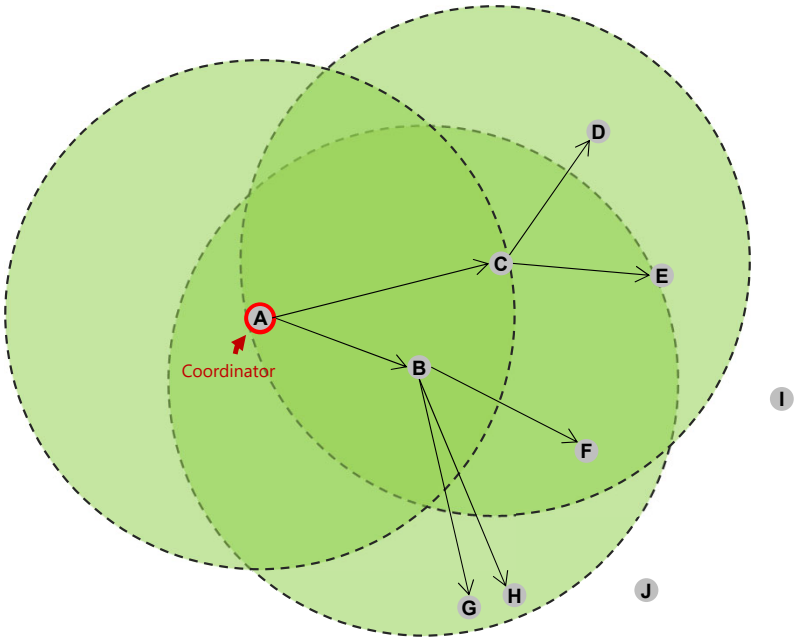
Beacon Tracking





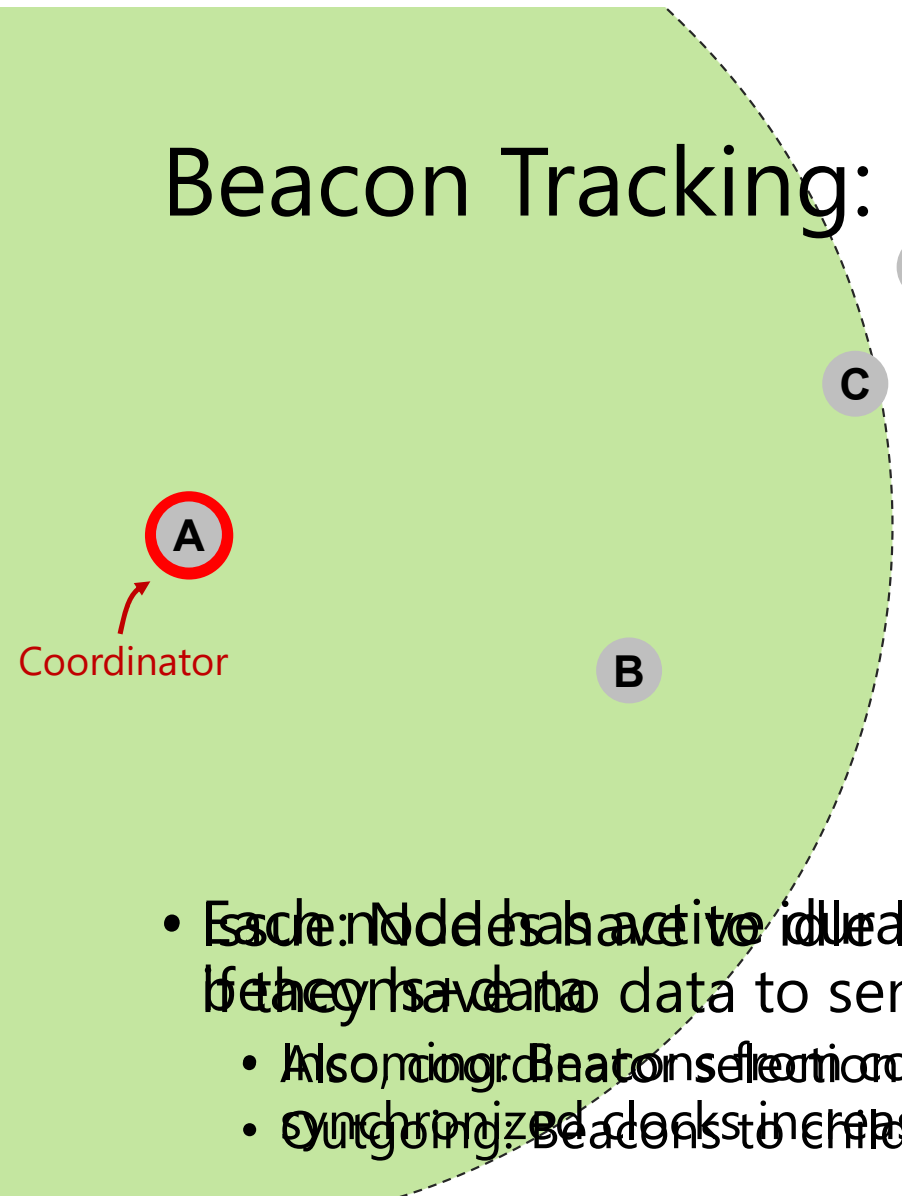
- Issues:
  - Synchronized clocks, increase hardware costs
  - What if beacon can't be heard by entire network (multihop network)?

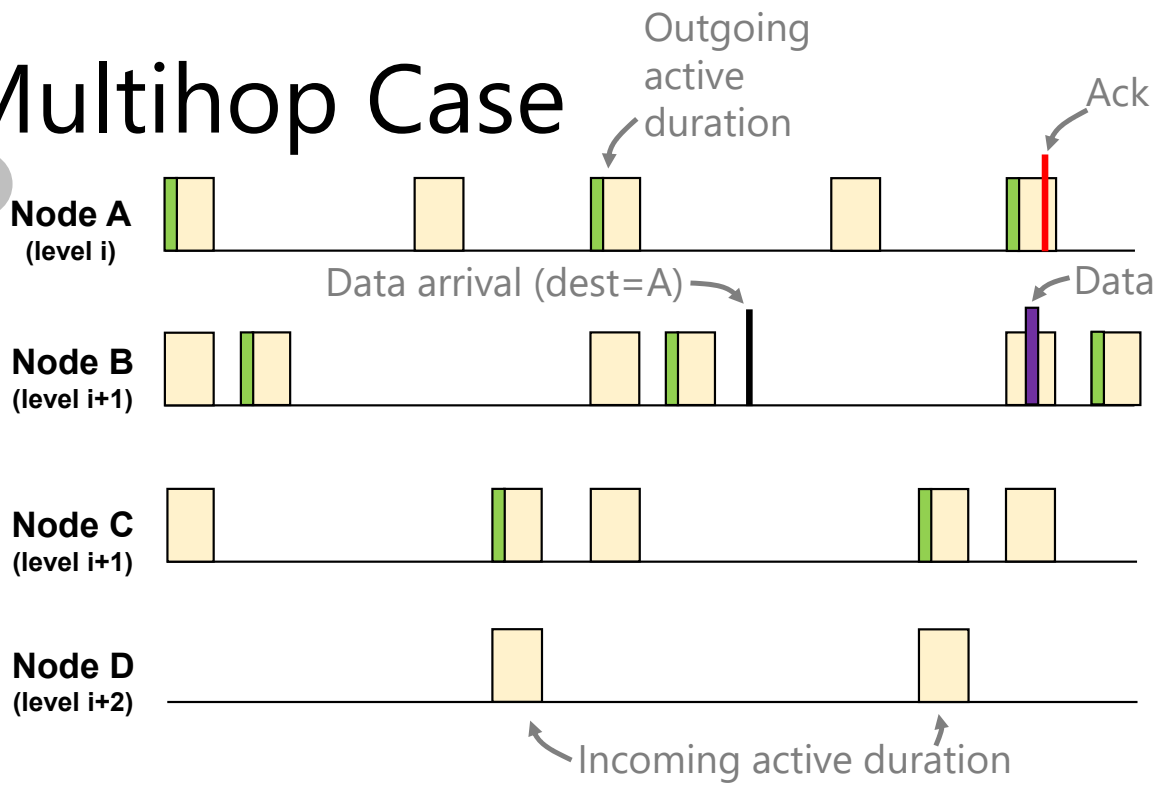
What if beacon can't be heard by entire network (multihop network)?



- Solution: Other nodes also send beacons
- Nodes "wake up" for any beacons in their broadcast region
- 802.15.4: Nodes form an "association tree"
  - Nodes wake up for parents in tree

Beacon Tracking: Multihop Case



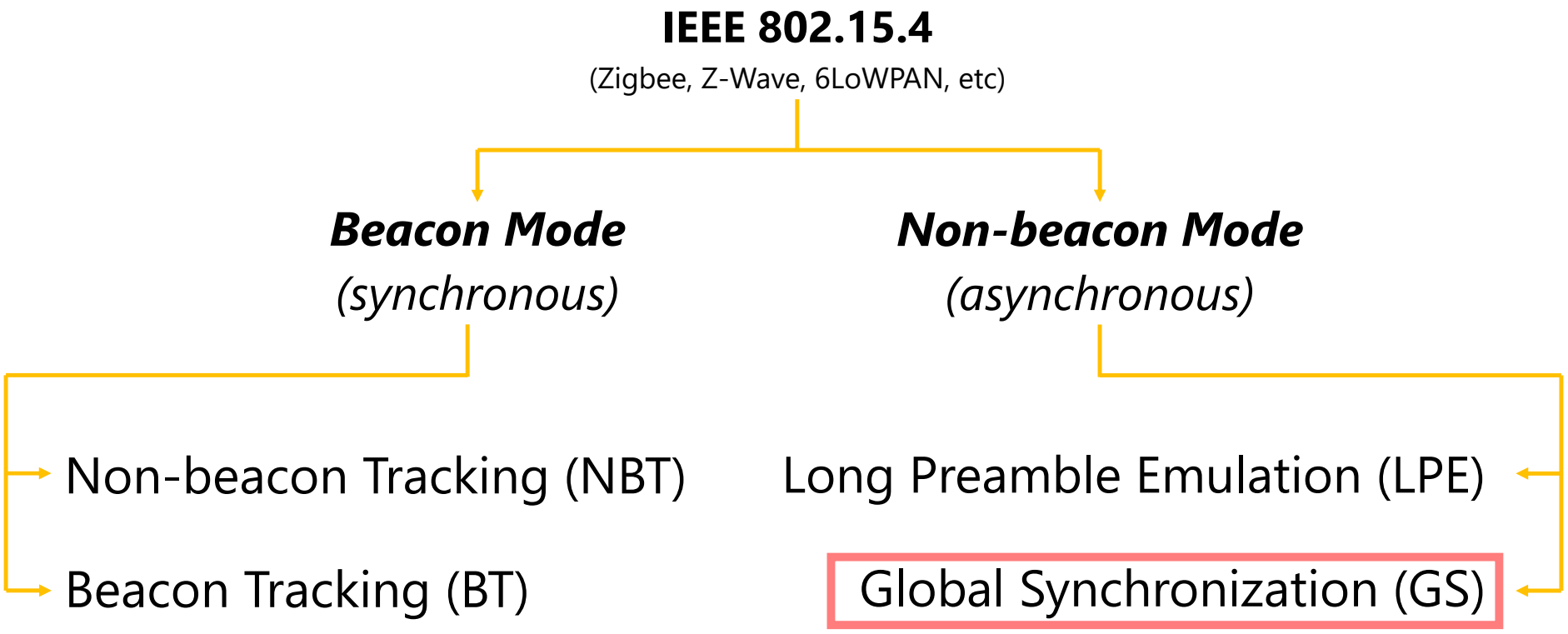


- Each node has active durations for incoming and outgoing beacons
  - Incoming: Beacons from controller or level up
  - Outgoing: Beacons to children one level down

Each node listens every superframe during even if they have no data to send

Also, coordinator selection procedure increases complexity, synchronized clocks increase hw costs

MAC Power Saving Algorithms



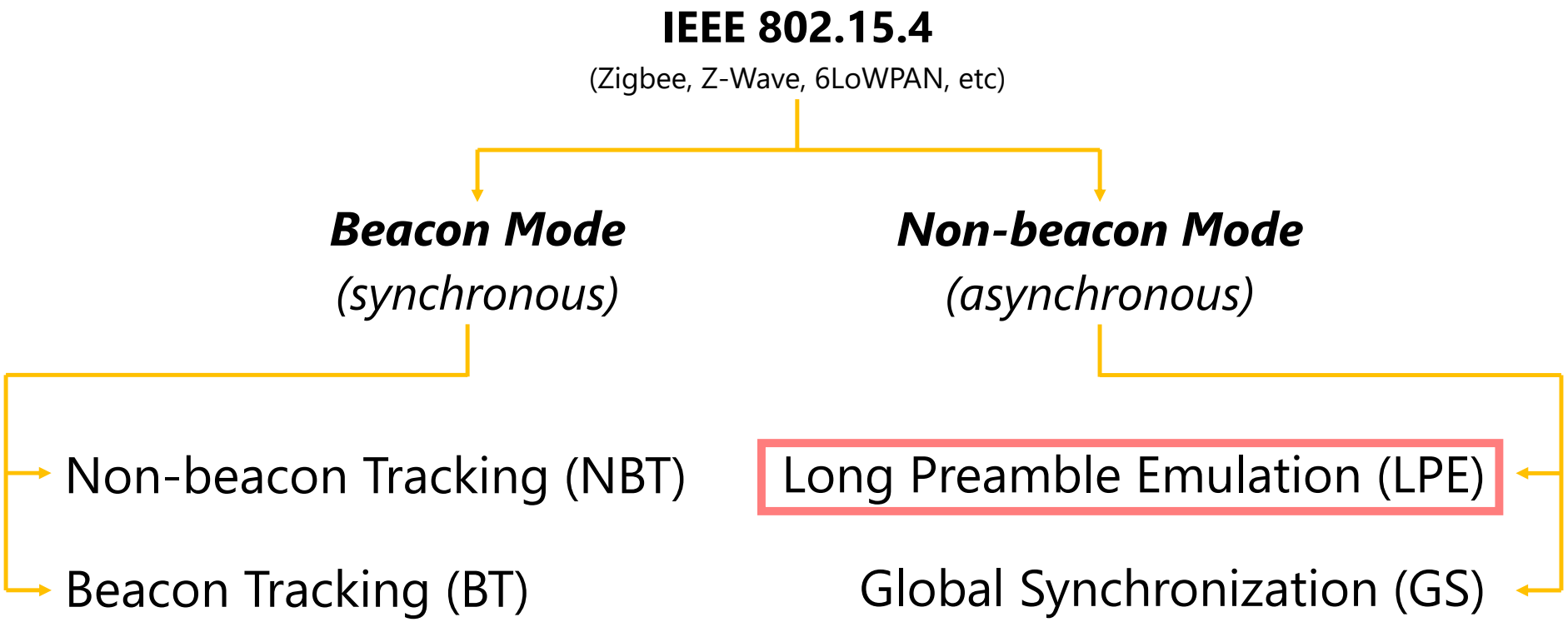
Global Synchronization (GS)

Similar to Beacon Tracking, but:

1. No continuous beacons; instead, a SYNC message is transmitted occasionally
2. Don’t need coordinators or tree; if a node doesn’t hear a SYNC message for a while it broadcasts one
3. SYNC message can’t collide; CSMA/CA is used on them to prevent that

Downsides: Requires more active time to receive a SYNC, clock errors aggregated over multiple hops in mesh network

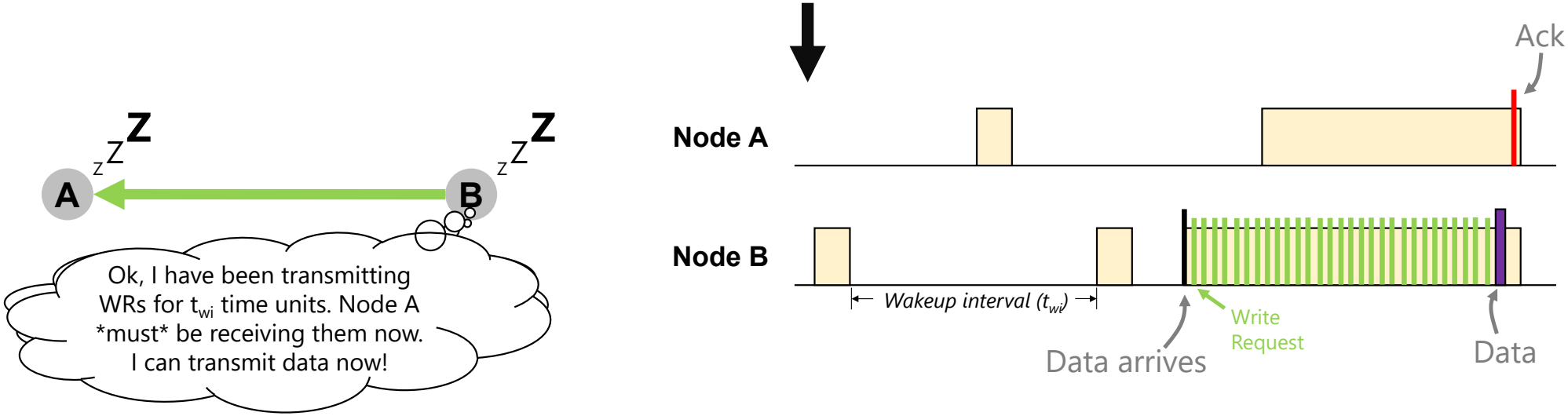
MAC Power Saving Algorithms



Long Preamble Emulation (LPE)

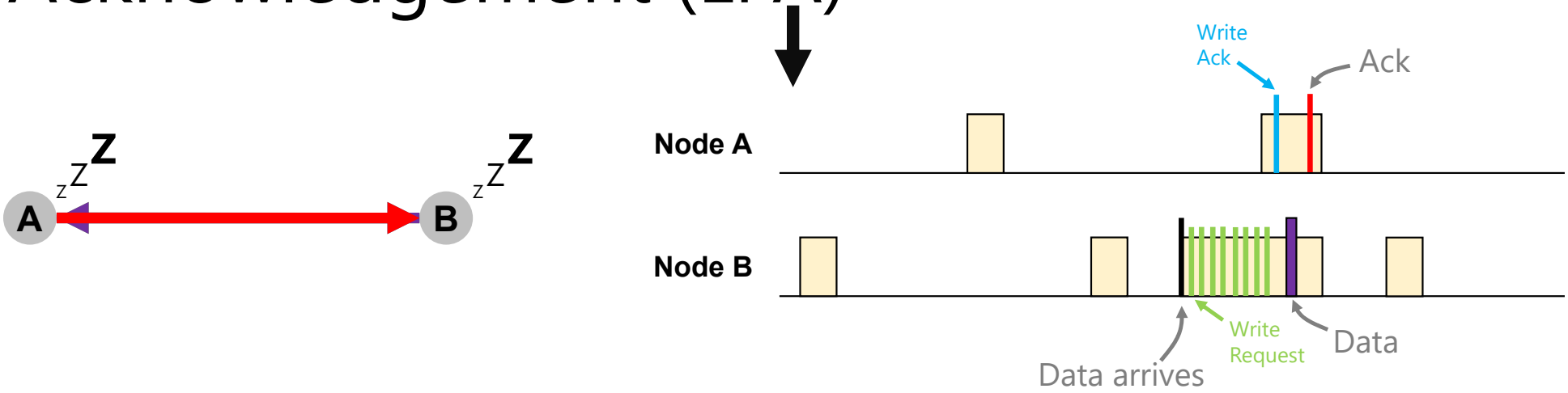
- Also known as B-MAC
- Non-beacon mode, asynchronous
  - Nodes don’t send beacons at all
- Instead, nodes wake up on regular intervals
  - Intervals are unsynchronized, but happen regularly
  - So if I need to send a node data, I know it will wake up within a certain amount of time
- Sender sends “write requests” until receiver would have woken up, then sends data
  - Receiver stays awake when it hears a write request, stays awake until data is received

Long Preamble Emulation (LP)



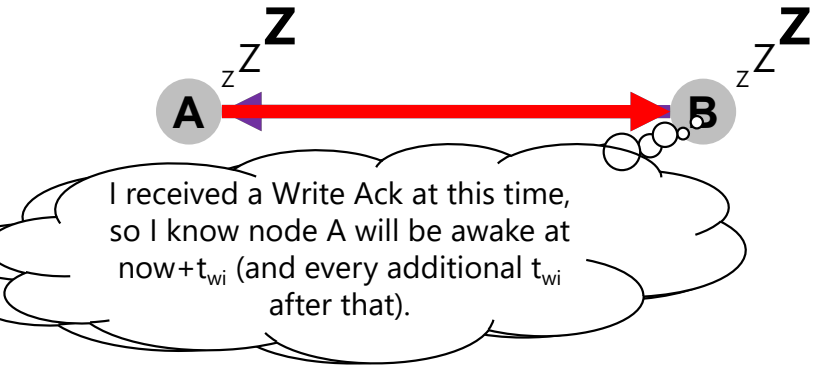
- Nodes wake up every Wakeup interval (t_{wi})
- When sender has data, it sends Write Requests for t_{wi}, then sends data
- When receiver gets a Write Request, it remains awake until data arrives
- Issue: Seems wasteful to send all those Write Requests

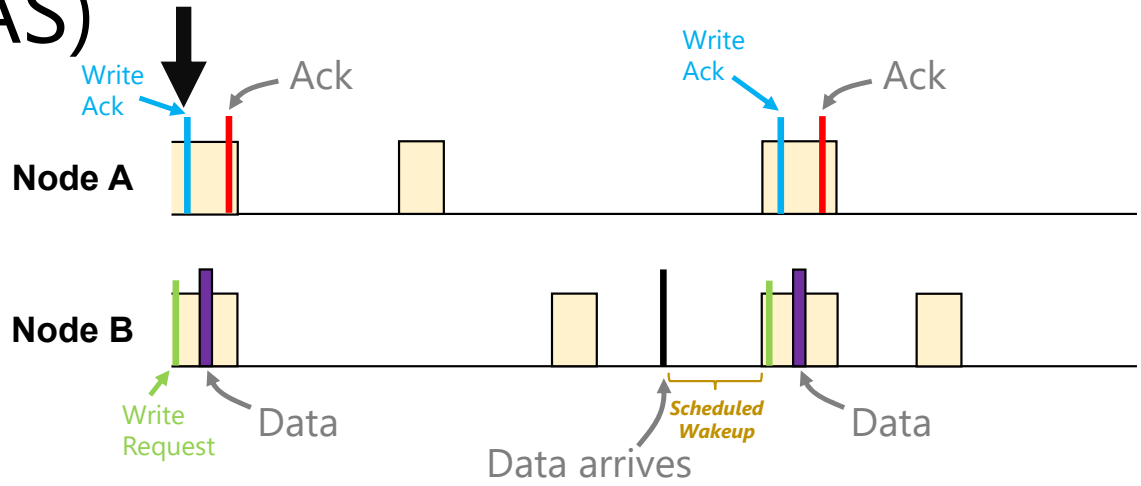
Long Preamble Emulation with Acknowledgement (LPA)



- Idea: Receiver sends ACK when data is received
- Sender can stop sending, go back to sleep, on receiving ACK
- Issue: Sender still needs to stay awake until receiver wakes up

Long Preamble Emulation with Acknowledgement after Local Synchronization (LPAS)





- When a node receives a Write Acknowledgement, it logs the time
  - Can estimate the node's schedule from that
- Later, if it needs to send to the same receiver, it turns on when it expects the receiver to wake up
  - Turns on a bit earlier to mitigate estimation delay

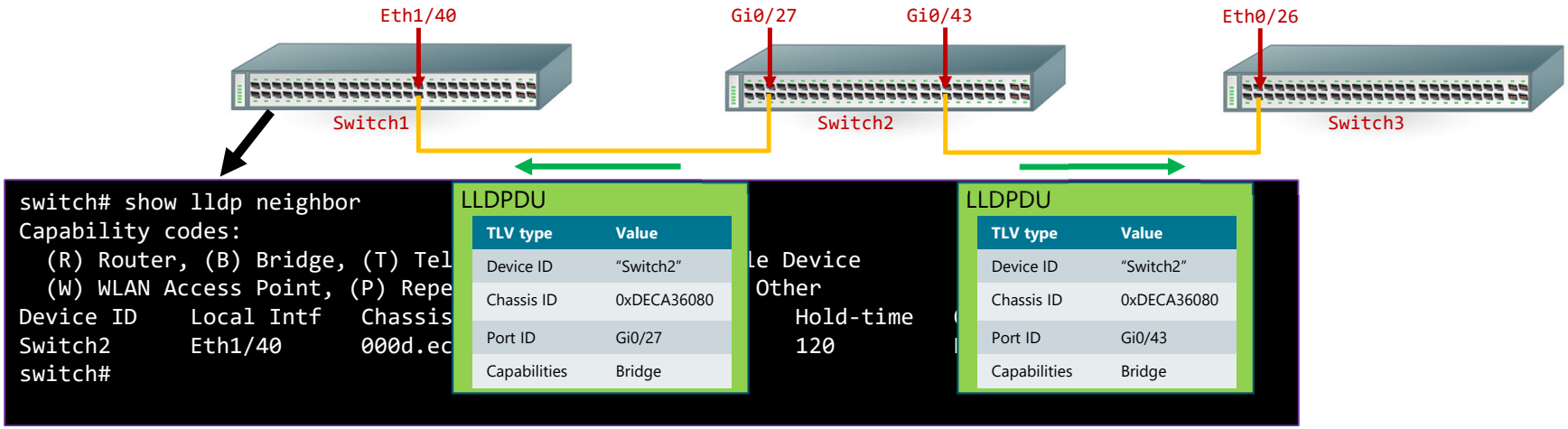
Wireless MAC

1. Collision detection and resolution
   → What if two nodes send at same time?

2. Power-saving algorithms
   → What if a node transmits but the destination is sleeping?

3. Association and neighbor discovery
   → Who is out there?

Neighbor Discovery in Wired Networks: Link-Layer Discovery Protocol (LLDP)

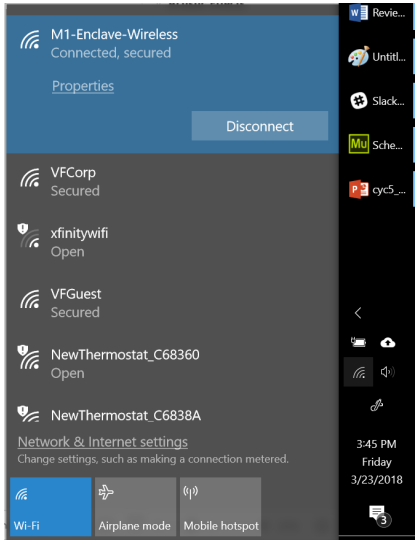


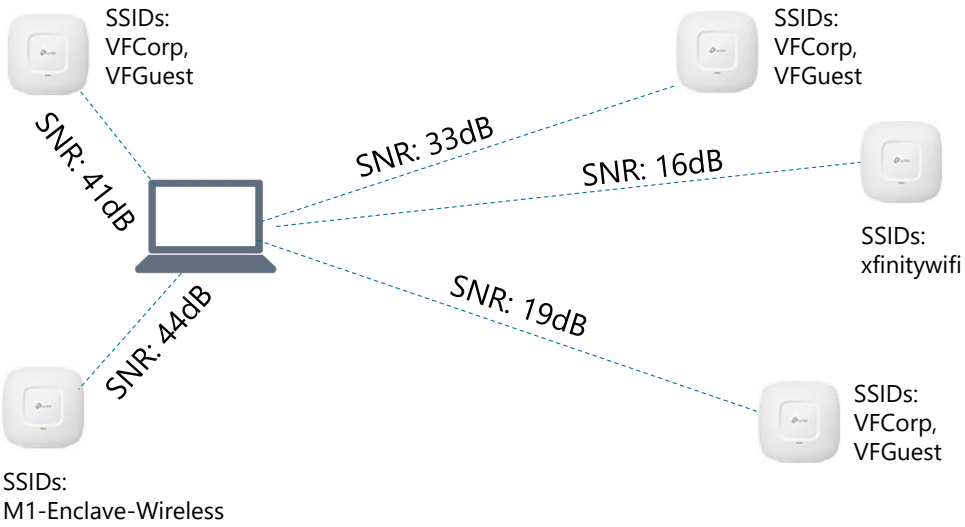
- Periodically (e.g., every 30 seconds) exchange LLDPDUs with physical neighbors
  - Each LLDPDU contains set of type-length-value (TLV) records
  - Example TLVs: Chassis ID, Port ID, System name, System description, Management address, etc.
- Common Cisco-Proprietary variant: Cisco Discovery Protocol (CDP)

Can we use LLDP to discover neighbors in wireless?

- Wireless has additional challenges
  - May have multiple “equivalent” neighbors (e.g., access points)
  - May need to discover “closest” neighbor
  - May need to reserve resources
  - Neighbors may lie about who they are
- Wireless protocols have been developed to solve these challenges
  - Association, discovery, authentication

How to know where access points are?

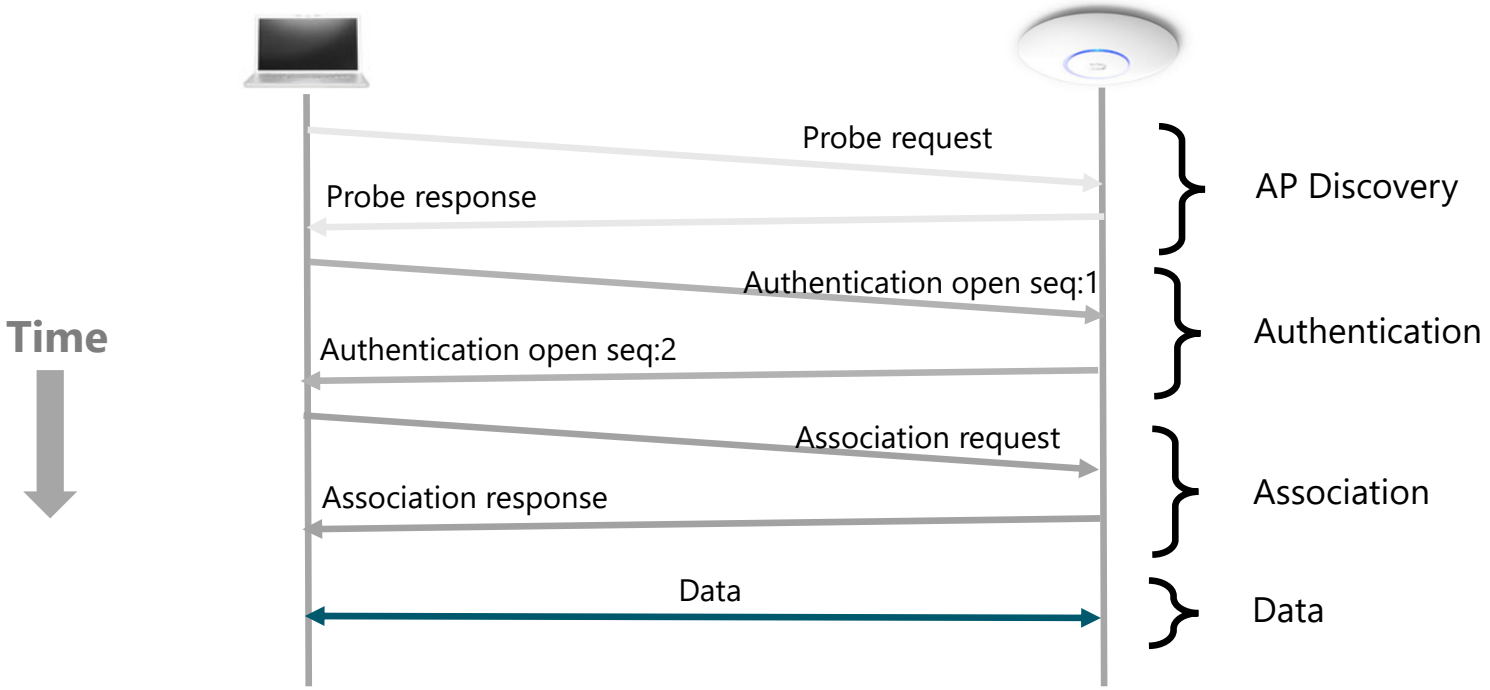




- NICs scan all channels searching for beacons
  - Can show list to user, e.g., sorted by signal strength or preference
  - NICs can also actively send probe requests to actively scan

- Access points periodically send beacon frames
  - Contain SSIDs, supported rates, transmission parameters
  - Default interval of 100ms

AP Association Process



- Done on first connection
- If signal becomes low, client repeats process to be associated with another AP

- TODO Need to flesh this out – association and such

Mesh Networking

Mesh Networking

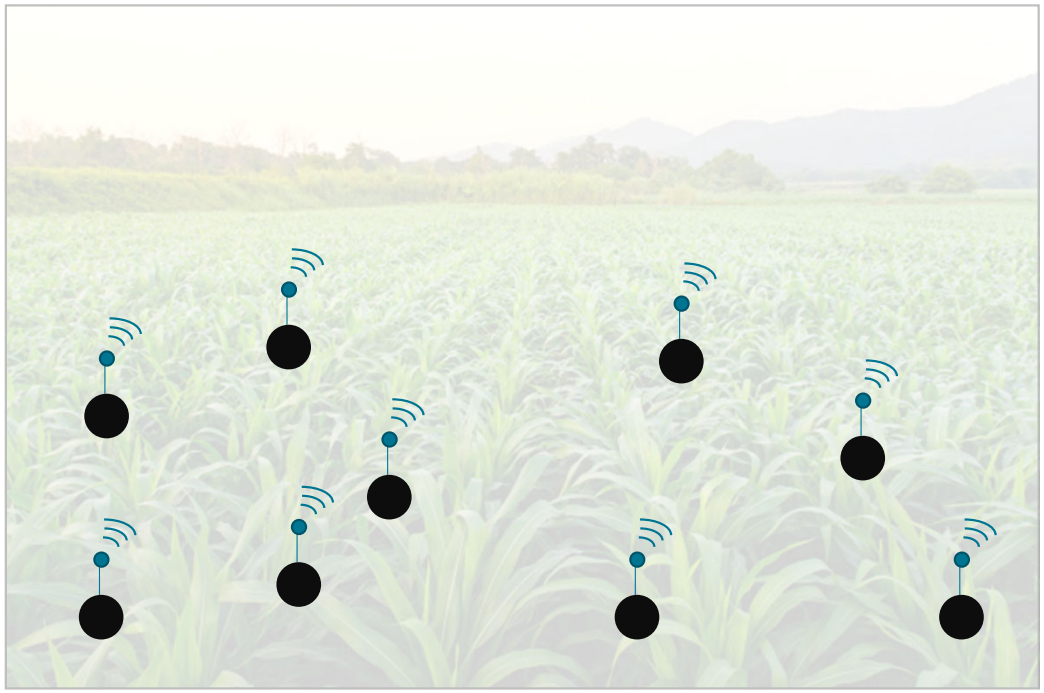
1. Addressing
   → How to identify destinations that may be multiple hops away?

2. Traditional Routing
   → How can nodes discover paths across intermediate hops?

3. Mesh Routing
   → How to route in dynamic, unstable wireless environment?

Motivating Scenario

- Distribute a bunch of nodes in some environment
  - Home, field, zoo, building
- Need to collect data, monitor
- No (reliable) wireless infrastructure
- They need to work together to do things



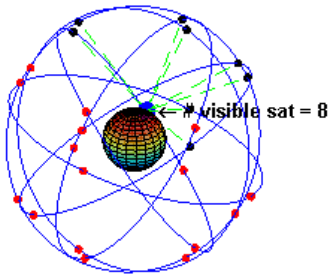
Mesh Networking

- Networking elements cooperatively connect directly, dynamically, and non-hierarchically to forward data
  - Dynamically self-organize and self-configure
- Distributed algorithms solve **key challenges**
  - Addressing and identifying nodes
  - Routing across multiple hops
  - Forming spanning topologies
  - Replicating and collecting data

Mesh Networking: Applications

Electric smart meters

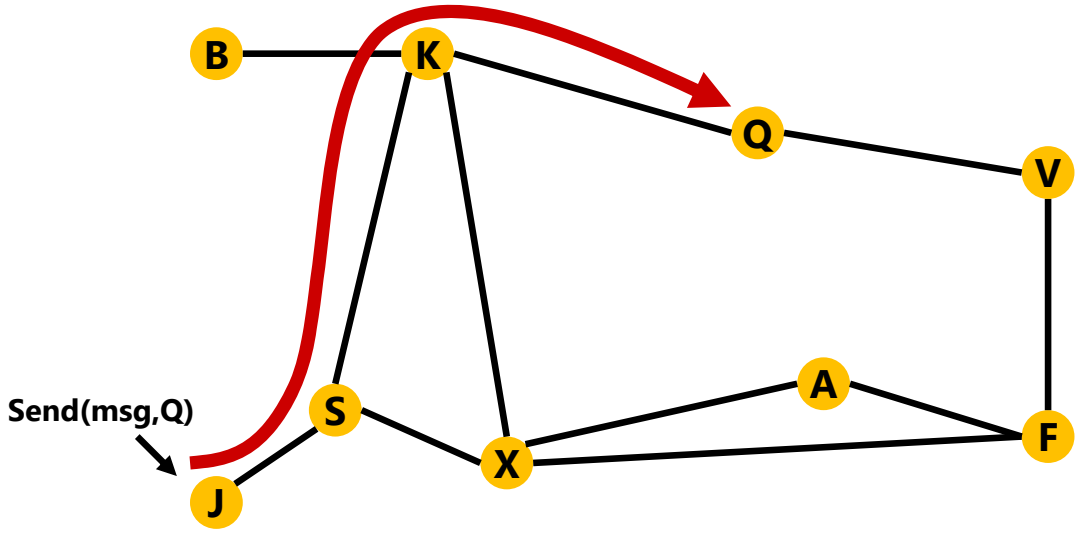
Smart homes (e.g., Google Home)



iridium Everywhere

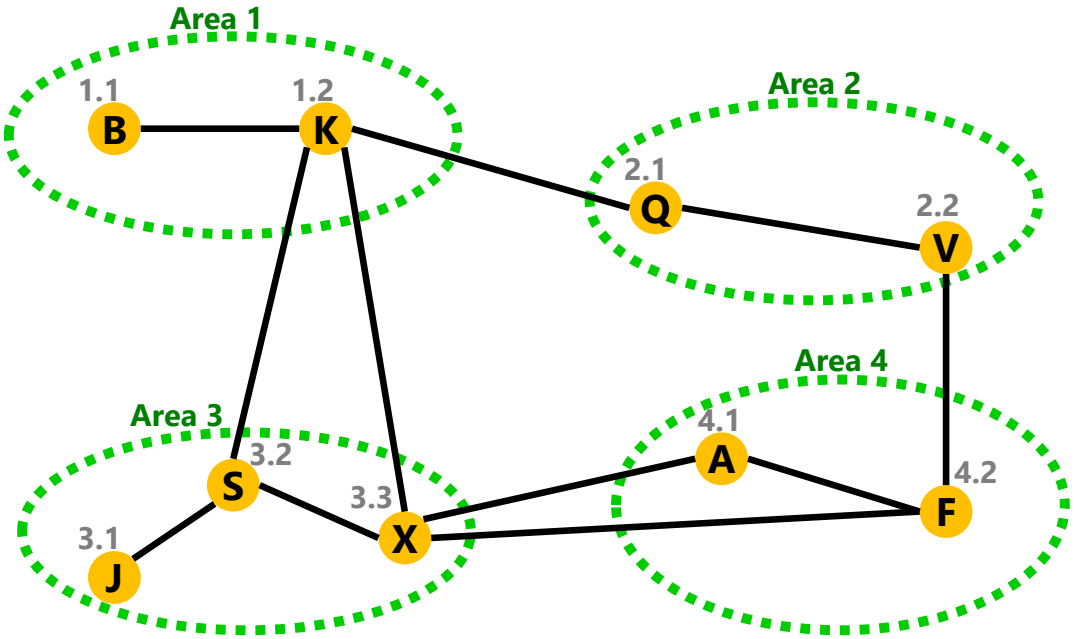
Space networking (e.g., Iridium satellite constellation)

How Routing Works



- Each node has an address
- Goal: find path to destination

Scaling Routing With Aggregation



- Pick addresses that depend on location
  - Topology-dependent addressing
- Aggregation provides excellent scaling properties

Types of Addressing

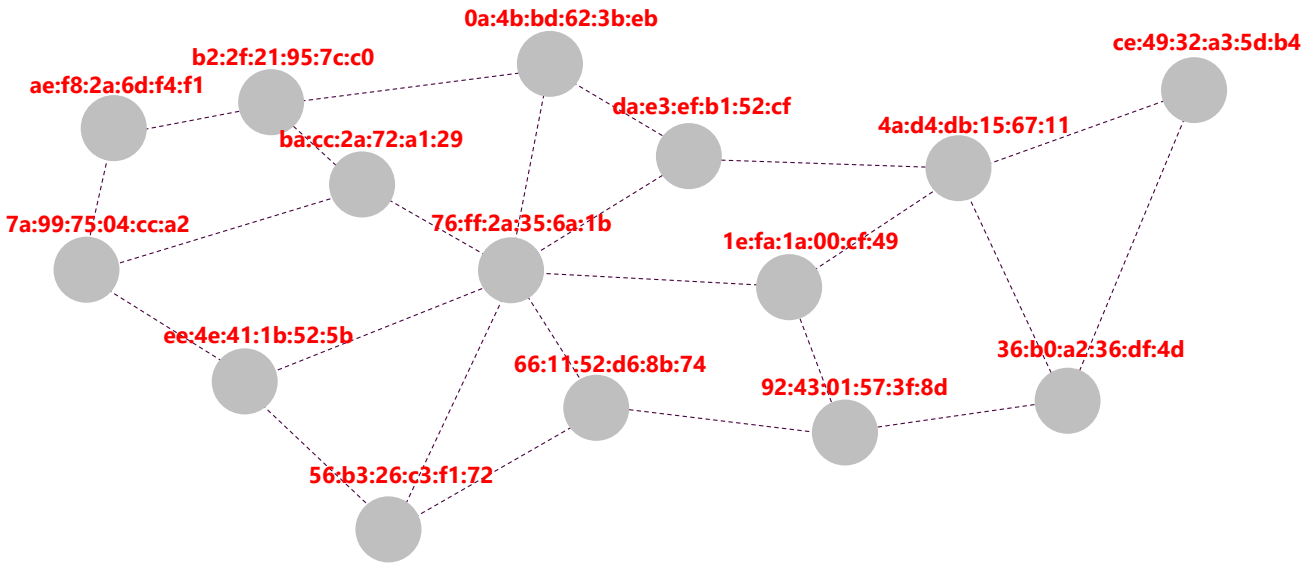
- What should addresses identify?
  - Device
  - Interface
  - Application
  - Service
  - User
  - Logical Node
- IoT presents additional requirements
  - Zero-management (don’t want home users to manage addresses)
  - Robust to churn (nodes may come and go over time)
  - Uniqueness (each node identity should be unique)

Common Approaches to Addressing

1. Hardware addressing
2. Geographic addressing
3. Hierarchical addressing
4. Stochastic (random) addressing

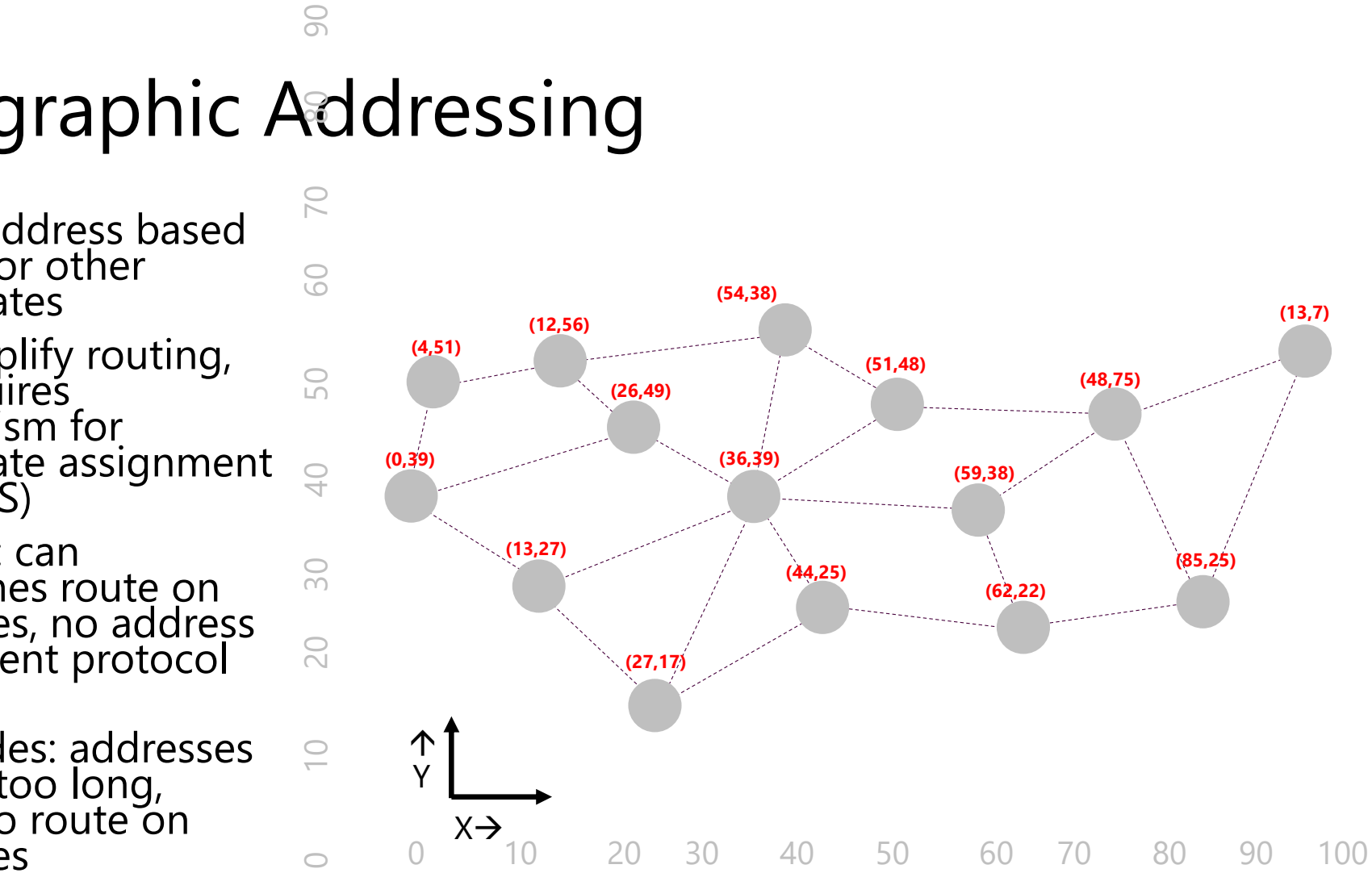
Hardware Addressing

- Interface/node comes with burned-in address/key from factory (e.g., MAC addresses)
  - E.g., vendor given block from standards body, allocates to production line
- Benefits: no address assignment protocol needed
- Downsides: addresses may be too long, harder to route on addresses



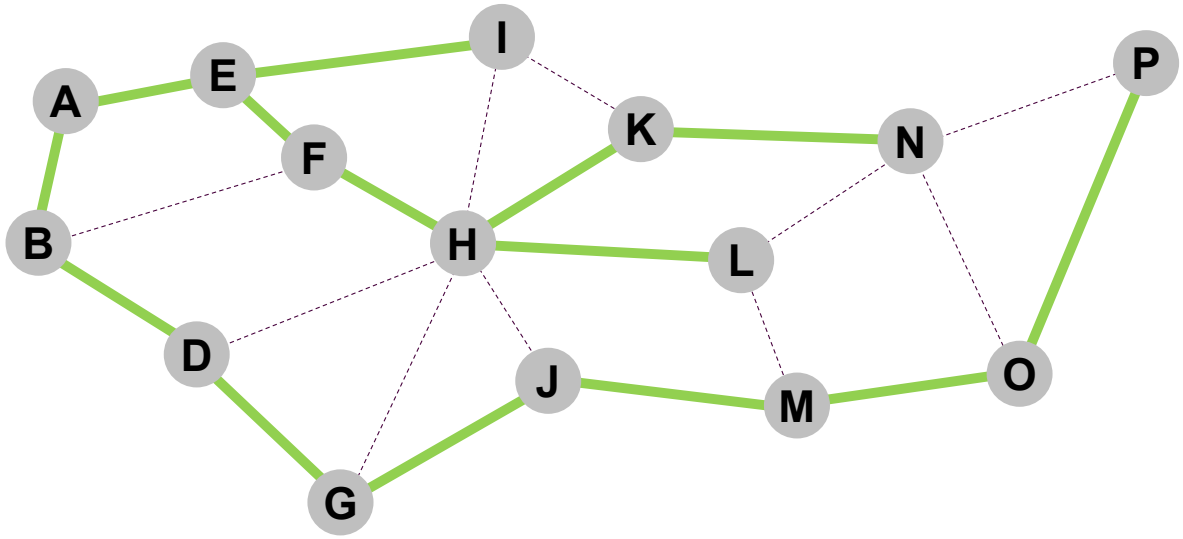
Geographic Addressing

- Assign address based on GPS or other coordinates
- Can simplify routing, but requires mechanism for coordinate assignment (e.g., GPS)
- Benefits: can sometimes route on addresses, no address assignment protocol needed
- Downsides: addresses may be too long, harder to route on addresses



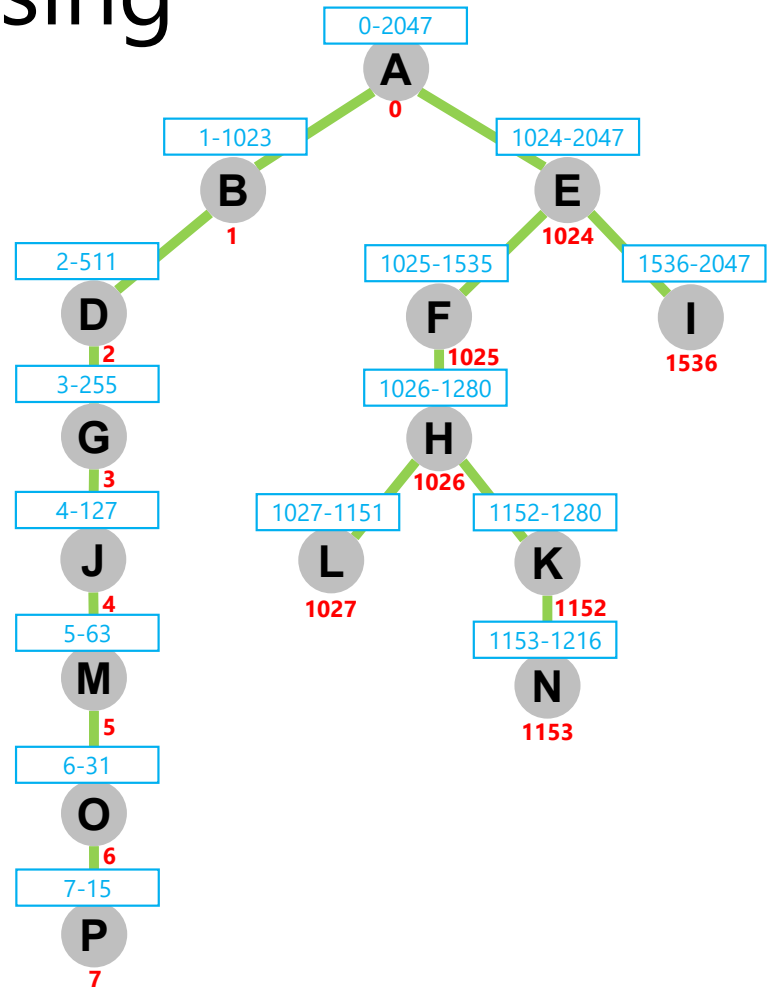
Hierarchical Addressing

- Organize nodes in a tree, parent gets block of addresses, gives itself and children addresses/sub-blocks from that range



Hierarchical Addressing

- Organize nodes in a tree, parent gets block of addresses, gives itself and children addresses/sub-blocks from that range
- Benefits: Easier to route on addresses
- Downsides: Requires tree-like organization, requires advance knowledge of future arrival patterns



Hierarchical Addressing: Zigbee's Distributed Addressing scheme

- Good for networks that happen to be structured as trees
- Assigns each node a unique 16-bit address
- Approach: Given limit on
  - Maximum depth L
  - Maximum children per parent C
  - Maximum number of forwarding nodes (Routers) R
- The address of the nth child is [parent]+1+n+(n-1)*S(d)
- Where S(d) is the amount of addresses to "skip" at each level:
  - If R=1, $S(d)=1+C(L-d)$
  - If R>1, $S(d)=(CR^{L-d-1}-1-C-R)/(R-1)$

Hierarchical Addressing: Zigbee's Distributed Addressing scheme

Example:

- Max depth L=2
- Max routers R=4
- Max children C=3

$S(d)=(CR^{L-d-1}-1-C+R)/(R-1)$

$S(0)=(3*4^{2-0-1}-1-3+4)/(4-1)=4$

- (Children of level 0 will have a pool of 4 addresses)

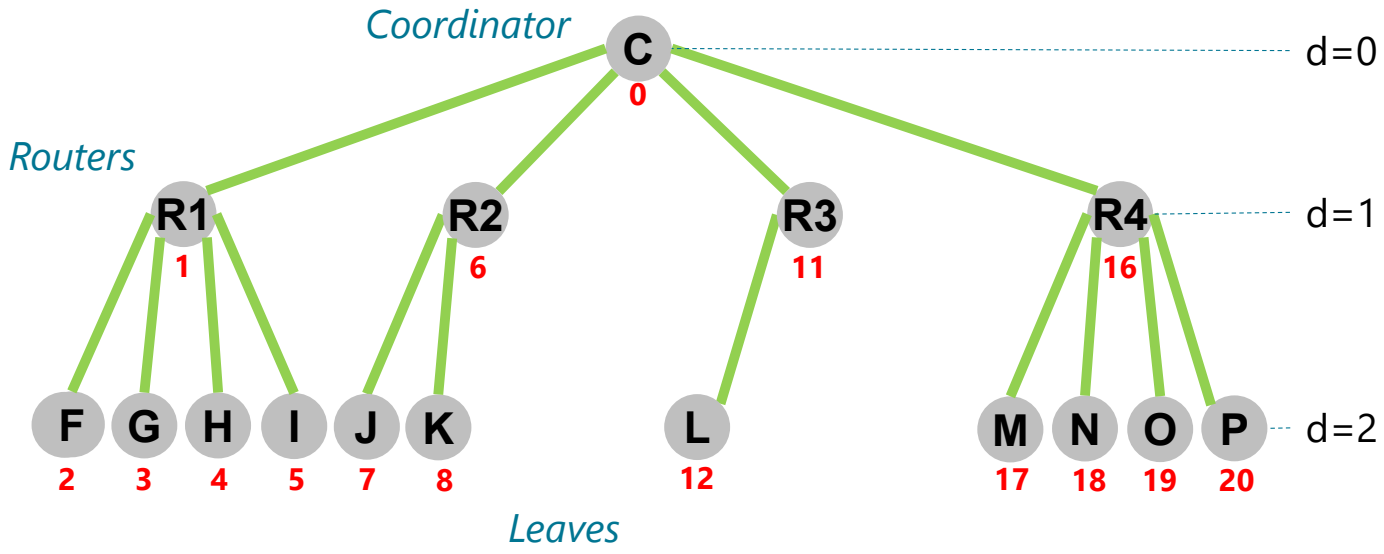
$S(1)=(3*4^{2-1-1}-1-3+4)/(4-1)=1$

- Children of level 1 will have a pool of 1 address)

Address of R1: 0+1+0=1

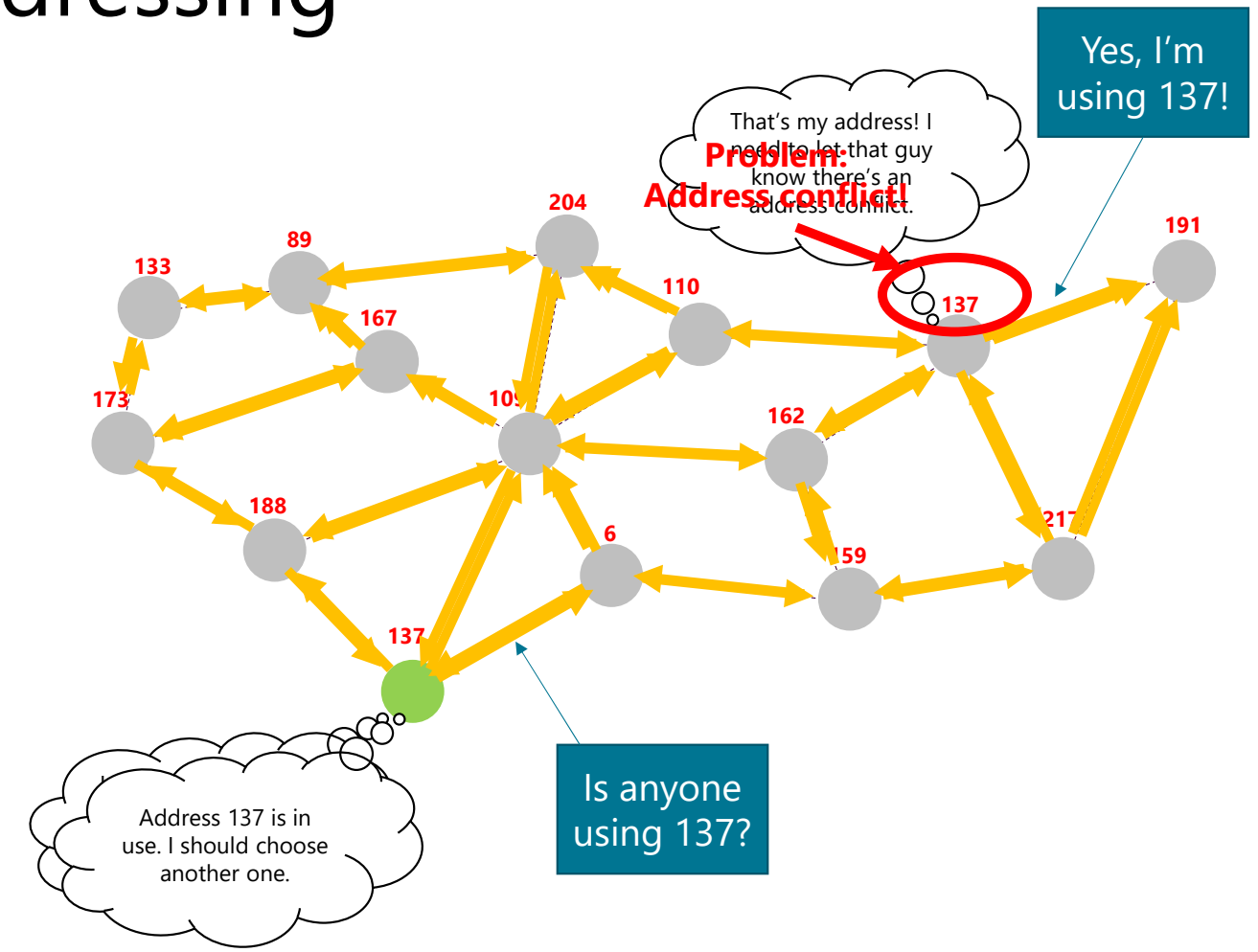
Address of R2: 0+1+1+S(0)=6

Address of R3: 0+1+2+2*S(0)=11



Stochastic Addressing

- Nodes choose random number for address
- Simple to implement, but requires conflict resolution to ensure uniqueness
  - Zigbee: broadcast address on selection, if a node has a conflict it broadcasts an error
  - Collisions more common than you may think (see "Birthday problem")
  - Also, addresses are location-independent



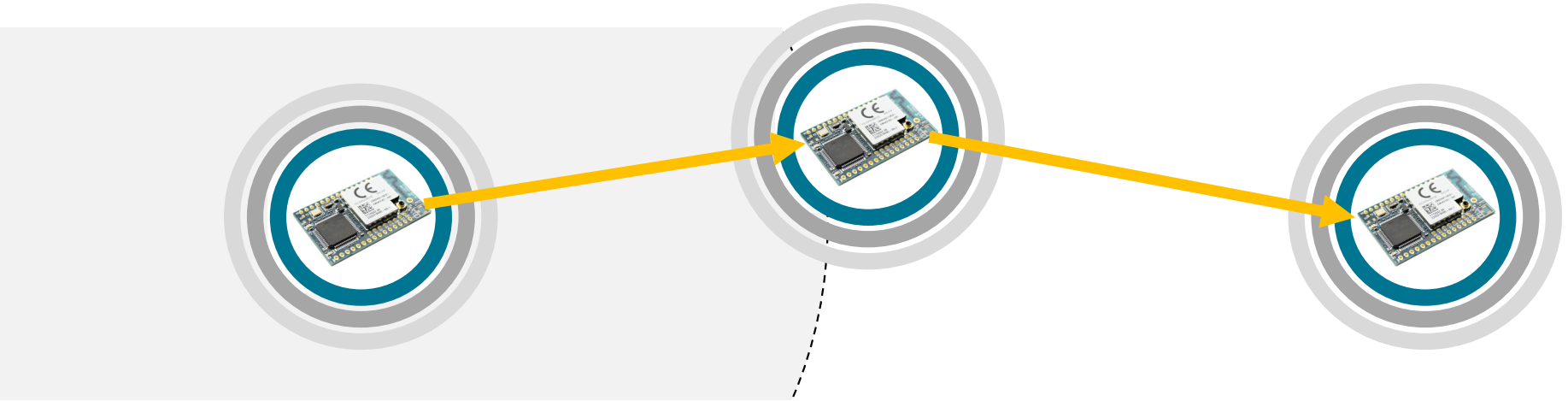
Mesh Routing

1. Addressing
   → How to identify destinations that may be multiple hops away?

2. Traditional Routing
   → How can nodes discover paths across intermediate hops?

3. Mesh Routing
   → How to route in dynamic, unstable wireless environment?

Problem: What if node you want to talk to isn't within radio range?



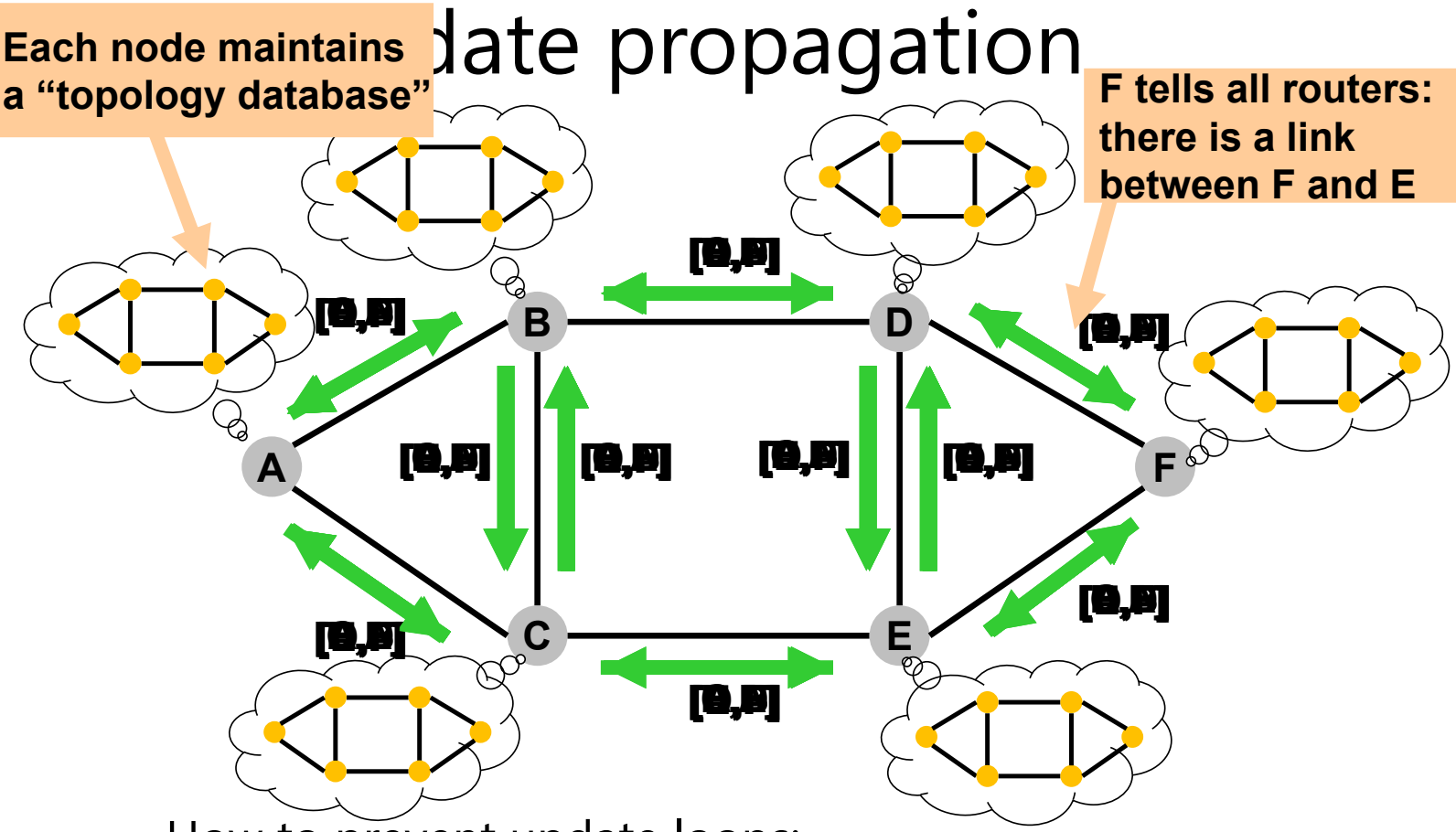
- Idea: nodes cooperate with each other to route data as far as possible
- Challenge: need distributed algorithms to discover and maintain paths across multiple hops
  - Requirements: loop-free, fully-distributed, unidirectional links

Key approaches

- "We should always have routes available to everyone, at all times!"
  - This is called proactive routing
  - Better for fixed/static environments, frequent communication
  - Requires more control overhead, memory, power; lower route acquisition time
  - Examples: OSLR, Tree Routing

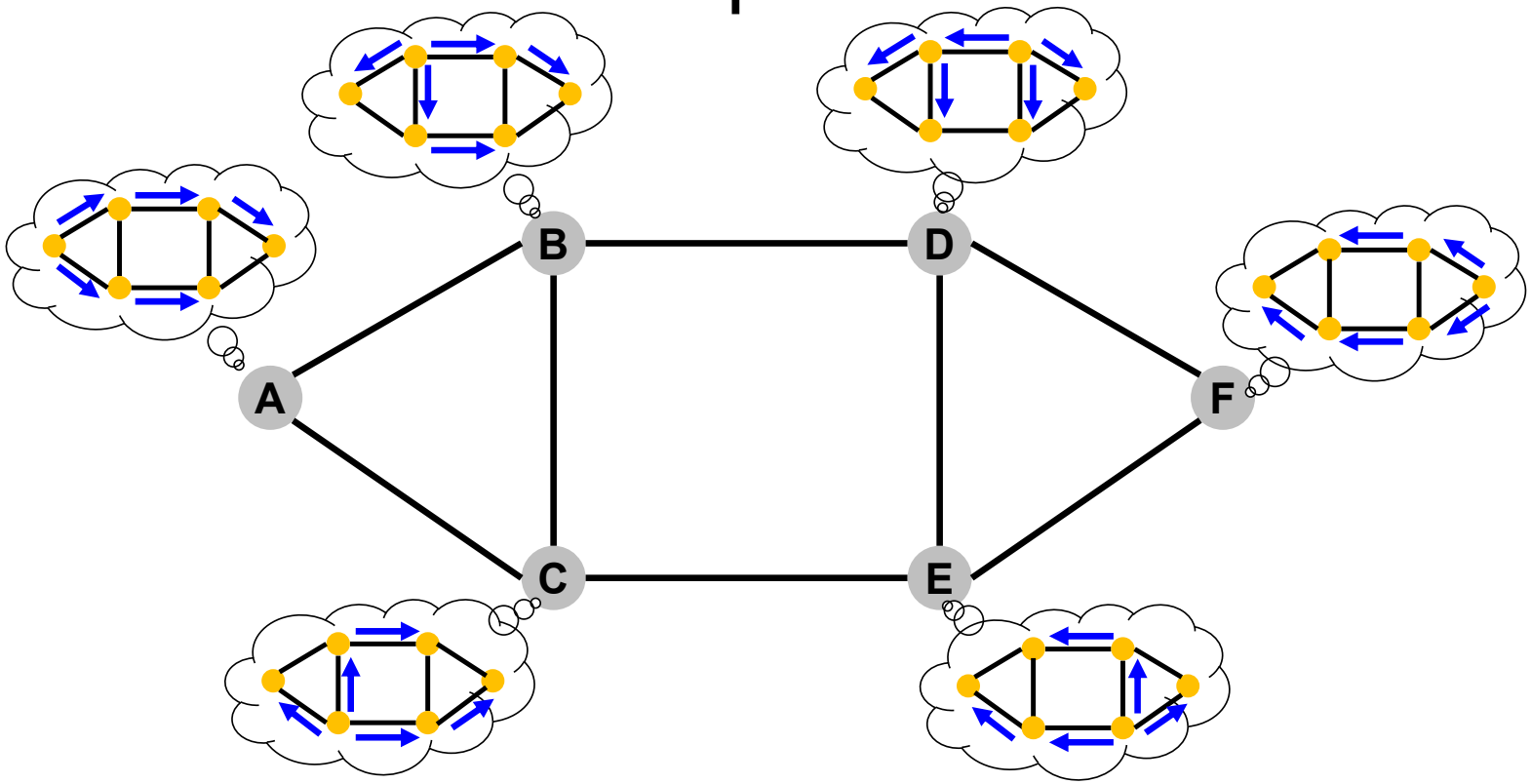
- "We should create routes only when we need them!"
  - This is called reactive routing
  - Better for dynamic environments, rare communication; higher route acquisition time
  - Requires less control overhead, memory, power
  - Examples: AODV, DSR

Lir date propagation



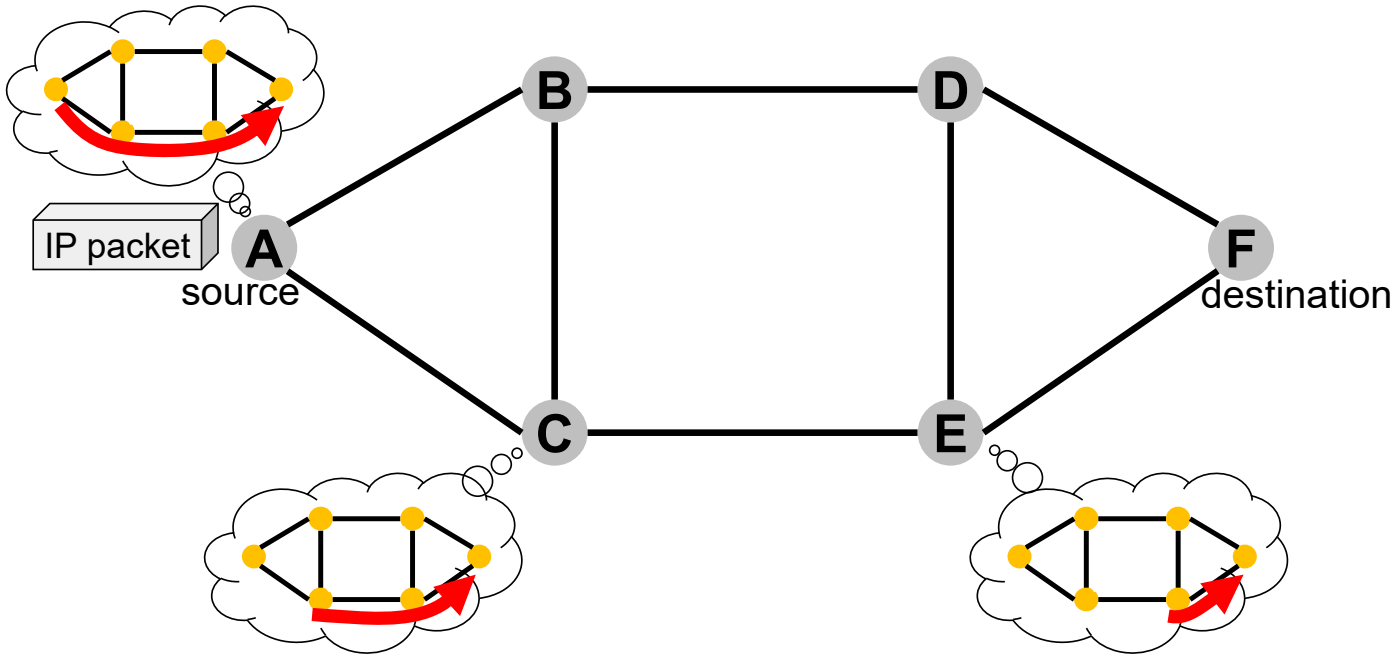
- How to prevent update loops:
- How to bring up new node:

Link state: route computation



- Each router computes shortest path tree, rooted at that router
- Determines next-hop to each destination, publish to forwarding table
- Protocols can assign link costs to control path selection

Link-state: packet forwarding

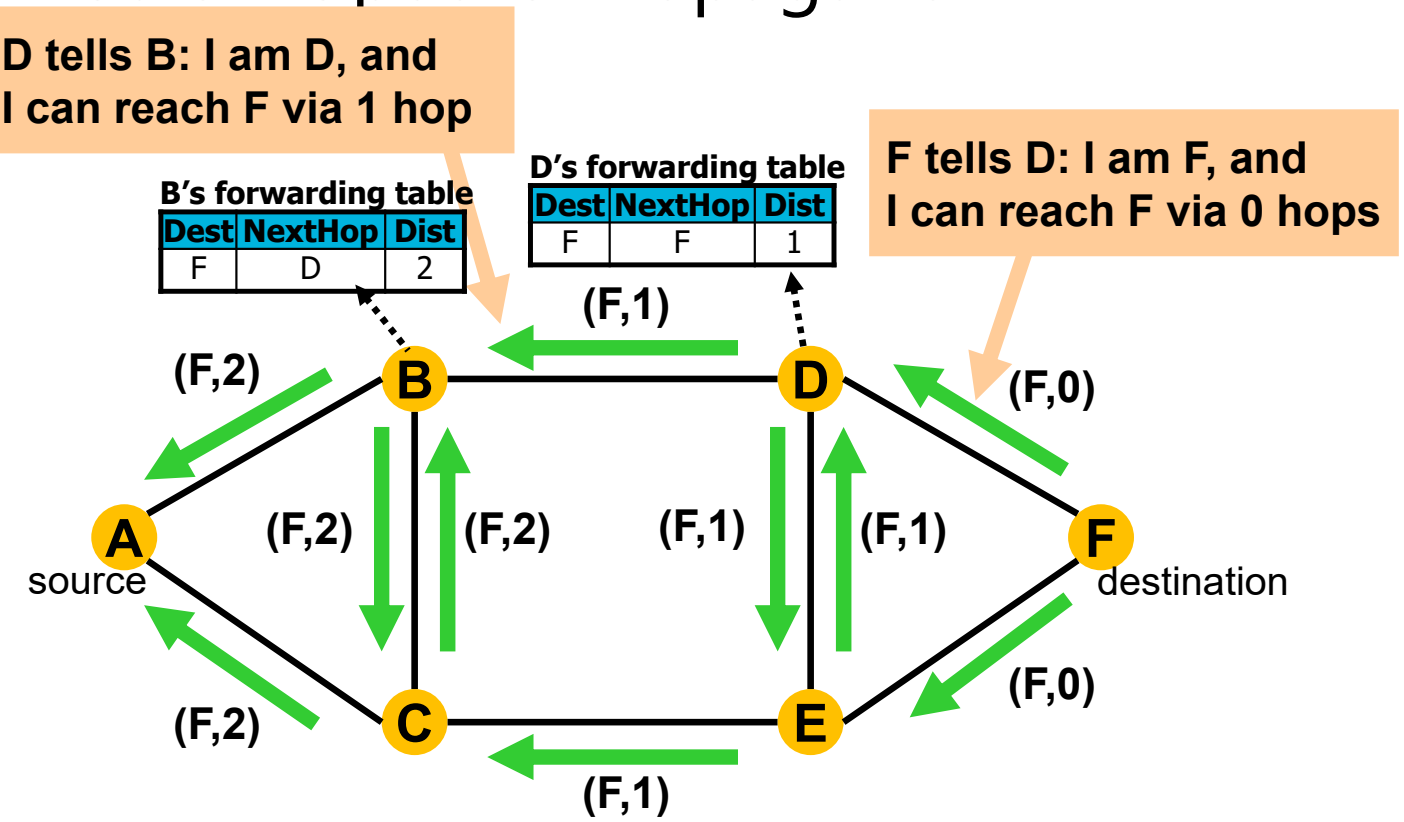


- In practice: shortest path precomputed, next-hops stored in forwarding table
- Downsides of link-state:
  - Lesser control on policy (certain routes can't be filtered), more cpu
  - Increased visibility (bad for privacy, but good for diagnostics)

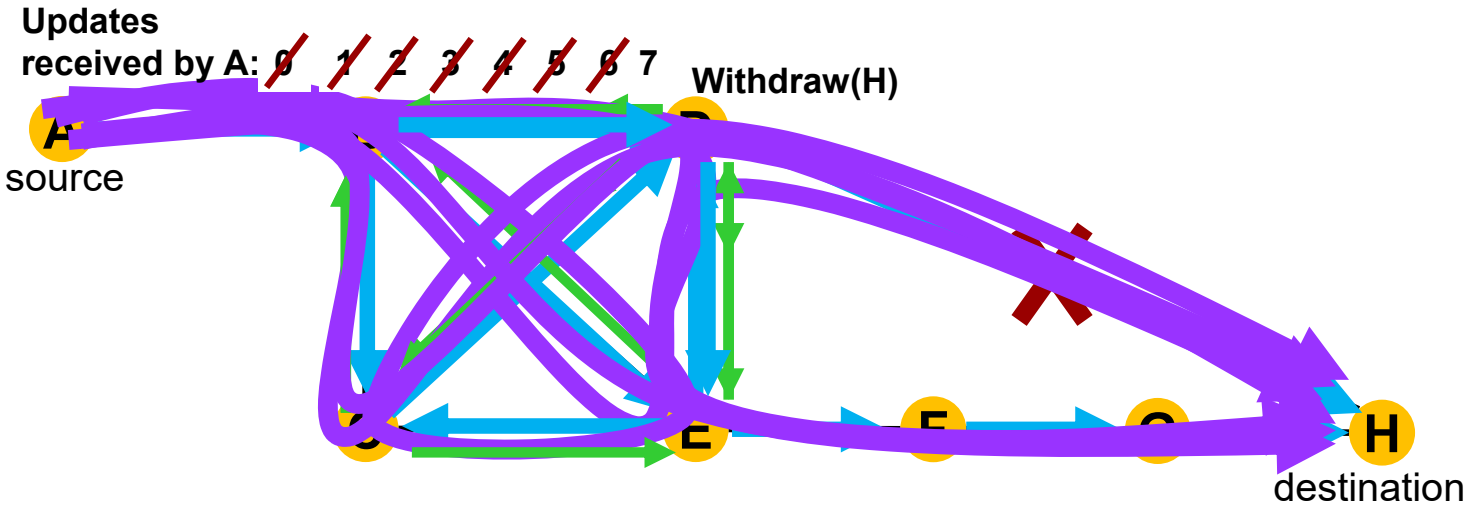
Distance Vector Routing

- Each router knows the links to its neighbors
  - Does *not* flood this information to the whole network
- Each router has provisional "shortest path" to every other router
  - E.g.: Router A: "I can get to router B with cost 11"
- Routers exchange this distance vector information with their neighboring routers
  - Vector because one entry per destination
- Routers look over the set of options offered by their neighbors and select the best one
- Iterative process converges to set of shortest paths

Distance Vector: Update Propagation



Distance Vector: Convergence



- How many updates would link-state require?
- Is link-state better or worse than distance vector?
- Idea: What if updates were destination-sequenced?

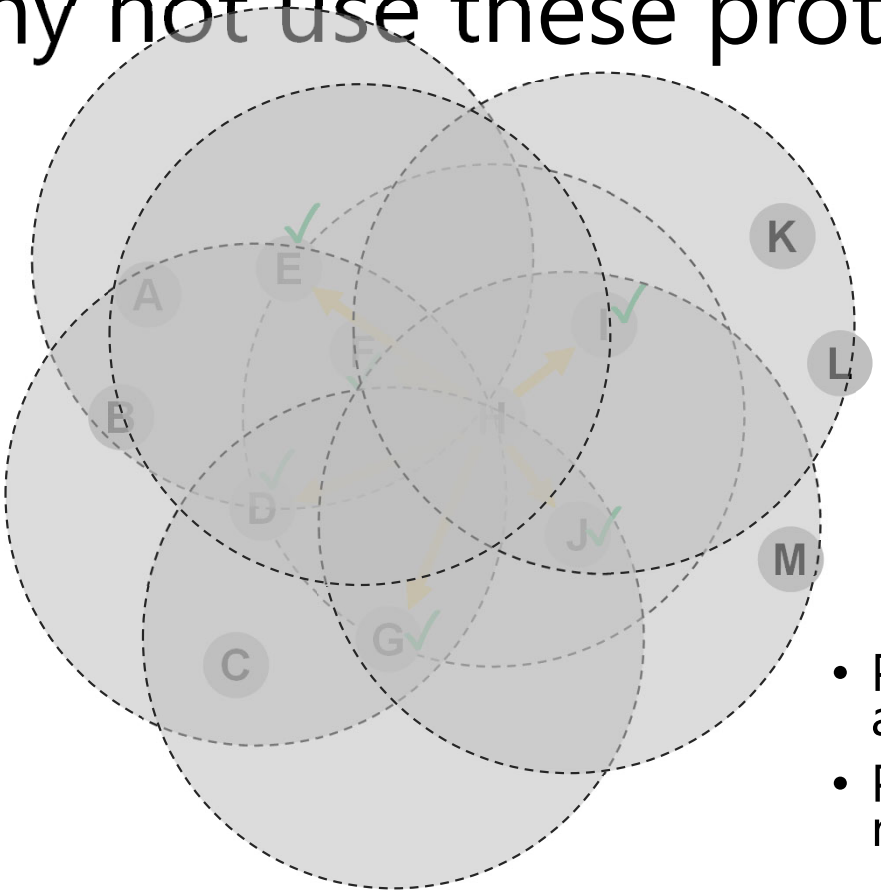
Mesh Routing

1. Addressing
   → How to identify destinations that may be multiple hops away?

2. Traditional Routing
   → How can nodes discover paths across intermediate hops?

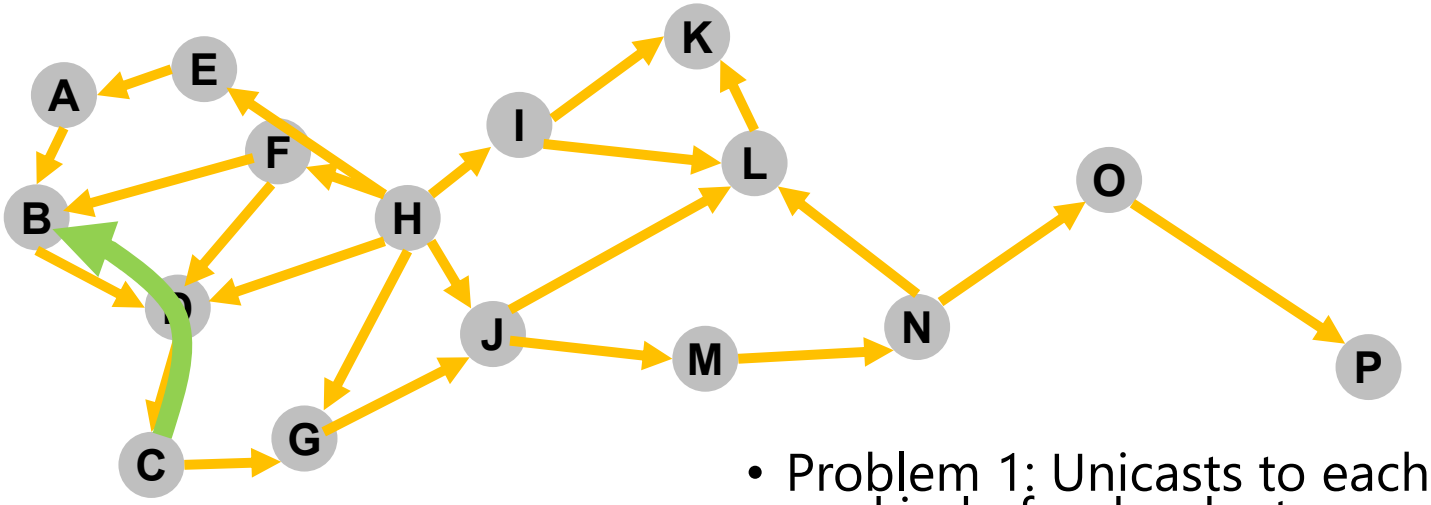
3. Mesh Routing
   → How to route in dynamic, unstable wireless environment?

Why not use these protocols in wireless?



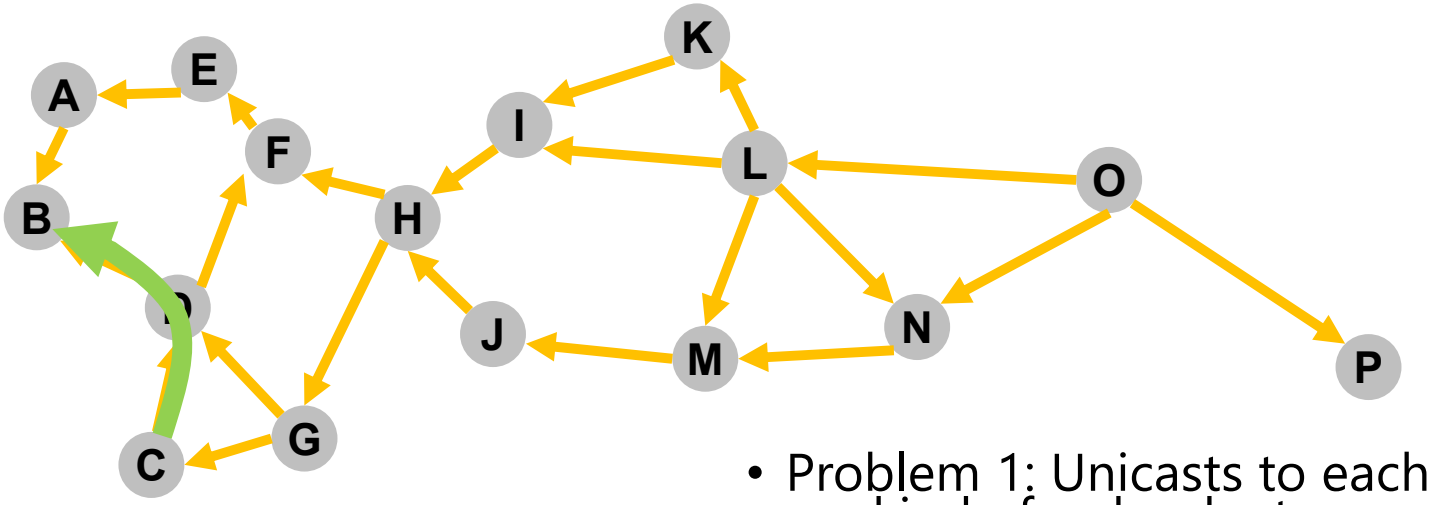
- Problem 1: Unicasts to each neighbor are kind of redundant
- Problem 2: Don't really need all neighbors to re-flood message

Why not use these protocols in wireless?



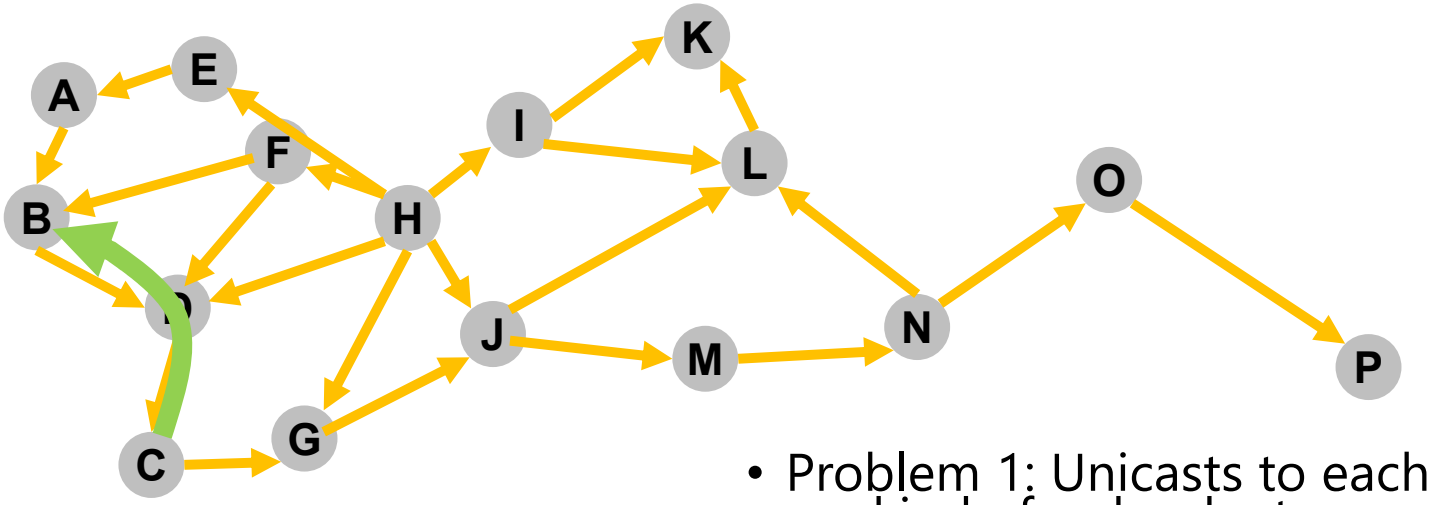
- Problem 1: Unicasts to each neighbor are kind of redundant
- Problem 2: Don't really need all neighbors to re-flood message

Why not use these protocols in wireless?



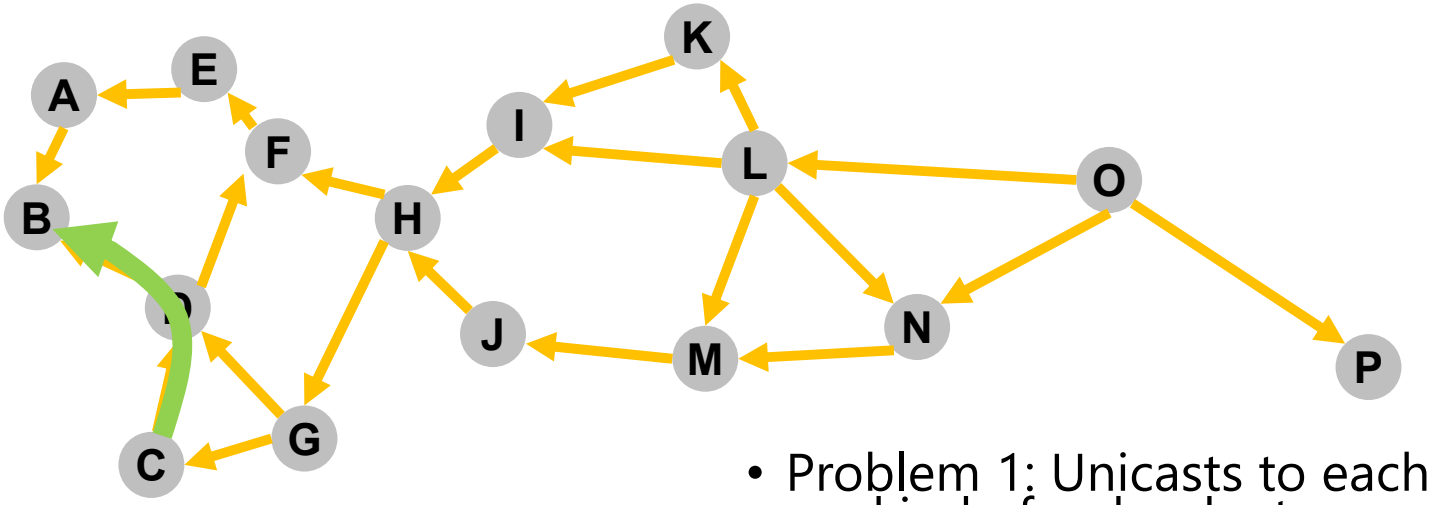
- Problem 1: Unicasts to each neighbor are kind of redundant
- Problem 2: Don't really need all neighbors to re-flood message

Why not use these protocols in wireless?



- Problem 1: Unicasts to each neighbor are kind of redundant
- Problem 2: Don't really need all neighbors to re-flood message

Why not use these protocols in wireless?



- Unnecessary transmissions waste battery, bandwidth
  - Unnecessary routes waste storage

- Problem 1: Unicasts to each neighbor are kind of redundant
- Problem 2: Don't really need all neighbors to re-flood message
- Problem 3: May not really need routes to all nodes

Solution: Mesh Networking

- Nodes cooperate to efficiently route data
- Nodes can act as relays, enabling multi-hop forwarding
- Mesh networks dynamically self-organize and self-configure
  - Reduce management and configuration overhead
  - Improves fault-tolerance
  - Dynamically distributes workloads

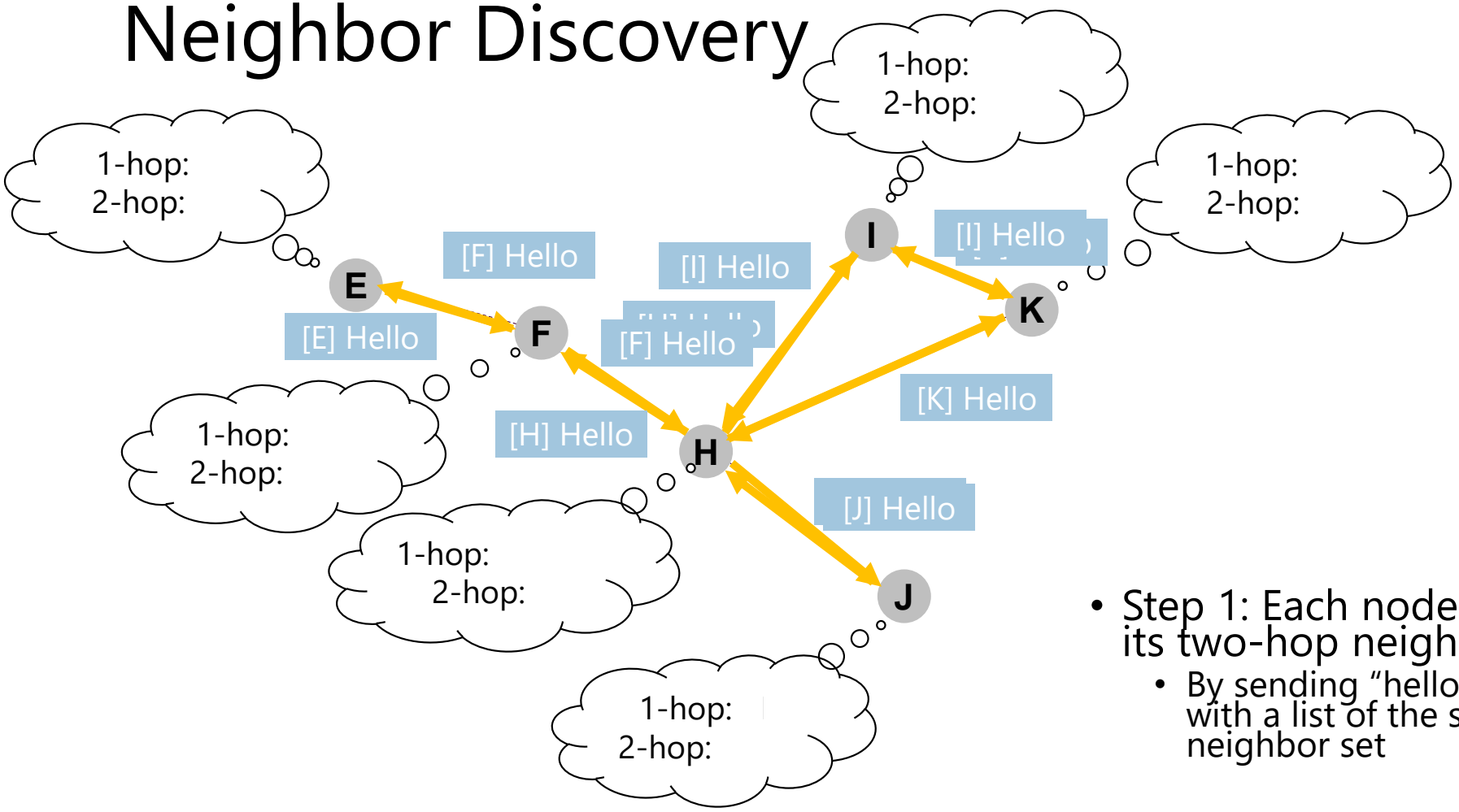
Mesh Networking Protocols

1. Optimized Link State Routing (OLSR)
2. Destination-Sequenced Distance Vector (DSDV)
3. Dynamic Source Routing (DSR)
4. Ad Hoc On-demand Distance Vector (AODV)
5. Tree Routing
6. Geographic/Perimeter Routing
7. Gossip Routing

Optimized Link State Routing (OLSR)

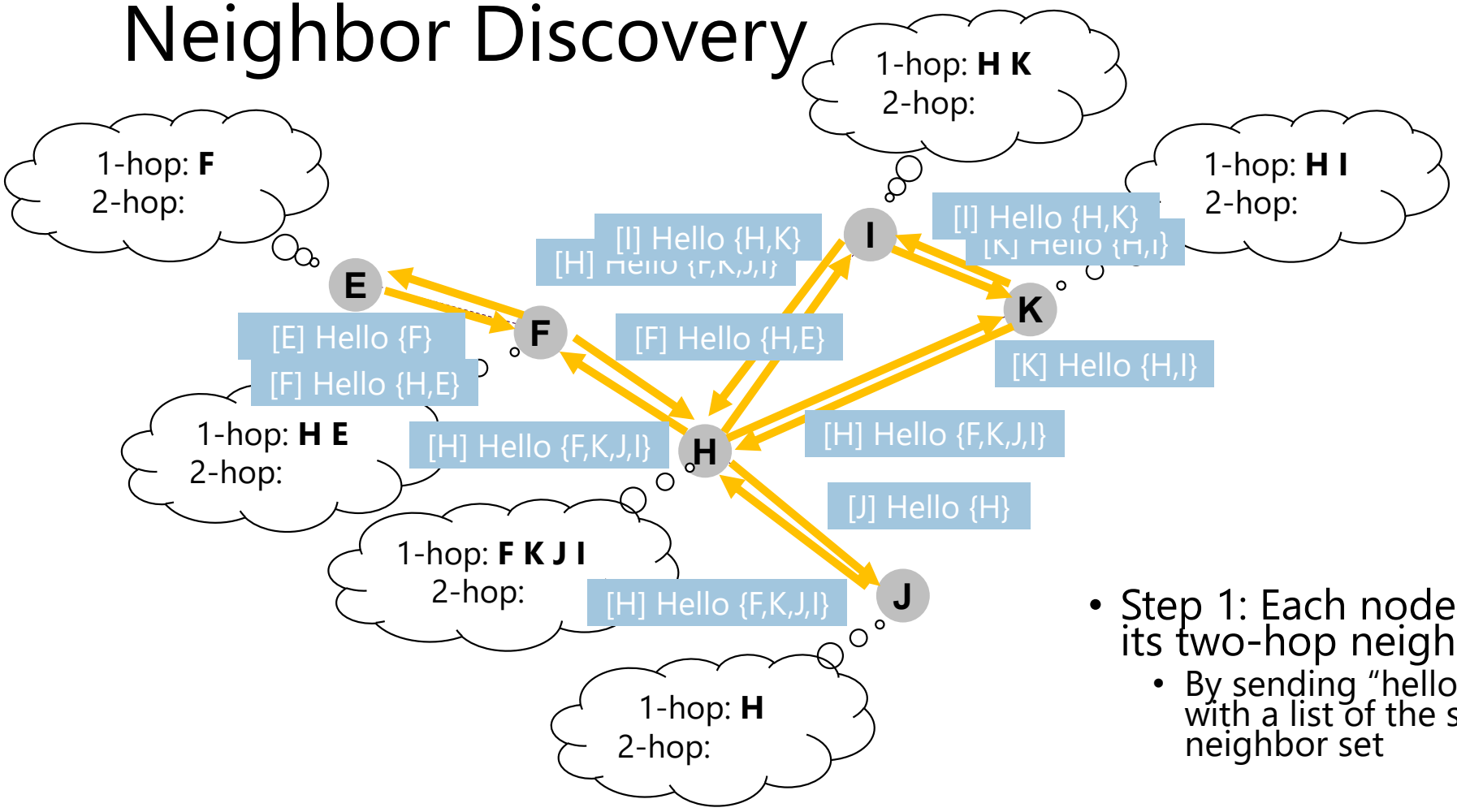
- Main idea: Do link state, but avoid unnecessary transmissions of link state updates
  - All nodes will still receive all link state updates
- Nodes broadcast; reach all neighbors with one message
- Each node decides which of its neighbors will flood
  - These neighbors are known as multipoint relays (MPRs)
  - MPRs selected based on nodes neighbors can reach
  - Only MPRs transmit updates, other nodes do not

Optimized Link State Routing (OLSR): Neighbor Discovery



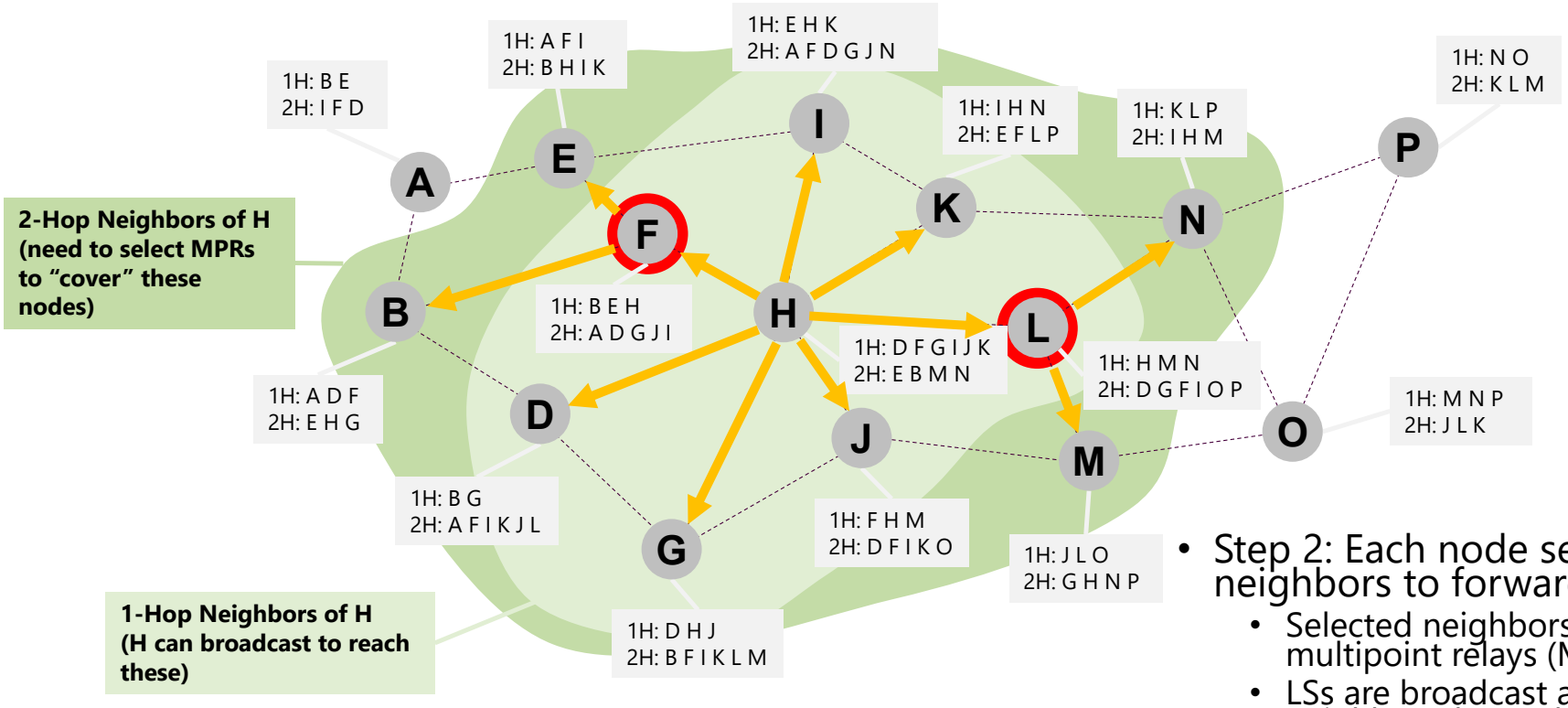
- Step 1: Each node discovers its two-hop neighbors
  - By sending "hello" packets with a list of the sender's neighbor set

Optimized Link State Routing (OLSR): Neighbor Discovery



- Step 1: Each node discovers its two-hop neighbors
  - By sending "hello" packets with a list of the sender's neighbor set

Optimized Link State Routing (OLSR): MPR Selection



- Step 2: Each node selects subset of neighbors to forward its link states
  - Selected neighbors are called multipoint relays (MPRs)
  - LSs are broadcast and received by all neighbors, but only MPRs forward them onward

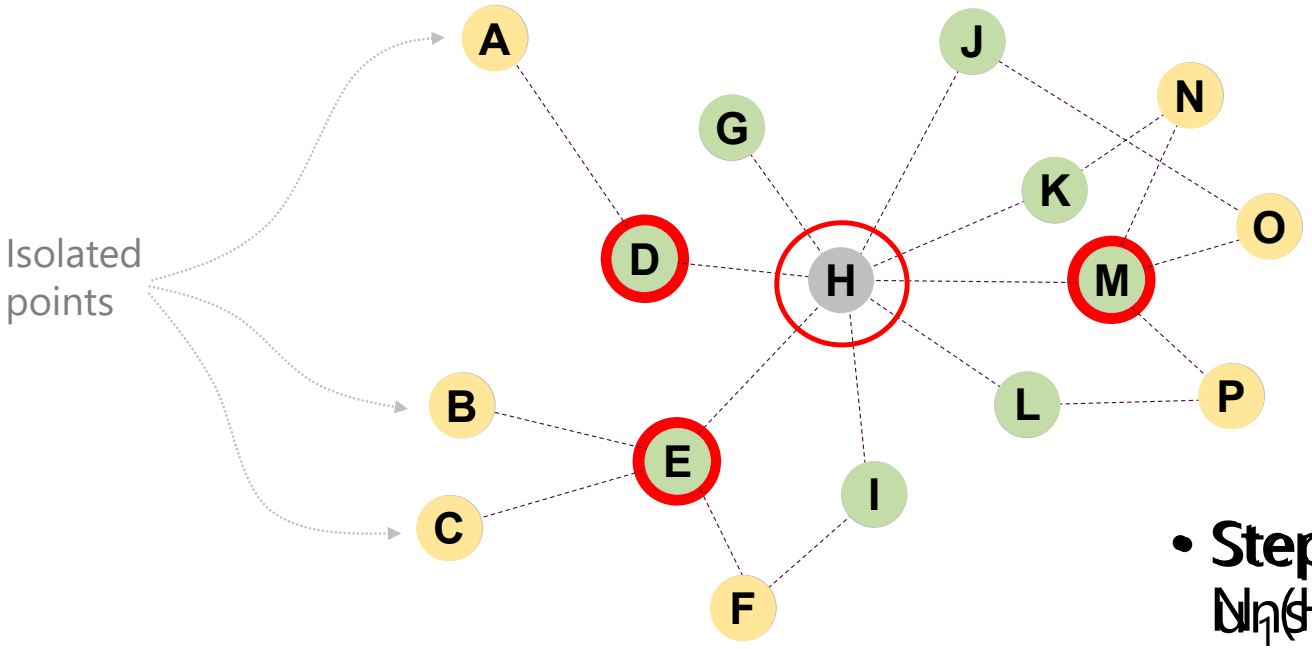
Optimized Link State Routing (OLSR): MPR Selection Algorithm

- **Goal:**
  - Given a node H, with 1-hop neighborhood $N_1(H)$, and 2-hop neighborhood $N_2(H)$
  - Select within $N_1(H)$ the smallest possible set of nodes that covers the entire set of nodes in $N_2(H)$

- **Algorithm:**
  - Step 1: Select all nodes in $N_1(H)$, which cover *isolated points* of $N_2(H)$
    - Isolated points are nodes that are reachable by only one member of $N_1(H)$
  - Step 2: Amongst all unselected nodes in $N_1(H)$, select the node, which covers the maximum number of uncovered nodes in $N_2(H)$
  - Step 3: Repeat Step 2 until all nodes are covered

- This is just a heuristic (optimal/minimum MPR set is NP-complete)

Optimized Link State Routing (OLSR): MPR Selection



- **Step 1:** Select all nodes in $N_1(H)$ which cover *isolated points* of $N_2(H)$
- **Step 2:** Select nodes in $N_1(H)$ which cover most nodes in $N_2(H)$

Optimized Link State Routing (OLSR): Forwarding

- Same as regular link state: Build a topology database, run Dijkstra's, store shortest-path next-hop for each destination
- However, observe that nodes tend to select higher-connected nodes as MPRs
  - Therefore, nodes tend to select same nodes as MPRs
  - MPRs tend to form a smaller "spanning" subgraph of the topology
- Optimization: MPRs forward all traffic, not just link state adverts
  - Reduces routing table size
  - Allows more nodes to sleep
  - But, asymmetric battery usage

What's wrong with OLSR?

- Great approach, used in some protocols

- However, it maintains routes to all nodes, all the time
  - Good if most nodes need to talk to each other
  - Wastes overhead if communication is more sparse/rare
  - Also: Reveals the entire topology to all nodes → bad for privacy

- Idea: What if we only build routes "on demand" when we need them?
  - Idea behind "reactive" routing
  - Examples: AODV, DSR

Dynamic Source Routing (DSR)

- When network starts, no routing updates are sent
  - Network performs a discovery process only when data needs to be sent
- When packet arrives at an interface, node floods “route request” (RREQ) message for destination
- Message records list of intermediate hops in the packet

Dynamic Source Routing (DSR)

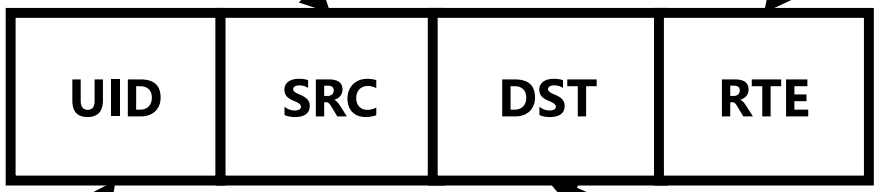
- When destination receives RREQ, it replies back to the source with a RREP message
  - RREP sent back along reverse of path contained in RREQ
- When source receives the RREP, it now knows a "source route" to the destination
  - It then sends the data, putting the source route in the header
  - Intermediate hops determine the next hop by inspecting the source route
  - Source caches the source route in case it needs it later

Dynamic Source Routing: Route Request (RREQ)

Source – node sending the RREQ

Route – the source route learned by the RREQ

Control packet format:

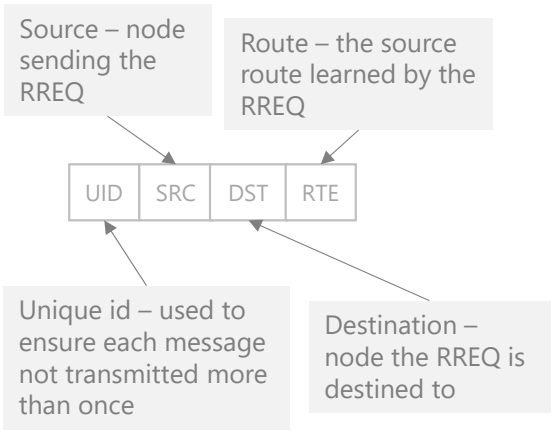


Unique id – used to ensure each message not transmitted more than once

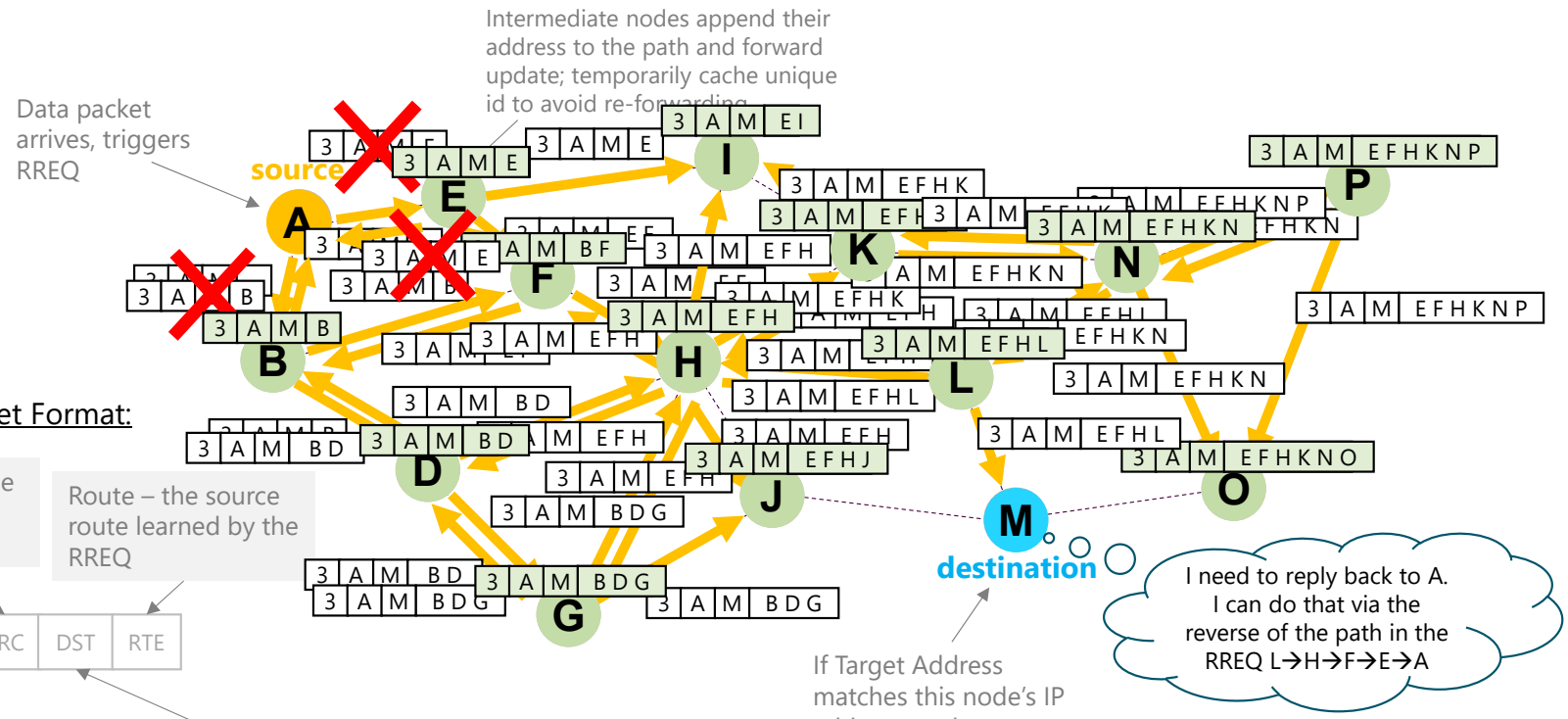
Destination – node the RREQ is destined to

Dynamic Source Routing: Route Request (RREQ)

RREQ Packet Format:



Dynamic Source Routing: Route Request (RREQ)



RREQ Packet Format:

Source – node sending the RREQ

Route – the source route learned by the RREQ

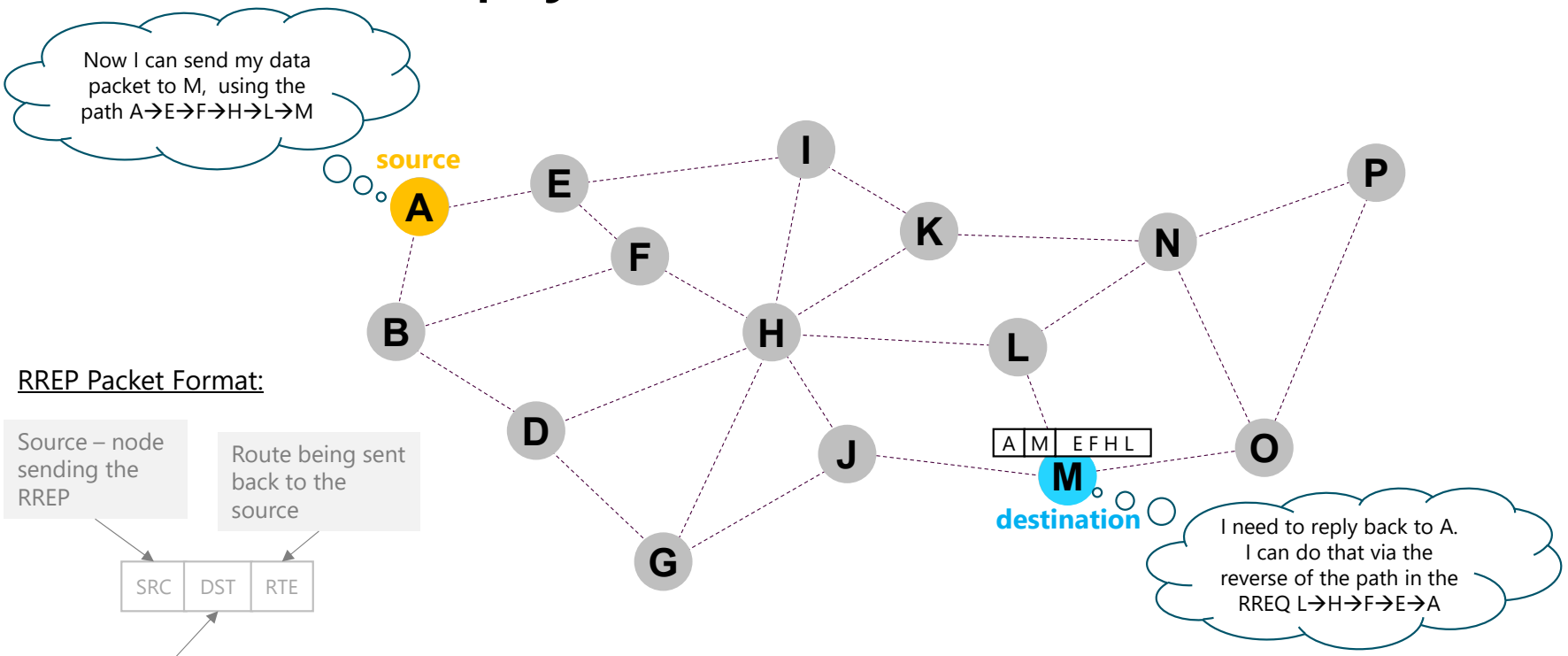
UID	SRC	DST	RTE

Unique id – used to ensure each message not transmitted more than once

Destination – node the RREQ is destined to

- Step 1: On data packet arrival, node sends RREQ to discover path to destination

Dynamic Source Routing: Route Reply (RREP)



RREP Packet Format:

Source – node sending the RREP

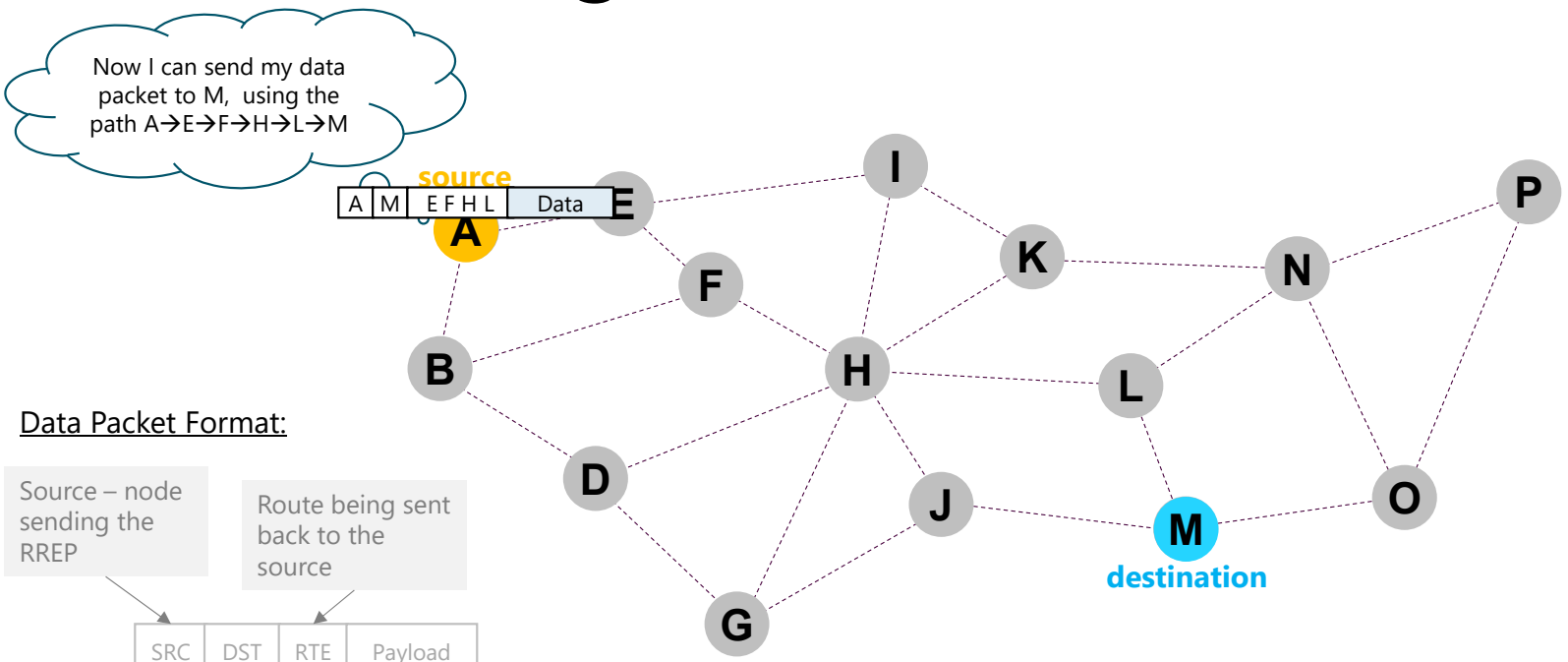
Route being sent back to the source

SRC	DST	RTE

Destination – node the RREP is destined to

- Step 2: Upon receiving an RREQ, destination returns discovered path back to the source in an Route Reply (RREP)

Dynamic Source Routing: Forwarding



Data Packet Format:

Source – node sending the RREP

Route being sent back to the source

SRC	DST	RTE	Payload

Destination – node the RREP is destined to

Payload – data being sent in packet

- Step 3: Source can then send data using source route contained in RREP

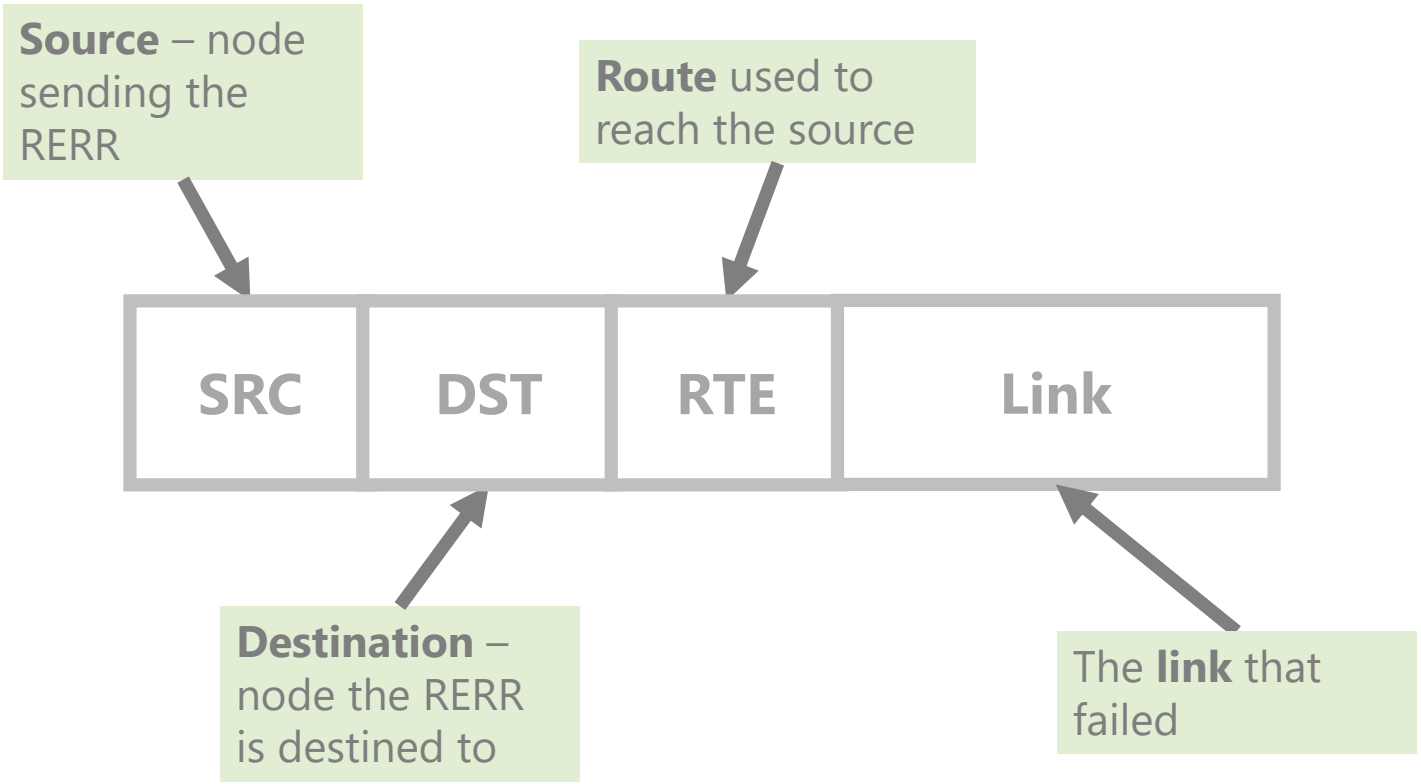
Dynamic Source Routing: Route Caching

- Source will cache route for some period of time, in case it wants to send more packets to that same destination later
  - Intermediate nodes and other nodes overhearing RREQs and RREPs may also cache paths they see
  - Entries are deleted after timeout (tunable parameter)
  - Route cache size can be limited (tunable parameter)

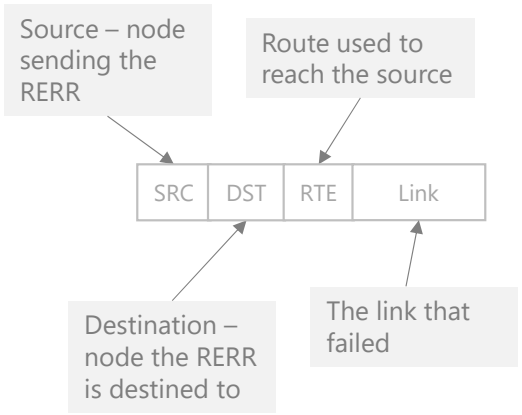
Dynamic Source Routing: Route Caching

- If an intermediate node receives an RREQ for which it has a cached route, it may reply even if it is not the final destination
  - Leads to “RREP Storms” where all nodes’ neighbors reply with an RREP simultaneously
  - Solution: RREP may be sent with some delay while listening if another node replies with a shorter route

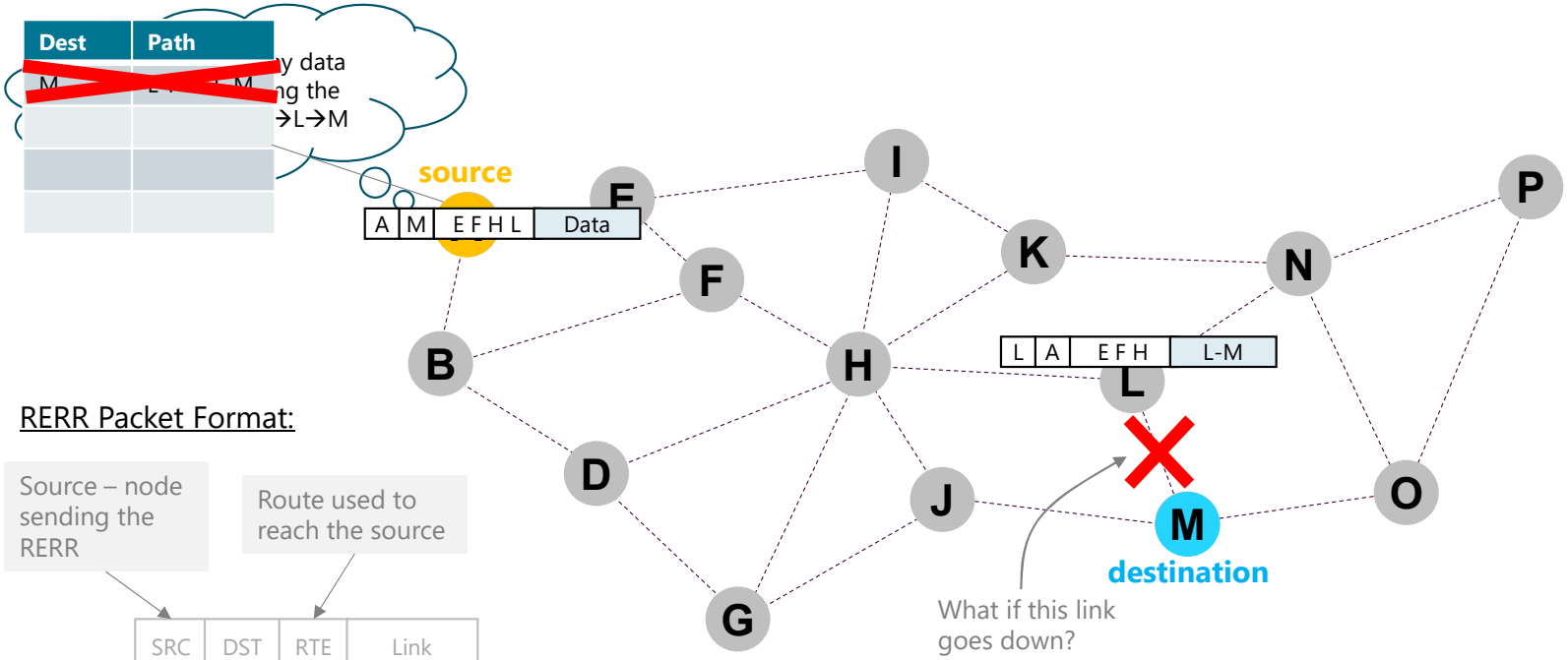
Dynamic Source Routing: Route Error (RERR)



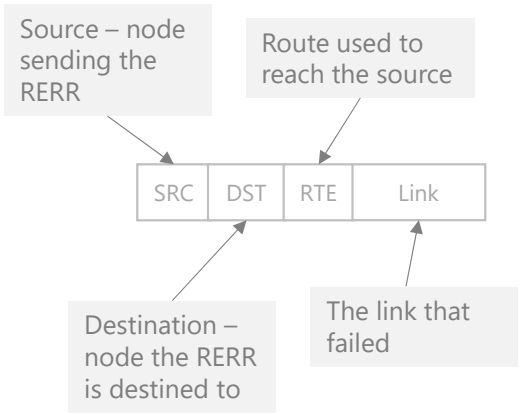
Dynamic Source Routing: Route Error (RERR)



Dynamic Source Routing: Route Error (RERR)



RERR Packet Format:



- Route Error: Sent back when data packet encounters a failed link in the path
  - Keeps route caches up-to-date in presence of churn

What's wrong with Dynamic Source Routing?

- Nothing, it's a great protocol
  - Used in Zigbee, etc.
- However, as network gets big, source routes get long
  - Lots of packet overhead
- Also caches can get stale
  - Outages not discovered until packets sent
- Idea: What if we had each intermediate hop store a pointer
  - More like distance vector, embed the path in the network

Ad Hoc On-Demand Distance Vector (AODV)

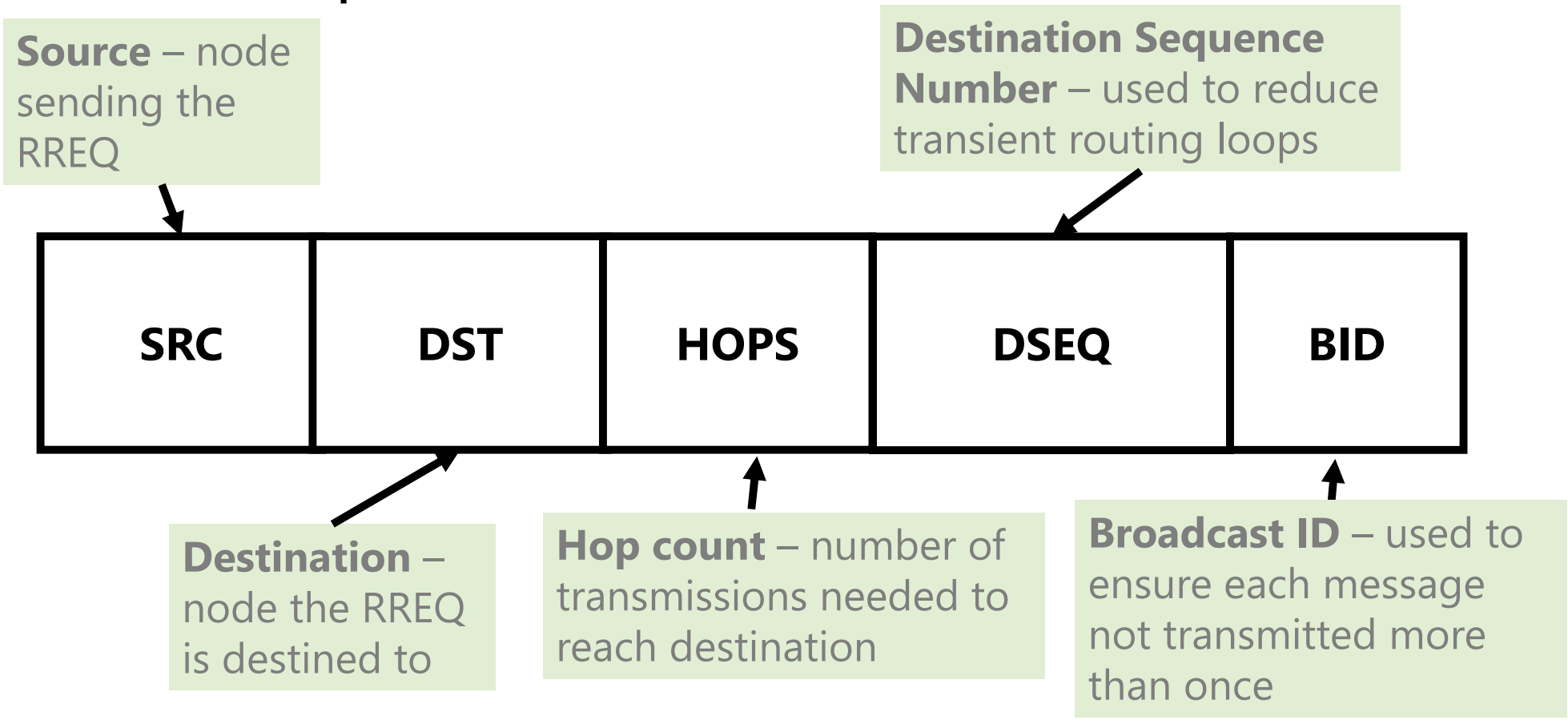
- Just like DSR, no route discovery process until data needs to be sent (reactive routing)

- Just like DSR, RREQ sent to find a route
  - But, no source routes are collected
  - Instead, RREQ creates distance-vector entries at each hop pointing back to source
  - RREP used to set up forward path, traverses entries RREQ created to reach source

Ad Hoc On-Demand Distance Vector (AODV)

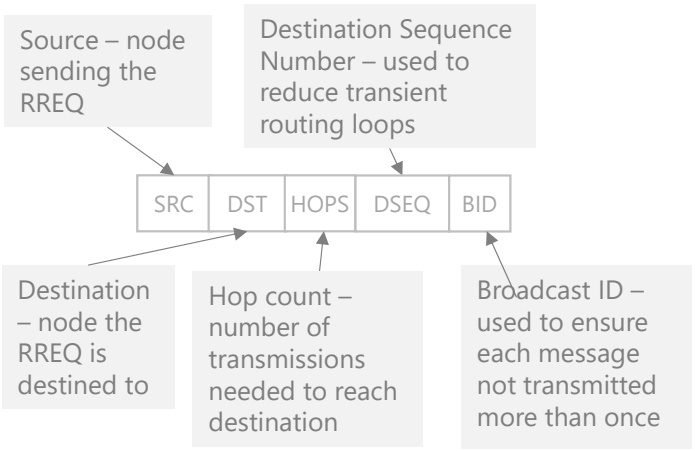
- Nodes on active path maintain routing information
  - Table entries are expired if not used recently by data packets
- Destination-controlled sequence number stored with each route
  - Prevents storing of stale routes
  - Also mitigates “count-to-infinity” problem and routing loops

Ad Hoc On-Demand Distance Vector (AODV): Route Request (RREQ)

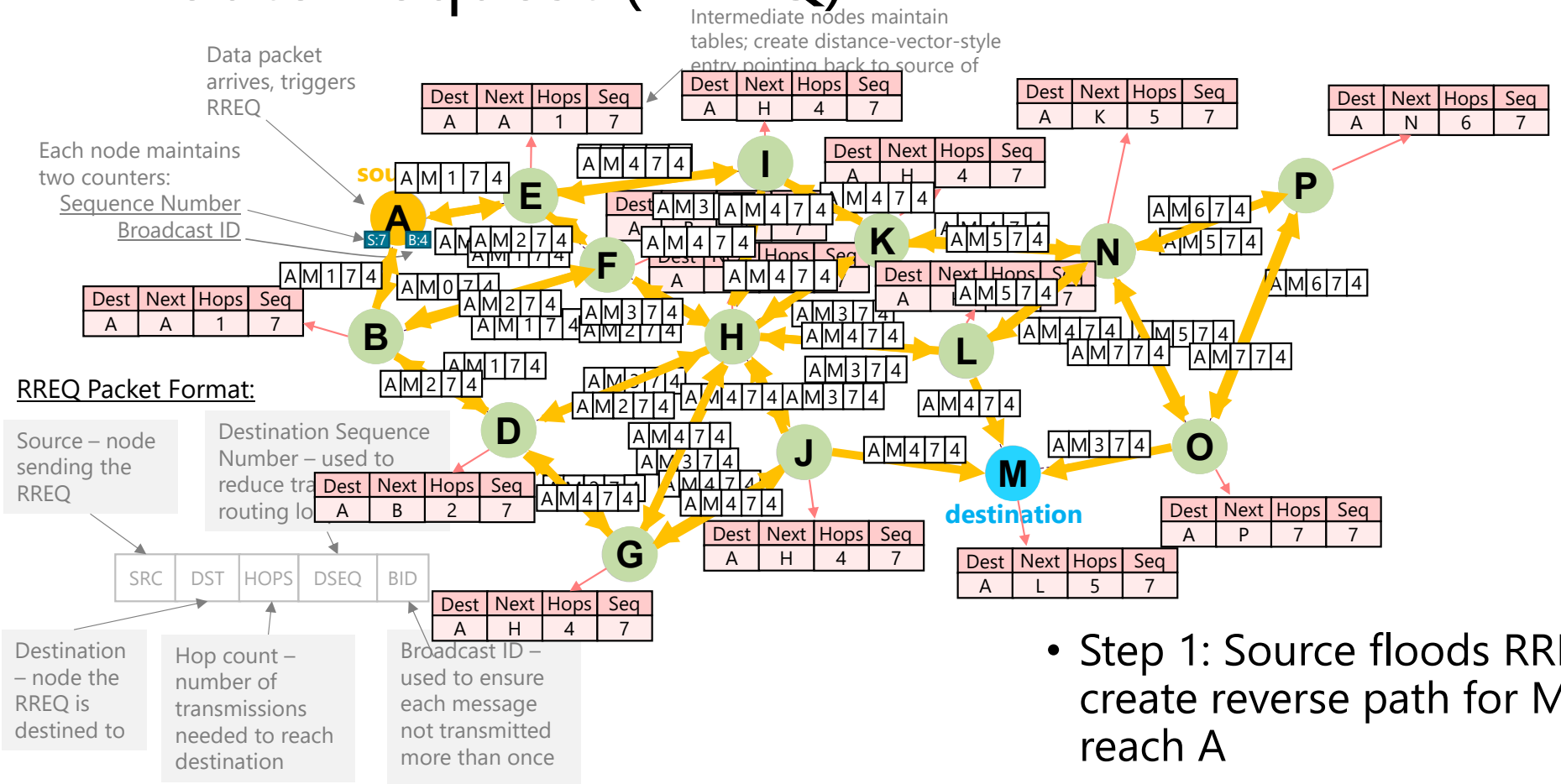


Ad Hoc On-Demand Distance Vector (AODV): Route Request (RREQ)

RREQ Packet Format:

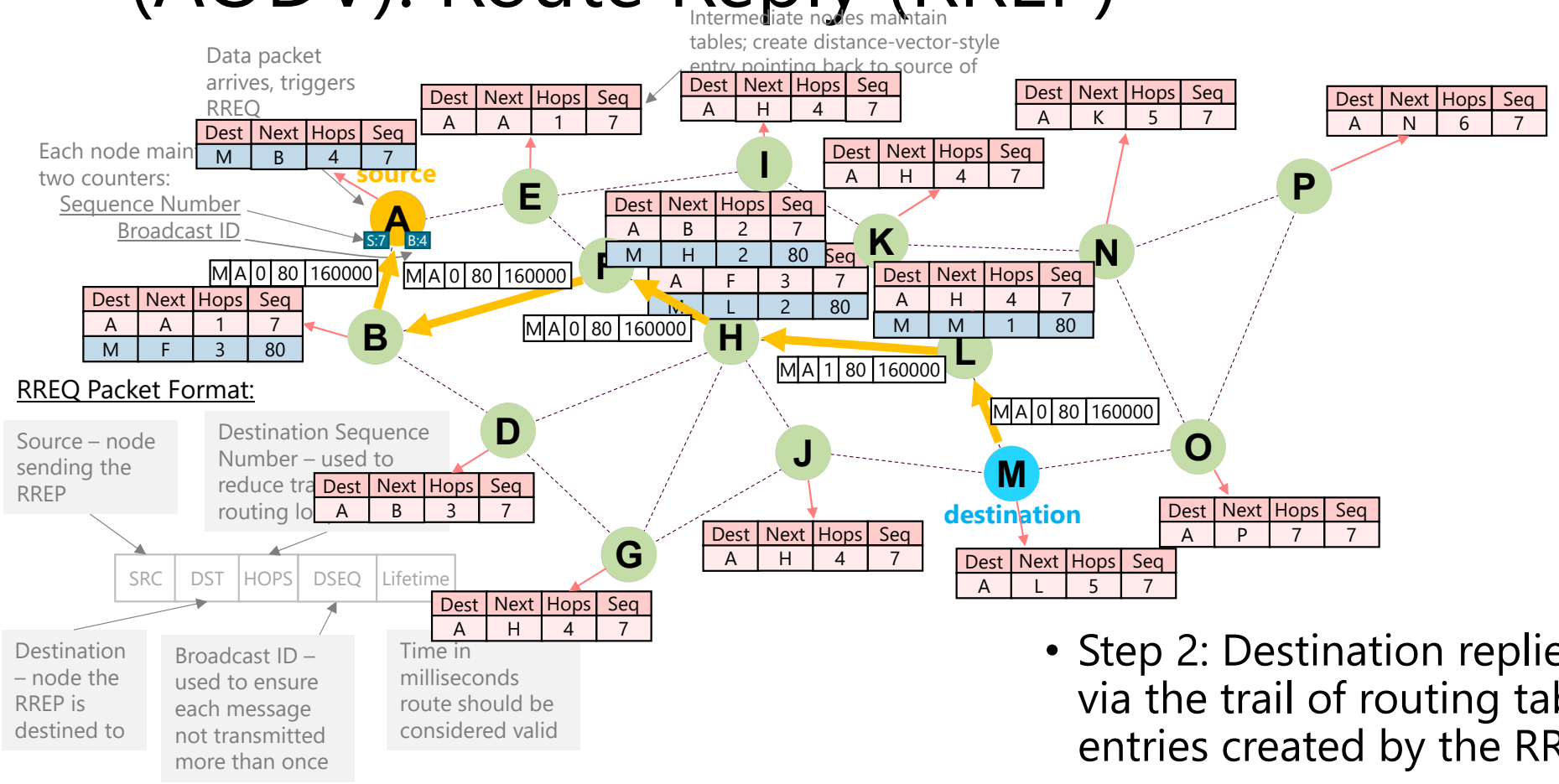


Ad Hoc On-Demand Distance Vector (AODV): Route Request (RREQ)



- Step 1: Source floods RREQ to create reverse path for M to reach A

Ad Hoc On-Demand Distance Vector (AODV): Route Reply (RREP)



- Step 2: Destination replies back via the trail of routing table entries created by the RREQ

Ad Hoc On-Demand Distance Vector (AODV): Route Error (RERR)

- Withdrawal message, sent when:
  - Node detects a link break for next hop of routing table entry
  - Receives data packet for which it has no route
  - It receives an RERR pertaining to one of node's active routes

- Nodes have the option to locally repair
  - Instead of forwarding RERR, initiate RREQ for affected destination
  - Only attempted if dest's # hops away < MAX_REPAIR_TTL
  - Can inflate path lengths over time

Hierarchical Addressing: Zigbee's Distributed Addressing Scheme

Example:

- Max depth L=2
- Max routers R=4
- Max children C=3

$S(d)=(CR^{L-d-1}-1-C+R)/(R-1)$

$S(0)=(3*4^{2-0-1}-1-3+4)/(4-1)=4$

- (Children of level 0 will have a pool of 4 addresses)

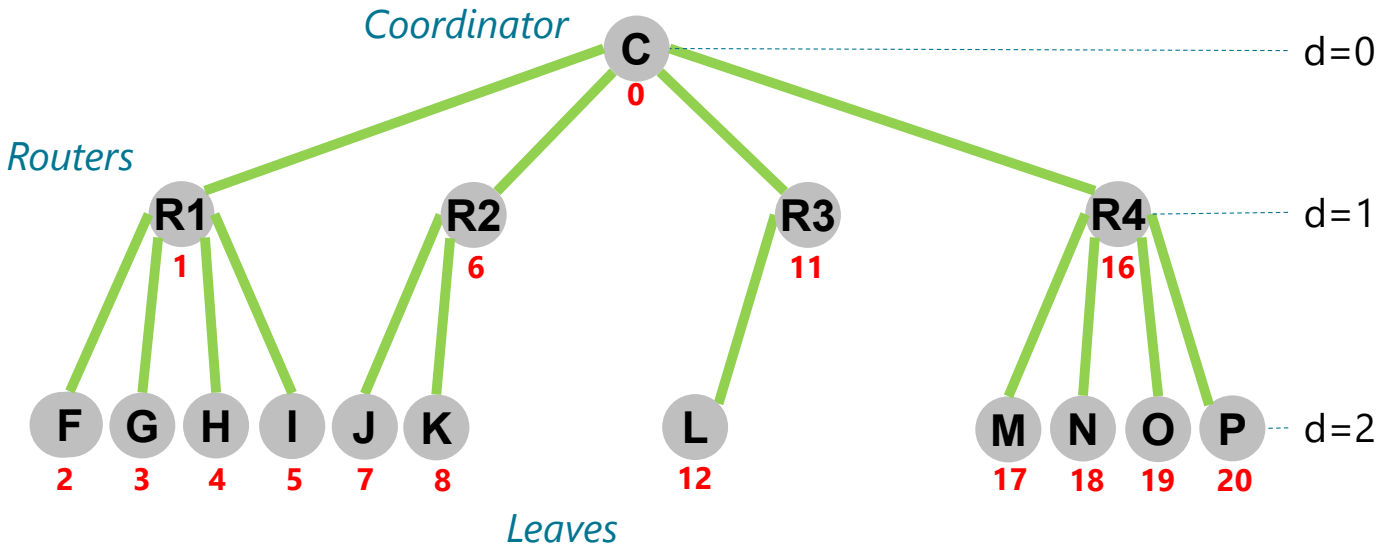
$S(1)=(3*4^{2-1-1}-1-3+4)/(4-1)=1$

- (Children of level 1 will have a pool of 1 address)

Address of R1: 0+1+0=1

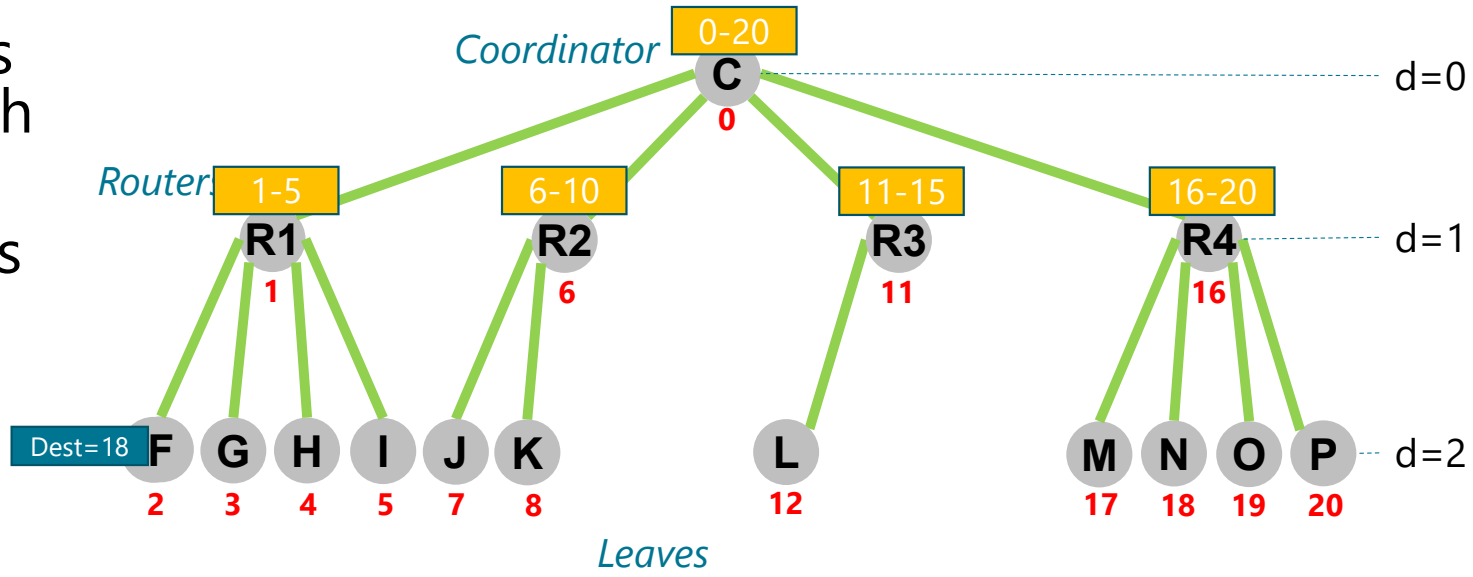
Address of R2: 0+1+1+S(0)=6

Address of R3: 0+1+2+2*S(0)=11



Tree Routing

- Each node knows subrange for each of children
- Each node checks to see if destination address is in children's subranges
  - If so, sends to appropriate child
  - If not, sends to parent



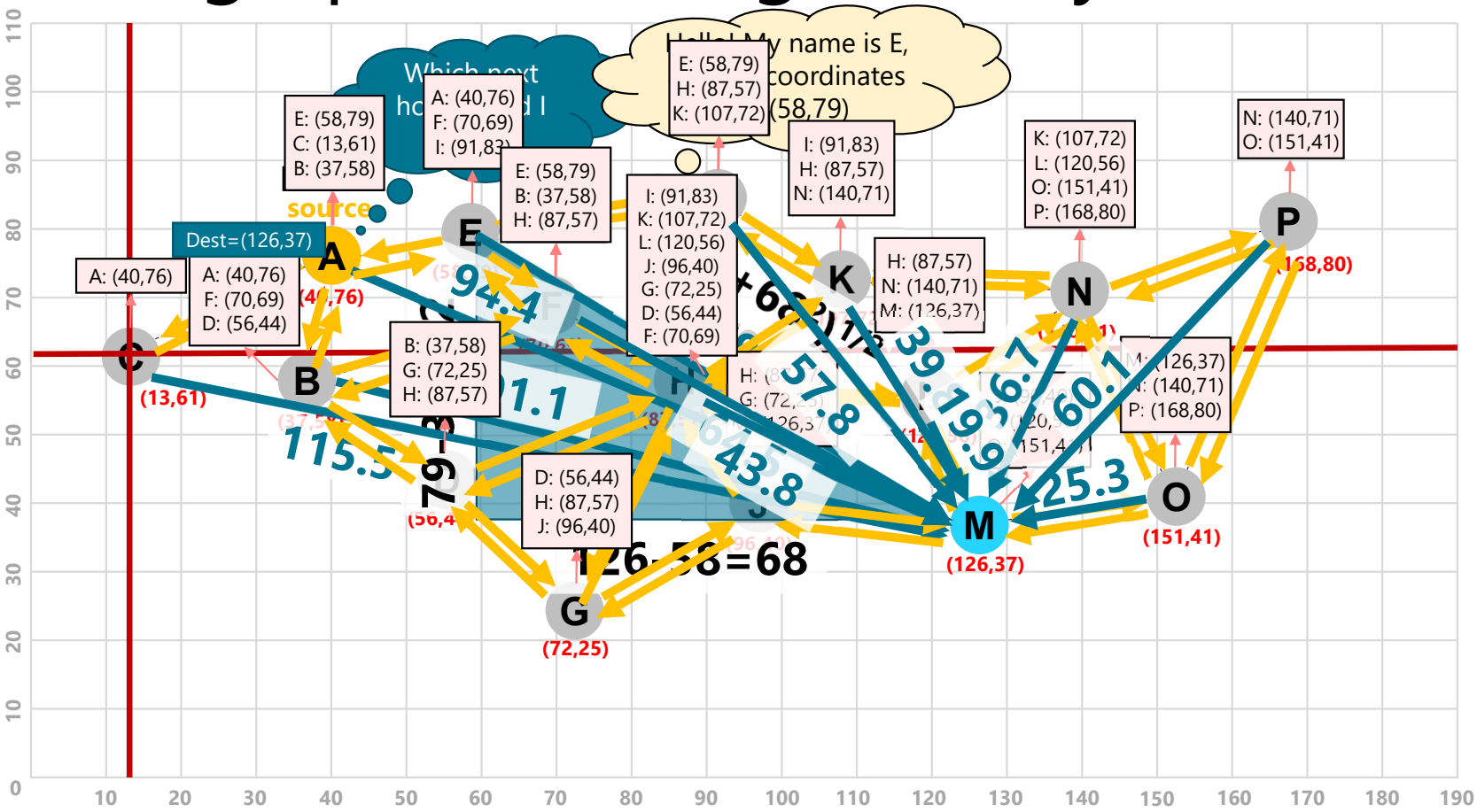
Geographic Routing

- Uses geographic position information to make progress to destination
- Source sends message to geographic location of the destination
- Each node keeps track of geographic location of its neighbors, so it knows which neighbor makes most progress to destination

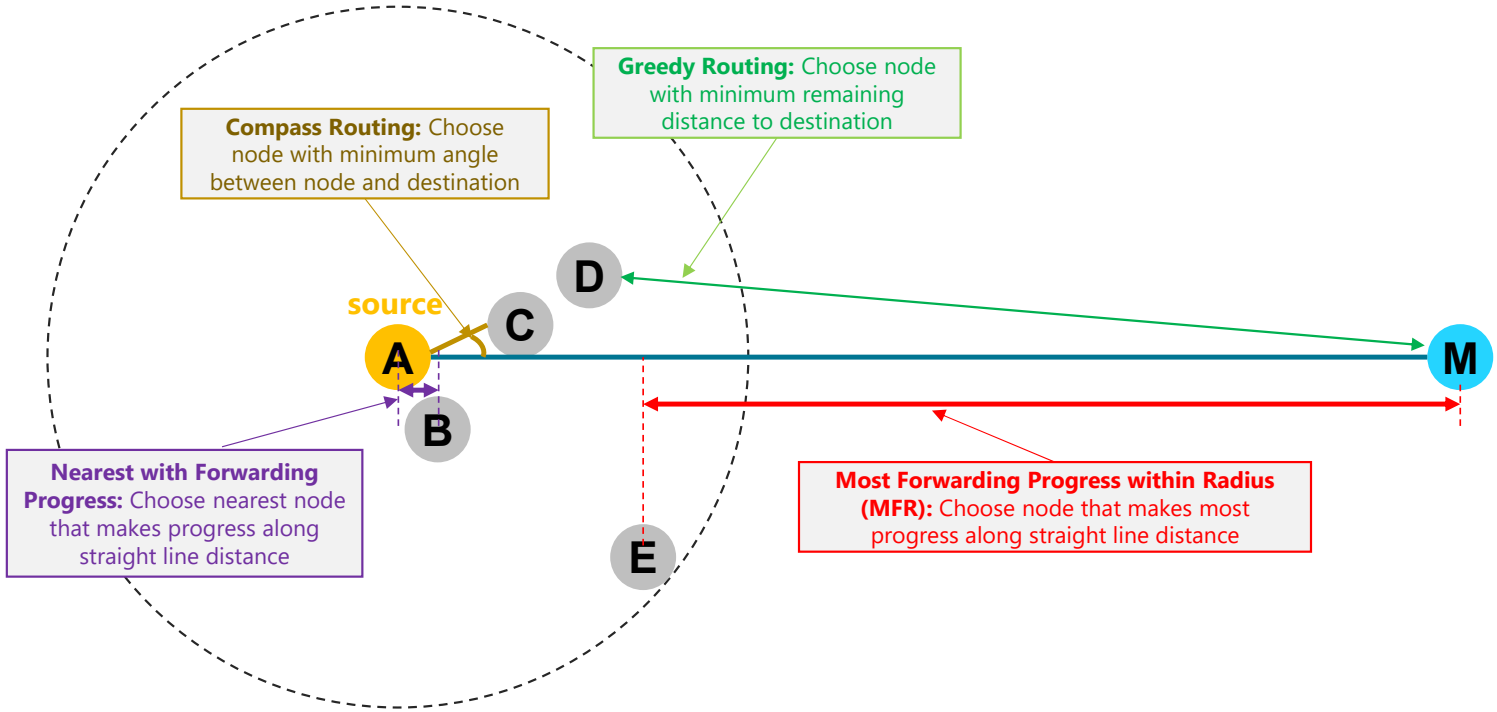
Geographic Routing

- More complex than you might think
  - You can get stuck in dead ends (“voids”)
  - You can get stuck in loops (two neighbors think each other makes the most progress to the destination)
- Example implementation of Geographic routing:
  - Greedy Perimeter Stateless Routing (GPSR)

Geographic Routing: Greedy Forwarding

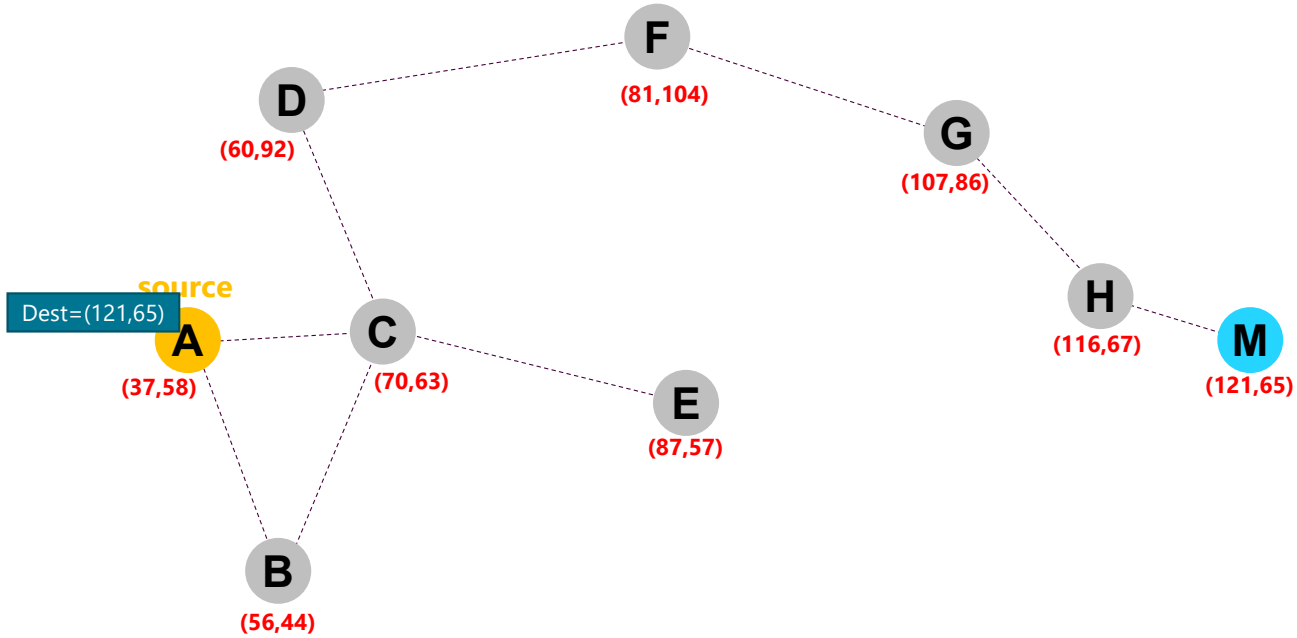


Geographic Routing: Variants of Greedy Forwarding



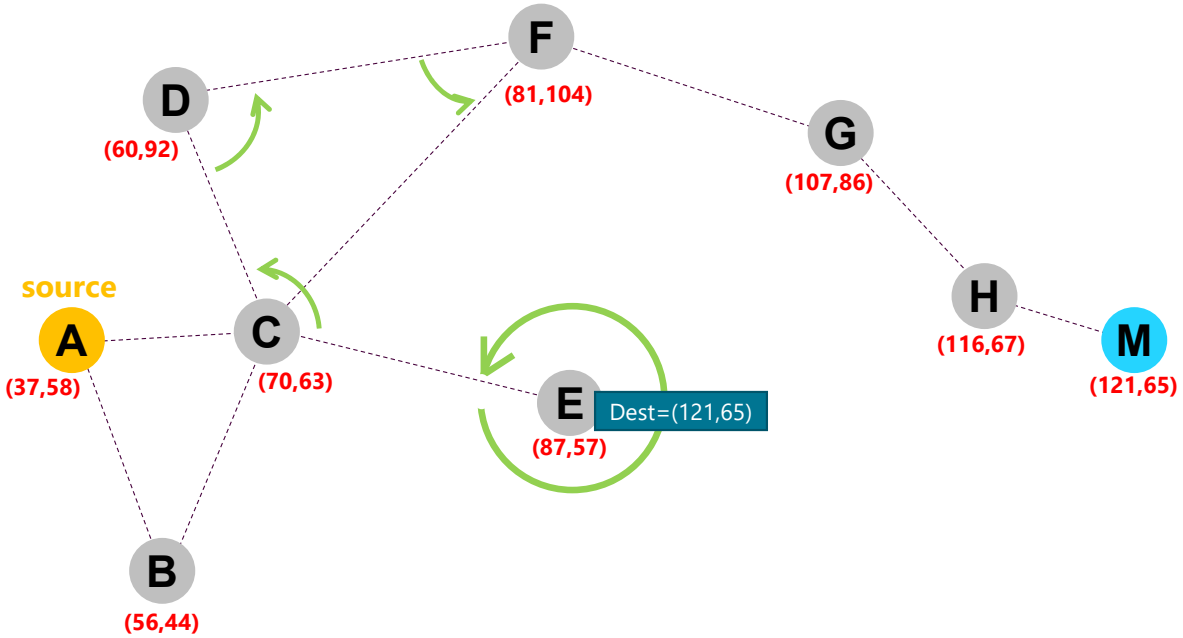
- How should we measure "progress" to the destination?

Geographic Routing:
Does greedy routing guarantee delivery?



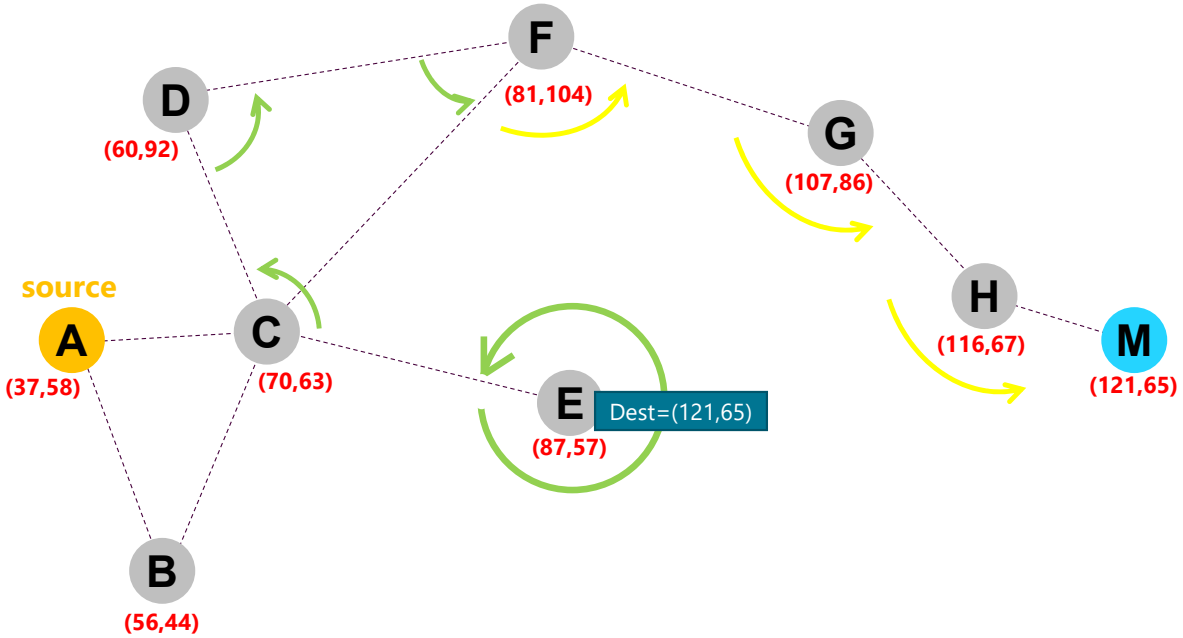
- Problem: Dead ends (“Voids”)

Geographic Routing: Perimeter Forwarding



- Problem: Perimeter forwarding breaks when there are closed faces
- Solution: “Face routing” – switch “faces” when we start going backwards

Geographic Routing: Face Routing



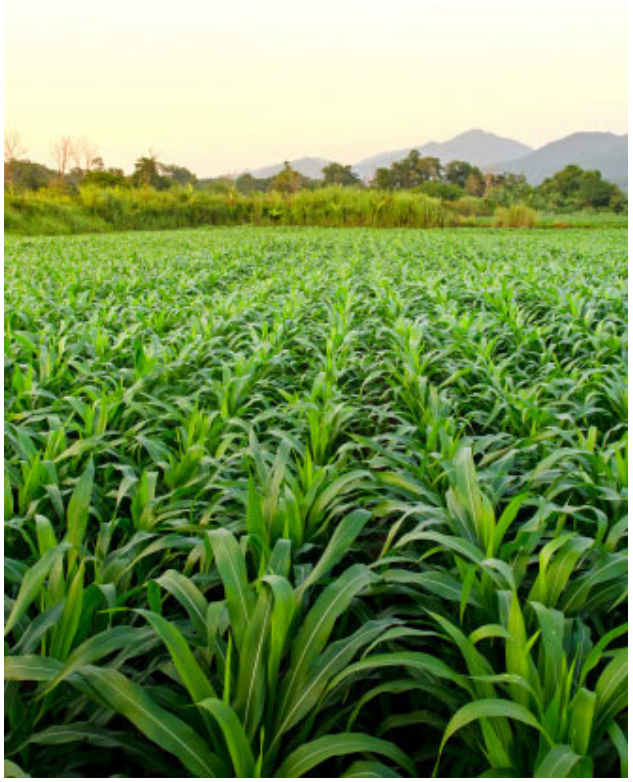
- Problem: Perimeter forwarding breaks when there are closed faces
- Solution: “Face routing” – switch “faces” when we start going backwards

Geographic Routing: Making Perimeter Forwarding Work

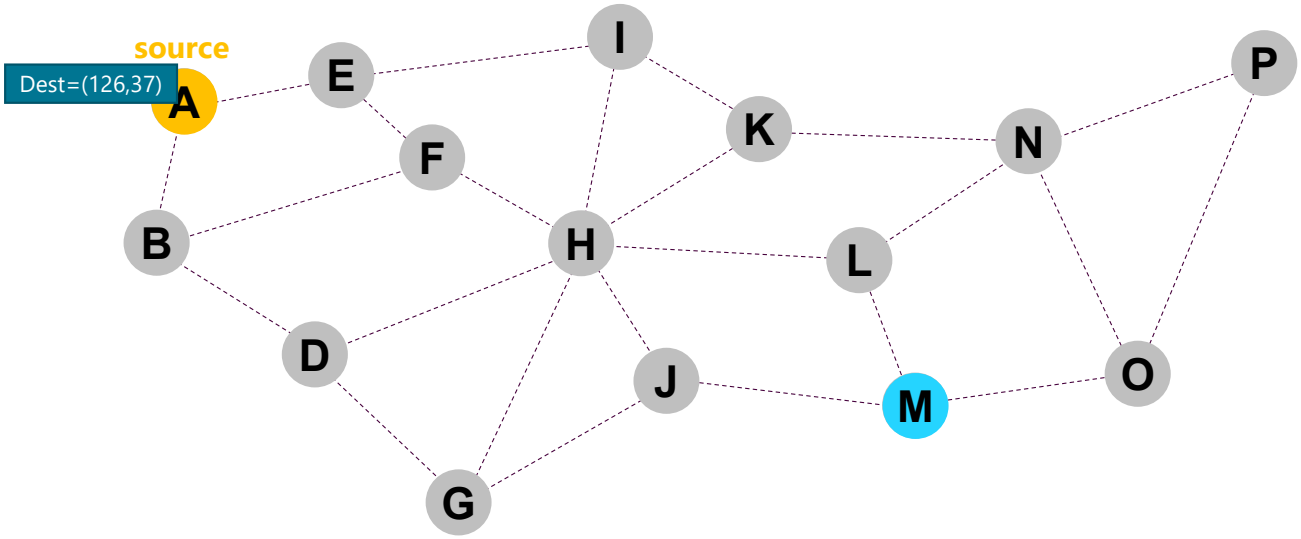
- Face Routing can fail if graph is non-planar (cross-edges)
- To address this, we "Planarize" the graph
  - Remove all cross edges
- There are distributed protocols to do this
  - Work by removing edges from the graph
- Example: **Relative Neighborhood Graphs**
  - Each edge (u,v) is allowed to remain in the graph only if there does not exist another connected node r that is closer to both u and v than they are to each other

Coming back to our motivating scenario…

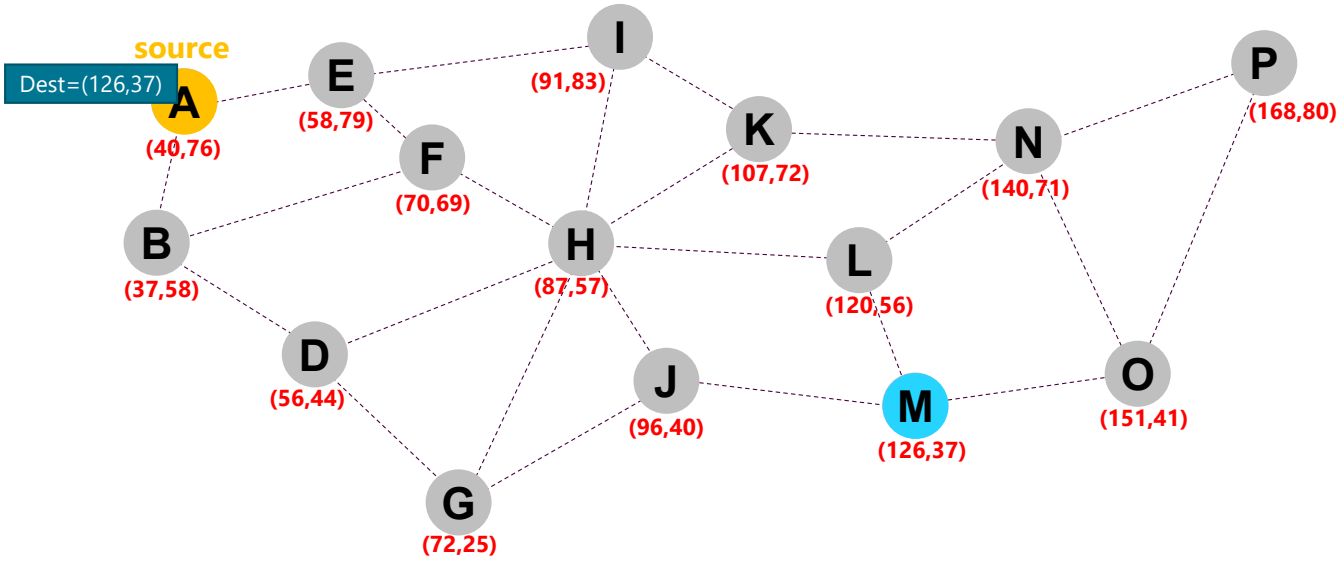
- Assumed network is not partitioned
  - Is it realistic?
- What if graph is not always connected?
  - **Occasionally-connected networks:** Sensors mounted on animals, floating in ocean; space satellites that "pass" occasionally
  - **Highly unreliable environments:** military networks suffering from jamming, acoustic links in air/water, free-space optical communications
  - **Low-power environments:** low-duty cycle sensors/actuators



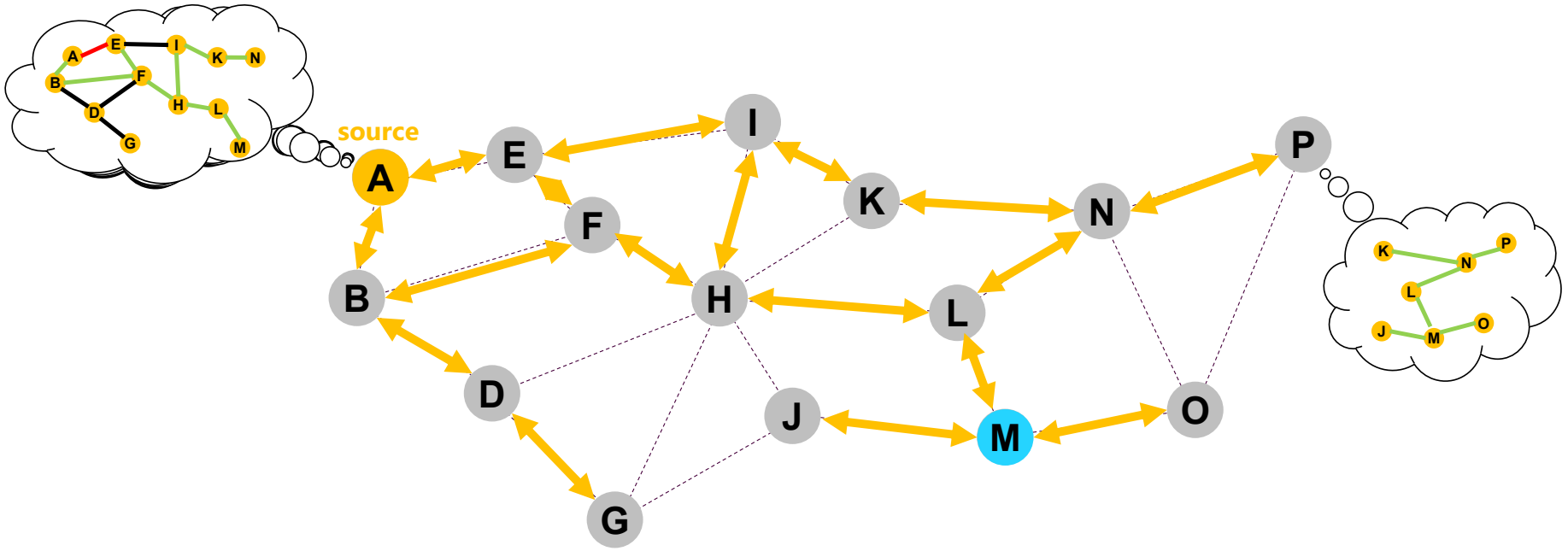
Delay-Tolerant Networking



Delay-Tolerant Networking: Geographic Routing

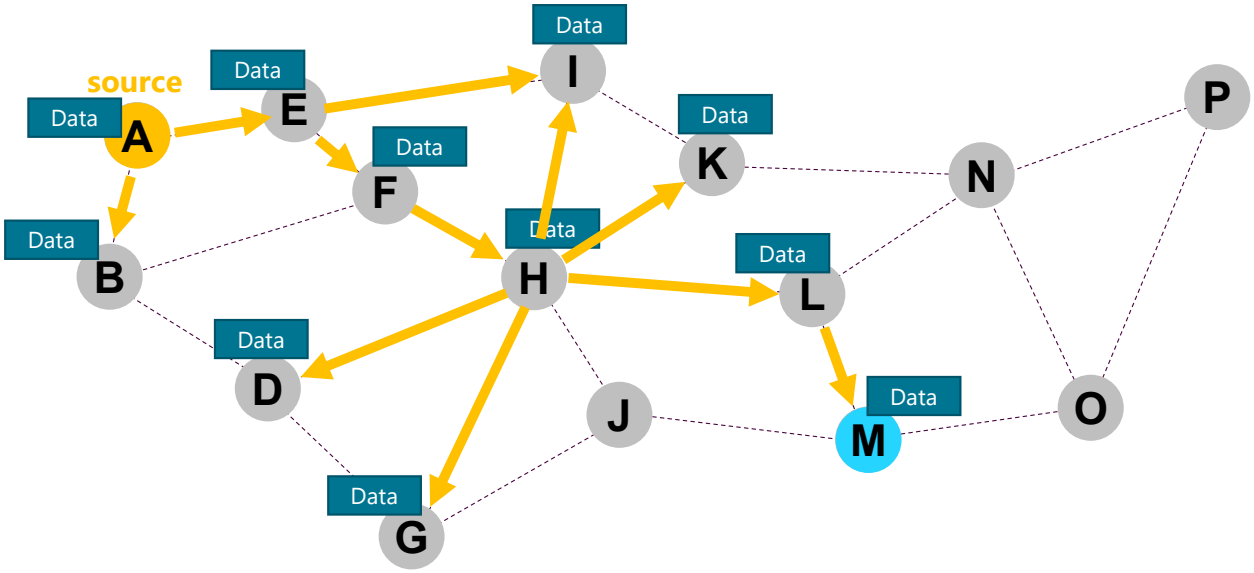


Delay-Tolerant Networking: Delay-Tolerant Link State Routing



- Run link state, but "age out" links when they go down
  - Data will flow along paths that will hopefully re-appear in the future

Delay-Tolerant Networking: Epidemic/Gossip Routing



- Every node waits random time, picks random target, and sends data

Delay-Tolerant Networking: Epidemic/Gossip Routing

- If all link failures are transient and reoccurring, message will eventually reach the destinations
  - Will reach all other nodes too – good for multicast/anycast/broadcast
- However, high control traffic overhead, slow propagation
- Improvement: **Rumor mongering**
  - When node gets new update it becomes a “hot rumor”
  - When it encounters many nodes with the update, it propagates it less frequently