# Lecture 11
# WiFi MAC

CS397/497 – Wireless Protocols for IoT

Branden Ghena – Spring 2024

Materials in collaboration with
Pat Pannuto (UCSD) and Brad Campbell (UVA)

Northwestern

# Today's Goals

- Introduce MAC layer concepts in 802.11
- Understand what exists, what is actually used, and why
- Explore two additional areas in 802.11
    - Microcontroller use of WiFi
    - Future of WiFi

# Outline

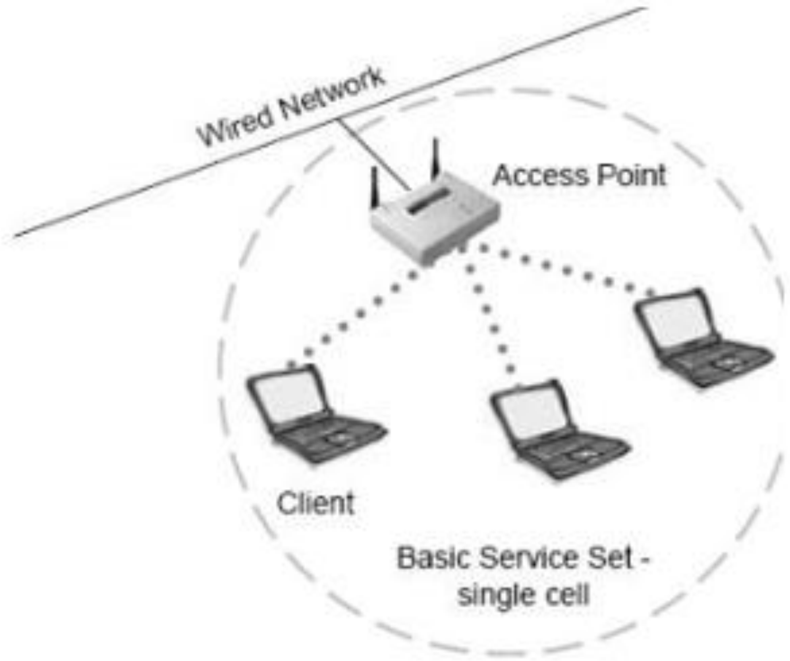
- **802.11 Access Control**
- 802.11 Frame format
- 802.11e Improvements to MAC
- Microcontrollers and WiFi
- MQTT

# Basic WiFi network

- Star topology network

- Basic Service Set (BSS)
  - Access point(s)
  - Multiple connected clients

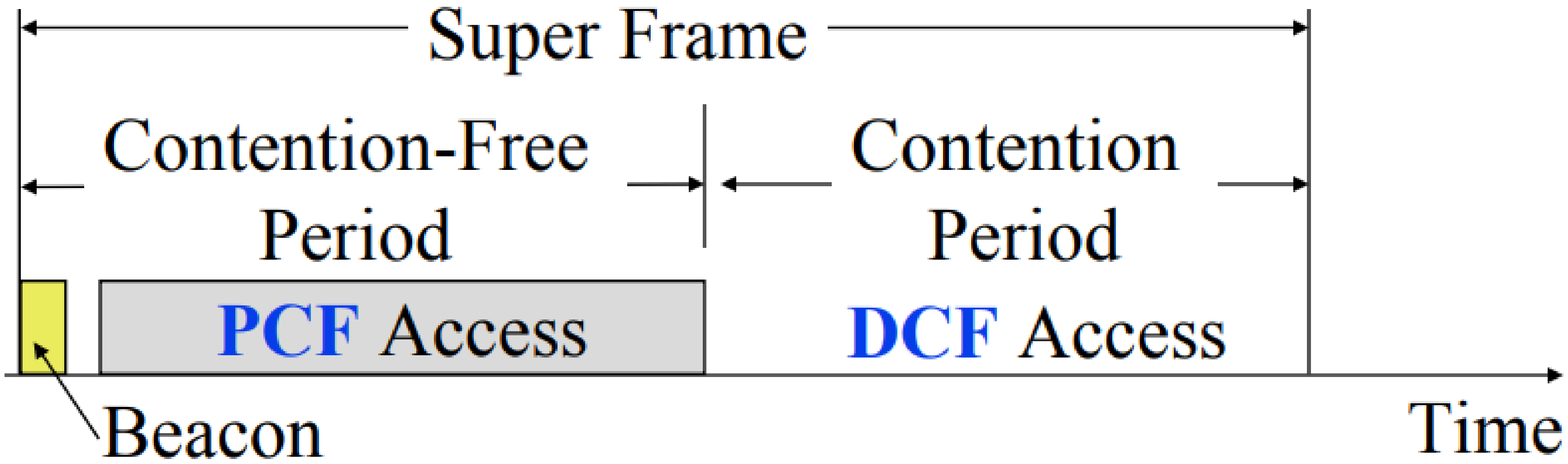
- Service Set ID (SSID)
  - Identifies network
  - Broadcast by access point in beacons



# WiFi superframe structure

- Beacon followed by contention-free period followed by contention
  - Repeats periodically (default ~100 ms)
  - 802.15.4 adopted a similar superframe

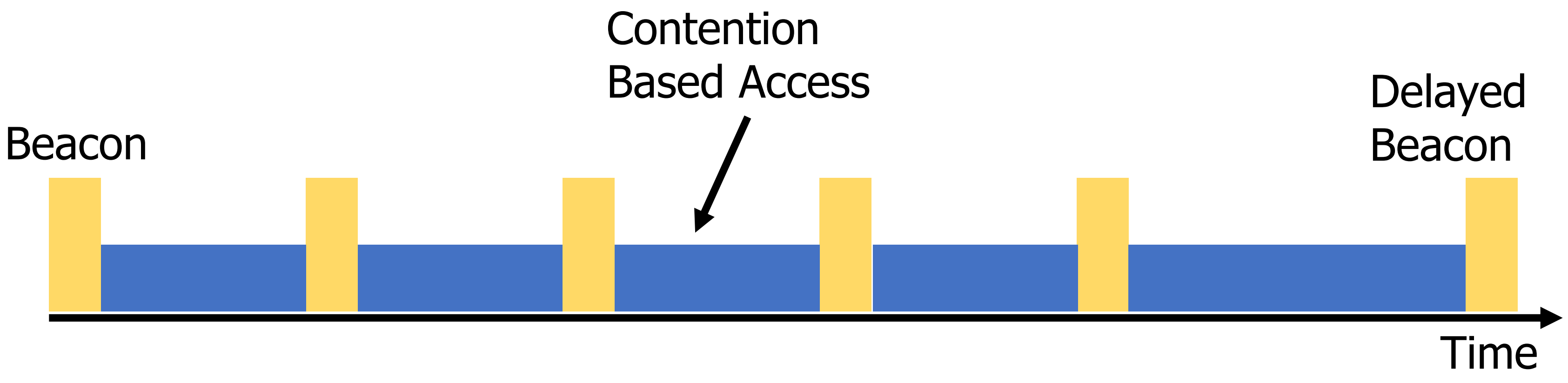
- This is more hypothetical than real



# WiFi superframe in practice

- Continuous contention access period
  - Any device may send at any time
  - PCF is unused in practice

- Periodic beacons
  - Which also use CSMA and therefore may be delayed



# 802.11 beacons

- Transmitted periodically (~100 ms by default)
  - Enable discovery of network
    - Contain capabilities and SSID for the network (802.11b/g/n/ac/ax…)
  - Assign contention-free slots if used
  - Notify devices of waiting packets
    - Traffic Indication Map (TIM) has a bitmap specifying which devices data is for
    - Enables devices to sleep, skipping a number of beacons
  - Handles broadcast/multicast messages
    - Every N beacons includes a notation of available broadcast messages
    - Messages are transmitted during next contention access period using normal CSMA
    - Defines maximum sleep period for devices (must listen to these beacons)

# Contention-free access

- Known as Point Coordination Function (PCF)
  - Allocates a contention-free period for specific devices
  - Access Point decides when to grant based on requests

- Drawbacks
  - Latency depends on beacon intervals
  - Mechanism for explicit Quality of Service is unclear

- PCF is not used in practice
  - Especially with the adoption of MU-MIMO and OFDMA techniques, it’s just not necessary

# Contention-based access

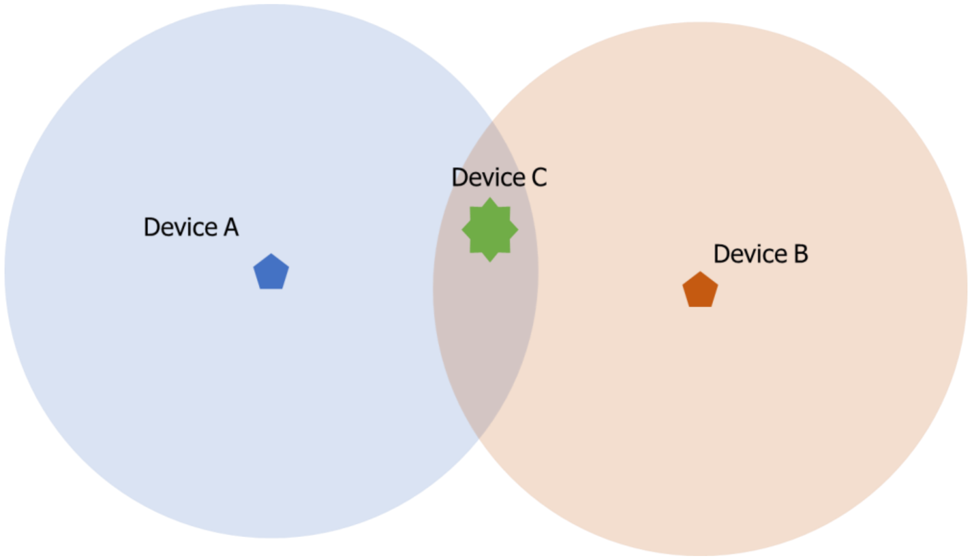
- Known as Distributed Coordination Function (DCF)
  - Base communication method for WiFi (essentially always)
  - All packets are immediately ACK’d by receiving device

  - Uses CSMA/CA to determine when it can send
    - With random backoff

  - Problem: packets can be very long (up to 20 milliseconds)
  - Solution: Network Allocation Vector (NAV)
    - Packets include a notation of their duration
    - Sensing the beginning of a packet allows backoff to skip the whole packet duration before continuing

# Reminder: hidden terminal problem

- Two devices communicating with Access Point may not be able to hear each other
  - CSMA fails and Access Point losses both messages




- A solution: RTS/CTS (Request/Clear To Send)

# Drawbacks of RTS/CTS

- Four packets per data (RTS, CTS, Data, Ack)
  - Could have just sent data instead of RTS
- Significant portion of traffic are application-layer Acks
  - Probably better to just have it fail and try again later
- RTS/CTS only used for very large packets in practice
  - *It’s mentioned still in 802.11n and 802.11ac, so not entirely unused

# Backoff in WiFi

- Listen for activity
  - If free
    - Wait for Inter Frame Spacing (IFS)
    - If still free, transmit

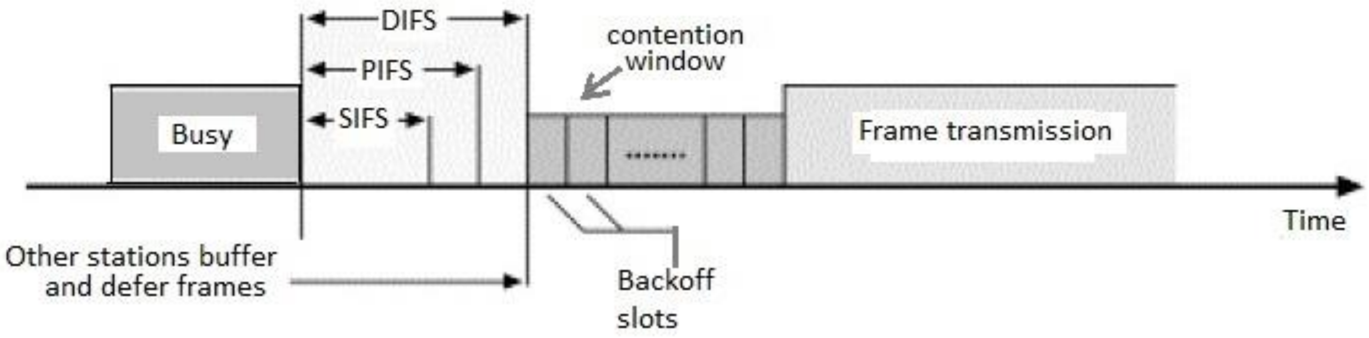
  - If busy
    - Randomly select a number of backoff **Slots**
    - Count down slots whenever medium is not busy
    - If busy when backoff completes:
      - Increase maximum backoff Slots
      - Repeat

- Slot time: basic time unit for protocol
  - Total time of: switch from Rx to Tx, plus processing time, plus propagation delay

# Prioritizing packets with varying IFS

- Tiered Contention Multiple Access (TCMA)
  - Idea: assign different inter-frame spacing based on traffic class
  - Inherently prioritizes communication

- Acknowledgements sent with Short IFS (SIFS)
  - Will always transmit before new data clears CSMA check
- New data sent with longer DCF IFS (DIFS)



# Putting backoff together

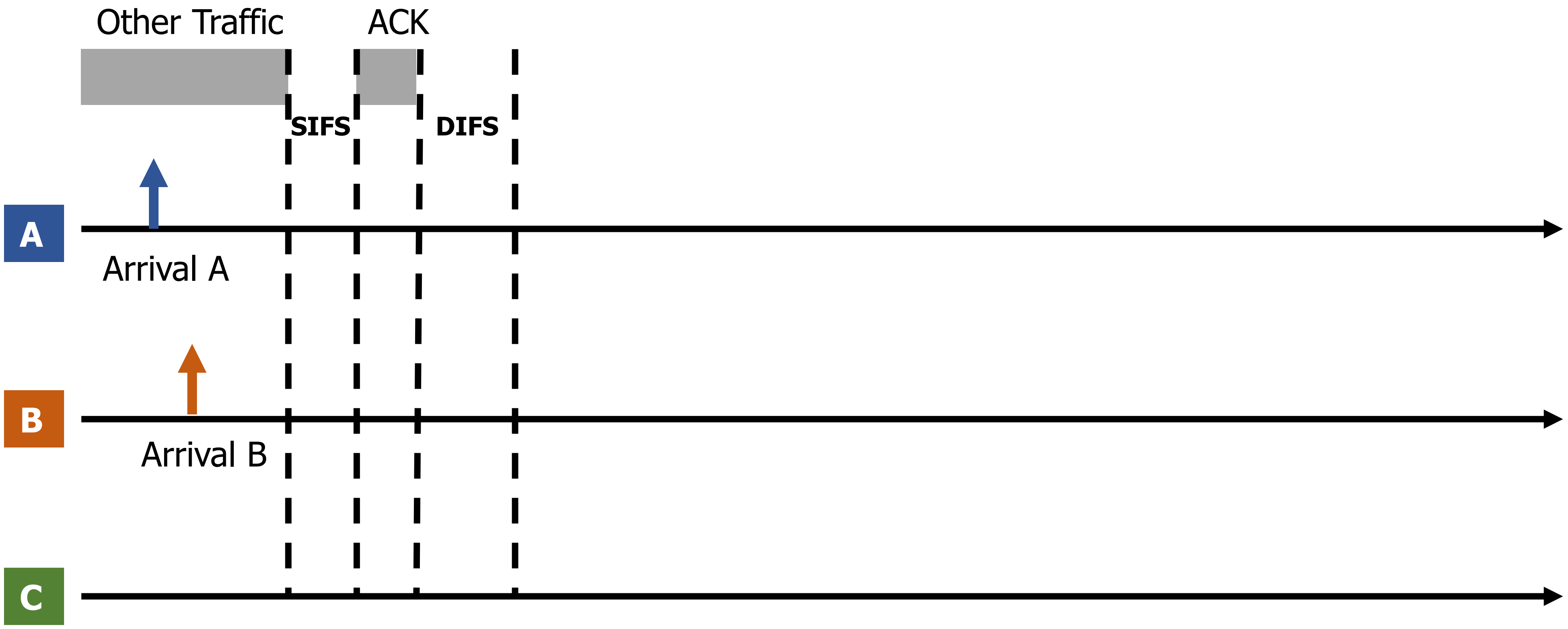
- Two variables
  - Contention Window (CW) – maximum backoff amount
  - Backoff Count (BO) – current remaining backoff

- When attempting to send, if busy Backoff selected in [0, CW]
  - Countdown Backoff slots whenever medium is not busy
  - At 0, attempt to transmit if not busy
  - If busy, double Window and select Backoff again

- 802.11g values:
  - Slot time= 20 us, CWmin= 15 slots (300 us), CWmax= 1023 slots (20 ms)
  - SIFS= 10 us, DIFS= 50 us

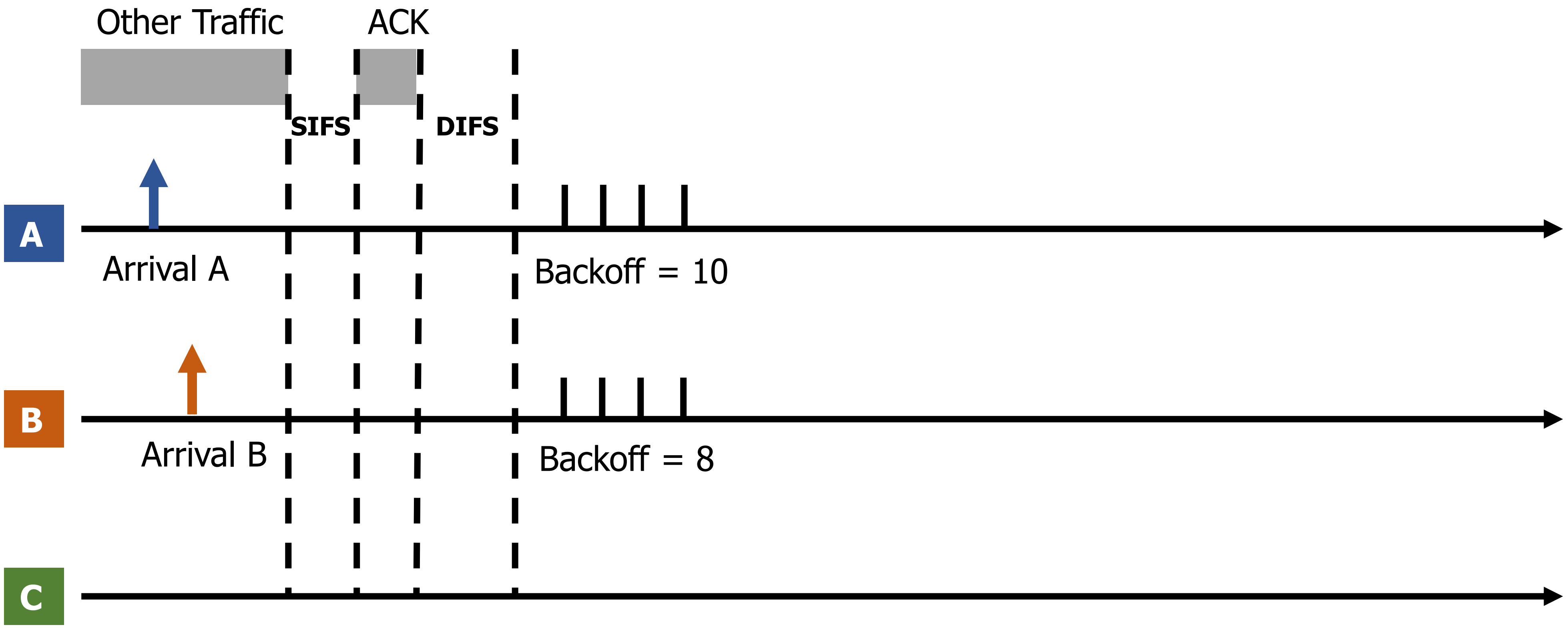
# 802.11 backoff example

- A and B want to send, but they see that the medium is busy
  - Followed by an Acknowledgement after SIFS



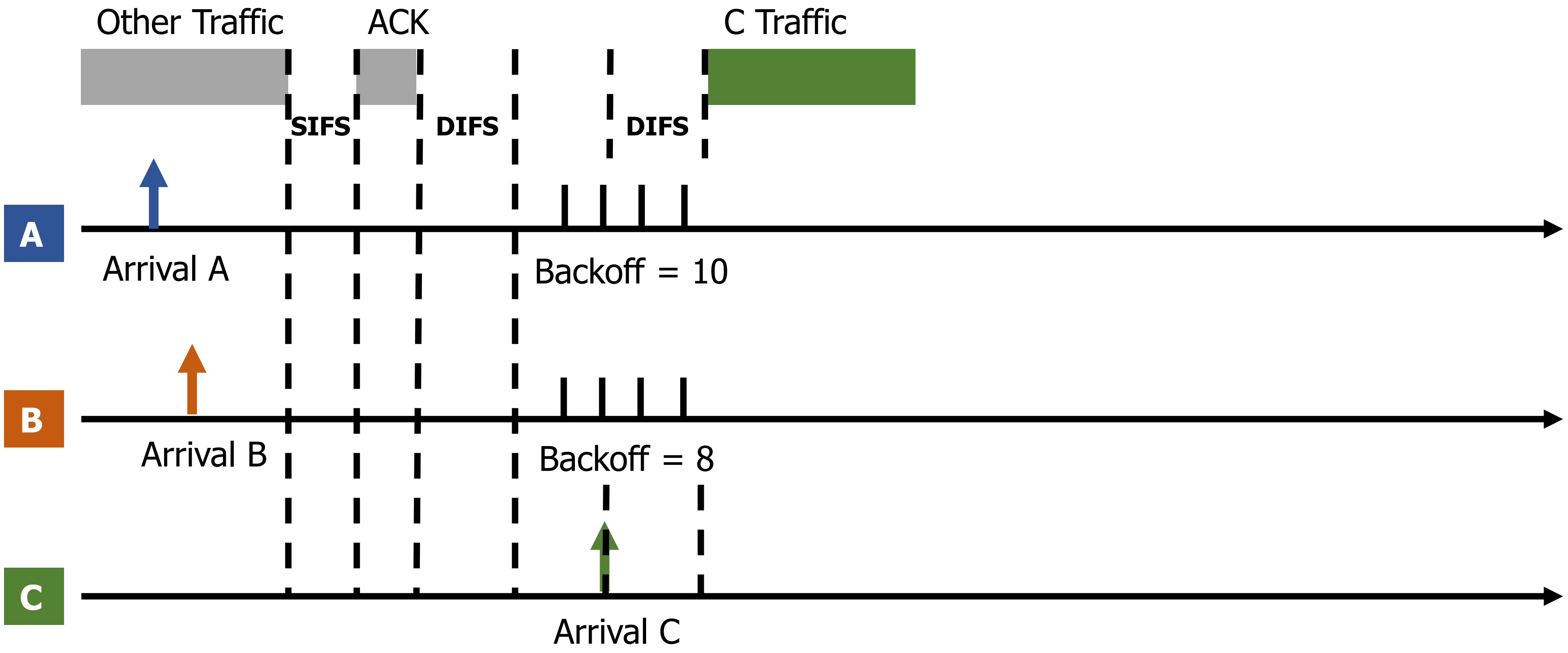
# 802.11 backoff example

- Each chooses a random backoff [0, CW] (we'll say CW is 32)
  - Start counting down backoff slots



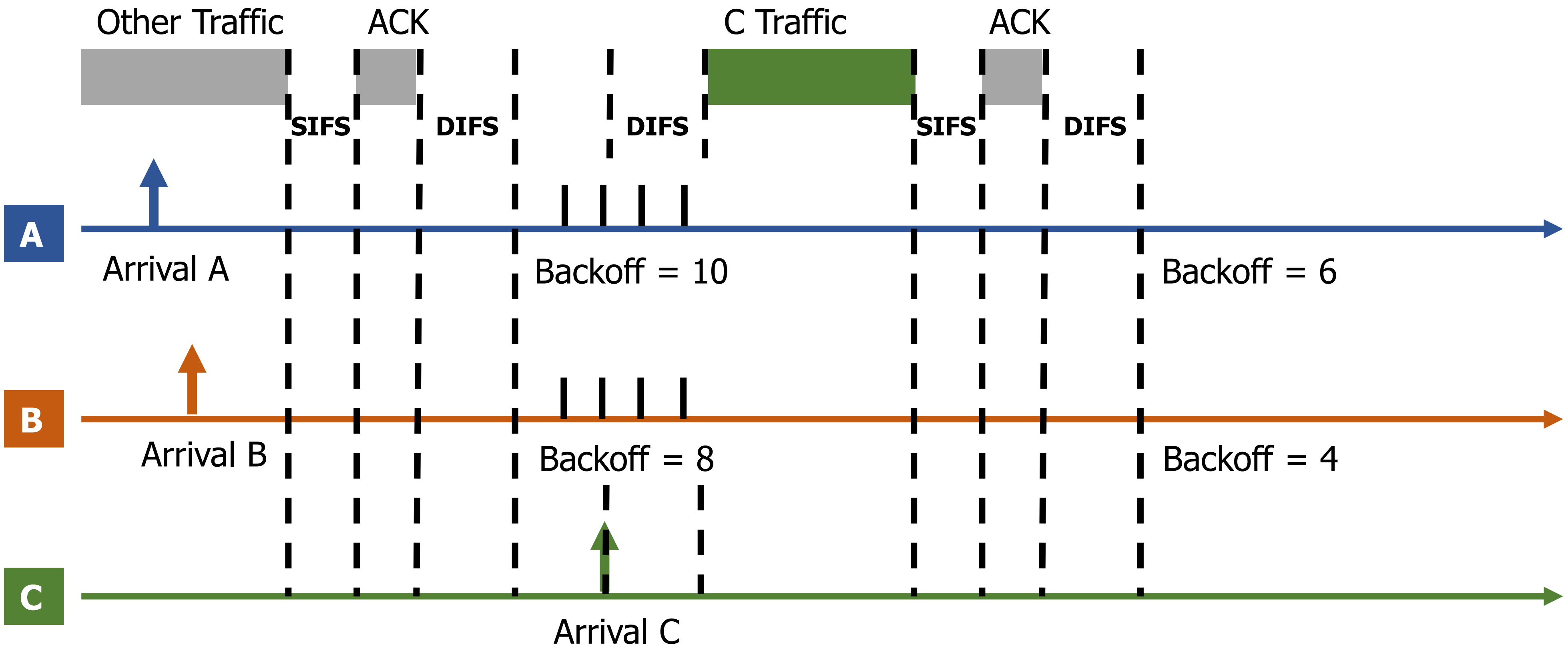
# 802.11 backoff example

- C wants to send, waits DIFS, and can send immediately
  - No other traffic is going on
  - A and B pause backoff for packet duration



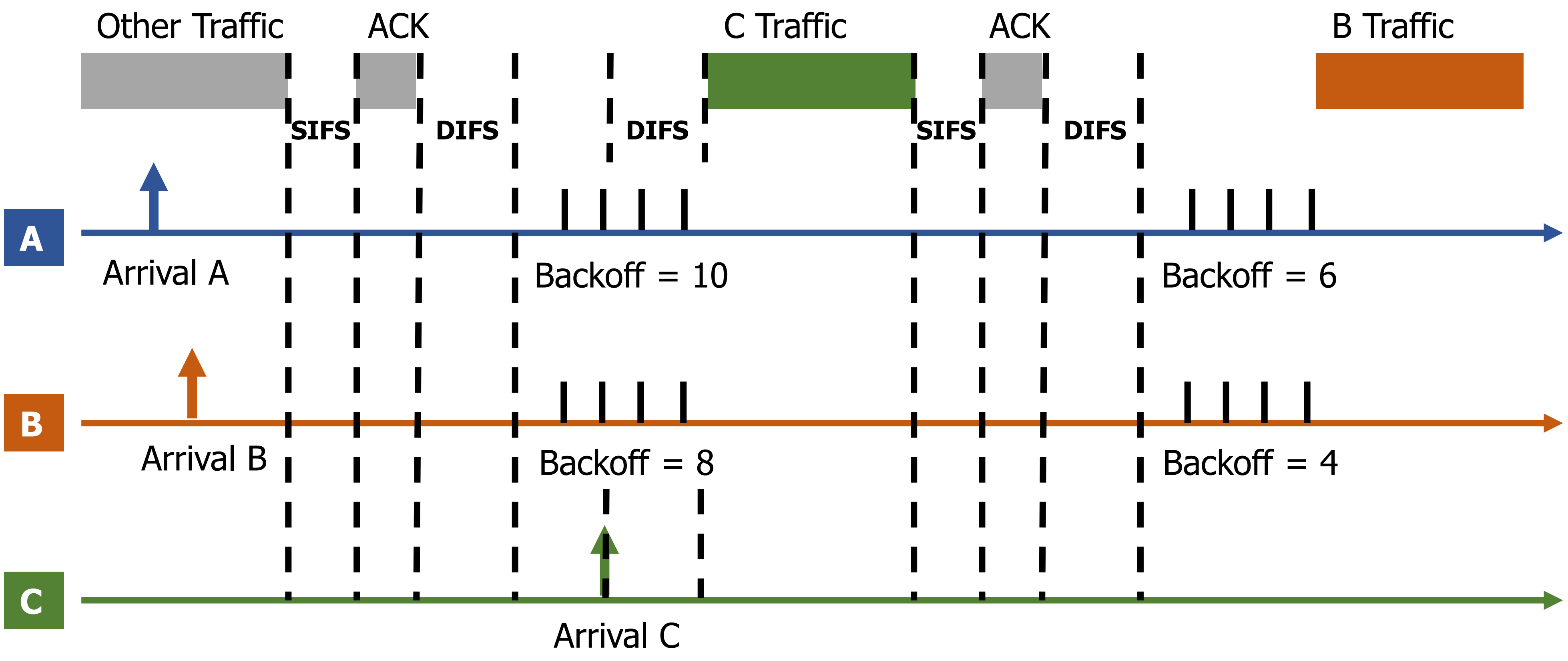
# 802.11 backoff example

- A and B used NAV to pause backoff for entire traffic plus ACK
  - After DIFS, resume backoff count from its previous value



# 802.11 backoff example

- B reaches zero backoff, finds channel empty, transmits
  - A pauses its backoff again for duration plus ACK



# Break + Hacking

- If you wanted maximum data throughput on a WiFi radio, and you were willing to be non-standards-compliant, what would you do?

# Break + Hacking

- If you wanted maximum data throughput on a WiFi radio, and you were willing to be non-standards-compliant, what would you do?

  - Never backoff at all. Just try during the next open period
    - Always be “device C” in our previous example

  - Use a shorter SIFS than other devices
    - If you start transmitting sooner, you get to keep transmitting!
    - Other devices will backoff on your transmission

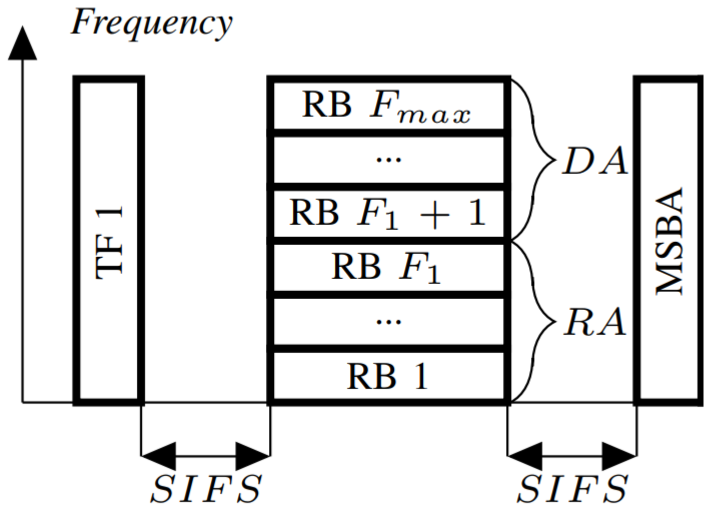
  - Tragedy of the Commons: this utterly fails if many radios follow it

# CSMA + OFDMA

- For WiFi 6 to be backwards compatible, the OFDMA stuff has to obey the normal CSMA/CA rules as well

- Downlink from AP
  - The AP wins contention and then transmits various data at various frequencies

- Uplink to AP
  - The AP wins contention,
    sends a “OFDMA uplink trigger frame”,
    then devices send their responses




https://ieeexplore.ieee.org/document/8422767

# Outline

- 802.11 Access Control
- **802.11 Frame format**
- 802.11e Improvements to MAC
- Microcontrollers and WiFi
- MQTT

# 802.11 frame

| Field | Frame control | Duration, id. | Address 1 | Address 2 | Address 3 | Sequence control | Address 4 | QoS control | HT control | Frame body | Frame check sequence |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Length (Bytes)** | 2 | 2 | 6 | 6 | 6 | 0, or 2 | 6 | 0, or 2 | 0, or 4 | *Variable* | 4 |

- Frame control (various bits)
  - Type of packet (Control, Management, Data)
  - Subtype (Association, RTS, CTS, Ack, etc.)
  - Indication of to/from "distribution system" (Internet rather than intranet)
- Duration
  - Specifies on-air time of full packet in microseconds
  - Note: no actual length field 😱

**Surprising, but smart!**

Recall MCS vary — but everyone needs to be able to parse header (for duration, for NAV)

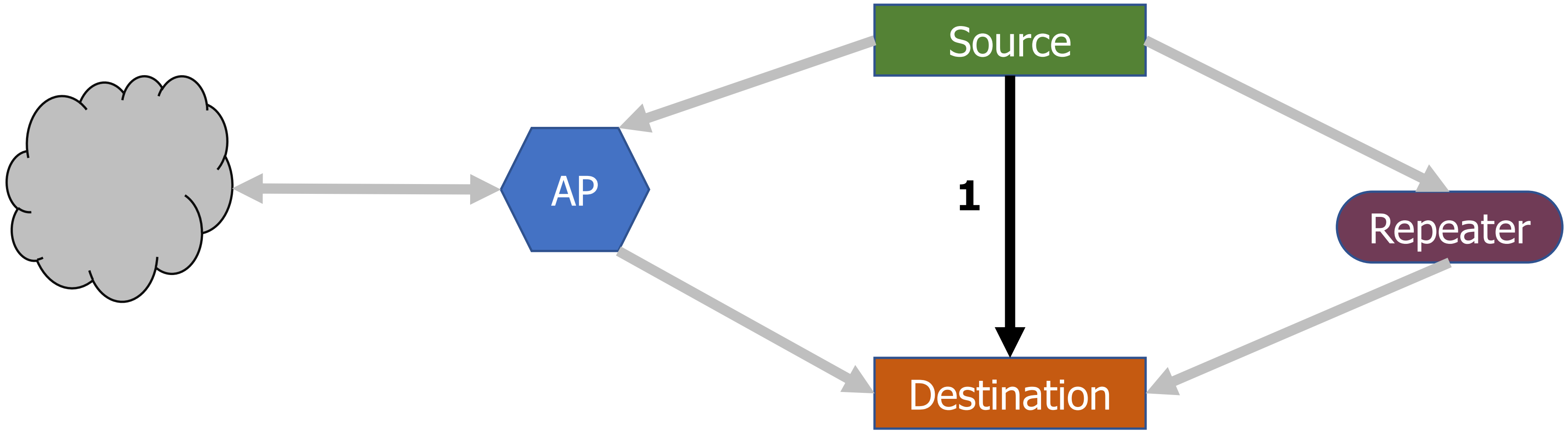
Length can be very large (e.g. in ac: 5.5 ms max duration is 4.5 MB length!); sent at full data rate

# 802.11 frame

| Field | Frame control | Duration, id. | Address 1 | Address 2 | Address 3 | Sequence control | Address 4 | QoS control | HT control | Frame body | Frame check sequence |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Length (Bytes)** | 2 | 2 | 6 | 6 | 6 | 0, or 2 | 6 | 0, or 2 | 0, or 4 | *Variable* | 4 |

- Sequence control
  - 4-bit fragment number
  - 12-bit sequence number
- Quality of Service control
  - Identifies traffic category
- High Throughput Control
  - Configurations for selecting best data rate
- Frame body
  - Max size depends on PHY
    - ~2000 for lower rates
    - ~8000 for 802.11n
    - ~11000 for 802.11ac
- Frame check sequence
  - 32-bit CRC

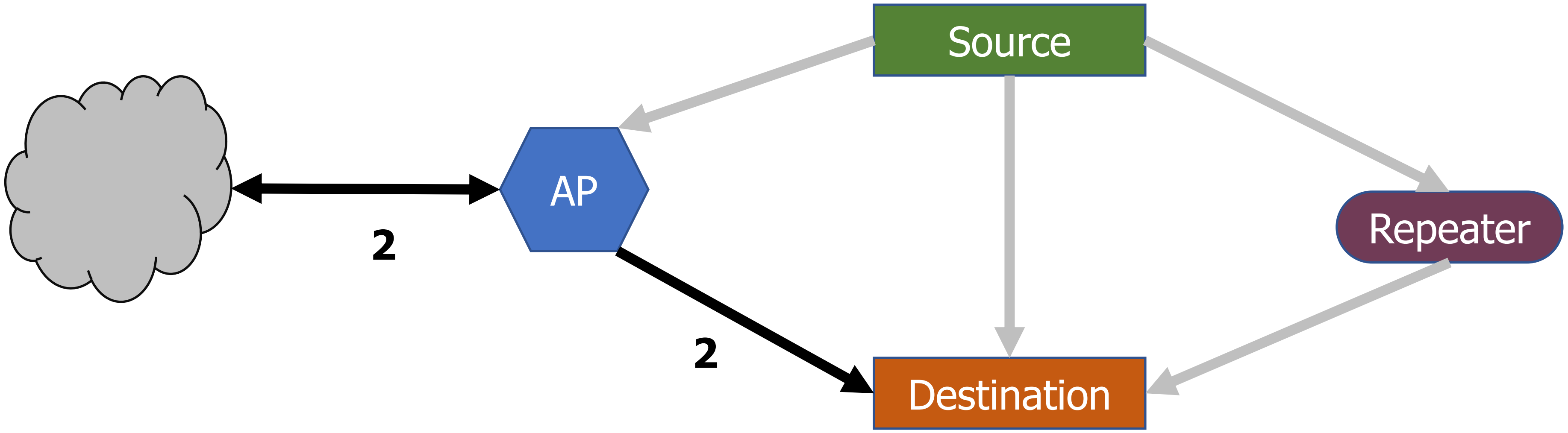
# Address field use cases




Devices filter on Address 1

| | To DS | From DS | Address 1 | Address 2 | Address3 | Address4 | Use Case |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | Destination Addr | Source Addr | BSS ID | - | Direct communication |
| 2 | 0 | 1 | Destination Addr | BSS ID | Source Addr | - | Traffic from Internet |
| 3 | 1 | 0 | BSS ID | Source Addr | Destination Addr | - | Traffic to Internet |
| 4 | 1 | 1 | Receiver Addr | Transmitter Addr | Destination Addr | Source Addr | Repeater |

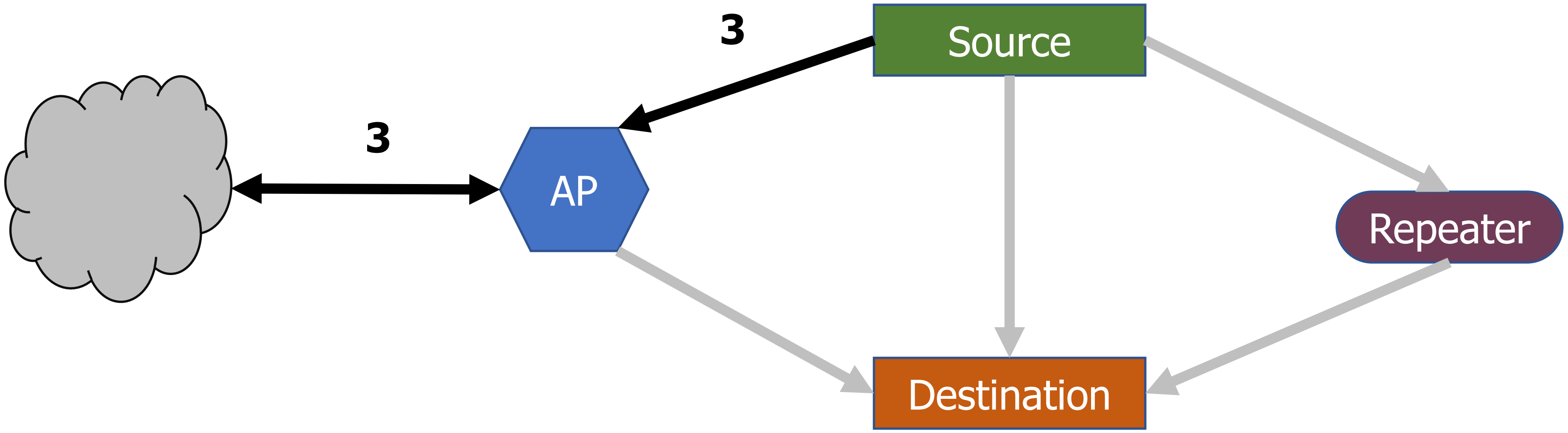
# Address field use cases




Devices filter on Address 1

| | To DS | From DS | Address 1 | Address 2 | Address3 | Address4 | Use Case |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | Destination Addr | Source Addr | BSS ID | - | Direct communication |
| 2 | 0 | 1 | Destination Addr | BSS ID | Source Addr | - | Traffic from Internet |
| 3 | 1 | 0 | BSS ID | Source Addr | Destination Addr | - | Traffic to Internet |
| 4 | 1 | 1 | Receiver Addr | Transmitter Addr | Destination Addr | Source Addr | Repeater |

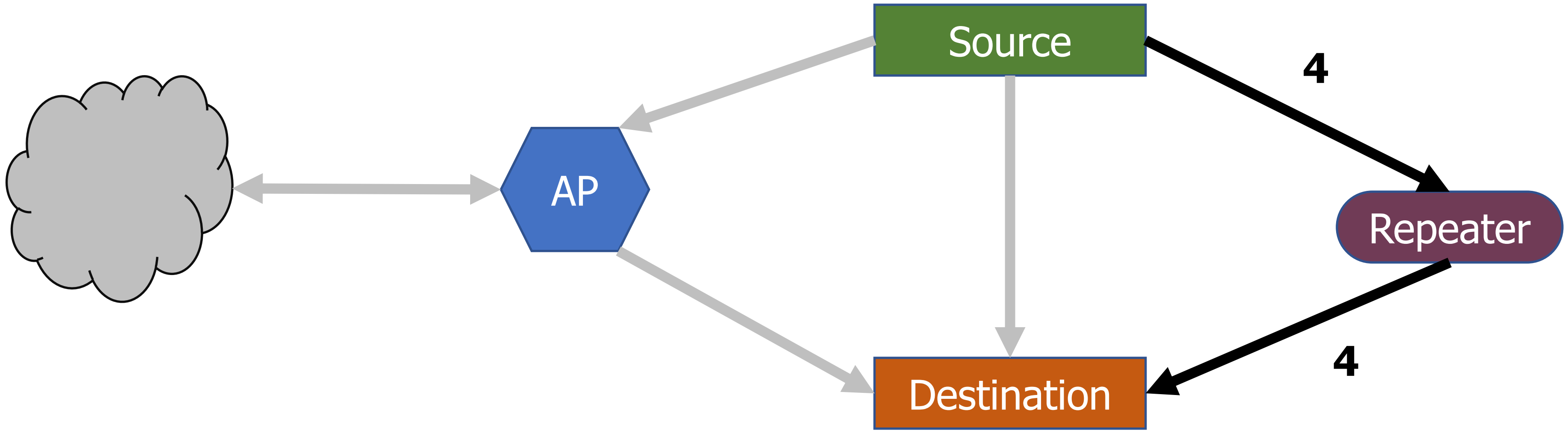
# Address field use cases




Devices filter on Address 1

| | To DS | From DS | Address 1 | Address 2 | Address3 | Address4 | Use Case |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | Destination Addr | Source Addr | BSS ID | - | Direct communication |
| 2 | 0 | 1 | Destination Addr | BSS ID | Source Addr | - | Traffic from Internet |
| 3 | 1 | 0 | BSS ID | Source Addr | Destination Addr | - | Traffic to Internet |
| 4 | 1 | 1 | Receiver Addr | Transmitter Addr | Destination Addr | Source Addr | Repeater |

# Address field use cases

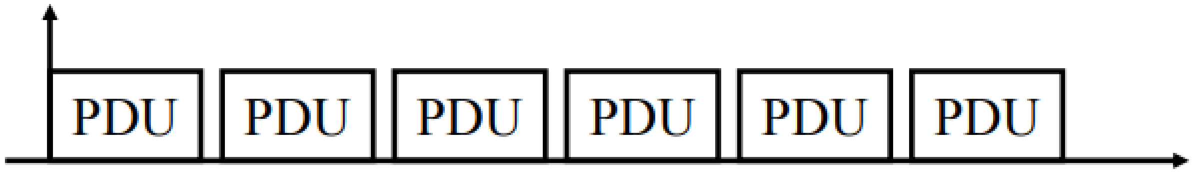



Devices filter on Address 1

| | To DS | From DS | Address 1 | Address 2 | Address3 | Address4 | Use Case |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | Destination Addr | Source Addr | BSS ID | - | Direct communication |
| 2 | 0 | 1 | Destination Addr | BSS ID | Source Addr | - | Traffic from Internet |
| 3 | 1 | 0 | BSS ID | Source Addr | Destination Addr | - | Traffic to Internet |
| 4 | 1 | 1 | Receiver Addr | Transmitter Addr | Destination Addr | Source Addr | Repeater |

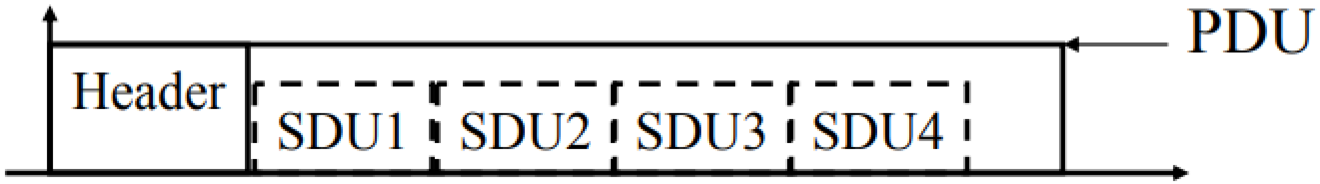
# Sending frames in WiFi

- Frame bursting (handles one really big packet)
  - Transmit multiple frames in a row




- Frame fragmentation
  - Split service data over multiple frames

- Frame aggregation (handles many very small packets)
  - Multiple service data in a single frame
  - Allows multiple packets to reach Access Point in a single transmission



# Calculating packet durations

- Example duration for a 1500 byte 802.11g packet
  - 6 Mbps for header
  - 24 Mbps for payload
  - 566 µs for total packet
    - Plus 10 µs for SIFS
    - Plus 34 µs for ACK

- https://sarwiki.informatik.hu-berlin.de/Packet_transmission_time_in_802.11

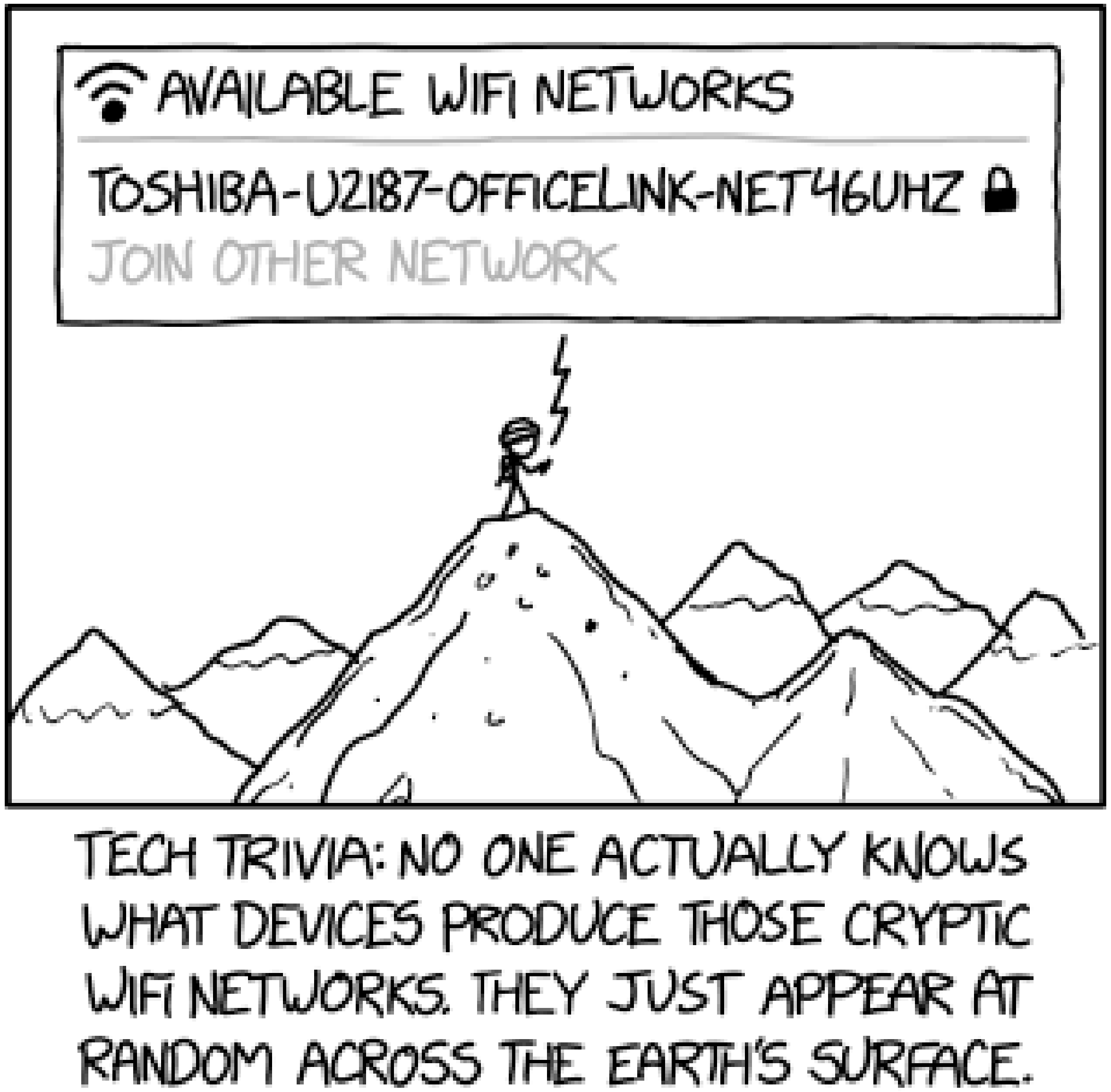
| | Data transmission bitrate (802.11g / a*): | | 24 Mbps | |
|---|---|---|---|---|
| | | Bitrate (Mbit/s) | Length (bits) | Time (µs) |
| | DIFS | | | 28 |
| D | PHY header: PLCP preamble | - | - | 16 |
| A | PHY header: PLCP header | 6 | 24 | 4 |
| T | MAC headers (28 bytes) + MAC body | 24 | 12246 | 512 |
| A | signal extension time | | | 6 |
| | **tx time data:** | | | **566** |
| | SIFS | | | 10 |
| A | PHY header: PLCP preamble | - | - | 16 |
| C | PHY header: PLCP header | 6 | 24 | 4 |
| K | MAC headers + PHY pad | 24 | 134 | 8 |
| | signal extension time | | | 6 |
| | **tx time ack:** | | | **44** |
| | **tx time data + ack:** | | | **610** |

# Implementation Drives Specification Sometimes

- SIFS nominally defined by processing time
  - Aside: Big challenge for SDRs
- Convolutional decoders need(ed) 16 µs to finish processing
  - For highest-rate MCS (ERP-OFDM)
  - SIFS is 10 µs, so extension needed
- Processing must finish before next packet starts
  - To be able to decode NAV in header

| | Data transmission bitrate (802.11g / a*): | | 24 Mbps | |
|---|---|---|---|---|
| | | Bitrate (Mbit/s) | Length (bits) | Time (µs) |
| | DIFS | | | 28 |
| D | PHY header: PLCP preamble | - | - | 16 |
| A | PHY header: PLCP header | 6 | 24 | 4 |
| T | MAC headers (28 bytes) + MAC body | 24 | 12246 | 512 |
| A | signal extension time | | | 6 |
| | **tx time data:** | | | **566** |
| | SIFS | | | 10 |
| A | PHY header: PLCP preamble | - | - | 16 |
| C | PHY header: PLCP header | 6 | 24 | 4 |
| K | MAC headers + PHY pad | 24 | 134 | 8 |
| | signal extension time | | | 6 |
| | **tx time ack:** | | | **44** |
| | **tx time data + ack:** | | | **610** |

# Break + xkcd




https://xkcd.com/2199/

# Outline

- 802.11 Access Control
- 802.11 Frame format
- **802.11e Improvements to MAC**
- Microcontrollers and WiFi
- MQTT

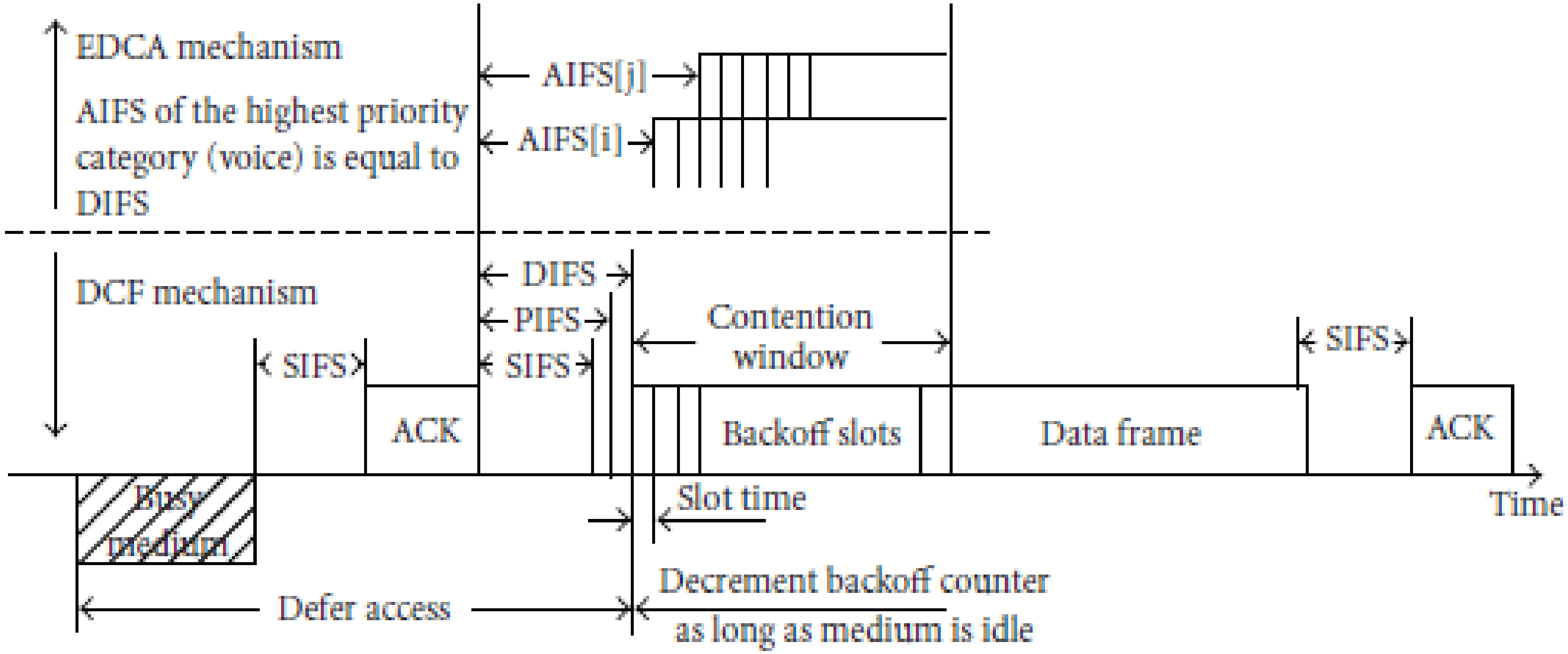
# 802.11e improves MAC layer

- Hybrid Coordination Function (HCF)
  - Modifies contention-free access (still no one uses it)
  - Modifies contention-based access: Enhanced Distributed Channel Access (EDCA)

- Adapts Quality of Service based on application
  - Example of breaking layering for an optimization
  - Categories (lowest to highest priority):
    - Background
    - Best Effort
    - Video
    - Voice

# Different priority for different application category

- Expand to more IFS lengths for different traffic categories
  - Smallest AIFS (equal to DIFS) goes to Voice, Largest to Background
  - Contention Window min and max also change for each category
    - Selects a *probability* that most important category goes first



# Multiple queues within a single device

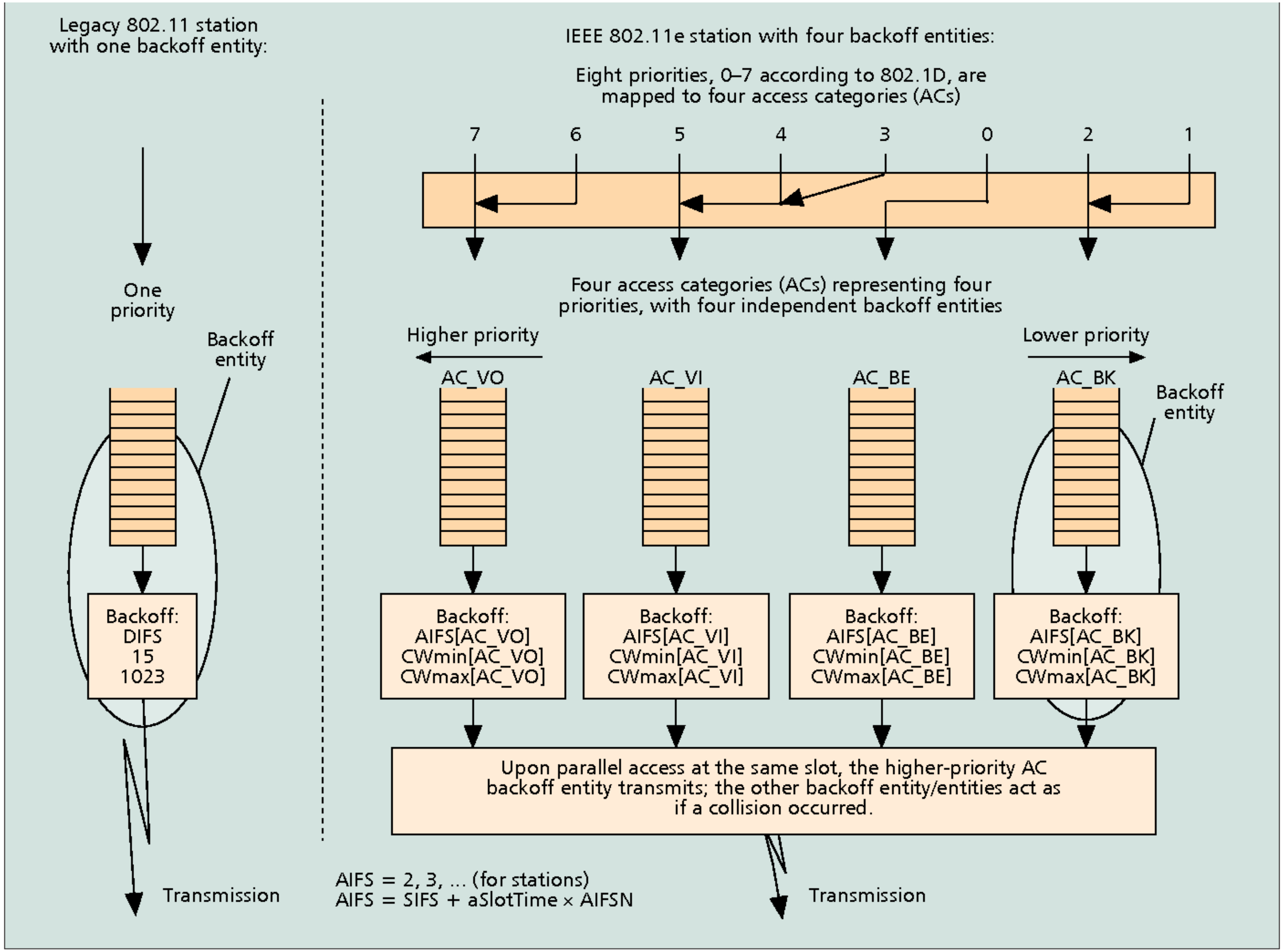



**Figure 4.** *[3] Legacy 802.11 station and 802.11e station with four ACs within one station.*

# 802.11e also adds maximum durations

- 802.11e also defines duration a device can transmit for
  - Based on PHY in use and Application category

  - Background/Best Effort: one frame per contention win

  - Example, up to 11 ms for Voice on 802.11ac
    - Could be one really big frame at a low data rate
    - Could be multiple frames in a row separated by SIFS

# Outline

- 802.11 Access Control
- 802.11 Frame format
- 802.11e Improvements to MAC
- **Microcontrollers and WiFi**
- MQTT

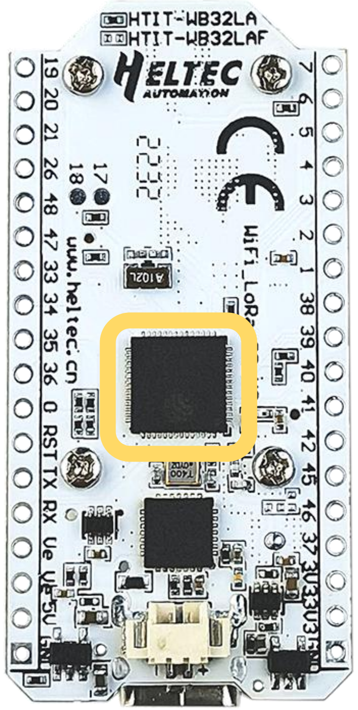
# Why, why not, talk WiFi in a wireless for IoT class

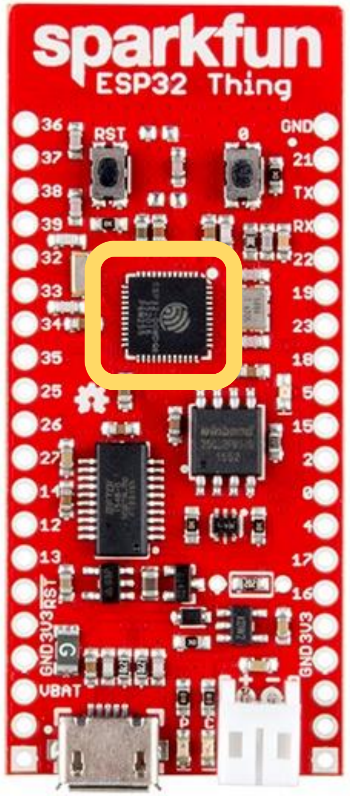
- Pros
  - Ubiquitous
  - High-performance
- Cons
  - Complex configuration
  - And security requirements
    - Device-Northwestern anyone?
  - Expensive in energy and money

# WiFi capability in microcontrollers

- ESP32
  - Microcontroller plus WiFi radio in single chip
  - (Same idea as nRF52840)

- Capabilities
  - 802.11b/g/n 2.4 GHz only
  - 20 MHz or 40 MHz channels
  - Single antenna only (no MIMO)
  - MCS0-7
    - 7 Mbps – 150 Mbps
  - Tx power up to 20.5 dBm



# Low power WiFi

- Question: should a microcontroller stay connected or reconnect?
  - Light sleep: stay connected always, only listening to beacons
  - Deep sleep: reconnect to network each time data is ready

- Answer for ESP32 depends on security and data interval
  - Resecuring during connection takes lots of energy
    - Crossover point is about 60 seconds

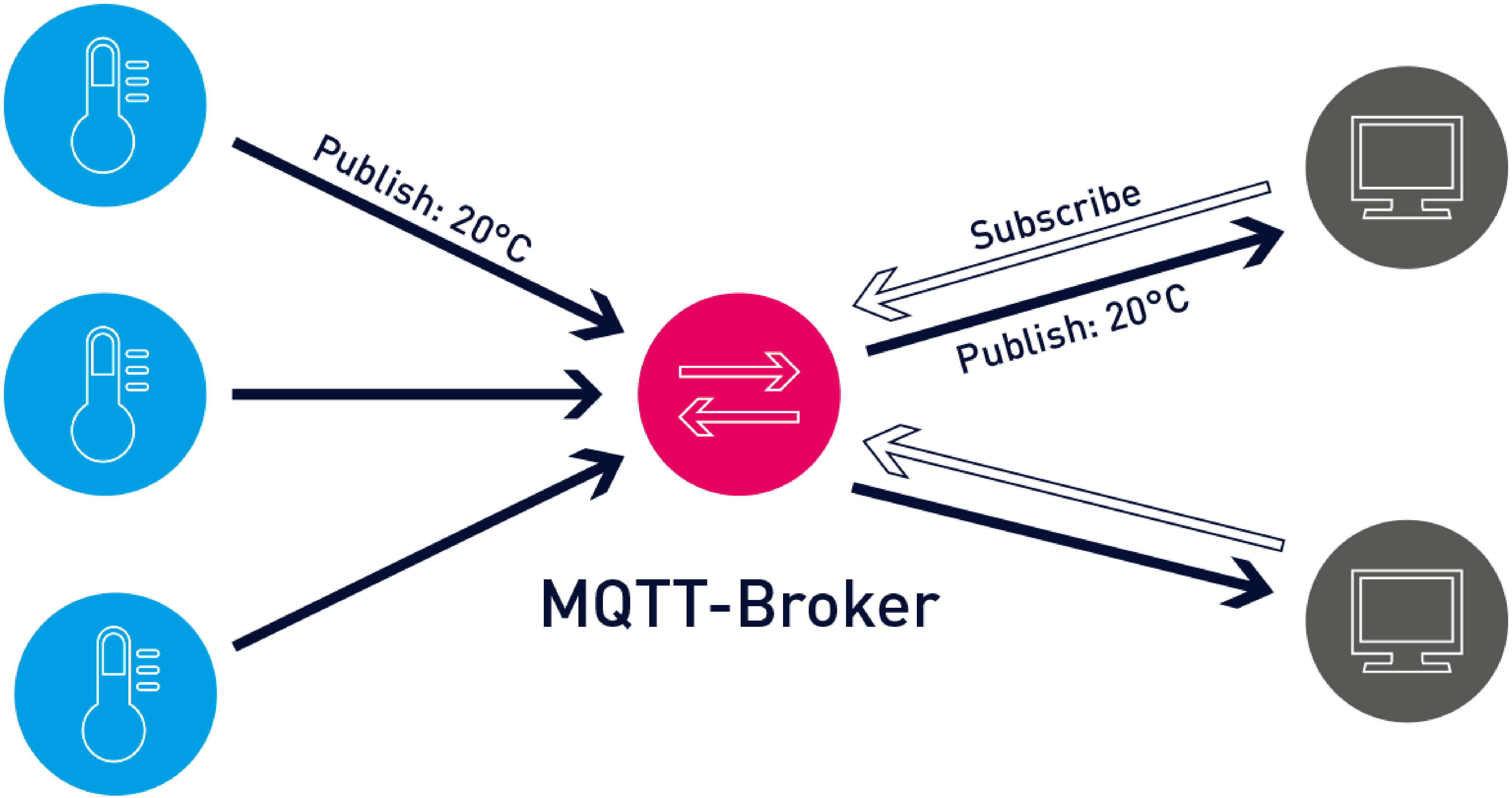
  - Insecure transmissions have a crossover of 5-15 seconds

https://blog.voneicken.com/2018/lp-wifi-esp-comparison/#conclusions

# Outline

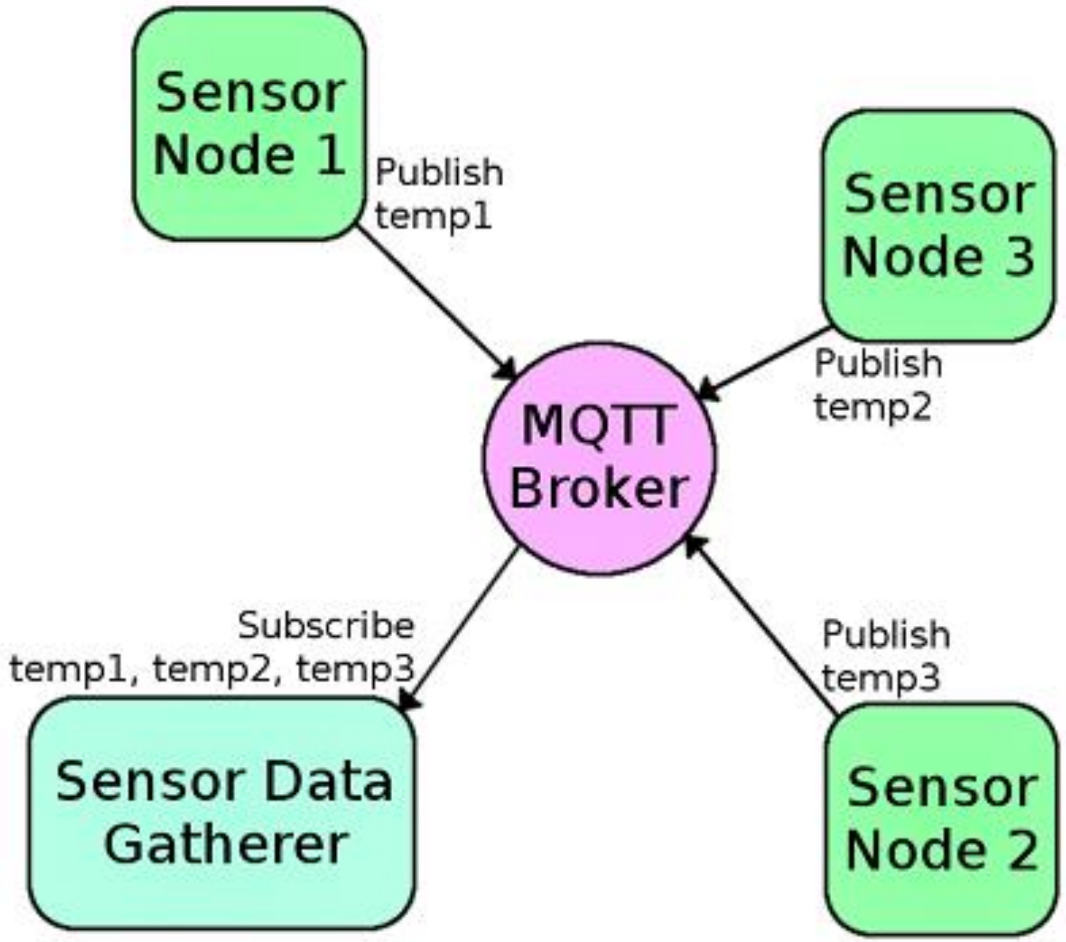
- 802.11 Access Control
- 802.11 Frame format
- 802.11e Improvements to MAC
- Microcontrollers and WiFi
- **MQTT**

# MQTT – an answer to the “where do you send data” question



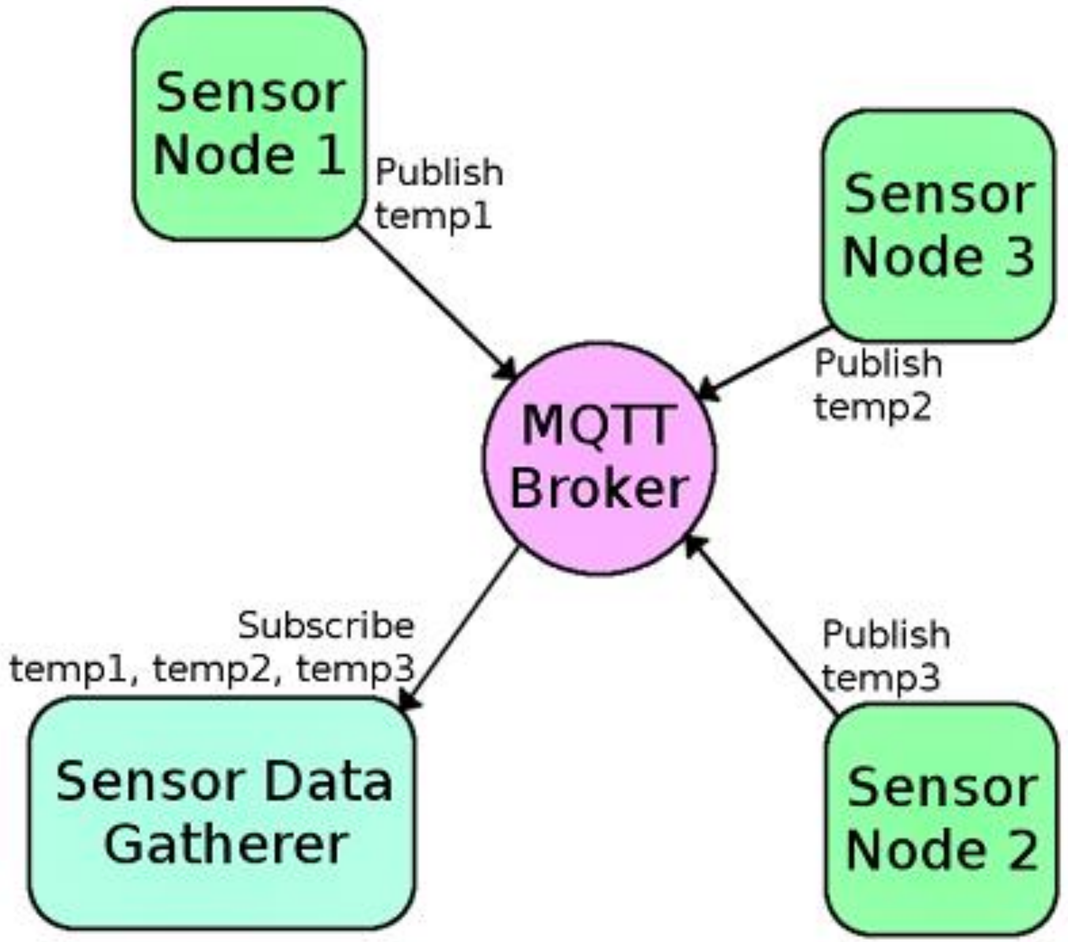
# MQTT Roles

- MQ Telemetry Transport
  - Built for IBM MQ products
- Broker
  - Server that distributes information
- Client
  - Any device connected to the Broker



# Pub/Sub Architecture

- Topic
  - String that names the data being sent
- Publish
  - Sends data with an associated Topic to Broker
- Subscribe
  - Inform Broker of which Topics you want data from
  - ALL subscribed Clients get copies of the data



# MQTT Access Control

- Nothing is required by default
  - Which is its own concern
- Can require a password to communicate with Broker
- Can make access control lists on a per-topic basis
  - Each user with a given password can only access certain topics

# Value of MQTT

- And that's pretty much everything there is to MQTT
  - MQTT is very simple, and that's a benefit
  - Packets are sent over TCP, so the reliability is already handled sufficiently
    - Variants can function over UDP instead

- How is data formatted?
  - However you want. It's just bytes attached to a Topic string
  - Could be an array of data bytes, JSON blob, image data, whatever
    - Up to MB of data in a single payload

  - Application-level probably knows how to decode

# Outline

- 802.11 Access Control
- 802.11 Frame format
- 802.11e Improvements to MAC
- Microcontrollers and WiFi
- MQTT