Lecture 4: Medium Access Control

CS 3035/GZ01: Networked Systems
Kyle Jamieson

Department of Computer Science
University College London
Structure of 3035/GZ01

1. Start with architecture
   – Protocols: How to structure communication
   – Layering: How to leverage modularity

2. Then move to the lower layers
   – Link technologies
   – Coping with errors

3. Study the “narrow waist” of IP
   – IP best-effort forwarding
   – IP addressing
   – Routing: Intra-, Inter-Domain (BGP)

4. Building on top of the narrow waist
   – TCP reliable transport, congestion control
   – Domain name system (DNS)
   – Security, web caching, CDNs
Review: The data link layer

• Enables exchange of atomic messages (*frames*) between end hosts

• Determine start and end of bits and frames (*framing*)
• Deliver information reliably
• Control errors (last lecture)

• Some link layers involve a shared medium
  – *e.g.*, Shared-wire Ethernet, satellite uplink, WiFi
  – **Today: Medium access control to share the medium**
Medium access: the problem

• Two questions:
  1. How should the shared medium be divided?
  2. Who gets to talk on a shared medium, and when?

• A medium access control (MAC) protocol specifies the above

• Three goals of a MAC protocol:
  1. Efficiency
     • High throughput (bits/second successfully received through the channel)
     • High utilization (throughput/raw channel rate)
  2. Fairness: all hosts with data to send should get a roughly equal share of the medium over time
  3. Latency: want to minimize the time a host waits before being granted permission to talk on the shared medium
Physical limitation: finite speed of light

From London to:

- WiFi AP: 3–30 m
- Mobile Base station: 300 m
- WiMAX Base station: 3 km
- Paris: 343 km
- San Francisco: 8,645 km
- Hawaii: 12,000 km
- Boston via Geosync Satellite: 73,000 km

Travel times:
- 3–30 m: 0.2 s
- 300 m: 58 ms
- 3 km: 43 ms
- 343 km: 1.7 ms
- 8,645 km: 10 μs
- 12,000 km: 1 μs
- 73,000 km: 10–100 ns
Similar MAC protocols, vastly different timescales

- WiFi: CSMA/CA
- Ethernet: CSMA/CD
- WiMAX Base station
- Packet radio: ALOHA
- Geosynchronous Satellite

Distances and Times:

- WiFi: 3–30 m, 73,000 km
- WiMAX: 3 km
- Ethernet: 10–100 ns
- Packet radio: 10 μs
- Geosynchronous Satellite: 0.2 s
Today

1. Channel partitioning:
   – Time division multiple access (TDMA)
   – Frequency division multiple access (FDMA)
   – Code division multiple access (CDMA)

2. Random access protocol: ALOHA
   – Unslotted ALOHA
   – Slotted ALOHA

3. Random access protocol: the Ethernet
TDMA: Time Division Multiple Access

- Access to channel in “rounds”
- Each station gets fixed length \textit{slot} (packet time) in each round (\textit{unused slots go idle})
- \textbf{Example:} six stations, only 1, 3, and 4 have data to send
FDMA: Frequency Division Multiple Access

- Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band (unused frequency bands are wasted)
- **Example:** six stations, only 1, 3, and 4 have data to send
CDMA: Code Division Multiple Access

• CDMA is a medium access protocol used in wireless networks

• All users transmit over the same frequencies, and at the same time

• Another example of the use of codes
  – But for sharing, instead of error control

• Codes allow multiple users to coexist and transmit simultaneously with no interference, in theory
  – In practice: performs well (many cellular mobile telephone networks use CDMA)
CDMA: User codes

• Each user $n$ has her own binary “chip” code $c^n$ of length $M$
  – Represent binary data and binary code using $\{1, -1\}$
  – Encoding at user $i$:
    • For each data bit $d_k$, form $M$ repetitions of $d_k$, then multiply the result, bit-by-bit, with $c^n$

• e.g. (user 1, $M = 8$):

User 1 data:
\[
d_0 = -1, \quad d_1 = 1
\]

User 1 code:
\[
c^1 = \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}
\]

Channel output for user 1:
\[
\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 \end{bmatrix}
\]
CDMA: Code rate

- Chip rate = raw channel rate \( B \)
- Data rate = \( B/M \) (\( M \) times slower than raw channel rate)
- Overall code rate \( R = 1/M \)

- \( e.g. \) (user 1, \( M = 8 \)):
Decoding a single CDMA transmission

- Receiver hears $Z_{i,m}$: $m^{th}$ chip for $i^{th}$ data bit

- To decode data bit $i$, receiver multiplies what it hears with user 1’s code, chip by chip:
  $$d_i = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} c_m^1$$

**Example:**

<table>
<thead>
<tr>
<th>Received data $Z_{i,m}$:</th>
<th>User 1 code:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>-1 -1 -1 1 -1 -1 -1</td>
<td>-1 -1 -1 -1 -1 -1 -1</td>
</tr>
</tbody>
</table>

$$d_i = \frac{1}{7} (1 + 1 + 1 + 1 + 1 - 1 - 1) = 1$$

$d_0 = 1$

$d_i = -1$
Sharing the medium with CDMA

• Let’s assume we have a way of:
  – Synchronizing all users’ data bits in time
  – Synchronizing all users’ chips in time

• In many shared mediums (*e.g.* wireless, but not optical), principle of linearity applies:
  – The channel *sums* the transmitted signals together, as shown:

\[
Z_{i,m}^1 = d_i^1 \cdot c_m^1
\]

\[
Z_{i,m}^2 = d_i^2 \cdot c_m^2
\]

Result: Wireless interference
CDMA: Separating users’ transmissions

- As before, receiver selects user 1’s code $c^1$, and multiplies the received signal $Z_{i,m}$ by $c^1$, chip by chip:

$$d_i = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} c^1_m$$

- User 2’s interference is completely cancelled
CDMA: How to choose codes

• Two users: user 1 (code $c^1$), user 2 (code $c^2$)

• When the receiver “tuned” to data bit 1 from user 1:

$$d_1^1 = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} c_m^1$$

$$= \frac{1}{M} \sum_{m=1}^{M} \left( -c_m^1 + c_m^2 \right) c_m^1$$

$$= \frac{1}{M} \sum_{m=1}^{M} \left( -c_m^1 c_m^1 + c_m^2 c_m^1 \right)$$
CDMA: How to choose codes

Orthogonality condition:
\[ \sum_{m=1}^{M} c_m^1 c_m^2 = 0 \]

\[
d_1^1 = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} c_m^1 \\
= \frac{1}{M} \sum_{m=1}^{M} \left( -c_m^1 + c_m^2 \right) c_m^1 \\
= \frac{1}{M} \sum_{m=1}^{M} \left( -c_m^1 c_m^1 + c_m^2 c_m^1 \right)
\]

User 1:
- Data bits: \[ d_1 = -1 \]
- Code: 1111 1

User 2:
- Data bits: \[ d_2 = 1 \]
- Code: 1111 11
Example of CDMA codes: Walsh codes

- $n^{th}$ user’s length-$N$ **Walsh code** is the $n^{th}$ row of an $N \times N$ Hadamard matrix $H(N)$

$$H(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H(2^k) = \begin{bmatrix} H(2^{k-1}) & H(2^{k-1}) \\ H(2^{k-1}) & -H(2^{k-1}) \end{bmatrix}$$

- Code rate: $1/N$
- Supports up to $N$ concurrent users

$$H(4) = \begin{bmatrix} H(2) & H(2) \\ H(2) & -H(2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$
Do Walsh codes have the orthogonality property?

\[ H(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H(2^k) = \begin{bmatrix} H(2^{k-1}) & H(2^{k-1}) \\ H(2^{k-1}) & -H(2^{k-1}) \end{bmatrix} \]

• Let’s look at the two rows of \( H(2) \)... yes!

• Assuming rows of \( H(2^{k-1}) \) is orthogonal, let’s imagine a walk across two rows of \( H(2^k) \), multiplying them term-by-term
  
  I. We pick two rows, both in upper or both in lower half \( \rightarrow \) Product of the two is \( 0 + 0 \)

  II. We pick one row in upper half, one in lower half, but same respective rows of \( H(2^{k-1}) \) \( \rightarrow \) Product of the two is \( 2^{k-1} - 2^{k-1} \)

  III. We pick one row in upper half, one in lower half, but different respective rows of \( H(2^{k-1}) \) \( \rightarrow \) Product of the two is \( 0 + 0 \)
Today

1. Channel partitioning:
   - Time division multiple access (TDMA)
   - Frequency division multiple access (FDMA)
   - Code division multiple access (CDMA)

2. Random access protocol: ALOHA
   - Unslotted ALOHA
   - Slotted ALOHA

3. Random access protocol: the Ethernet
Random access MAC protocols

- When a station has a frame to send:
  - Transmit at full channel data rate $B$
  - No *a priori* coordination among nodes

- Two or more frames overlapping in time: **collision**
  - Both frames are lost, resulting in diminished throughput

- A random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions
ALOHAnet: Context

- Norm Abramson, 1970 at the University of Hawaii
  - Seven campuses on four islands
  - Want to keep campus terminals in contact with mainframe
  - Telephone costs high, so build a packet radio network
Unslotted ALOHA

• **Simplest possible medium access control:** no control at all, anyone can just transmit a packet without delay

• Let’s assume that the probability a packet begins in any time interval of length $\Delta t$ is $\lambda \times \Delta t$
  – $N$ senders in total, sending frames of time duration 1
  – This is called a **Poisson process with rate $\lambda$**
    • $\lambda$ **is the aggregate rate** from all $N$ senders
    • Individual rate $\lambda/N$ for each sender
Unslotted ALOHA: Performance

• Suppose some node $i$ is transmitting; let’s focus on $i$’s frame

I. If others send between $t_0 - 1$ and $t_0$, their frames will overlap with the start of $i$’s frame → collision

II. If others send between $t_0$ and $t_0 + 1$, their frames will overlap with end of $i$’s frame → collision

III. Otherwise, no collision, and node $i$’s frame is delivered

• Therefore, there is a “vulnerable period” of length 2 around $i$’s frame
Unslotted ALOHA: Performance

- What’s the chance no one else sends in the vulnerable period of length 2?
  \[ \Pr(\text{no send from one node in 2}) = 1 - \frac{2 \lambda}{N} \]
  \[ \Pr(\text{no send at all in 2}) = \left(1 - \frac{2 \lambda}{N}\right)^{N-1} \]
  \[ \lim_{N \to \infty} \left(1 - \frac{2 \lambda}{N}\right)^{N-1} \to e^{-2\lambda} \]
• Recall from our definition of the Poisson process: $\lambda$ is the aggregate rate from all senders
  • So, utilization $= \lambda \times \Pr(\text{no other transmission in 2})$
    $= \lambda e^{-2\lambda}$
Today

1. Channel partitioning:
   – Time division multiple access (TDMA)
   – Frequency division multiple access (FDMA)
   – Code division multiple access (CDMA)

2. Random access protocol: ALOHA
   – Unslotted ALOHA
   – Slotted ALOHA

3. Random access protocol: the Ethernet
Slotted ALOHA

- **Divide time into slots** of duration 1, **synchronize** so that nodes transmit **only** in a slot
  - Each of $N$ nodes transmits with probability $p$ in each slot
  - So **aggregate transmission rate** $\lambda = N \times p$

- As before, if there is **exactly one** transmission in a slot, can receive; if **two or more** in a slot, no one can receive (**collision**)
Suppose $N$ nodes, each transmit with probability $p$ in each slot. What is the utilization as a function of aggregate rate $\lambda = N \times p$?

- $\Pr[\text{A node is successful in a slot}] = p(1-p)^{N-1}$
- $\Pr[\text{Success in a slot}] = Np(1-p)^{N-1}$

$$\Pr(\text{success}) = \lambda \left(1 - \frac{\lambda}{N}\right)^{N-1}$$

$$\lim_{N \to \infty} \lambda \left(1 - \frac{\lambda}{N}\right)^{N-1} = \lambda e^{-\lambda}$$

$$\lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = e$$

$1/e \approx 37\%$
ALOHA throughput: slotted versus unslotted

Just by forcing nodes to transmit on slot boundaries, we double peak medium utilization!
Today

1. Tour of medium access protocols (TDMA, FDMA, CDMA)

2. Random access protocol: ALOHA
   - Unslotted ALOHA
   - Slotted ALOHA

3. Random access protocol: the Ethernet
How did the Ethernet get built?

- Bob Metcalfe, PhD student at Harvard in early 1970s
  - Working on protocols for the ARPAnet
  - Intern at Xerox Palo Alto Research Center (PARC), 1973
  - Needed a way to network the \( \approx 100 \) Alto workstations in-building
  - Adapt ALOHA packet radio

- Metcalfe later founds 3Com, acquired by HP in April ’10 for USD $2.7 bn
The Ethernet: Physical design

• Coaxial cable, propagation delay $\tau$
  – Propagation speed: $\frac{3}{5} \times$ speed of light

• Experimental Ethernet
  – Data rate: $B = 3$ Mbits/s
  – Maximum length: 1000 m

$$\tau = \frac{10^3 \text{ m}}{\frac{3}{5}(3 \times 10^8 \text{ m/s})} \approx 5 \mu s$$

Propagation delay: $\tau$
Building the link: Framing bits

• **Goal:** Move bits from one place to another
  – Sender and receiver have independent clocks
  – No separate “clock signal” sent on the Ethernet

• **Problem:** Agree on clock tick period

```
| Sender clock | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Receiver clock | "1" | "0" | "0" | "0" | "1" | "?" | "?" | "?" | "?" |
```

• **Problem:** Agree on clock tick alignment (*phase*)

```
| Sender clock | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Receiver clock | "?" | "?" | "?" | "?" | "?" | "?" | "?" | "?" | "?" |
```
How to encode bits?

- Simple binary encoding is called **Nonreturn to Zero** (NRZ)
  - Drawback: *Baseline wander* prevents receiver from using average of received signal to distinguish between 1 and 0
  - Drawback: Clock recovery in the presence of long runs of 0s or 1s in the data

- **Nonreturn to Zero Inverted** (NRZI): Transition for a “1”, no transition for a “0”
Manchester (phase) encoding

- Manchester encoding:
  - Exclusive-OR of the NRZ signal and the clock signal
  - “0” is a low-to-high transition; “1” is a high-to-low

- Transition guaranteed on every bit

- “Phase encoding” in the experimental Ethernet [Metcalfe et al.]

- Drawback: Halves data rate
4B/5B encoding

• So instead, later Ethernet standards use a block code called 4B/5B

<table>
<thead>
<tr>
<th>4-bit data</th>
<th>5-bit code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>11110</td>
</tr>
<tr>
<td>0001</td>
<td>01001</td>
</tr>
<tr>
<td>0010</td>
<td>10100</td>
</tr>
<tr>
<td>0011</td>
<td>10101</td>
</tr>
<tr>
<td>0100</td>
<td>01010</td>
</tr>
<tr>
<td>0101</td>
<td>01011</td>
</tr>
<tr>
<td>0110</td>
<td>01110</td>
</tr>
<tr>
<td>0111</td>
<td>01111</td>
</tr>
<tr>
<td>1000</td>
<td>10010</td>
</tr>
<tr>
<td>1001</td>
<td>10011</td>
</tr>
<tr>
<td>1010</td>
<td>10110</td>
</tr>
<tr>
<td>1011</td>
<td>10111</td>
</tr>
<tr>
<td>1100</td>
<td>11010</td>
</tr>
<tr>
<td>1101</td>
<td>11011</td>
</tr>
<tr>
<td>1110</td>
<td>11100</td>
</tr>
<tr>
<td>1111</td>
<td>11101</td>
</tr>
</tbody>
</table>

• Properties
  – No code has more than one leading zero
  – No code has more than two trailing zeros
  – When sent back-to-back, no pair of 5-bit codes contains more than three consecutive zeros

• Encoding process:
  1. Encode data using lookup table
  2. Send coded bits with NRZI
Ethernet framing

- **Framing**
  - Beginning of frame determined by presence of carrier
  - End of frame determined by absence of carrier
  - **Preamble**: 10101010 produces a square wave that allows receiver to frame bits

- **CRC (Cyclic Redundancy Check)** protects against errors on the Ether
  - Does not guard against errors introduced by the tap: rely on higher-layer checksums

- **Destination** address allows filtering at the link layer
Collisions

- Packet of $N$ bits: $N/B$ seconds on the wire
- From the perspective of a receiver (B):
  - Overlapping packets at B means signals sum
  - Not time-synchronized: result is bit errors at B
  - No fate-sharing among receivers: C receives okay in this example

Propagation delay: $\tau$ seconds
Who gets to transmit?

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

1. Begin the transmission procedure at any time
2. **Carrier sensing**: never transmit a frame if you sense that another station is transmitting
3. **Collision detection**: while sending, immediately abort your transmission if you detect another station transmitting
Carrier sensing

- Mechanism: measure voltage on the wire
- Binary encoding: voltage depends on the data
  - Bit stream
  - Binary encoding
  - Manchester encoding: constant average voltage
Collision detection

- Paper isn’t clear on this point (authors did have a patent in the filing process)
- Mechanism: monitor average voltage on cable
  - Manchester encoding means your transmission will have a predictable average voltage $V_0$; others will increase $V_0$
  - Abort transmission immediately if $V_{\text{measured}} > V_0$
When does a collision happen?

- Suppose Station A begins transmitting at time 0
- Assume that the packet lasts much longer than $\tau$
- All stations sense transmission and defer by time $\tau$
  - Don’t begin any new transmissions
- At time $\tau$, will a packet be collision-free?
  
  Only if no other transmissions began before time $\tau$
How long does a collision take to detect?

• Suppose Station A begins transmitting at time 0
• $\tau$ seconds after Z starts, A hears Z’s transmission
• When does A know whether its packet collided or not? **At time $2\tau$**
Collision detection and packet size

- How big must packets be for collisions to be detectable?

- Transmit rate $B$ bits/second
- If packets take time $2\tau$, A will still be transmitting when Z’s packet arrives at A, so A will detect collision
- So minimum packet size > $2\tau B$ bits
- Experimental Ethernet:
  - $\tau = 5$ μs, $B = 3$ Mbits/s $\rightarrow 2\tau B = 30$ bits

- Why doesn’t Metcalfe & Boggs paper mention this?
Commercial Ethernet

- Commercial Ethernet
  - Data rate $B = 10$ Mbits/s
  - Maximum length: 500 m per segment with up to two repeaters (hubs)

- Repeater receives bits, relays them onto wire
  - $\tau = 20 \mu s$ worst case $\rightarrow 2\tau B = 400$ bits $= 50$ bytes
  - Could send complete packet; not see collision
Resolving collisions

• Upon abort (carrier detect), a station enters the **backoff state**

• **Key idea:** the colliding stations all wait a random time before carrier sensing and transmitting again
  
  — *How to pick the random waiting time?* (Should be based on how stations have data to send)
  
  — *How to estimate the number of colliding stations?*

• **Goal:** Engineer such that nodes will wait different amounts of time, carrier sense, and not collide
Slotted Ethernet backoff

• Backoff time is **slotted** and **random**
  – Station’s view of the where the first slot begins is at the end of the busy medium
  – Random choice of slots within a window, the **contention window** \((CW)\)

• **Goal:** Choose slot time so that **different** nodes picking **different** slots carrier sense and defer, thus **don’t collide**
Picking the length of a backoff slot

• Consider from the perspective of one packet
  1. Transmissions beginning > $\tau$ before will cause packet to defer
  2. Transmissions beginning > $\tau$ after will not happen (why not?)
• Transmissions beginning < time $\tau$ apart will collide with packet

• So should we pick a backoff slot length of $\tau$?
The problem of clock skew

• **No!** Slots are timed off the tail-end of the last packet
  – Therefore, stations’ clocks **differ by at most** $\tau$
  – This is called **clock skew** $\Delta$ ($-\tau < \Delta < \tau$)

• Suppose we use a backoff slot length of $\tau$
  – **Different** stations picking **different** slots **may collide**!
Picking slot time in presence of clock skew

- Want **other** station’s **other** slots to be in “OK” region
  - Then, transmissions in different slots won’t collide
  - Worst case clock skew: $\tau$
  - So, pick a slot time of $\tau + \tau = 2\tau$
Binary exponential backoff

• Binary exponential backoff (**BEB**): double CW size on each consecutive collision

• Stations wait some number of slots chosen uniformly at random from $CW = [0, 2^m-1]$
  – Reset $m \leftarrow 1$ upon a successful transmission
  – First retransmit ($m = 1$): pick from $[0, 1]$
  – Second retransmit ($m = 2$): pick from $[0, 1, 2, 3]$
  – ...
  – Tenth and higher retransmissions ($m \geq 10$): pick from $[0, 1, ..., 2^{10}-1]$

• Observe: stations transmitting new frames don’t take into account recent collisions, might transmit before stations in backoff
Enforcing consensus on collisions

- Ethernet as described so far: collision at B and C but not at A or Z
- Lack of consensus results in differing backoff windows
- Metcalfe & Boggs: When a station detects collision, it “momentarily jams the Ether to [ensure] that all other participants in the collision will detect interference and, because of deference, will be forced to abort.”
- Result: All stations agree there was a collision, double backoff windows, backoff, and retransmit
- Jamming signal of length $2\tau$ suffices to enforce consensus
Ethernet performance

- Divide time into
  - Variable-sized contention intervals,
  - Fixed size transmission intervals (of duration $t_{\text{packet}}$)

Efficiency:

$$\frac{t_{\text{packet}}}{t_{\text{packet}} + (2\tau)W}$$

Number of slots to acquire the Ether
Ethernet performance: Acquisition

• Suppose there are $Q$ stations waiting to send

• Assume stations know $Q$ and send with probability $1/Q$ (BEB approximates this)

• What’s the probability $p_{acquire}$ that one station acquires the medium without a collision?

\[
p_{acquire} = Q \left( \frac{1}{Q} \right) \left( 1 - \frac{1}{Q} \right)^{Q-1}
\]

\[
= \left( 1 - \frac{1}{Q} \right)^{Q-1}
\]

\[
\approx \frac{1}{e}
\]

\[
\approx 37\%
\]
Ethernet performance: Waiting time

$W$: number of slots in a contention window before acquisition of the Ether

- Probability of no wait: $p_{\text{acquire}}$
- Probability wait one slot: $(1 - p_{\text{acquire}})p_{\text{acquire}}$
- Probability wait two slots: $(1 - p_{\text{acquire}})^2 p_{\text{acquire}}$
- $E[\text{slots to wait}] = E[W] = (1 - p_{\text{acquire}})/p_{\text{acquire}} = e - 1$
Today

1. Channel partitioning:
   - Time division multiple access (TDMA)
   - Frequency division multiple access (FDMA)
   - Code division multiple access (CDMA)

2. Random access protocol: ALOHA
   - Slotted ALOHA
   - Unslotted ALOHA

3. Random access protocol: the Ethernet
Comparing CDMA and ALOHA random access

• **CDMA wireless**
  ✓ No interference between transmitting stations
  ✓ Adaptation to varying numbers of users possible by changing Hadamard matrix
  ❌ Reduced rate of individual transmissions
  ❌ Unused codes waste overall capacity

• **ALOHA random access**
  ✓ Stations can transmit using the entire medium, at full rate if alone
  ✓ Almost-instant adaptation to varying traffic loads
  ❌ Concurrent transmissions result in collisions, loss of throughput
Wireless Networks: 802.11

Pre-Reading: P & D Section 2.7, to 2.7.1 (inclusive)

NEXT TIME