6.033: Networking: Wireless
Lecture 13
Katrina LaCurts, lacurts@mit.edu

0. Today
  - 802.11 wireless. Relatively new: standardized in 1997.
  - What's different about it? What challenges must it overcome? How does it interact with existing protocols?

1. Broadcast
  - Wireless channels are broadcast media: Anyone in range can hear your wireless communications.
  - Typical 802.11 scenario: multiple clients trying to send their data to an access point.
  - Problem: need to prevent two (or more) clients from sending at once.
    - Analogy: if everyone in the room talks at once, I can't tell what any of you are saying.
    - Real life: communications are ultimately signals. If you're sending on the same channel, and two signals overlap in space/time, it's impossible to pull those signals apart.
  - When two clients send at the same time, we say their packets have collided. Clients know when collisions happen, but we want to prevent them if possible.

2. MAC Protocols
  - MAC -- Media Access Control -- protocols mitigate collisions
  - In wired networks:
    - Two endpoints on a wire can't send at the same time. It's easy to coordinate: A sends, then B sends, then A sends, etc.
    - In practice: modern ethernet provides full duplex communications. An ethernet cable contains wires running in each direction; there is one set for sending in one direction and another for the other. Effectively, two separate channels.
  - In wireless networks:
    - More complicated. Clients can join or leave the channel, and there are frequently more than two clients.
  - TDMA (Time Division Multiple Access)
    - Divide time into slots, give all clients an ID (1, 2, .., N). Transmit when the timeslot mod N == your ID.
    - Assume there is some way for the AP to communicate which slot we're in, what N is, etc.
    - Pros: zero collisions
    - Cons: can't adapt to skewed (bursty) workloads. If you have no data to send in your slot, that slot is wasted. Also hard to coordinate when nodes leave/join.
  - CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)
    - Protocol:
      1. Each node keeps a probability, p_i.
      2. When the a node has a packet to send, it sends with

probability p_i.

3. If the send is successful, p_i = min(2*p_i, p_max)
   Else p_i = max(p_i/2, p_min)

   The p_max, p_min values are there to improve fairness, among other things.

(We could improve this protocol a bit by also having nodes sense whether the medium is idle before they send.)

- 802.11 uses CSMA/CA. Good because it can handle bursty traffic, which most traffic is. Is not perfect: collisions can happen.
  - Tradeoff between collisions and performance

3. MACs don't solve everything
  - Hidden terminals: some clients can't sense each other, will send, but their communication collides at the receiver.
  - Exposed terminals: some clients *can* sense each other, and so won't send at the same time. However, their communications *don't* collide at their respective receivers, so it would actually be okay to send.
  - (See slides for diagrams of both of these scenarios)
  - Underlying issue: sensing happens at the sender, but its interference at the receiver that matters.
  - Solution (?): RTS/CTS
    - Senders send requests to send (RTS)
    - If okay, AP will respond with clear to send (CTS)
    - If a client hears a CTS message meant for someone else, it won't send (knows that someone else has control of the channel for now)
    - (See slides for how this mechanism addresses both the hidden and exposed terminal issues)
  - In practice: there is a ton of overhead. Tried to fix a boundary case, but screwed up the common case.

4. Loss
  - Loss in wireless networks comes not just from collisions. There can be interference from other parts of the environment.
  - As a result, the quality of the channel can change over time (rapidly, in fact).
  - Ultimately, we're just communicating bits. We have choices about how to encode data: more redundancy = more robustness to errors, but worse performance.
  - We do "channel coding": match your encoding (level of redundancy) to the characteristics of the channel. Bad channel = more redundancy.
  - Easy for wired links: channel conditions don't really change.
  - In wireless, we do "bit rate selection"
    - 802.11 wireless offers different types of data encoding, which results in different bit rates (less redundancy = higher bit rate)
    - Senders try different rates, pick the one that achieves the

highest throughput after accounting for packet loss.

(Aside: the encoding scheme is one of a few things that can affect the bit rate in 802.11. We don't care about the others in this lecture, but it's important to me that you know that other things are going on.)

5. 802.11 vs. TCP
   - In 802.11:
     - Delay varies a lot, due to mobility, changing channel conditions, MAC protocol, etc. Makes RTT hard to estimate. Thus, can screw up TCP's timeout.
     - Packet loss is not always -- sometimes, but not always -- a signal of congestion. Backing off might not be the right thing to do in some cases.
   - Should we design a specific TCP for wireless networks? Maybe, but your traffic usually traverses both wireless and wired networks. Hard to reason about those interactions.

6. 802.11 vs. addressing (and, again, TCP)
   - In 802.11:
     - Can move around. Your IP address will (likely) change as you move from one AP to another.

     => Have to restart all of your TCP connections
   - If we keep IP addresses the same?
     - Might help, but changes our assumption about what an address means (right now: it indicates your attachment point to the network, not you yourself)
     - We would still likely experience periods of loss when we switch APs, causing TCP to timeout

7. Cool things about 802.11
   - Mobility, though complicated, is awesome
   - Opportunistic routing
     - Can have clients in a mesh network (lots of wireless clients with no wired "backhaul" links) that need to communicate.
     - Can exploit broadcast: sometimes the channel will be good, and packets might get farther in than you expected.
   - Other ways to exploit broadcast: Can use it to disseminate data to a lot of people at once, instead of running N separate connections.
   - Can see through walls: https://www.youtube.com/watch?v=sbFZPPC7REc

8. Up next in 6.033
   - You understand the client/server model: how the client and servers work (operating systems), what's going on in between them (networking)
   - After spring break: break things (see what I did there?)
     - Random failures: how do we deal with machine or network failures, especially as we run applications across many machines

  such as in DNS? What do those applications need the underlying
  technologies to provide?
- Malicious failures: How do we design systems that are resilient
  against not just random, but targeted attacks?