6.033: Networking: P2P Networks + CDNs
Lecture 12
Katrina LaCurts, lacurts@mit.edu

0. Introduction
  - Delivering content on the Internet. In some ways, one of the original applications!

1. Options for delivery
  - Client/server
    - client requests file, server responds with the data
    - HTTP, FTP work this way
  - Downsides: single point of failure, though we could add backups; Doesn't scale
  - CDNs
    - Buy multiple servers, put them near clients to decrease latency
    - No single point of failure, scales better
  - Peer-to-peer (P2P) networks for file-sharing
    - Distribute the architecture to the extreme
    - Once a client downloads (part of) the file from the server, that client can upload (part of) the file to others. Put clients to work!
    - In theory: infinitely scalable

2. P2P basics (note: some of these details are specific to the BitTorrent protocol, but they illustrate issues with many P2P networks)
  - Basics of original BitTorrent (BT) protocol:
    - Create a .torrent file, which contains meta-information about the file (file name, length, info about pieces that comprise the file, URL of tracker)
    - Have a tracker. A server that knows the identity of all the peers involved in your file transfer.
    - To download:
      - Peer contacts tracker
      - Tracker responds with list of other peers involved in transfer
      - Peer connects to these other peers, begins to transfer blocks (see below)
      - Some peers are seeders: already have the entire file (maybe servers that host the file, or just nice peers who are sticking around)
  - In the actual download, peers request blocks: pieces of pieces
    - Details/terminology doesn't matter. Just know that blocks are small (~16KB) chunks of the file.
    - Request blocks in a random order (more or less)
  - What incentivizes users to upload (UL) rather than just download(DL)ing?
    - High-level: users aren't allowed to DL from a user unless they're also ULing to that user
      - So peers want mutual interest: A has to have blocks that B

needs, and vice versa.
    - Protocol is divided into rounds. In round n, some number of peers upload blocks to Peer X. In round n+1, Peer X will send blocks to the peers that uploaded the most in round n. (Typically, to the top four peers.)
    - How do peers get started? Each peer reserves some (small) amount of bandwidth to give away freely
  - This method of incentivizing peers is part of what allowed P2P file-sharing to take off.
  - Problem #1: tracker is central point of failure
    - Most BT clients today are "trackerless", and use Distributed Hash Tables (DHTs) instead.
  - Problem #2: Capacity of these networks is limited by users' upload capacity, which tends to be drastically lower than their download capacity.

3. CDN basics (note: some details here are specific to the Akamai CDN)
  - CDNs can help us solve the capacity issue. They are run by a single company. Customers own the content, and pay CDNs to deliver it to users.
  - Setup: Thousands of servers around the world. Replicate content on those servers, users download from the "best" server (typically one very close to them, physically)
  - Questions:
    - Where to put those servers?
    - How to keep content up to date? (that's for the next part of 033)
    - How to get clients to the "right" server?
  - Getting users to the right place: DNS! Akamai's DNS servers redirect you to a particular host. Try it out: dig web.mit.edu
  - Akamai cares a lot about latency because of how it affects throughput (especially wrt TCP)
  - To reduce latency further, they do clever transport-layer tricks, and also find better routes than BGP in some cases (like RON!)

4. Summary
  - File-sharing has changed as the way we use the Internet has changed
  - Different architectures are affected by the underlying network. It's not just that the network exists to exchange messages; the network itself -- what it provides, its limitations -- *matter*.
  - These large, distributed systems (such as CDNs) also involve interesting problems in the realm of failures, and in keeping data consistent across multiple servers, spreading data across servers, etc. We'll begin to tackle those problems next week.