0. Introduction
- Last time: TCP CC. Massive success. Doesn't require us to change the network, is something machines can opt-in to (don't have to have reliable transport if you don't need it), lets us prevent congestion in a distributed manner.
- But:
  - Can result in long delays when routers have too much buffering
  - Doesn't work well in some scenarios (DCTCP)
  - Most important for today: doesn't react to congestion until queues are full.
  - Full queues = long delay
  - Queues = necessary to absorb bursts
- Goal: Transient queues, not persistent queues
- Idea: drop packets *before* the queues are full. TCP senders will back off before congestion is too bad.

1. DropTail
   - The original queue management scheme. When a packet arrives, if the queue is full, drop it; else, enqueue it.
   - Pro: Simple
   - Pro (or con?): Only drops packets when it needs to
     - Remember: dropped packet => retransmission, which wastes resources
   - Con: Synchronizes sources. Flow synchronization = decreased utilization
   - More cons:
     - Not very fair
     - Tends to result in mostly-full queues
     - Bad for bursty traffic

2. RED
   - Active queue management scheme
   - Idea: drop packets before the queue is full to give senders an early signal
   - Requires a measure of the average queue size, q_avg.
     \[ q_{\text{avg}} = a*q_{\text{instant}} + (1-a)*q_{\text{avg}} \quad ; \quad 0 < a \ll 1 \]
   - Drop packets with increasing probability as queue grows
   - Results:
     - Queue length doesn't oscillate as much (good)
     - Flows are desynchronized (good)
     - But, it still drops packets (bad?)

3. ECN
   - RED, but "mark" packets instead of dropping them
     - "Mark" = set a bit in the header to 1. Sources learn about congestion via marked ACKs
- Seems great as long as senders react to marks

4. RED/ECN vs. DropTail
- Advantages of RED/ECN
  - Smaller persistent queues => smaller delays
  - Less dramatic queue oscillation
  - Less biased against bursty traffic (in theory)
- Disadvantages
  - More complex
  - Hard to pick parameters (q_min, q_max, etc.)
    - "Right" parameters depend on number of flows, bottleneck, etc.
    - Bad parameters make things worse
  - This is the main disadvantage, from our point of view
- Neither RED nor ECN are the final word on active queue management

5. Traffic Differentiation
- As long as we're changing the switches themselves, why stop at queue management?
- Idea of traffic differentiation: put different types of traffic in different queues, and do something fancy with the queues.

6. Delay-based scheduling
- Suppose we want to prioritize latency-sensitive traffic. Say, zoom traffic (latency-sensitive) over email (not)
- Solution: priority queueing
  - Two queues: zoom queue, email queue. Serve zoom queue if it has a packet. If not, serve email queue.
  - Can extend this idea to more than two queues
- "What queue to send a packet from" is the problem of scheduling.
  That's different from queue management: "When to drop/mark packets in a single queue"
- Lingering problem: a lot of zoom traffic => starving out the email traffic

7. Bandwidth-based scheduling
- What if we, instead, want to allocate a certain amount of bandwidth to each queue?
- Round robin: Can't handle variable packet sizes
- Weighted round robin: Handles variable packet sizes, but needs avg queue length, and we'd have to decide how to deal with that
- Deficit round robin: Queues accumulate "credit" which specifies how many bytes they're allowed to send in the next round. Credit carries over to handle larger packet sizes.