Recap of CPU Scheduling

- Trade-off between throughput, response time, and fairness
- First Come First Serve (FCFS)
  - Simple, but long waiting times
- Round Robin (RR)
  - Fair, but context switching overhead
- Shortest Time to Completion First (STCF)
  - Optimal response times, but need to predict completion times

Example

- Job A: compute for 1000 seconds
- Job B: compute for 1000 seconds
- Job C
  ```c
  while (1) {
    compute for 1 ms
    Disk I/O for 10 ms
  }
  ```
  - A and B can each use 100% of CPU; C can use 91% of disk I/O
  - Goal: keep both CPU and disk busy

Real-time scheduling

- Focused so far on average response time
- Alternate goal: finish job before deadline
  - How far ahead of time job completes is irrelevant
  - Requires worst-case analysis
- Examples?
  - Video or audio output
  - Control of physical systems
  - How do we schedule for deadlines in life?

Earliest-deadline first (EDF)

- Always run job with the earliest deadline
- Preempt current job if a new job arrives with earlier deadline
- Optimal: Will meet all deadlines if possible to do so

Example of EDF

- Job A: New job every 30 secs, takes 15 secs, deadline is 20 secs after arrival
- Job B: New job every 45 secs, takes 10 secs, deadline is 30 secs after arrival
- Job C: New job every 35 secs, takes 5 seconds, deadline is 10 seconds after arrival
  ```text
  0  5  10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
  A +                    +                    +
  B +                    +                    +
  C                       +
  ```
Real-world scenario

- What scheduling strategy is used in grocery stores?

Threads and Concurrency

- Concurrent programming using threads simpler than event-based programming
- Threads must synchronize access to shared data
- Over-constrained synchronization → deadlock
- We can provide abstraction of infinite CPUs
- CPU scheduling controls policy in multiplexing CPU across threads