



ILLINOIS TECH

College of Computing

CS 450 Operating Systems

CPU Scheduling

Yue Duan

The Problem



- You're the cook at State Street Diner
 - customers continuously enter and place orders 24 hours a day
 - dishes take varying amounts to prepare
- What is your goal?
 - minimize average turnaround time?
 - minimize maximum turnaround time?
 - maximize throughput
- Which strategy achieves your **goal**?

Schedulers in OS



- CPU Scheduler
 - selects a process to run from the run queue
- Disk Scheduler
 - selects next read/write operation
- Network Scheduler
 - selects next packet to send or process
- Page Replacement Scheduler
 - selects page to evict
- We focus on **CPU scheduling**

Kernel Operation (conceptual, simplified)



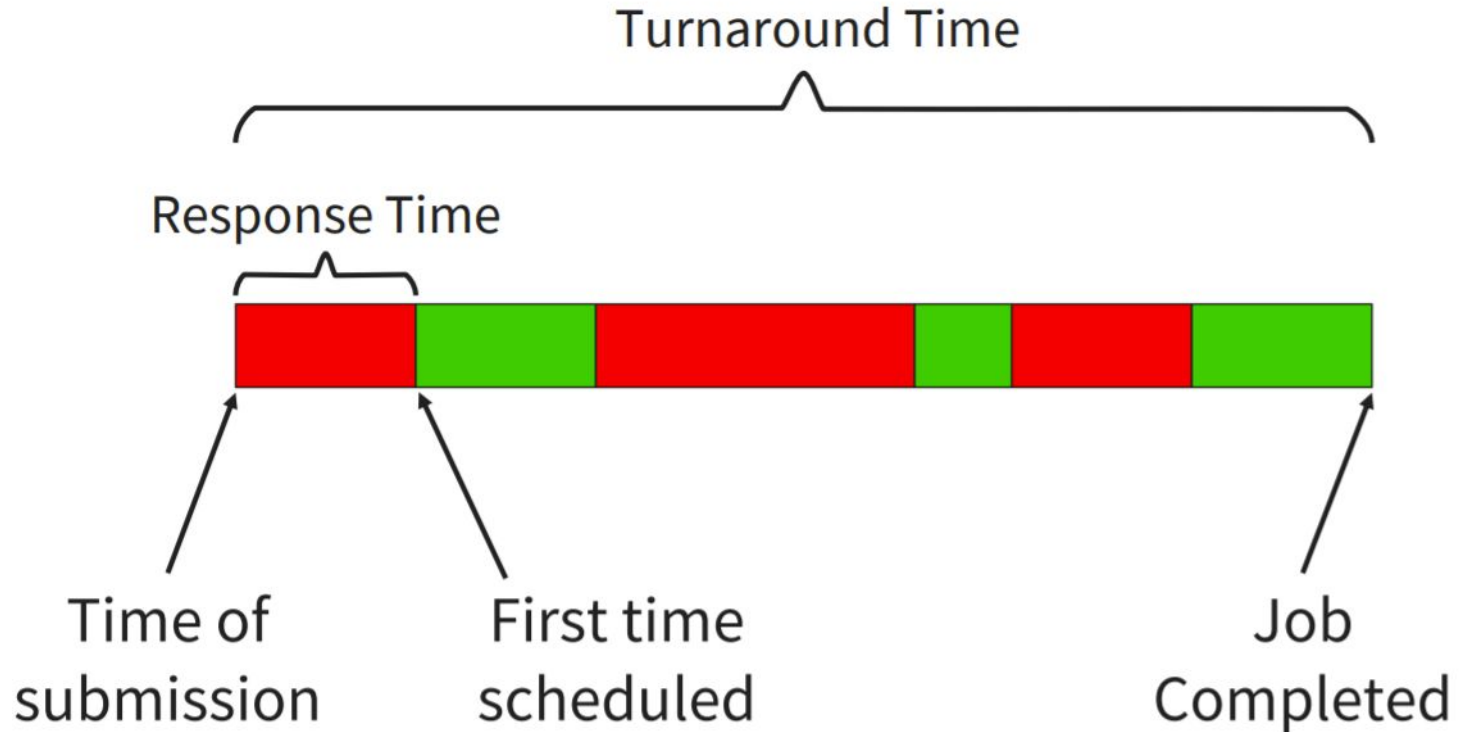
- 1. Initialize devices
- 2. Initialize “first process”
- 3. while (TRUE) :
 - while device interrupts pending
 - handle device interrupts
 - while system calls pending
 - handle system calls
 - **if run queue is non-empty**
 - **select process and switch to it**
 - otherwise
 - wait for device interrupt

Job Characteristics



- Job or Task
 - e.g., mouse click, web request, shell command, ...
- Job Arrival time
- Job Execution time
 - Time needed to run the task without contention

Important Metrics of Scheduling



Performance Terminology



- **Response Time:** When does it start?
 - User-perceived time before job can produce first
- **Execution Time:**
 - sum of green periods
- **Total Waiting Time:** How much thumb-twiddling?
 - sum of red periods
 - Time on the run queue but not running.
- **Turnaround Time:** How long?
 - sum of both
 - User-perceived time to complete some job.

Performance Terminology



- **Throughput:** How many jobs over time?
 - The rate at which jobs are completed.
- **Predictability:** How consistent?
 - Low variance in turnaround time for repeated jobs.
- **Overhead:** How much useless work?
 - Time lost due to switching between jobs.
- **Fairness:** How equal is performance?
 - Equality in the resources given to each job.
- **Starvation:** How bad can it get?
 - No progress for one job, due to resources given to other jobs.

The Perfect Scheduler

- Minimizes response time and turnaround time
- Maximizes throughput
- Maximizes utilization (aka “work conserving”):
 - keeps all devices busy
- Meets deadlines:
 - think watching a video, car brakes, etc.
- Is Fair:
 - everyone makes progress, no one starves
- Is Envy-Free:
 - no job wants to switch its schedule with another



No such scheduler exists!

When does scheduler run?



- Non-preemptive
 - Job runs until it voluntarily yields CPU:
 - job blocks on an event (e.g., I/O or P(sem))
 - job explicitly yields
 - job terminates
- Preemptive
 - All of the above, plus:
 - Timer and other interrupts
 - When jobs cannot be trusted to yield explicitly
 - Incurs some context switching overhead

Process Model

- Jobs switch between CPU & I/O bursts

CPU-bound jobs: Long CPU bursts



I/O-bound jobs: Short CPU bursts



- Problems:
 - don't know job's type before running
 - jobs also change over time

Basic scheduling algorithms:



- First in first out (FIFO)
- Shortest Job First (SJF)
- Round Robin (RR)

First In First Out (FIFO)

- Processes (jobs) P₁, P₂, P₃ with execution time 12, 3, 3 All have same arrival time

Scenario 1: schedule order P₁, P₂, P₃

Average Turnaround Time: $(12+15+18)/3 = 15$



Scenario 2: schedule order P₂, P₃, P₁

Average Turnaround Time: $(3+6+18)/3 = 9$



First In First Out (FIFO)



- + Simple
- + Low-overhead
- + No Starvation



- Average turnaround time very sensitive to schedule order



- Not responsive to interactive jobs

First In First Out (FIFO)



How to minimize average turnaround time?

Shortest Job First (SJF)

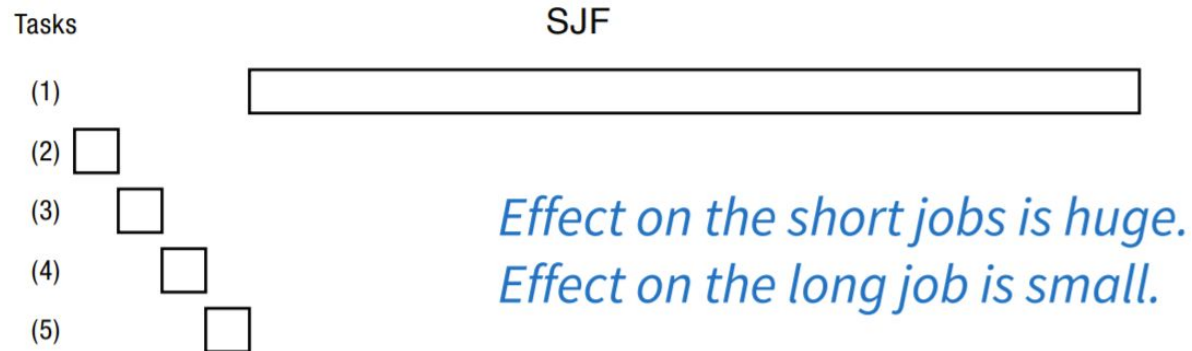
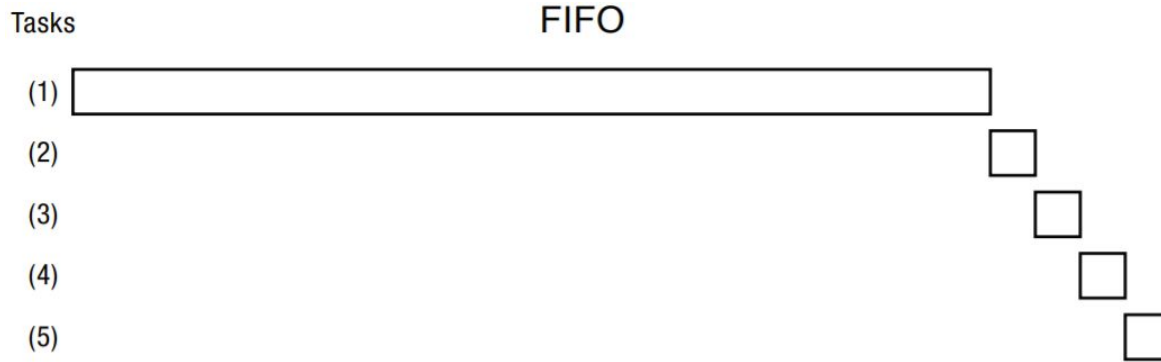
- Schedule in order of estimated execution time
- Scenario : each job takes as long as its number

Average Response Time: $(1+3+6+10+15)/5 = 7$



Would another schedule improve avg turnaround time?

FIFO vs. SJF



Shortest Job First Prediction

- How to approximate duration of next CPU-burst
 - Based on the durations of the past bursts
 - Use past as a predictor of the future
- No need to remember entire past history!
- Use exponential moving average:
 - t_n actual duration of nth CPU burst t_n predicted duration of n
 - τ_n CPU burst
 - τ_{n+1} predicted duration of (n+1)th CPU burst

$$\tau_{n+1} = \alpha \tau_n + (1 - \alpha) t_n$$

SJF Roundup



+ Optimal average
turnaround time



- Pessimal variance in
turnaround time
- Needs estimate of
execution time



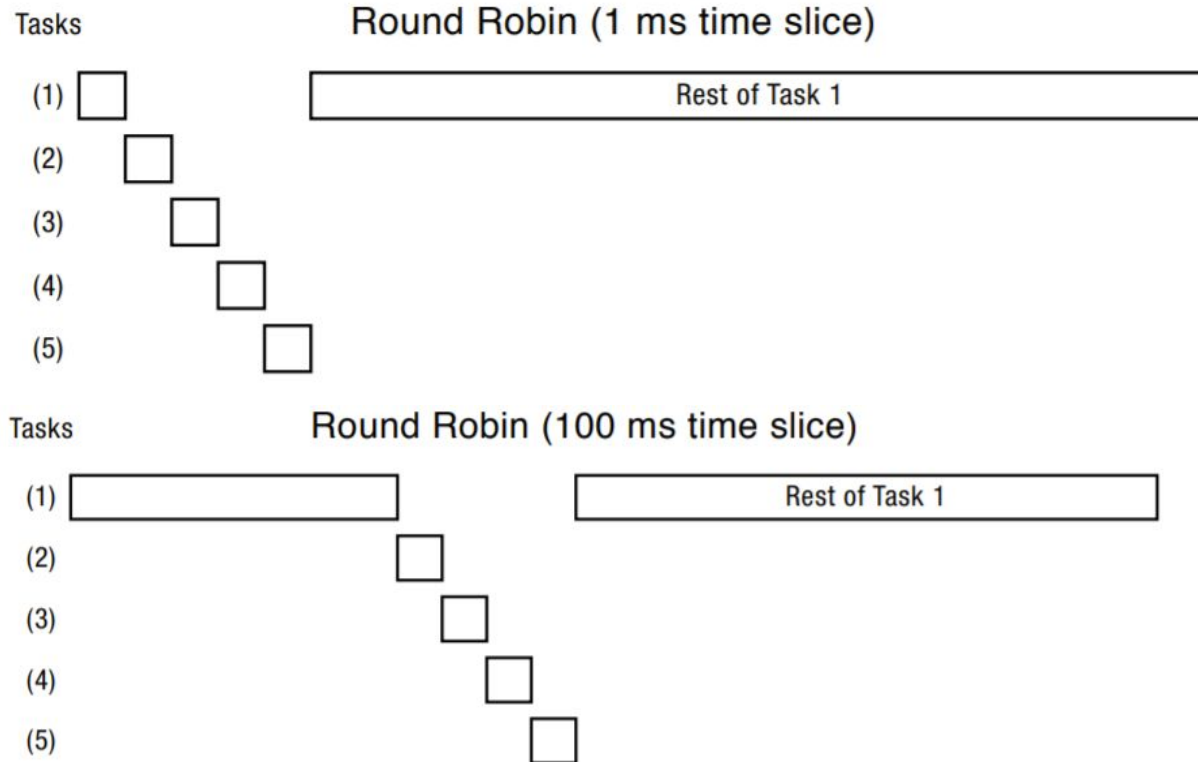
- Can starve long jobs

Round Robin (RR)



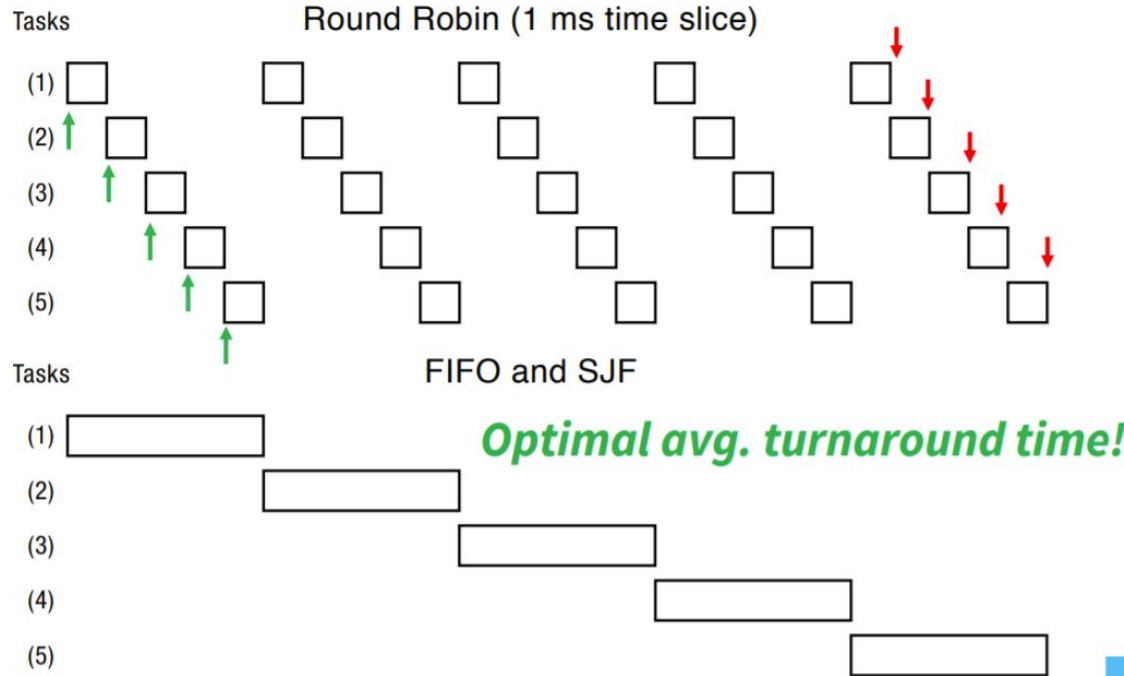
- Each job allowed to run for a **quantum**
- Context is switched (at the latest) at the end of the quantum
- What is a good quantum size?
 - Too long, and it morphs into FIFO
 - Too short, and much time is wasted on context switching
 - Typical quantum:
 - about 100X cost of context switch (~100ms vs. $\ll 1$ ms)

Effect of Quantum Choice in RR



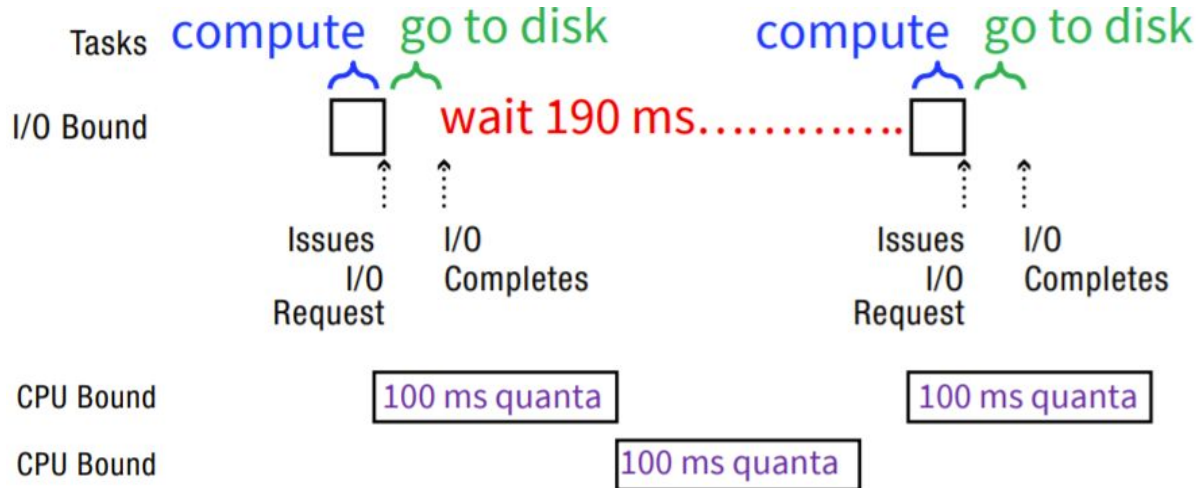
Round Robin vs. FIFO

- Tasks of same length that start ~same time



More Problems with Round Robin

- Mixture of one I/O Bound tasks + two CPU Bound Tasks
- I/O bound: compute, go to disk, repeat
- RR doesn't seem so fair after all....



Round Robin (RR)



- + No starvation
- + Can reduce response time



- Context switch overhead
- Mix of I/O and CPU bound



- bad avg. turnaround time for equal length jobs

Priority-based scheduling algorithms:



- Priority Scheduling
- Real-Time Scheduling
- Multi-level Queue Scheduling
- Multi-level Feedback Queue Scheduling

Priority Scheduling



- Assign a number to each job and schedule jobs in (increasing) order
- Can implement any scheduling policy
 - e.g., reduces to SJF if t_n is used as priority
- To avoid starvation, improve job's priority with time (aging)

Real-Time Scheduling



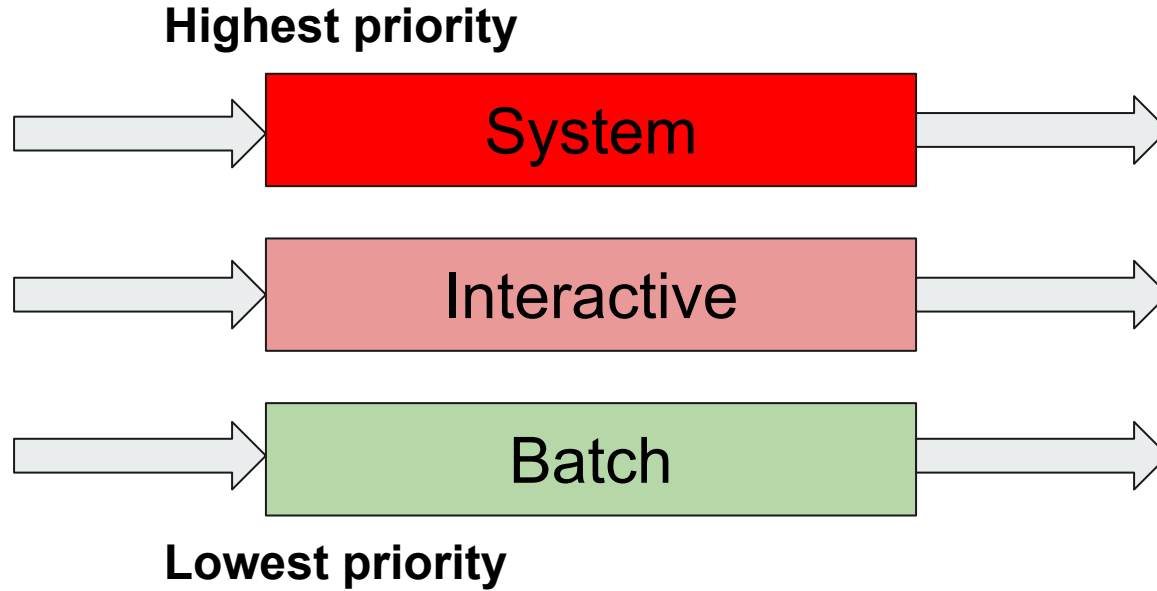
- Real-time processes have timing constraints
 - Expressed as deadlines or rate requirements
- Common RT scheduling policies
 - **Earliest deadline first** (EDF) (priority = deadline)
 - **Priority Inheritance**
 - Recall priority inversion: high priority process wants to get lock held by low priority process
 - Solution: High priority process (needing lock) temporarily donates priority to lower priority process (with lock)

Multi-Level Queue Scheduling



- Multiple ready queues based on job “type”
 - system jobs
 - interactive jobs
 - background batch jobs
 - Different queues may be scheduled using different algorithms
- Queue classification difficult (Job may have CPU-bound and interactive phases)
- No queue re-classification

Multi-Level Queue Scheduling

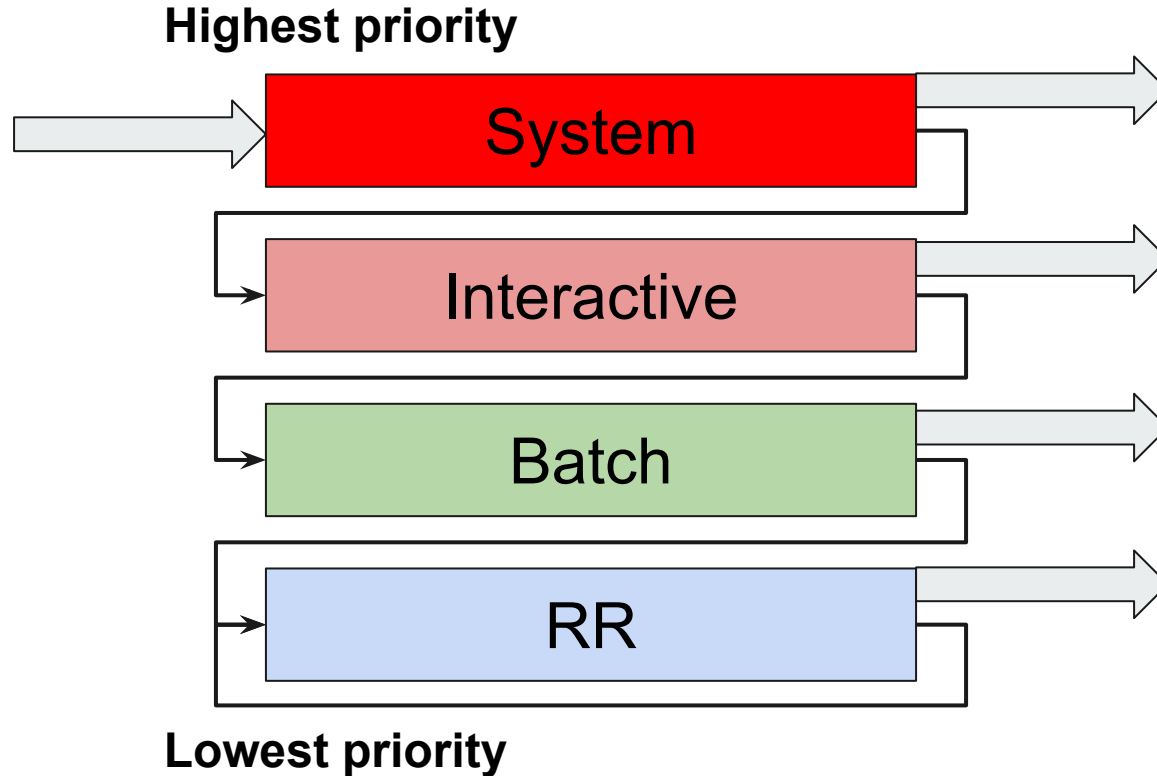


Multi-Level Feedback Queues



- Like multilevel queue, but assignments are not static
- Jobs start at the top
 - Use your quantum? **move down**
 - Don't? **Stay where you are**
- Need parameters for:
 - Number of queues
 - Scheduling alg. per queue
 - When to upgrade/downgrade job

Multi-Level Feedback Queues





THANK YOU!