CS 423
Operating System Design: Scheduling

Professor Adam Bates
Spring 2017

Goals for Today

- <u>Learning Objective</u>:
  - Introduce goals, definitions, and policies related to uniprocessor and multiprocessor scheduling
  - Reason about advantages and disadvantages of different foundational scheduling algorithms
- <u>Announcements, etc:</u>
  - MP1 is is out! Due **Feb 19**

Reminder: Please put away devices at the start of class

What Are Scheduling Goals?

- What are the goals of a scheduler?
- Scheduling Goals:
  - Generate illusion of concurrency
  - Maximize resource utilization (e.g., mix CPU and I/O bound processes appropriately)
  - Meet needs of both I/O-bound and CPU-bound processes
    - Give I/O-bound processes better interactive response
    - Do not starve CPU-bound processes
  - Support Real-Time (RT) applications

Definitions

- Task/Job
  - Something that needs CPU time: a thread associated with a process or with the kernel…
  - … a user request, e.g., mouse click, web request, shell command, …
- Latency/response time
  - How long does a task take to complete?
- Throughput
  - How many tasks can be done per unit of time?

Definitions

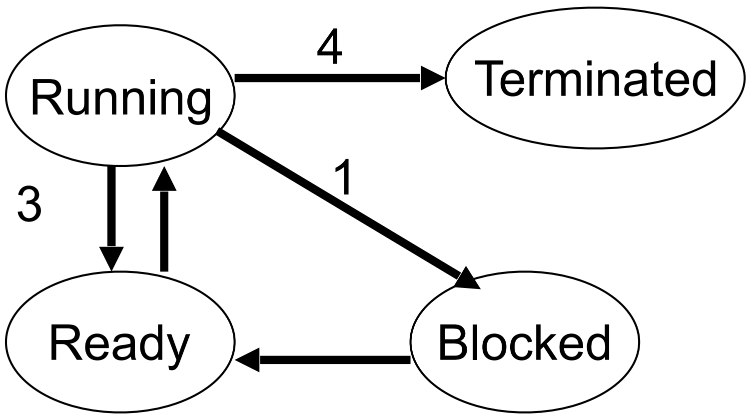
- Overhead
  - How much extra work is done by the scheduler?
- Fairness
  - How equal is the performance received by different users?
- Predictability
  - How consistent is the performance over time?
- Starvation
  - A task ‘never’ receives the resources it needs to complete
  - Not very fair : - (

Definitions

- Workload
  - Set of tasks for system to perform
- Work-conserving
  - Resource is used whenever there is a task to run
  - For non-preemptive schedulers, work-conserving is not always better

Definitions

- Non-preemptive scheduling:
  - The running process keeps the CPU until it voluntarily gives up the CPU
    - process exits
    - switches to blocked state
    - 1 and 4 only (no 3)



- Preemptive scheduling:
  - The running process can be interrupted and must release the CPU (can be forced to give up CPU)

Definitions

- Scheduling algorithm
  - takes a workload as input
  - decides which tasks to do first
  - Performance metric (throughput, latency) as output
  - Only preemptive, work-conserving schedulers to be considered

First In First Out (FIFO)

- Schedule tasks in the order they arrive
  - Continue running them until they complete or give up the processor
- Example: memcached
  - Facebook cache of friend lists, …
- On what workloads would FIFO be particularly bad?

First In First Out (FIFO)

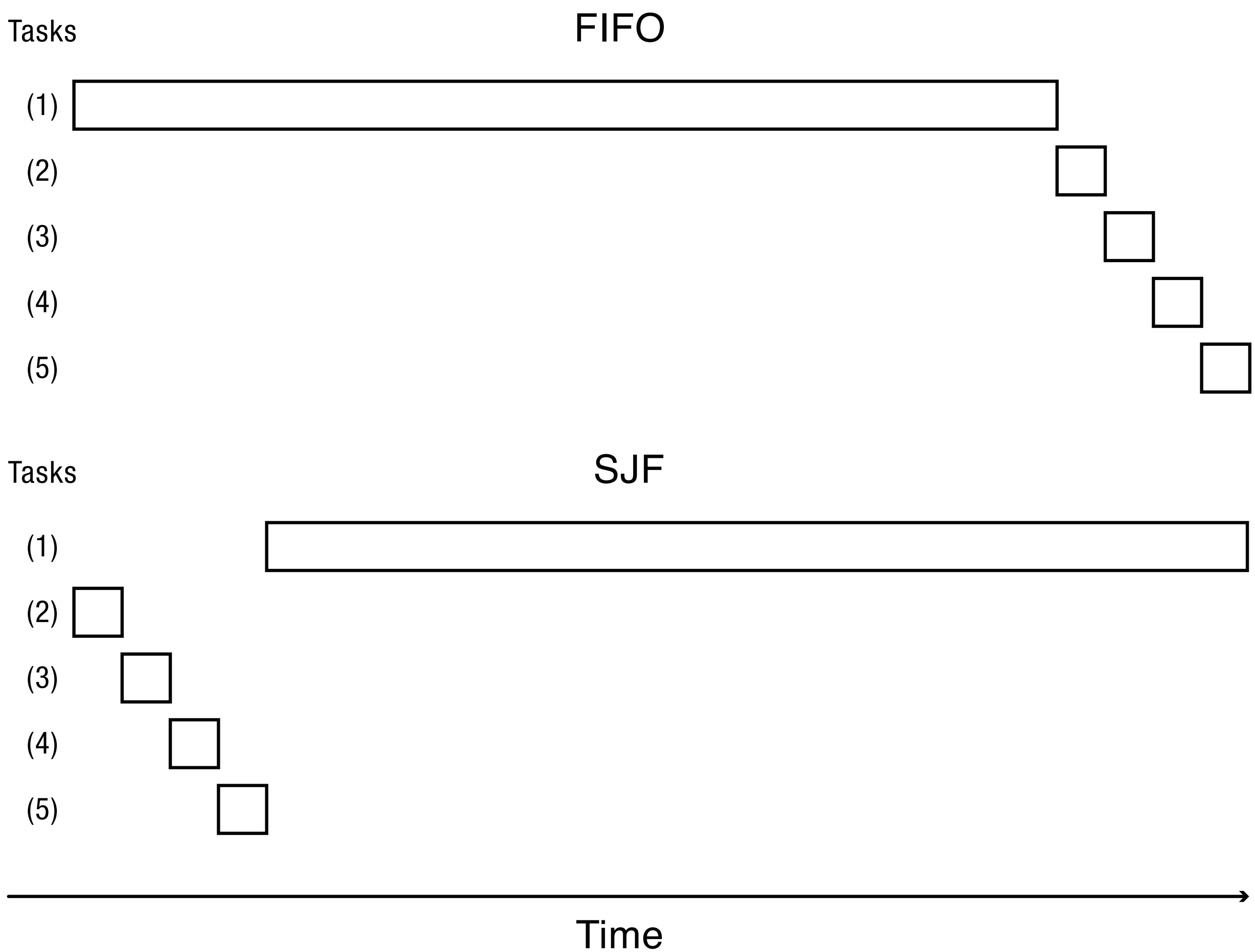
- Schedule tasks in the order they arrive
  - Continue running them until they complete or give up the processor
- Example: memcached
  - Facebook cache of friend lists, …
- On what workloads would FIFO be particularly bad?

Shortest Job First (SJF)

- Always do the task that has the shortest remaining amount of work to do
  - Often called Shortest Remaining Time First (SRTF)

- Suppose we have five tasks arrive one right after each other, but the first one is much longer than the others
  - Which completes first in FIFO? Next?
  - Which completes first in SJF? Next?

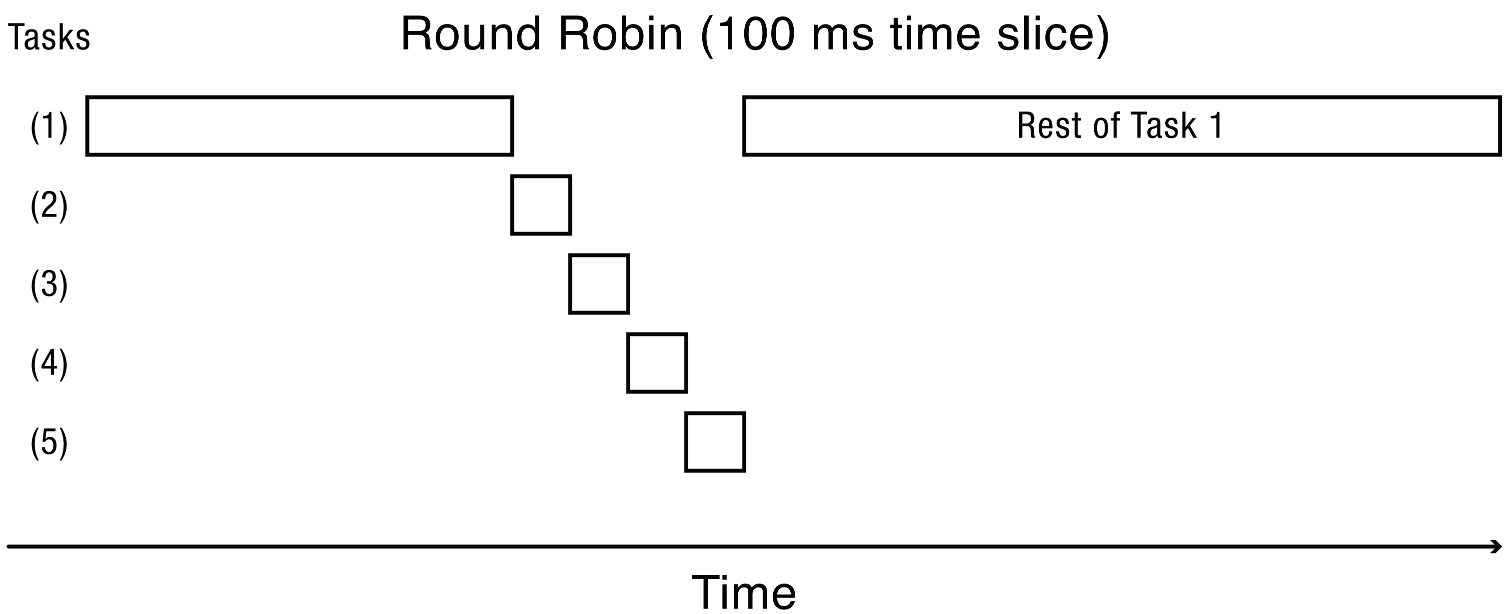
FIFO vs. SJF



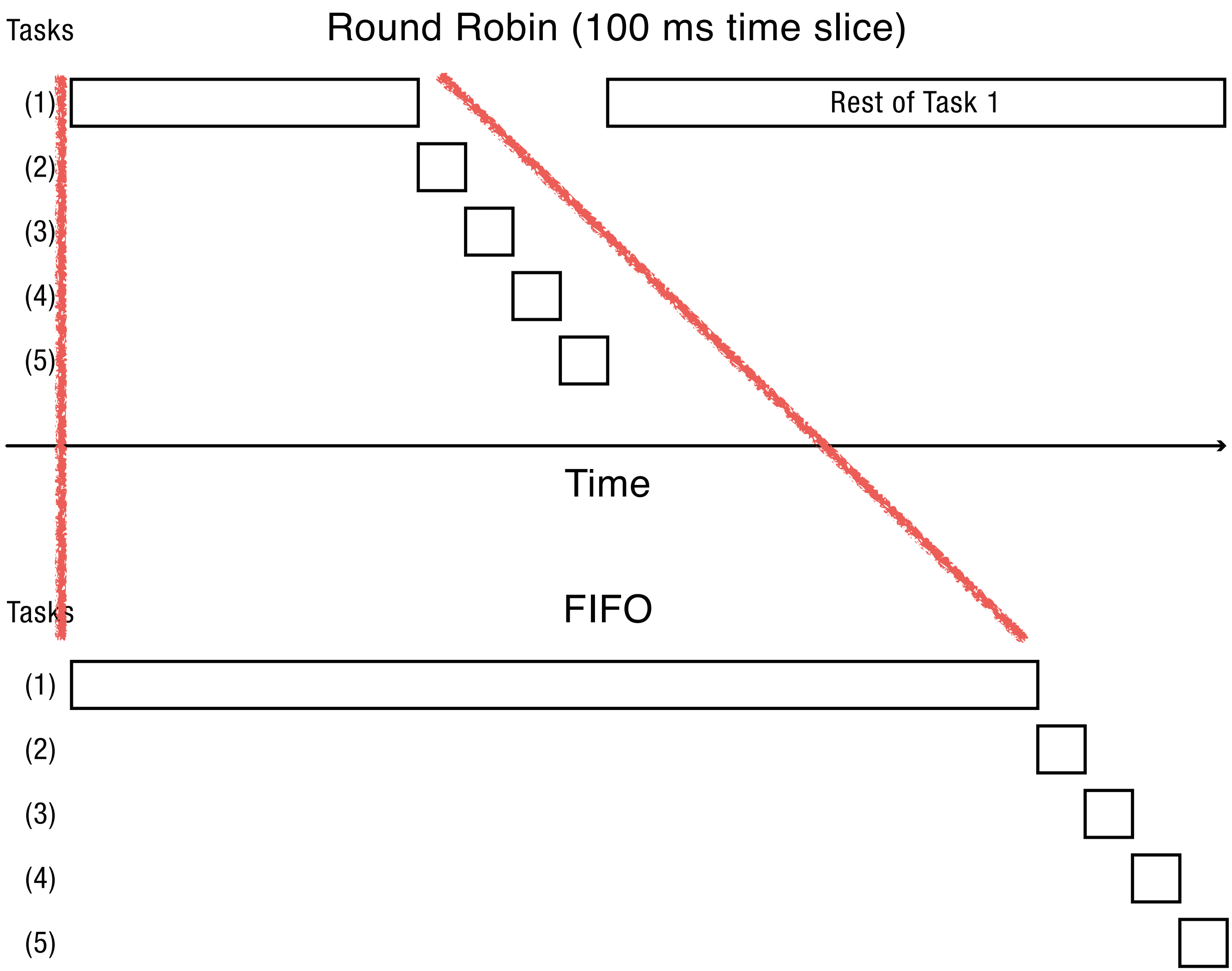
Round Robin (RR)

- Each task gets resource for a fixed period of time (time quantum)
  - If task doesn't complete, it goes back in line
- Performance changes based on time quantum size
  - What if time quantum is too short?
    - One instruction?
  - What if time quantum is too long?
    - Infinite?

Round Robin

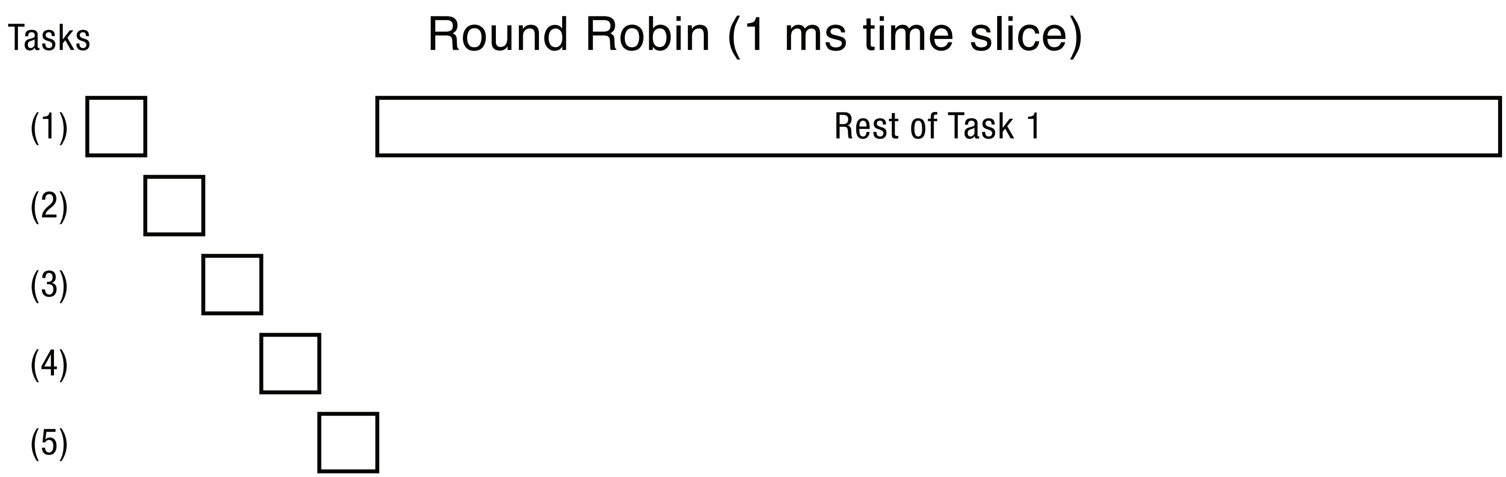


Round Robin

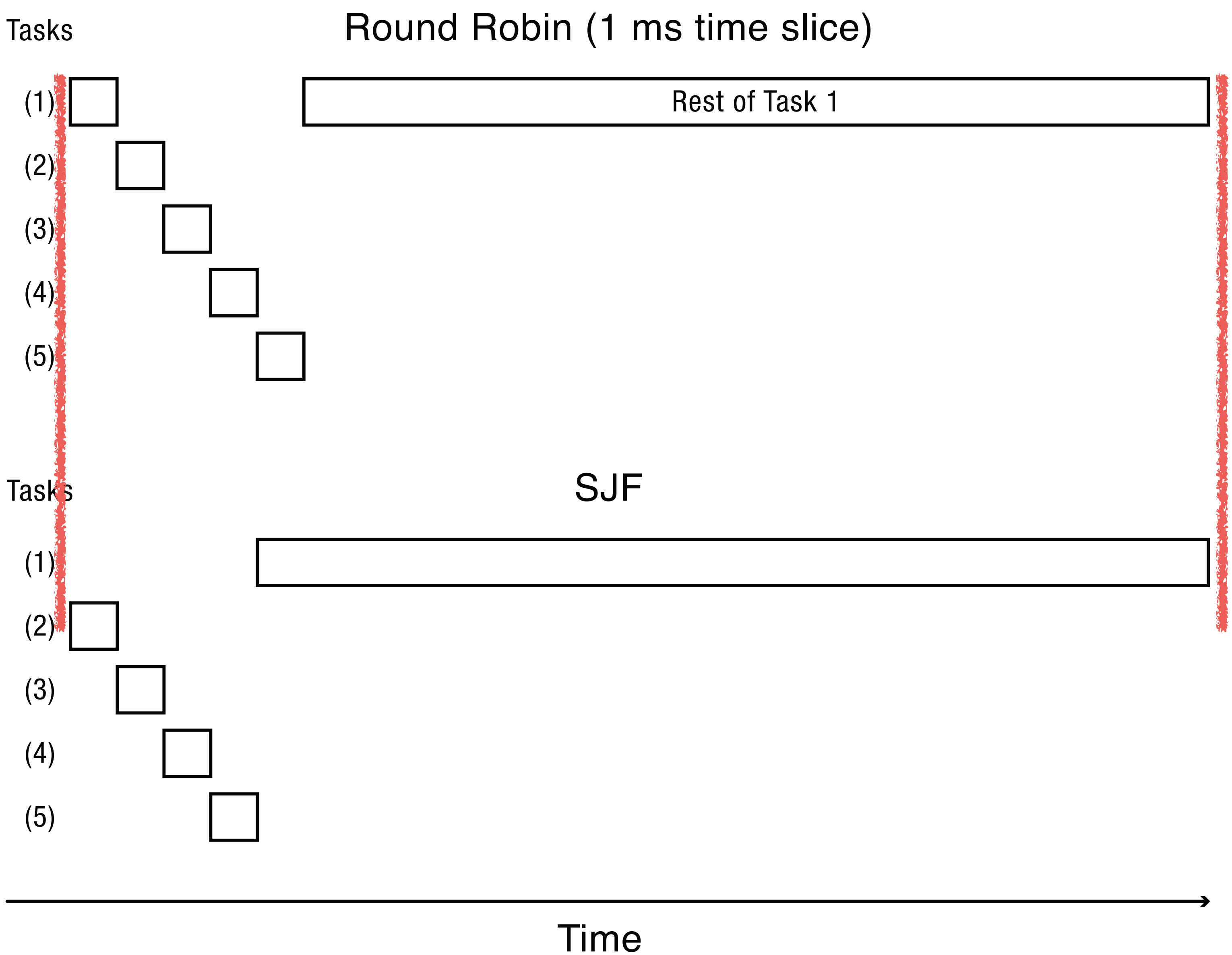


Tasks
Round Robin (100 ms time slice)
(1)
(2)
(3)
(4)
(5)
Rest of Task 1
Time
Tasks
FIFO
(1)
(2)
(3)
(4)
(5)

Round Robin



Round Robin



Tasks
Round Robin (1 ms time slice)
(1)
Rest of Task 1
(2)
(3)
(4)
(5)
Tasks
SJF
(1)
(2)
(3)
(4)
(5)
Time

Scheduling

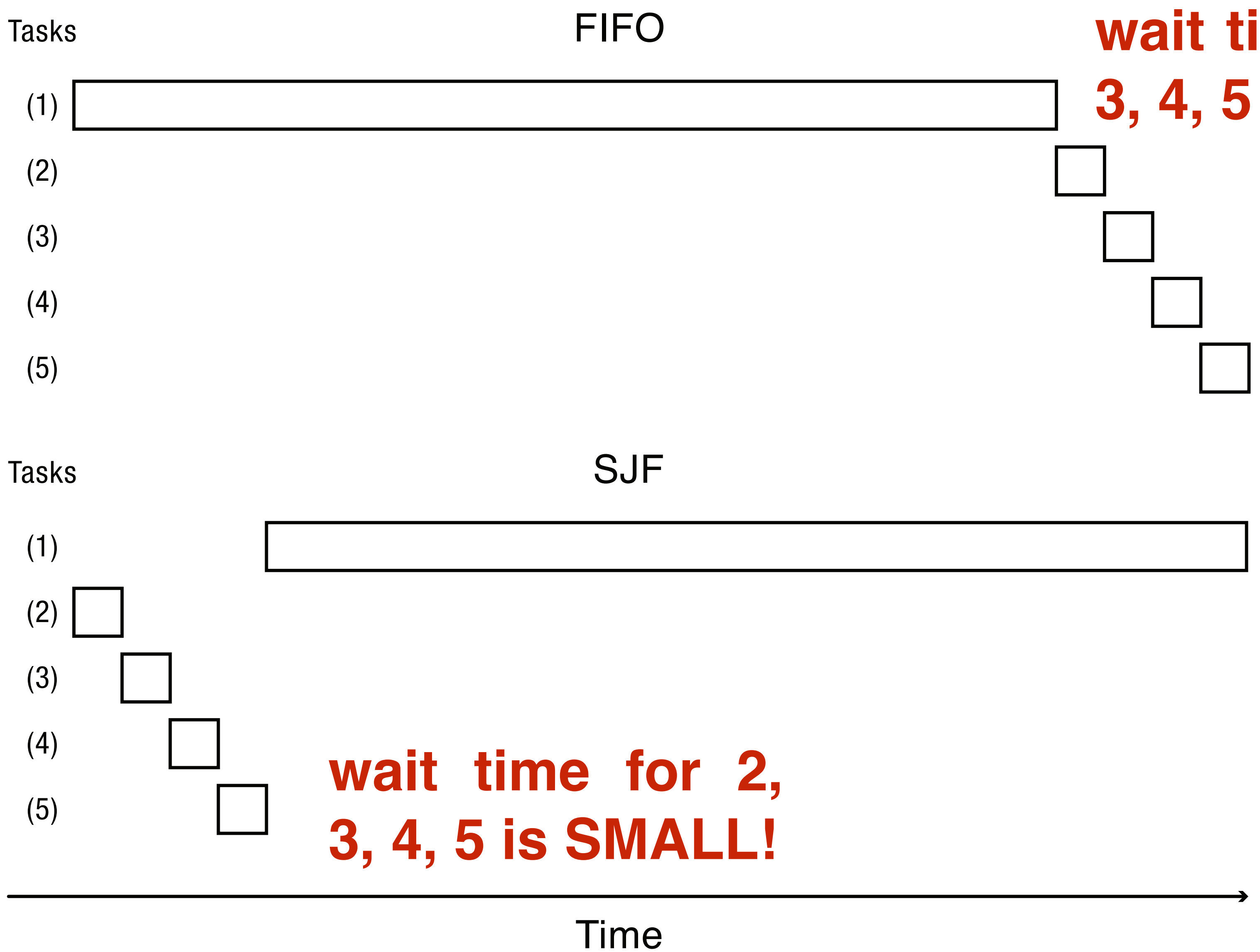
- Basic scheduling algorithms
  - FIFO (FCFS)
  - Shortest job first
  - Round Robin

Scheduling

- Basic scheduling algorithms
  - FIFO (FCFS)
  - Shortest job first
  - Round Robin

- What is an optimal algorithm in the sense of maximizing the number of jobs finished (i.e., minimizing average response time)?

FIFO vs. SJF



wait time for 2, 3, 4, 5 is BIG!

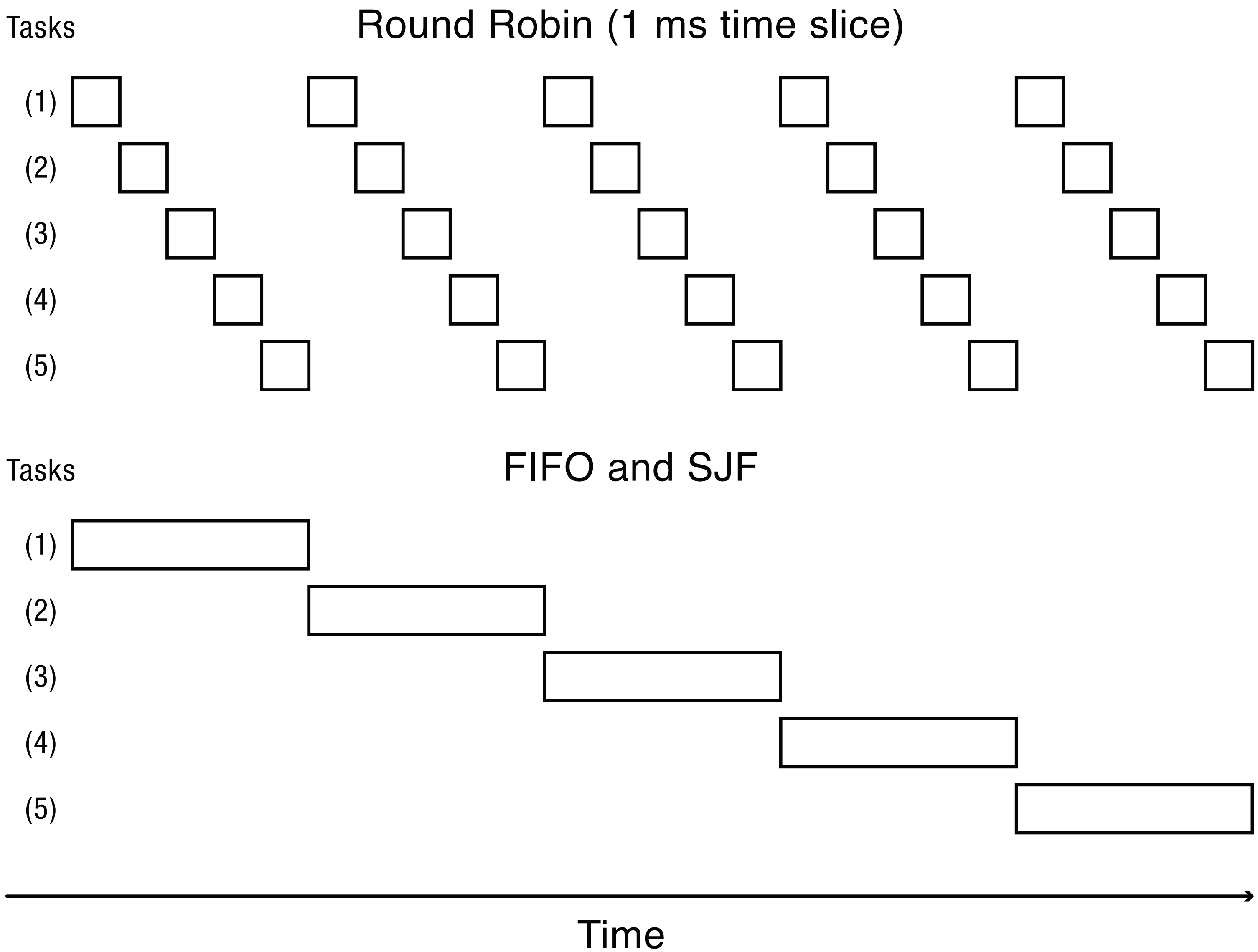
wait time for 2, 3, 4, 5 is SMALL!

Scheduling

- Basic scheduling algorithms
  - FIFO (FCFS)
  - Shortest job first
  - Round Robin

- Assuming zero-cost to time slicing, is Round Robin always better than FIFO?

RR v. FIFO (fixed size tasks)



Starvation, Sample Bias

- Suppose you want to compare two scheduling algorithms
  - Create some infinite sequence of arriving tasks
  - Start measuring
  - Stop at some point
  - Compute average response time as the average for completed tasks between start and stop
- Is this valid or invalid?

Sample Bias Solutions

- Measure for long enough that # of completed tasks >> # of uncompleted tasks
  - For both systems!
- Start and stop system in idle periods
  - Idle period: no work to do
  - If algorithms are work-conserving, both will complete the same tasks

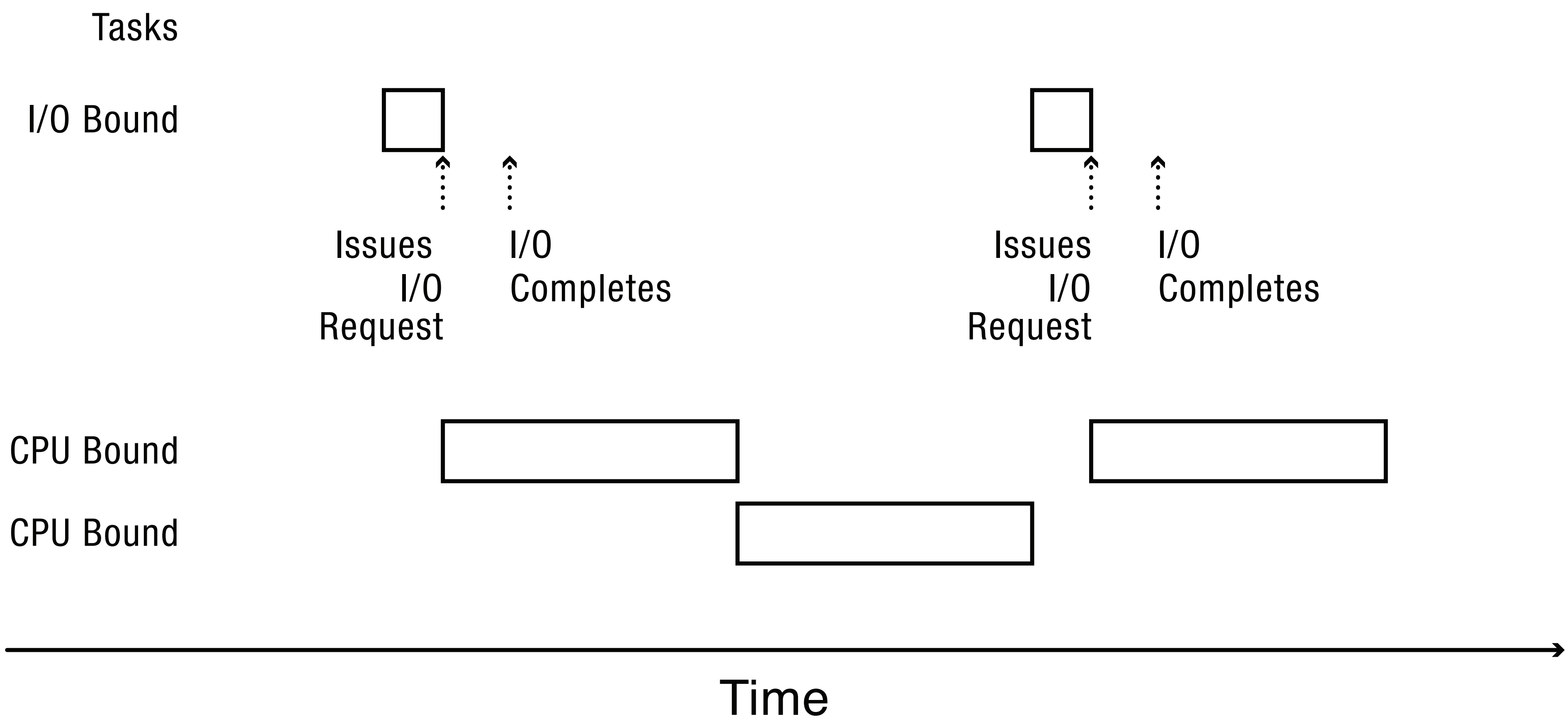
Round Robin = Fairness?

Is Round Robin the fairest possible algorithm?

What is fair?

- FIFO?
- Equal share of the CPU?
- What if some tasks don't need their full share?
- Minimize worst case divergence?
- Time task would take if no one else was running
- Time task takes under scheduling algorithm

Mixed Workloads??



Max-Min Fairness

- How do we balance a mixture of repeating tasks?
  - Some I/O bound, need only a little CPU
  - Some compute bound, can use as much CPU as they are assigned
- One approach: maximize the minimum allocation given to a task
  - If any task needs less than an equal share, schedule the smallest of these first
  - Split the remaining time using max-min
  - If all remaining tasks need at least equal share, split evenly

Multi-Level Feedback Queue

- Goals:
  - Responsiveness
  - Low overhead
  - Starvation freedom
  - Some tasks are high/low priority
  - Fairness (among equal priority tasks)
- Not perfect at any of them!
  - Used in Linux (and probably Windows, MacOS)

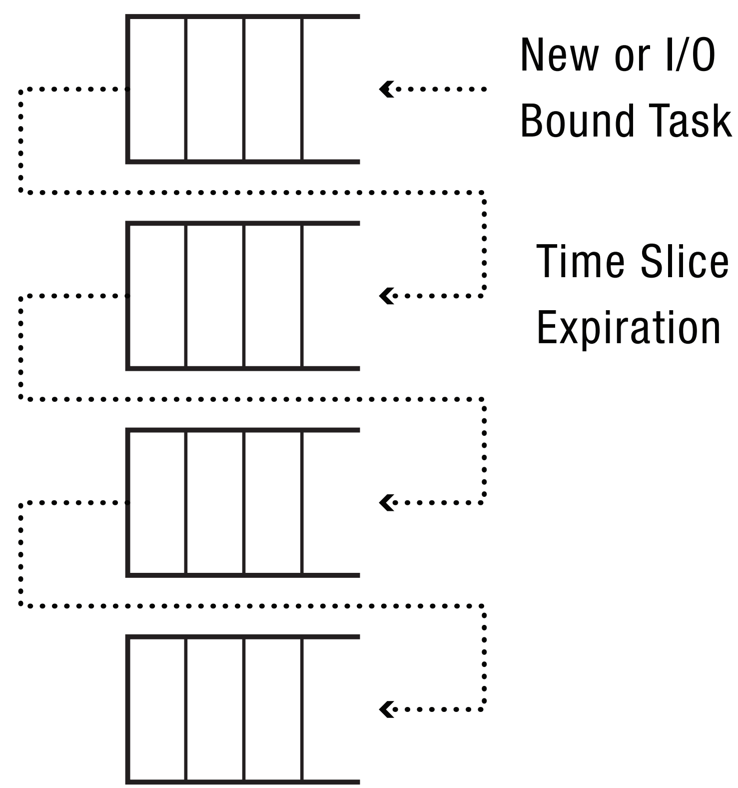
Multi-Level Feedback Queue

- Set of Round Robin queues
  - Each queue has a separate priority
- High priority queues have short time slices
  - Low priority queues have long time slices
- Scheduler picks first thread in highest priority queue
- Tasks start in highest priority queue
  - If time slice expires, task drops one level

Multi-Level Feedback Queue

Priority	Time Slice (ms)
1	10
2	20
3	40
4	80

Round Robin Queues



Summary

- FIFO is simple and minimizes overhead.
- If tasks are variable in size, then FIFO can have very poor average response time.
- If tasks are equal in size, FIFO is optimal in terms of average response time.
- Considering only the processor, SJF is optimal in terms of average response time.
- SJF is pessimal in terms of variance in response time.

Summary

- If tasks are variable in size, Round Robin approximates SJF.
- If tasks are equal in size, Round Robin will have very poor average response time.
- Tasks that intermix processor and I/O benefit from SJF and can do poorly under Round Robin.

Summary

- Max-Min fairness can improve response time for I/O-bound tasks.
- Round Robin and Max-Min fairness both avoid starvation.
- By manipulating the assignment of tasks to priority queues, an MFQ scheduler can achieve a balance between responsiveness, low overhead, and fairness.