

# Introduction to Operating Systems

CPSC/ECE 3220 Summer 2018

Lecture Notes  
OSPP Chapter 7 – Part A

(adapted by Mark Smotherman from Tom Anderson's slides on OSPP web site)

# Main Points

- Scheduling policy: what to do next, when there are multiple threads ready to run
  - Or multiple packets to send, or web requests to serve, or ...
- Definitions
  - Response time, throughput, predictability
- Uniprocessor policies
  - FIFO, Round Robin (RR), optimal
  - Multilevel feedback (MFQ) as approx. of optimal

# Example

- You manage a web site that suddenly becomes wildly popular. Do you?
  - Buy more hardware?
  - Implement a different scheduling policy?
  - Turn away some users? Which ones?
- How much worse will performance get if the web site becomes even more popular?

# Definitions

- Task/Job
  - 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?
- 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?

# More Definitions

- Workload
  - Set of tasks for system to perform
- Preemptive scheduler
  - If we can take resources away from a running task
- Work-conserving
  - Resource is used whenever there is a task to run
  - For non-preemptive schedulers, work-conserving is not always better
- 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 is FIFO particularly bad?

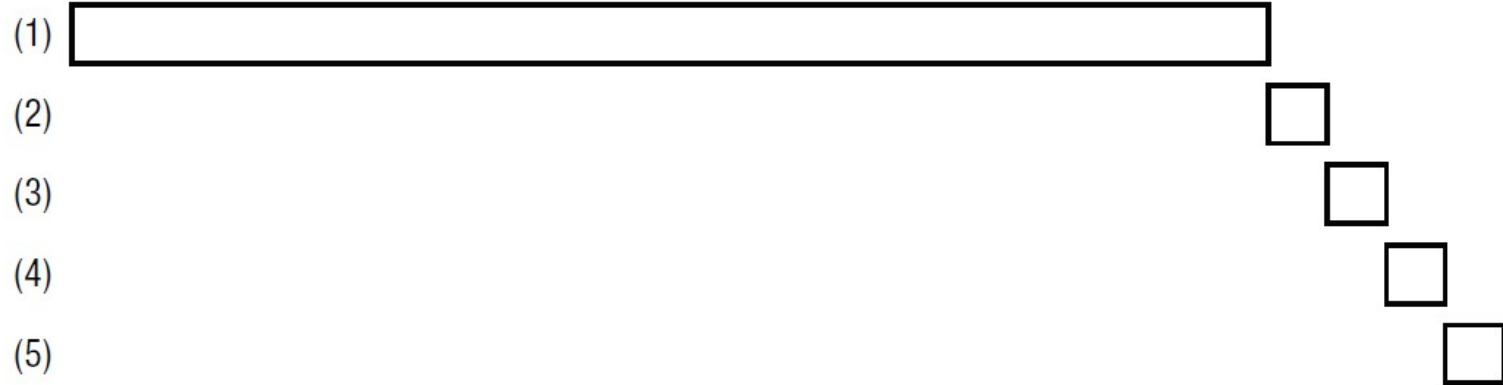
# Shortest Job First (SJF) Preemptive

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

# FIFO vs. SJF(preemptive)

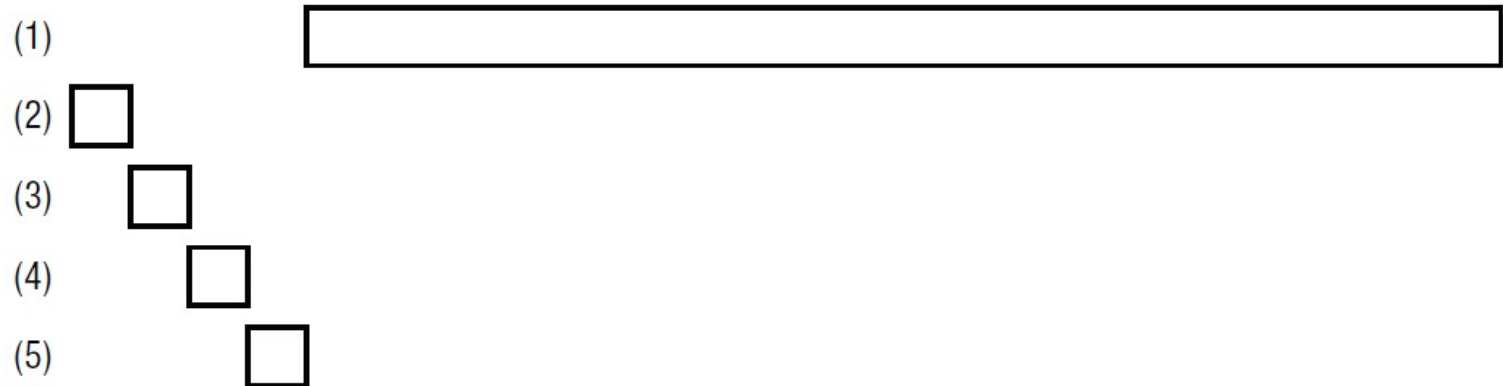
Tasks

FIFO



Tasks

SJF



Time



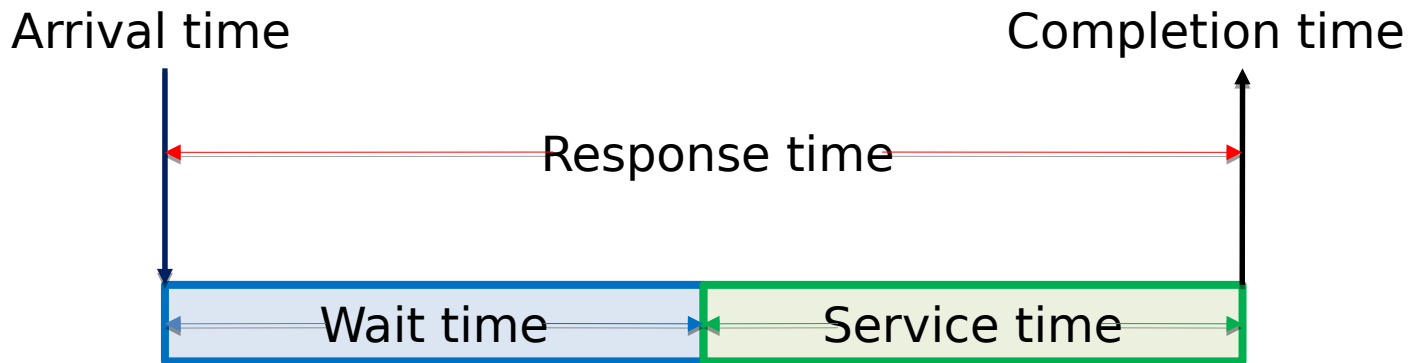
# Question

- Claim: SJF(preemptive) is optimal for average response time
  - Why?
- Does SJF(preemptive) have any downsides?

# Question

- Is FIFO ever optimal?
- Pessimal?

# Relationships Among 5 “Time” Values



Wait time (if any) precedes service time in FIFO

Wait time and service time can be intermixed in other policies

# Range of Classic Policies

- Policies that do not use service times:
  - FIFO – easiest to implement
  - RR
  - MFQ
- Future knowledge policies – decisions made based on knowledge of service times:
  - SJF(non-preemptive)
  - SJF(preemptive) – minimum avg. response time
  - Approximate SJF(preemptive) by predicting service times
    - E.g., based on running average of CPU burst lengths, file sizes

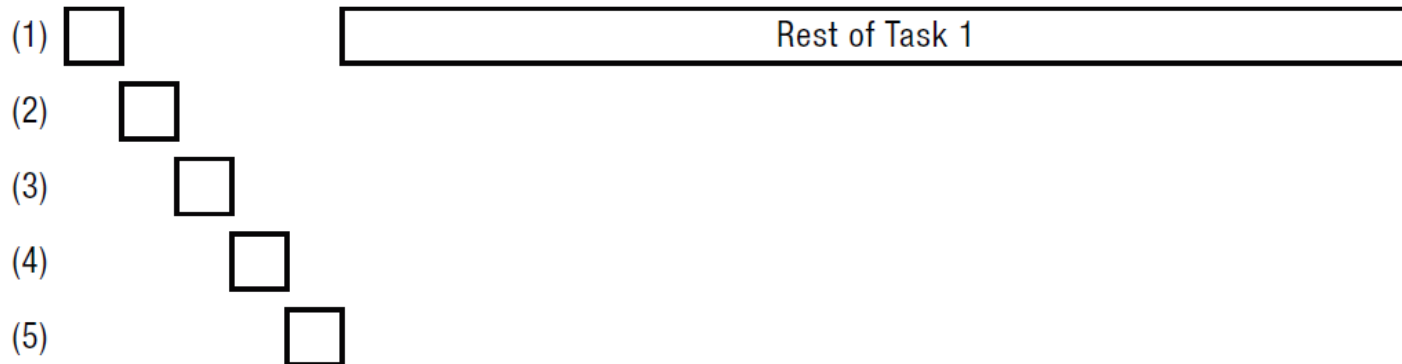
# Round Robin

- Each task gets resource for a fixed period of time = time quantum (or time slice)
  - If task doesn't complete, it goes back in line
- Need to pick a time quantum
  - What if time quantum is too long?
    - Infinite?
  - What if time quantum is too short?
    - One instruction?
  - Rule of thumb: 80%+ of tasks finish in one quantum

# Round Robin

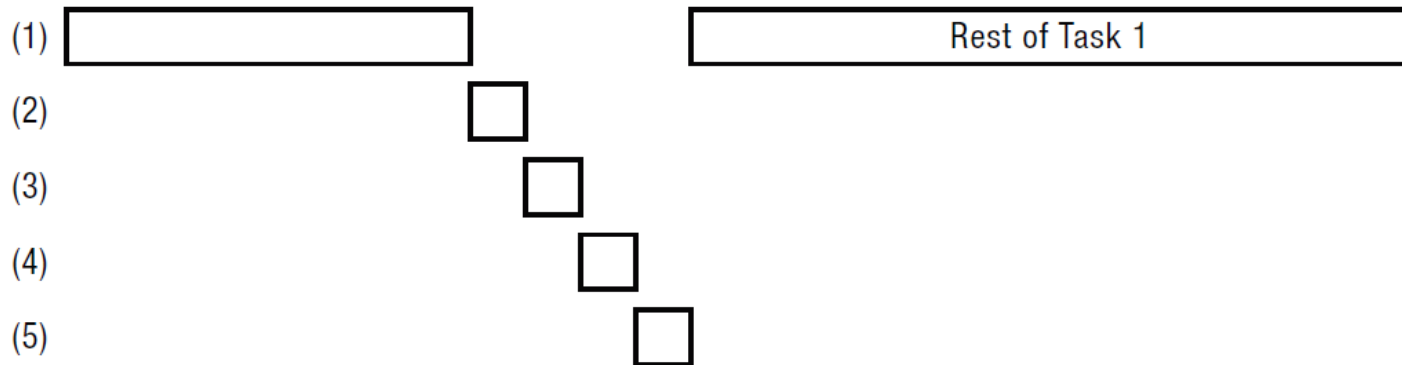
Tasks

Round Robin (1 ms time slice)



Tasks

Round Robin (100 ms time slice)

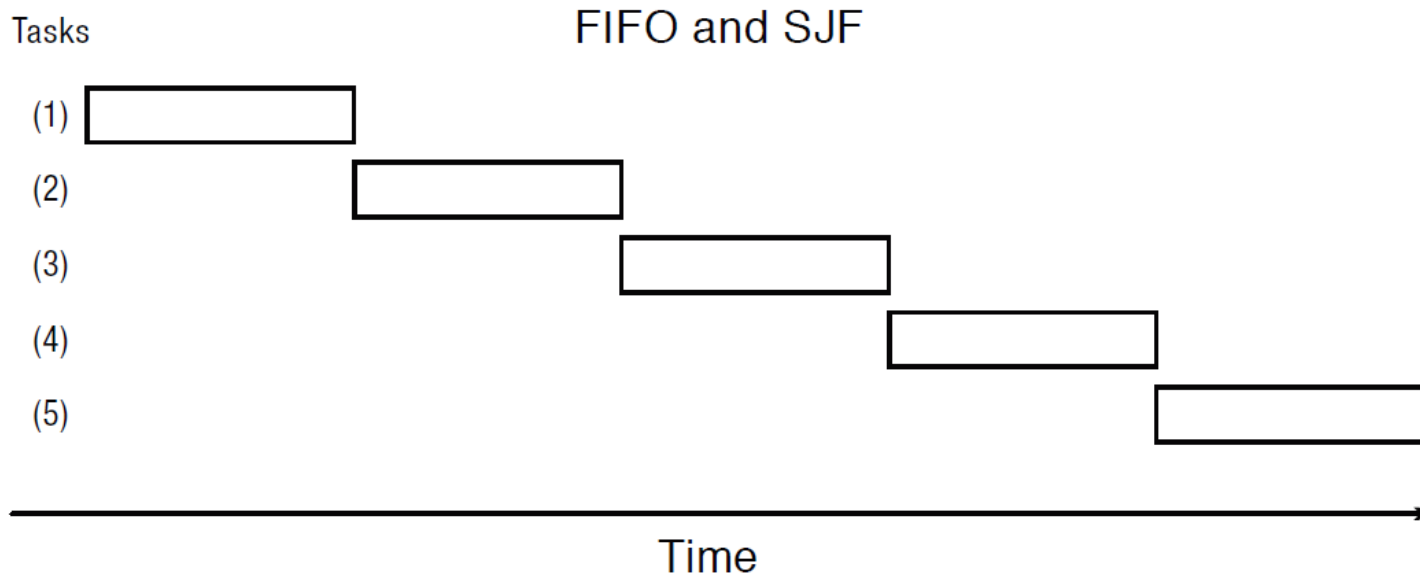
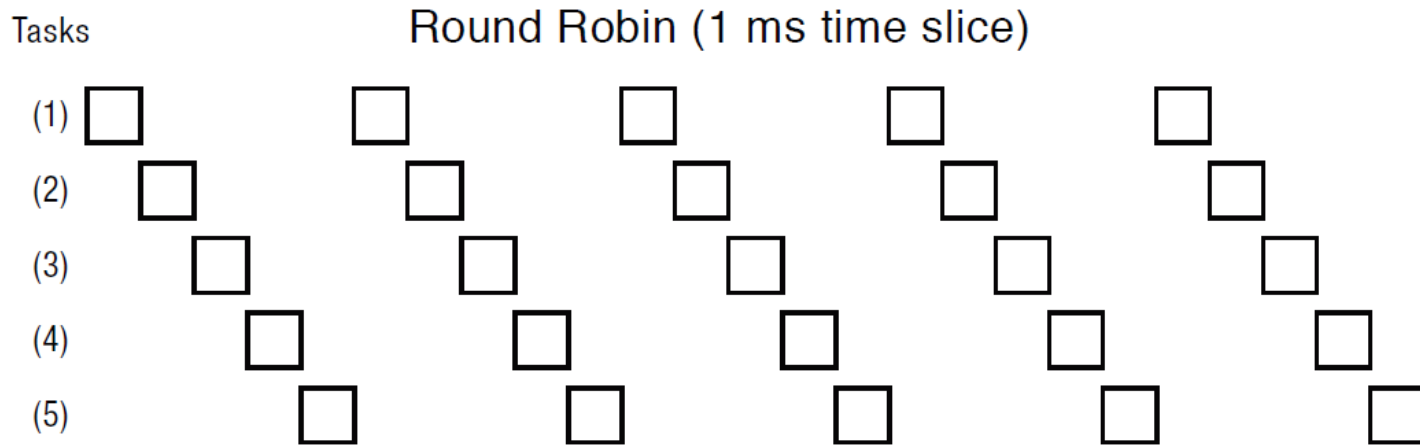


Time

# Round Robin vs. FIFO

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

# Round Robin vs. FIFO





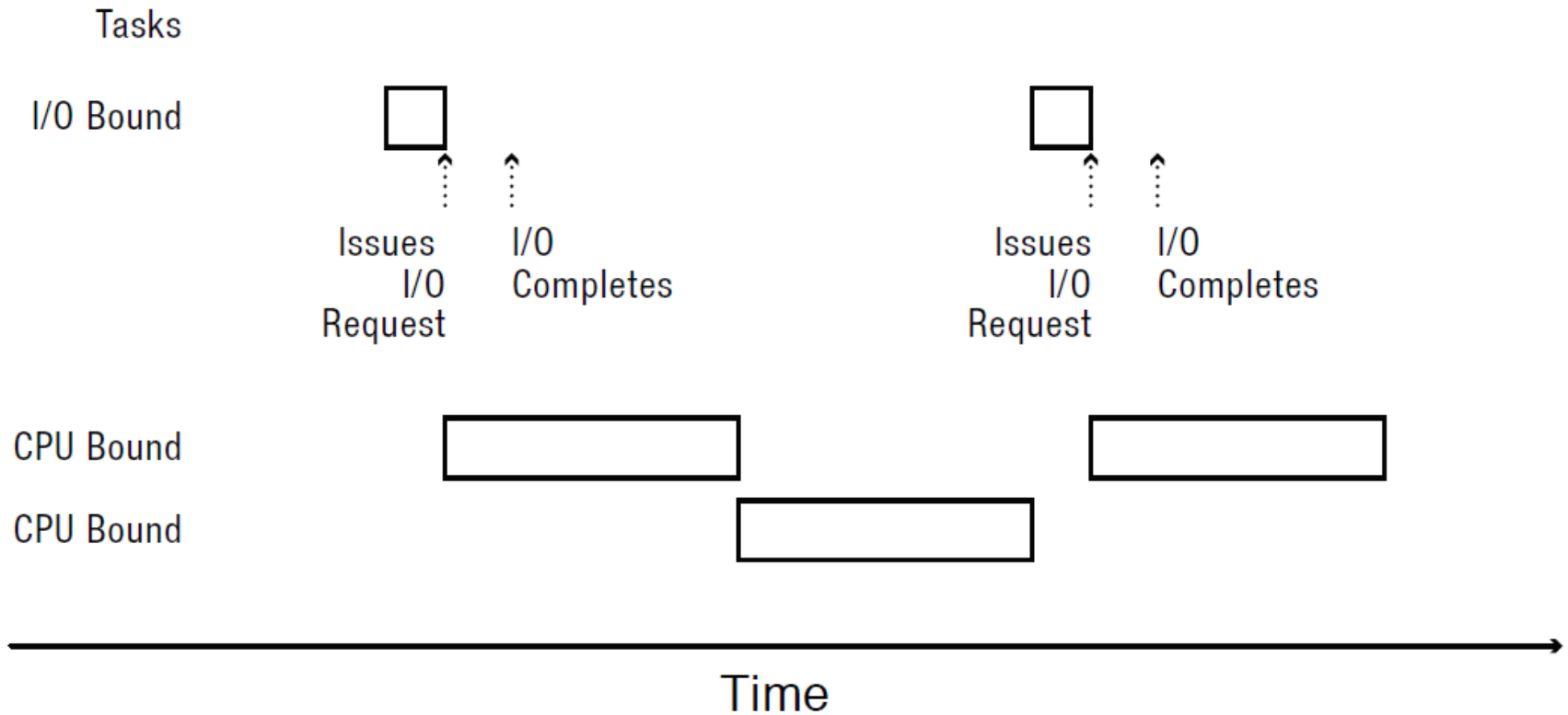
# Round Robin = Fairness?

- Is Round Robin always fair?
- 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

# Policy vs. Mechanism

- Policy = decision-making rule
  - “what to do”
- Mechanism = hardware or software that is used to implement a policy
  - “how to do it”
- What are mechanisms for Round Robin?

# Mixed Workload



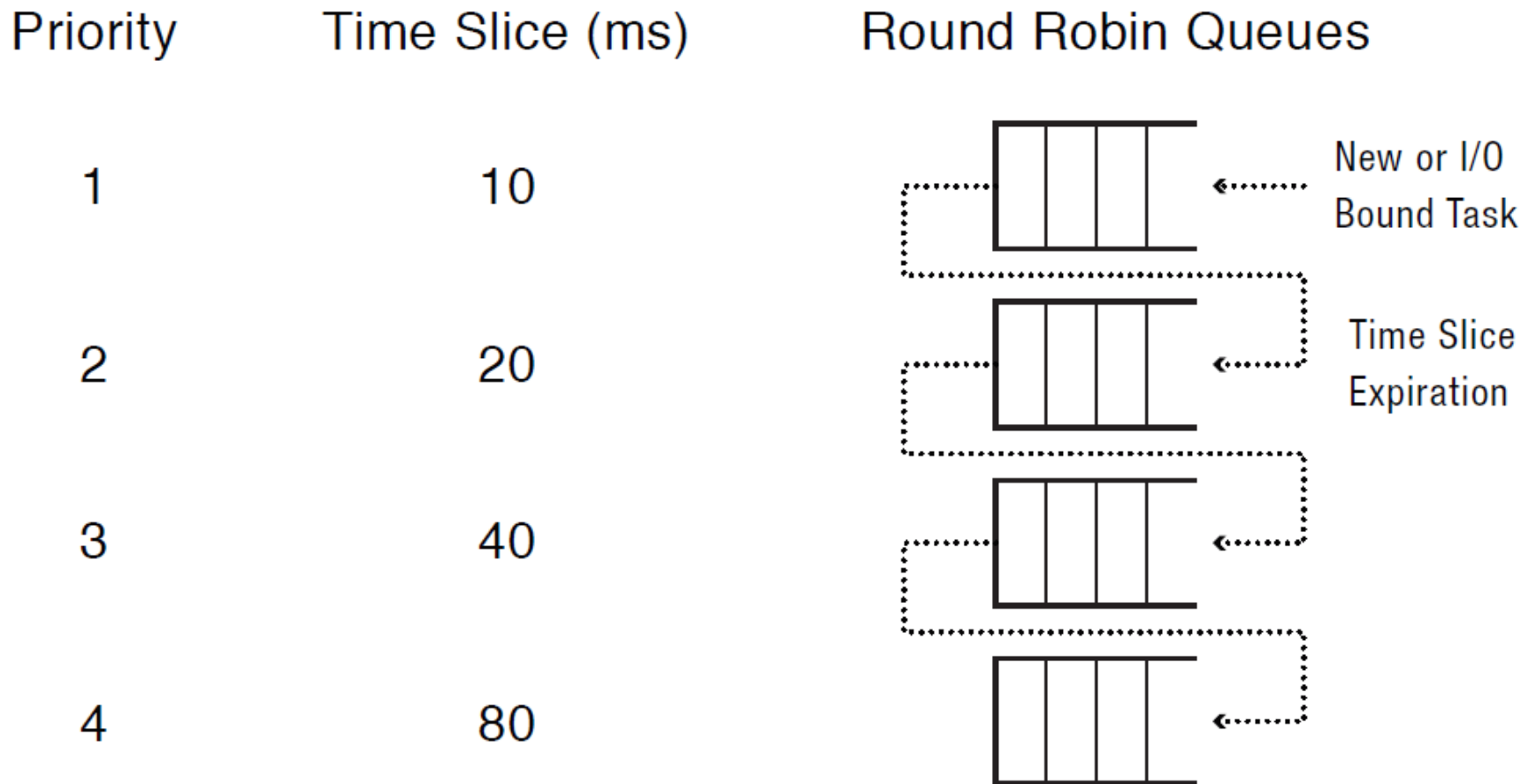
# Max-Min Fairness

- 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 (MFQ)

- 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

# MFQ



# Uniprocessor Summary (1)

- 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(preemptive) is optimal in terms of average response time
- SJF(preemptive) is pessimal in terms of variance in response time

# Uniprocessor Summary (2)

- 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(preemptive) and can do poorly under Round Robin



# Uniprocessor Summary (3)

- Max-Min fairness can improve response time for I/O-bound tasks
- Round Robin and Max-Min fairness both avoid starvation

# Simulation Comparisons

fcfs results:

1-th 10-percentile avg. wait is	19.800	*****
2-th 10-percentile avg. wait is	21.796	*****
3-th 10-percentile avg. wait is	19.908	*****
4-th 10-percentile avg. wait is	18.432	*****
5-th 10-percentile avg. wait is	19.444	*****
6-th 10-percentile avg. wait is	19.618	*****
7-th 10-percentile avg. wait is	19.984	*****
8-th 10-percentile avg. wait is	21.264	*****
9-th 10-percentile avg. wait is	21.188	*****
10-th 10-percentile avg. wait is	18.490	*****

+-----  
scaled to max value  
overall avg.

19.992

rr results:

1-th 10-percentile avg. wait is	4.932	**
2-th 10-percentile avg. wait is	5.164	**
3-th 10-percentile avg. wait is	5.142	**
4-th 10-percentile avg. wait is	9.858	***
5-th 10-percentile avg. wait is	9.696	***
6-th 10-percentile avg. wait is	15.340	*****
7-th 10-percentile avg. wait is	19.858	*****
8-th 10-percentile avg. wait is	26.128	*****
9-th 10-percentile avg. wait is	35.976	*****
10-th 10-percentile avg. wait is	61.794	*****

+-----  
scaled to max value  
overall avg.

19.389

mlfq results:

1-th 10-percentile avg. wait is	2.690	*
2-th 10-percentile avg. wait is	2.808	*
3-th 10-percentile avg. wait is	3.262	*
4-th 10-percentile avg. wait is	7.084	***
5-th 10-percentile avg. wait is	6.562	**
6-th 10-percentile avg. wait is	6.566	**
7-th 10-percentile avg. wait is	24.858	*****
8-th 10-percentile avg. wait is	33.758	*****
9-th 10-percentile avg. wait is	34.030	*****
10-th 10-percentile avg. wait is	68.922	*****

+-----  
scaled to max value  
overall avg.

19.054

srtn results:

1-th 10-percentile avg. wait is	0.058	
2-th 10-percentile avg. wait is	0.170	
3-th 10-percentile avg. wait is	0.470	
4-th 10-percentile avg. wait is	0.874	
5-th 10-percentile avg. wait is	1.234	*
6-th 10-percentile avg. wait is	2.286	*
7-th 10-percentile avg. wait is	4.222	**
8-th 10-percentile avg. wait is	7.232	***
9-th 10-percentile avg. wait is	14.148	*****
10-th 10-percentile avg. wait is	47.980	*****

+-----  
scaled to max value  
overall avg.

7.867