CPU SCHEDULING

Shivaram Venkataraman
CS 537, Spring 2019

HI!

OK?

ADMINISTRIVIA

- Project 1a is due ~~today!~~ Thursday at 11.59pm
- No office hours from 5pm Tue to noon Thu
- Fill out office hours form? https://goo.gl/forms/5VxrwRawtEFkrjO23
- No more waitlist!
- Project 1b out tomorrow. Schedule updates

AGENDA / LEARNING OUTCOMES

Scheduling

- How does the OS decide what process to run?
- What are some of the metrics to optimize for?

Policies

- How to handle interactive and batch processes?
- What to do when OS doesn't have complete information?

RECAP

RECAP: SCHEDULING MECHANISM

Process: Abstraction to virtualize CPU

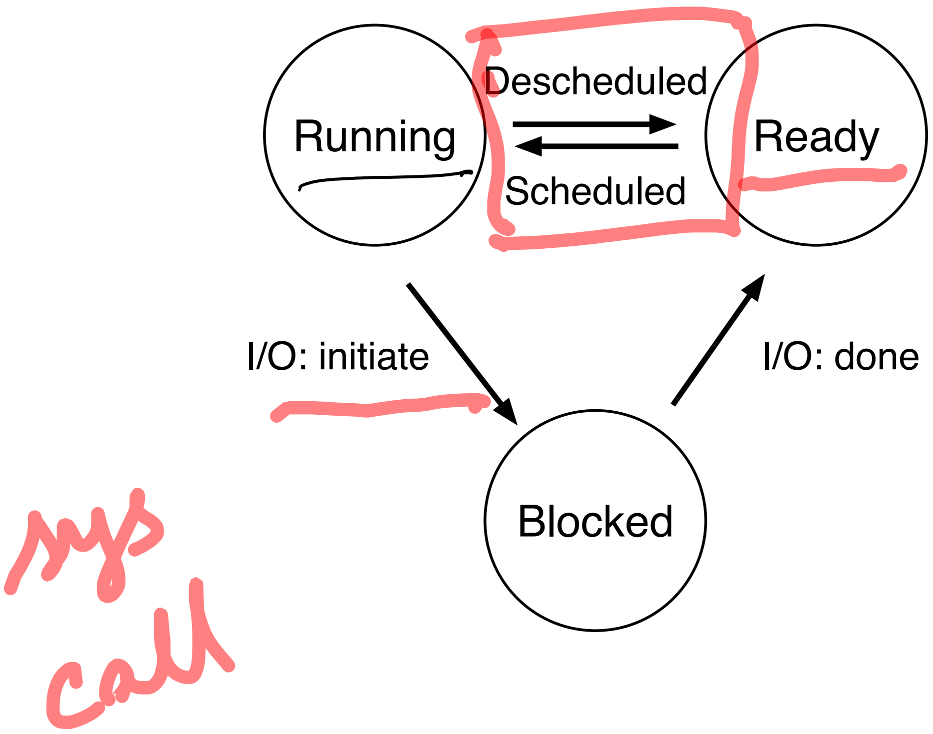
Address space
File Desc.

Use time-sharing in OS to switch between processes

P1 → Pause

CPU ← P2

PROCESS STATE TRANSITIONS



RECAP: SCHEDULING MECHANISM

Limited Direct Execution

Use system calls to run access devices etc. from user mode

Context-switch using interrupts for multi-tasking

Operating System

Handle the trap
Call switch() routine
  save kernel regs(A) to proc-struct(A)
  restore kernel regs(B) from proc-struct(B)
  switch to k-stack(B)
return-from-trap (into B)

Hardware

timer interrupt
save regs(A) to k-stack(A)
move to kernel mode
jump to trap handler

user stack pointer

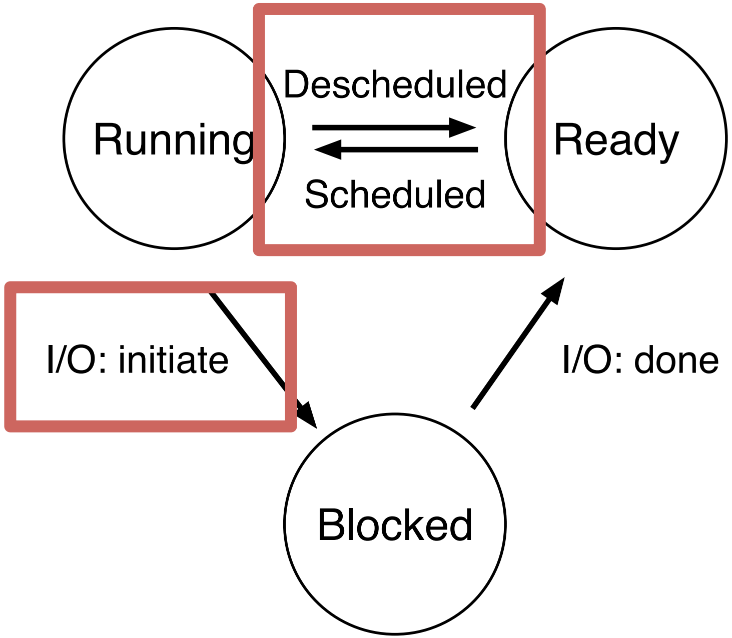
restore regs(B) from k-stack(B)
move to user mode
jump to B's IP

Program

Process A

Cat
my-file

Process B


Running
Descheduled
Scheduled
Ready
I/O: initiate
I/O: done
Blocked

POLICY ?

VOCABULARY

Workload: set of **jobs** (arrival time, run_time)

Job ~ Current execution of a process

Alternates between CPU and I/O

Moves between ready and blocked queues

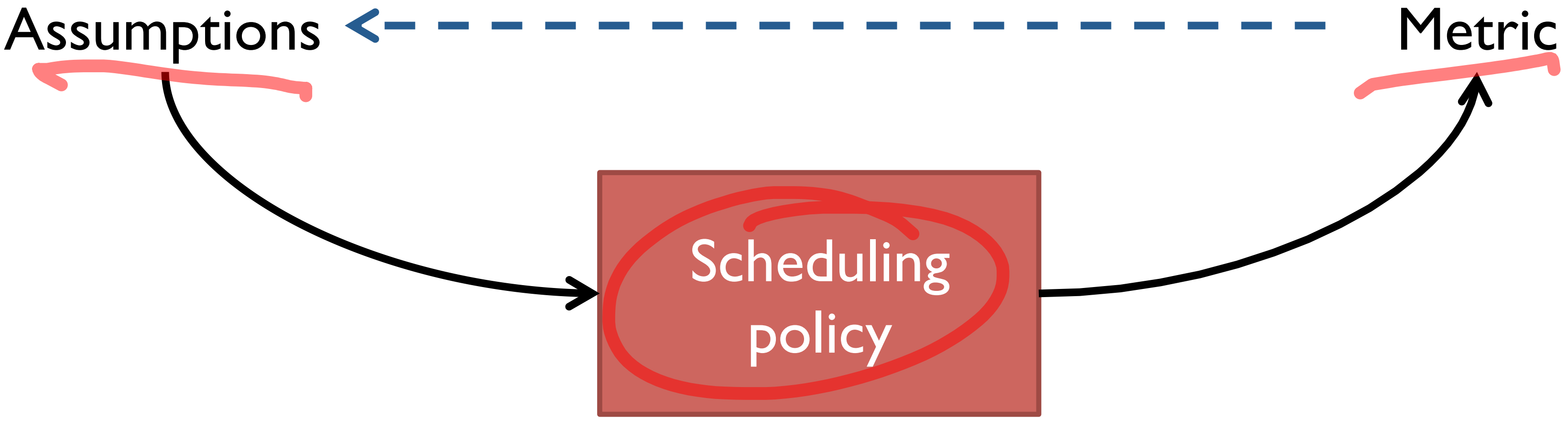
Scheduler: Decides which ready job to run

Metric: measurement of scheduling quality

J1 t=0 r=10s

J2 t=10 r=20s

APPROACH



ASSUMPTIONS

1. Each job runs for the same amount of time
2. All jobs arrive at the same time
3. All jobs only use the CPU (no I/O)
4. Run-time of each job is known

METRIC 1: TURNAROUND TIME

Turnaround time = *completion_time - arrival_time*

Example:

Process A arrives at time t = 10, finishes t = 30

Process B arrives at time t = 10, finishes t = 50

Turnaround time

A = 20, B = 40

Average = 30

minimize avg Turnaround time!

FIFO / FCFS

FIFO / FCFS

FIFO: First In, First Out

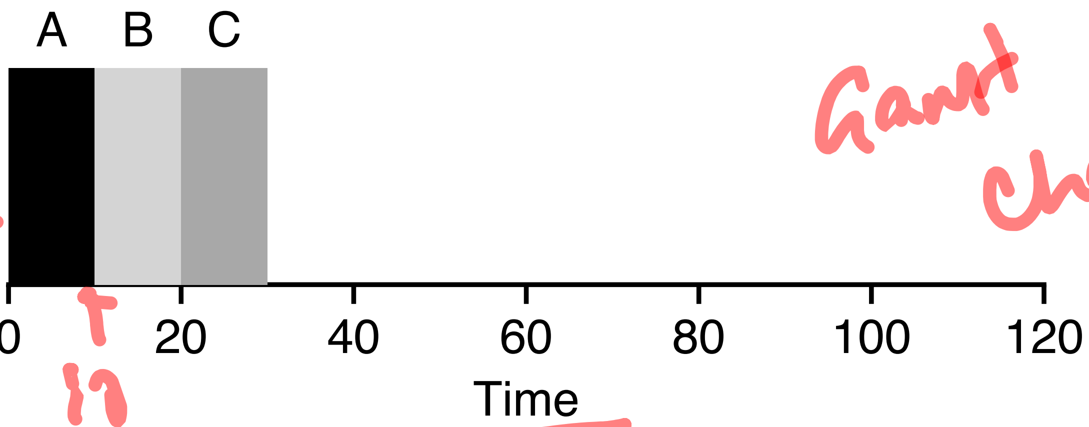
FCFS: First Come, First Served

Job	Arrival(s)	run time (s)
A	~0	10
B	~0	10
C	~0	10

FIFO / FCFS

Job	Arrival(s)	run time (s)
A	~0	10
B	~0	10
C	~0	10

Process



10

Gantt chart

Average Turnaround Time ?

10, 20, 30

= 20 s

ASSUMPTIONS

1. ~~Each job runs for the same amount of time~~
2. All jobs arrive at the same time
3. All jobs only use the CPU (no I/O)
4. Run-time of each job is known

2-MINUTE QUIZ

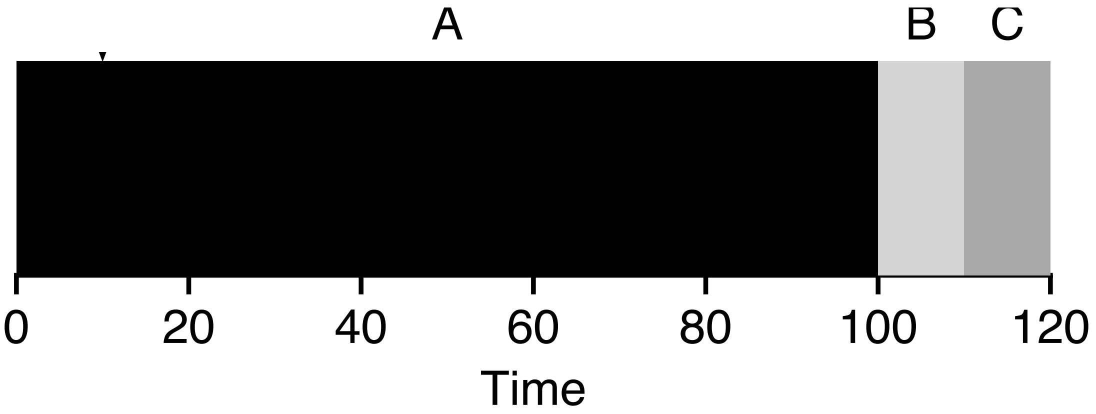
How will FIFO perform without this assumption ?

What scenarios can lead to bad performance?

Avg Turnaround time

BIG FIRST JOB

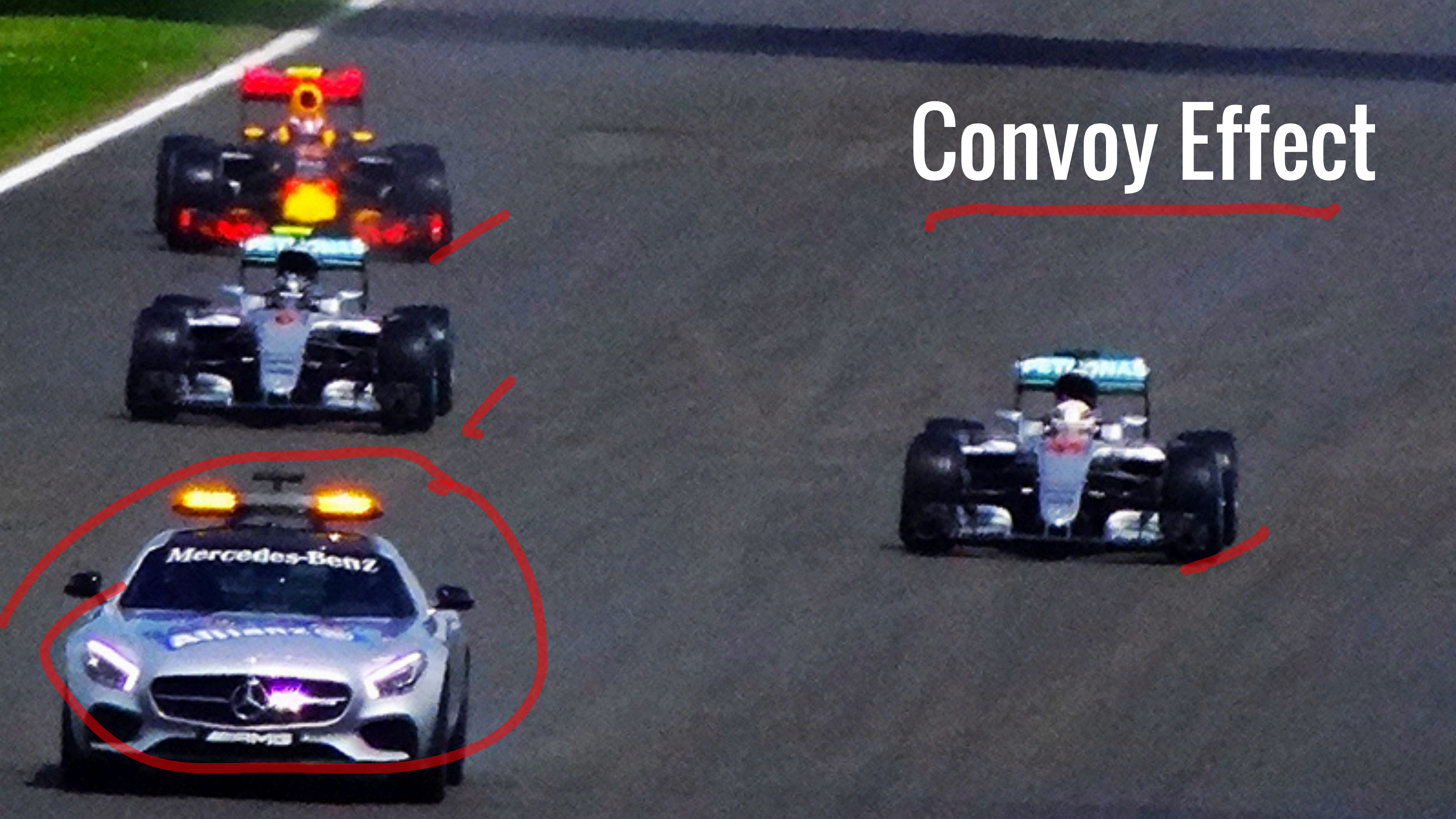
Job	Arrival(s)	run time (s)
A	~0	100
B	~0	10
C	~0	10



Average Turnaround Time

(100 + 110 + 120)/ 3
= 110s

Convoy Effect



CHALLENGE

Turnaround time suffers when short jobs must wait for long jobs

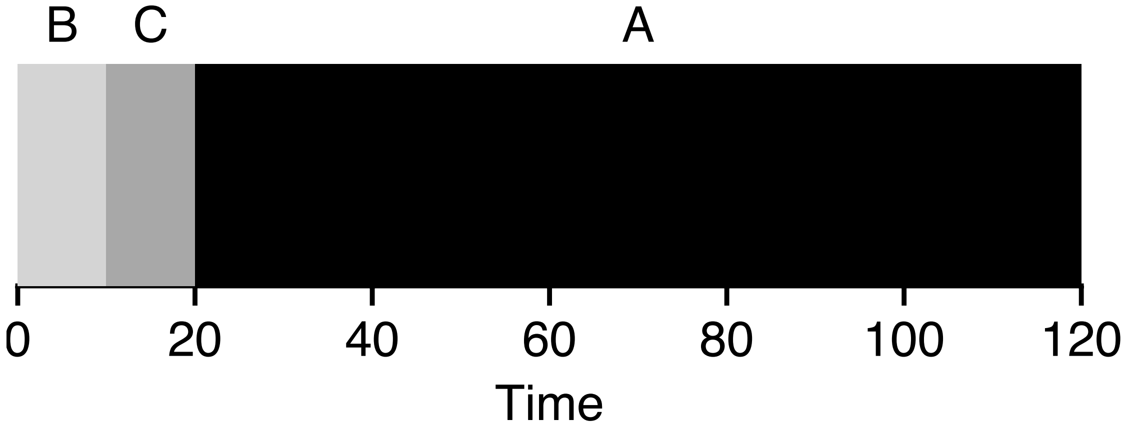
New scheduler:

SJF (Shortest Job First)

Choose job with smallest run_time!

SHORTEST JOB FIRST (SJF)

Job	Arrival(s)	run time (s)
A	~0	100
B	~0	10
C	~0	10



Average Turnaround Time

(10 + 20 + 120)/ 3
= 50s!

FIFO: 110s ?!

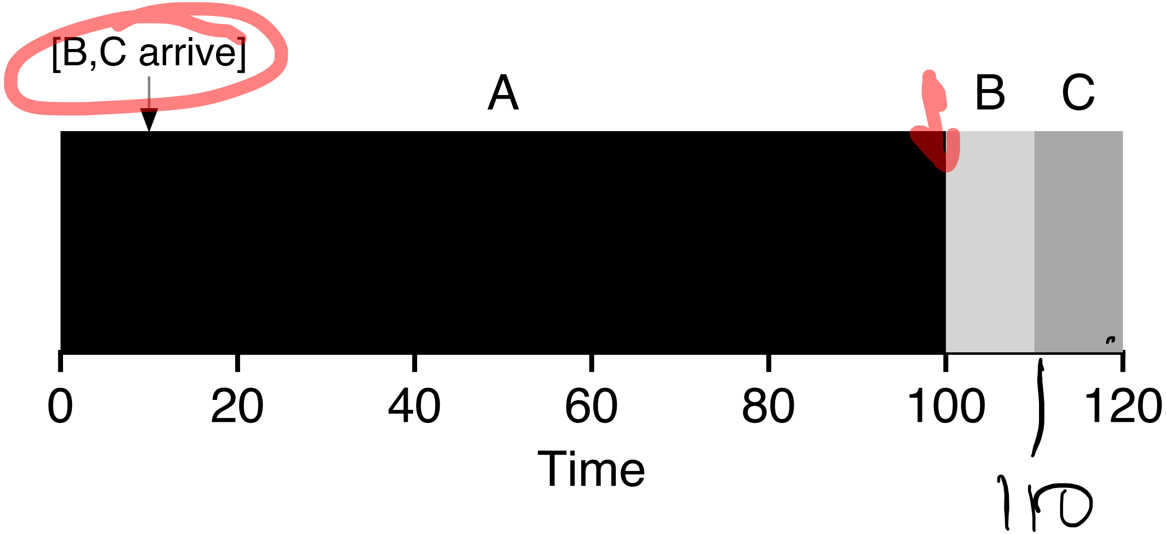
ASSUMPTIONS

1. ~~Each job runs for the same amount of time~~
2. ~~All jobs arrive at the same time~~
3. All jobs only use the CPU (no I/O)
4. Run-time of each job is known

Job	Arrival(s)	run time (s)
A	~0	100
B	10	10
C	10	10

Average Turnaround Time with SJF?

Job	Arrival(s)	run time (s)
A	~0	100
B	10	10
C	10	10



Average Turnaround Time ?

(100 + 110 + ~~120~~ 100)/ 3

=

103.33

100 + 110 + 100

PREEMPTIVE SCHEDULING

Prev schedulers:

FIFO and SJF are non-preemptive

Only schedule new job when previous job voluntarily relinquishes CPU

New scheduler:

Preemptive: Schedule different job by taking CPU away from running job

STCF (Shortest Time-to-Completion First)

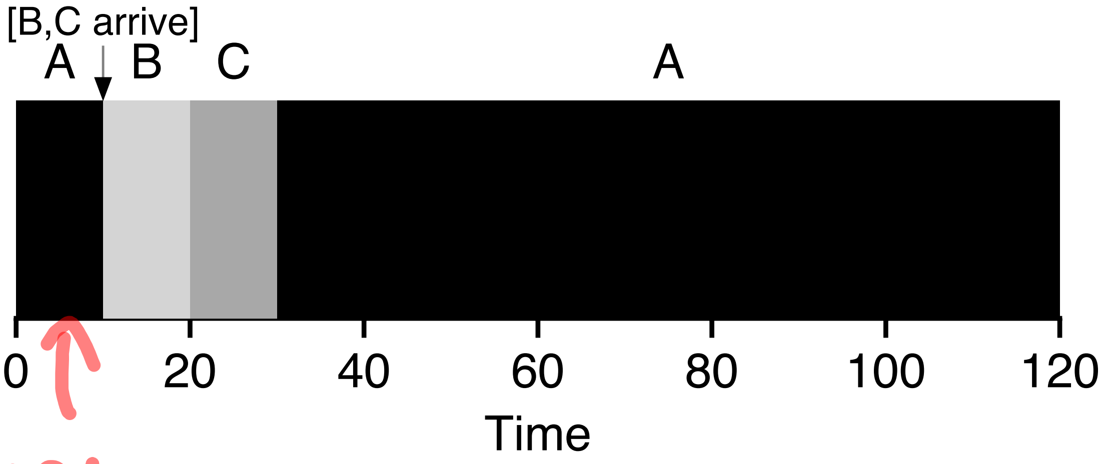
Always run job that will complete the quickest

PREMPTIVE SCTF

Job	Arrival(s)	run time (s)
A	~0	100
B	10	10
C	10	10

90 90
10 0
10 10

Average Turnaround Time



100

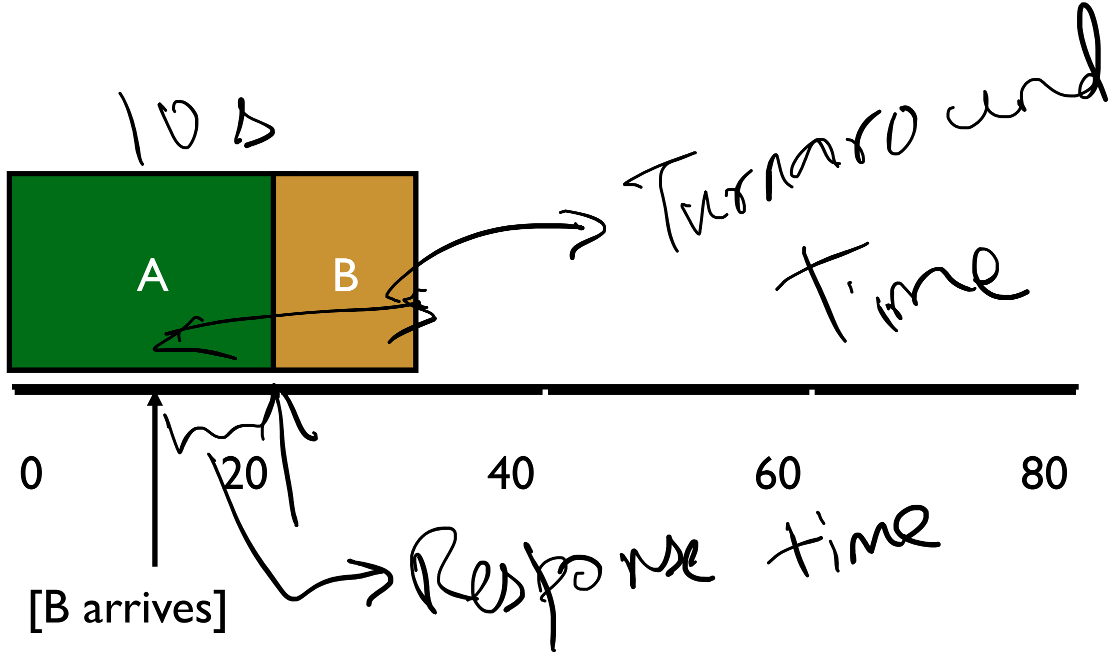
(10 + 20 + 120)/ 3
= 50s

METRIC 2: RESPONSE TIME

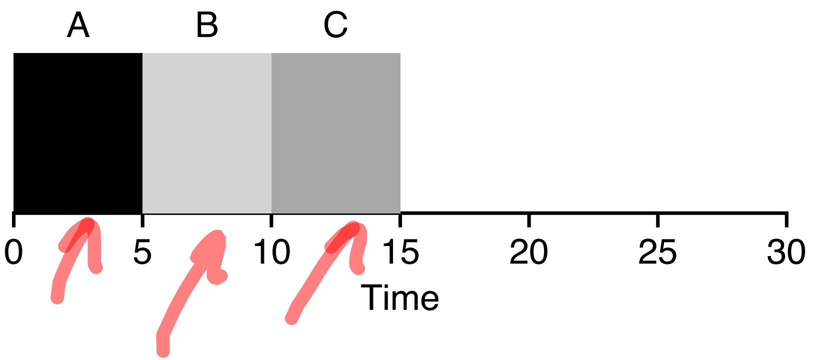
Response time = *first_run_time - arrival_time*

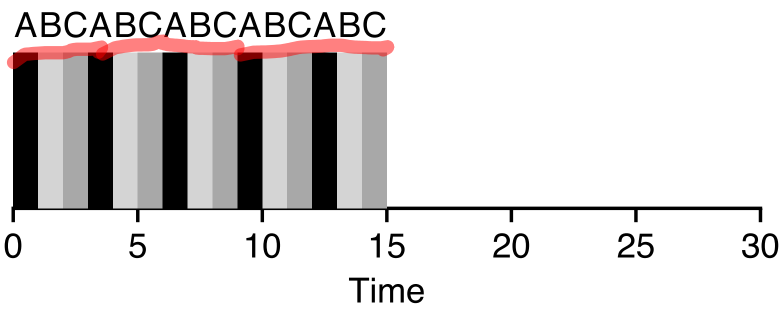
B's turnaround: 20s

B's response: 10s



ROUND ROBIN SCHEDULER



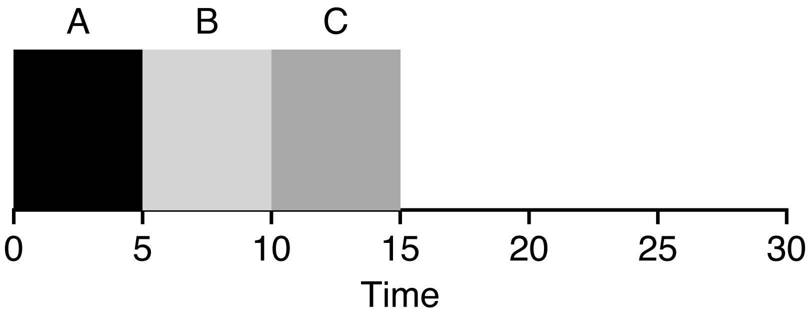


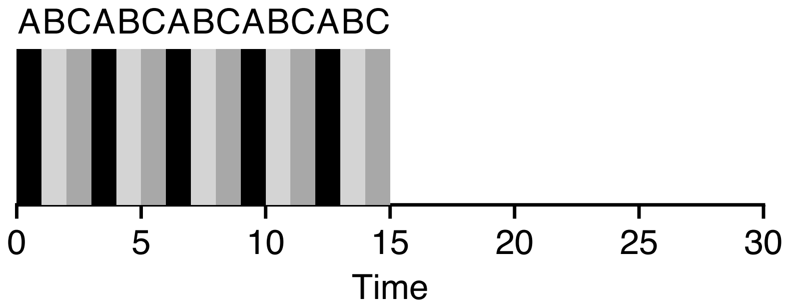
Average Response Time

(0 + 5 + 10)/3 = 5s

(0 + 1 + 2)/3 = 1s

2-MINUTE QUIZ





Avg 10s ?

14s

Turn around time

What is the turnaround time for two cases ?

Is round robin better or worse?

TRADE-OFFS

Round robin increases turnaround time decreases response time

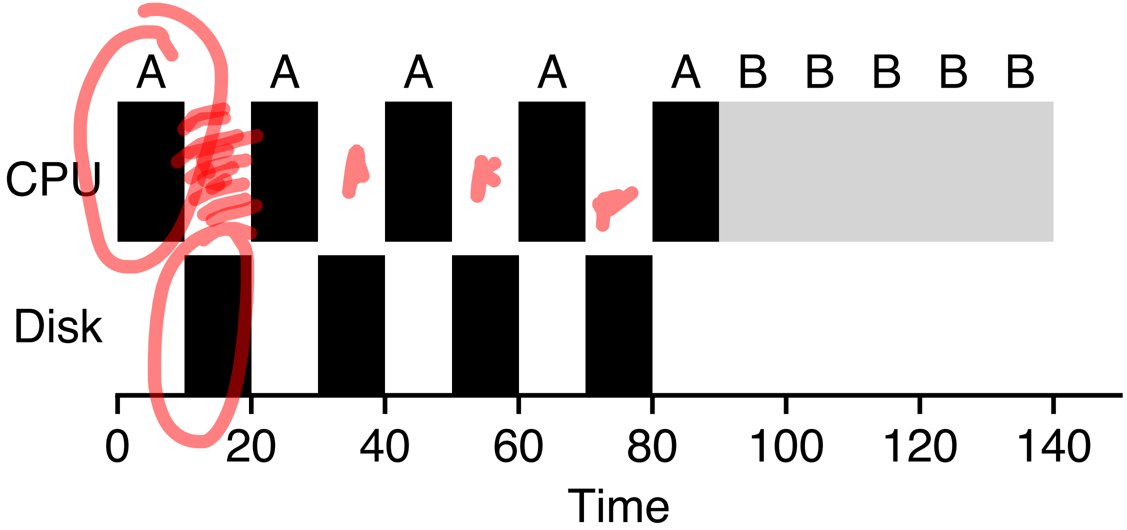
Tuning challenges:

- What is a good time slice for round robin?
- What is the overhead of context switching?

ASSUMPTIONS

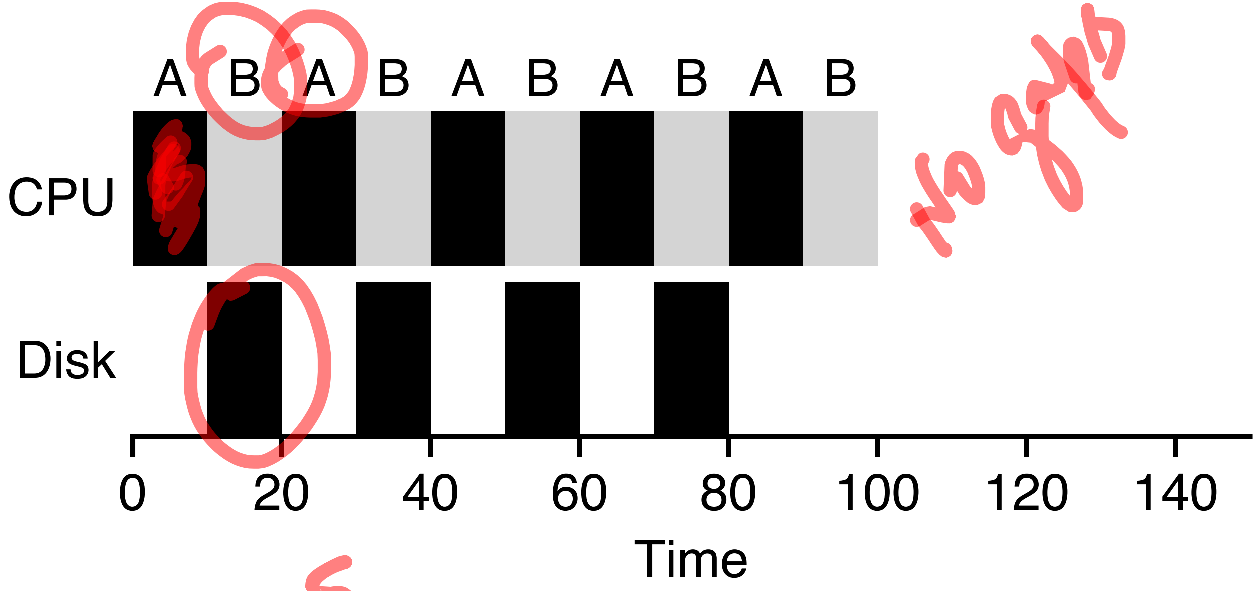
1. ~~Each job runs for the same amount of time~~
2. ~~All jobs arrive at the same time~~
3. ~~All jobs only use the CPU (no I/O)~~
4. Run-time of each job is known

NOT IO AWARE



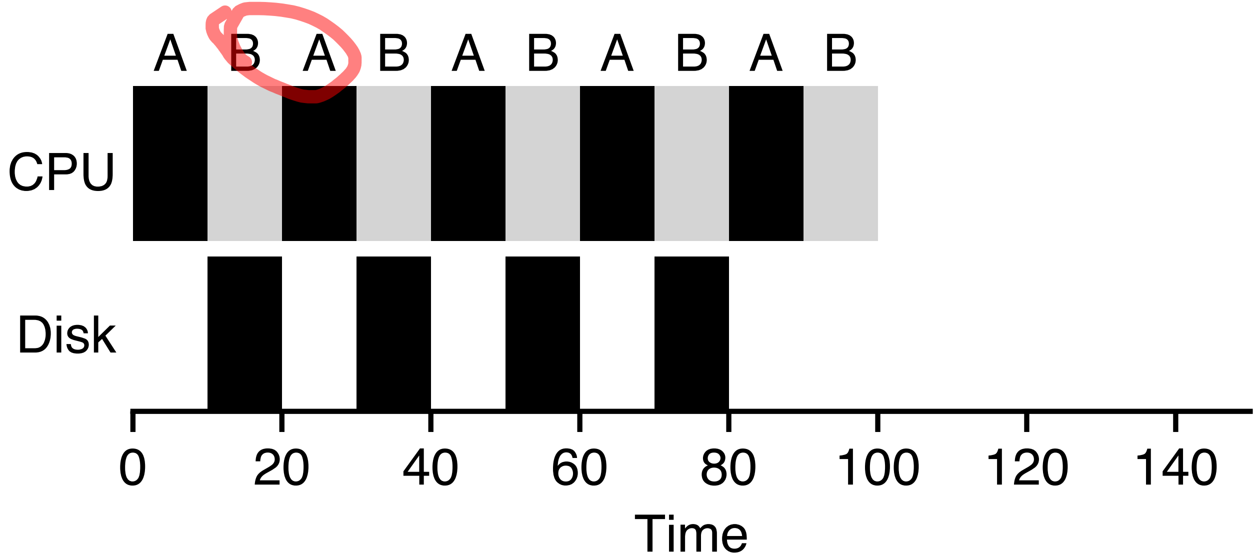
Job holds on to CPU while blocked on disk!

I/O AWARE SCHEDULING



Treat Job A as ~~3~~ 5 separate CPU bursts.
When Job A completes I/O, another Job A is ready

I/O AWARE SCHEDULING



Each CPU burst is shorter than Job B

With SCTF,
Job A preempts Job B

Treat Job A as 3 separate CPU bursts.
When Job A completes I/O, another Job A is ready

ASSUMPTIONS

1. ~~Each job runs for the same amount of time~~
2. ~~All jobs arrive at the same time~~
3. ~~All jobs only use the CPU (no I/O)~~
4. ~~Run-time of each job is known~~

MULTI-LEVEL FEEDBACK QUEUE

MLFQ

MLFQ: GENERAL PURPOSE SCHEDULER

Must support two job types with distinct goals

- "interactive" programs care about response time
- "batch" programs care about turnaround time

Keyboard

zip

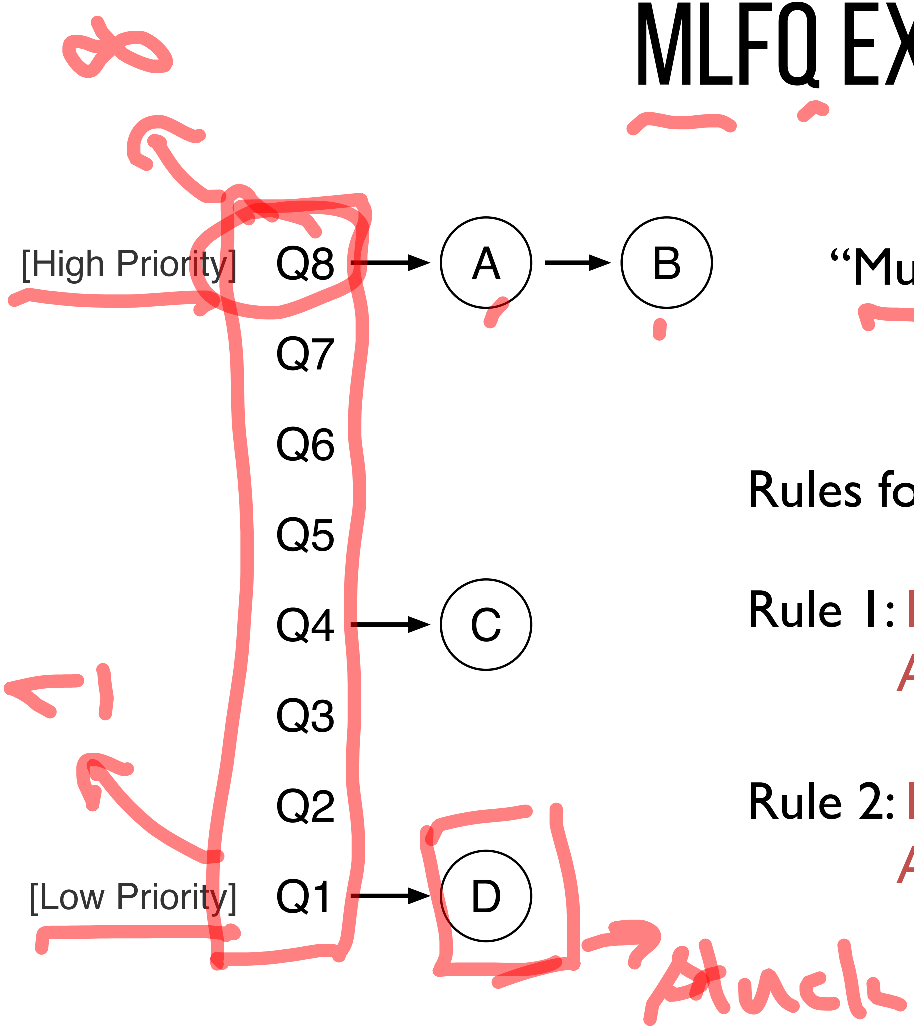
Approach:

Multiple levels of round-robin

Each level has higher priority than lower level

Can preempt them

MLFQ EXAMPLE



"Multi-level" – Each level is a queue!

Rules for MLFQ

Rule 1: If priority(A) > Priority(B)
A runs

Rule 2: If priority(A) == Priority(B),
A & B run in RR

CHALLENGES

How to set priority?

What do we do when a new process arrives?

Does a process stay in one queue or move between queues?

Approach: Use past behavior of process to predict future!

Guess how CPU burst (job) will behave based on past CPU bursts

MORE MLFQ RULES

Rule 1: If priority(A) > Priority(B), A runs

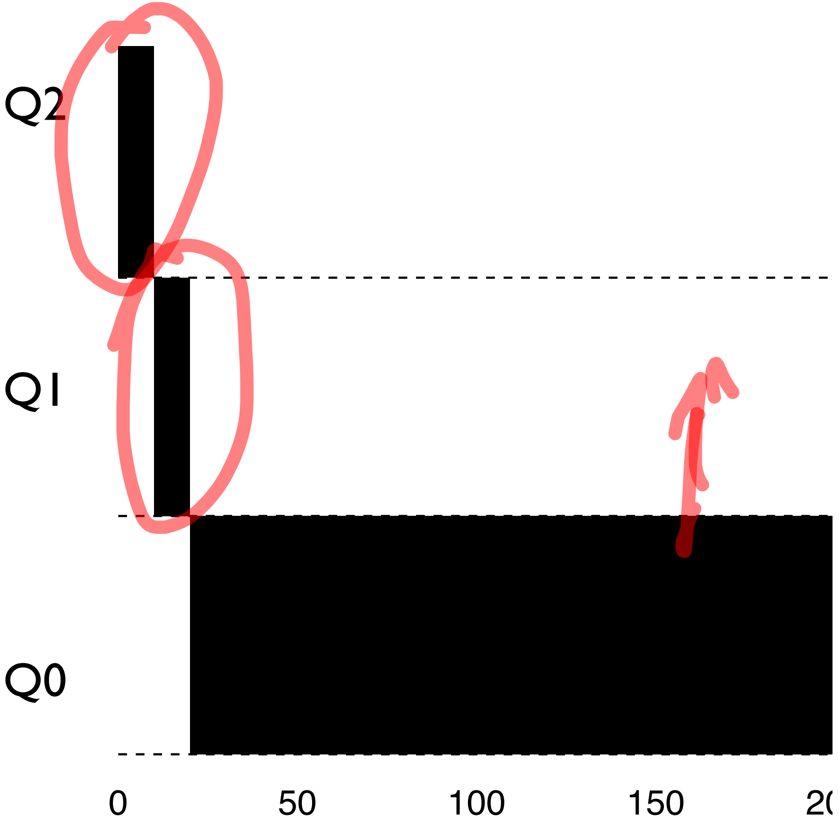
Rule 2: If priority(A) == Priority(B), A & B run in RR

Rule 3: Processes start at top priority

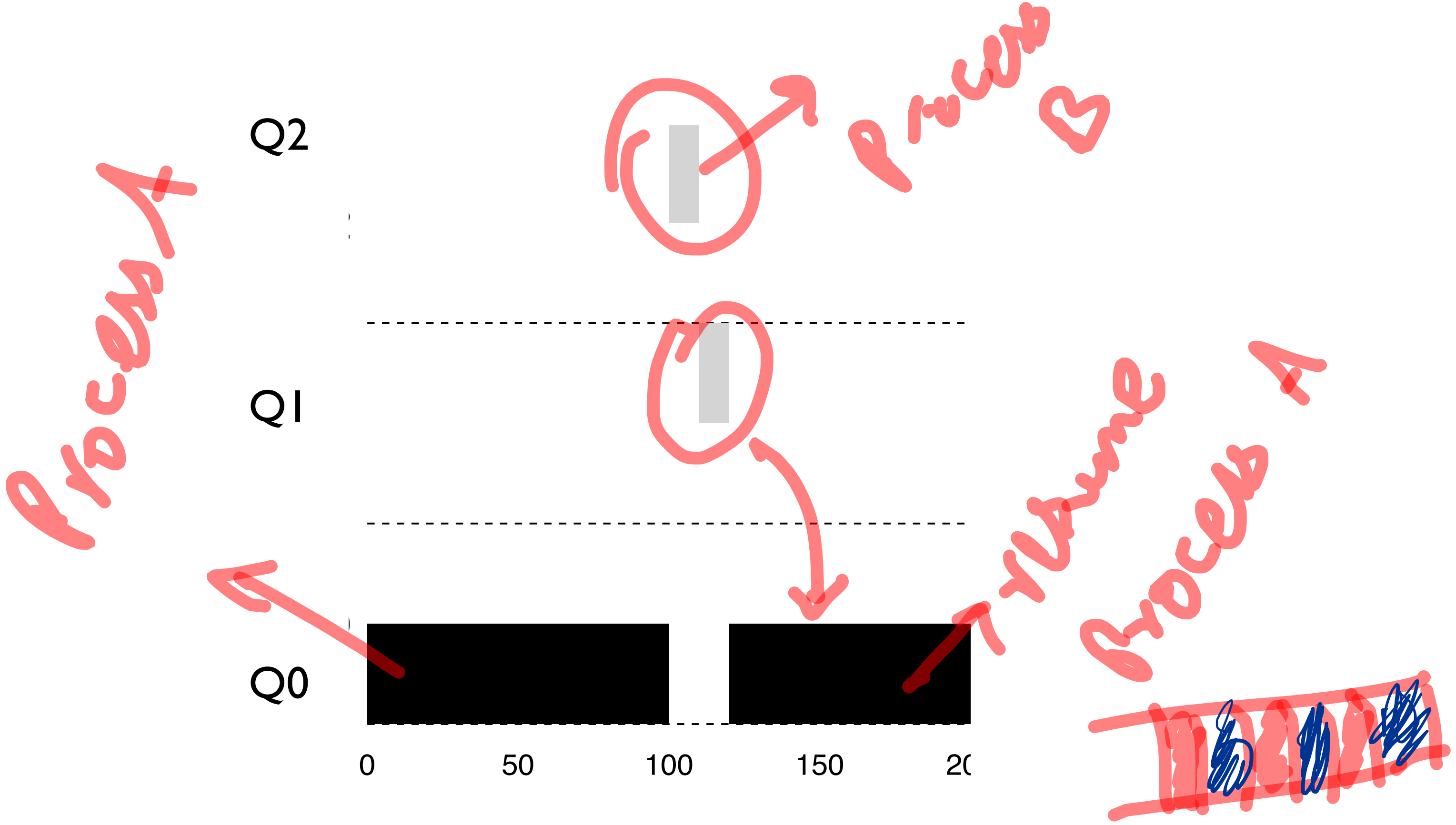
Rule 4: If job uses whole slice, demote process

(longer time slices at lower priorities)

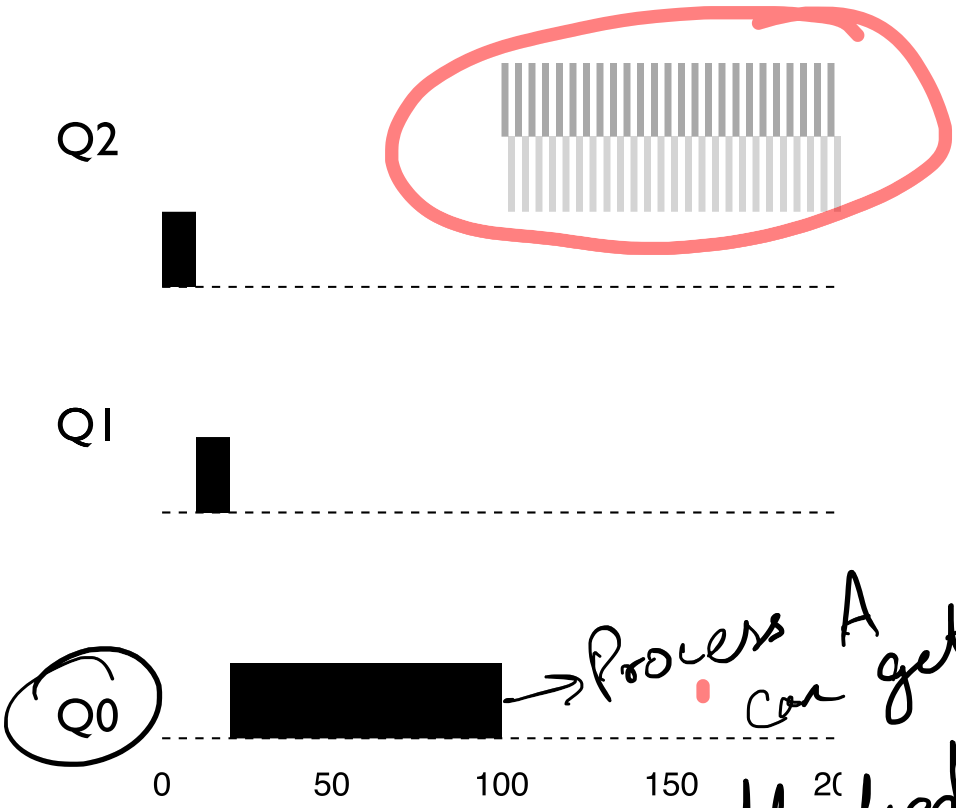
ONE LONG JOB



INTERACTIVE PROCESS JOINS



MLFQ PROBLEMS?

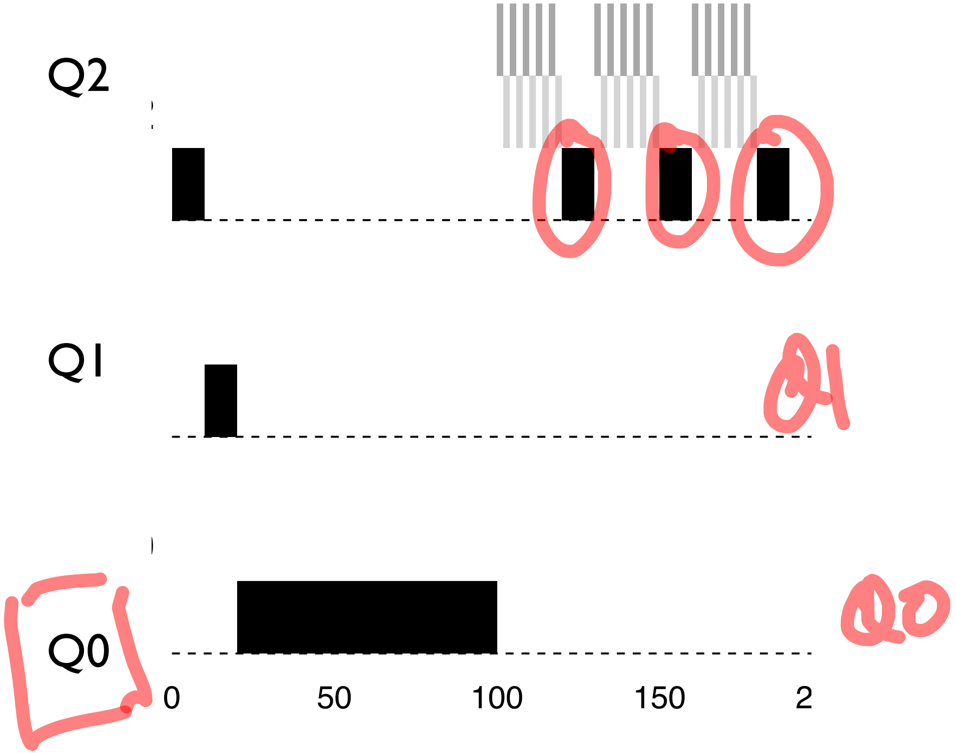


Process A can get blocked

What is the problem with this schedule ?

Starvation

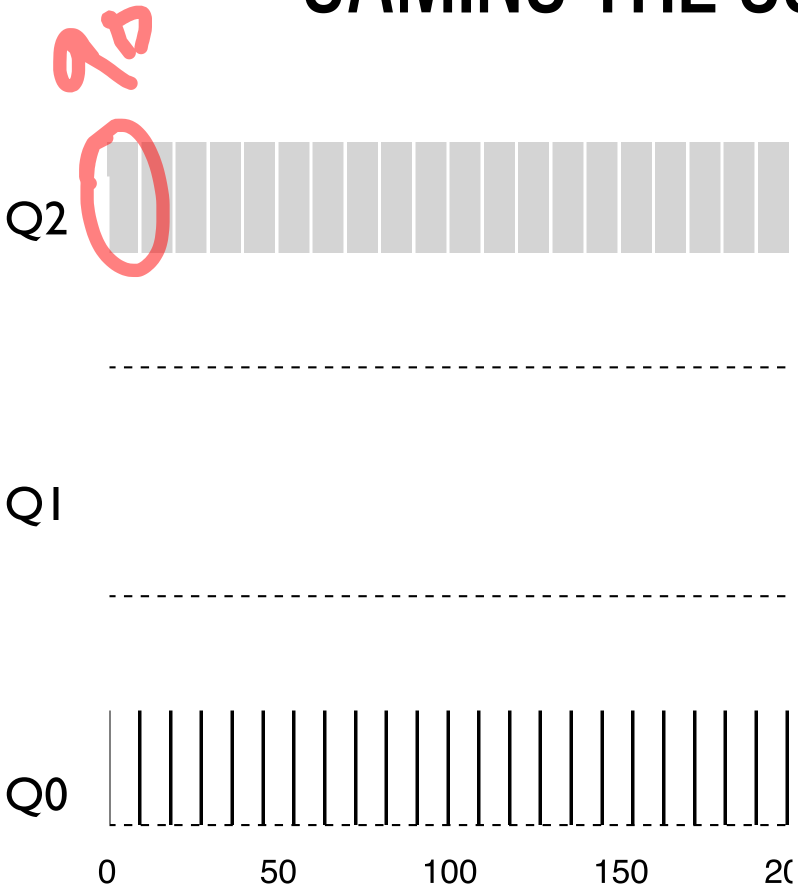
AVOIDING STARVATION



Problem: Low priority job may never get scheduled

Periodically **boost** priority of all jobs (or all jobs that haven't been scheduled)

GAMING THE SCHEDULER ?



95 10

timer interrupt

time slice 10s

Job could trick scheduler by doing I/O just before time-slice end

Account for total run time at priority
Downgrade when exceed threshold

SUMMARY

Scheduling Policies

Understand workload characteristics like arrival, CPU, I/O

Scope out goals, metrics (turnaround time, response time)

Approach

Trade-offs based on goals, metrics (RR vs. SCTF)

Past behavior is good predictor of future behavior?

MLFQ

NEXT STEPS

Project 1a: Due Jan 31 (Thursday) at 11.59pm

Project 1b: Out on Jan 30th

Thursday class, discussion

- More advanced scheduling policies
- Summary / review of process, CPU scheduling
- xv6 introduction, walk through
- Go through xv6 context switch / syscall?