

Processes & Scheduling

Operating Systems

SAELEE:CS450-S12,#3

policies

& mechanisms

(policies **vs.** mechanisms)

allocate **resources**

CPU

Memory

I/O devices

I/O bus

to **entities**

Processes

Threads

Users

Server requests, etc.

concurrency required
(otherwise, much easier)

concurrency requires
tracking of **process state**

user-level:
code & data

kernel level:
system state & accounting

system state

PC/SP/FP + other registers,
memory mgmt tables,
open files, etc.

system state typically
updated on context switches

accounting info

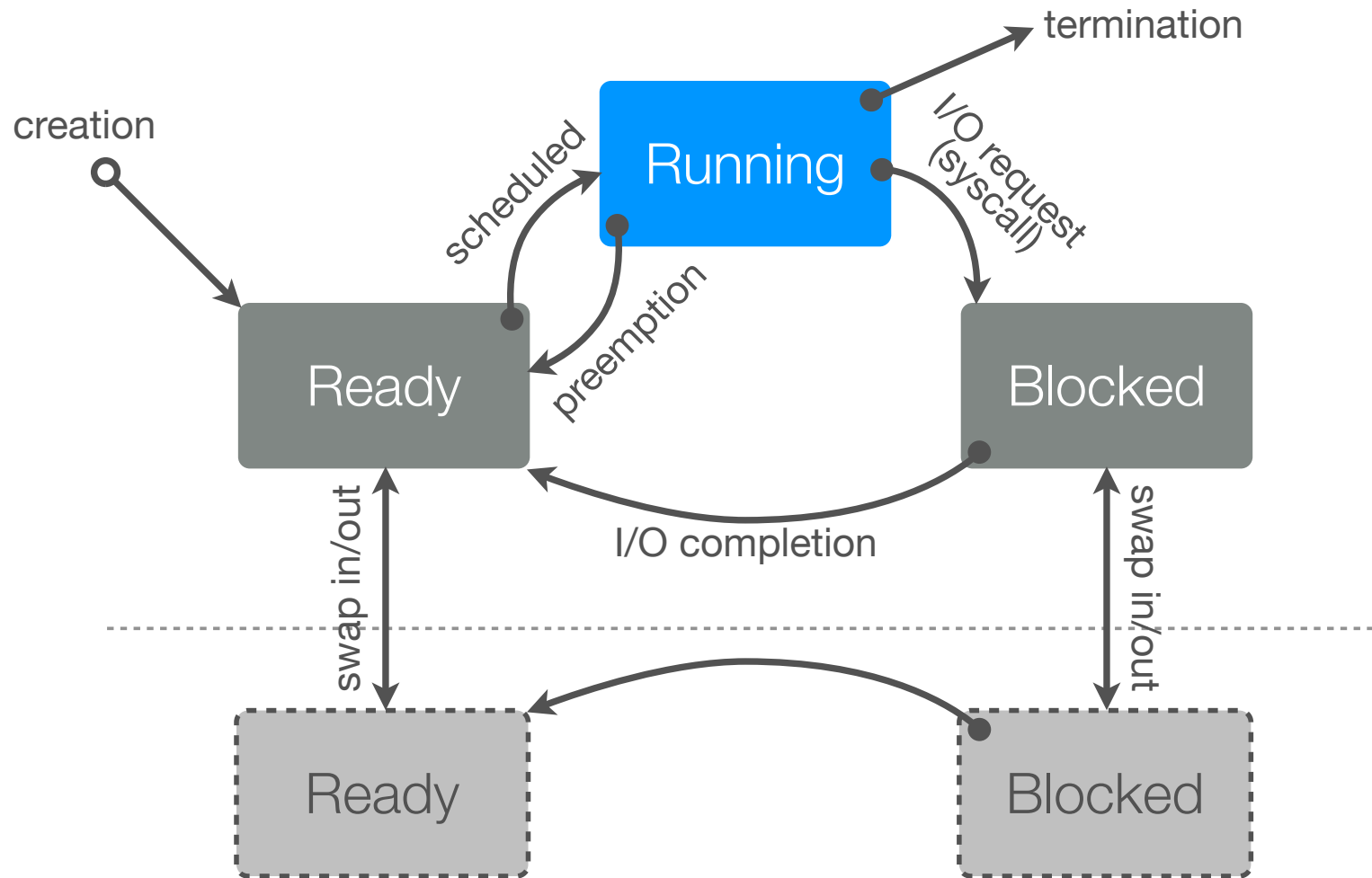
run/wait/swapped time,
memory usage,
priorities, etc.

accounting info typically
updated per clock tick

tracked by central
kernel data structure:
Process Control Block

scheduling = managing
process state transitions

(macro) **process state**
ready / running / blocked



process (CPU) scheduling

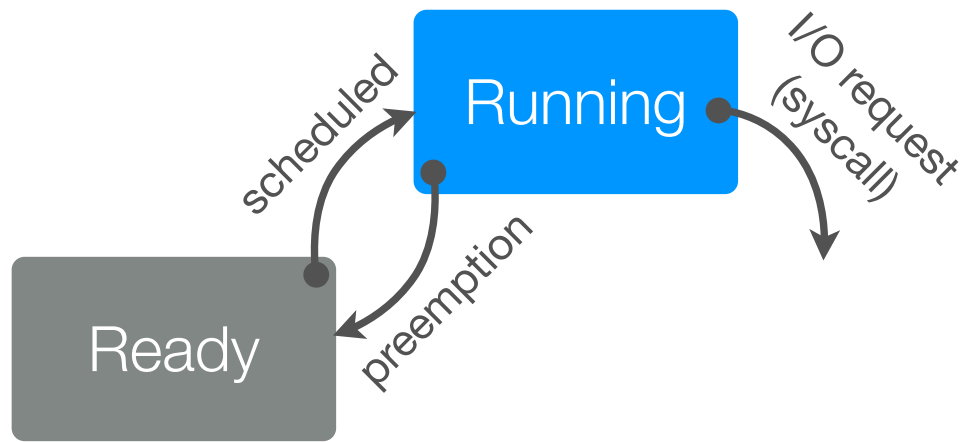
preemptive scheduling

running → ready

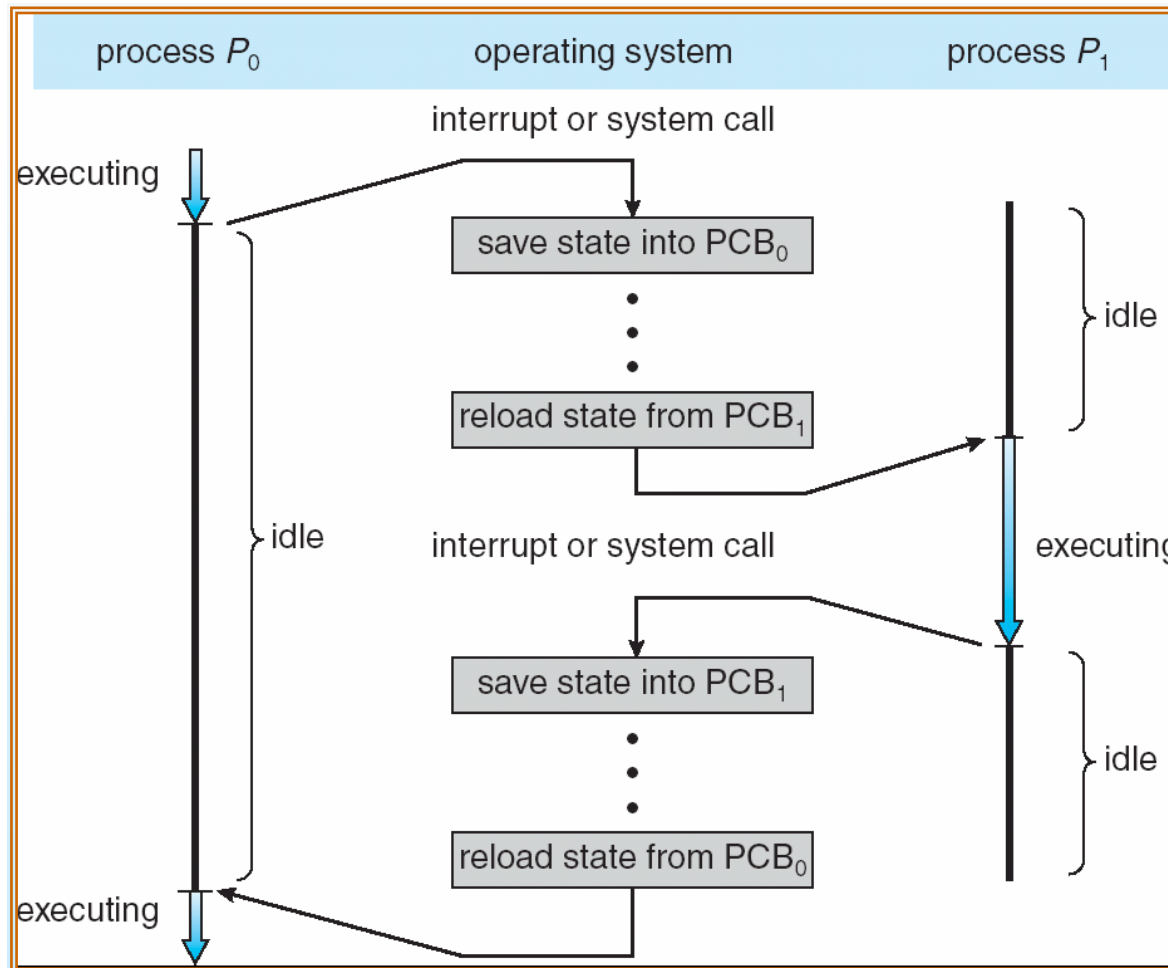
non-preemptive scheduling

☒ running → ready

not = batch!



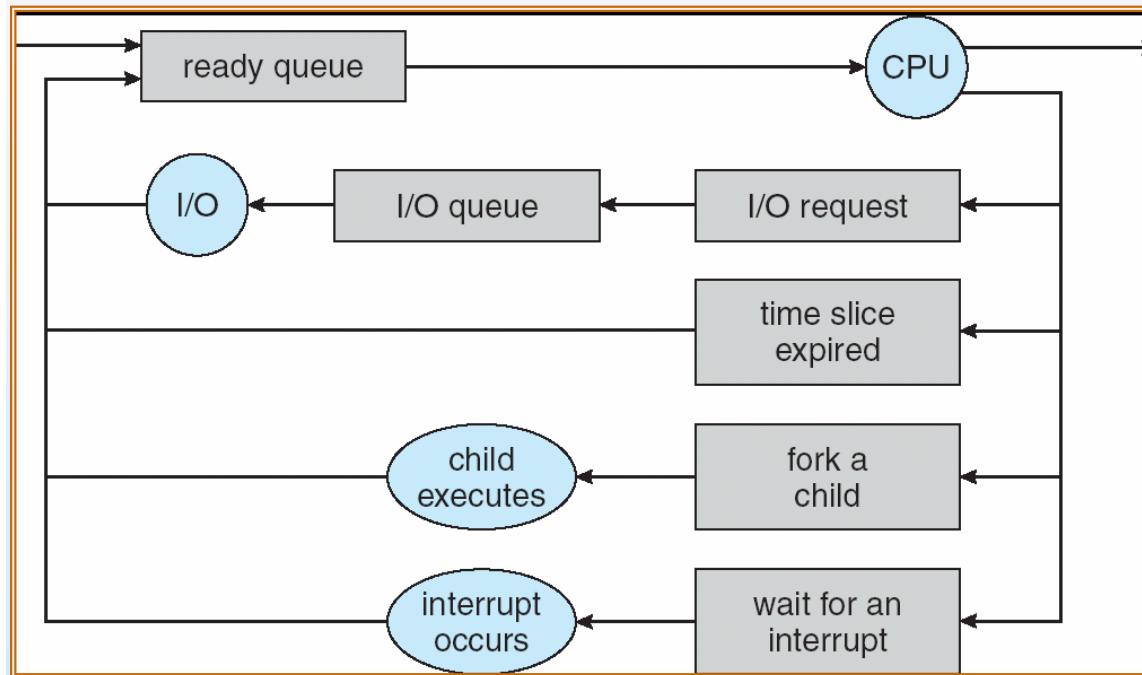
context switch



context switch

Ready

ready queue
“eligible” processes



re-entering the ready queue

scheduling metric:

wait time:

time in ready queue


scheduling metric:

turnaround time:

total start to end latency
(ready+running+blocked)

no control over total time
running & blocked

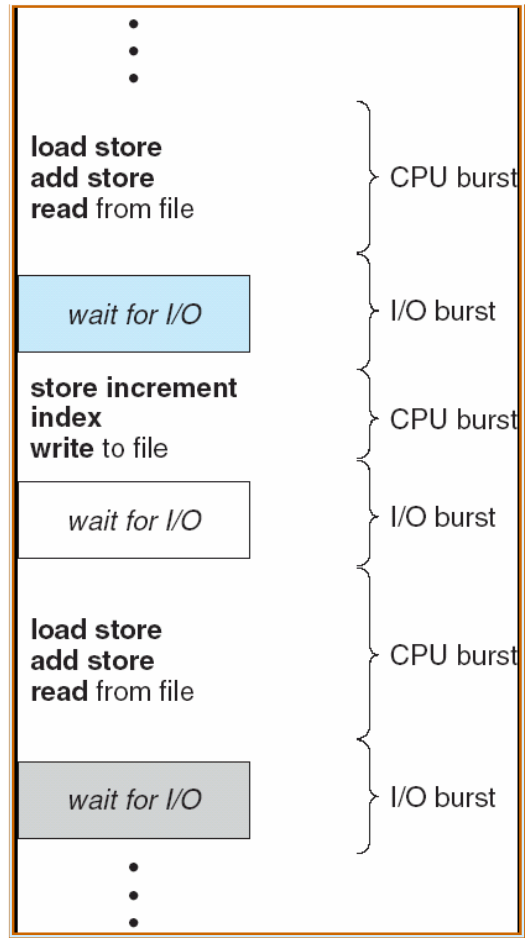
(at least, not controlled
by CPU scheduler)



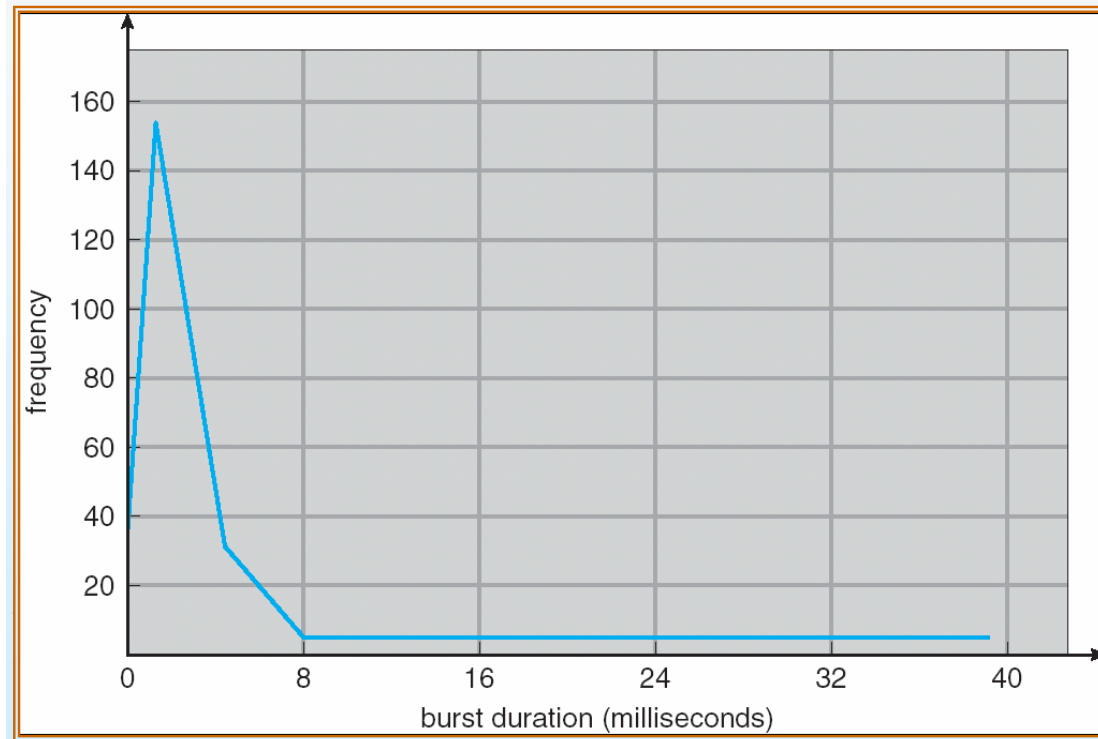
CPU bound vs. I/O bound



not as common
as you might think!



“Bursty” Execution

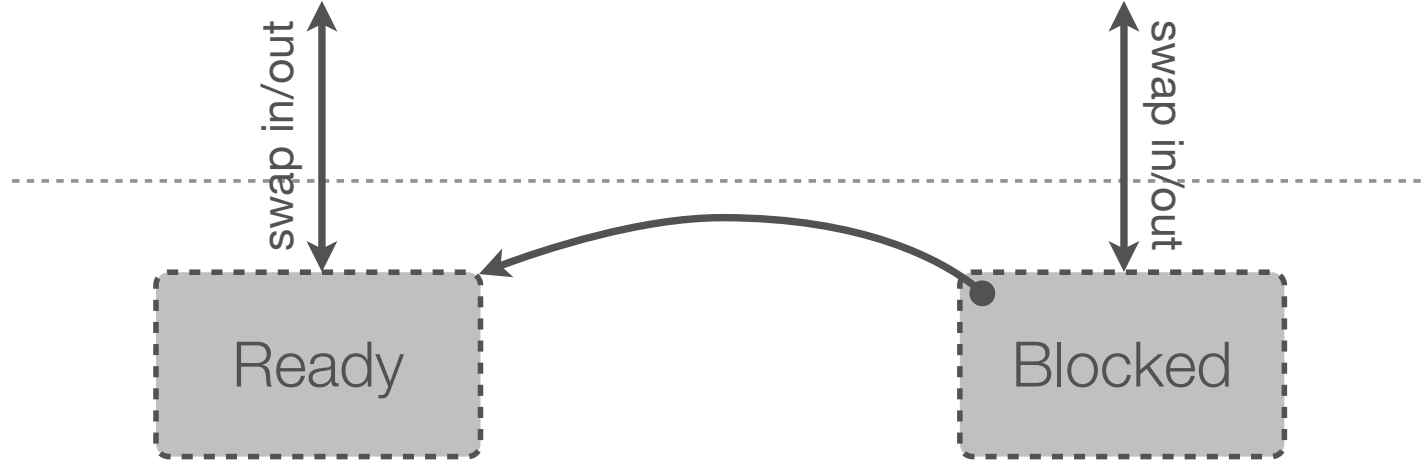


CPU Burst Histogram

scheduling metric:

response time:

CPU burst latency



short term scheduling

VS.

long term scheduling
(swapping)

frequency: ms vs. sec/min

conflicting schedulers?

and others:
e.g., I/O, Disk schedulers

goal:

minimize wait, turnaround,
and response time

macro-scheduling metrics

scheduling metric:

throughput:

scheduled processes/
bursts per time unit (sec)

scheduling metric:

utilization:

% time CPU is

kept “busy” (vs. idle)

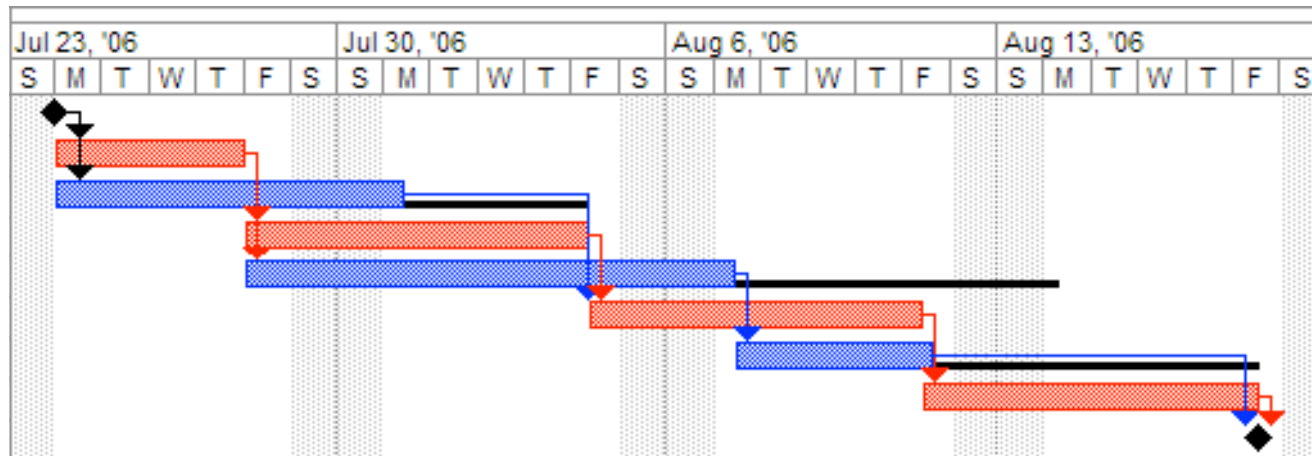
goal:

maximize throughput
and utilization

fairness?

Scheduling Algorithms

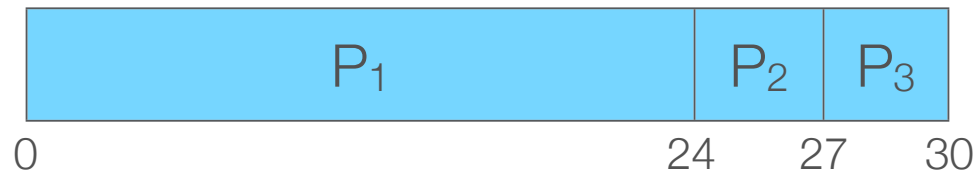
ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



Gantt Charts

First Come **First S**erved

Process	Arrival Time	Burst Time
P ₁	0	24
P ₂	0	3
P ₃	0	3



Waiting times: P₁ = 0, P₂ = 24, P₃ = 27

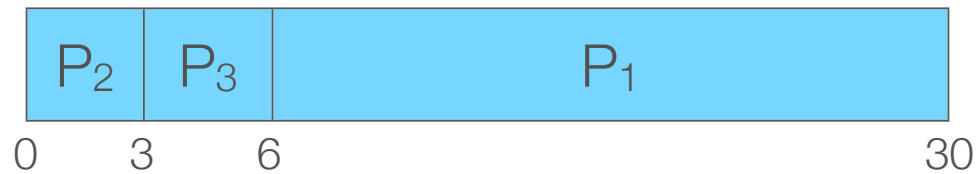
Average: $(0+24+27)/3 = 17$

First Come First Served



Convoy Effect

Process	Arrival Time	Burst Time
P ₃	0	3
P ₂	0	3
P ₁	0	24



Waiting times: P₂ = 0, P₃ = 3, P₁ = 6

Average: $(0+3+6)/3 = 3$

“Optimized” FCFS

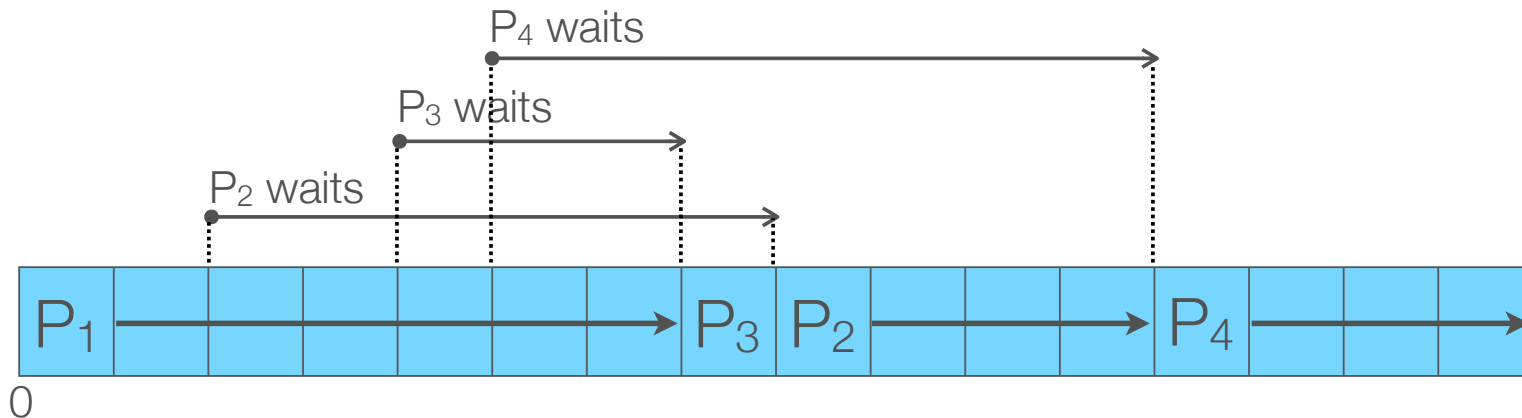
Shortest Job First

“greedy” optimization
always pick local optimum

globally optimal?

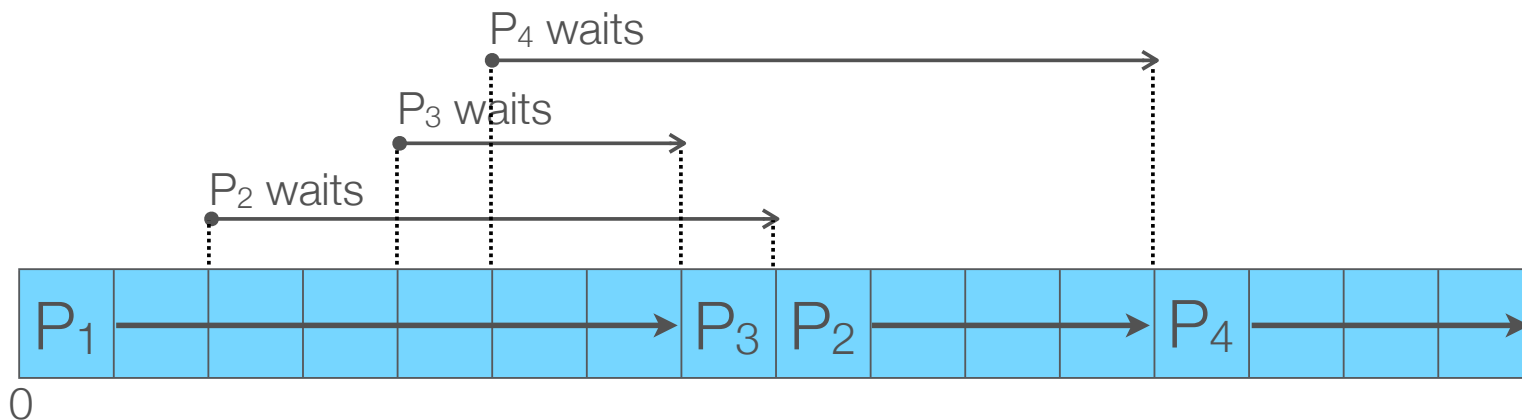
- ordered wait times:
 - t_1, t_2, t_3, t_4
 - avg wait:
 - $(4t_1 + 3t_2 + 2t_3 + t_4) / 4$
- largest wait time at end is provably optimal!

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4



Non-Preemptive SJF

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4



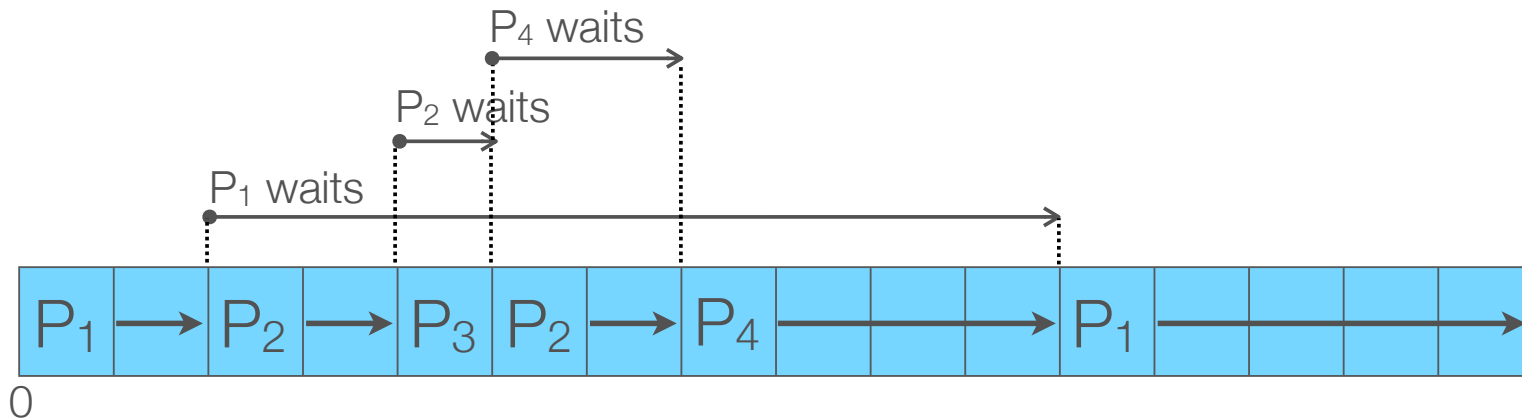
Wait times: P₁ = 0, P₂ = 6, P₃ = 3, P₄ = 7
Average: $(0+6+3+7)/4 = 4$

Preemptive SJF

a.k.a.

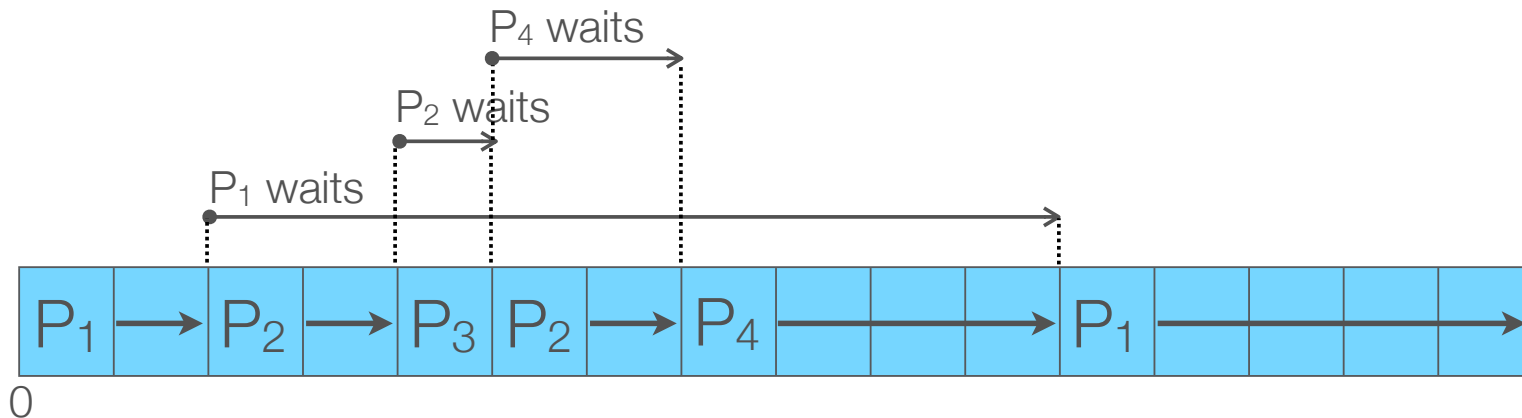
Shortest-Remaining-Time-First

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4



Preemptive SJF

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4



Wait times: P₁ = 9, P₂ = 1, P₃ = 0, P₄ = 2

Average: $(9+1+0+2)/4 = 3$

optimizes average
waiting time

at what cost?

potential **starvation**

simplification & assumption

1. discounting context
switches

2. known upcoming
CPU burst length

prediction

moving average

exponentially weighted

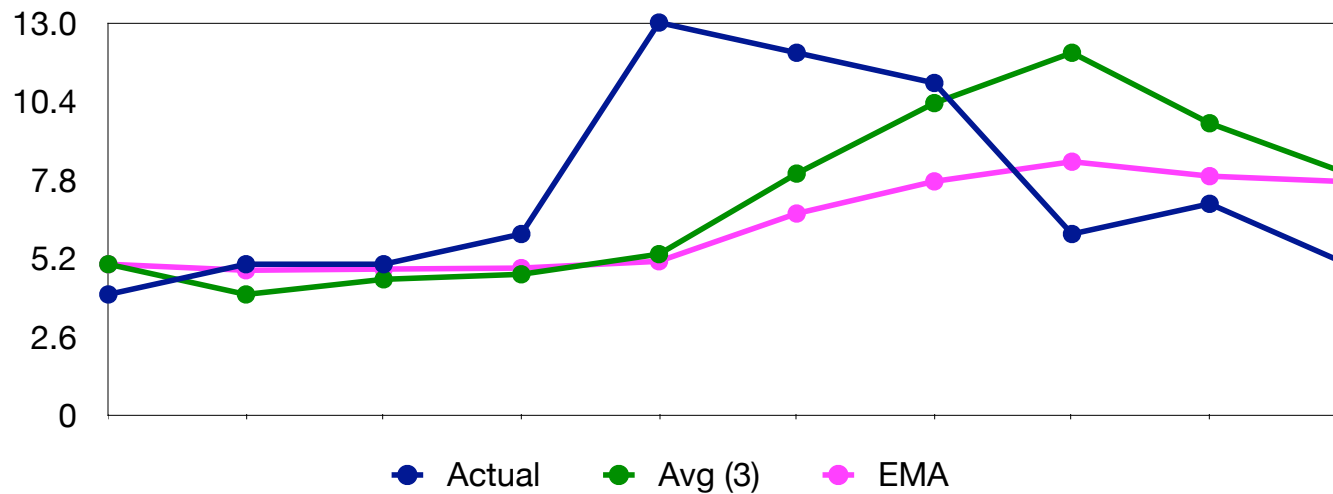
Observed: Y_{n-1}

Estimated: S_{n-1}

Weight (α): $0 \leq \alpha \leq 1$

EMA: $S_n = \alpha \cdot Y_{n-1} + (1 - \alpha) \cdot S_{n-1}$

Actual	Avg (3)	Error	EMA	Error			
4	5.00	1.00	5.00	1.00		EMA Alpha:	0.2
5	4.00	1.00	4.80	0.20			
5	4.50	0.50	4.84	0.16			
6	4.67	1.33	4.87	1.13			
13	5.33	7.67	5.10	7.90			
12	8.00	4.00	6.68	5.32			
11	10.33	0.67	7.74	3.26			
6	12.00	6.00	8.39	2.39			
7	9.67	2.67	7.92	0.92			
5	8.00	3.00	7.73	2.73			
Avg err:		2.78		2.50			

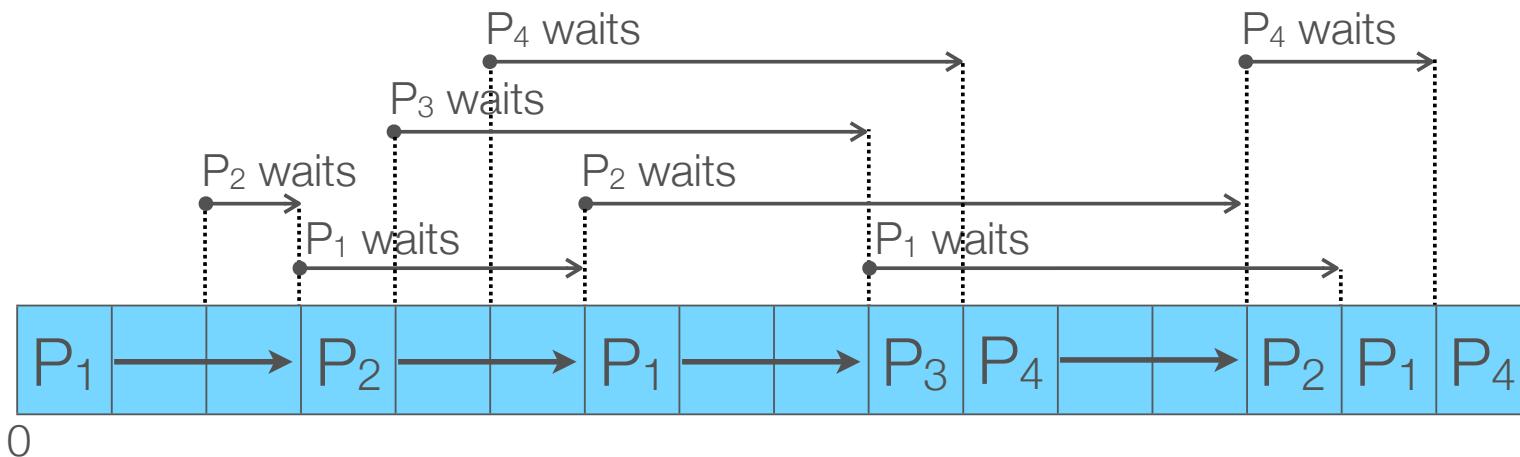


combatting starvation:
enforce *fairness*

Round Robin
circular queue
time quantum

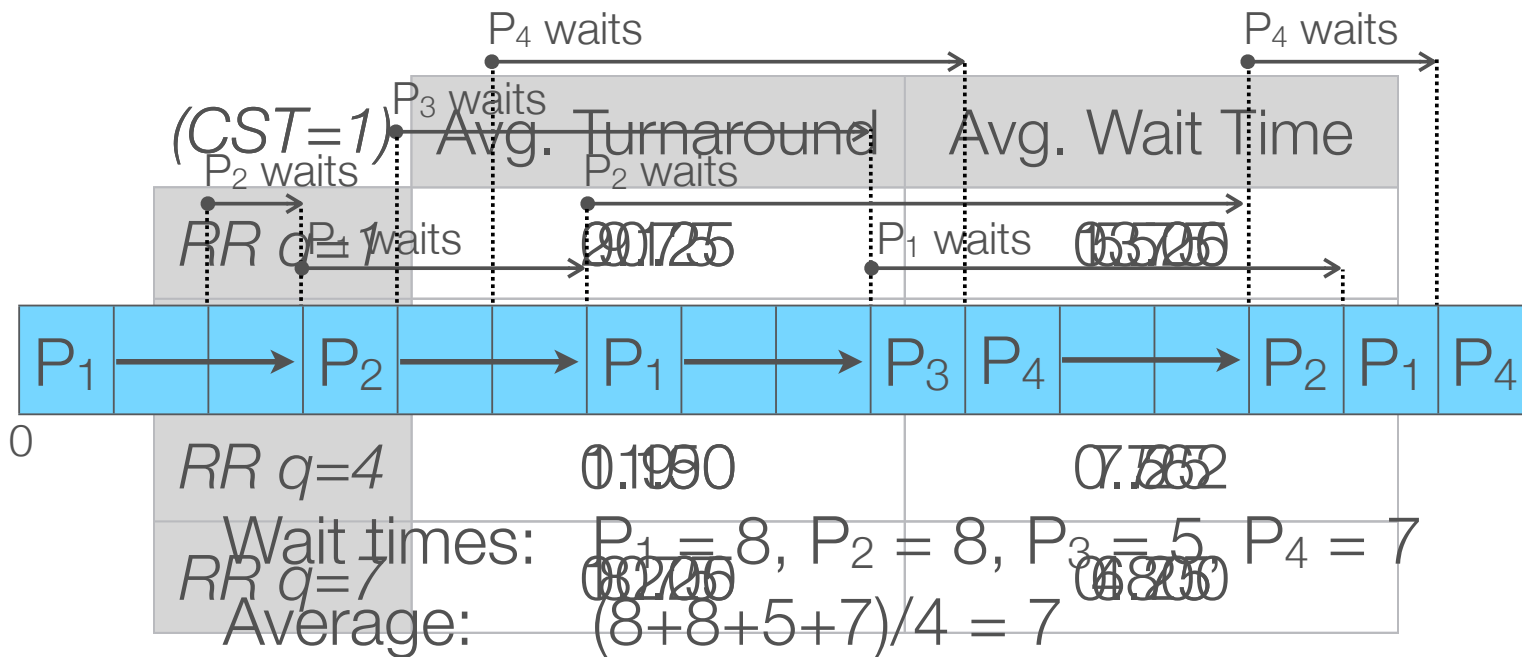
n processes, quantum **q**
 $1/n$ time share per process
wait $\leq \mathbf{q \times (n-1)}$

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4



Round Robin, $q=3$

Process	Arrival Time	Burst Time
P ₁	0	7
P ₂	2	4
P ₃	4	1
P ₄	5	4



q large \Rightarrow FIFO

q small \Rightarrow big CST overhead

quantum length ↓
throughput ↓
responsiveness ↑

...to a point

ideally, tune q
to CPU burst lengths

(how?)

max response time
median of EMAs
process profiling

fairness is overrated!

fine grained control
arbitrary *priorities*
highest priority goes next

priority schedulers

e.g., SJF

prone to starvation

combat with priority **aging**

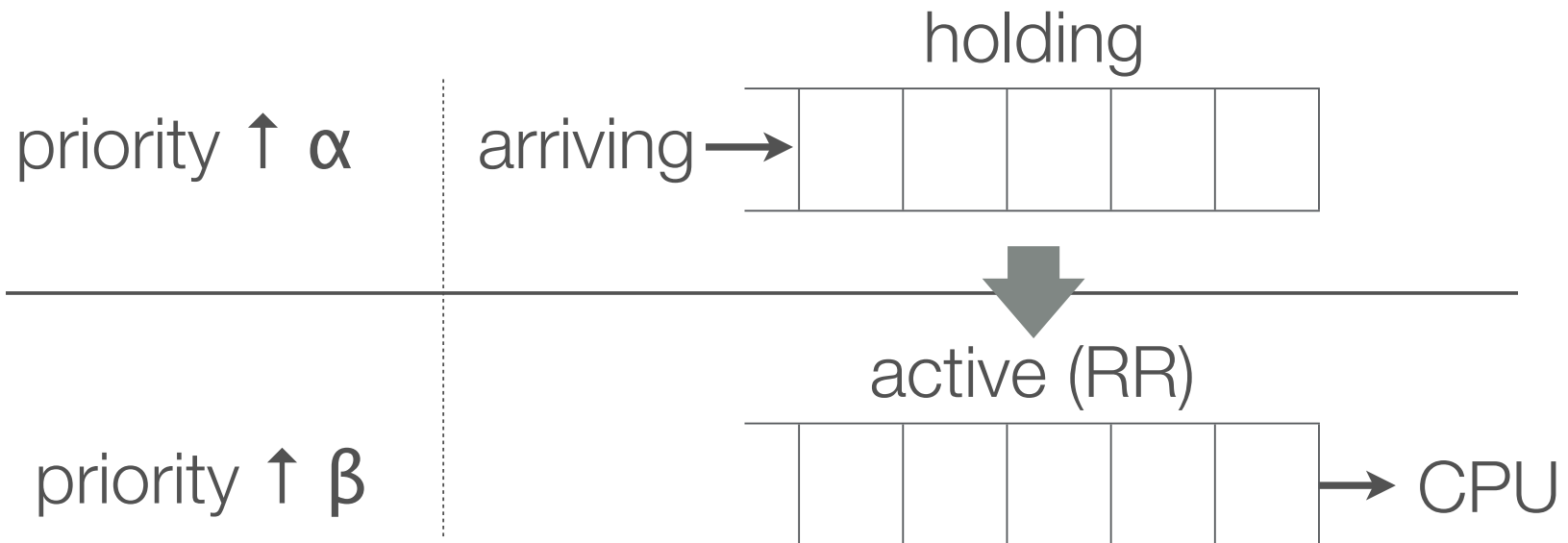
t : execution time

T : “wall clock” age

penalty ratio: T/t

Highest Penalty Ratio Next
combats starvation

Selfish RR



$\beta = 0$: RR

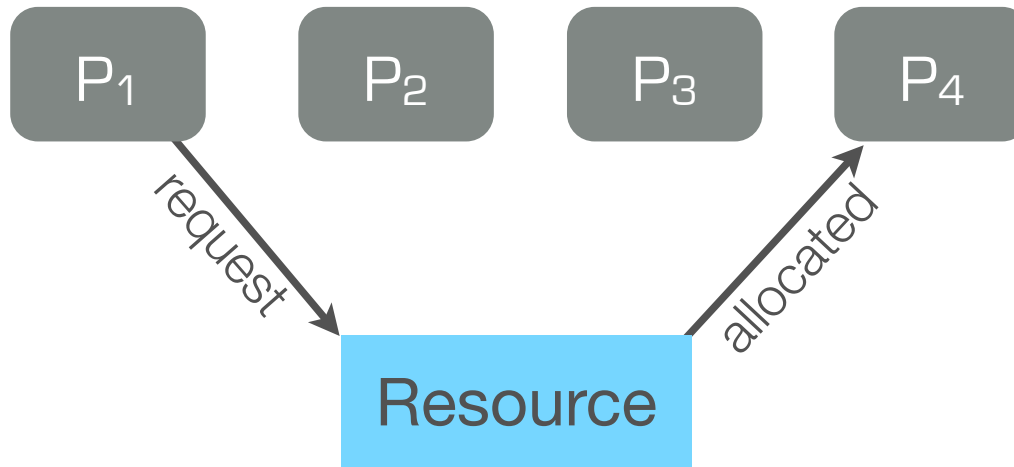
$\beta \geq (\alpha \neq 0)$: FCFS

$\beta > (\alpha = 0)$: RR in batches

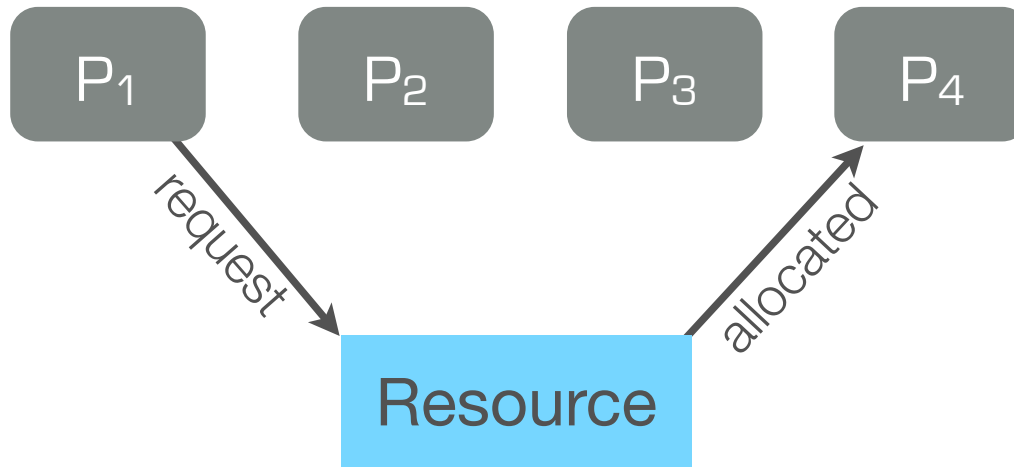
$\alpha > \beta > 0$: “Selfish” RR

ageism

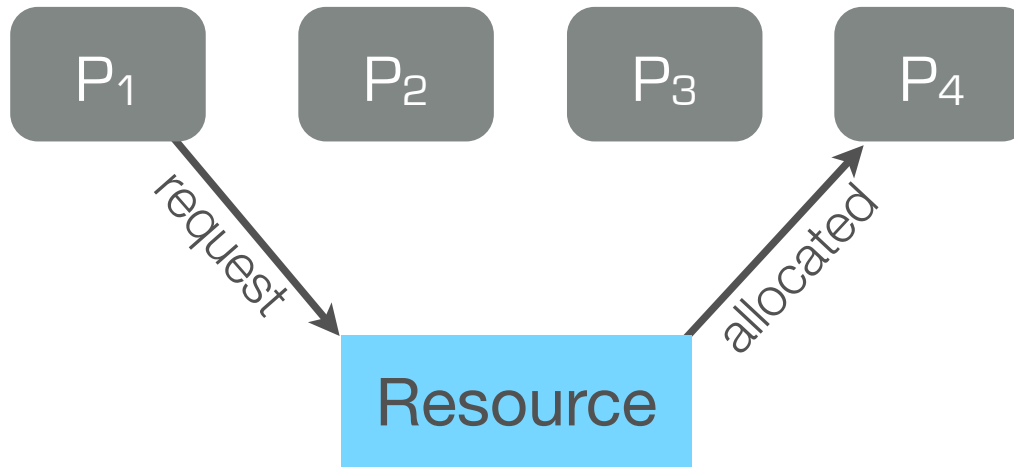
Process	Priority	State
P ₁	High	Running
P ₂	Mid	Ready
P ₃	Mid	Ready
P ₄	Low	Ready



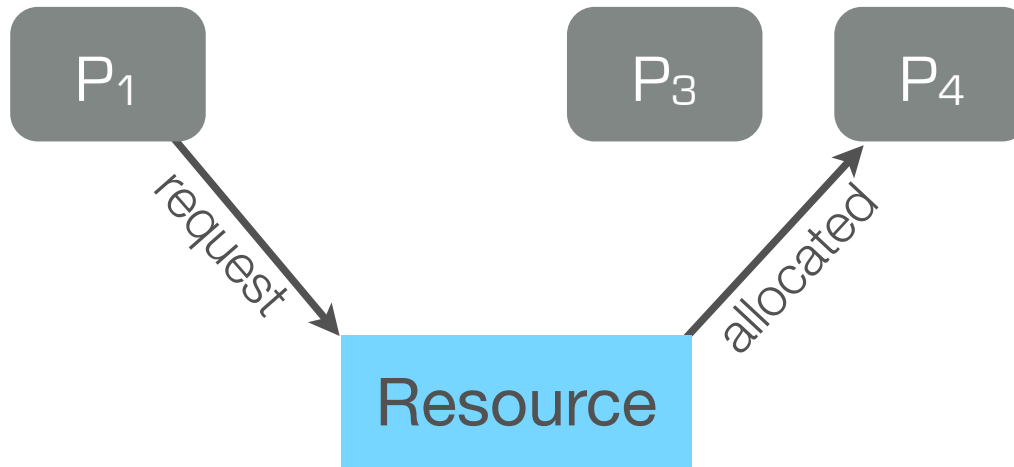
Process	Priority	State
P ₁	High	Blocked
P ₂	Mid	Ready
P ₃	Mid	Ready
P ₄	Low	Ready



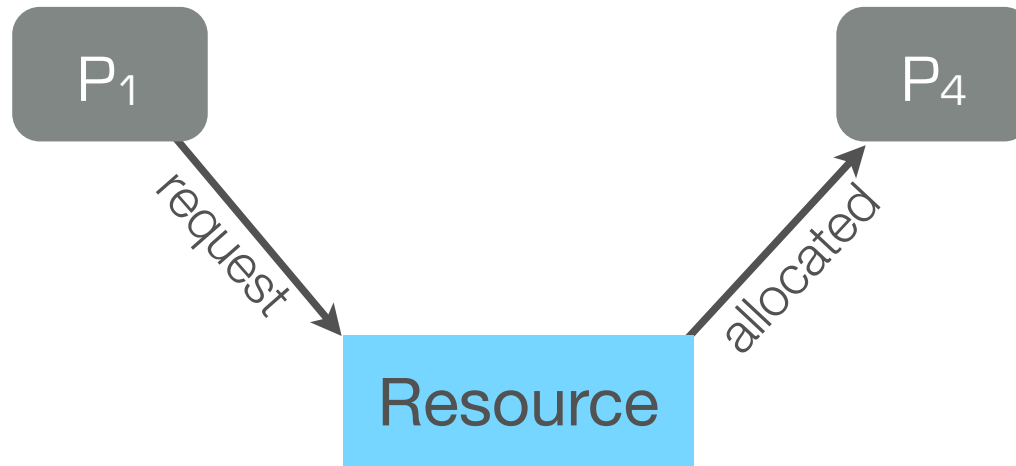
Process	Priority	State
P ₁	High	Blocked
P ₂	Mid	Running
P ₃	Mid	Ready
P ₄	Low	Ready



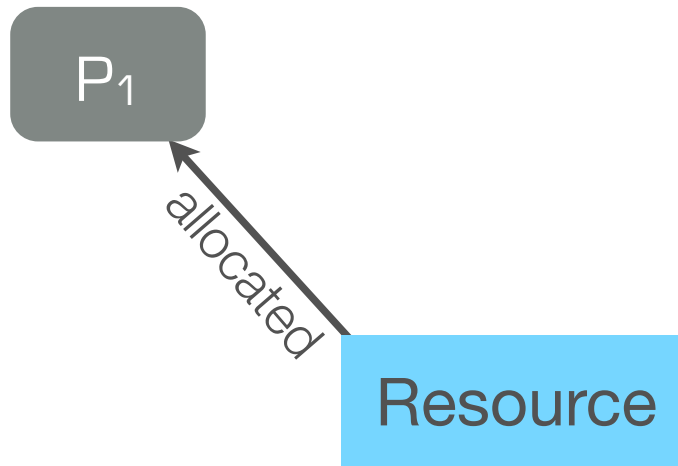
Process	Priority	State
P ₁	High	Blocked
P ₃	Mid	Running
P ₄	Low	Ready



Process	Priority	State
P ₁	High	Blocked
P ₄	Low	Running



Process	Priority	State
P ₁	High	Running



priority **inversion**

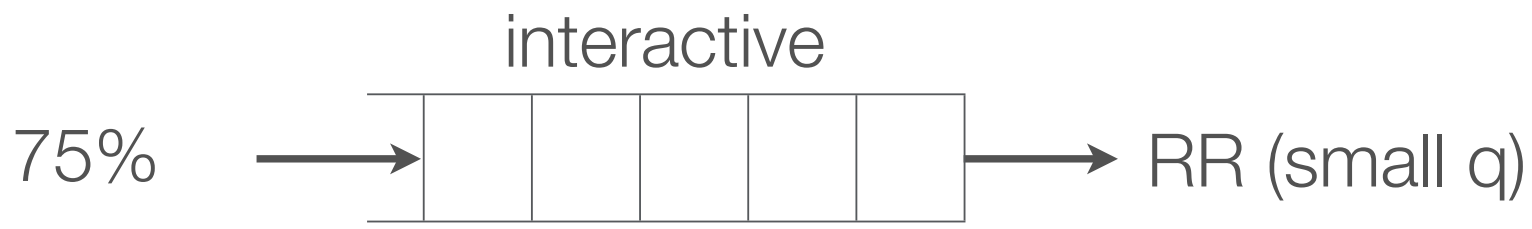
fix?

artificially elevate priority
mechanism?

scheduling requirements vary
process requirements vary
process characteristics change

Multi-Level Queue

disparate schedulers
queue selection

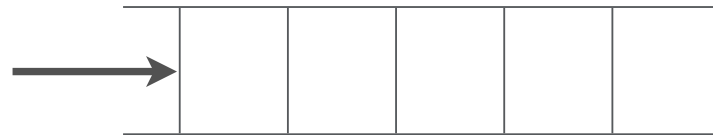


queue selection criteria?

dynamic process
requirements?

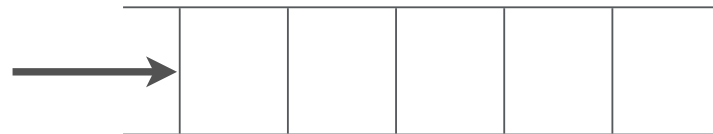
Multi-Level Feedback Queue

∞



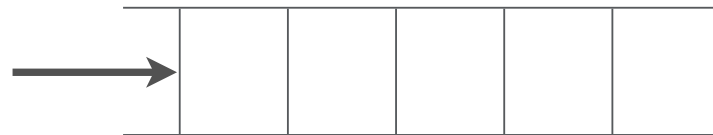
Fixed priority

75%



RR (small q)

25%



RR (larger q)

*



FCFS

exponential backoff
to avoid queue-hopping

§ Scheduler Evaluation

- paper & pencil
- simulation
- queueing theory
- real world implementation

Demo

Which is which?

Table Data										Entries		Average Time	
Name	Key	Time	Processes	Finished	CPU Utilization	Throughput	CST	LA	CPU	I/O	CPU	I/O	
secret_1	ALG 1	10550.00	100	100	.947867	.009479	550.00	91.36	1375	770	7.27	50.20	
secret_2	ALG 2	10511.66	100	100	.951324	.009513	348.00	59.74	870	770	11.49	50.20	
secret_3	ALG 3	10376.90	100	100	.963679	.009637	348.00	88.01	870	770	11.49	50.20	
secret_4	ALG 4	10588.08	100	100	.944459	.009445	440.80	59.72	1102	770	9.07	50.20	

Name	Key	Turnaround Time				Waiting Time			
		Average	Minimum	Maximum	SD	Average	Minimum	Maximum	SD
secret_1	ALG 1	10124.63	8887.82	10549.80	405.48	9637.08	8435.62	10046.80	3.72
secret_2	ALG 2	6765.84	1956.80	10511.46	2342.38	6279.30	1455.20	10045.31	23.57
secret_3	ALG 3	9619.54	7277.89	10376.70	712.98	9133.00	6926.89	9774.70	6.65
secret_4	ALG 4	6809.22	1967.20	10587.88	2370.05	6322.22	1465.60	10121.12	23.85

Done

SJF, PSJF, RR (q=10), RR (q=20)
(processes: uniform burst ≤ 20 , CST = 1.0)