CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Scheduling Algorithms -- SJF

First-come, first-served is the simplest CPU-scheduling algorithm

However, tends to lead to long waiting times

Shortest job first (SJF) scheduling is an alternative that can increase throughput

In SJF scheduling, the CPU is assigned to the job with the shortest next CPU burst

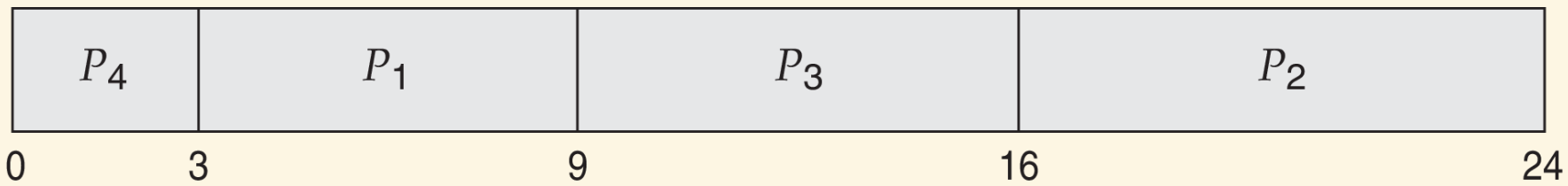
FCFS can be used to break ties in case two jobs have the same duration of next CPU burst

Note that the "shortest job" isn't really run -- it is the job with the shortest next CPU burst

Four processes arrive at time 0 as follows

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

(times are in milliseconds)


P_4
P_1
P_3
P_2
0
3
9
16
24

Note that, unlike with FCFS, we do not need to know the order in which the processes arrived (except in the case of breaking a tie)

Resulting waiting times are

Process	Waiting Time
P1	3
P2	16
P3	9
P4	0

Average waiting time: 7 ms

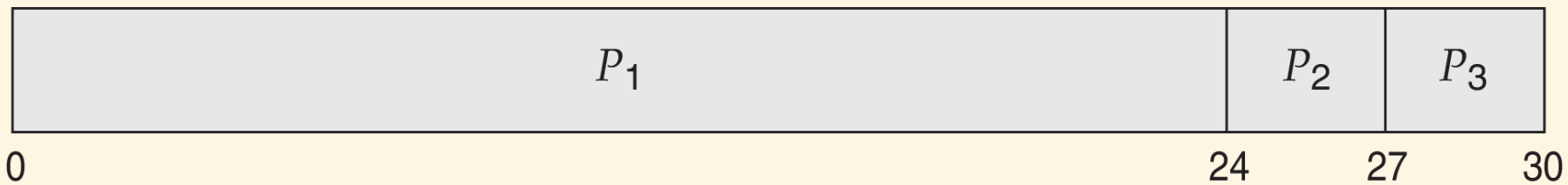
How does this compare to FCFS?

SJF: 7 ms

FCFS: 10.25 ms

SJF scheduling is provably optimal at producing minimum average waiting time

Proving optimality is easy in this case:



SJF can be either preemptive or nonpreemptive

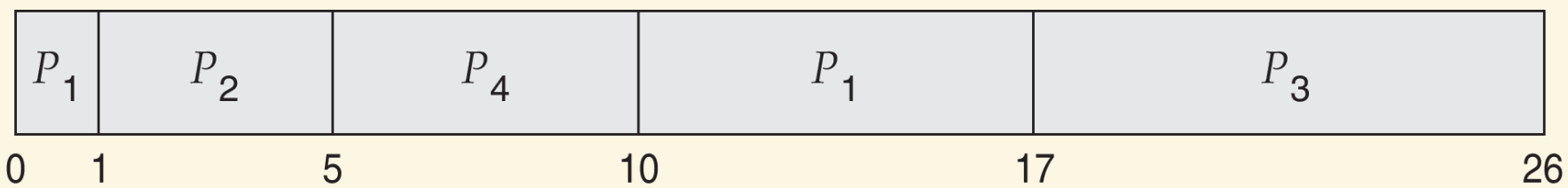
This determines what is done when a new process arrives at the ready queue

With preemption, compare time of newly arrived process against *remaining* time of current process

What does the Gantt chart look like for the following scenario with preemptive scheduling?

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



Average waiting times?

Preemptive: 6.5 ms

Nonpreemptive: 7.75 ms

Important to remember that we are comparing *remaining* time

Process	Arrival Time	Burst Time
P1	0	20
P2	18	4

Something to consider: how do we know the future burst time of a process?

SJF is easy for long-term scheduling, but is not truly possible for short-term scheduling

Instead, we must estimate (guess) the duration of the next CPU burst