

CS162
Operating Systems and
Systems Programming
Lecture 10

Scheduling
Core Concepts and Classic Policies

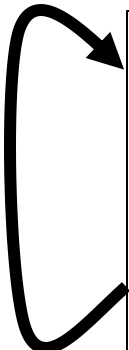
Professor Natacha Crooks & Matei Zaharia

<https://cs162.org/>

Goals for Today

- What is scheduling?
- What makes a good scheduling policy?
- What are existing schedulers and how do they perform?

The Scheduling Loop!



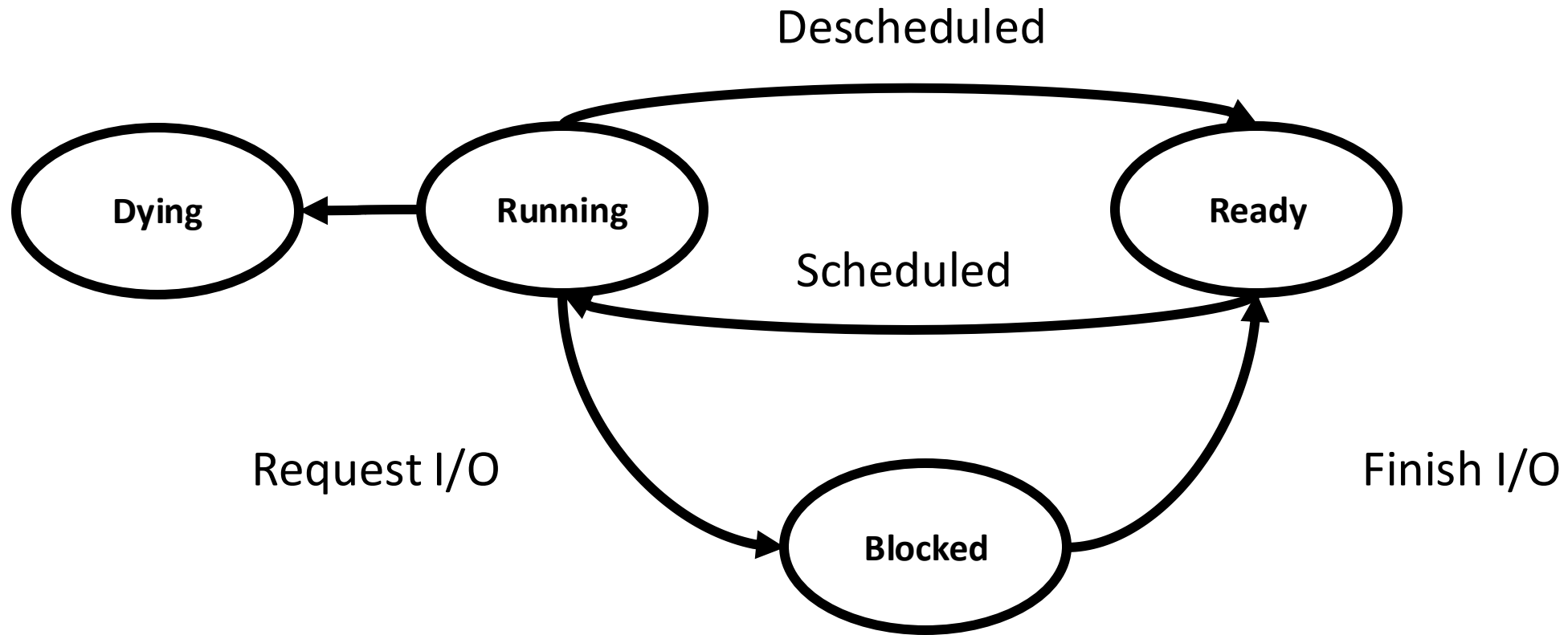
```
if (readyThreads(TCBs) ) {  
    nextTCB = selectThread(TCBs);  
    run(nextTCB);  
} else {  
    run_idle_thread();  
}
```

1. Which task to run next?

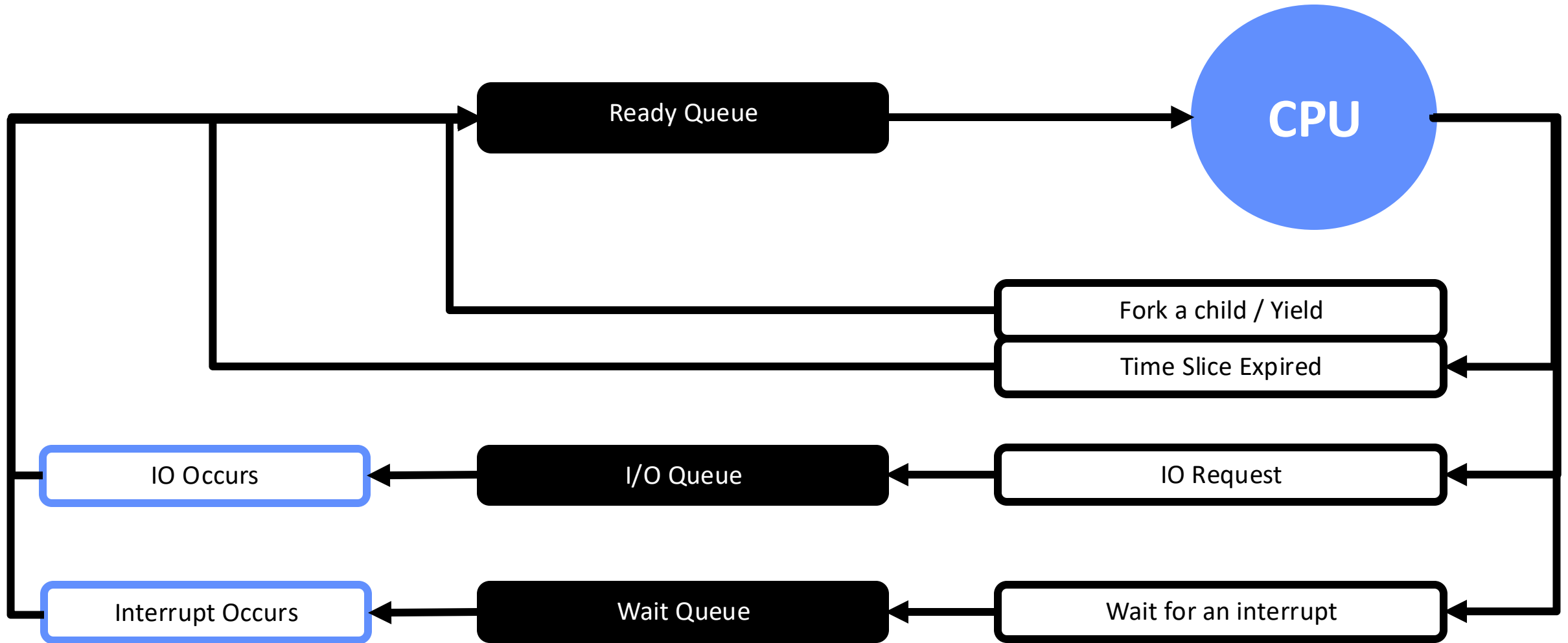
2. How frequently does this loop run?

3. What happens if `run` never returns?

Recall: Thread Life Cycle



Recall: What triggers a scheduling decision?



What makes a good scheduling policy?

A hopeless queue.

48-hour queue for tickets to the Beatles in 1963.



A bad but more realistic queue.

The DMV.

What makes a good scheduling policy?

What does the DMV care about?



What do individual users care about?



Important Performance Metrics

Response time (or latency).

User-perceived time to do some task

Throughput.

The rate at which tasks are completed

Scheduling overhead.

The time to switch from one task to another.

Predictability.

Variance in response times for repeated requests.

Important Performance Metrics

Fairness

Equality in the performance perceived by one task

Starvation

The lack of progress for one task, due to resources being allocated to different tasks

Sample Scheduling Policies

Assume DMV job A takes 1 second, job B takes 2 days

Policy Idea: Only ever schedule users with Job A

What is the metric we are optimizing?

A) Throughput B) Latency C) Predictability D) Low-Overhead

Can the schedule lead to starvation?

A) Yes B) No

Is the schedule fair?

A) Yes B) No

Sample Scheduling Policies

Assume DMV consists only of jobs of type A.

Policy Idea: Schedule jobs randomly

What is the metric we are optimizing?

A) Throughput B) Latency C) Predictability D) Low-Overhead

Can the schedule lead to starvation?

A) Yes B) No

Is the schedule fair?

A) Yes B) No

Sample Scheduling Policies

Assume DMV consists only of 100 different types of jobs. Some jobs need Clerk A, some Clerks A&B, others Clerk C.

Policy Idea Every time schedule a job, compute all possible orderings of jobs, pick one that finishes quickest

What is the metric we are optimizing?

A) Throughput B) Latency C) Predictability D) Low-Overhead

Can the schedule lead to starvation?

A) Yes B) No

Is the schedule fair?

A) Yes B) No

Scheduling Policy Goals/Criteria

Minimise Latency

Maximise Throughput

While remaining fair and starvation-free

Useful metrics

Waiting time for P

Total Time spent waiting for *CPU*

**WARNING: our textbooks
use different terms for
some of these metrics!**

Average waiting time

Average of all processes' wait time

Response Time for P

Time to when process gets first scheduled

Completion time

Waiting time + Run time

Average completion time

Average of all processes' completion time

Assumptions

Threads are independent!

One thread = One User

Unrealistic but simplify the problem so it can be solved

Only look at **work-conserving** schedulers
=> Never leave processor idle if work to do

Workload Assumptions

A workload is a set of tasks for some system to perform, including how long tasks last and when they arrive

Compute-Bound

Tasks that primarily perform compute

Fully utilize CPU

IO Bound

Mostly wait for IO, limited compute in “bursts”

Often in the Blocked state

First-Come, First-Served (FCFS)

Run tasks in order of arrival.

Run task until completion (or blocks on IO).
No preemption

This is the DMV model.

Also called FIFO (First-In First-Out).

First-Come, First-Served (FCFS)

Process

Burst Time

P_1

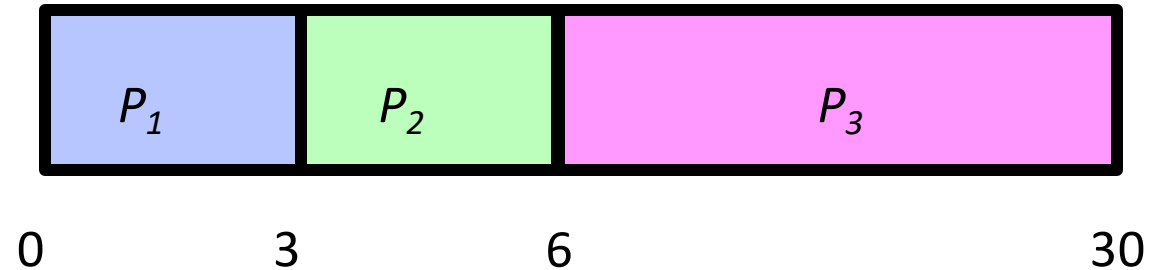
3

P_2

3

P_3

24



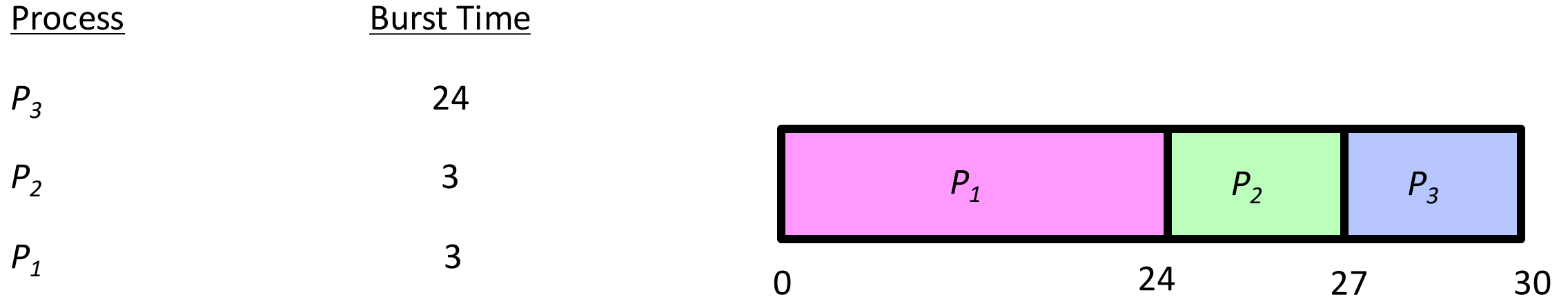
What is the average completion time?

$$\left(\frac{3+6+30}{3} = 13 \right)$$

What is the average waiting time?

$$\left(\frac{0+3+6}{3} = 3 \right)$$

First-Come, First-Served (FCFS)



What is the average completion time?

$$\left(\frac{24+27+30}{3} = 27\right)$$

What is the average waiting time?

$$\left(\frac{0+24+27}{3} = 17\right)$$

The Convoy Effect

FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is non-preemptible

The Convoy Effect

FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

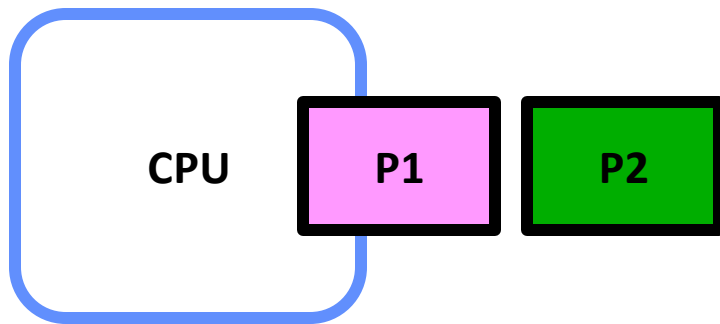
FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

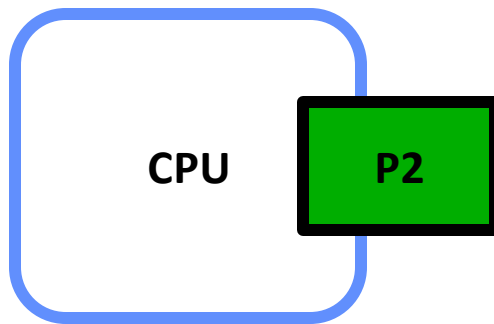
FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

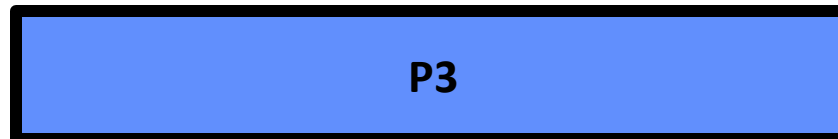
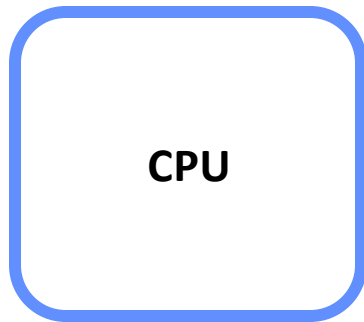
FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

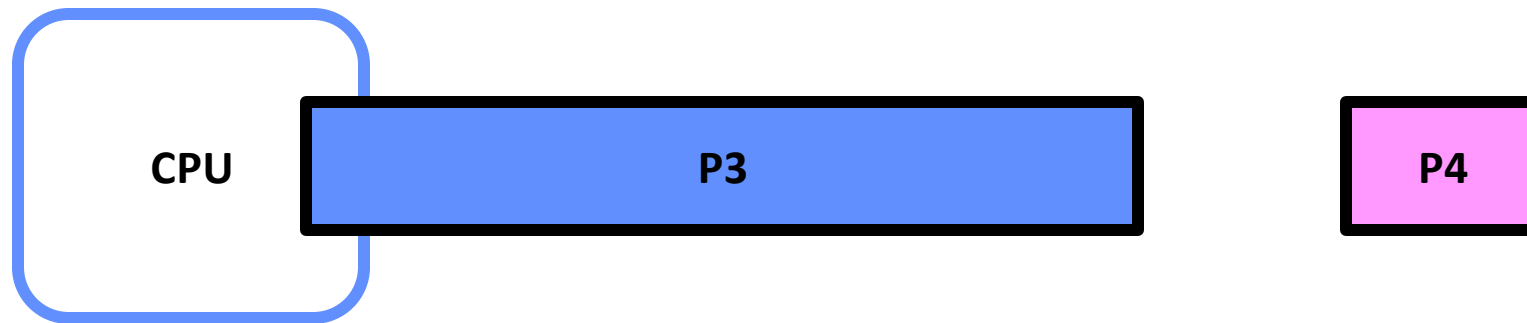
FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

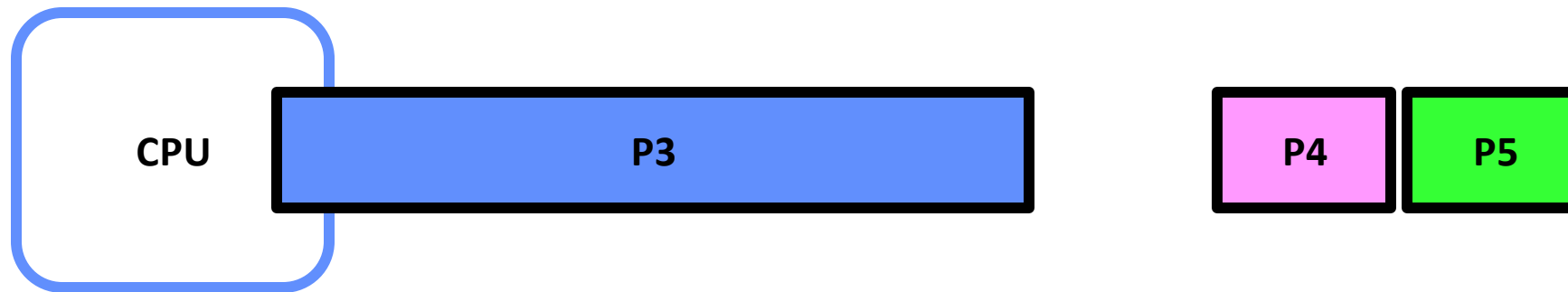
FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

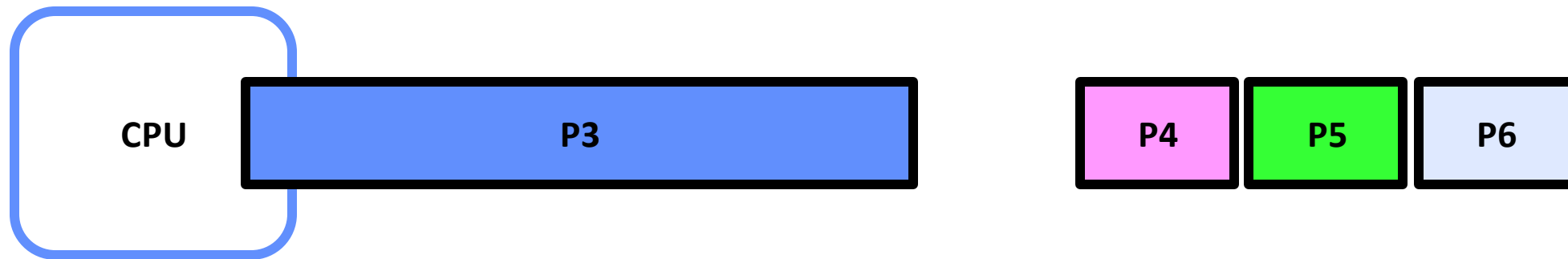
FCFS/FIFO very sensitive to arrival order

Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*



The Convoy Effect

FCFS/FIFO very sensitive to arrival order

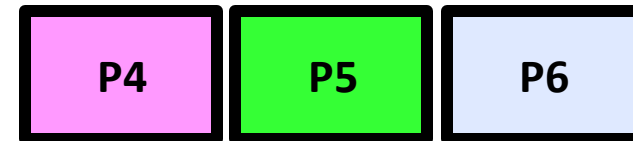
Convoy effect

Short process stuck behind long process

Lots of small tasks build up behind long tasks

FCFS is *non-preemptible*

Can FCFS lead to starvation?



FCFS/FIFO Summary

The good

Simple
Low Overhead
No Starvation*

The bad

Sensitive to arrival order (poor predictability)

The ugly

Convoy Effect.
Bad for Interactive Tasks

*Assuming jobs eventually finish

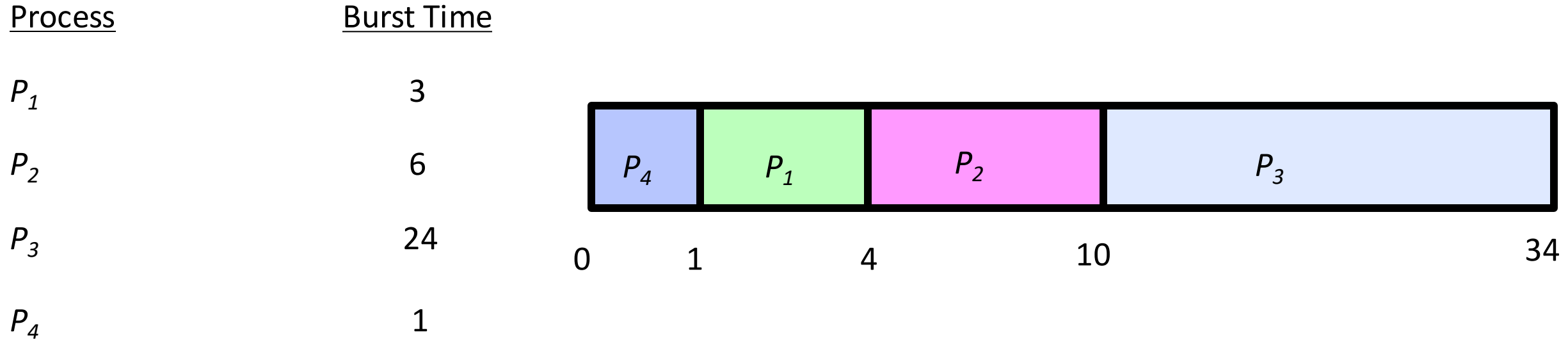
Shortest Job First

How can we minimize average completion time?

By scheduling jobs in order of
estimated completion time

This is the “10 items or less” line at Safeway

Shortest Job First



What is the average completion time?

$$\left(\frac{1+4+10+34}{4} = 12.25 \right)$$

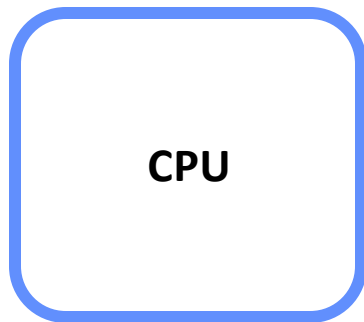
Can prove that SJF generates optimal average completion time if
all jobs arrive at the same time

Are we done?

Can SJF lead to starvation?

Yes

Any scheduling policy that always favours a **fixed property** for scheduling leads to starvation

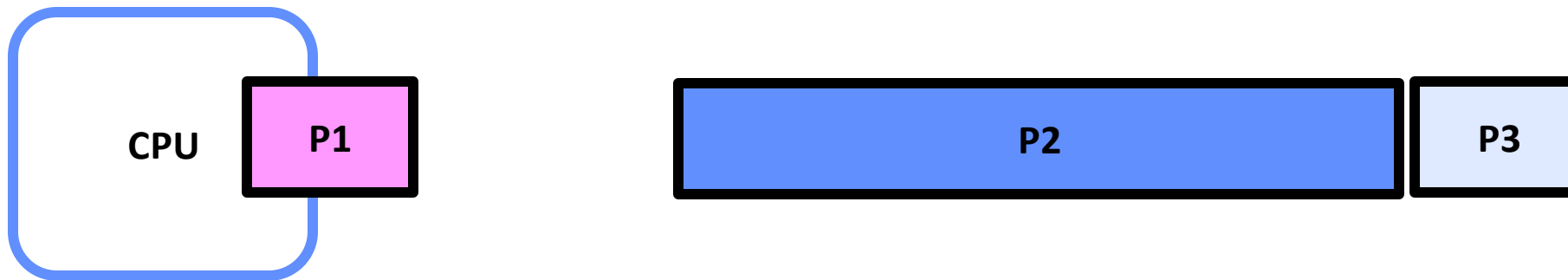


Are we done?

Can SJF lead to starvation?

Yes

Any scheduling policy that always favours a **fixed property** for scheduling leads to starvation

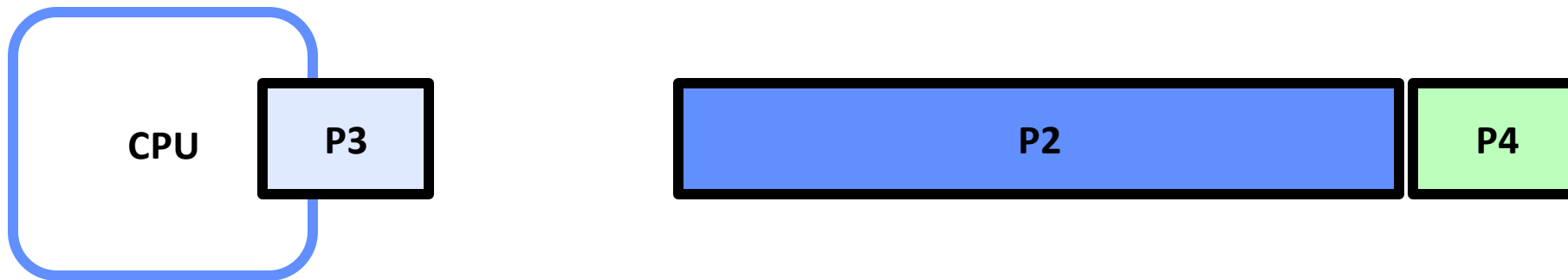


Are we done?

Can SJF lead to starvation?

Yes

Any scheduling policy that always favours a **fixed property** for scheduling leads to starvation

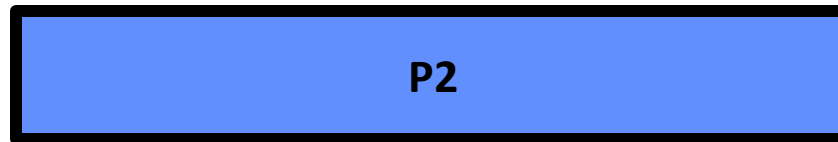
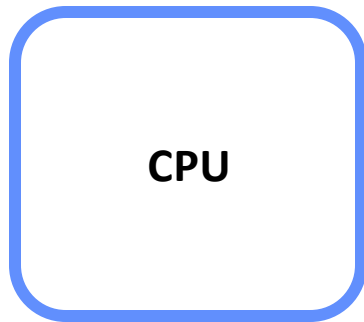


Are we done?

Is SJF subject to the convoy effect?

Yes

Any **non-preemptible** scheduling policy suffers from
convoy effect

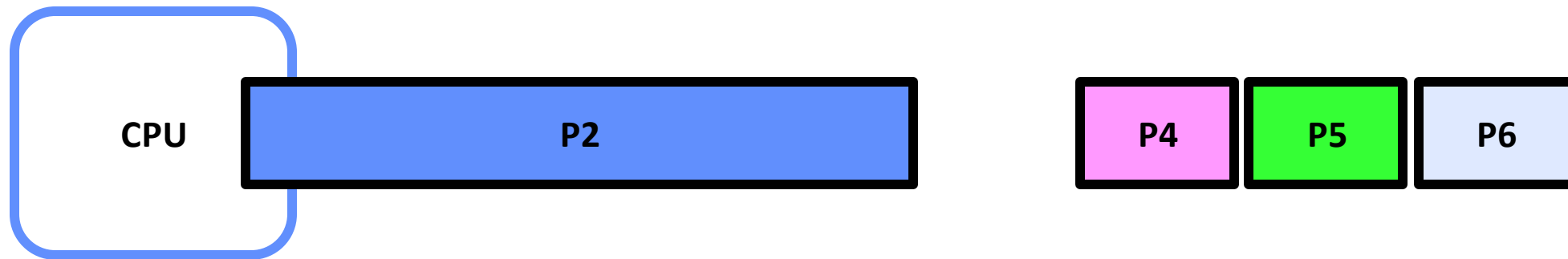


Are we done?

Is SJF subject to the convoy effect?

Yes

Any **non-preemptible** scheduling policy suffers from
convoy effect



SJF Summary

The good

Optimal Average Completion Time
when jobs arrive simultaneously

The bad

Still subject to convoy effect
when jobs arrive at different
times

The ugly

Can lead to starvation!

Requires knowing duration of job

Shortest Time to Completion First (STCF)

Introduce the notion of **preemption**

A running task can be de-scheduled before completion.

STCF

Schedule the task with the **least amount of time left**

Shortest Time to Completion First (STCF)

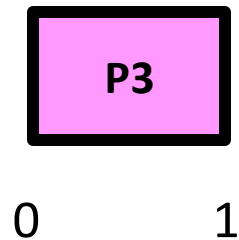
STCF

Schedule the task with the **least amount of time left**

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 3 | 10 |
| P_2 | 6 | 1 |
| P_3 | 24 | 0 |
| P_4 | 16 | 20 |

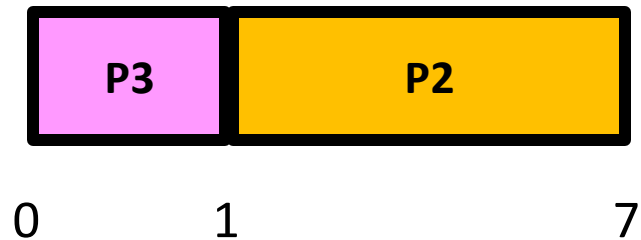
Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 3 | 10 |
| P_2 | 6 | 1 |
| P_3 | 24 | 0 |
| P_4 | 16 | 18 |



Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 3 | 10 |
| P_2 | 6 | 1 |
| P_3 | 23 | 0 |
| P_4 | 16 | 18 |



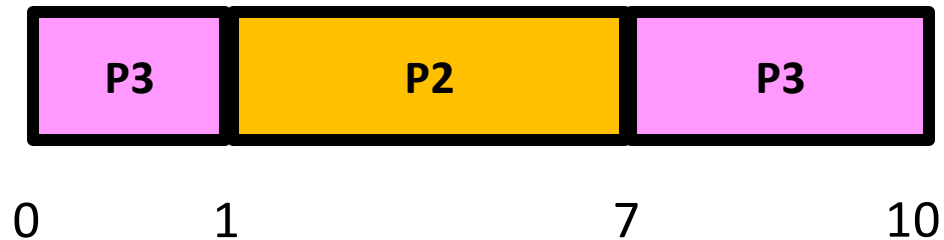
Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 3 | 10 |
| P_2 | 0 | 1 |
| P_3 | 23 | 0 |
| P_4 | 16 | 18 |



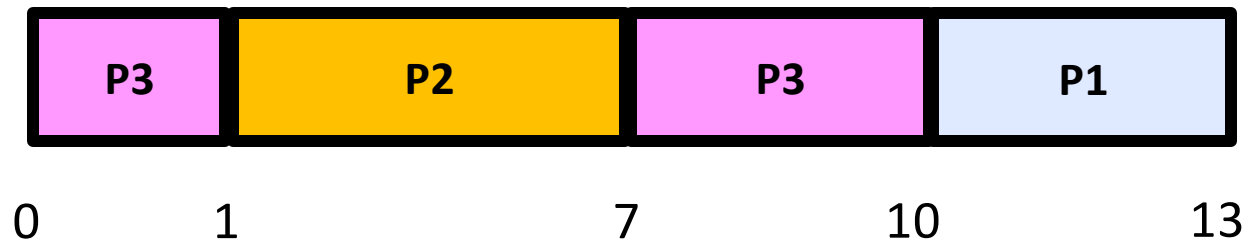
Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 3 | 10 |
| P_2 | 0 | 1 |
| P_3 | 20 | 0 |
| P_4 | 16 | 18 |



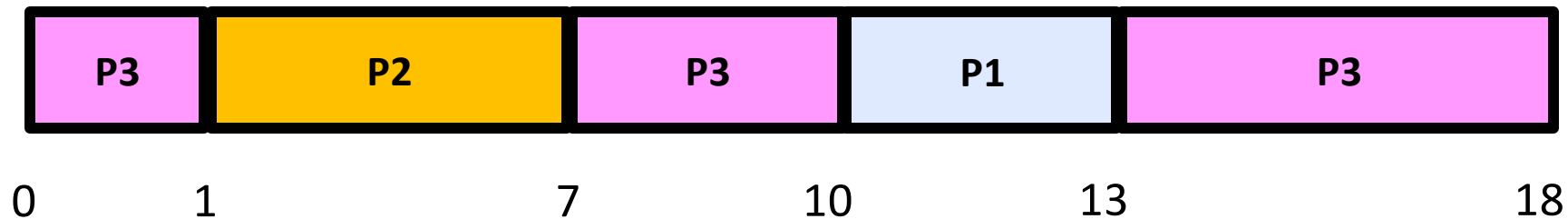
Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 0 | 10 |
| P_2 | 0 | 1 |
| P_3 | 20 | 0 |
| P_4 | 16 | 18 |



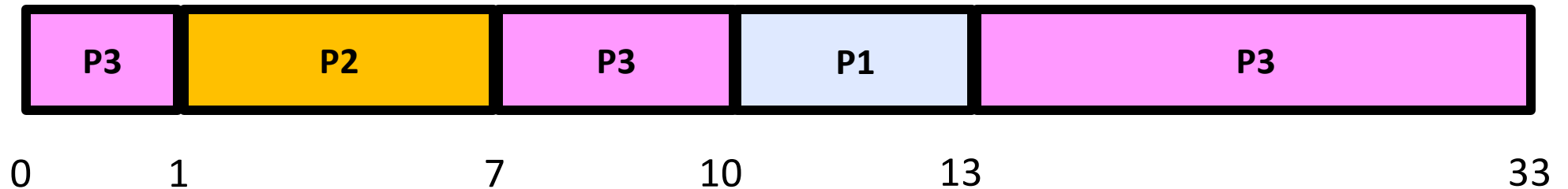
Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 0 | 10 |
| P_2 | 0 | 1 |
| P_3 | 15 | 0 |
| P_4 | 16 | 18 |



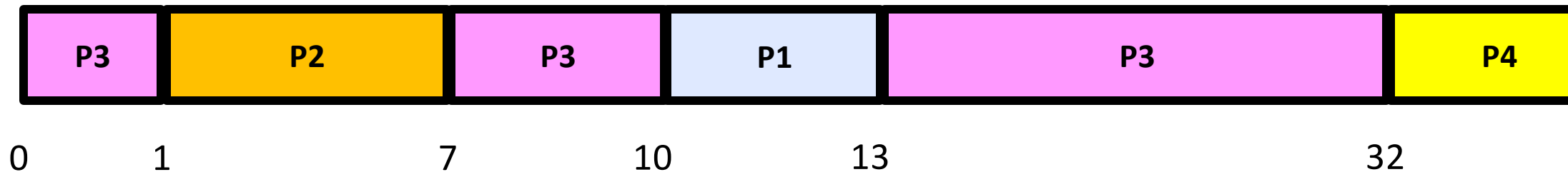
Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 0 | 10 |
| P_2 | 0 | 1 |
| P_3 | 0 | 0 |
| P_4 | 15 | 18 |



Shortest Time to Completion First (STCF)

| <u>Process</u> | <u>Burst Time (left)</u> | <u>Arrival Time</u> |
|----------------|--------------------------|---------------------|
| P_1 | 0 | 10 |
| P_2 | 0 | 1 |
| P_3 | 0 | 0 |
| P_4 | 15 | 18 |



Are we done?

Can STCF lead to starvation?

Yes

Any scheduling policy that always favours a **fixed property** for scheduling leads starvation

No change!

Are we done?

Is STCF subject to the convoy effect?

No!

STCF is a preemptible policy

STCF Summary

The good

Optimal Average Completion Time
Always

The bad

The ugly

Can lead to starvation!

Requires knowing duration of job

Taking a step back

| Property | FCFS | SJF | STCF |
|----------------------------------|------|-----|------|
| Optimise Average Completion Time | | ✓ | ✓ |
| Prevent Starvation | ✓ ★ | | |
| Prevent Convoy Effect | | | ✓ |
| Psychic Skills Not Needed | ✓ | | |

Can we design a non-psychic, starvation-free scheduler with good response time?

Round-Robin Scheduling

RR runs a job for a **time slice**
(a **scheduling quantum**)

Once time slice over,
switch to next job in ready queue.
=> Called **time-slicing**

RR with Time Quantum = 20

| <u>Process</u> | <u>Burst Time</u> |
|----------------|-------------------|
| P_1 | 53 |
| P_2 | 8 |
| P_3 | 68 |
| P_4 | 24 |

RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

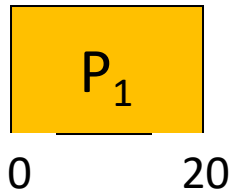
Burst Time

53 => 33

8

68

24



RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

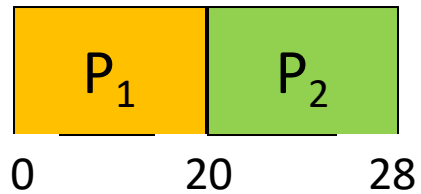
Burst Time

33

8 \Rightarrow 0

68

24



RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

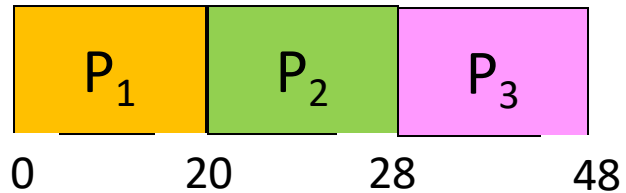
Burst Time

33

0

68 => 48

24



RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

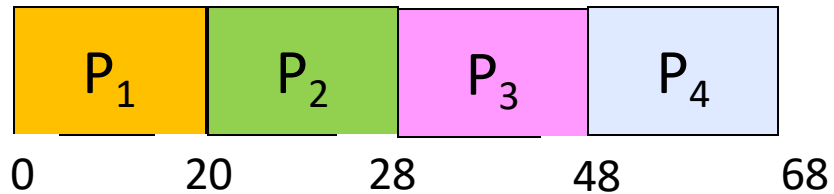
Burst Time

33

0

48

24 => 4



RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

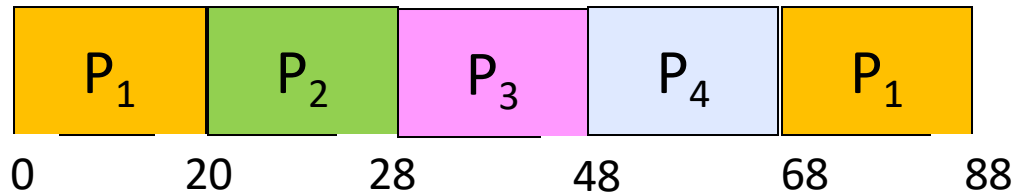
Burst Time

33 => 13

0

48

4



RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

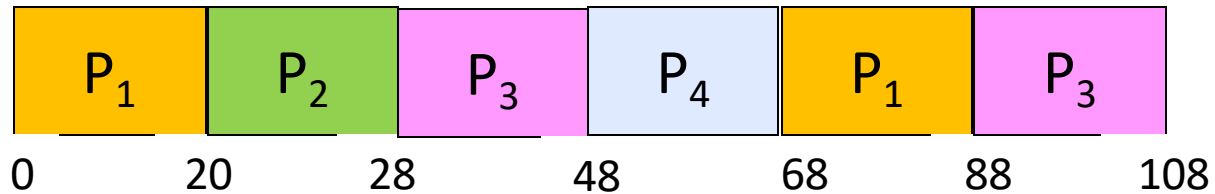
Burst Time

13

0

48 \Rightarrow 28

4



RR with Time Quantum = 20

Process

P_1

P_2

P_3

P_4

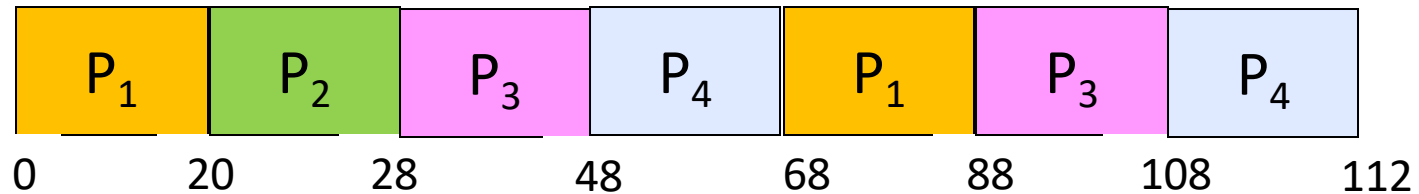
Burst Time

13

0

28

4 \Rightarrow 0



RR with Time Quantum = 20

Waiting time

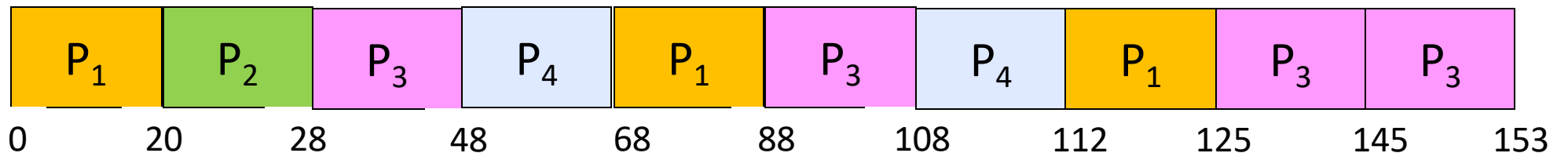
- $P_1 = 0 + (68-20) + (112-88) = 72$
- $P_2 = (20-0) = 20$
- $P_3 = (28-0) + (88-48) + (125-108) + 0 = 85$
- $P_4 = (48-0) + (108-68) = 88$

Average waiting time

$$\left(\frac{72+20+85+88}{4} = 66.25 \right)$$

Average completion time

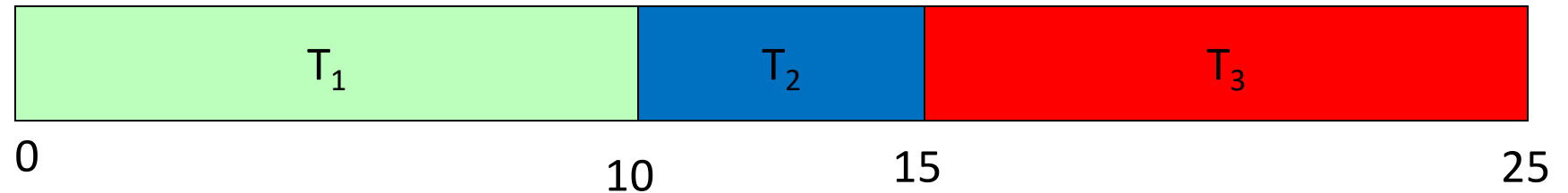
$$\left(\frac{125+28+153+112}{4} = 104.25 \right)$$



Decrease Completion Time

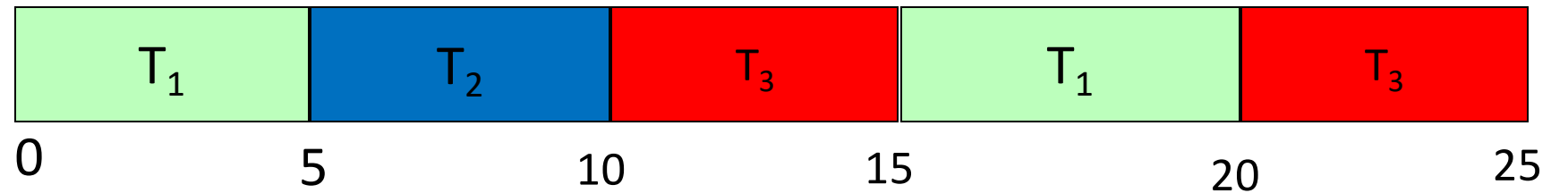
- T_1 : Burst Length 10 T_3 : Burst Length 10
- T_2 : Burst Length 5

$Q = 10$



Average Completion Time = $(10 + 15 + 25)/3 = 16.7$

$Q = 5$



Average Completion Time = $(20 + 10 + 25)/3 = 18.3$

Switching is not free!

Small scheduling quantas lead to
frequent context switches

- Mode switch overhead
- Trash cache-state

q must be large with respect to context switch,
otherwise overhead is too high

Are we done?

Can RR lead to starvation?

No

No process waits more than $(n-1)q$ time units

Are we done?

Can RR suffer from convoy effect?

No

Only run a time-slice at a time

RR Summary

The good

Bounded response time

The bad

Completion time can be high
(stretches out long jobs)

The ugly

Overhead of context switching

Taking a step back

| Property | FCFS | SJF | STCF |
|----------------------------------|------|-----|------|
| Optimise Average Completion Time | | ✓ | ✓ |
| Prevent Starvation | ✓ | | |
| Prevent Convoy Effect | | | ✓ |
| Psychic Skills Not Needed | ✓ | | |

Taking a step back

| Property | FCFS | SJF | STCF | RR |
|----------------------------------|------|-----|------|----|
| Optimise Average Completion Time | | ✓ | ✓ | |
| Optimise Average Response Time | | | | ✓ |
| Prevent Starvation | ✓ | | | ✓ |
| Prevent Convoy Effect | | | ✓ | ✓ |
| Psychic Skills Not Needed | ✓ | | | ✓ |

FCFS and Round Robin Showdown

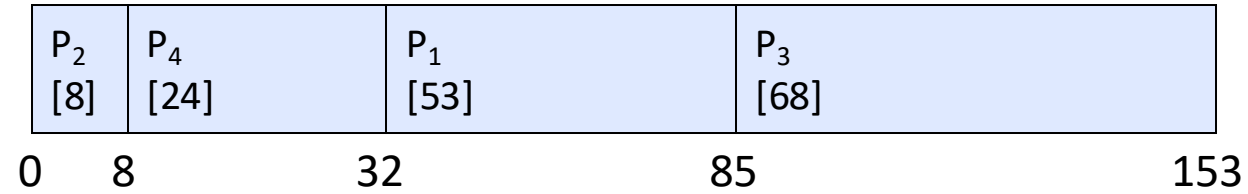
Assuming zero-cost context-switching time,
is RR always better than FCFS?

10 jobs, each take 100s of CPU time
RR scheduler quantum of 1s
All jobs start at the same time

| Job # | FIFO | RR |
|-------|------|------|
| 1 | 100 | 991 |
| 2 | 200 | 992 |
| ... | ... | ... |
| 9 | 900 | 999 |
| 10 | 1000 | 1000 |

Earlier Example with Different Time Quanta

Best FCFS:



| Scheduler | P1 | P2 | P3 | P4 | Average |
|------------|-----|-----|-----|-----|---------|
| Best FCFS | 85 | 8 | 16 | 32 | 69.5 |
| Worst FCFS | 121 | 153 | 68 | 145 | 121.75 |
| RR, Q=1 | 137 | 30 | 153 | 81 | 100.5 |
| RR, Q=5 | 135 | 28 | 153 | 82 | 99.5 |
| RR, Q=8 | 133 | 16 | 153 | 80 | 99,5 |
| RR, Q=10 | 135 | 18 | 153 | 92 | 104.5 |
| RR, Q=20 | 125 | 28 | 153 | 112 | 104.5 |