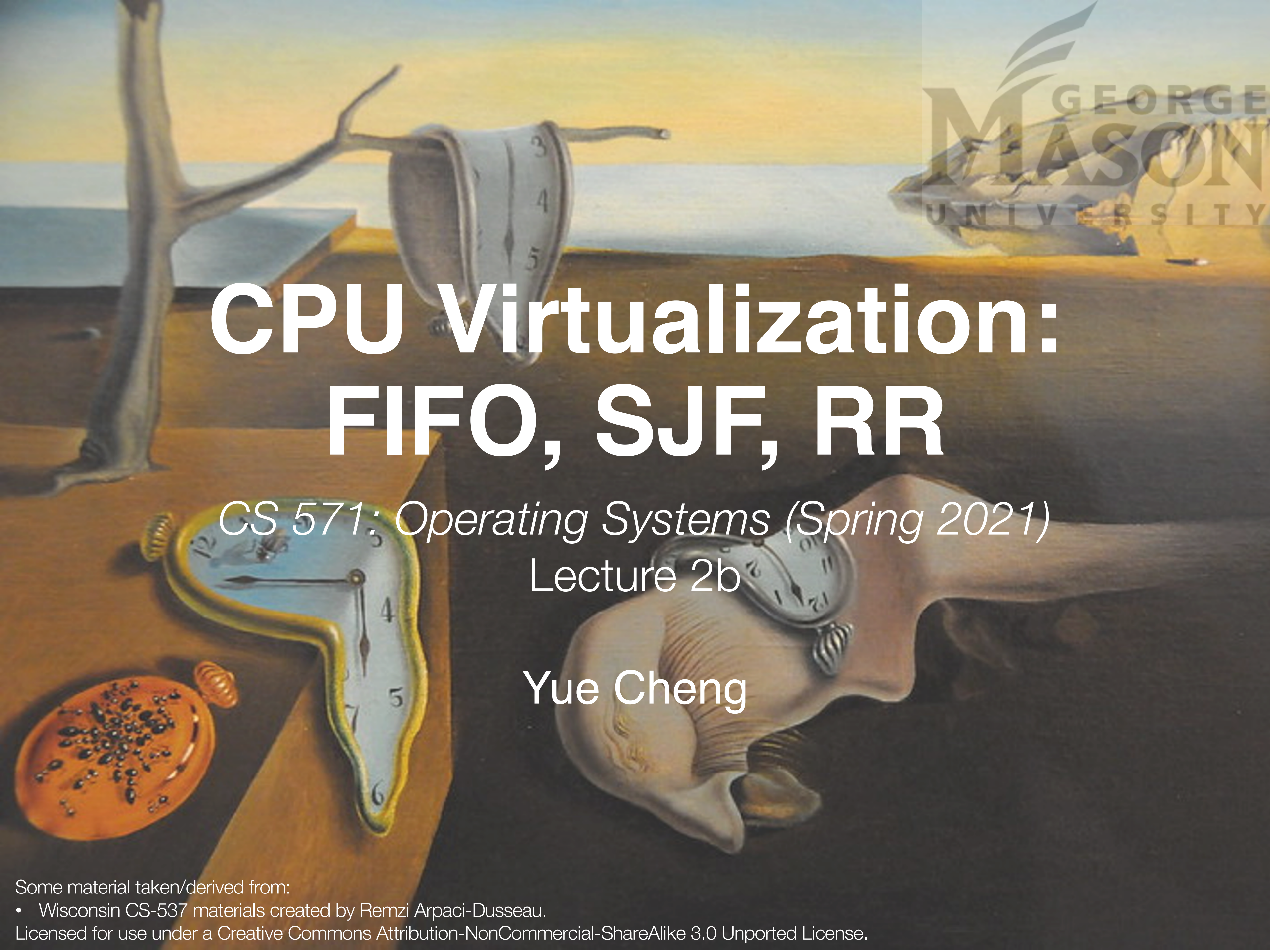


CPU Virtualization: FIFO, SJF, RR

CS 571: Operating Systems (Spring 2021)
Lecture 2b

Yue Cheng

CPU Scheduling: Outline

- Basic concept
- Scheduling criteria
- Scheduling algorithms
  - First In, First Out (FIFO)
  - Shortest Job First (SFJ)
  - Shortest Time-to-Completion First (STCF)
  - Round Robin (RR)
  - Priority
  - Multi-Level Feedback Queue (MLFQ)
  - Completely Fair Scheduler (CFS)

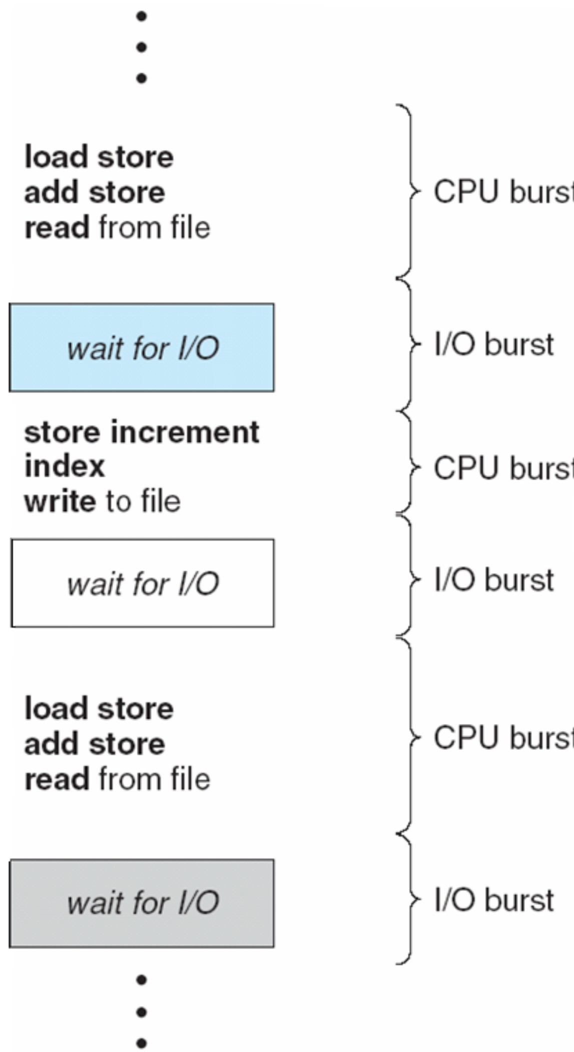
Basic Concepts

- During its lifetime, a process goes through a sequence of CPU and I/O bursts

- The CPU scheduler will select one of the processes in the ready queue for execution

- The CPU scheduler algorithm may have tremendous effects on the system performance
  - Interactive systems: Responsiveness
  - Real-time systems: Not missing the deadlines

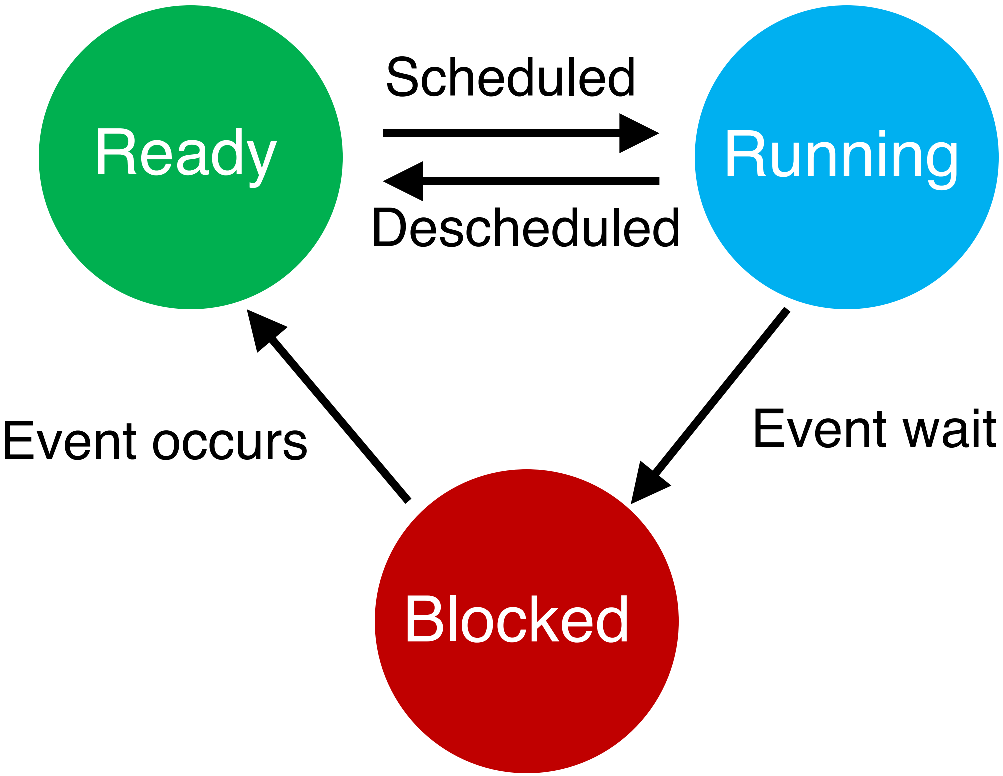
Alternating Sequence of CPU and I/O Bursts



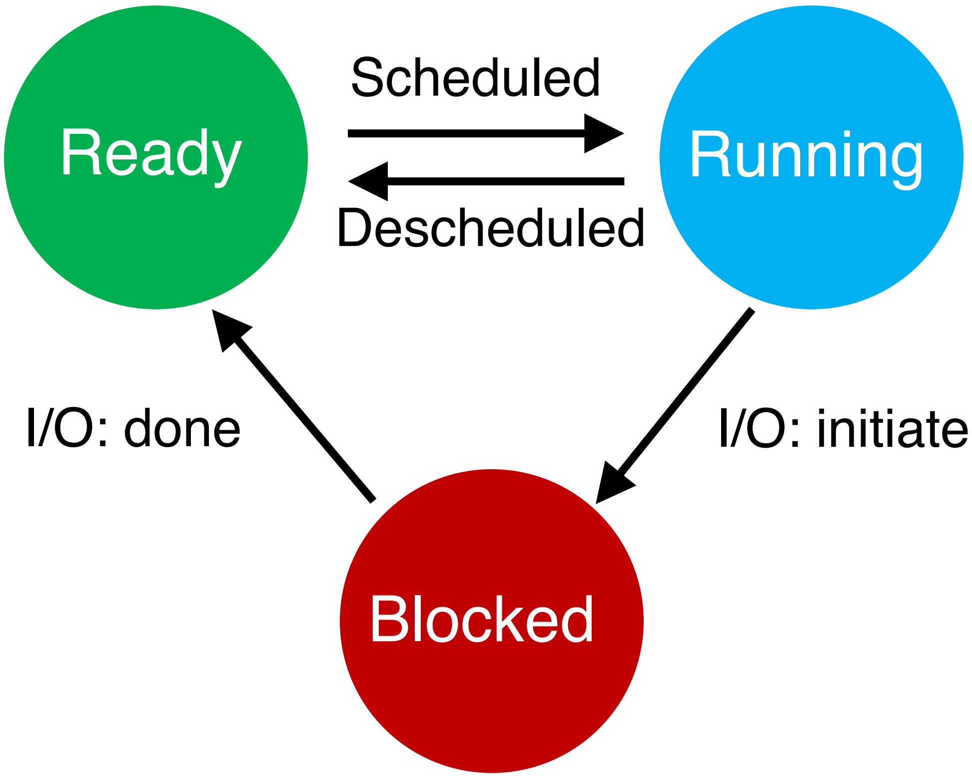
When to Schedule?

- Under the simple process state transition model, CPU scheduler can be potentially invoked at five different points:
  1. When a process switches from the new state to the ready state
  2. When a process switches from the running state to the waiting (or blocked) state
  3. When a process switches from the running state to the ready state
  4. When a process switches from the waiting state to the ready state
  5. When a process terminates

Process State Transitions



Process State Transitions



Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper (previously saved) location in the user program to restart that program

- Scheduler → Policy: When and how to schedule
- Dispatcher → Mechanism: Actuator following the commands of the scheduler

Scheduling Metrics

- To compare the performance of scheduling algorithms
  - CPU utilization – percentage of time CPU is busy executing jobs
  - Throughput – # of processes that complete their execution per time unit
  - Turnaround time – amount of time to execute a particular process
  - Waiting time – amount of time a process has been waiting in the ready queue or waiting for some event
  - Response time – amount of time it takes from when a request was submitted until the first response is produced, not the complete output

Optimization Goals

- To **maximize**:
  - Maximize the CPU utilization
  - Maximize the throughput

- To **minimize**:
  - Minimize the (average) turnaround time
  - Minimize the (average) waiting time
  - Minimize the (average) response time

Waiting Time

- Waiting time definition

$$T_{waiting} = T_{start} - T_{arrival}$$

- Average waiting time = Sum($T_{waiting}$)/ #processes

- For now, we assume
  - Average waiting time is the performance measure
  - Only one CPU burst (e.g., in milliseconds or ms) per process
  - Only CPU, No I/O
  - All processes arrive at the same time
  - Once started, each process runs to completion

First In, First Out (FIFO)

Workload 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. The run-time of each job is known

FIFO

- First-In, First-Out: Run jobs in arrival (time) order

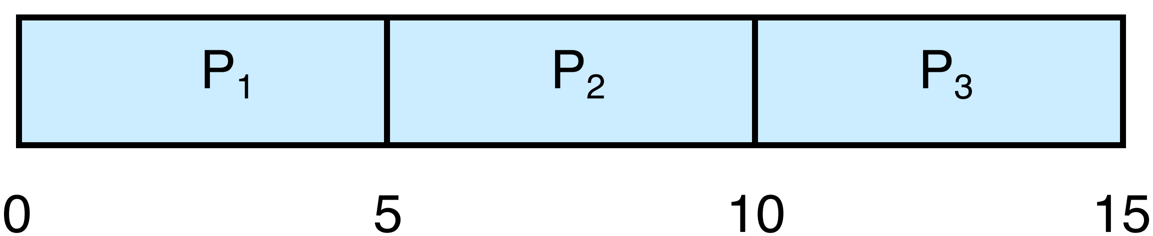
FIFO

First-In, First-Out: Run jobs in arrival (time) order

Def: waiting_time = start_time – arrival_time

Process	Burst Time
P_1	5
P_2	5
P_3	5

- Suppose that the processes arrive in order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule:



- Waiting time for P_1 = 0; P_2 = 5; P_3 = 10
- Average waiting time: 5

FIFO

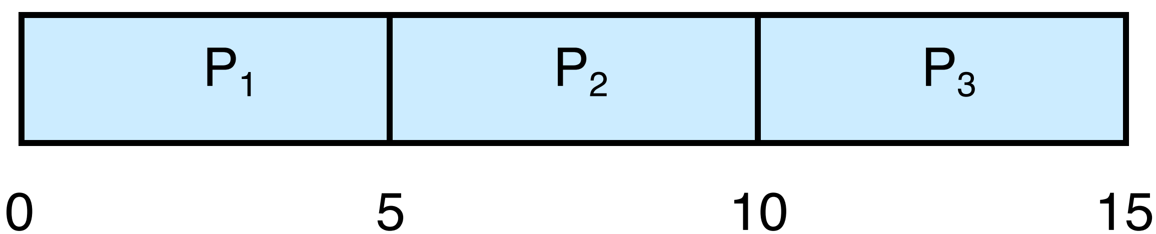
First-In, First-Out: Run jobs in arrival (time) order

What is the average turnaround time? (Q2)?

Def: turnaround_time = completion_time – arrival_time

Process	Burst Time
P_1	5
P_2	5
P_3	5

- Suppose that the processes arrive in order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule:



- Waiting time for $P_1 = 0$; $P_2 = 5$; $P_3 = 10$
- Average waiting time: 5

FIFO

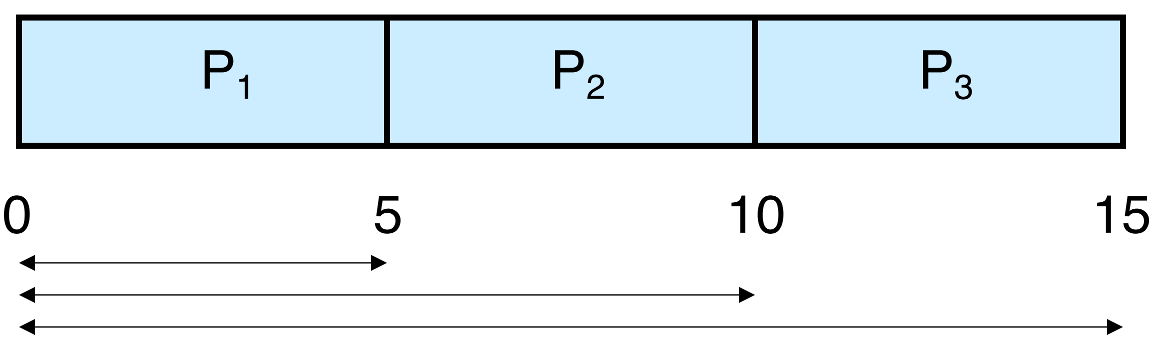
First-In, First-Out: Run jobs in arrival (time) order

What is the average turnaround time? (Q2)?

Def: turnaround_time = completion_time – arrival_time

Process	Burst Time
P_1	5
P_2	5
P_3	5

- Suppose that the processes arrive in order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule:



FIFO

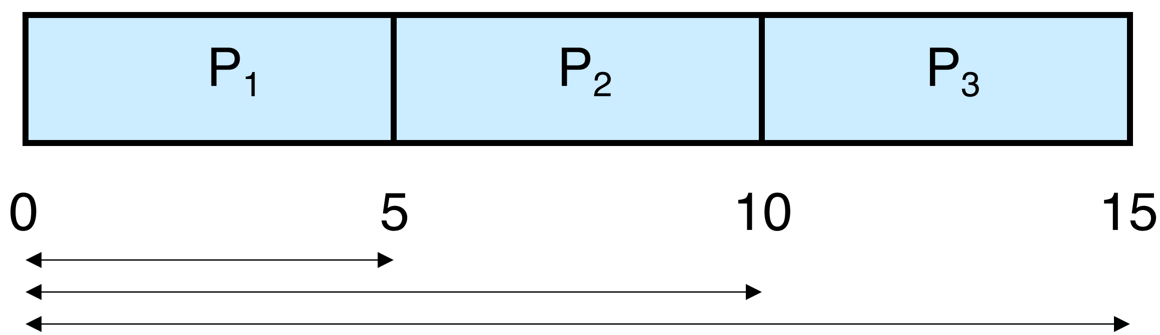
First-In, First-Out: Run jobs in arrival (time) order

What is the average turnaround time?

Def: turnaround_time = completion_time – arrival_time

Process	Burst Time
P_1	5
P_2	5
P_3	5

- Suppose that the processes arrive in order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule:



Average turnaround time: (5+10+15)/3 = 10

Workload 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. The run-time of each job is known

Workload 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. The run-time of each job is known

Example: Big First Job

JOB	arrival_time	run_time
P1	~0	80
P2	~0	5
P3	~0	5

What is the average turnaround time?

Example: Big First Job

JOB	arrival_time	run_time
P1	~0	80
P2	~0	5
P3	~0	5

Example: Big First Job

JOB	arrival_time	run_time
P1	~0	80
P2	~0	5
P3	~0	5

Average turnaround time: (80+85+90) / 3 = 85

Convoy Effect

Better Schedule?

Shortest Job First (SJF)

Passing the Tractor

- New scheduler: SJF (Shortest Job First)

- Policy: When deciding which job to run, choose the one with the smallest run_time

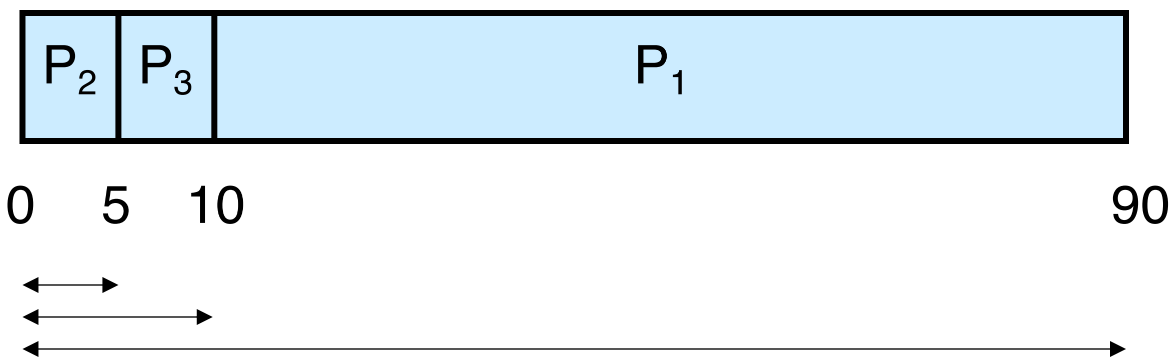
Example: SJF

JOB	arrival_time	run_time
P1	~0	80
P2	~0	5
P3	~0	5

What is the average turnaround time with SJF?

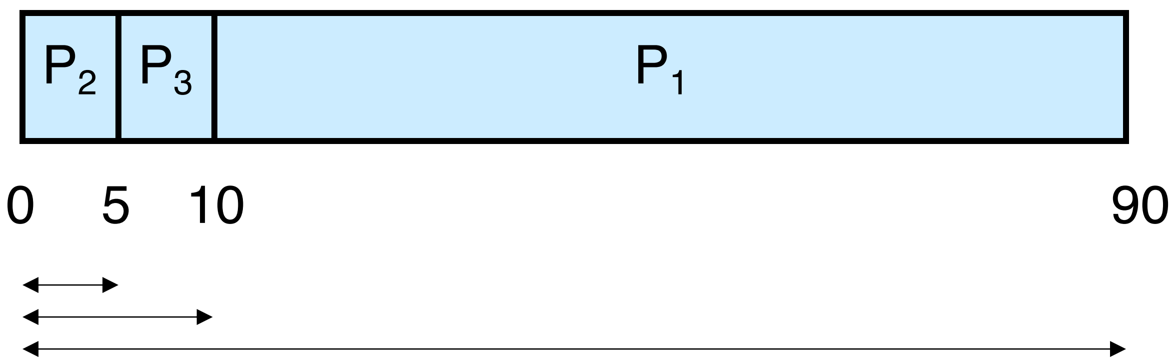
Example: SJF

JOB	arrival_time	run_time
P1	~0	80
P2	~0	5
P3	~0	5



Example: SJF

JOB	arrival_time	run_time
P1	~0	80
P2	~0	5
P3	~0	5



Average turnaround time: (5+10+90) / 3 = 35

Workload 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. The run-time of each job is known

Workload 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. The run-time of each job is known

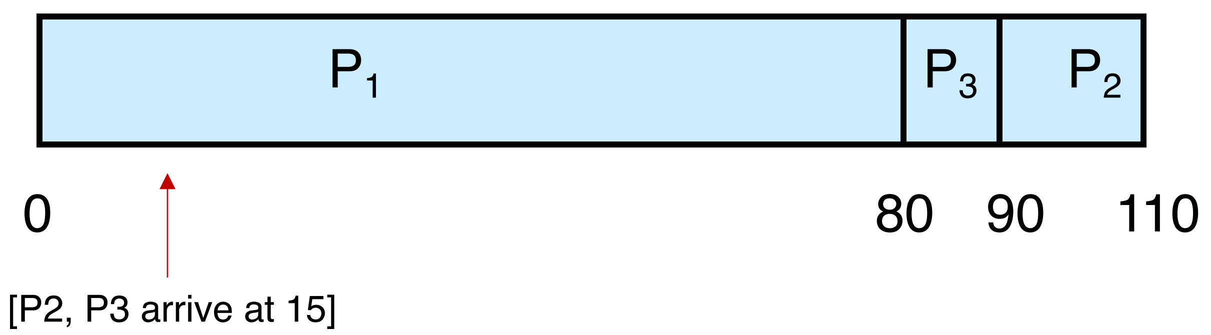
Shortest Job First (Arrival Time)

JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10

What is the average turnaround time with SJF?

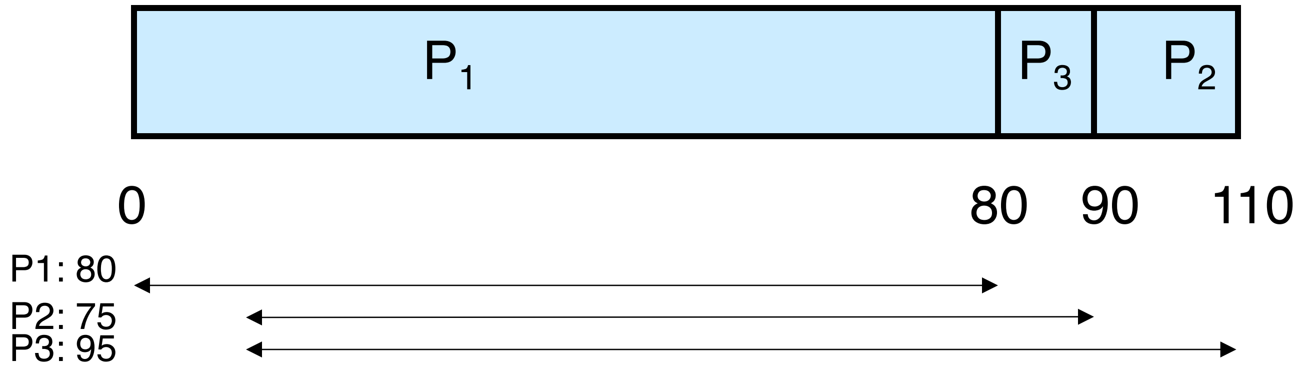
Shortest Job First (Arrival Time)

JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



Shortest Job First (Arrival Time)

JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



Average turnaround time: (80+75+95) / 3 = ~83.3

A Preemptive Scheduler

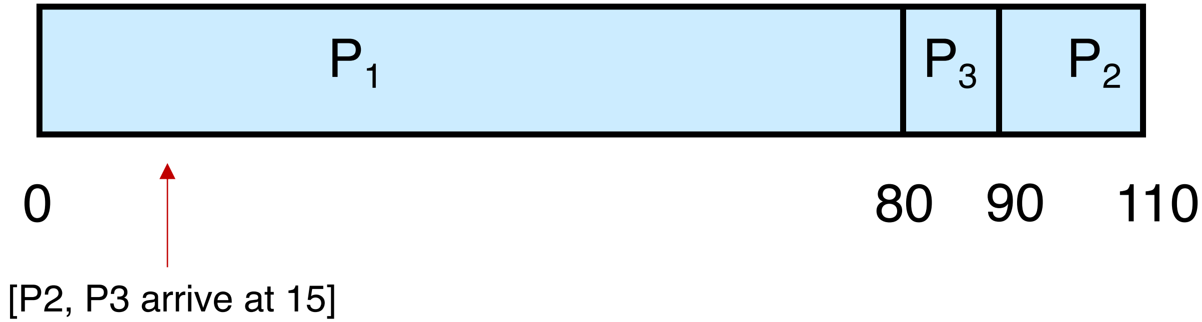
- Previous schedulers: FIFO and SJF are non-preemptive

- New scheduler: STCF (Shortest Time-to-Completion First)

- Policy: Switch jobs so we always run the one that will complete the quickest

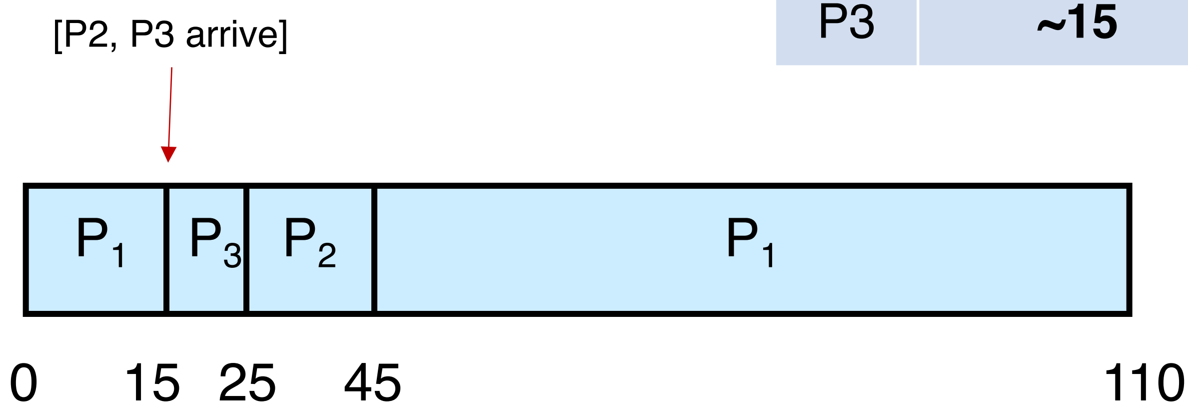
SJF

JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



STCF

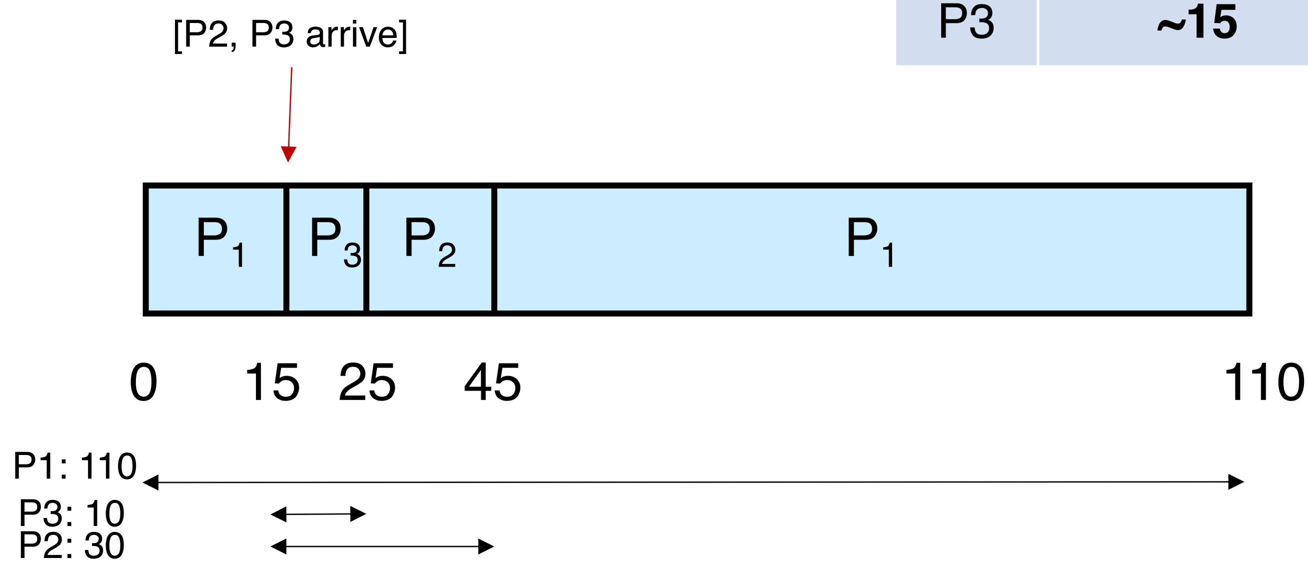
JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



What is the average turnaround time with STCF?

STCF

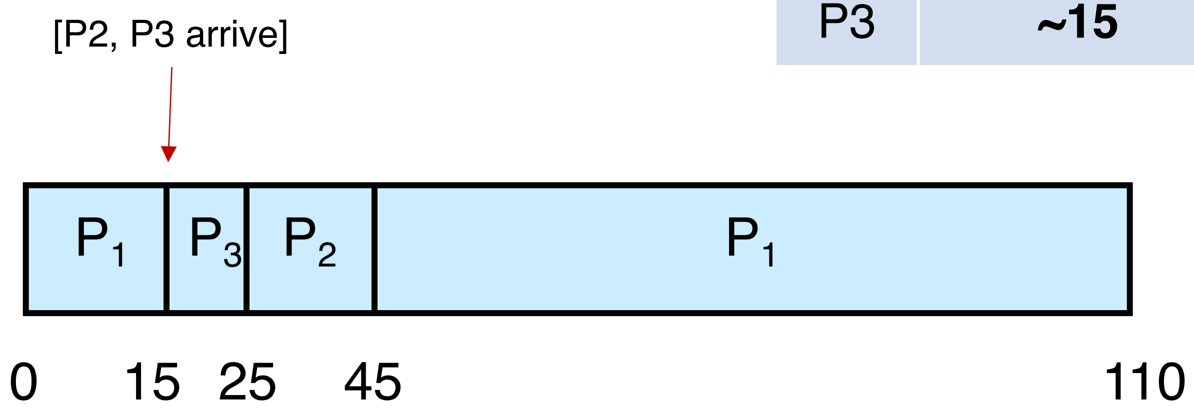
JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



Average turnaround time: (110+30+10) / 3 = 50

STCF

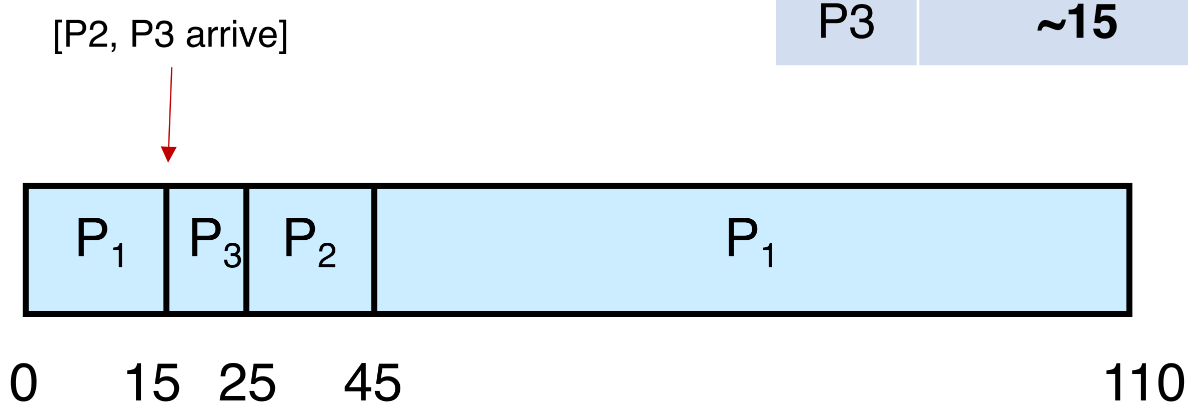
JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



What is the average waiting time with STCF?

STCF

JOB	arrival_time	run_time
P1	~0	80
P2	**~15**	20
P3	**~15**	10



P1: 30
P3: 0
P2: 10

Average waiting time: (30+10+0) / 3 = ~13.3

Optimality of SJF and STCF

- Non-preemptive SJF is optimal if all the processes are ready simultaneously
  - Gives minimum average waiting time for a given set of processes

Optimality of SJF and STCF

- Non-preemptive SJF is optimal if all the processes are ready simultaneously
  - Gives minimum average waiting time for a given set of processes

- What is the intuition behind the optimality of STCF?

Optimality of SJF and STCF

- Non-preemptive SJF is optimal if all the processes are ready simultaneously
  - Gives minimum average waiting time for a given set of processes

- What is the intuition behind the optimality of STCF?
  - A: STCF is optimal, considering a more realistic scenario where all the processes may be arriving at different times

Optimality of SJF and STCF

- Non-preemptive SJF is optimal if all the processes are ready simultaneously
  - Gives minimum average waiting time for a given set of processes

Q: What’s the problem?

We don’t know how long a job would run!

- What is the intuition behind the optimality of SRTF?
  - A: SRTF is optimal, considering a more realistic scenario where all the processes may be arriving at different times

Estimating the Length of Next CPU Burst

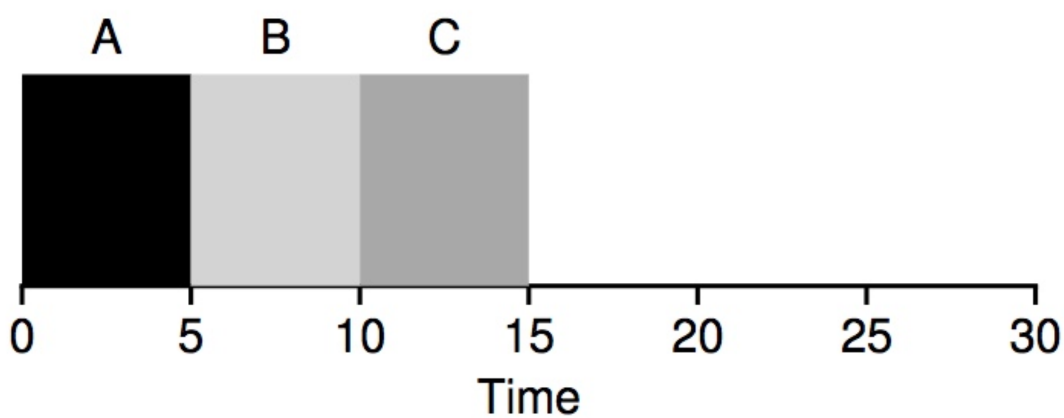
- Idea: Based on the observations in the recent past, we can try to predict
- Techniques such as exponential averaging are based on combining the observations in the past and our predictions using different weights
- Exponential averaging
  - t_n: actual length of the n^{th} CPU burst
  - z_{n+1}: predicted value for the next CPU burst
  - $z_{n+1} = k.t_n + (1-k).z_n$
  - Commonly, `k` is set to ½

Response Time

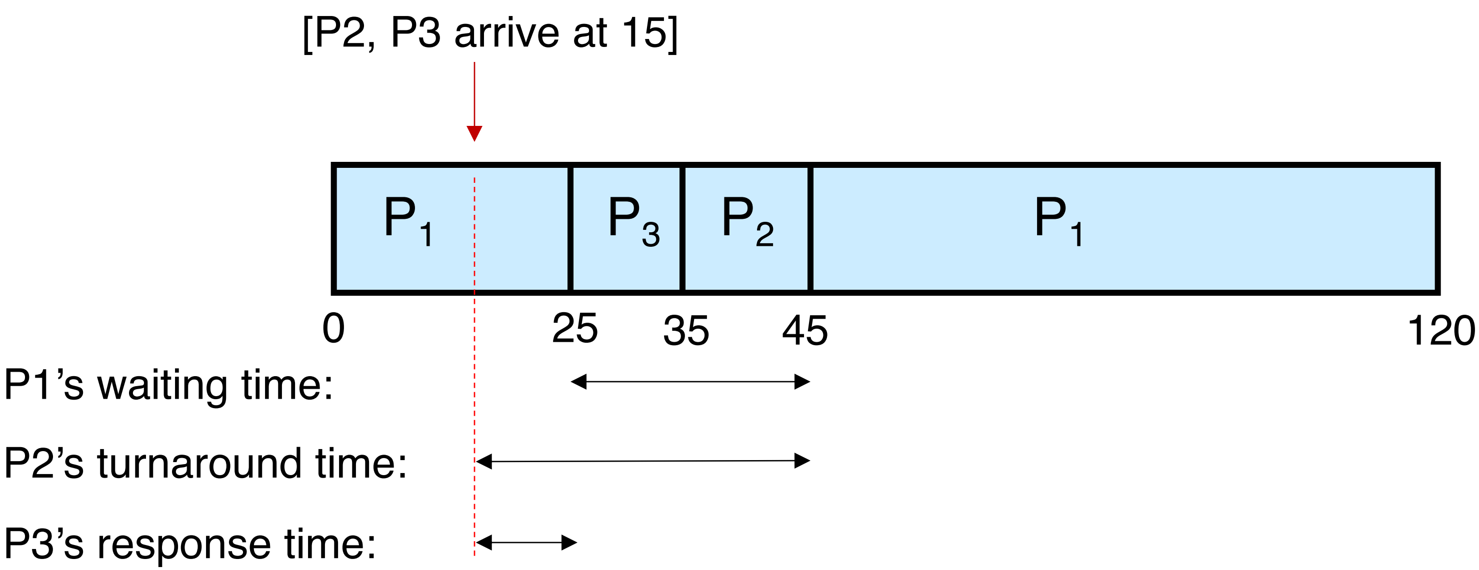
- Response time definition

$$T_{response} = T_{first\_run} - T_{arrival}$$

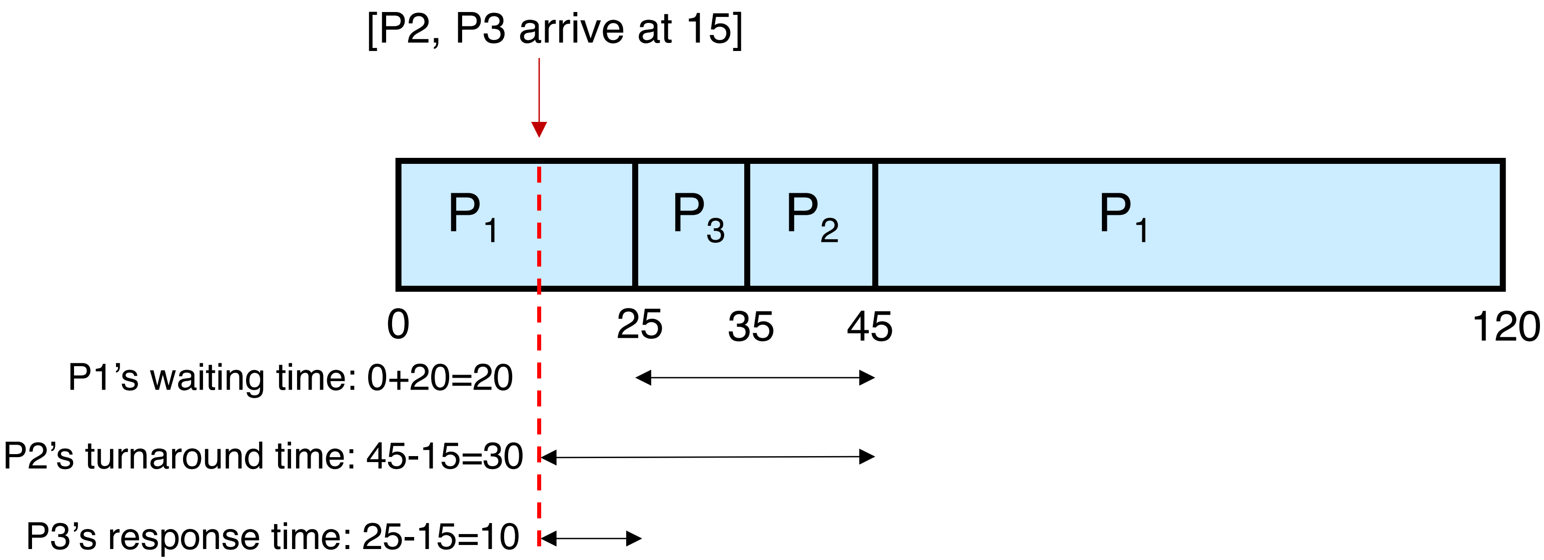
- SJF's average response time (all 3 jobs arrive at same time)
  - (0 + 5 + 10)/3 = 5



Waiting, Turnaround, Response



Waiting, Turnaround, Response



Q: What is P1's response time?

Round Robin (RR)

Workload 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. The run-time of each job is known

Workload 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. The run-time of each job is known

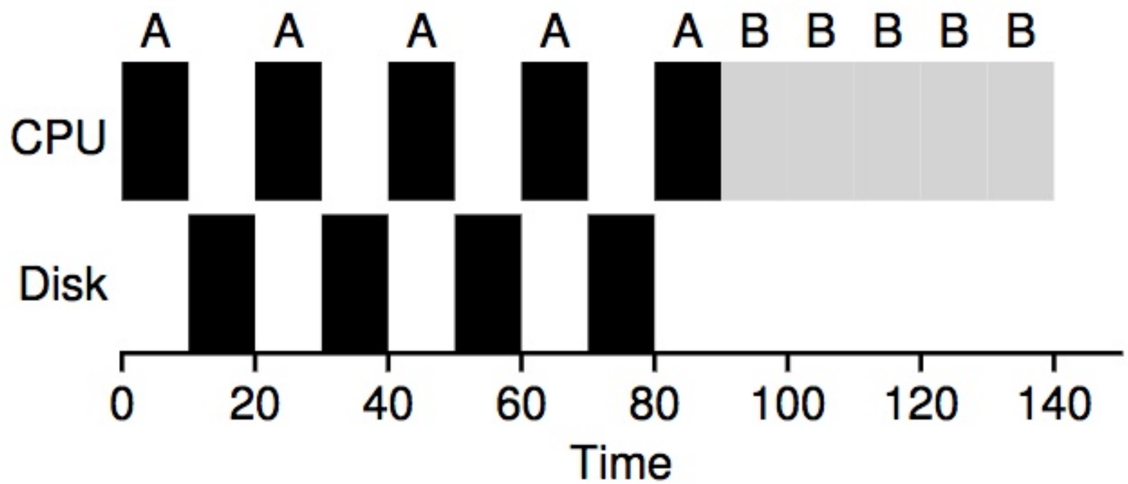
Extension to Multiple CPU & I/O Bursts

- When the process arrives, it will try to execute its first CPU burst
  - It will join the ready queue
  - The priority will be determined according to the underlying scheduling algorithm and considering only that specific (i.e. first) burst
- When it completes its first CPU burst, it will try to perform its first I/O operation (burst)
  - It will join the device queue
  - When that device is available, it will use the device for a time period indicated by the length of the first I/O burst.
- Then, it will re-join the ready queue and try to execute its second CPU burst
  - Its new priority may now change (as defined by its second CPU burst)!

Round Robin (RR)

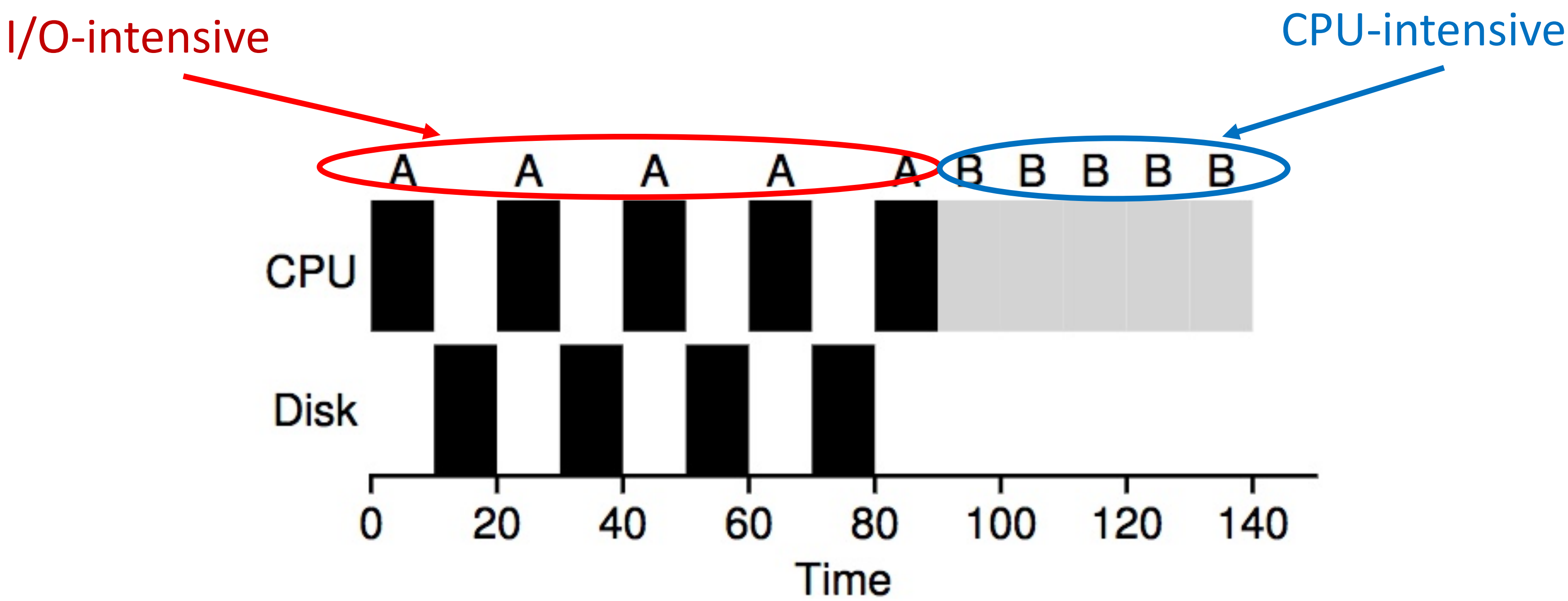
- Each process gets a small unit of CPU time (time quantum). After this time has elapsed, the process is preempted and added to the end of the ready queue
- Newly-arriving processes (and processes that complete their I/O bursts) are added to the end of the ready queue
- If there are n processes in the ready queue and the time quantum is q, then no process waits more than (n-1)q time units
- Performance
  - q large ⇒ FIFO
  - q small ⇒ Processor Sharing (The system appears to the users as though each of the n processes has its own processor running at the $(1/n)^{th}$ of the speed of the real processor)

Not I/O Aware



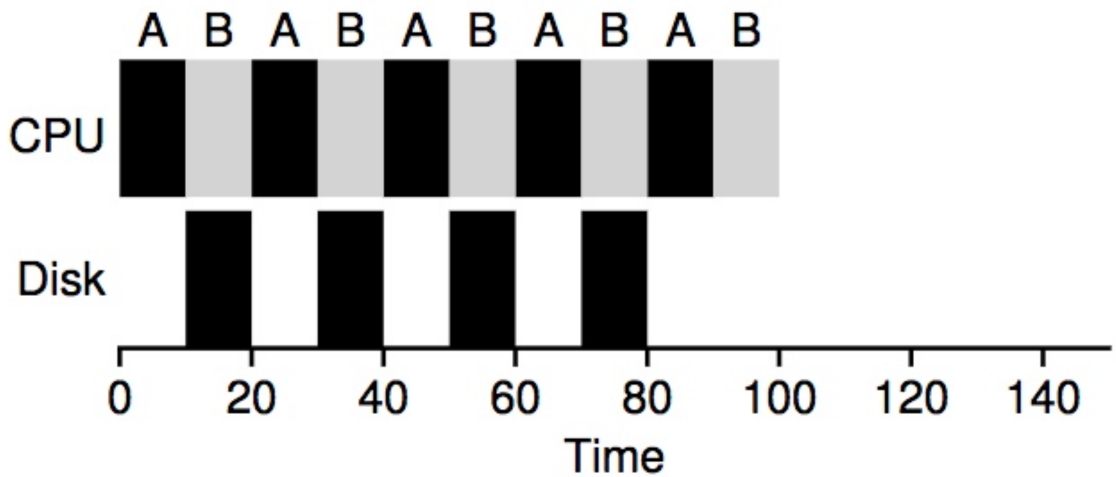
Poor use of resources

Not I/O Aware



Poor use of resources

I/O Aware (Overlap)

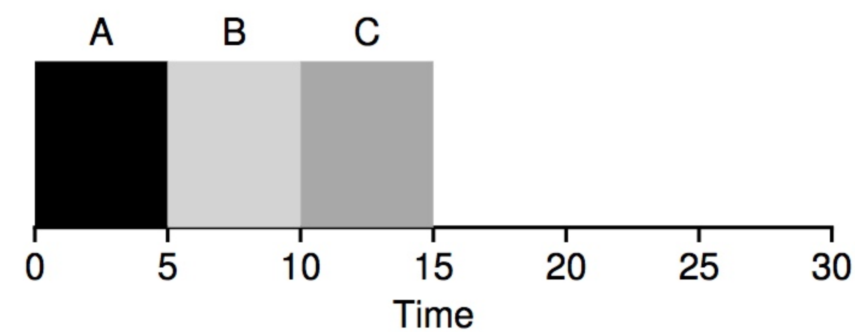


Overlap allows better use of resources!

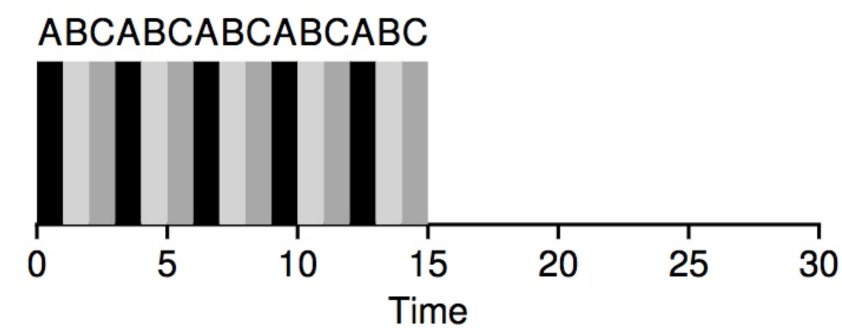
RR

Process	Burst Time
A	5
B	5
C	5

- SJF's average response time
  - (0 + 5 + 10) / 3 = 5



- RR's average response time (time quantum = 1)
  - (0 + 1 + 2) / 3 = 1



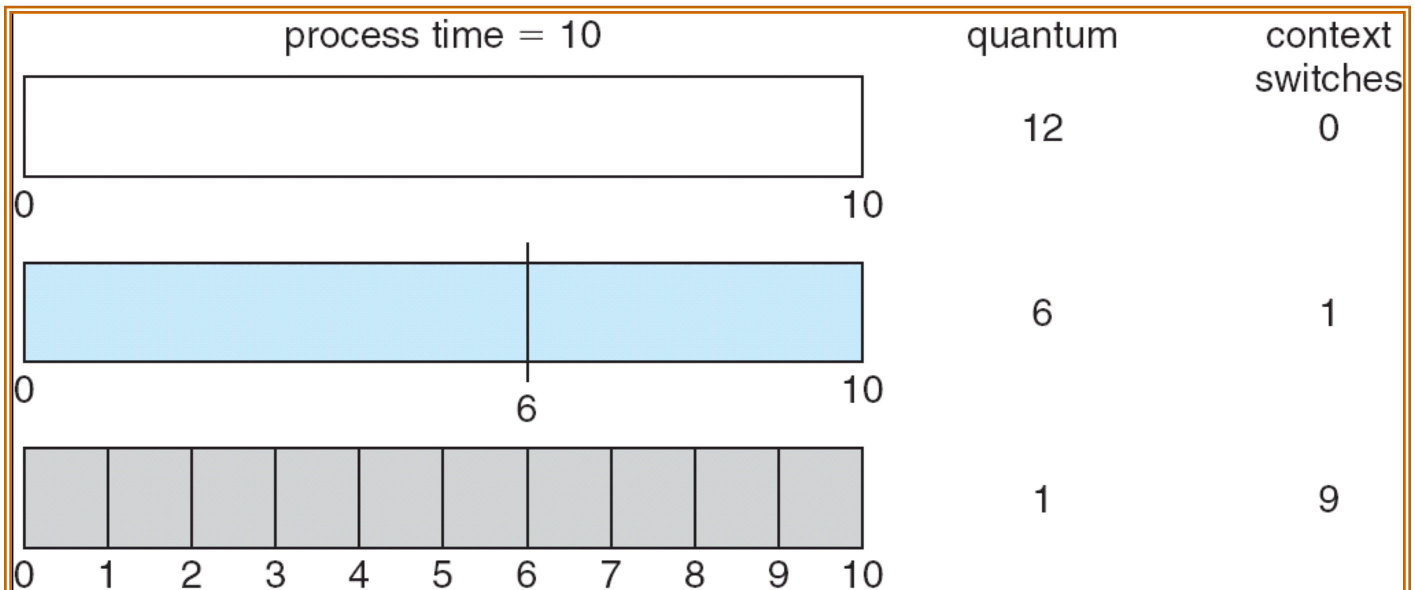
Tradeoff Consideration

- Typically, RR achieves higher average turnaround time than SJF, but better response time
  - Turnaround time only cares about when processes finish

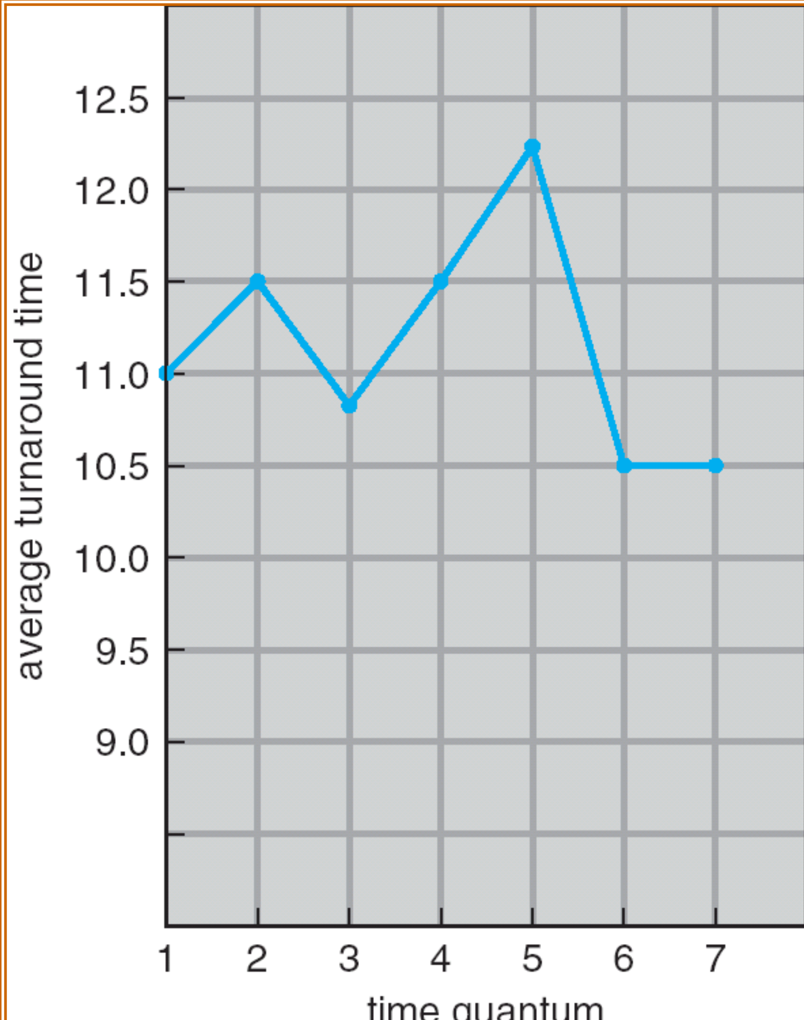
- RR is one of the worst policies
  - if turnaround time is the metric

Choosing a Time Quantum

- The effect of quantum size on context-switching time must be carefully considered
- The time quantum must be large with respect to the context-switch time
- Turnaround time also depends on the size of the time quantum

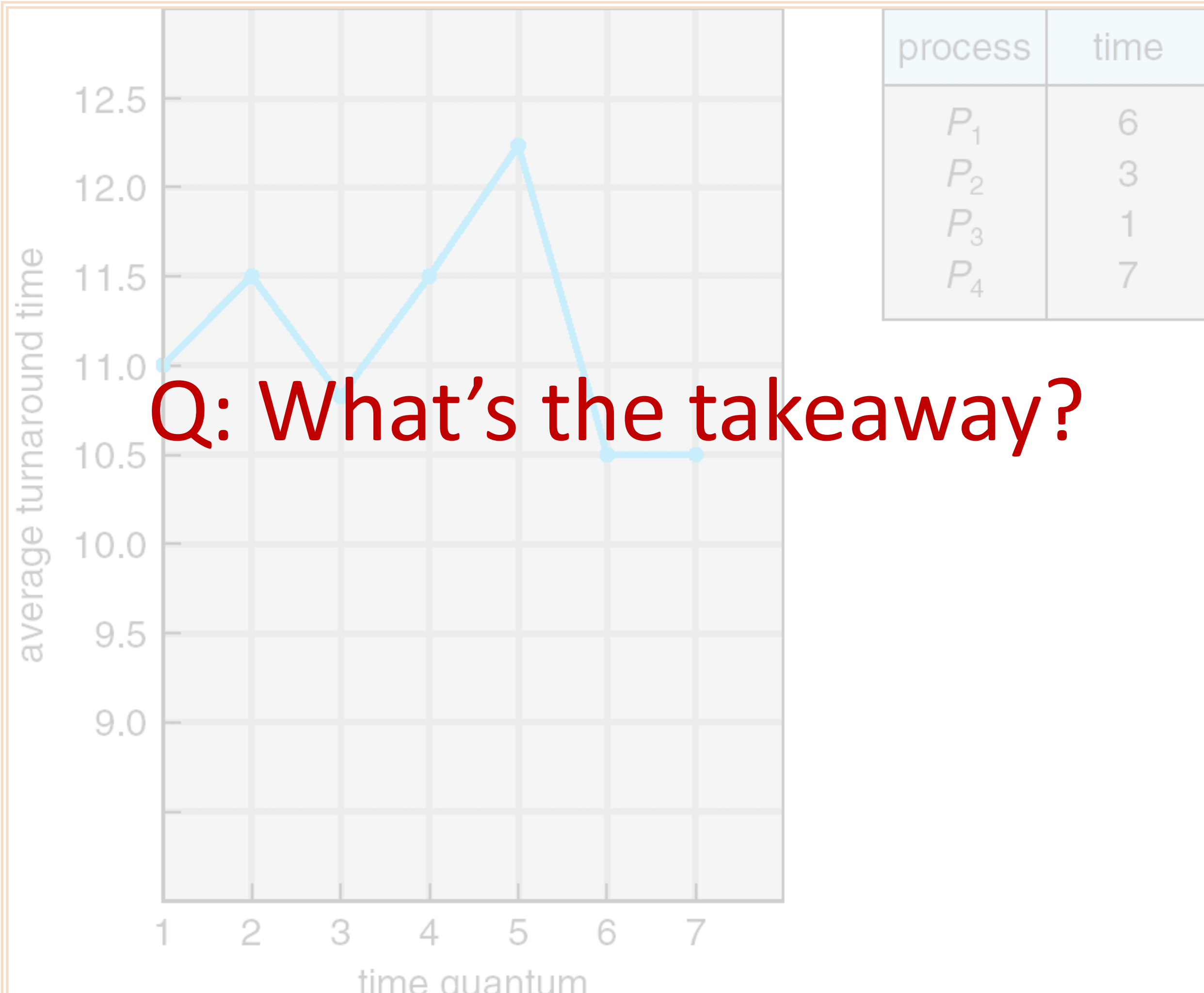


Time Quantum vs. Turnaround Time



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Time Quantum vs. Turnaround Time



Q: What's the takeaway?