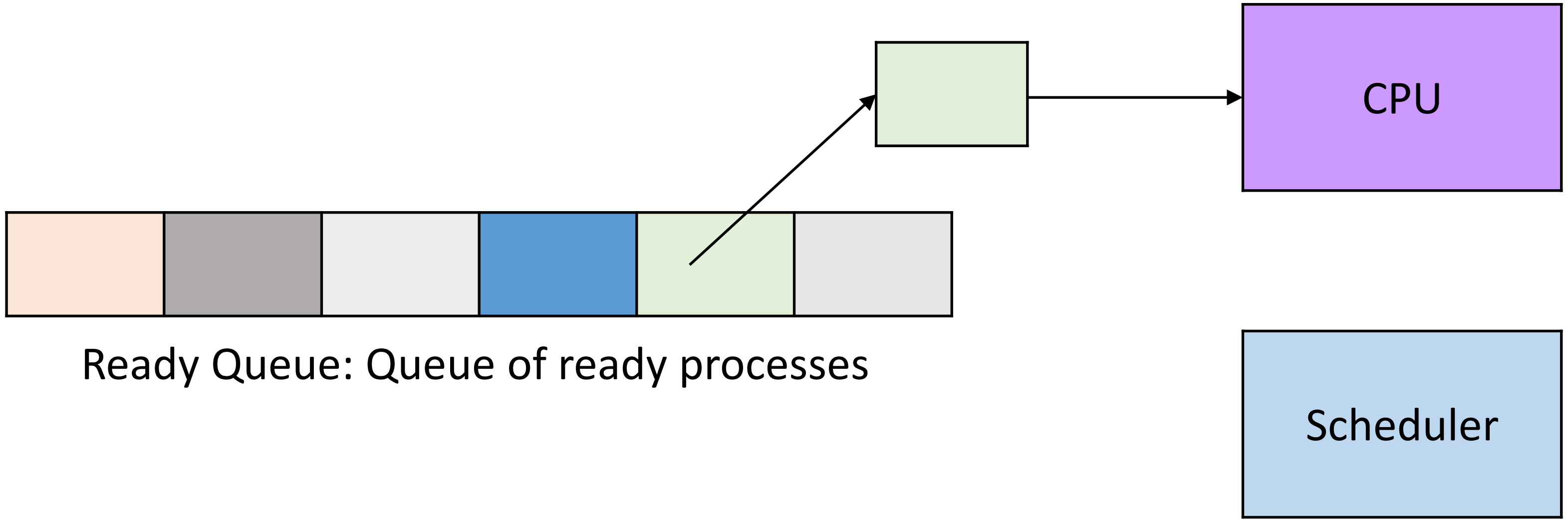
Scheduling

Scheduling



Ready Queue: Queue of ready processes

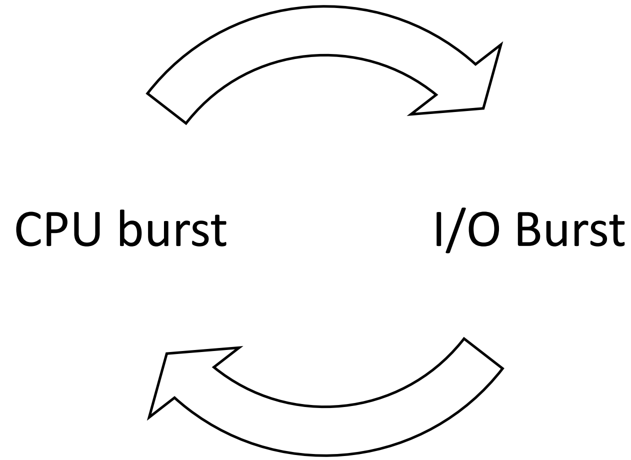
Question: How the OS decides which thread should run on cpu next?

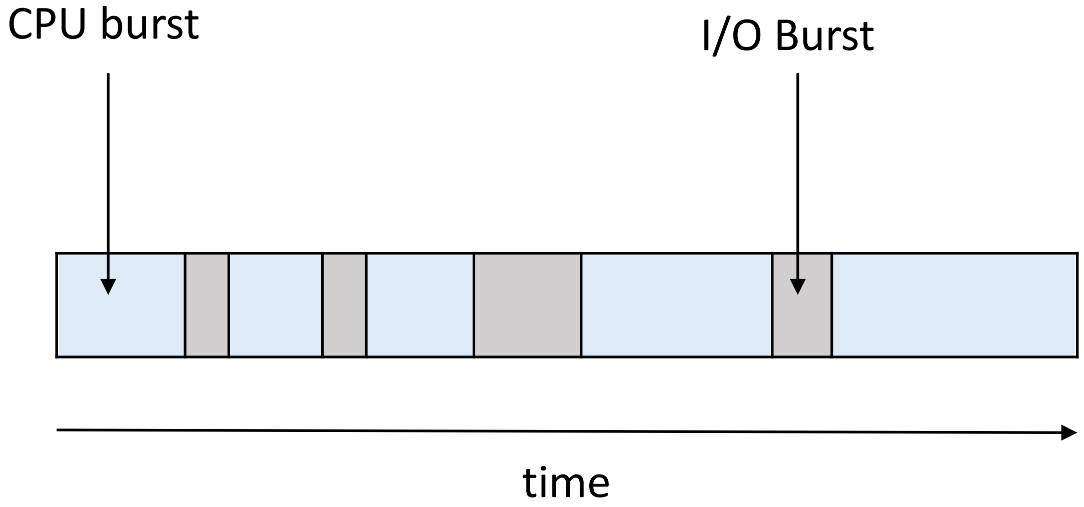
Scheduling: deciding which threads are given access to cpu from time to time

Execution Phases of a Process

CPU Burst: the process uses the processor to execute certain instructions before it is no longer ready

I/O Burst: when a process enters the blocked state waiting for I/O





Scheduling Policies

- Minimize waiting time
  - Processes should not wait long in the ready queue
- Maximize CPU utilization
  - CPU should not be idle
- Maximize throughput
  - Complete as many processes as possible per unit time
- Minimize response time
  - CPU should respond immediately
- Fairness
  - Give each process a fair share of CPU

Scheduling Policies

Waiting time for process p*:* $\sum$ time before p gets scheduled

Average waiting time: Average of all processes’ waiting time.

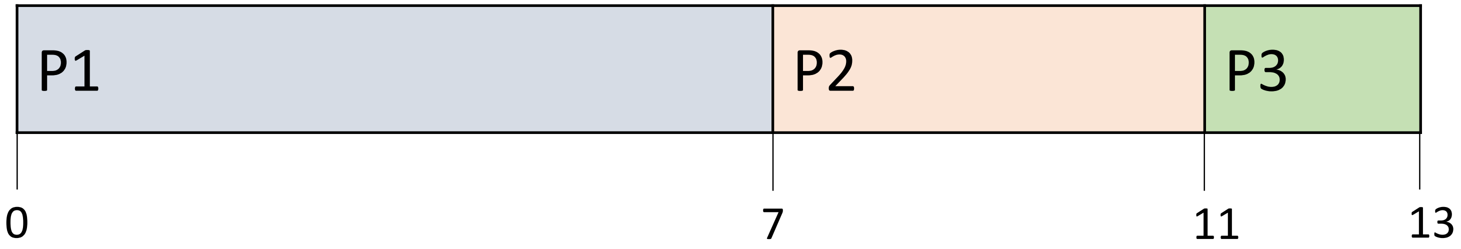
Completion time (response time): waiting time + run time /
time of completeion – time of arrival

Average completion time (response time): Average of all processes' completion time

First-Come, First-Served (FCFS) Scheduling

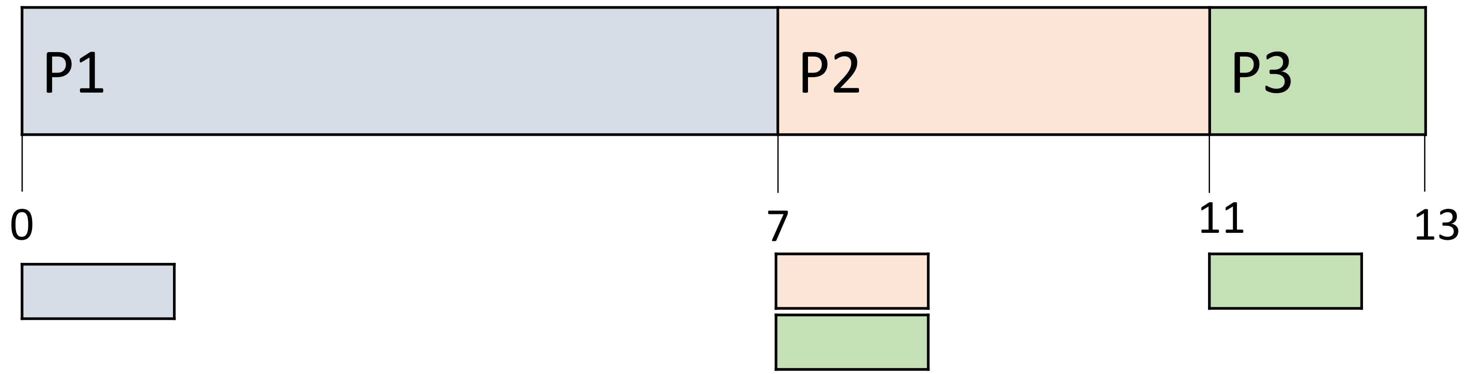
- <u>First-Come First-Served (FCFS)</u>
  - Process that requests the CPU FIRST is allocated the CPU FIRST.
  - Also called FIFO (First in First Out) “Run until done”
    - In early systems, FCFS meant one program scheduled until done (including I/O)

Proce ss	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2



First-Come, First-Served (FCFS) Scheduling

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2

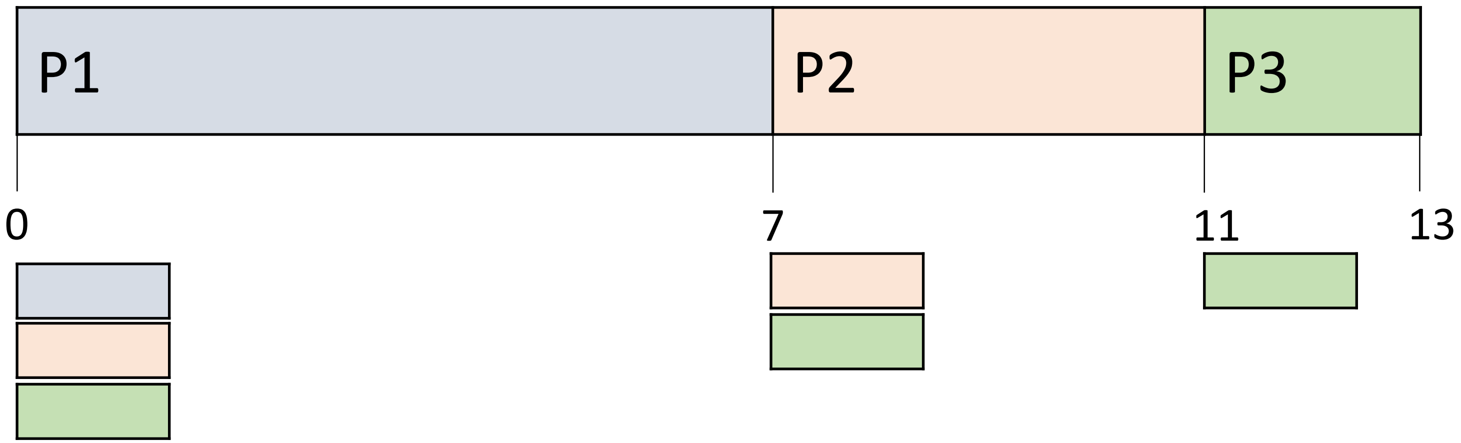


waiting time for P1 =
waiting time for P2 =
waiting time for P3 =
average waiting time =
average completion time =

completion time for P1 =
completion time for P2 =
completion time for P3 =
average completion time =

First-Come, First-Served (FCFS) Scheduling

Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	2

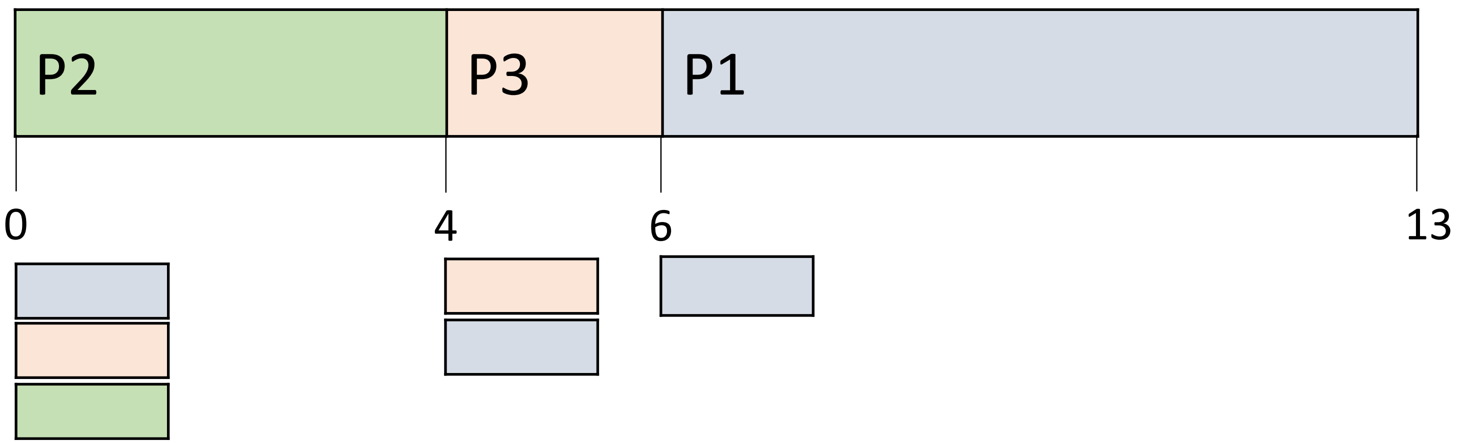


waiting time for P1 =
waiting time for P2 =
waiting time for P3 =
average waiting time =
average completion time =

completion time for P1 =
completion time for P2 =
completion time for P3 =
average completion time =

First-Come, First-Served (FCFS) Scheduling

Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	2



waiting time for P1 =
waiting time for P2 =
waiting time for P3 =
average waiting time =
average completion time =

completion time for P1 =
completion time for P2 =
completion time for P3 =
average completion time =

First-Come, First-Served (FCFS) Scheduling

- Pros and Cons

  \+ simple – Just a FIFO queue

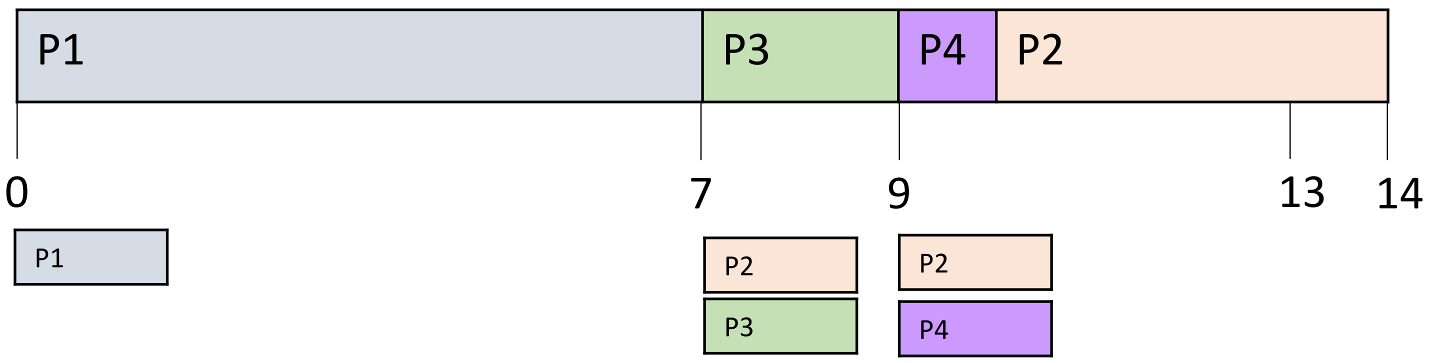
  – waiting time depends on arrival order. Bad for short processes

  – short processes get stuck waiting for long process to complete (convoy effect)

Shortest Job First (SJF)

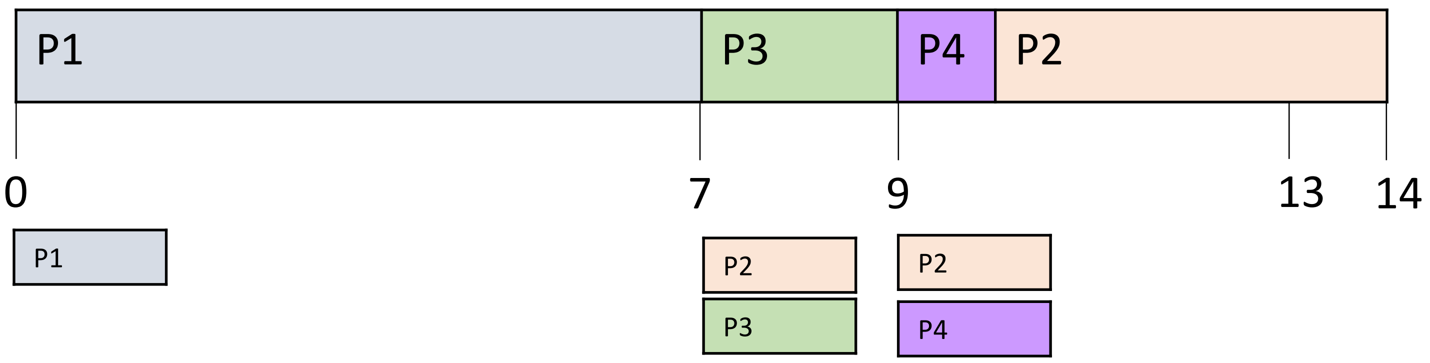
- Shortest Job First (SJF)
  - schedule the process that has the shortest burst time
    - Randomly pick one if all the processes have the same burst time.
  - sometimes called shortest time to completion first
  - **No preemption**. Once a process is holding a resource (i.e. once its request has been granted), then that resource cannot be taken away from that process until the process voluntarily releases it.

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



Shortest Job First (SJF)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



waiting time for P1 =
waiting time for P2 =
waiting time for P3 =
waiting time for P4 =
average waiting time =

completion time for P1 =
completion time for P2 =
completion time for P3 =
completion time for P4 =
average completion time =

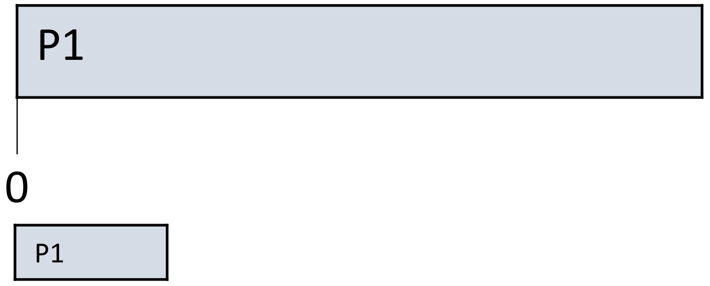
Shortest Remaining Time First (SRTF)

- Shortest Remaining Time First (SRTF)/ SJF with preemption
  - If a new process arrives with a shorter burst time than the remaining of the process then schedule the new process

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1

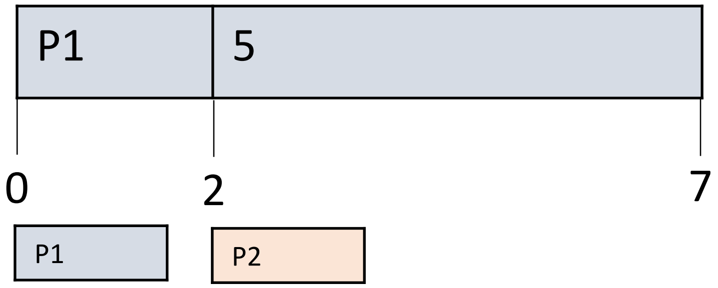
Shortest Remaining Time First (SRTF)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



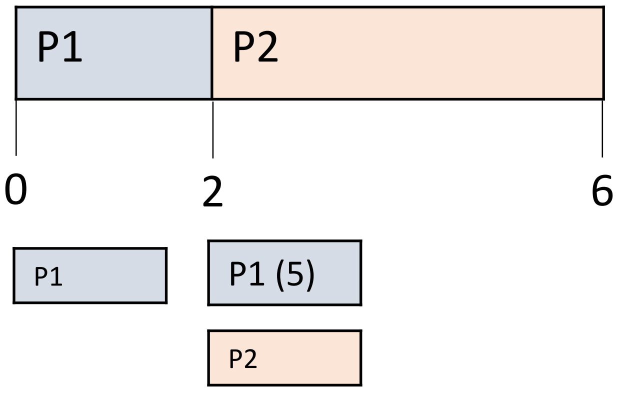
P2 arrives
CPU burst of P2 < remaining time of P1
Hence, preempt P1

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1

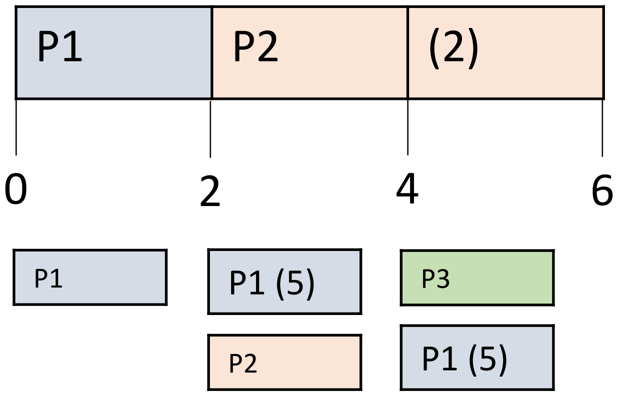


Shortest Remaining Time First (SRTF)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



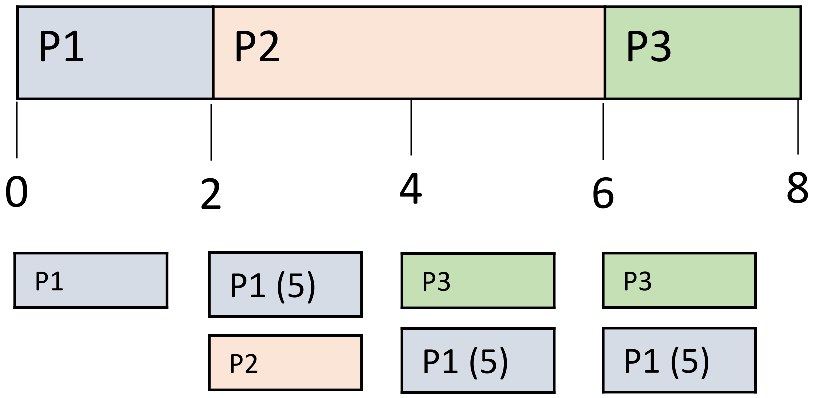
Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



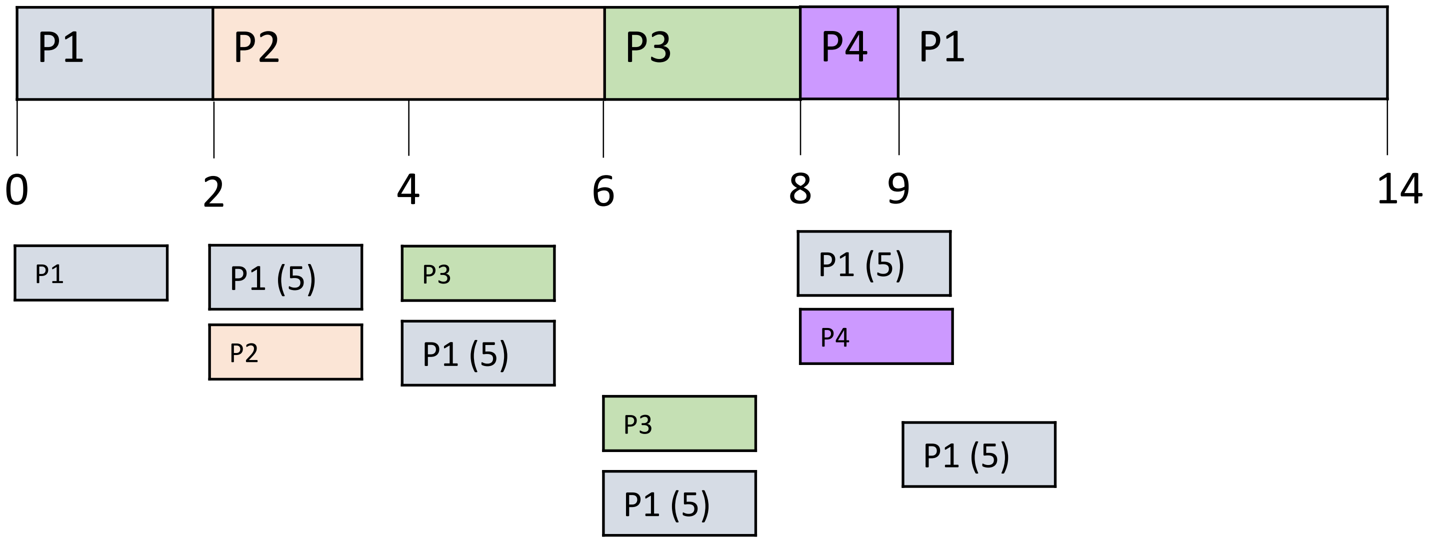
P3 arrives
CPU burst of P3 = remaining time of P2
Hence, no preemption

Shortest Remaining Time First (SRTF)

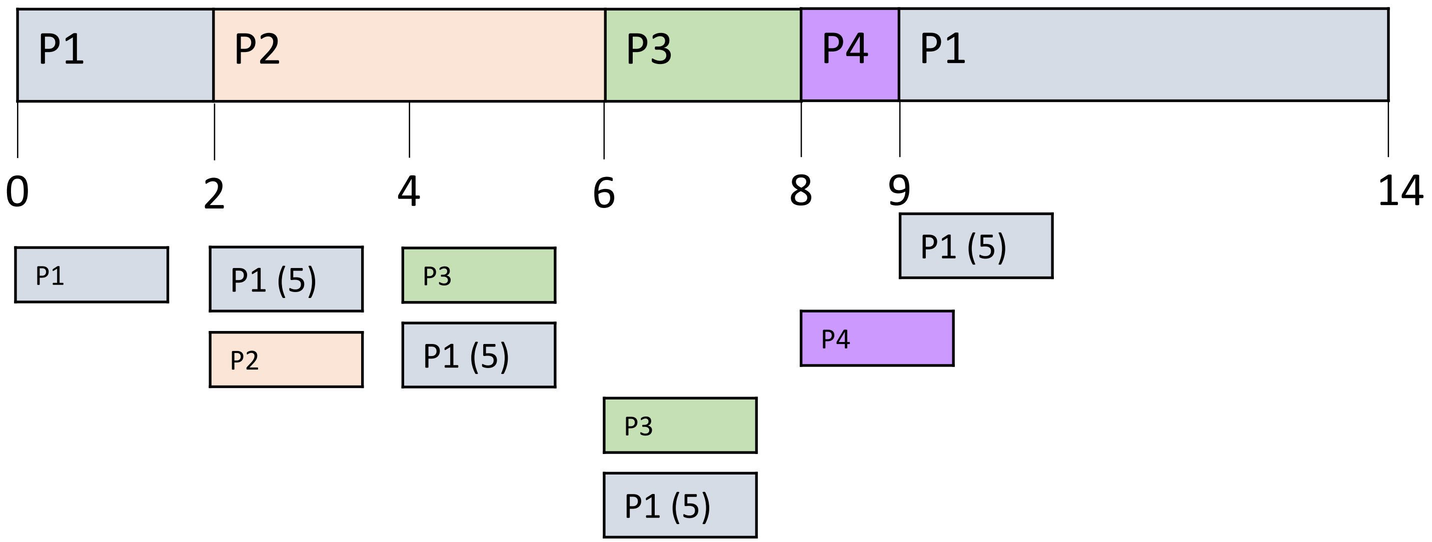
Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



Shortest Remaining Time First (SRTF)



waiting time for P1 =
waiting time for P2 =
waiting time for P3 =
waiting time for P4 =
average waiting time =

completion time for P1 =
completion time for P2 =
completion time for P3 =
completion time for P4 =
average completion time =

Shortest Remaining Time First (SRTF)

- <u>Pros and Cons</u>

  \+ Optimal average response time

  − you have to predict future

  − Unfair

  Multiple small jobs this can lead to starvation
  Large jobs may never get to run

SJF / SRTF

- SJF / SRTF are the best you can do at minimizing average response time
  - Provably optimal (SJF among non-preemptive, SRTF among preemptive)
  - SRTF is always at least as good as SJF

- <u>Comparison of SRTF with FCFS</u>
  - What if all jobs the same length?
    - SRTF becomes the same as FCFS (i.e. FCFS is best can do if all jobs the same length)

  - What if jobs have varying length?
    - SRTF: short jobs not stuck behind long ones

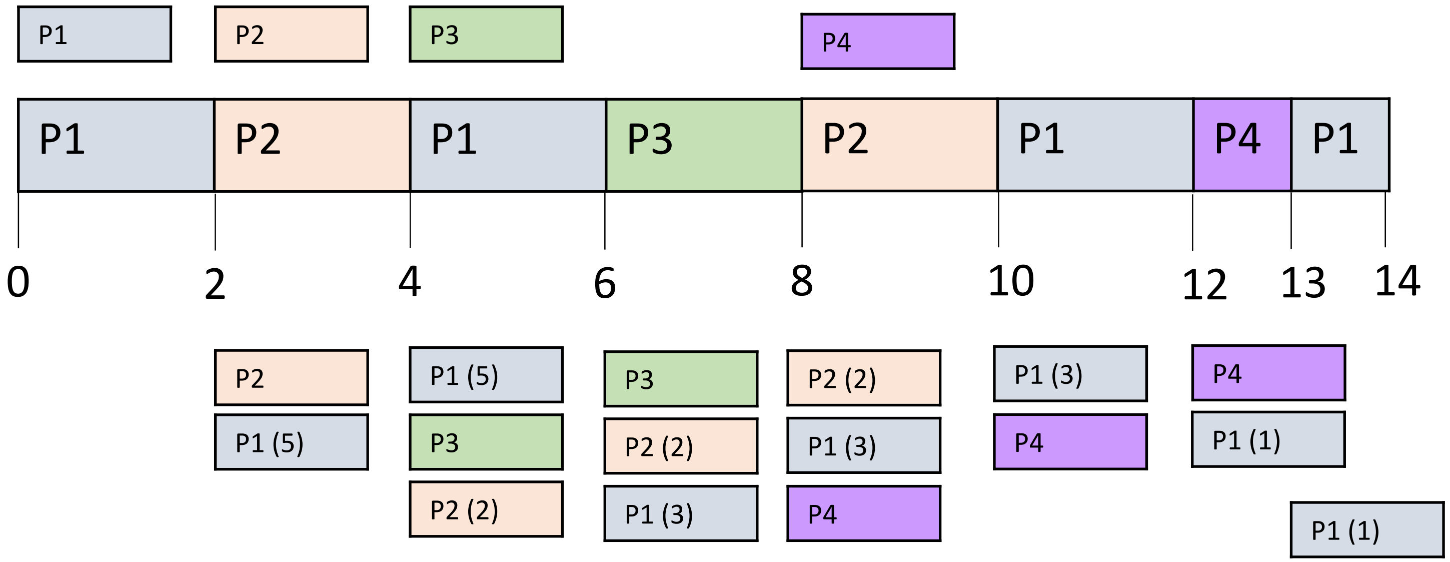
Robin Round Scheduling

- Robin Round Scheduling
  - Each process gets a small unit of CPU time (time quantum / time slice)
  - After quantum / slice expires the process is moved to a FIFO

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1

Robin Round Scheduling

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1

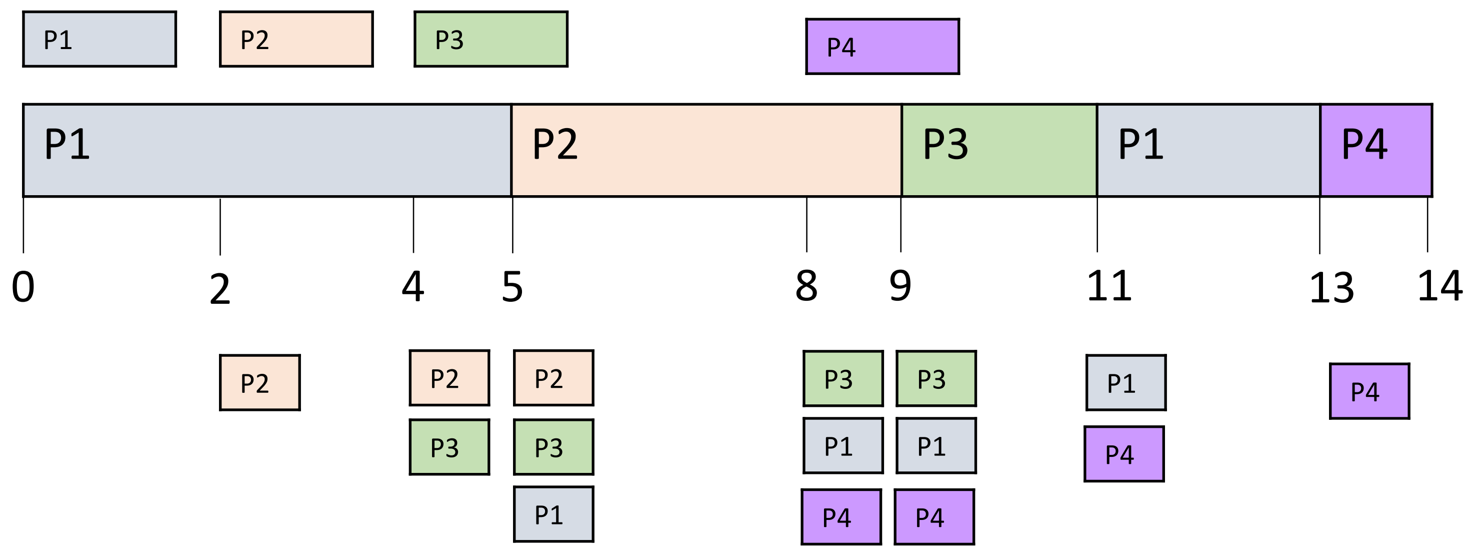


average waiting time =
average response time =
Context switches =

Time slice / quantum (q)= 2

Robin Round Scheduling

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	2
P4	8	1



average waiting time =
average response time =
Context switches =

Time slice / quantum (q) = 5

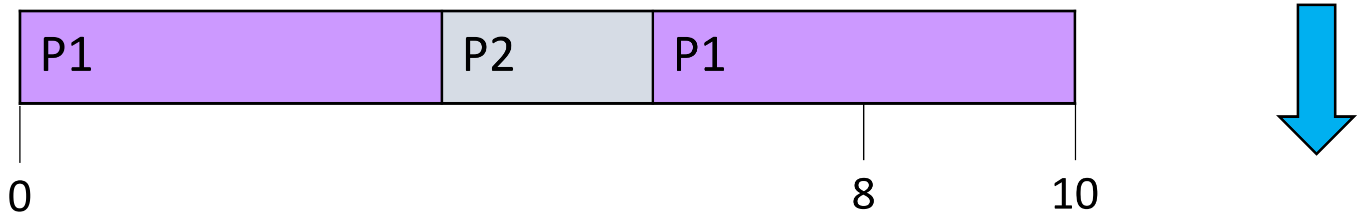
Robin Round Scheduling

P1: Burst Length 8 (Arrival: 0)
P2: Burst Length 2 (Arrival: 0)

- q = 8

Average Response time = (8 + 10) / 2 = 9

- q = 4

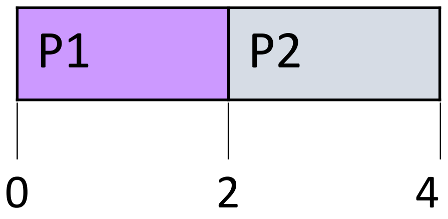


Average Response time = (10 + 6) / 2 = 6

Robin Round Scheduling

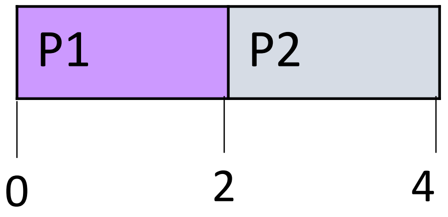
P1: Burst Length 2 (Arrival: 0)
P2: Burst Length 2 (Arrival: 0)

- q = 8



Average Response time = (2 + 4) / 2 = 3

- q = 4

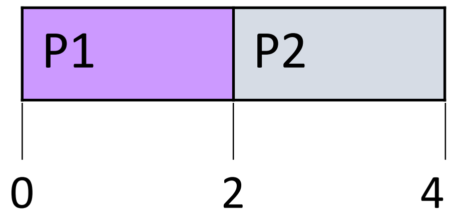


Average Response time = (2+ 4) / 2 = 3

Robin Round Scheduling

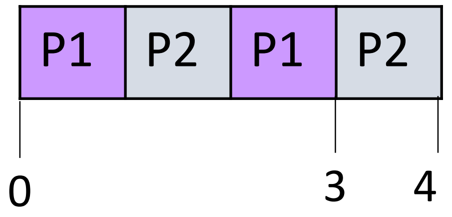
P1: Burst Length 2 (Arrival: 0)
P2: Burst Length 2 (Arrival: 0)

- q = 2



Average Response time = (2 + 4) / 2 = 3

- q = 1



Average Response time = (3+ 4) / 2 = 3.5

Robin Round Scheduling

- How to choose q?
  - Too Large: FCFS
  - Too Small: High Context Switches – Increased Overhead
  - q must be large with respect to the number of context switch

Robin Round Scheduling

<u>Pros and Cons:</u>

+ Fair (Each process gets a fair chance to run on the CPU)

+ Faster response time

− Increased context switching (increased overhead)

− Bad when the processes have same burst length