Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 2: Process Management

Unit 4

Process scheduling

fSO

DISCA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

- Goals
  - Understanding why the operating system requires a **CPU scheduler**
  - Knowing the **criteria to optimize** in order to select an appropiate **scheduler**
  - Studying different CPU **scheduling algorithms**

- Bibliography
  - A. Silberschatz, P. B. Galvin. “Sistemas Operativos Concepts” 9ª ed. Chapter 5

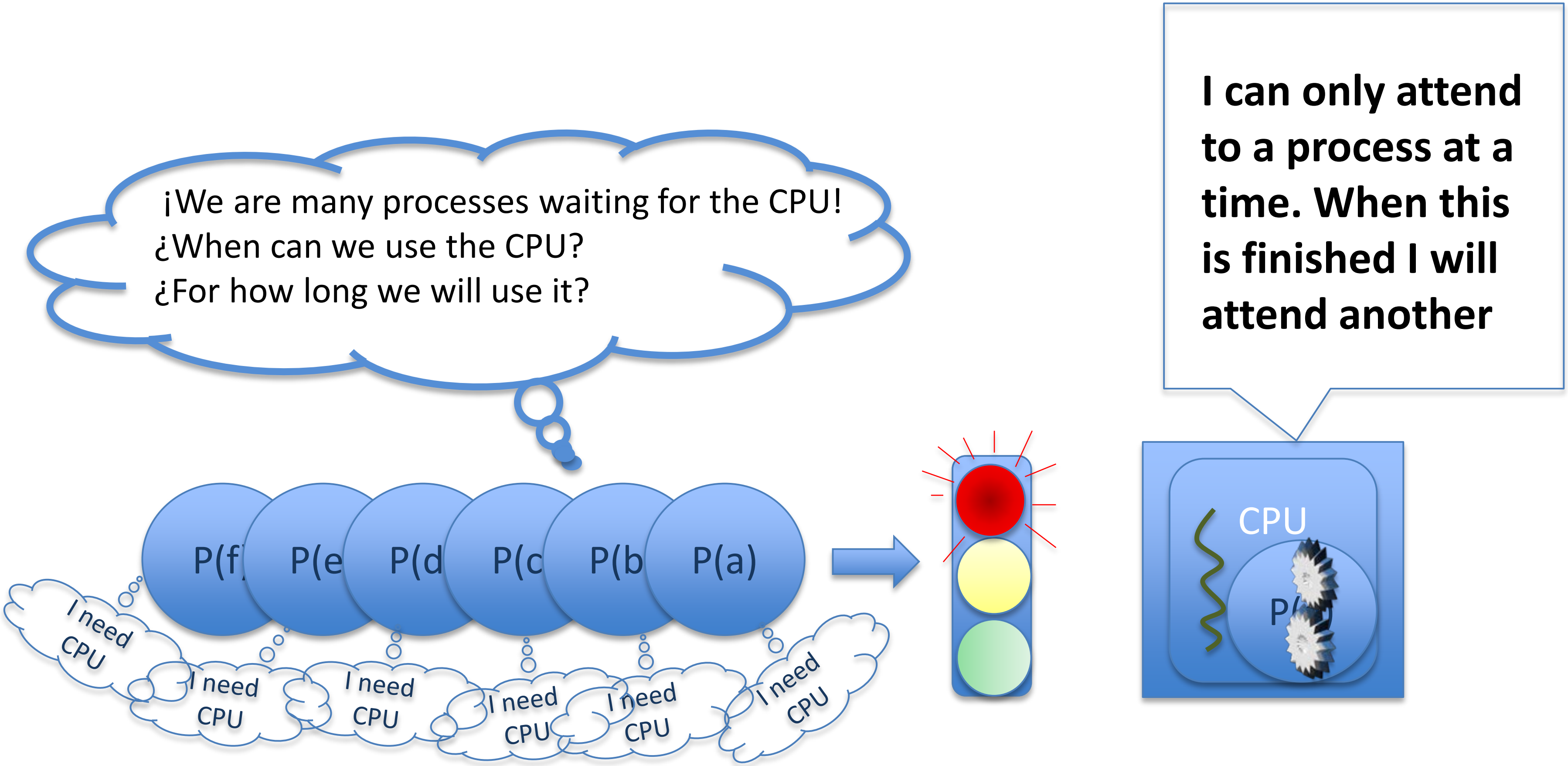
Content

- Scheduling concept
- Scheduling criteria
- Scheduling algorithms
  - FCFS
  - SJF
  - SRTF
  - Priorities
  - Round robin
- Multilevel queue

Content

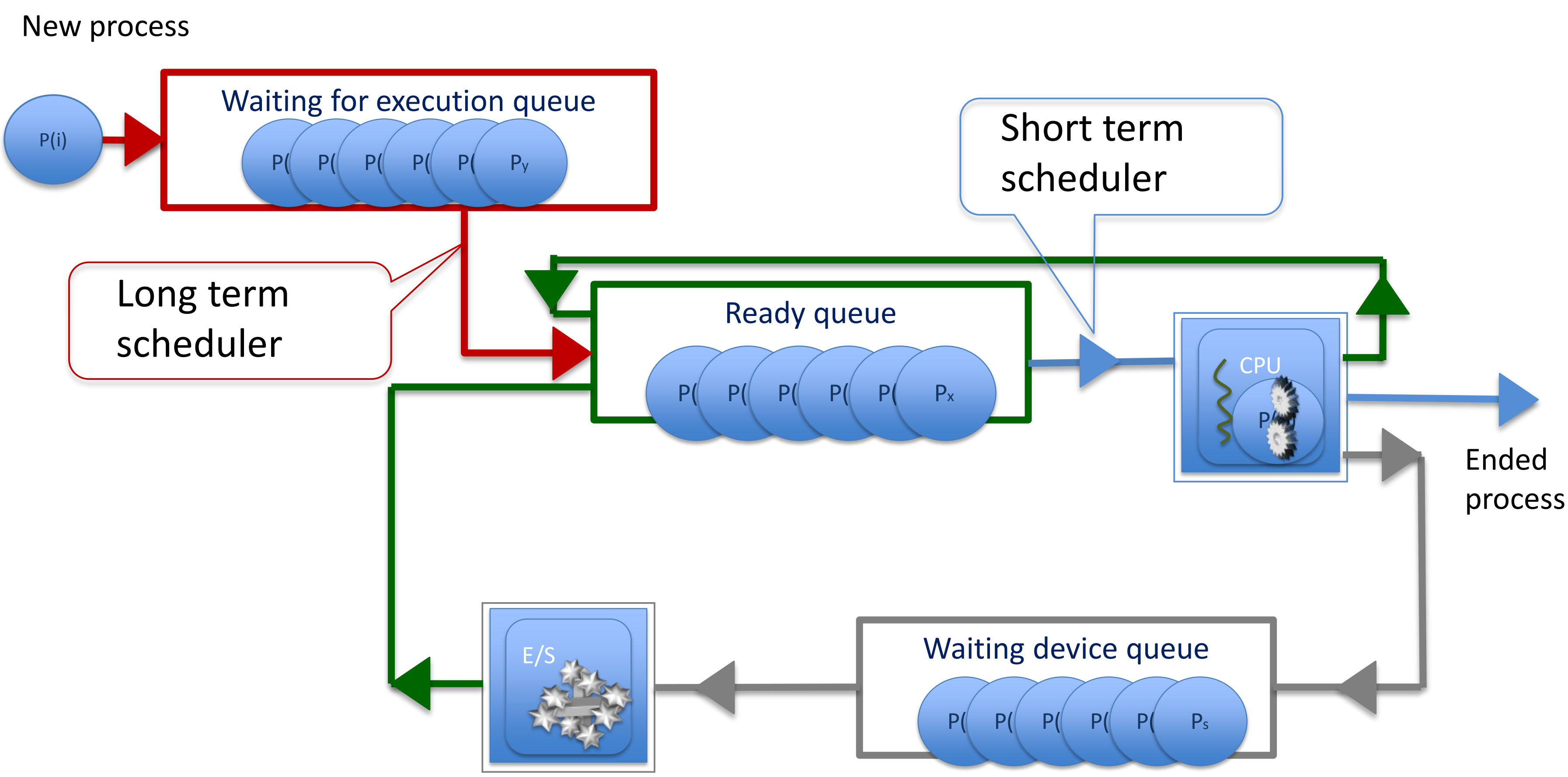
- **Scheduling concept**
- Scheduling criteria
- Scheduling algorithms
  - FCFS
  - SJF
  - SRTF
  - Priorities
  - Round robin
- Multilevel queue

Scheduling concept



- **Lack of resources:** Many processes competing for a single resource
- The OS has to implement a policy to allocate resources

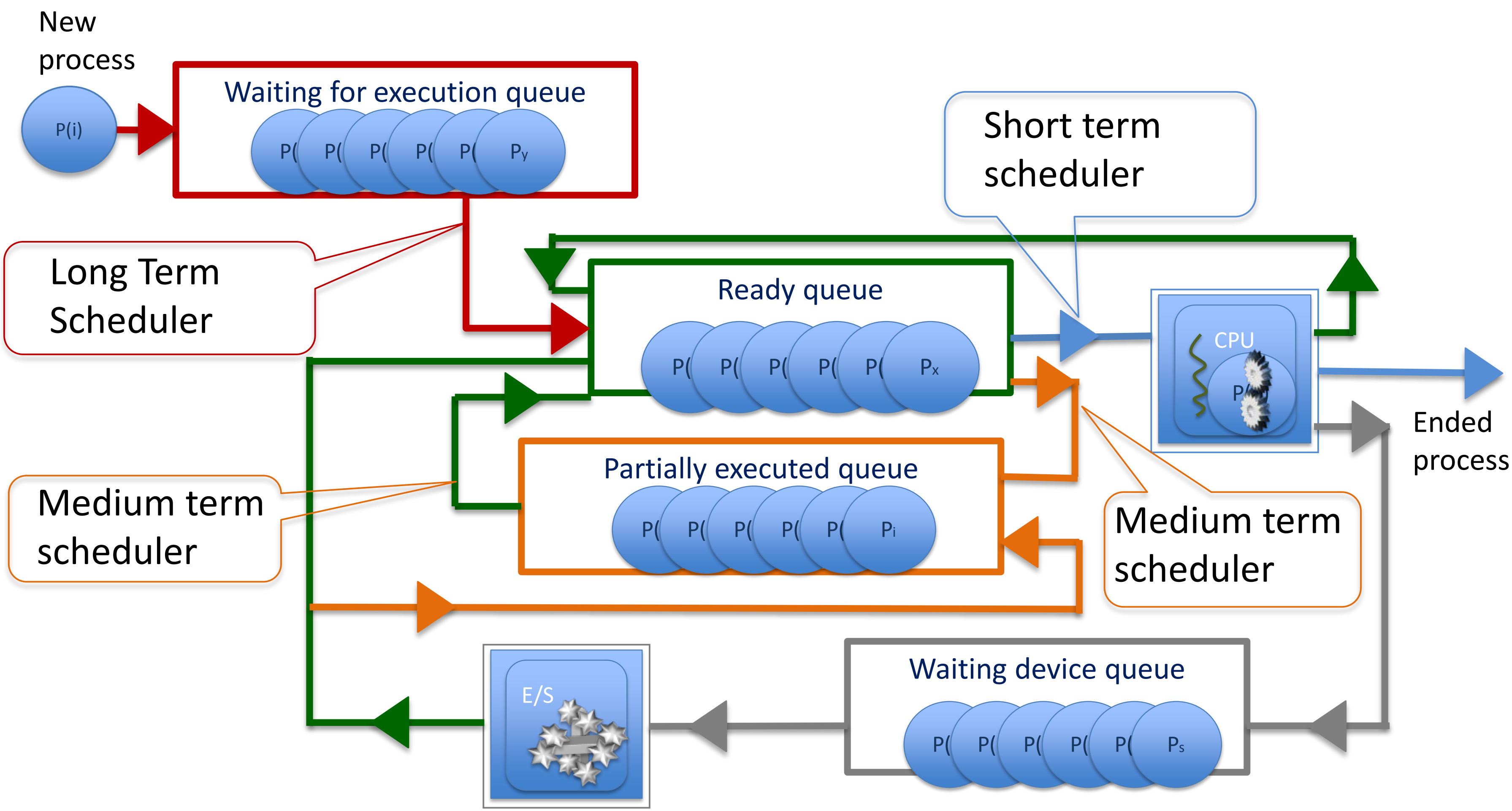
- Short and long term schedulers



Scheduler: OS component that decides what process gets a particular resource (i.e. the CPU) at every time instant, following a certain policy

Scheduling concept

- Short, medium and long term schedulers

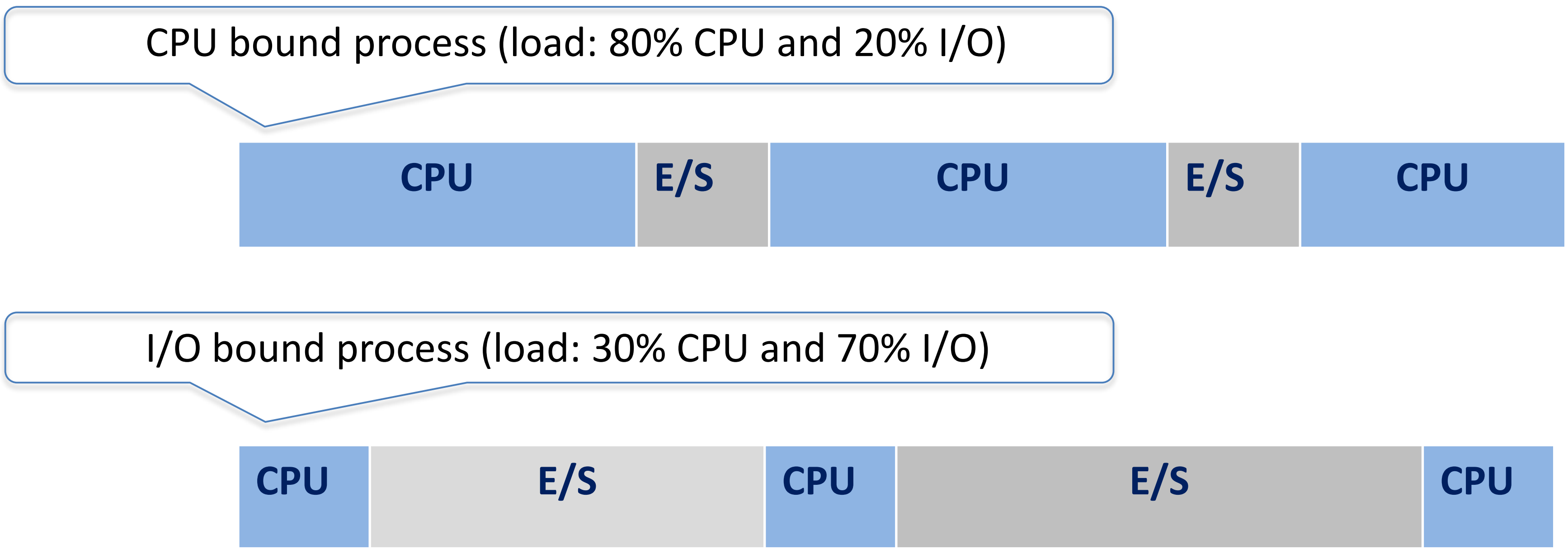


Medium term scheduler: It controls which processes, among the initialized ones, should be in memory and which in the swapping area.

Short term scheduler: It chooses a process from the ready queue for execution assigning to it the CPU.

Scheduling concept

- **Process types**: the active life of a process is a sequence of CPU bursts and I/O bursts
  - **CPU-bound** process: It spends most of its life time making calculations (i.e. MathLab)
  - **I/O bound** process: It spends more time making I/O than making calculations (i.e. SQL server)

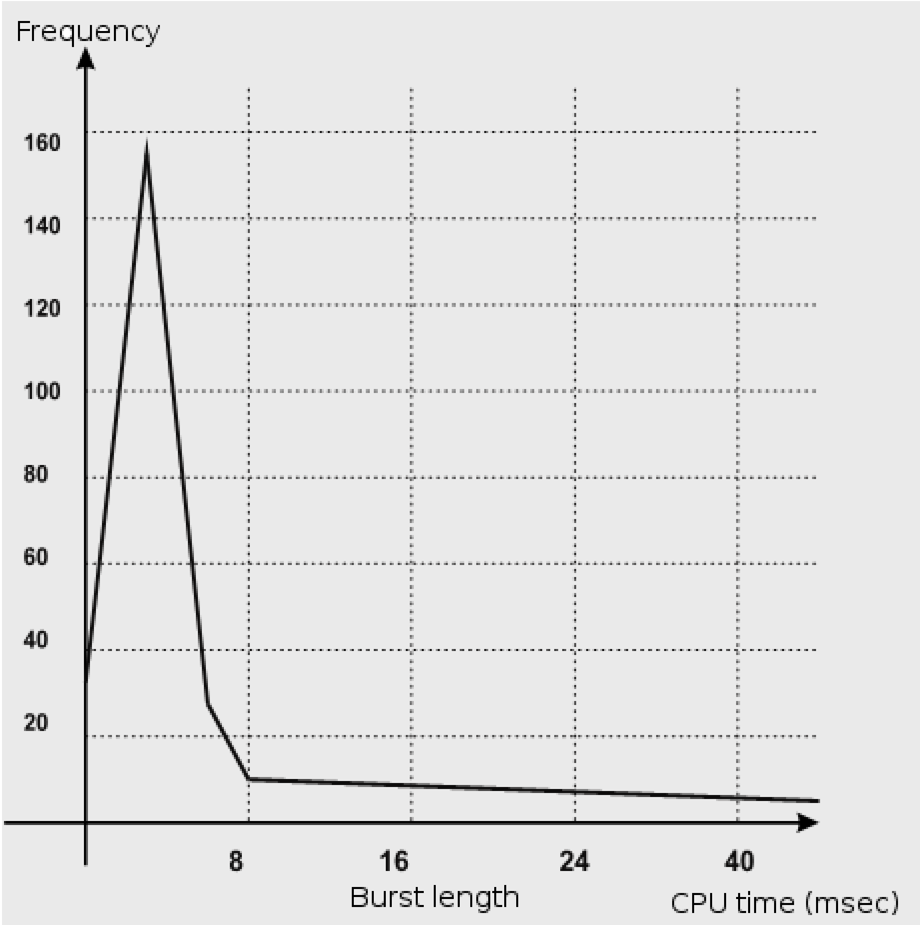


Scheduling concept

- CPU burst length

Statistical studies show that most processes have short CPU bursts together with I/O bursts

- **A large number** of **short** CPU bursts
- **An small number** of **large** CPU bursts



Content

- Scheduling concept
- **Scheduling criteria**
- Scheduling algorithms
  - FCFS
  - SJF
  - SRTF
  - Priorities
  - Round robin
- Multilevel queue

Scheduling criteria

- How to schedule depending on the load type?
  - **CPU Utilization:** Relative CPU busy time

    Resource_busy_time / Total_time
  - **Throughput**: Number of jobs processed per time unit

    Number_of_ended_processes / Total_time
  - **Turnaround Time:** Time elapsed between the arrival of a process and its completion

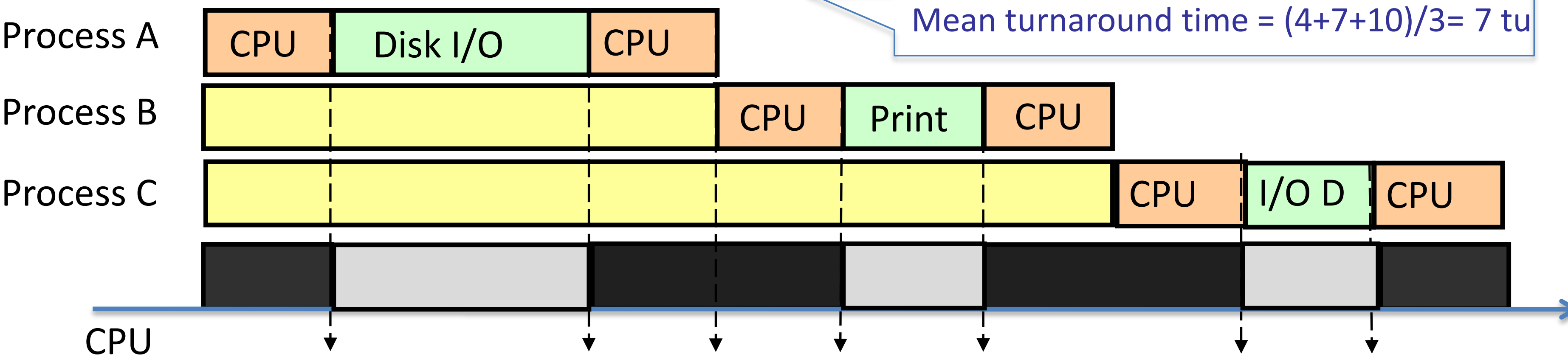
    $t_{out} - t_{in} = \sum T_{CPU} + \sum T_{I/O} + \sum T_{Queuing}$
  - **Waiting time:** Total time that a process spends in the ready queue
  - **Response time:** Time from launching a process until the CPU starts to execute its first instruction
  - **Fairness:** Ensuring that every process gets its fair share of CPU. That is, processes are treated equally. The opposite end of fairness is starvation

Scheduling criteria

- **Multiprogramming** itself improves many of the scheduling criteria compared to sequential execution

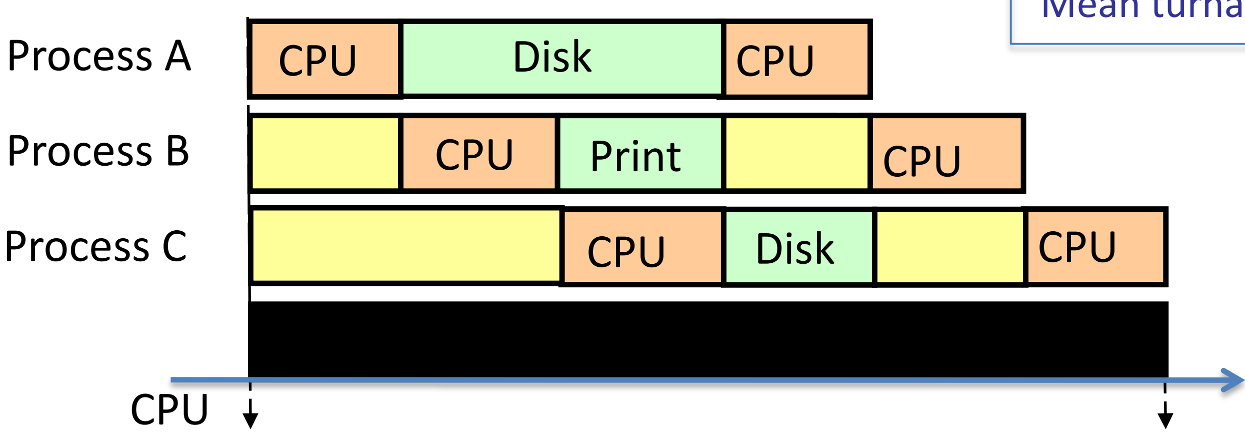
Without multiprogramming /Sequetial execution

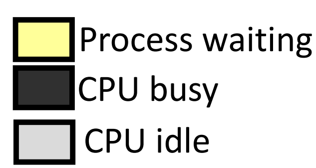
CPU utilization= 6 / 10 = 0.6 = 60%
Throuput = 3 / 10 = 0.3 jobs/tu
Mean turnaround time = (4+7+10)/3= 7 tu



With multiprogramming /Concurrent execution

CPU utilization = 6/6 = 1 = 100%
Throuput = 3 jobs/6 = 0.5 jobs/tu
Mean turnaround time = (4+5+6)/3= 5 tu



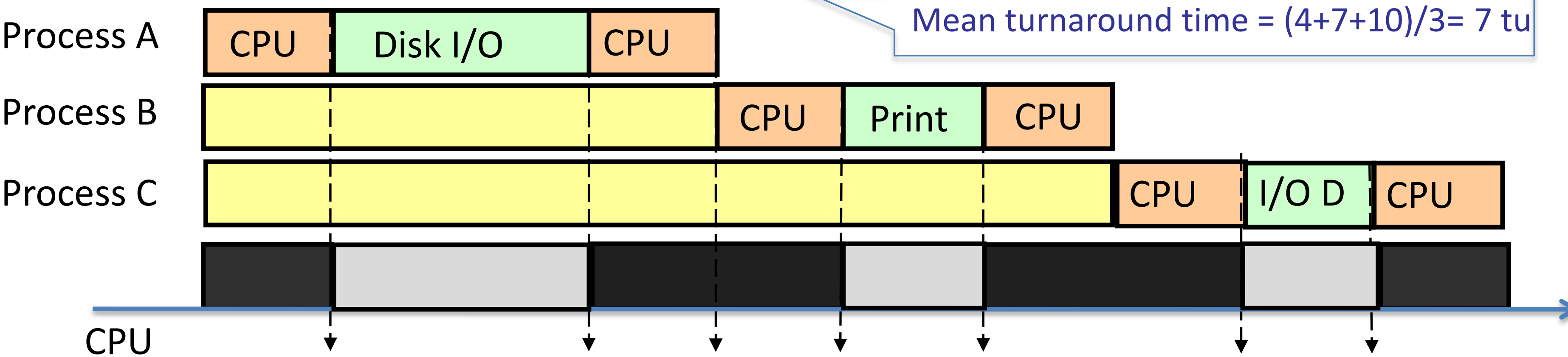


Scheduling criteria

- **Multiprogramming** itself improves many of the scheduling criteria compared to sequential execution

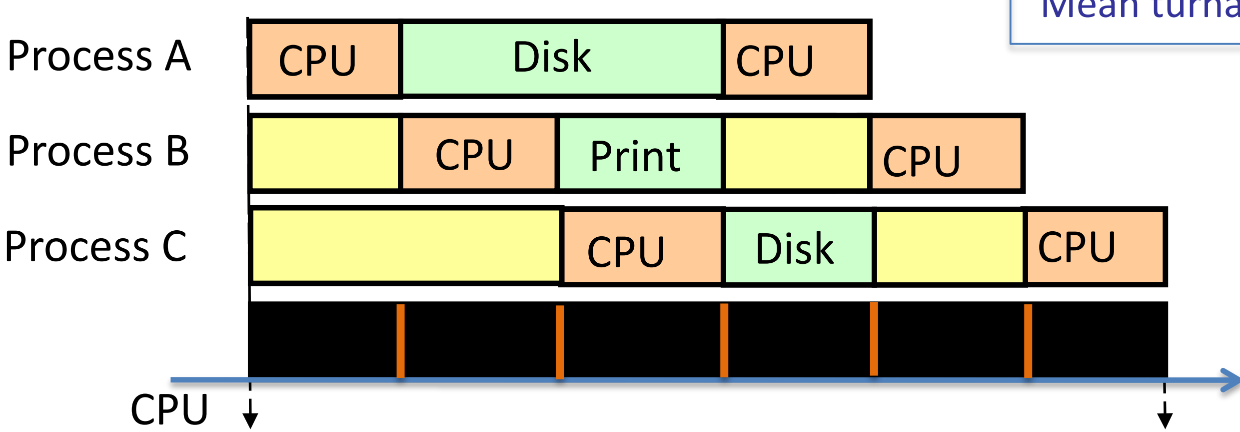
Without multiprogramming /Sequetial execution

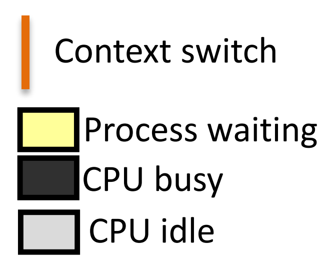
CPU utilization= 6 / 10 = 0.6 = 60%
Throuput = 3 / 10 = 0.3 jobs/tu
Mean turnaround time = (4+7+10)/3= 7 tu



With multiprogramming /Concurrent execution

CPU utilization = 6/6 = 1 = 100%
Throuput = 3 jobs/6 = 0.5 jobs/tu
Mean turnaround time = (4+5+6)/3= 5 tu



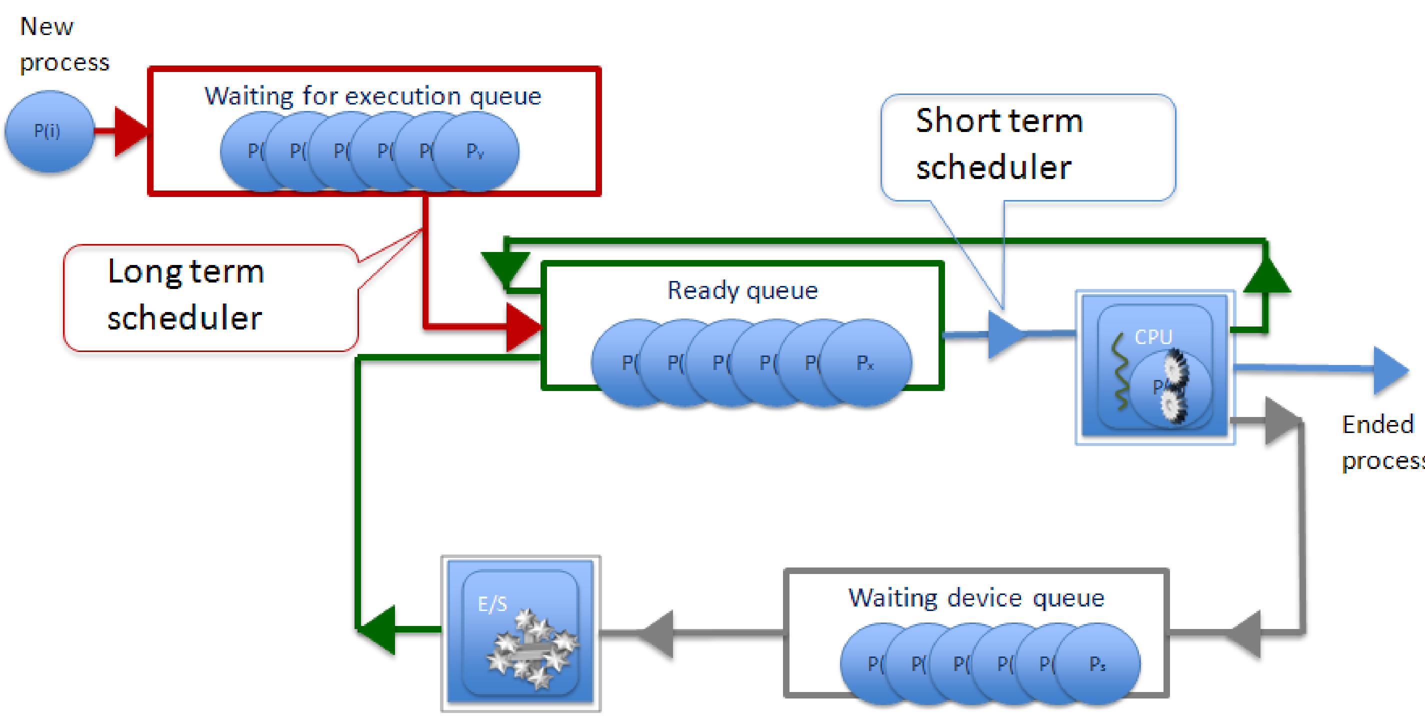


Content

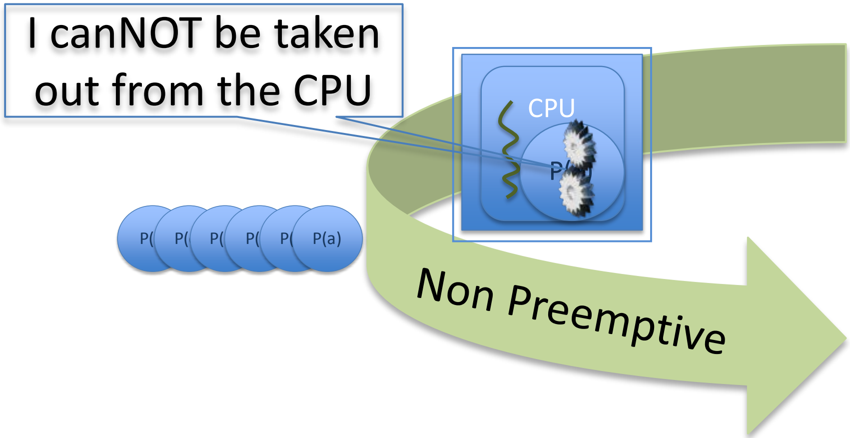
- Scheduling concept
- Scheduling criteria
- **Scheduling algorithms**
  - FCFS
  - SJF
  - SRTF
  - Priorities
  - Round robin
- Multilevel queue

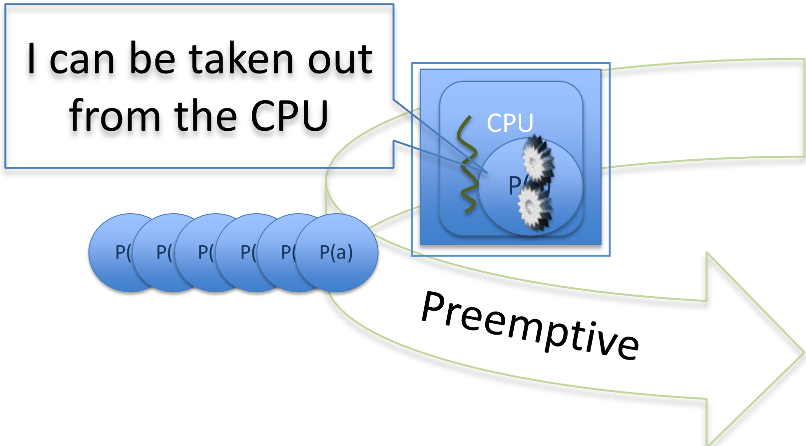
Scheduling algorithms

- Short-term scheduler target
  - Deciding to which process from those in the ready queue is assigned the CPU
- When the scheduler should act:
  - If the CPU is idle (process ends or gets suspended)
  - If a process arrives to the ready queue
  - Timer interrupt (round-robin)



- Scheduling policies: Non Preemptive/Preemptive
  - **Non Preemptive**: the process owns the CPU until voluntarily leaves (i.e. FCFS)
    - Less context switches, CPU grabbing can happen (i.e. Windows 3.11)
  - **Preemptive**: the scheduler can take out a process from the CPU
    - It is required to implement time sharing and real time (i.e. Unix, Windows NT and Mac OS X)





Scheduling algorithms

- Scheduling algorithms:
  - First-Come First-Served (FCFS).
  - Shortest-job-first (SJF)

NON Preemptive

  - Shortest-remaining-time-first (SRTF)
  - Round robin

Preemptive

  - Priorities
    - Preemption optional
    - Static / dynamic
  - Multilevel queue

Scheduling algorithms: FCFS

- **FCFS (first-come, first-served)**
  - **Non Preemptive:** When a process is assigned to the CPU it keeps it until ending or starting an I/O access
  - The CPU is allocated to processes in **arrival order to the ready queue**
  - Advantages: Easy to implement
  - Disadvantages:
    - Waiting time in not optimized
    - **Convoy effect**: short delay for long jobs
    - Not suitable for interactive systems

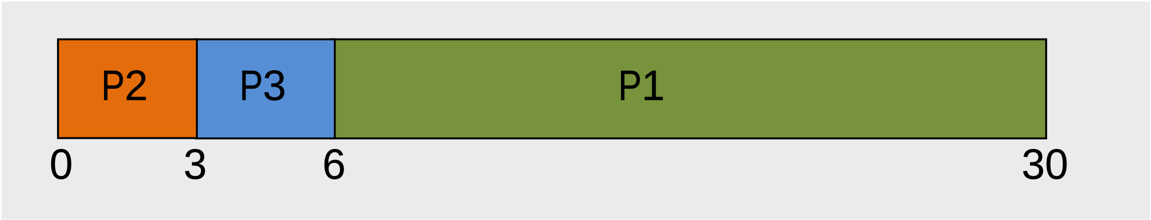
Process	Arrival time	CPU burst
P1	0	24
P2	0	3
P3	0	3

Case 1) Arrival order P1, P2, P3

Mean waiting time:
(0 + 24 + 27) / 3 = 17

Case 2) Arrival order P2, P3, P1

Mean waiting time:
(6 + 0 + 3) / 3 = 3

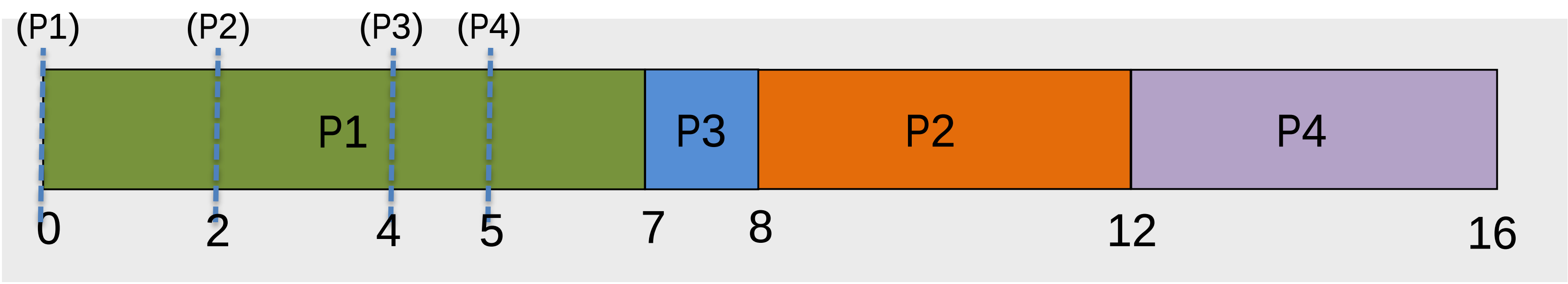


Scheduling algorithms: SJF

- **SJF (Shortest-Job-First)**
  - Each job is associated with the length of the next CPU burst
  - CPU is assigned to the job with smaller CPU burst
  - **Non Preemptive**

Process	Arrival time	CPU burst
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Arrival instant



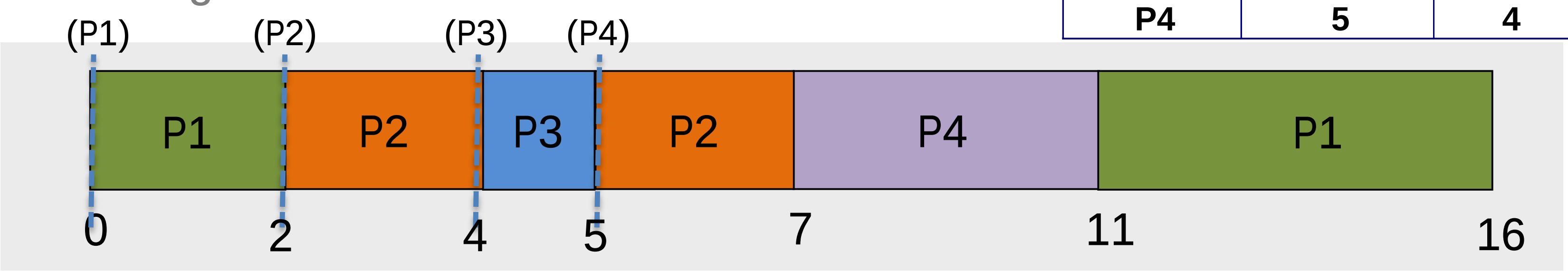
Mean waiting time: (0 + 6 + 3 + 7) / 4 = 4

Scheduling algorithms: SRTF

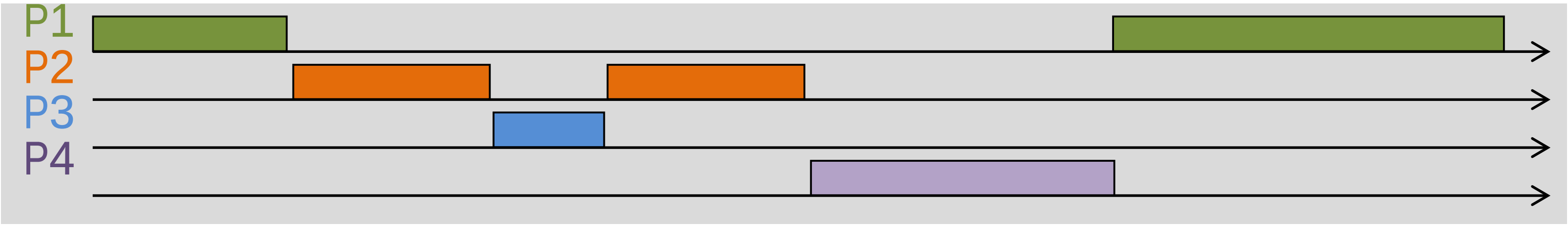
- **SRTF (Shortest-Remaining-Time-First)**
  - The CPU is allocated to the process with less reamining time to finish its CPU burst
  - **Preemptive**
  - Advantages: Optimize the average waiting time
  - Disadvantages:
    - Predicting the duration of the next range of CPU
    - Starvation risk on long jobs

Process	Arrival time	CPU burst
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Gantt diagram



Process time line



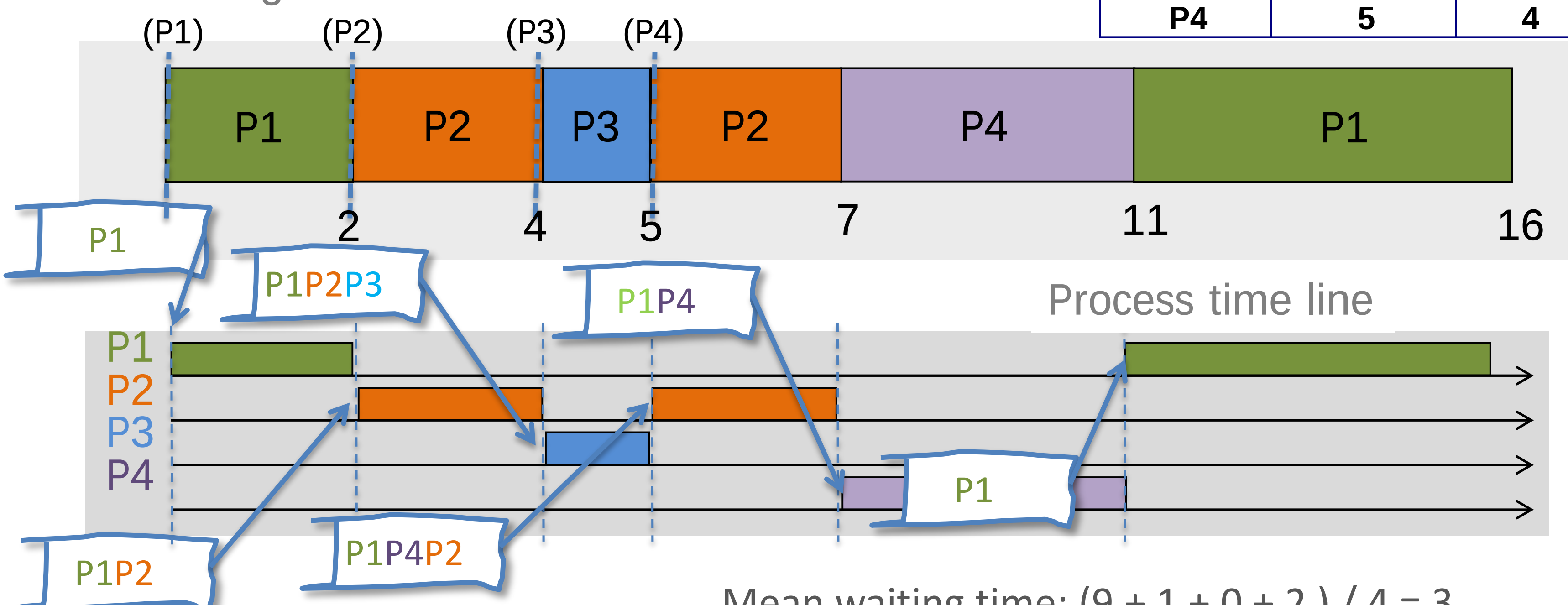
Mean waiting time: (9 + 1 + 0 + 2) / 4 = 3

Scheduling algorithms: SRTF

- **SRTF (Shortest-Remaining-Time-First)**
  - The CPU is allocated to the process with less reamining time to finish its CPU burst
  - **Preemptive**
  - Advantages: Optimize the average waiting time
  - Disadvantages:
    - Predicting the duration of the next range of CPU
    - Starvation risk on long jobs

Process	Arrival time	CPU burst
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Gantt diagram



Mean waiting time: (9 + 1 + 0 + 2) / 4 = 3

Scheduling algorithms: Priorities

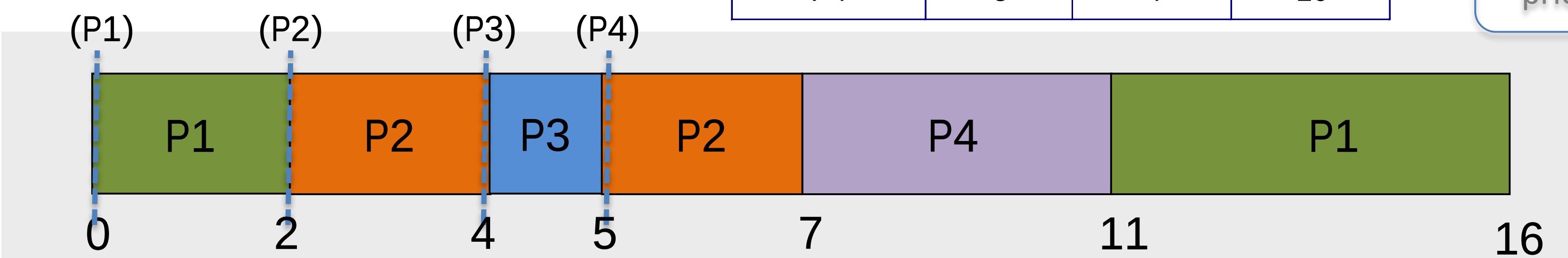
- **Scheduling with priorities (Preemptive)**
  - Every process is associated with a number (integer), called **priority** according to some criteria
  - CPU is allocated according to job priority (usually lower value means higher priority)

Process	Arrival time	CPU burst	Priority
P1	0	7	15
P2	2	4	10
P3	4	1	5
P4	5	4	10

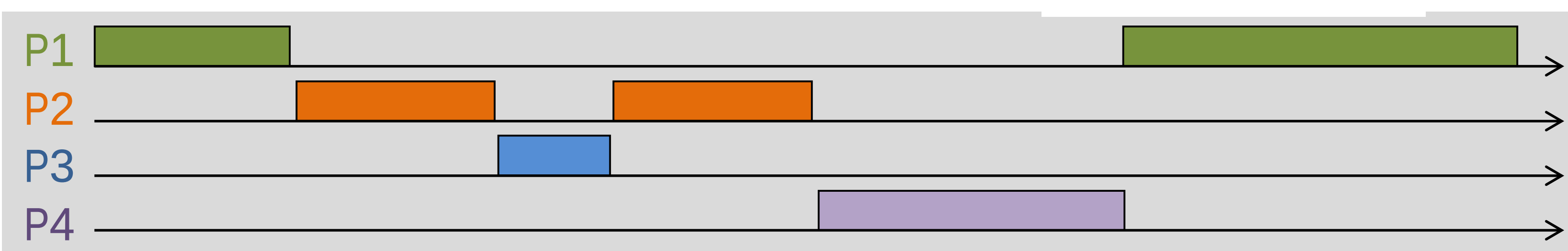
Less priority

More priority

Gantt diagram



Process time line



Mean waiting time : (9 + 1 + 0 + 2) / 4 = 3

Scheduling algorithms: Priorities

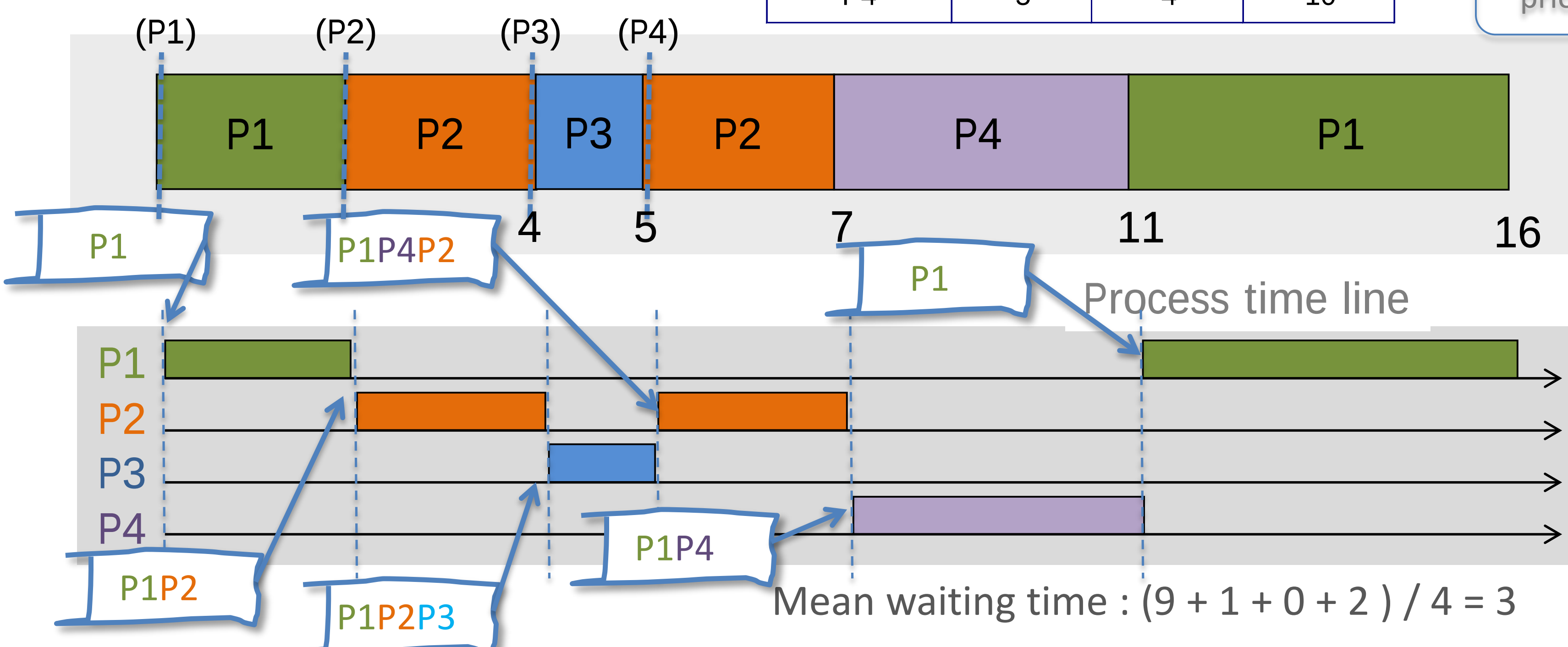
- **Scheduling with priorities (Preemptive)**
  - Every process is associated with a number (integer), called **priority** according to some criteria
  - CPU is allocated according to job priority (usually lower value means higher priority)

Process	Arrival time	CPU burst	Priority
P1	0	7	15
P2	2	4	10
P3	4	1	5
P4	5	4	10

Less priority

More priority

Gantt diagram



Mean waiting time : (9 + 1 + 0 + 2) / 4 = 3

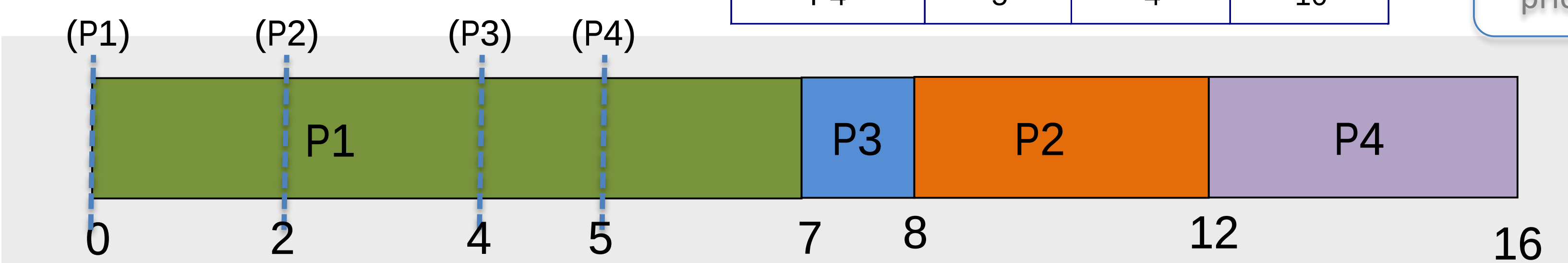
- **Scheduling with priorities (non Preemptive)**
  - Every process is associated with a number (integer), called **priority** according to some criteria
  - CPU is allocated according to job priority (usually lower value means higher priority)

Process	Arrival time	CPU burst	Priority
P1	0	7	15
P2	2	4	10
P3	4	1	5
P4	5	4	10

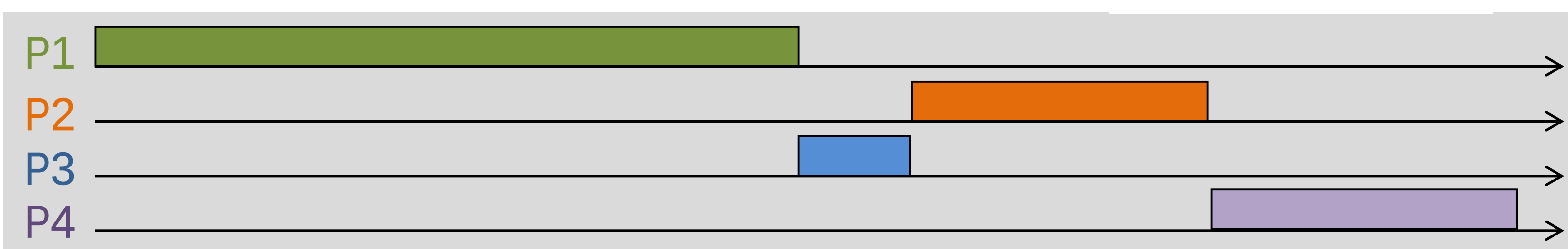
Less priority

More priority

Gantt diagram



Process time line



Mean waiting time : (0 + 6 + 3 + 7) / 4 = 4

Scheduling algorithms: Priorities

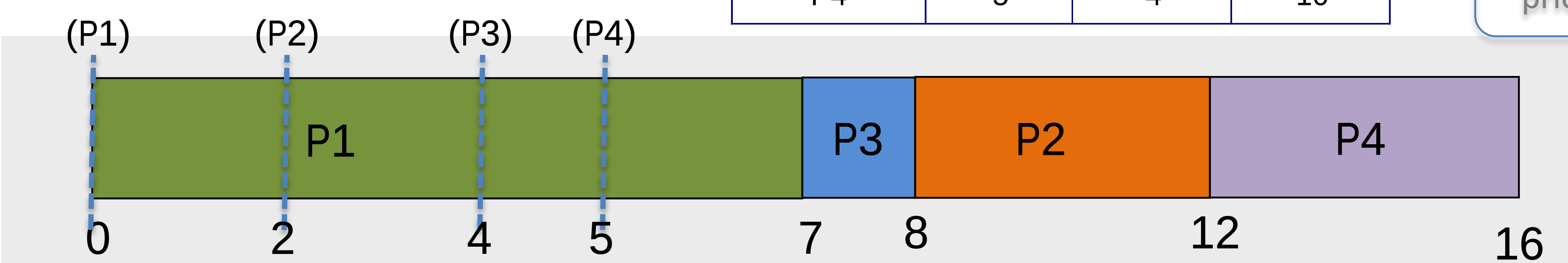
- **Scheduling with priorities (non Preemptive)**
  - Every process is associated with a number (integer), called **priority** according to some criteria
  - CPU is allocated according to job priority (usually lower value means higher priority)

Process	Arrival time	CPU burst	Priority
P1	0	7	15
P2	2	4	10
P3	4	1	5
P4	5	4	10

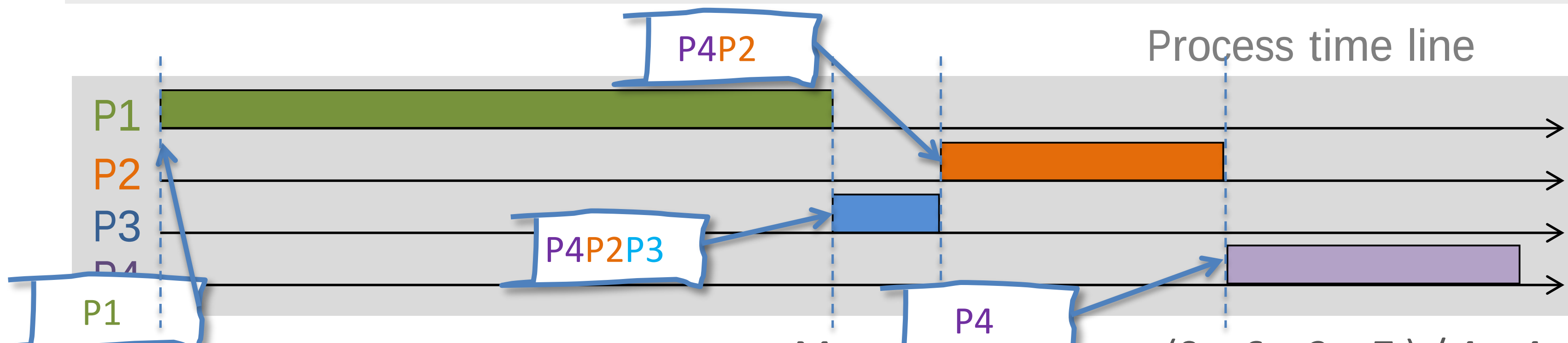
Less priority

More priority

Gantt diagram



Process time line



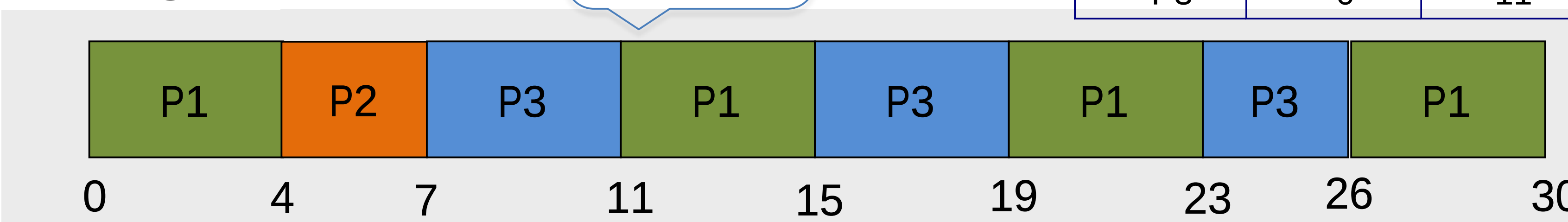
Mean waiting time : (0 + 6 + 3 + 7) / 4 = 4

Scheduling algorithms: Round robin

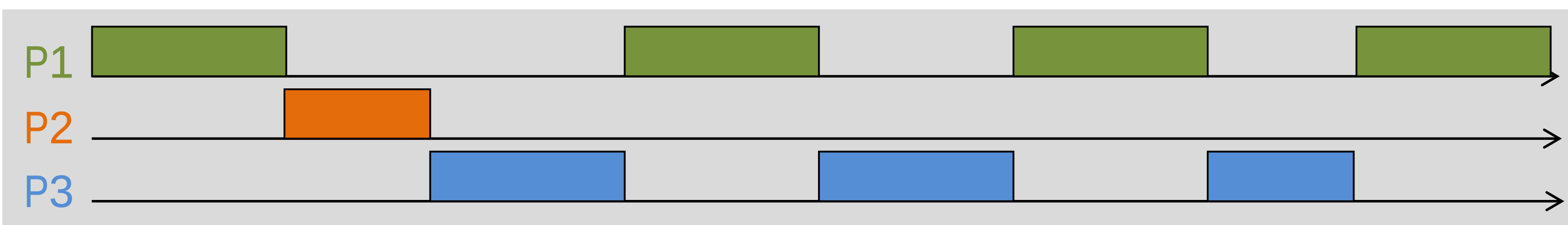
- **Round-Robin (RR) or circular scheduling**
  - Every process is assigned with a CPU time packet or "quantum" (**q**)
  - If the CPU burst is greater than **q**, then the process is get out from the CPU and it is put into the ready queue
  - If there are n processes in the ready queue, each gets 1/n of the CPU time in intervals of **q** units.

Process	Arrival time	CPU burst
P1	0	16
P2	0	3
P3	0	11

Gantt diagram



Process time line



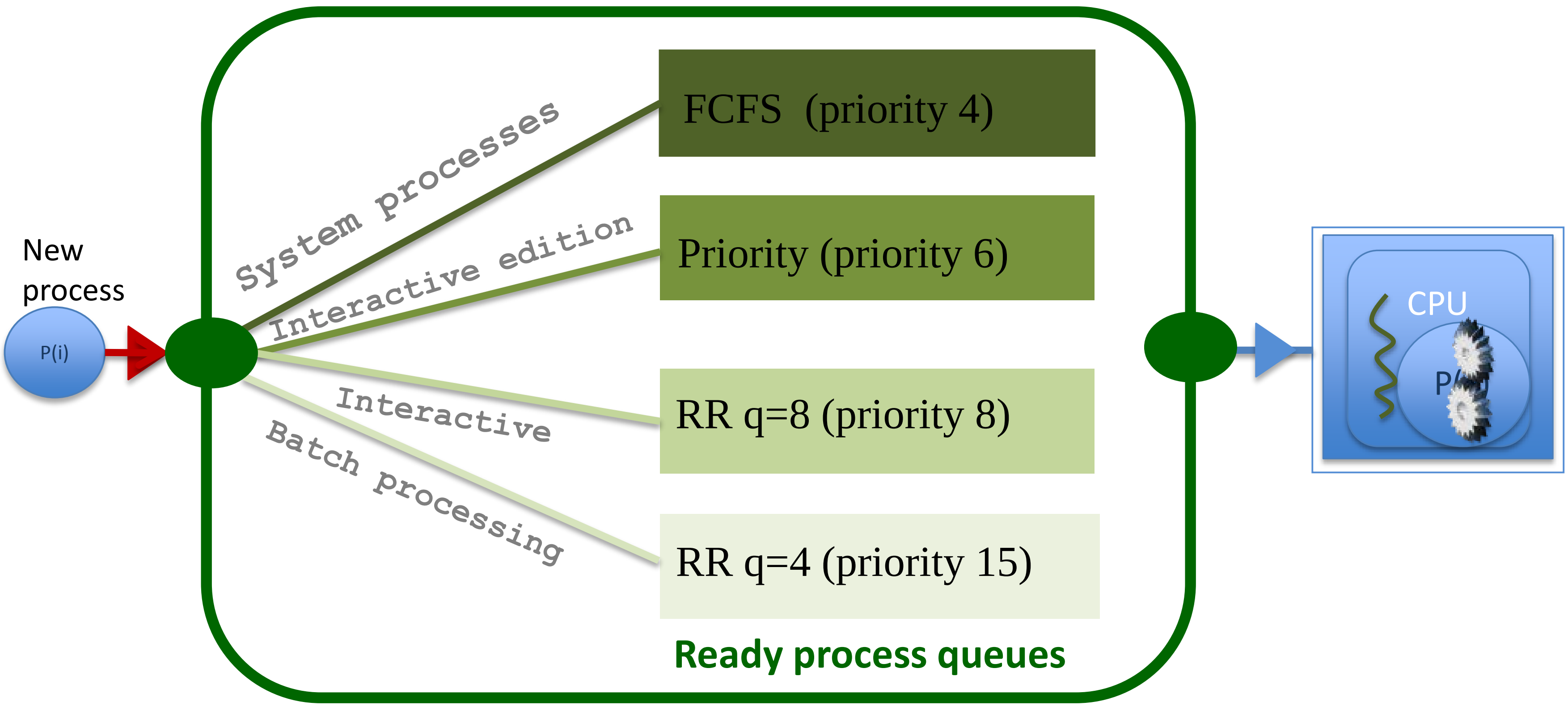
Mean waiting time : (14 + 4 + 15) / 3 = 11

Content

- Scheduling concept
- Scheduling criteria
- **Scheduling algorithms**
  - FCFS
  - SJF
  - SRTF
  - Priorities
  - Round robin
- **Multilevel queue**

Multilevel queue

- **Several queues of ready processes**
  - Every queue has its own scheduling policy
  - It is required an inter-queue scheduling
    - Preemptive priorities
    - CPU utilization (%)



Multilevel queue

- **Multilevel queue with feedback**
  - Parameters
    - Number of queues
    - Scheduling algorithm in every queue
    - Priority of every queue
    - Process promoting method
    - Process demoting method
    - Method to select the queue to enter every process