The CPU Scheduling

CPU Scheduling

Stefano Quer and Stefano Scanzio

Dipartimento di Automatica e Informatica

Politecnico di Torino

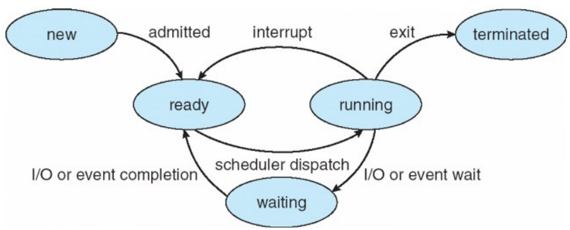
skenz.it/os stefano.scanzio@polito.it

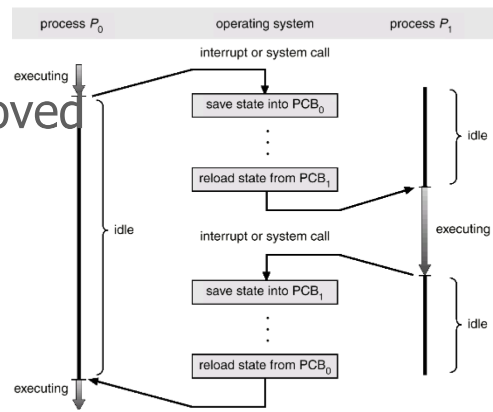
Fundamental concepts

- One of the main targets of multiprogramming is to maximize the use of the CPU resource
- To reach this target, more than one task (i.e., process or thread) is assigned to each
  - The scheduler must implement the better scheduling **algorithm** for the assign of the CPU to a task
  - Scheduler performance is evaluated through **cost functions**
  - Different applications require different algorithms and cost functions

Algorithms

- General scheduling procedure
  - Each time a process enters a waiting state, terminates, an interrupt is received, etc., it is necessary to perform a context switching operation
  - For each context switching
    - The task in the running state is moved in the ready queue
    - A task in the ready queue is moved in the running state





Algorithms

Extension

Algorithms without preemption	Algorithms with preemption
FCFS (First Come First Served) Scheduling in order of arrival	RR (Round Robin) Circular scheduling
SJF (Shortest Job First) Scheduling in order of length	SRTF (Shortest Remaining Time First) Scheduling for minimum remaining time
PS (Priority Scheduling) Scheduling in order of priority	
MQS (Multilevel Queue Scheduling) Multi-level queues scheduling	

Non preemptive
The CPU is **not** subtracted to another task, i.e., the task must release the CPU voluntarily

Preemptive
The CPU can be subtracted to another task, i.e., CPU burst are defined (e.g., maximum execution times) at the end of which the CPU is reassigned to another task

Cost functions

Cost function	Description	Optimum
CPU utilization	Percentage of CPU utilization	[0-100%] Maximum
Throughput	Number of processes completed in a time unit	Maximum
Turnaround time	Time that passes from the submission to the termination of a process	Minimum
Waiting time	**Total time spent in the ready queue (sum of the times spent in the queue)**	**Minimum**
Response time	Time elapsed between the submission and the production of the first response	Minimum

FCFS (First Come First Served)

- Algorithm
  - The CPU is assigned to the tasks following the order in which they requested it
    - Tasks are managed through a FIFO queue
      - A new task is inserted in the queue tail
      - A task to serve is extracted from the queue head
  - Scheduling can be sketched by means of a **Gantt diagram** (1917)
    - Bar chart showing the planning (start and end times) of the activities

Remember: No task is interrupted, i.e., the CPU can **only** be released voluntarily

FCFS (First Come First Served)

P	Arrival Time	Burst Time
P_1	0	24
P_2	0	3
P_3	0	3

Example 1

P	Waiting Time
P_1	(0-0) = 0
P_2	(24-0) = 24
P_3	(27-0) = 27
Average waiting time: (0+24+27)/3=17	

Task arrival order

Expected duration (unit of time)

P_1, P_2, P_3

Arrival Time

Gantt diagram

FCFS (First Come First Served)

P	Arrival Time	Burst Time
P_2	0	3
P_3	0	3
P_1	0	24

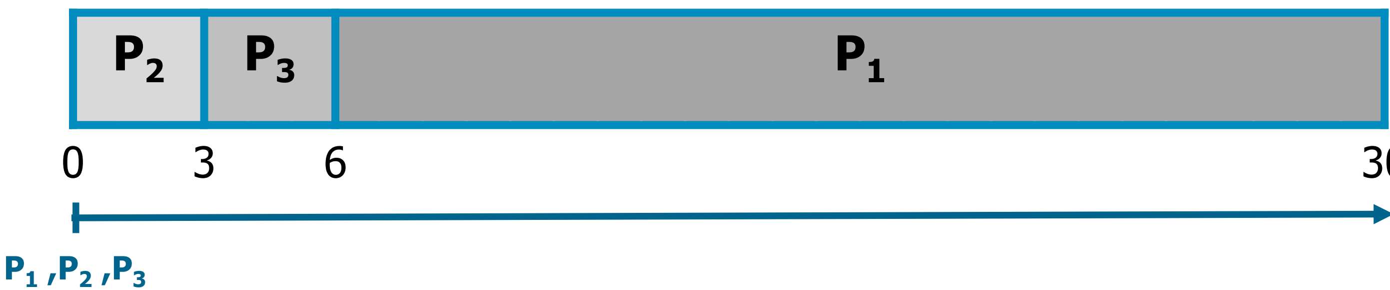
Example 2

P	Waiting Time
P_1	(6-0)=6
P_2	(0-0)=0
P_3	(3-0)=3
Average waiting time: (6+0+3)/3=3	

Task arrival order

Expected duration (unit of time)

Much **better** than the previous one: long processes delay short ones



FCFS (First Come First Served)

- Advantages
  - Easy to understand
  - Easy to implement
- Disadvantages
  - Waiting times
    - Relatively long
    - Variables and not optimal
  - Unsuitable for real-time systems (no preemption)
  - Queue effect
    - Short tasks queued after long tasks, wait for a long time uselessly

SJF (Shortest-Job-First)

- Algorithm
  - To each task is associated the duration of the next CPU request (**next CPU burst**)
  - The tasks are scheduled in order of duration of their next request
    - Scheduling in order of length
    - In case of ex-aequo (i.e., tasks with the same length) the FCFS scheduling is applied

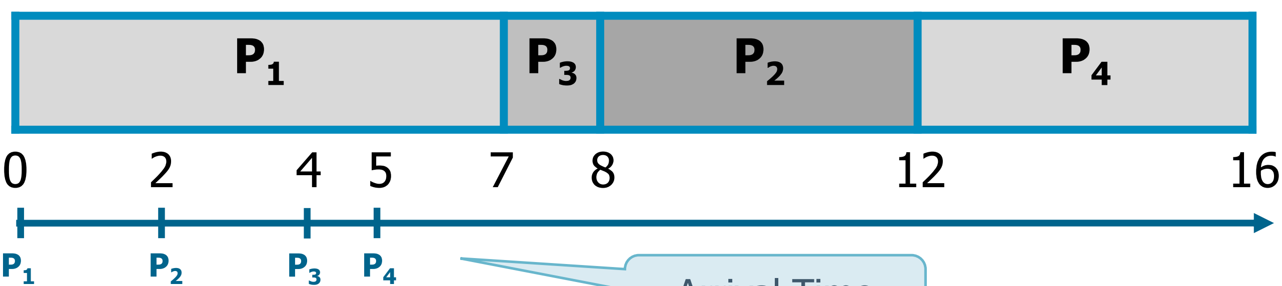
SJF (Shortest-Job-First)

P	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Task arrival order

Expected duration

P	Waiting Time
P_1	(0-0) = 0
P_2	(8-2) = 6
P_3	(7-4) = 3
P_4	(12-5) = 7
Average waiting time: (0+6+3+7)/4=4	



Arrival Time

SJF (Shortest-Job-First)

- Advantages
  - It can be demonstrated that SJF is an optimal algorithm, using the waiting time as a criterion
    - By moving the short processes before the long ones, the waiting time of the first decreases more than the increase of the waiting time of the seconds
- Disadvantages
  - Possible starvation
  - Difficult of application, due to the impossibility to know a priori the future behavior of the task
    - Next burst time is **unknown**
    - It is possible to **estimate** this time using different methods (e.g., the exponential average)

SJF (Shortest-Job-First)

- Exponential average

$$\tau_{n+1} = \alpha \cdot t_n + (1 - \alpha) \cdot \tau_n$$

Estimated n-th burst

Expected value for the next burst

(Real) duration of the n-th burst

α = [0, 1]
control the relative weight recent vs. past history
$\alpha = 0 \rightarrow \tau_{n+1} = \tau_n$
$\alpha = 1 \rightarrow \tau_{n+1} = t_n$

- Proceeding by substitution

$$\tau_{n+1}=\alpha \cdot t_n+(1-\alpha) \cdot \alpha \cdot t_{n-1}+\ldots +(1-\alpha)^j \cdot \alpha \cdot t_{n-j}+\ldots+ (1-\alpha)^{n+1} \cdot \tau_0$$

  - Since both α and 1-α are minor than 1, older terms weight less

PS (Priority Scheduling)

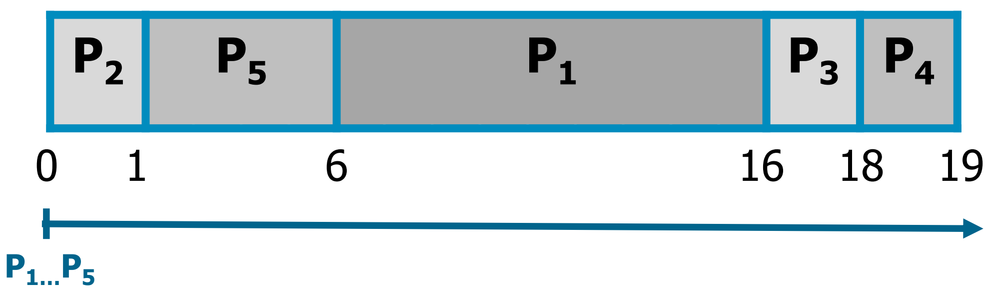
- Algorithm
  - A priority associated to each task
    - Priority is typically represented with integer number
    - The higher the priority the smaller the integer number
    - Priorities can be determined based on
      - Internal criteria: used memory, number of used files, etc.
      - External criteria: owner of the task, etc.
  - CPU is allocated to the task with higher priority
    - PS = SJF with the duration of the CPU burst substituted with the priority

PS (Priority Scheduling)

P	Arrival Time	Priority	Burst Time
P_1	0	3	10
P_2	0	1	1
P_3	0	4	2
P_4	0	5	1
P_5	0	2	5

P	Waiting Time
P_1	(6-0) = 6
P_2	(0-0) = 0
P_3	(16-0) = 16
P_4	(18-0) = 18
P_5	(1-0) = 1

Average waiting time:
(6+0+16+18+1)/5=8.2



PS (Priority Scheduling)

- Drawbacks
  - Possible starvation
    - In highly loaded systems, tasks with low priority can wait forever
      - MIT: IBM stopped in 1973 with a process queued since 1967
    - A possible solution to starvation is aging of tasks
      - The tasks priority is gradually increase over time

RR (Round Robin)

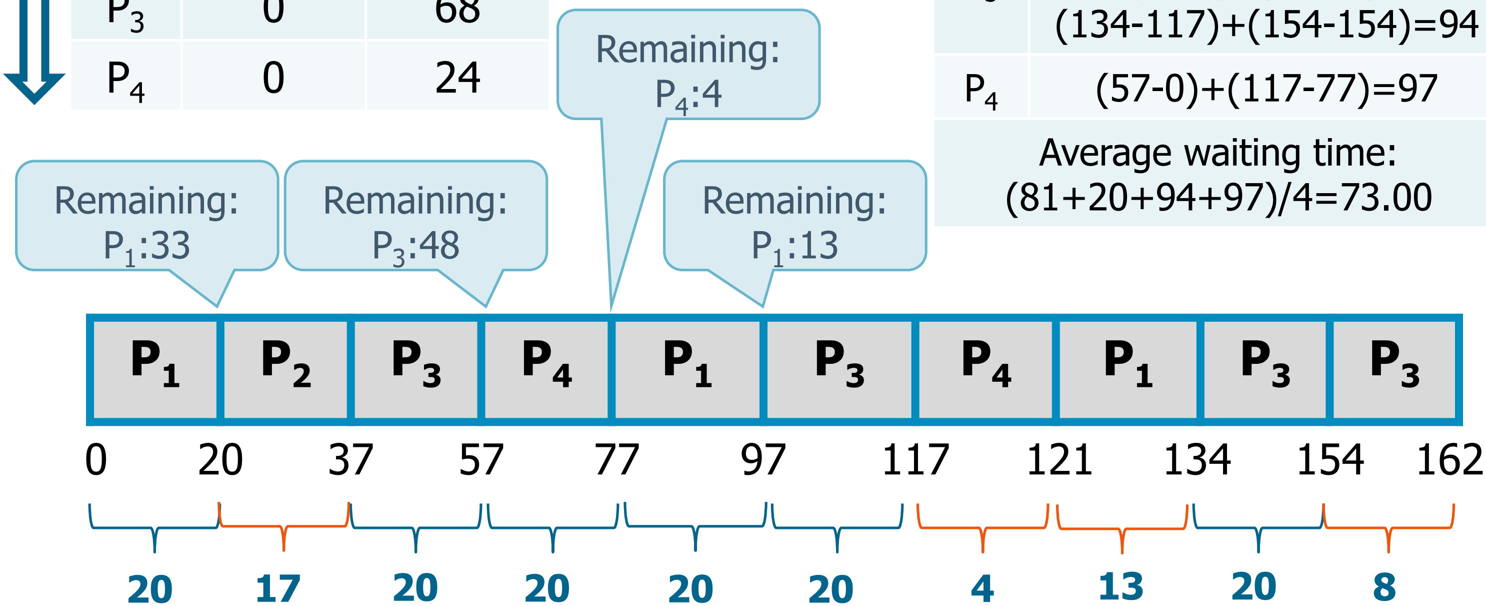
- Round Robin or circular scheduling
- Version of FCFS with preemption
- Algorithm
  - The CPU usage is divided into "time quantum" (i.e., discrete temporal intervals)
  - Each task can use the CPU for a maximum time equal to the quantum, and then it is inserted again in the ready queue
  - The ready queue is managed using a FIFO policy
    - New processes are inserted in the ready queue
- Designed specifically for time sharing (and some basic real-time systems)

RR (Round Robin)

P	Arrival Time	Burst Time
P_1	0	53
P_2	0	17
P_3	0	68
P_4	0	24

Quantum: 20 units

P	Waiting time
P_1	(0-0)+(77-20)+(121-97)=81
P_2	(20-0)=20
P_3	(37-0)+(97-57)+(134-117)+(154-154)=94
P_4	(57-0)+(117-77)=97
Average waiting time: (81+20+94+97)/4=73.00	



RR (Round Robin)

- Drawbacks
  - The average waiting time is relativelly long
  - Substantial dependence of performance on the length of the quantum
    - Quantum long: RR degenerates into FCFS
    - Quantum short: to much context switching are performed, and switching/management times are very high (if compared with useful work)

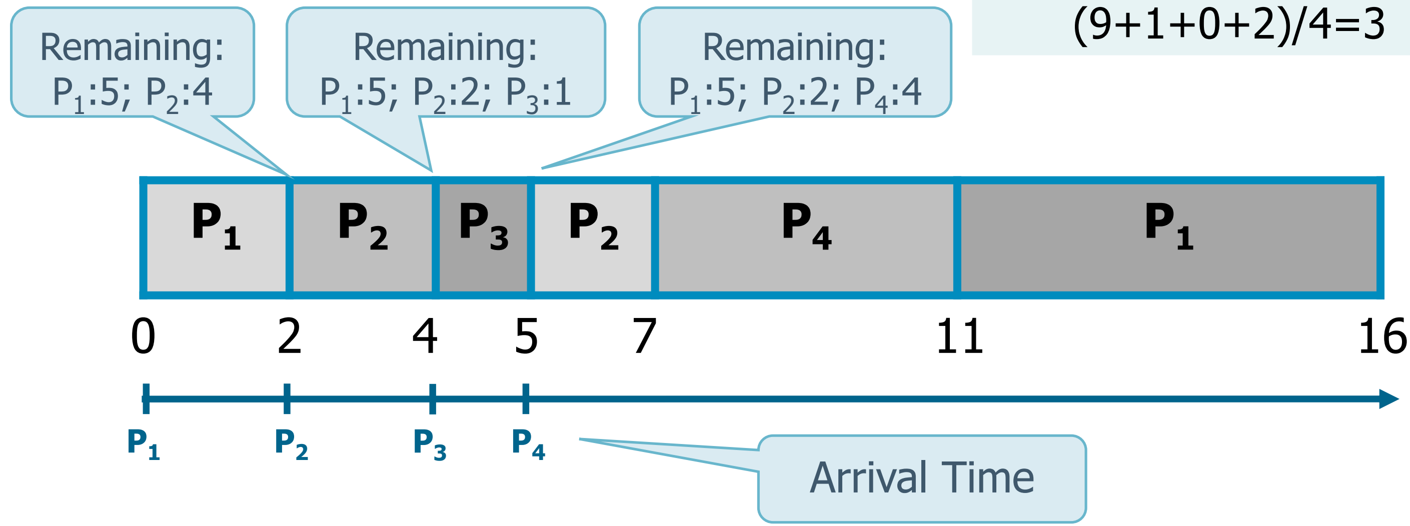
SRTF (Shortest-Remaining-Time-First)

- Version of SJF with preemption
- Algorithm
  - It proceeds with a scheduling of type SJF, but
  - if a task with smaller burst time (than the running one) is submitted, the CPU is preempted in favor of the new task
- Similar characteristics of the SJF scheduler

SRTF (Shortest-Remaining-Time-First)

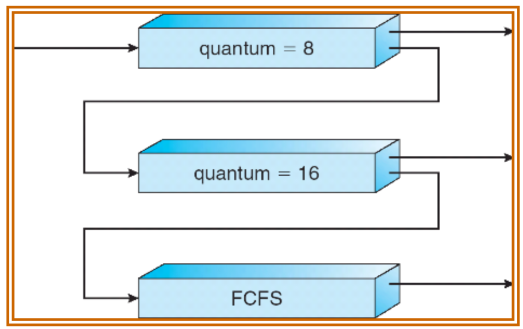
P	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

P	Waiting time
P_1	(0-0)+(11-2) = 9
P_2	(2-2)+(5-4) = 1
P_3	(4-4) = 0
P_4	(7-5) = 2
Average waiting time: (9+1+0+2)/4=3	



MQS (Multilevel Queue Scheduling)

- Applied to situations where tasks can be classified into different groups
  - Foreground, background, system, etc.
- Algorithm
  - The ready queue is divided into different queues
  - Each queue can be managed with its own scheduling algorithm
  - It can be modified to allow the transfer of tasks between the various queues
    - MQS with feedback



Additional considerations

- Scheduling can be performed at the process or thread level
  - If the OS allows the use of threads, the scheduling is normally performed at the threads level (processes are not taken into account)
- Threads scheduling
  - The SO takes into account only T at kernel level, and it ignores T at user level (which are managed through a library)
  - As a consequence, the scheduling can be performed only for T at kernel level (if they exist)

Additional considerations

- Scheduling for multiprocessors systems
  - All previous examples have been made assuming the existence of a single CPU
  - In the case of more than one CPU, load can be shared
    - The **load balance** is automatic for OS with waiting queues common to all processors
  - There are several schemes
    - **Asymmetric** multi-processing: a master processor distribute the load among slave processors
    - **Symmetric** multi-processing: each processor provides for its own scheduling

Additional considerations

- Scheduling for real-time systems
  - They try to respond in real-time and within predefined deadline to events
    - Events (e.g., raise of a signal and subsequent interrupt) guide the scheduling
    - **Latency** is defined as the time elapsing between the occurrence of an event and its management
  - There are two types of real-time systems
    - Soft real-time
      - They give priority to critical processes, but do not guarantee response times (only probabilistic guarrantees)
    - Hard real-time
      - The execution of the tasks is guaranteed within a maximum time limit (deadline)

Exam Italian Course: 2017/02/17

Exercise

- Considering the following set of processes

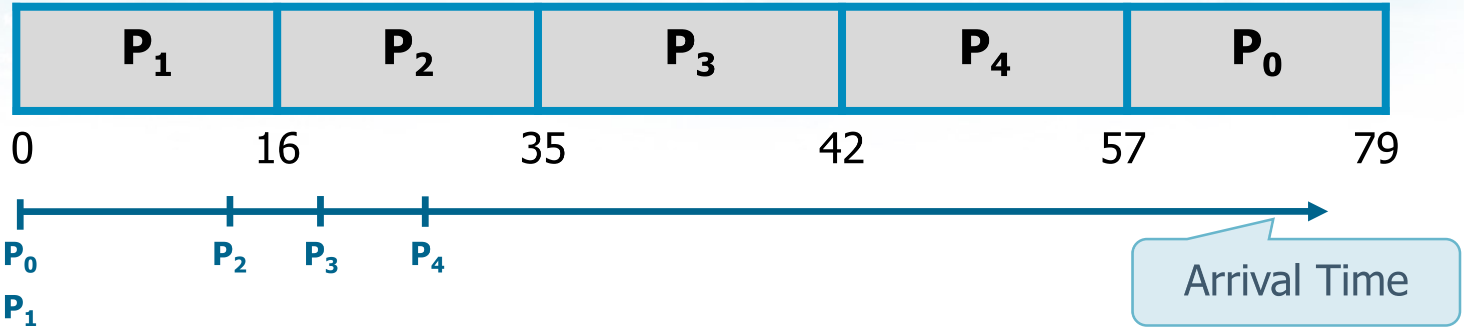
P	Arrival Time	Burst Time	Priority
P_0	0	22	5
P_1	0	16	2
P_2	15	19	4
P_3	17	7	1
P_4	25	15	1

  - Draw the Gantt diagram for the PS (Priority Scheduling), RR (Round Robin), and SRTF (Shortest Remaining Time First) algorithms
  - Compute the average waiting time

Arrival order of the tasks

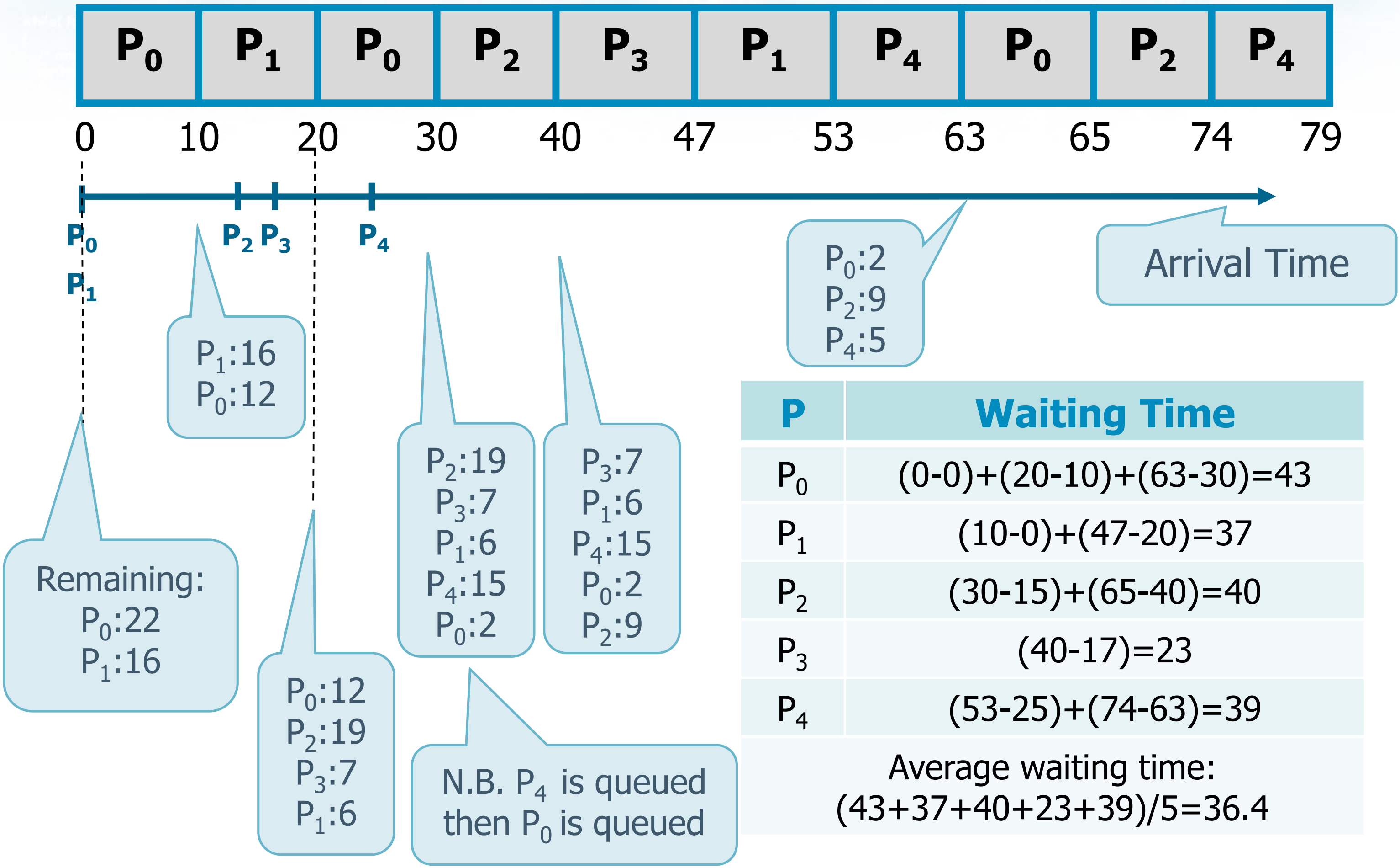
Maximum priority= smaller value
Temporal Quantum = 10

Exercise: PS



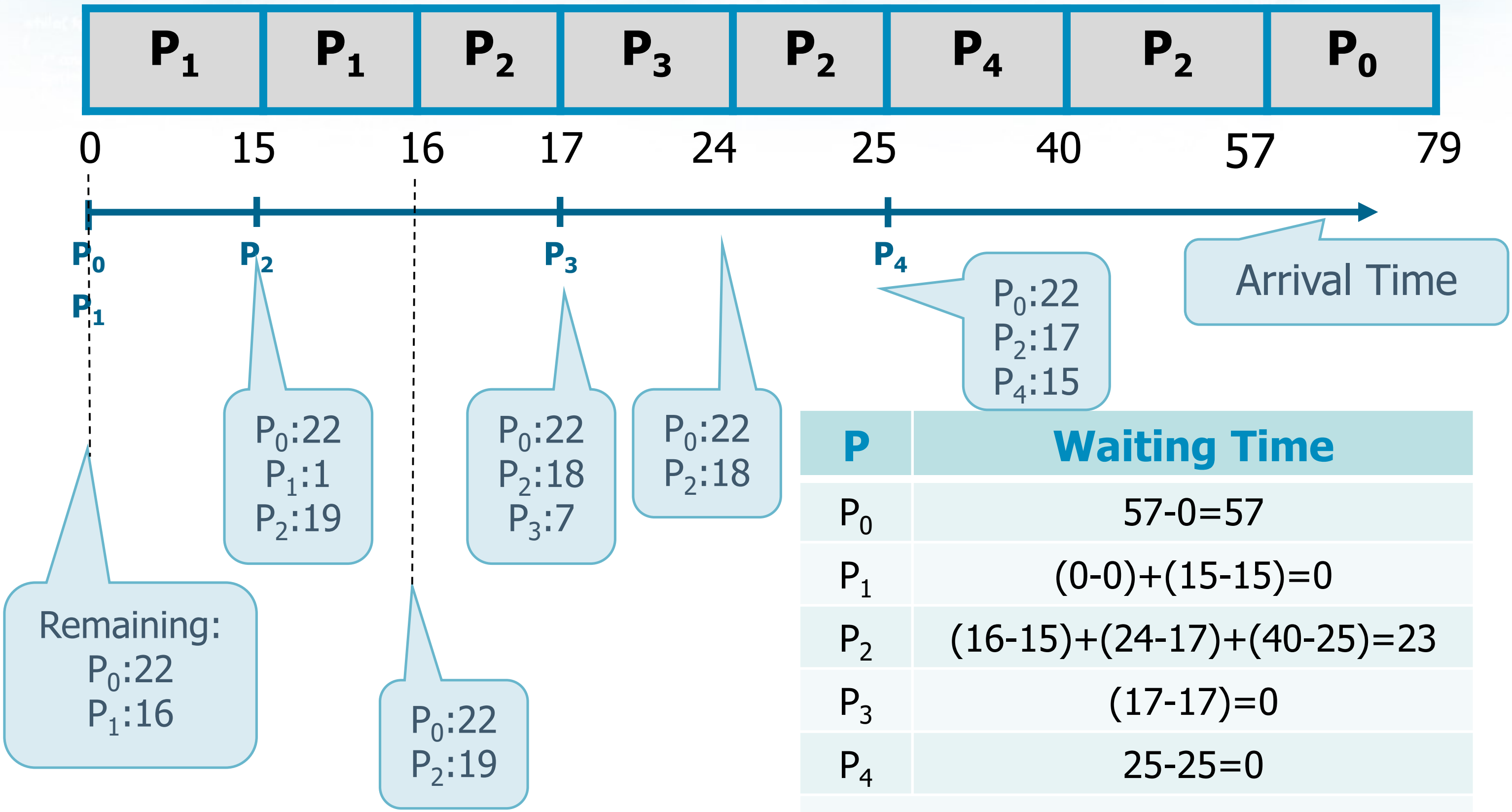
P	Waiting time
P_0	57-0=57
P_1	0-0=0
P_2	16-15=1
P_3	35-17=18
P_4	42-25=17
Average waiting time: (57+0+1+18+17)/5=18.6	

Exercise: RR



P	Waiting Time
P_0	(0-0)+(20-10)+(63-30)=43
P_1	(10-0)+(47-20)=37
P_2	(30-15)+(65-40)=40
P_3	(40-17)=23
P_4	(53-25)+(74-63)=39
Average waiting time: (43+37+40+23+39)/5=36.4	

Exercise: SRTF



P	Waiting Time
P_0	57-0=57
P_1	(0-0)+(15-15)=0
P_2	(16-15)+(24-17)+(40-25)=23
P_3	(17-17)=0
P_4	25-25=0

Average waiting time:
(57+0+23+0+0)/5=16.0