RATE MONOTONIC (RM) SCHEDULING ALGORITHM

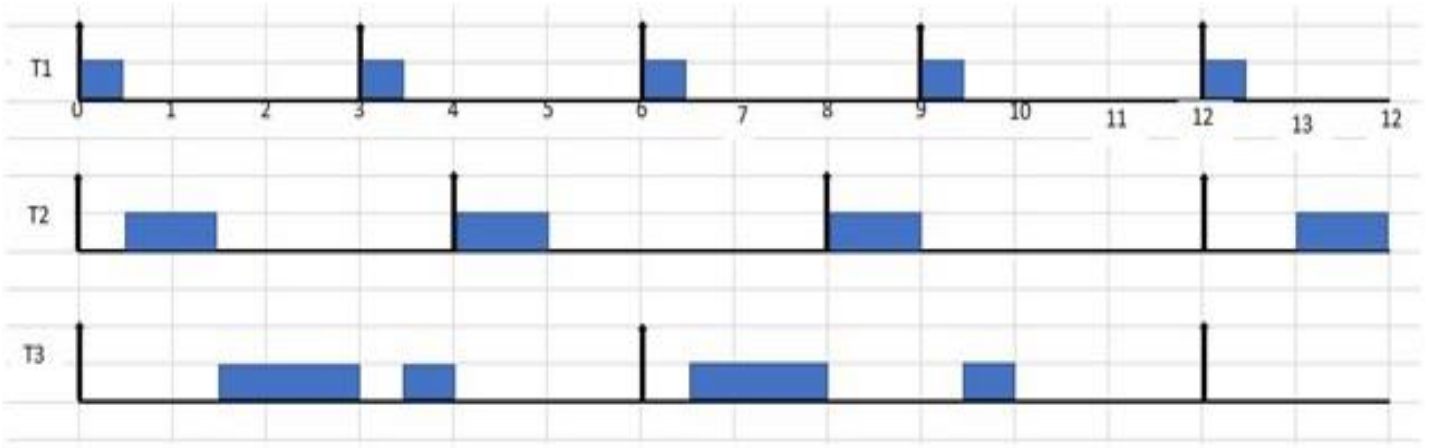
The shortest period = the highest priority

Tasks	Release time	Execution time	Time period
T1	**0**	**0.5**	**3**
T2	**0**	**1**	**4**
T3	**0**	**2**	**6**

The CPU utilization = 0.5/3 +1/4 +2/6 = 0.167+ 0.25 + 0.333 = 0.75 -> 75%

As processor utilization is less than 1 or 100% so task set is schedulable.

The timing diagram



Steps:

1. According to RM scheduling algorithm task with shorter period has higher priority so T1 has high priority, T2 has intermediate priority and T3 has lowest priority.
2. At t=0 all the tasks are released. Now T1 has highest priority so it executes first till t=0.5.
3. At t=0.5, task T2 has higher priority than T3 so it executes first for 1-time unit till t=1.5. After its completion only one task is remained in the system that is T3, so it starts its execution and executes till t=3.

4. At t=3, T1 releases, as it has higher priority than T3 so it preempts or blocks T3 and starts it execution till t=3.5. After that, the remaining part of T3 executes.
5. At t=4, T2 releases and completes it execution as there is no task running in the system at this time.
6. At t=6, both T1 and T3 are released at the same time but T1 has higher priority due to shorter period so it preempts T3 and executes till t=6.5, after that T3 starts running and executes till t=8.
7. At t=8, T2 with higher priority than T3 releases so it preempts T3 and starts its execution.
8. At t=9, T1 is released again, and it preempts T3 and executes first and at t=9.5 T3 executes its remaining part. Similarly, the execution goes on.

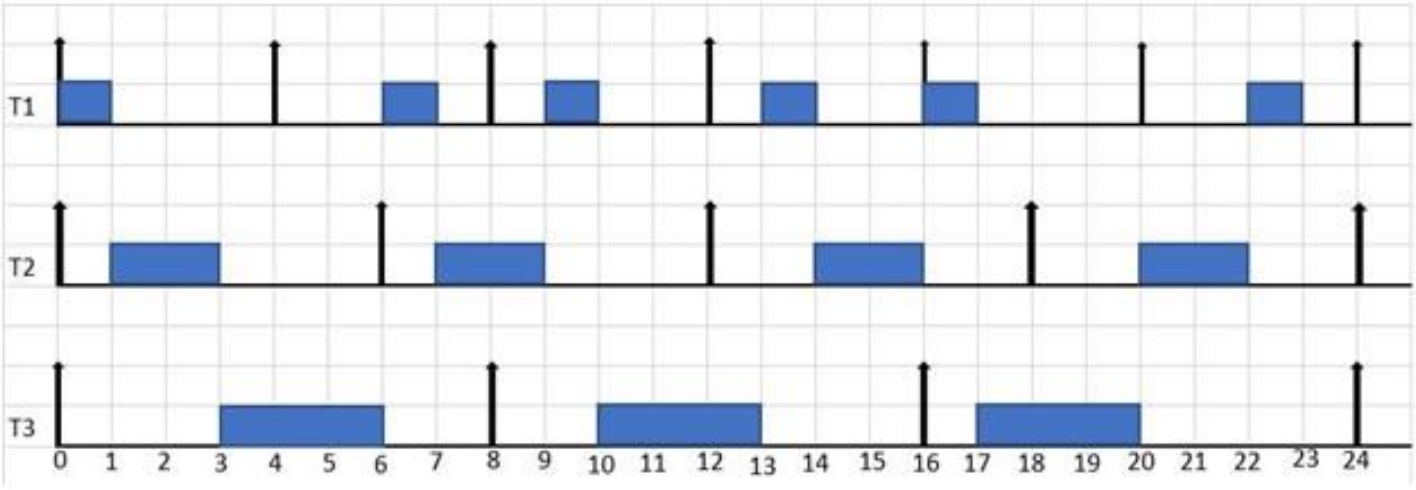
EARLIEST DEADLINE FIRST (EDF) SCHEDULING ALGORITHM

earlier deadline = highest priority

Task	Release time	Execution Time	Deadline
T1	**0**	**1**	**4**
T2	**0**	**2**	**6**
T3	**0**	**3**	**8**

The CPU utilization= 1/4 +2/6 +3/8 = 0.25 + 0.333 +0.375 = 0.95 → 95%

The timing diagram.



1. At t=0, all the tasks are released, but priorities are decided according to their deadlines so T1 has higher priority as its deadline is 4 earlier than T2 whose deadline is 6 and T3 whose deadline is 8, that's why it executes first.
2. At t=1, again deadline is compared and T2 has shorter deadline, so it executes and after that T3 starts execution but at t=4 T1 comes in the system and deadlines are compared, at this instant both T1 and T3 has same deadlines so continue to execute T3.

3. At t=6, T2 is released, now deadline of T1 is earliest than T2 so it starts execution and after that T2 begins to execute.
4. At t=8 again T1 and T2 have same deadlines i.e. t=16, so ties are broken randomly an T2 continues its execution and then T1 completes. Now at t=12 T1 and T2 come in the system simultaneously so by comparing deadlines, T1 and T2 has same deadline, and we continue to execute T3.
5. At t=13, T1 begins it execution and ends at t=14. Now T2 is the only task in the system so it completes it execution.
6. At t=16, T1 and T2 are released together, priorities are decided according to deadlines so T1 execute first as its deadline is t=20 and T3's deadline is t=24. After T1 completion T3 starts and reaches at t=17 where T2 comes in the system now by deadline comparison both have same deadline t=24 so continue to execute T3.
7. At t=20, both T1 and T2 are in the system and both have same deadline t=24 so and T2 executes. After that T1 completes it execution.

Earliest Deadline First Scheduling Algorithm

earlier deadline = highest priority.

Consider the following set of tasks in a real-time system:

Task	Arrival time	Execution time	Deadline
T1	0	4	40
T2	2	7	15
T3	5	10	20

Perform EDF scheduling on this task set.

Solution

T1 is the first task to arrive, so it starts executing.

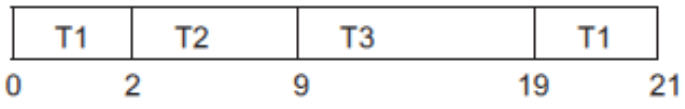
After 2 time units, T2 arrives, whose deadline is shorter than T1. Therefore, it pre-empts T1 and T2 starts executing.

After 5 time-units, T3 arrives but T2 continues, as deadline of T3 is more than T2. Therefore T2 continues and completes its execution.

After this, now there are T1 and T3 tasks that need the processor. Since T3 has a less deadline, it starts executing and completes execution.

Finally, T1 resumes and completes its execution.

The tasks on the timing diagram are as follows:



Burst time, Arrival time, Exit time, Response time, Waiting time, Turnaround time, and Throughput

These parameters are used to find the performance of a system.

Burst time

Every process in a computer system requires some amount of time for its execution. This time is both the CPU time and the I/O time. The CPU time is the time taken by CPU to execute the process. While the I/O time is the time taken by the process to perform some I/O operation. In general, we ignore the I/O time and we consider only the CPU time for a process. So, **Burst time is the total time taken by the process for its execution on the CPU.**

Arrival time

Arrival time is the time when a process enters into the ready state and is ready for its execution.

Process	Arrival time	Burst time
P1	0 ms	8 ms
P2	1 ms	7 ms
P3	2 ms	10 ms

Here in the above example, the arrival time of all the 3 processes are 0 ms, 1 ms, and 2 ms respectively.

Exit time

Exit time is the time when a process completes its execution and exit from the system.

Response time

Response time is the time spent when the process is in the ready state and gets the CPU for the first time. For example, here we are using the First Come First Serve CPU scheduling algorithm for the below 3 processes:

Process	Arrival time	Burst time
P1	0 ms	8 ms
P2	1 ms	7 ms
P3	2 ms	10 ms

Here, the response time of all the 3 processes are:

- **P1:** 0 ms
- **P2:** 7 ms because the process P2 have to wait for 8 ms during the execution of P1 and then after it will get the CPU for the first time. Also, the arrival time of P2 is 1 ms. So, the response time will be 8-1 = 7 ms.
- **P3:** 13 ms because the process P3 have to wait for the execution of P1 and P2 i.e. after 8+7 = 15 ms, the CPU will be allocated to the process P3 for the first time. Also, the arrival of P3 is 2 ms. So, the response time for P3 will be 15-2 = 13 ms.

Response time = Time at which the process gets the CPU for the first time - Arrival time

Waiting time

Waiting time is the total time spent by the process in the ready state waiting for CPU. For example, consider the arrival time of all the below 3 processes to be 0 ms, 0 ms, and 2 ms and we are using the First Come First Serve scheduling algorithm.

Process	Arrival time	Burst time
P1	0 ms	8 ms
P2	0 ms	7 ms
P3	2 ms	10 ms

Gantt Chart

P1		P2		P3	
0 ms	8 ms	8 ms	15 ms	15 ms	25 ms

Then the waiting time for all the 3 processes will be:

- **P1:** 0 ms
- **P2:** 8 ms because P2 have to wait for the complete execution of P1 and arrival time of P2 is 0 ms.
- **P3:** 13 ms becuase P3 will be executed after P1 and P2 i.e. after 8+7 = 15 ms and the arrival time of P3 is 2 ms. So, the waiting time of P3 will be: 15-2 = 13 ms.

Waiting time = Turnaround time - Burst time

In the above example, the processes have to wait only once. But in many other scheduling algorithms, the CPU may be allocated to the process for some time and then the process will be moved to the waiting state and again after some time, the process will get the CPU and so on.

There is a difference between waiting time and response time. Response time is the time spent between the ready state and getting the CPU for the first time. But the waiting time is the total time taken by the process in the ready state. Let's take an example of a round-robin scheduling algorithm. The time quantum is 2 ms.

Process	Arrival time	Burst time
P1	0 ms	4 ms
P2	0 ms	6 ms

Time Quantum = 2ms

Gantt Chart

P1	P2	P1	P2	P2
0 2	2 4	4 6	6 8	8 10

In the above example, the response time of the process P2 is 2 ms because after 2 ms, the CPU is allocated to P2 and the waiting time of the process P2 is 4 ms i.e turnaround time - burst time (10 - 6 = 4 ms).

Turnaround time

Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to its completion.

Turnaround time = Burst time + Waiting time

or

Turnaround time = Exit time - Arrival time

For example, if we take the First Come First Serve scheduling algorithm, and the order of arrival of processes is P1, P2, P3 and each process is taking 2, 5, 10 seconds.

Then the turnaround time of P1 is 2 seconds because when it comes at 0th second, then the CPU is allocated to it and so the waiting time of P1 is 0 sec and the turnaround time will be the Burst time only i.e. 2 seconds.

The turnaround time of P2 is 7 seconds because the process P2 have to wait for 2 seconds for the execution of P1 and hence the waiting time of P2 will be 2 seconds. After 2 seconds, the CPU will be given to P2 and P2 will execute its task. So, the turnaround time will be 2+5 = 7 seconds.

Similarly, the turnaround time for P3 will be 17 seconds because the waiting time of P3 is 2+5 = 7 seconds and the burst time of P3 is 10 seconds. So, turnaround time of P3 is 7+10 = 17 seconds.

Different CPU scheduling algorithms produce different turnaround time for the same set of processes. This is because the waiting time of processes differ when we change the CPU scheduling algorithm.

Throughput

Throughput is a way to find the efficiency of a CPU. It can be defined as the number of processes executed by the CPU in a given amount of time.

For example, let's say, the process P1 takes 3 seconds for execution, P2 takes 5 seconds, and P3 takes 10 seconds. So, throughput, in this case, the throughput will be (3+5+10)/3 = 18/3 = 6 seconds.

Solution:

- Actual burst time of processes: **4, 7, 8, 16**.
- a = 0.5
- Predicted burst time for 1st process = 10 ms

Predicted burst time for 2nd process

= a x Actual burst time of 1st process + (1 - a) x Predicted burst time for 1st process

= 0.5 x **4** + 0.5 x 10 = 2 + 5 = **7** ms

Actual burst time of processes : **4, 7, 8, 16**.

Predicted burst time for 3rd process

= a x Actual burst time of 2nd process + (1 - a) x Predicted burst time for 2nd process

= 0.5 x **7** + 0.5 x **7** = **7** ms

Predicted burst time for 4th process

= a x Actual burst time of 3rd process + (1 - a) x Predicted burst time for 3rd process

= 0.5 x **8** + 0.5 x **7** = **7.5** ms

Actual burst time of processes: **4, 7, 8, 16**.

Predicted burst time for 5th process

= a x Actual burst time of 4th process + (1 - a) x Predicted burst time for 4th process

= 0.5 x **16** + 0.5 x **7.5** = **11.75** ms

Xi'an Jiaotong-Liverpool University 西交利物浦大学

CPT104 - Operating Systems Concepts

CPU Scheduling I Tutorial

Length of Next CPU Burst

- Burst time for the process to be executed is taken as the average of all the processes that are executed till now.
- Can only estimate the length of Next CPU Burst .
- Can be done by using the length of previous CPU bursts, using exponential averaging.

  t_n = actual burst time of process P_n

  τ_n = predicted burst time for process P_n

  τ_{n+1} = predicted value for the next CPU burst

  a = weighing (smoothening) factor ($0 \leq a \leq 1$)

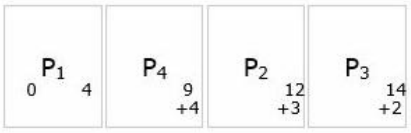
$$\tau_{n+1} = a \cdot t_n + (1 - a)\, \tau_n$$

Example

- Calculate the predicted burst time using exponential averaging for **the fifth process** if the predicted burst time for the first process is **10** ms and previous runs of the first four processes are **8, 7, 4, 16**.
- The scheduling algorithm is SJF.
- Given **a = 0.5**.

Priority Scheduling (Ex.1.)

Process	Burst Time	Priority
P1	4	1
P2	3	3
P3	2	4
P4	5	2

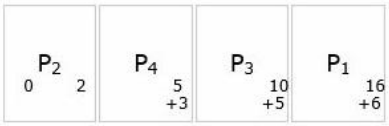


Average waiting time = = (0 + 4 + 9 + 12) / 4 = 6.25

Shortest-Job-First (SJF) Scheduling (Ex.1.)

Process	Burst Time (ms)
P1	6
P2	2
P3	5
P4	3

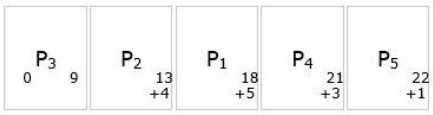
SJF scheduling Gantt chart



Average waiting time = (0 + 2 + 5 + 10) / 4 = 4.25

Priority Scheduling (Ex.2.)

Process	Burst Time	Priority
P1	5	3
P2	4	2
P3	9	1
P4	3	4
P5	1	5

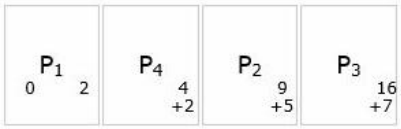


Average waiting time = = (0 + 9 + 13 + 18 + 21) / 5 = 12.2

Shortest-Job-First (SJF) Scheduling (Ex.2.)

Process	Burst Time(ms)
P1	2
P2	5
P3	7
P4	2

SJF scheduling Gantt chart



Average waiting time = (0 + 2 + 4 + 9) / 4 = 3.75

Round Robin (RR) Scheduling (Ex.1.)

Process	Burst Time
P1	10
P2	7
P3	5

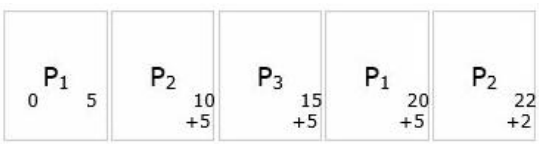
Time quantum = 5ms

$P_{1_w} = 20 - 10 = 10$

$P_{2_w} = 22 - 7 = 15$

$P_{3_w} = 15 - 5 = 10$

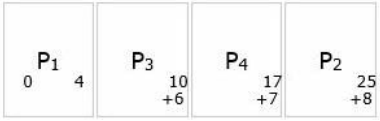
AWT = (10 + 15 + 10) / 3 = 11.67ms



Shortest-Job-First (SJF) Scheduling (Ex.3)

Process	Burst Time(ms)
P1	4
P2	8
P3	6
P4	7

SJF scheduling Gantt chart



Average waiting time = (0 + 4 + 10 + 17) / 4 = 7.75

Round Robin (RR) Scheduling (Ex.2.)

Process	Burst Time
P1	8
P2	4
P3	3

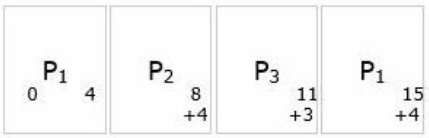
Time quantum = 4 ms

$P_{1_w} = 15 - 8 = 7$

$P_{2_w} = 8 - 4 = 4$

$P_{3_w} = 11 - 3 = 8$

$AWT = (7 + 4 + 8) / 3 = 6.33ms$



Round Robin (RR) Scheduling (Ex.3.)

Process	Burst Time
P1	5
P2	6
P3	3

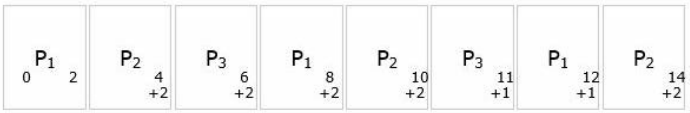
Time quantum = 2 ms

$P_{1_w} = 12 - 5 = 7$

$P_{2_w} = 14 - 6 = 8$

$P_{3_w} = 11 - 3 = 8$

$AWT = (7 + 8 + 8) / 3 = 7.66ms$



CPU Scheduling Exercises

NOTE: All time in these exercises are in msec.

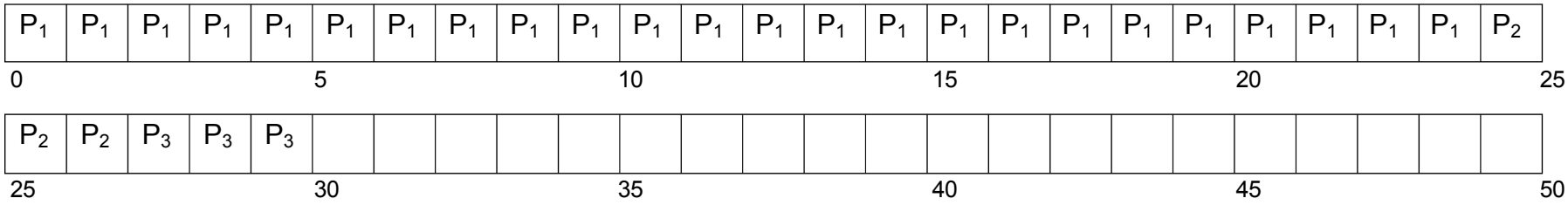
FCFS: The process that request the CPU first is allocated the CPU first.

Processes P_1, P_2, P_3 arrive at the same time, but enter the job queue in the order presented in the table.

Process	Arrive Time	Burst Time	Waiting Time
P_1	0	24	0
P_2	0	3	24
P_3	0	3	27

Time	Queue
0	**P_1**, P_2, P_3
24	**P_2**, P_3
27	**P_3**

Gantt chart:



Average waiting time:

(0 + 24 + 27) / 3 = 17 msec

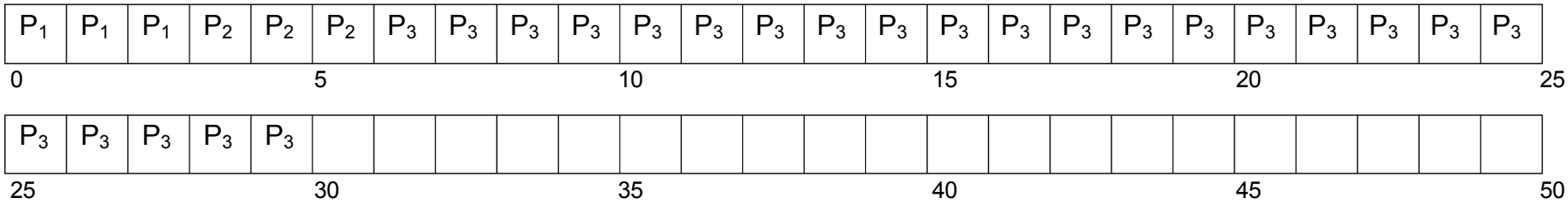
FCFS: The process that request the CPU first is allocated the CPU first.

Processes P_1, P_2, P_3 arrive at the same time, but enter the job queue in the order presented in the table.

Process	Arrive Time	Burst Time	Waiting Time
P_1	0	3	0
P_2	0	3	3
P_3	0	24	6

Time	Queue
0	**P_1**, P_2, P_3
3	**P_2**, P_3
6	**P_3**

Gantt chart:



Average waiting time:

(0 + 3 + 6) / 3 = 3 msec

Compare this example with the previous example, what do you observe?

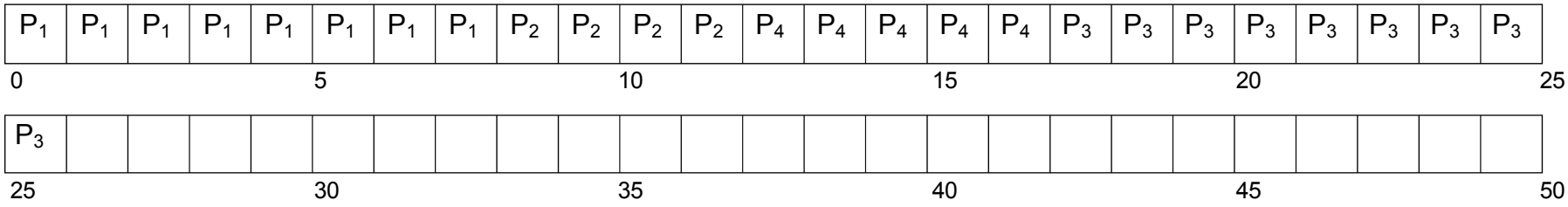
The average waiting time reduced substantially. The average waiting time under a FCFS policy is generally not minimal, and may vary substantially according to the order in the queue and the CPU-burst time.

Shortest Job First: When the CPU is available, it is assigned to the process that has the smallest CPU burst.

Process	Arrive Time	Burst Time	Waiting Time
P_1	0	8	8 – 8 = 0
P_2	1	4	12 – 4 – 1 = 7
P_3	2	9	26 – 9 – 2 = 15
P_4	3	5	17 – 5 – 3 = 9

Time	Queue
0	**P_1 = 8**
8	**P_2 = 4**, P_3 = 9, P_4 = 5
12	P_3 = 9, **P_4 = 5**
17	**P_3 = 9**

Gantt chart:



Average waiting time:

(0 + 7 + 15 + 9) / 4 = 7.75 msec

Shortest Remaining Time First : A preemptive version of the shortest job first algorithm.

Note: We use FCFS to break the tie when comparing the remaining time of the processes.

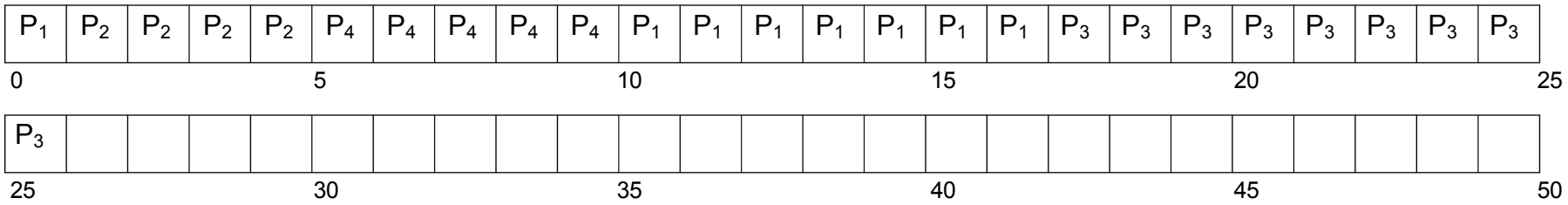
Process	Arrive Time	Burst Time	Waiting Time
P_1	0	8	17 – 8 = 9
P_2	1	4	5 – 4 – 1 = 0
P_3	2	9	26 – 9 – 2 = 15
P_4	3	5	10 – 5 – 3 = 2

Time	Queue
0	**P_1 = 8**
1	P_1 = 7, **P_2 = 4**
2	P_1 = 7, **P_2 = 3**, P_3 = 9
3	P_1 = 7, **P_2 = 2**, P_3 = 9, P_4 = 5
5	P_1 = 7, P_3 = 9, **P_4 = 5**
10	**P_1 = 7**, P_3 = 9
17	**P_3 = 9**

Average waiting time:

(9 + 0 + 15 + 2) / 4 = 6.5 msec

Gantt chart:



Compare this example with the result from the SRTF example, what do you observe?

Compare to the SJF example, we can achieve better average waiting time. Again, we see the benefits of preemptive scheduling.