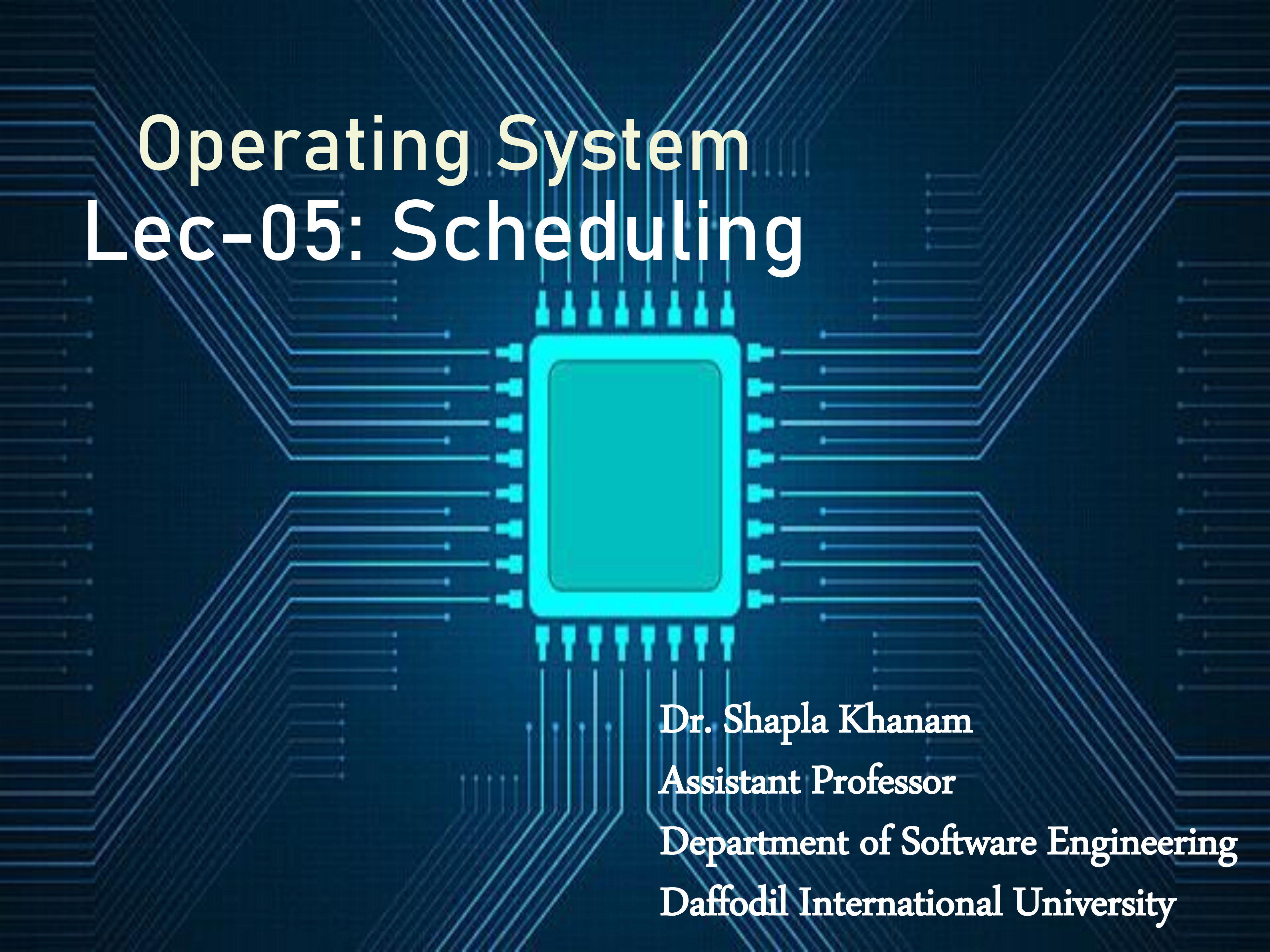
Operating System

Lec-05: CPU Scheduling

Dr. Shapla Khanam
Assistant Professor
Daffodil International University

Operating System
Lec-05: Scheduling



Dr. Shapla Khanam
Assistant Professor
Department of Software Engineering
Daffodil International University

CPU Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

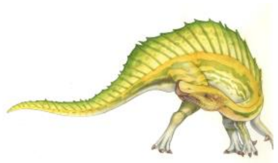
- Thread Scheduling
- Multiple-Processor Scheduling
- Real-Time CPU Scheduling
- Operating Systems Examples
- Algorithm Evaluation

Topics to Cover

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
  - First Come First Serve (FCFS)
  - Shortest-Job-First (SJF) Scheduling
  - Shortest Remaining Time
  - Priority Scheduling
  - Round Robin Scheduling
  - Multilevel Queue Scheduling
- Thread Scheduling
- Multiple-Processor Scheduling
- Real-Time CPU Scheduling
- Operating Systems Examples
- Algorithm Evaluation

Objectives

- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
- To describe various CPU-scheduling algorithms
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system
- To examine the scheduling algorithms of several operating systems



Basic Concept of CPU Scheduling

What is CPU Scheduling?

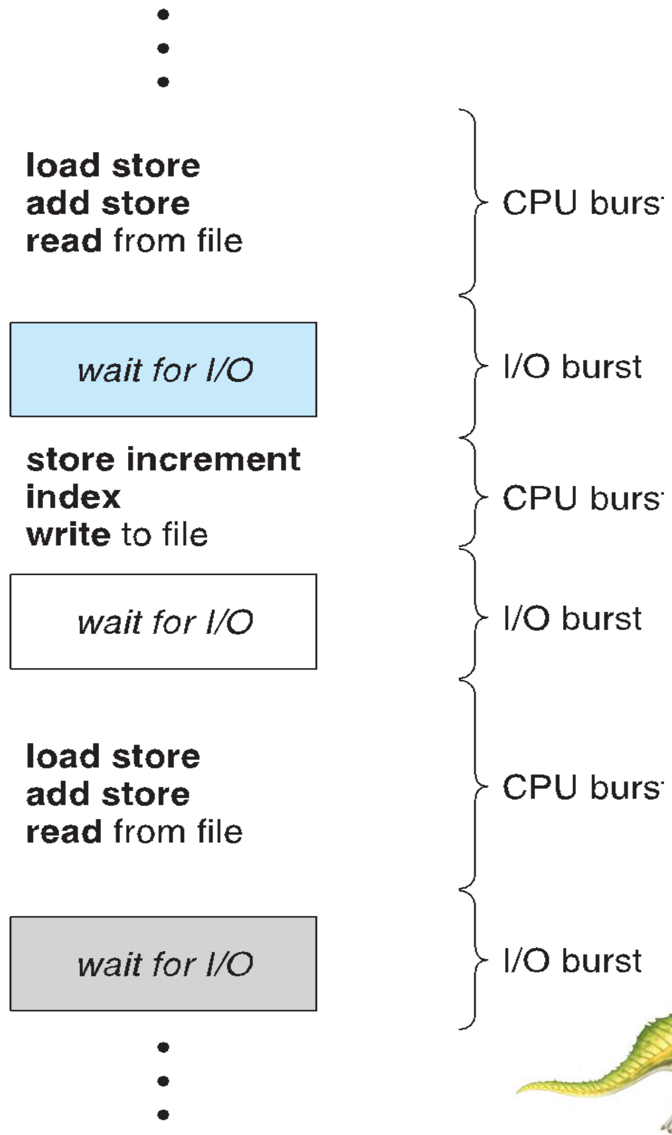
- CPU scheduling refers to the switching between processes that are being executed.
- It forms the basis of multiprogrammed systems.

Why Need CPU Scheduling?

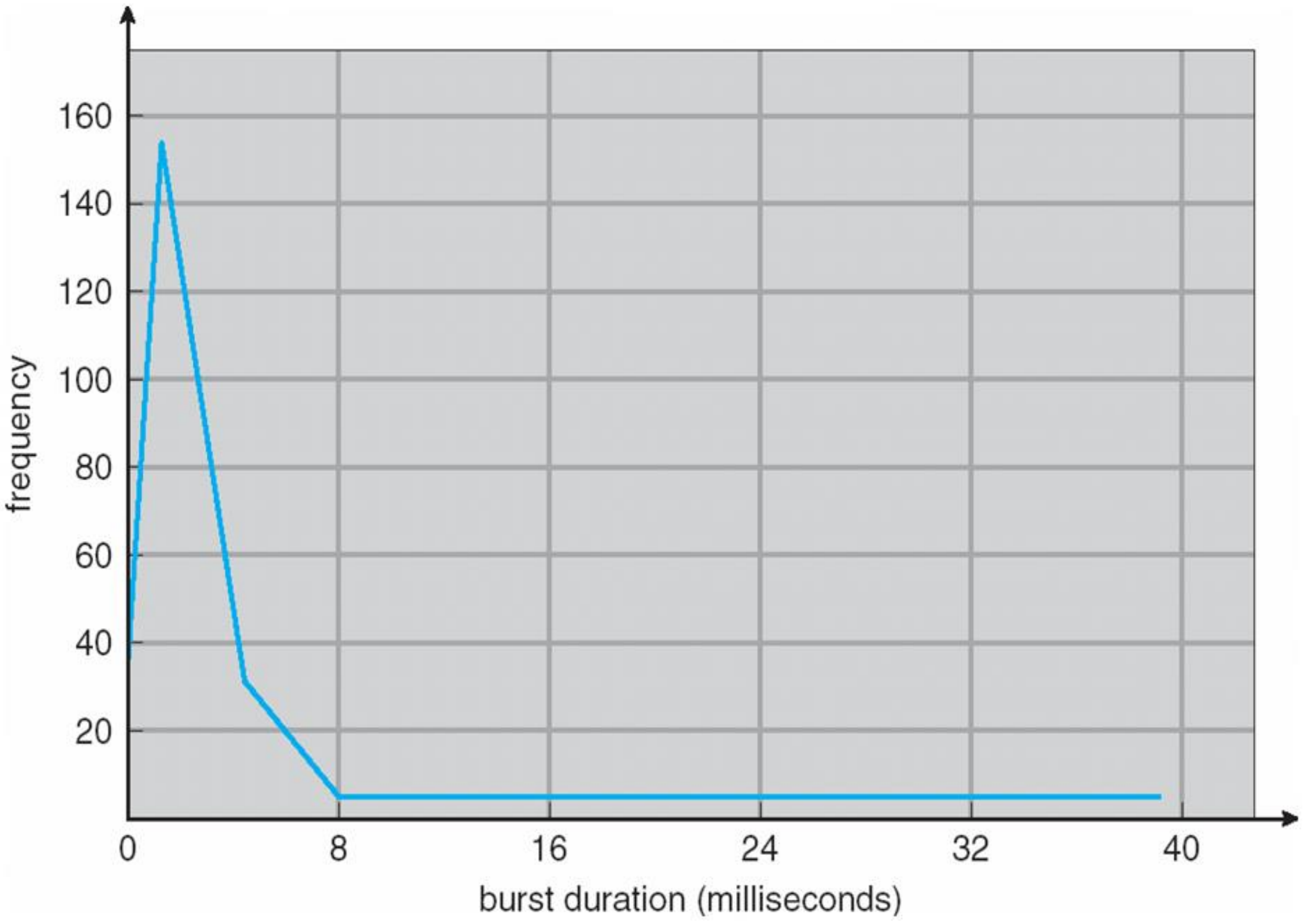
- Multiprogramming Operating Syste
- Limited CPU

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a cycle of CPU execution and I/O wait
- CPU burst followed by I/O burst
- CPU burst distribution is of main concern



Histogram of CPU-burst Times



CPU Scheduler

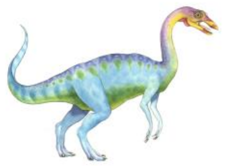
- Short-term scheduler selects from among the processes in ready queue, and allocates the CPU to one of them
  - Queue may be ordered in various ways
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates
- Scheduling under 1 and 4 is nonpreemptive
- All other scheduling is preemptive
  - Consider access to shared data
  - Consider preemption while in kernel mode
  - Consider interrupts occurring during crucial OS activities

Scheduling Category

What are the different types of CPU Scheduling Algorithms?

There are mainly two types of scheduling methods:

- **Preemptive Scheduling:** Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
- **Non-Preemptive Scheduling:** Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.



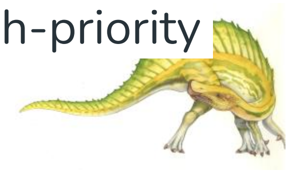
Preemptive Scheduling

Advantages

1. Because a process may not monopolize the processor, it is a more reliable method.
2. Each occurrence prevents the completion of ongoing tasks.
3. The average response time is improved.
4. Utilizing this method in a multi-programming environment is more advantageous.
5. The operating system makes sure that every process using the CPU is using the same amount of CPU time.

Disadvantages

1. Limited computational resources must be used.
2. Suspending the running process, change the context, and dispatch the new incoming process all take more time.
3. The low-priority process would have to wait if multiple high-priority processes arrived at the same time.



Non Preemptive Scheduling

Advantages

1. It has a minimal scheduling burden.
2. It is a very easy procedure.
3. Less computational resources are used.
4. It has a high throughput rate.

Disadvantages

1. Its response time to the process is super.
2. Bugs can cause a computer to freeze up.

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 location in the user program to restart that program
- Dispatch latency – time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **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
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

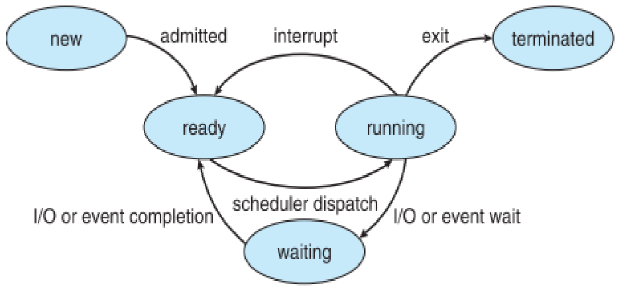


Figure 3.2 Diagram of process state.

Scheduling Algorithm Optimization Criteria

Objectives of Process Scheduling Algorithm:

- Utilization of CPU at maximum level. **Keep CPU as busy as possible**.
- **Allocation of CPU should be fair**.
- **Throughput should be Maximum**. i.e. Number of processes that complete their execution per time unit should be maximized.
- **Minimum turnaround time**, i.e. time taken by a process to finish execution should be the least.
- There should be a **minimum waiting time** and the process should not starve in the ready queue.
- **Minimum response time.** It means that the time when a process produces the first response should be as less as possible.

Scheduling Algorithm Optimization Criteria

- Max ↑ CPU utilization
- Max ↑ Throughput
- Min ↓ Turnaround time
- Min ↓ Waiting time
- Min ↓ Response time

Terminologies used in Scheduling Algorithm

- **Gantt chart –** Gantt chart is a visualization which helps to scheduling and managing particular tasks in a project. It is used while solving scheduling problems, for a concept of how the processes are being allocated in different algorithms.

- **Arrival time (AT)** – Arrival time is the time at which the process arrives in ready queue.

- **Burst time (BT) or CPU time of the process** – Burst time is the unit of time in which a particular process completes its execution.

- **Completion time (CT)** – Completion time is the time at which the process has been terminated.

- **Turn Around Time:** Time Difference between completion time and arrival time.

Terminologies of CPU Scheduling

What are the different terminologies to take care of in any CPU Scheduling algorithm?

Turn Around Time = Completion Time – Arrival Time

- **Waiting Time(W.T):** Time Difference between turn around time and burst time.

Waiting Time = Turn Around Time – Burst Time

Scheduling Algorithms

There are mainly six types of process scheduling algorithms

1. First Come First Serve (FCFS)
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time
4. Priority Scheduling
5. Round Robin Scheduling
6. Multilevel Queue Scheduling

First Come First Serve (FCFS)

FCFS Algorithm

FCFS considered to be the **simplest** of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that **requests** the CPU first is **allocated** the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

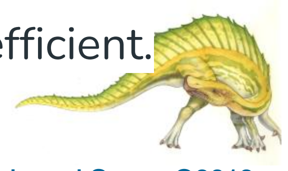
- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

- Easy to implement
- First come, first serve method

Disadvantages of FCFS:

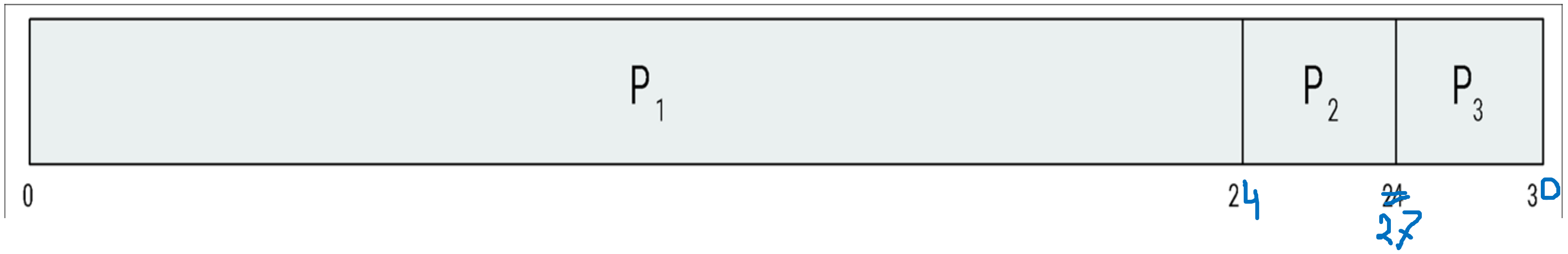
- FCFS suffers from **Convoy effect**.
- The average waiting time is much higher than the other algorithms.
- FCFS is very simple and easy to implement and hence not much efficient.



First- Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24 ms
P_2	3 ms
P_3	3 *ms*

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



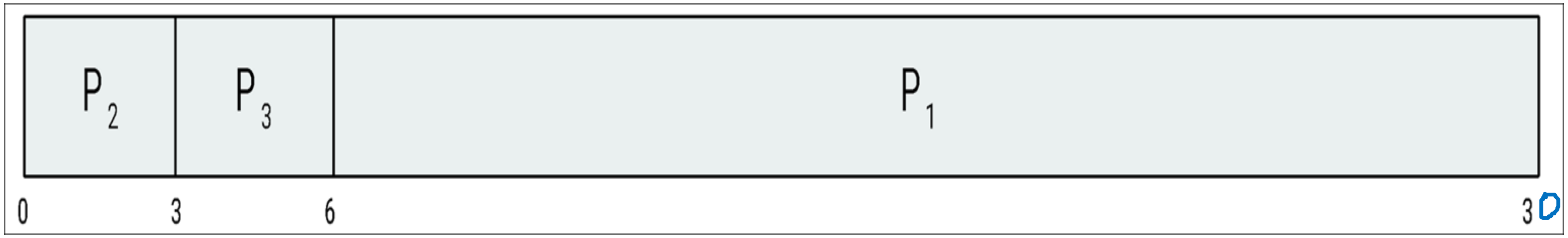
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27ms
- Average waiting time: (0 + 24 + 27)/3 = 17ms

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6$; $P_2 = 0$, $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case
- Convoy effect - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes

Algorithm1: First Come First Serve (FCFS) Example 1

Problem 1: Consider the given table below, find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Average Turn-around time and Average Waiting time with Gantt Chart if First Come First Serve (FCFS) is followed.

Process ID	Arrival time	Burst time
P1	2	2
P2	0	1
P3	2	3
P4	3	5
P5	4	5

Algorithm1: First Come First Serve (FCFS)

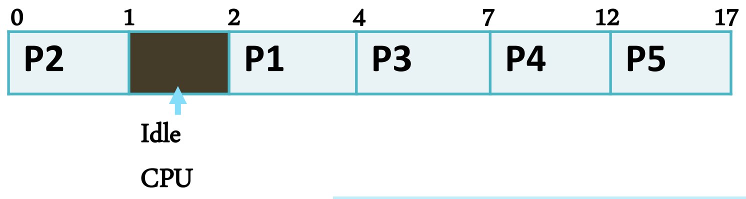
Solution:

* Turn-around time (TAT) = Completion time (CT) – Arrival time (AT)

* Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

Process ID	Arrival time(AT)	Burst time(BT)	CT	TAT=CT-AT	WT=TAT-BT
P1	2	2	4	4-2= 2	2-2= 0
P2	0	1	1	1-0= 1	1-1= 0
P3	2	3	7	7-2= 5	5-3= 2
P4	3	5	12	12-3= 9	9-5= 4
P5	4	5	17	17-4= 13	13-5= 8

Gantt Chart:

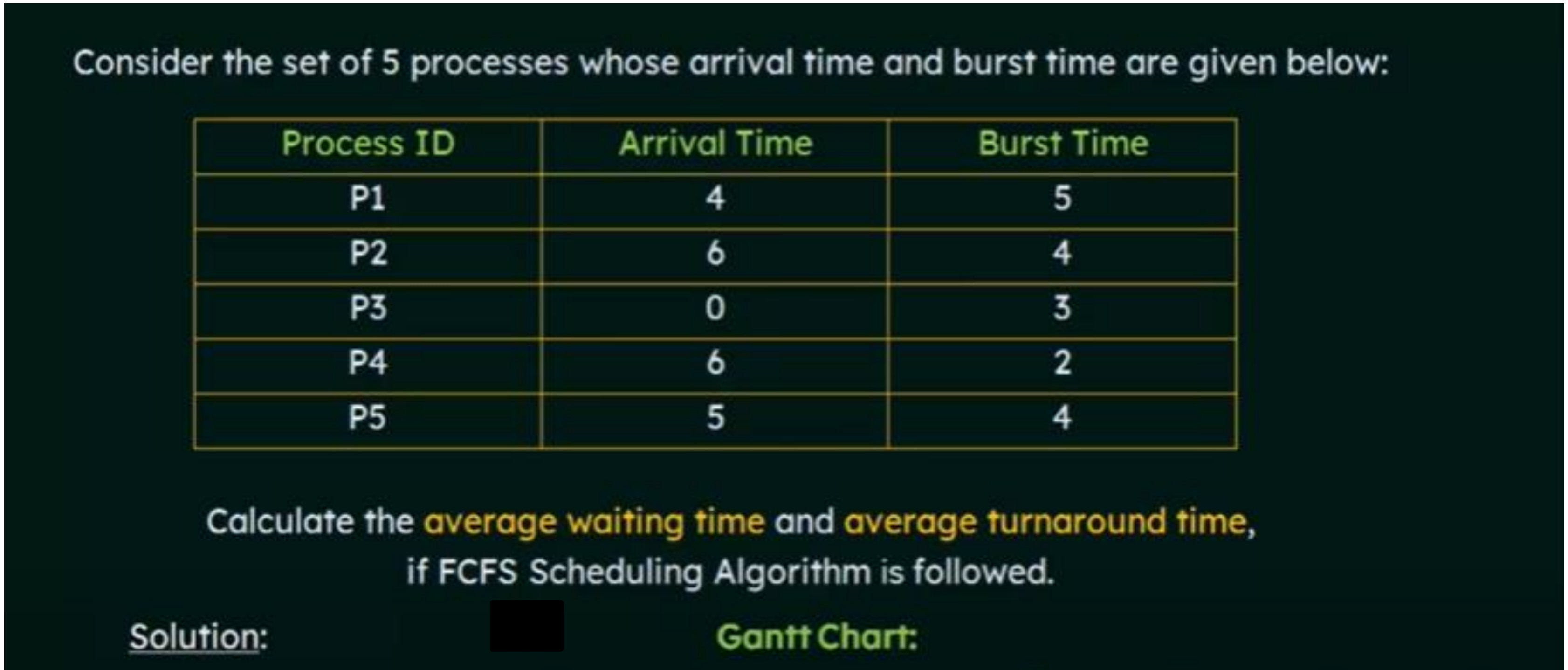


Average Waiting time = (0+0+2+4+8)/5 = 14/5 = 2.8 time unit

Average Turn-around time = (2+1+5+9+13)/5 = 30/5 = 6 time unit

Algorithm1: First Come First Serve (FCFS) Example 2

Turn Around Time = Completion Time - Arrival Time
Waiting Time = Turn Around Time - Burst Time

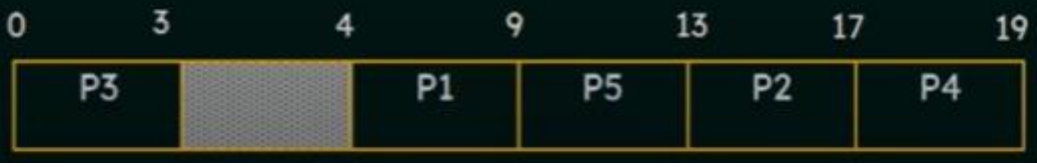


Consider the set of 5 processes whose arrival time and burst time are given below:

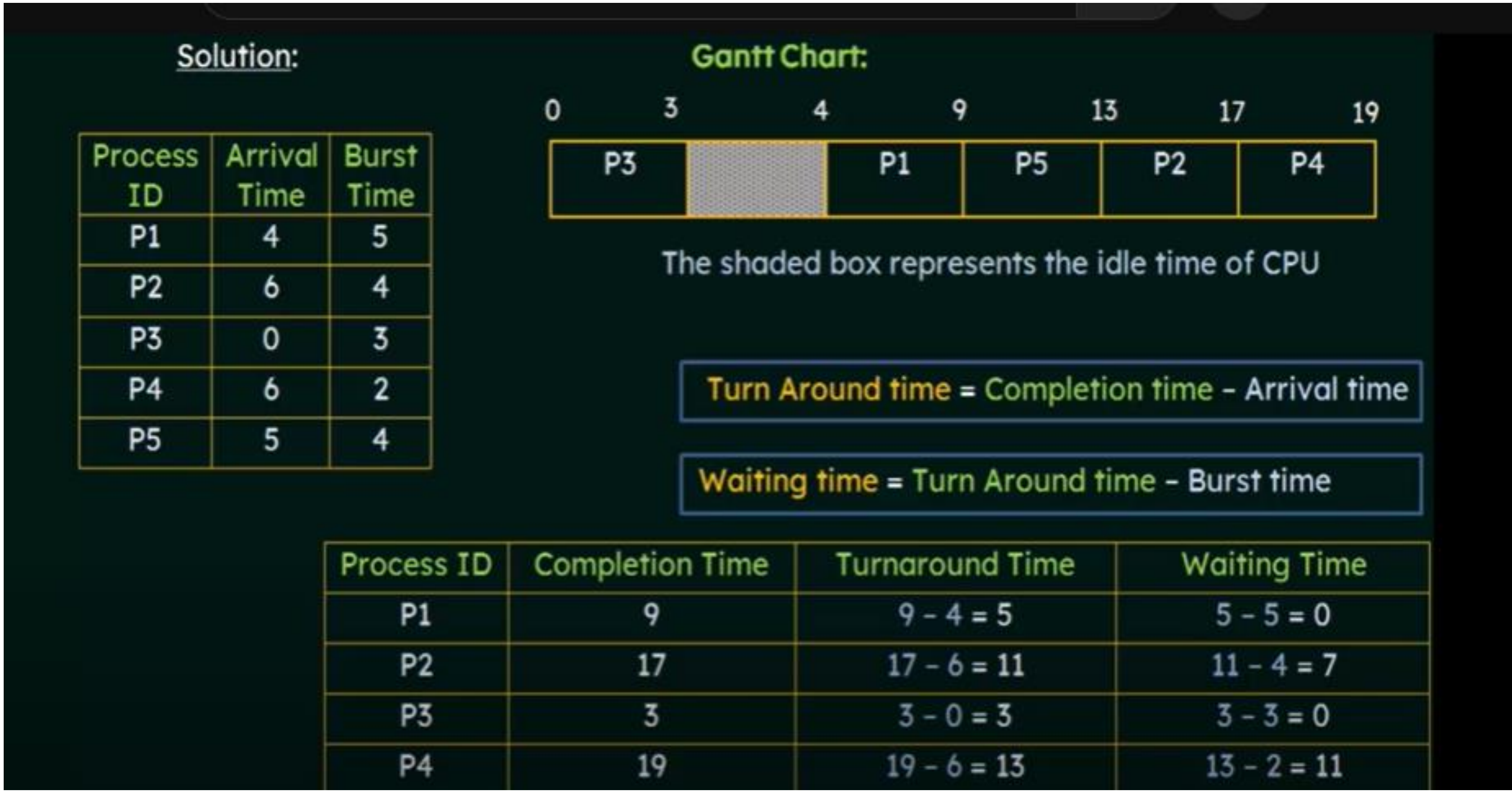
Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Calculate the average waiting time and average turnaround time,
if FCFS Scheduling Algorithm is followed.

Solution: [redacted] Gantt Chart:



Algorithm1: First Come First Serve (FCFS) Example 2 cont..



Solution:

Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Gantt Chart:

0–3	3–4	4–9	9–13	13–17	17–19
P3	▨	P1	P5	P2	P4

The shaded box represents the idle time of CPU

Turn Around time = Completion time - Arrival time

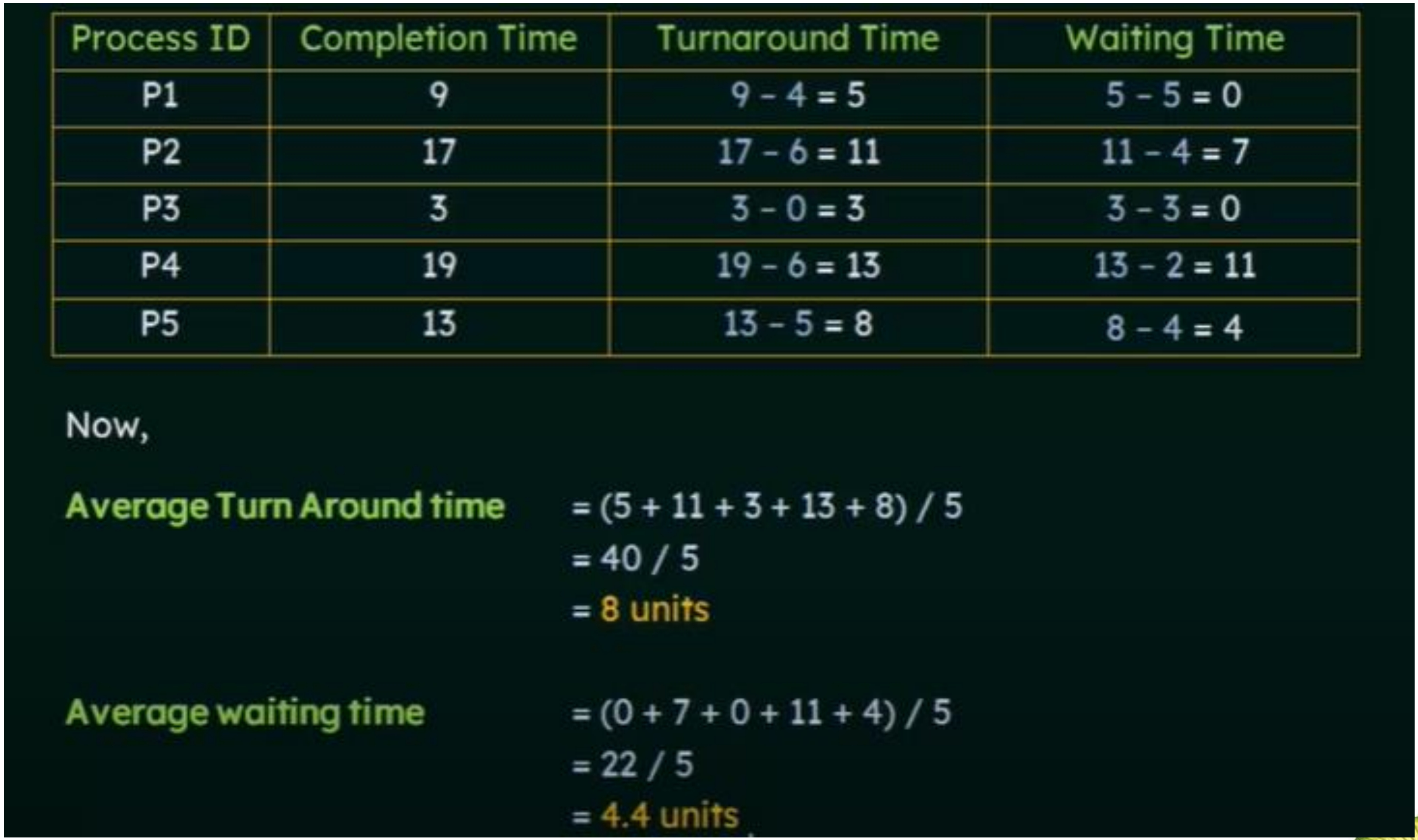
Waiting time = Turn Around time - Burst time

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 - 4 = 5	5 - 5 = 0
P2	17	17 - 6 = 11	11 - 4 = 7
P3	3	3 - 0 = 3	3 - 3 = 0
P4	19	19 - 6 = 13	13 - 2 = 11

How about P5

How to compute Average waiting time?

Algorithm1: First Come First Serve (FCFS) Example 2 cont..



Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 - 4 = 5	5 - 5 = 0
P2	17	17 - 6 = 11	11 - 4 = 7
P3	3	3 - 0 = 3	3 - 3 = 0
P4	19	19 - 6 = 13	13 - 2 = 11
P5	13	13 - 5 = 8	8 - 4 = 4

Now,

Average Turn Around time = (5 + 11 + 3 + 13 + 8) / 5
= 40 / 5
= 8 units

Average waiting time = (0 + 7 + 0 + 11 + 4) / 5
= 22 / 5
= 4.4 units

Shortest-Job-First (SJF)

Algorithm2: Shortest-Job-First (SJF) Scheduling

- **Shortest job first (SJF)** is a scheduling process that selects the waiting process with the **smallest execution time** to execute next. scheduling method may or may not be preemptive.

Advantages of Shortest Job first:

- As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
- SJF is generally used for long term scheduling

Disadvantages of SJF:

- One of the demerit SJF has is **starvation.**
- Many times it becomes complicated to predict the length of the upcoming CPU request

Shortest-Job-First (SJF) Scheduling

- **Shortest job first (SJF)** is a scheduling process that selects the waiting process with the **smallest execution time** to execute next.
- This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.

Characteristics of SJF:

- Shortest Job first has the advantage of having a minimum average waiting time among all **operating system scheduling algorithms.**
- It is associated with each task as a unit of time to complete.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

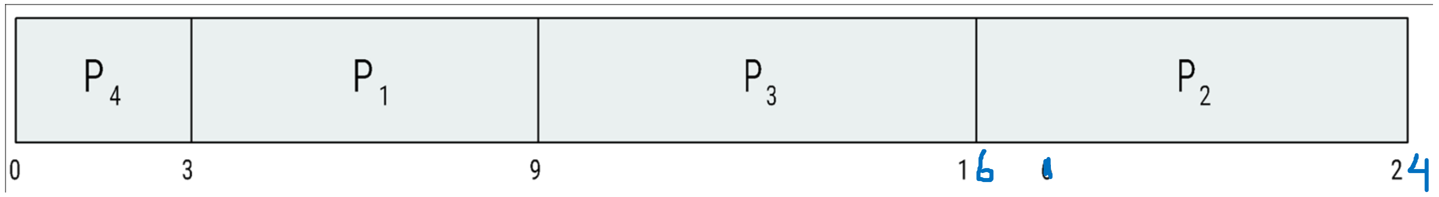
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Algorithm2: Shortest-Job-First (SJF) Scheduling Example 1

Problem 1: Consider the given table below, find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Average Turn-around time and Average Waiting time with Gantt Chart if Shortest-Job-First (SJF) is followed.

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Algorithm2: Shortest-Job-First (SJF) Scheduling Example 1

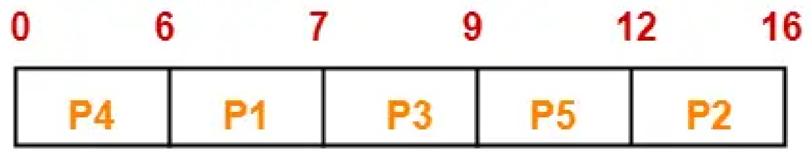
Solution:

* Turn-around time (TAT) = Completion time (CT) – Arrival time (AT)

* Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

Process Id	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT
P1	3	1	7	7 – 3 = 4	4 – 1 = 3
P2	1	4	16	16 – 1 = 15	15 – 4 = 11
P3	4	2	9	9 – 4 = 5	5 – 2 = 3
P4	0	6	6	6 – 0 = 6	6 – 6 = 0
P5	2	3	12	12 – 2 = 10	10 – 3 = 7

Gantt Chart:



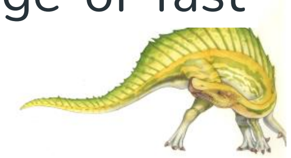
•Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit
•Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit

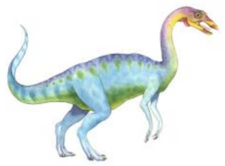
Algorithm 3 Shortest Remaining Time First

Shortest remaining time first is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to the job closest to completion. In SRTF the process with the smallest amount of time remaining until completion is selected to execute.

Characteristics of Shortest remaining time first:

- SRTF algorithm makes the processing of the jobs faster than SJF algorithm, given it's overhead charges are not counted.
- The context switch is done a lot more times in SRTF than in SJF and consumes the CPU's valuable time for processing. This adds up to its processing time and diminishes its advantage of fast processing.





SRTF

Advantages of SRTF:

- In SRTF the short processes are handled very fast.
- The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.

Disadvantages of SRTF:

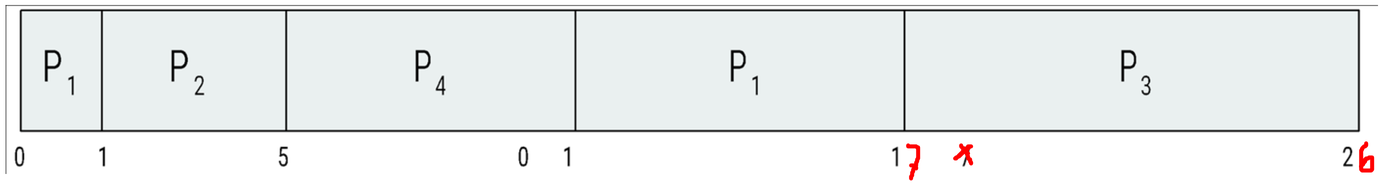
- Like the shortest job first, it also has the potential for process starvation.
- Long processes may be held off indefinitely if short processes are continually added.

Example of Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis

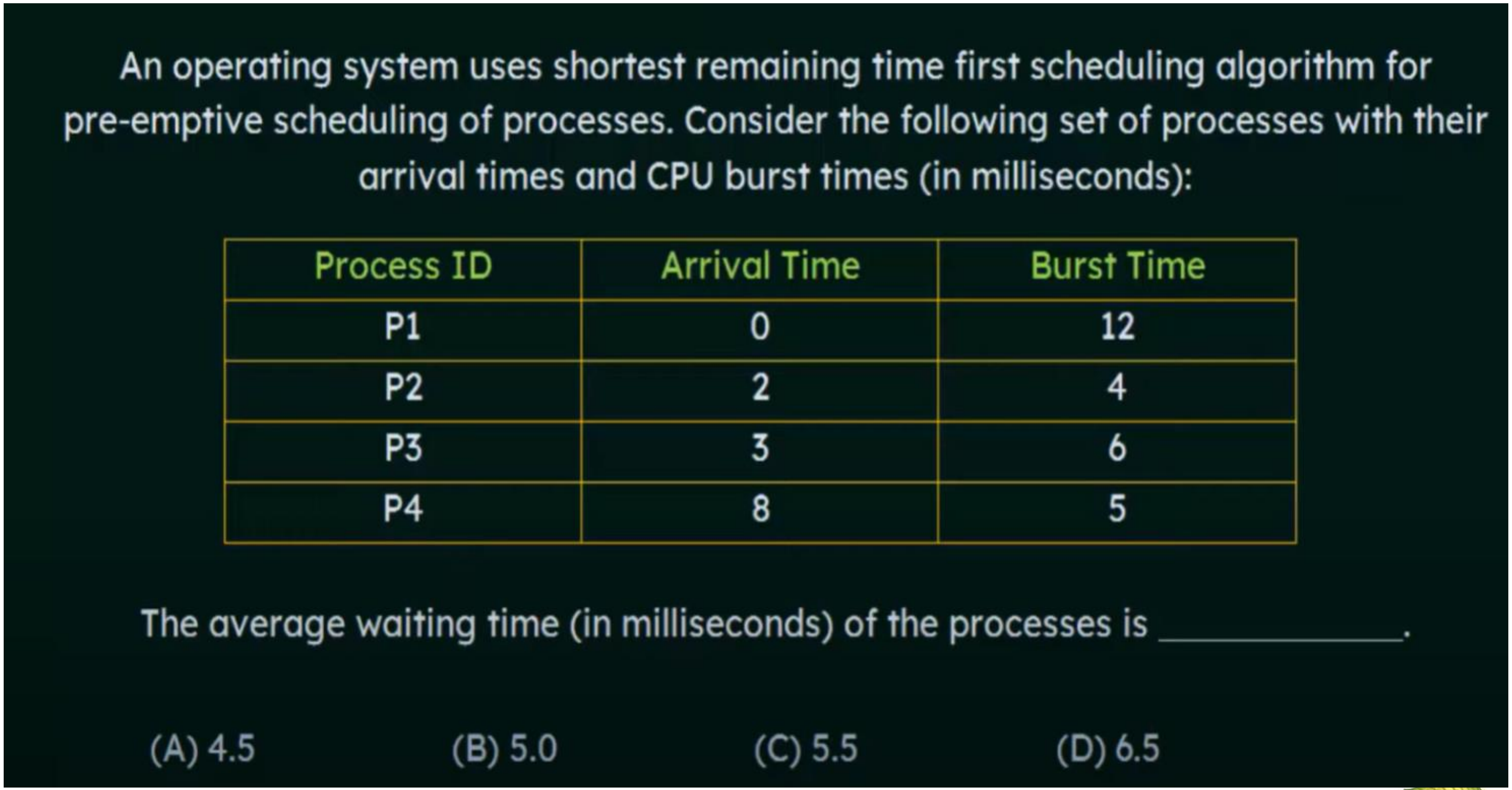
Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

Example of SRTF (preempted)

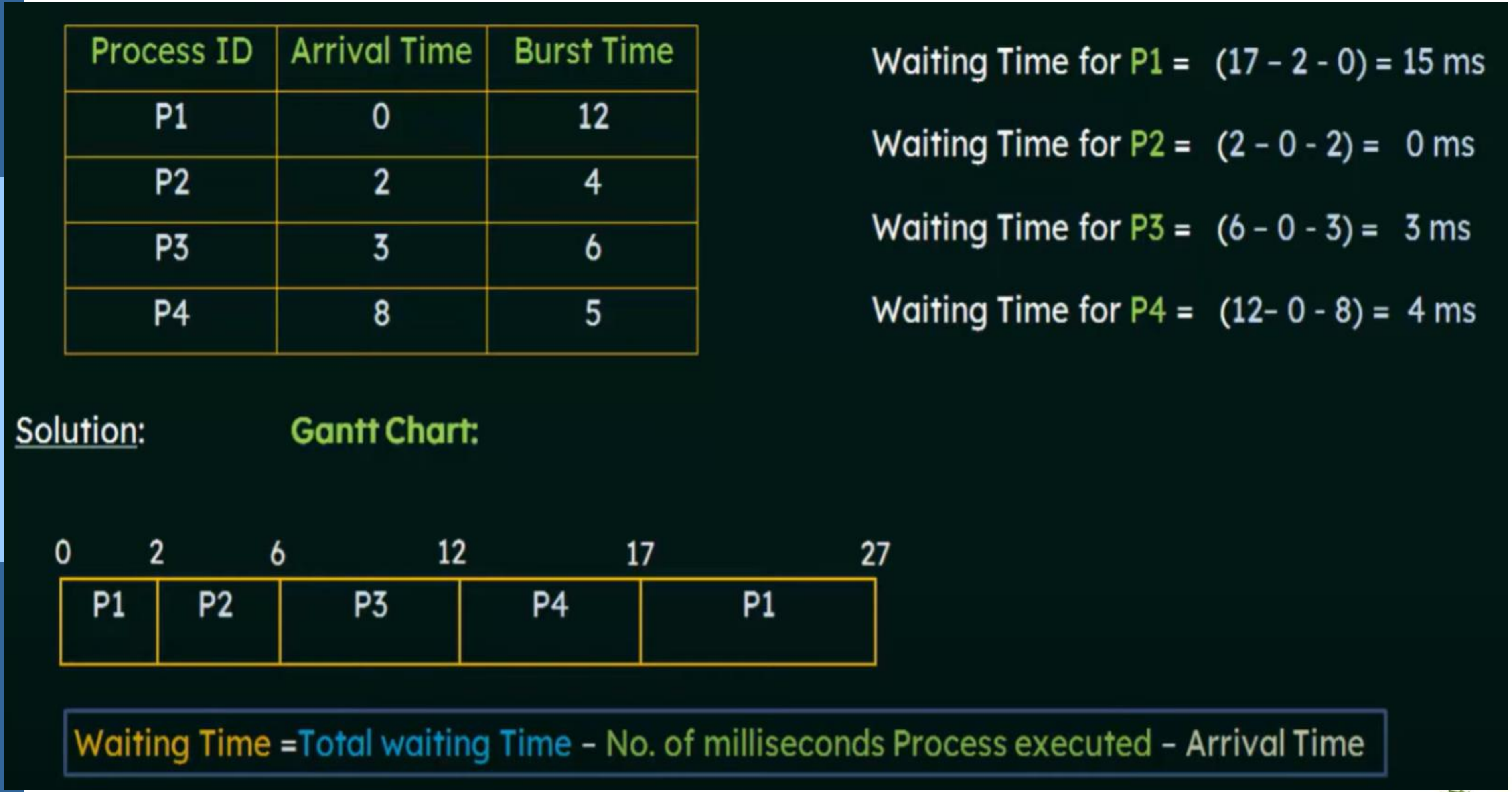

An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process ID	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

The average waiting time (in milliseconds) of the processes is _____________.

(A) 4.5 (B) 5.0 (C) 5.5 (D) 6.5

Solution preempted



Process ID	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

Waiting Time for P1 = (17 – 2 - 0) = 15 ms

Waiting Time for P2 = (2 – 0 - 2) = 0 ms

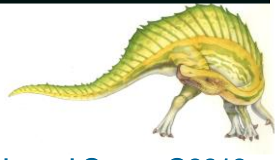
Waiting Time for P3 = (6 – 0 - 3) = 3 ms

Waiting Time for P4 = (12– 0 - 8) = 4 ms

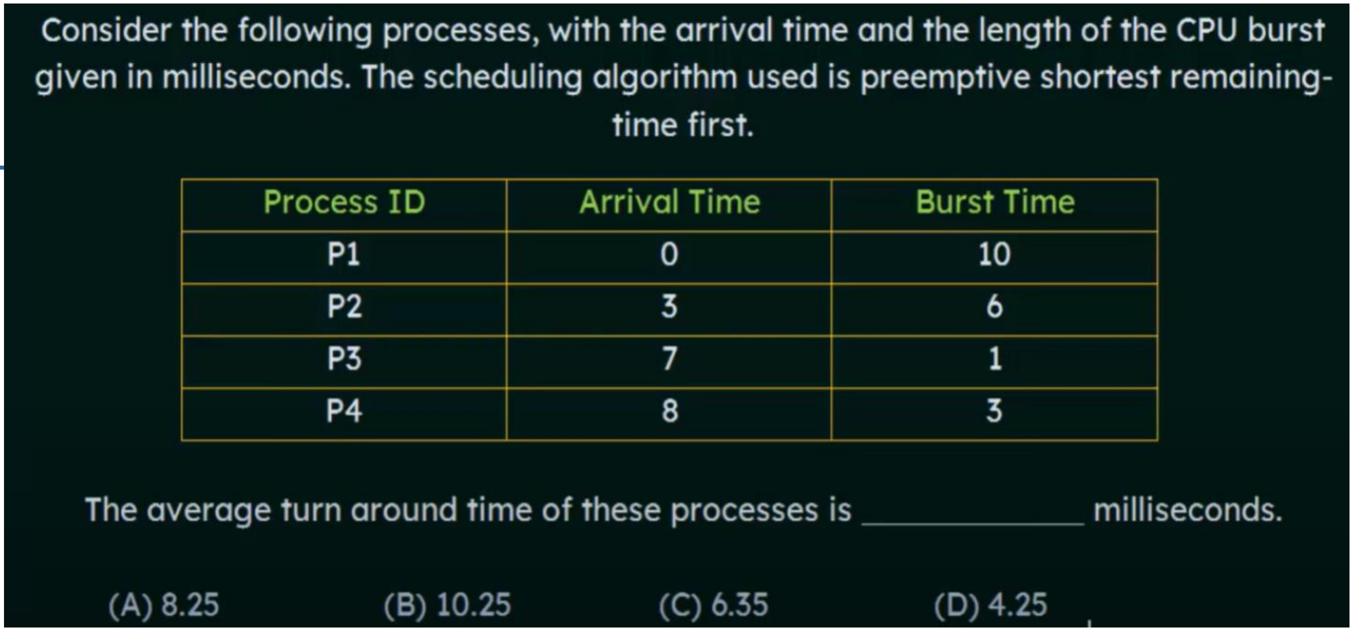
Solution: Gantt Chart:

0	2	6	12	17	27
P1	P2	P3	P4	P1	

Waiting Time = Total waiting Time – No. of milliseconds Process executed – Arrival Time



Example 2
SRTF
preempted

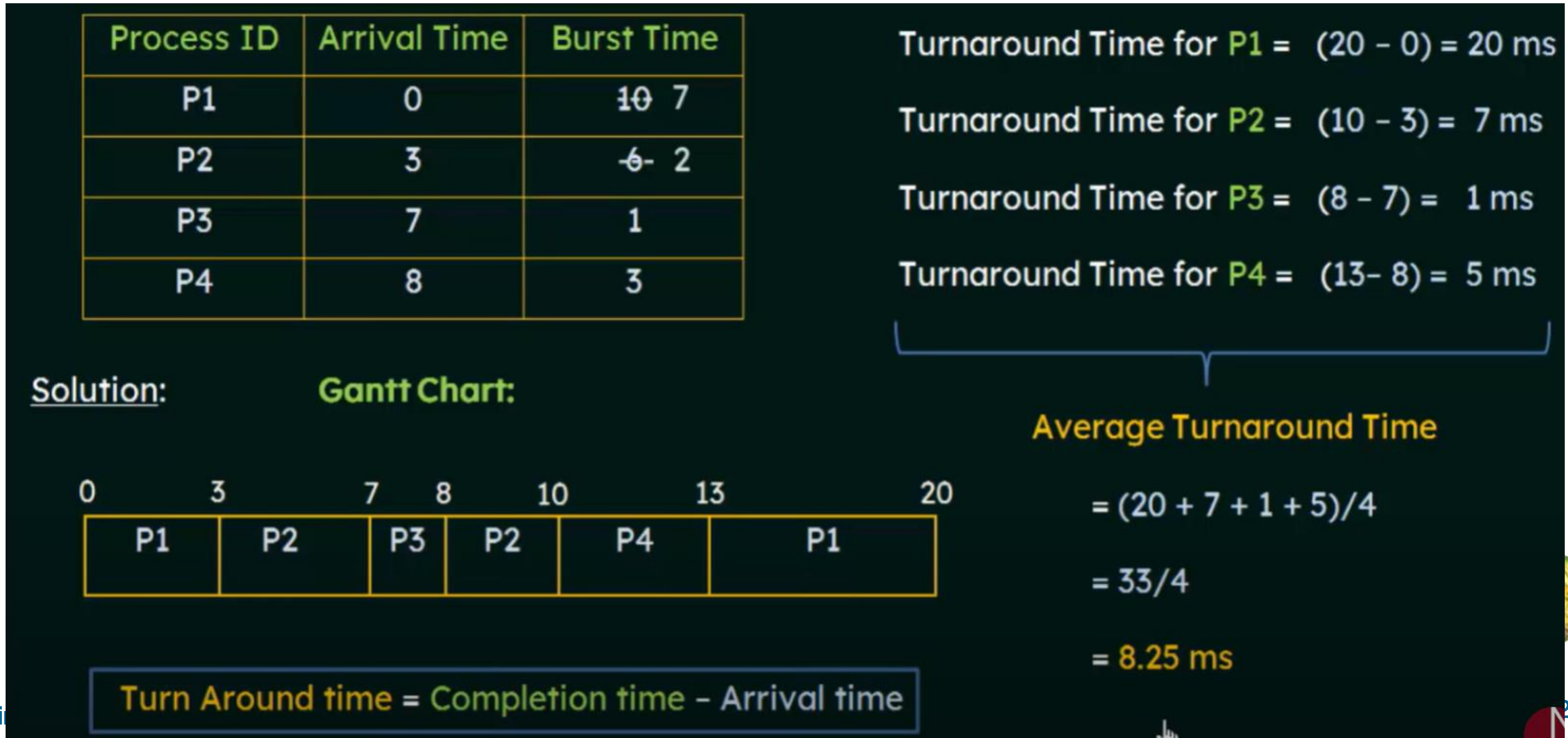


Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is preemptive shortest remaining-time first.

Process ID	Arrival Time	Burst Time
P1	0	10
P2	3	6
P3	7	1
P4	8	3

The average turn around time of these processes is ____________ milliseconds.

(A) 8.25 (B) 10.25 (C) 6.35 (D) 4.25



Process ID	Arrival Time	Burst Time
P1	0	~~10~~ 7
P2	3	~~6~~ 2
P3	7	1
P4	8	3

Turnaround Time for P1 = (20 – 0) = 20 ms

Turnaround Time for P2 = (10 – 3) = 7 ms

Turnaround Time for P3 = (8 – 7) = 1 ms

Turnaround Time for P4 = (13– 8) = 5 ms

Solution: Gantt Chart:

0–3	3–7	7–8	8–10	10–13	13–20
P1	P2	P3	P2	P4	P1

Average Turnaround Time

= (20 + 7 + 1 + 5)/4

= 33/4

= 8.25 ms

Turn Around time = Completion time – Arrival time

Determining Length of Next CPU Burst

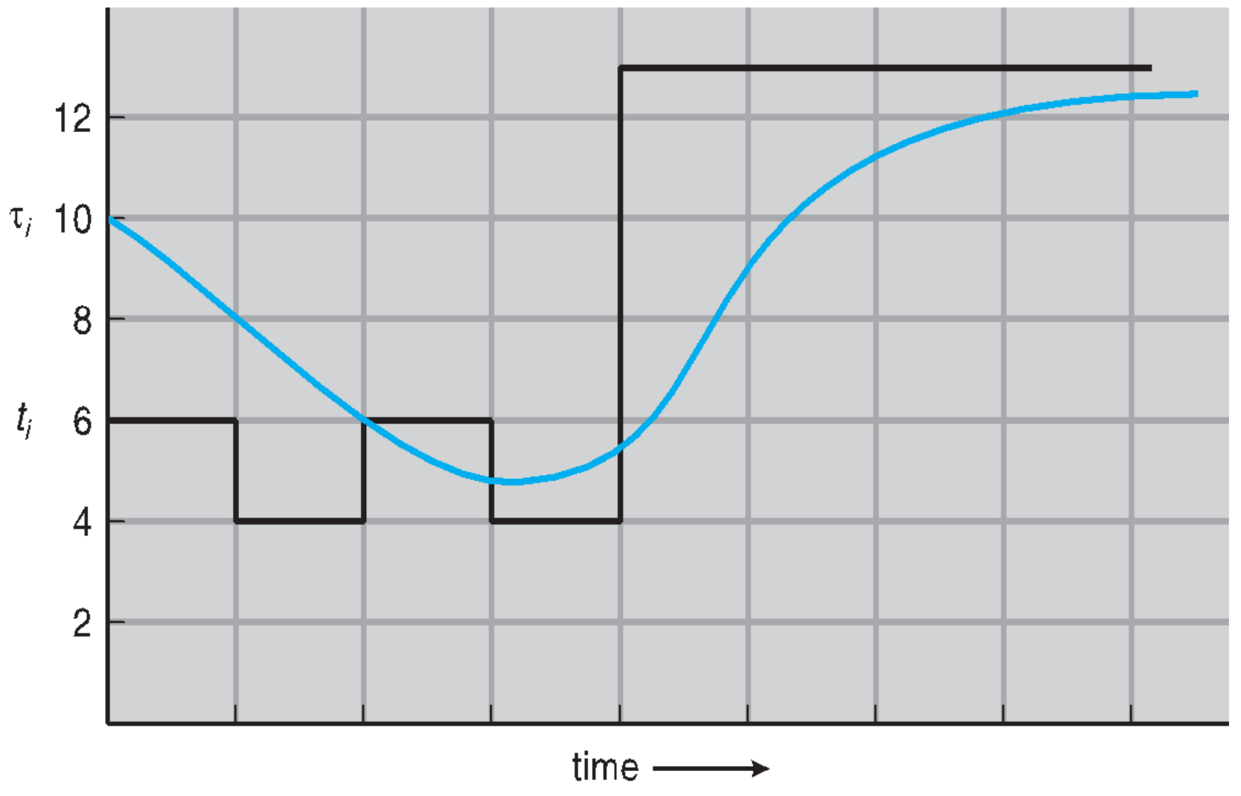
- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst

- Can be done by using the length of previous CPU bursts, using exponential averaging

  1. $t_n =$ actual length of n^{th} CPU burst
  2. $\tau_{n+1} =$ predicted value for the next CPU burst
  3. $\alpha, 0 \leq \alpha \leq 1$
  4. Define: $\tau_{n=1} = \alpha\ t_n + (1 - \alpha)\tau_n.$

- Commonly, α set to ½
- Preemptive version called shortest-remaining-time-first

Prediction of the Length of the Next CPU Burst



CPU burst (t_i)		6	4	6	4	13	13	13	…
"guess" (τ_i)	10	8	6	6	5	9	11	12	…

Examples of Exponential Averaging

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:

$$\begin{aligned}\tau_{n+1} = \alpha\, t_n &+ (1 - \alpha)\alpha\, t_{n-1} + \dots \\ &+ (1 - \alpha)^j \alpha\, t_{n-j} + \dots \\ &+ (1 - \alpha)^{n+1} \tau_0\end{aligned}$$

- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

Priority Scheduling

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - Nonpreemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem ≡ Starvation – low priority processes may never execute
- Solution ≡ Aging – as time progresses increase the priority of the process

Priority Scheduling

- **Preemptive Priority CPU Scheduling Algorithm** is a pre-emptive method of **CPU scheduling algorithm** that works **based on the priority** of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first.

- In the case of any conflict, that is, where there are more than one processor with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.

- A priority number (integer) is associated with each process. The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - Nonpreemptive

Priority Scheduling

Characteristics of Priority Scheduling:

- Schedules tasks based on priority.
- When the higher priority work arrives while a task with less priority is executed, the higher priority work takes the place of the less priority one and
- The latter is suspended until the execution is complete.
- Lower is the number assigned, higher is the priority level of a process.

Advantages of Priority Scheduling:

- The average waiting time is less than FCFS
- Less complex

Disadvantages of Priority Scheduling:

- One of the most common demerits of the Preemptive priority CPU scheduling algorithm is the **Starvation** Problem. This is the problem in which a process has to wait for a longer amount of time to get scheduled into the CPU. This condition is called the starvation problem.

Algorithm 4: Priority Scheduling Example 1

Problem 1: Consider the given table below, find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Average Turn-around time and Average Waiting time with Gantt Chart if Priority Scheduling is followed.

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

Algorithm 4: Priority Scheduling Example 1

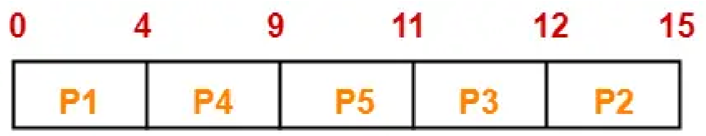
Solution:

* Turn-around time (TAT) = Completion time (CT) – Arrival time

* Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

Process Id	Arrival time	Burst time	Priority	CT	Turn Around time	Waiting time
P1	0	4	2	4	4 – 0 = 4	4 – 4 = 0
P2	1	3	3	15	15 – 1 = 14	14 – 3 = 11
P3	2	1	4	12	12 – 2 = 10	10 – 1 = 9
P4	3	5	5	9	9 – 3 = 6	6 – 5 = 1
P5	4	2	5	11	11 – 4 = 7	7 – 2 = 5

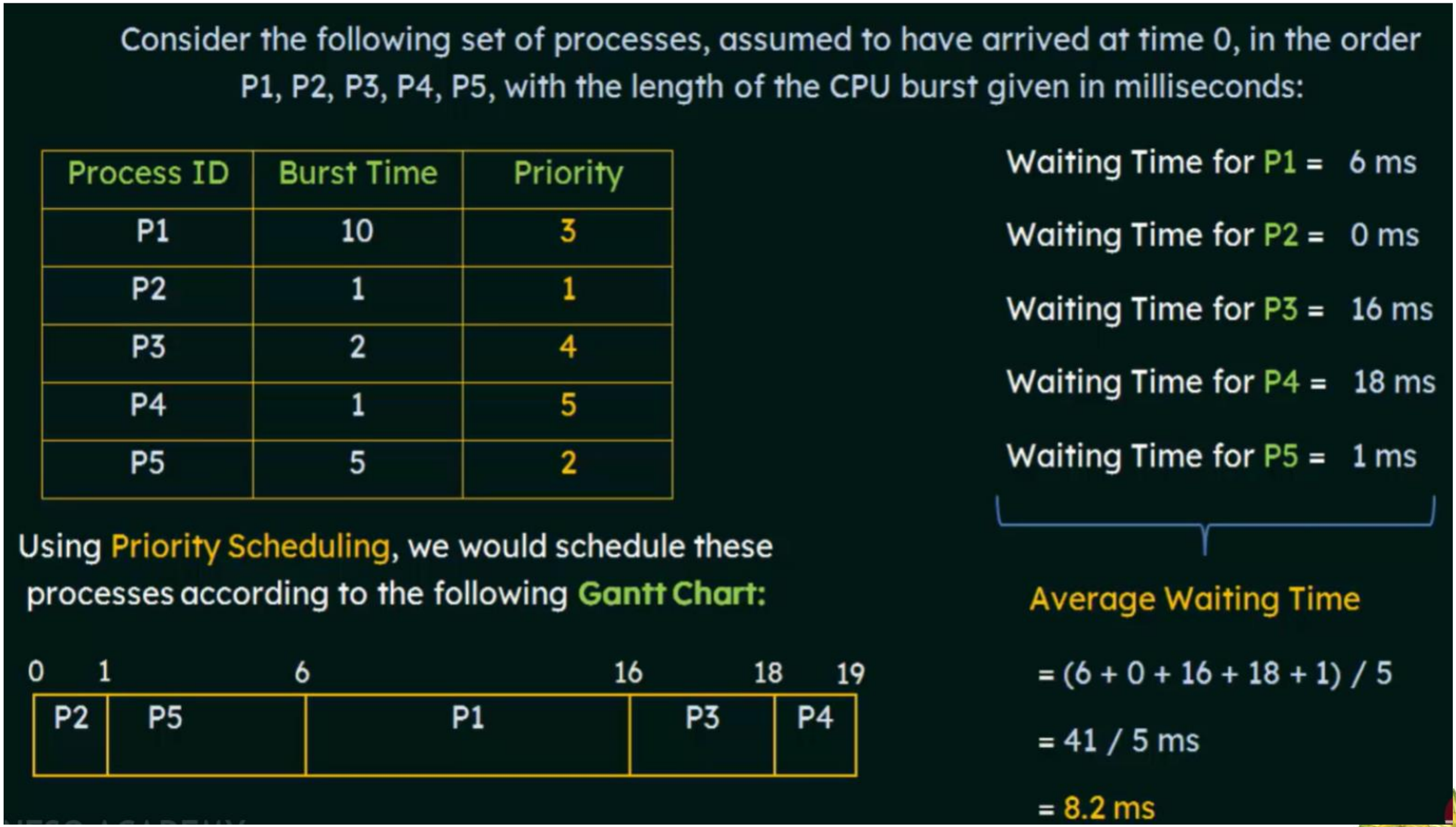
Gantt Chart:



*largest integer ~ highest priority

•Average Turn Around time = (4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = 8.2 unit
•Average waiting time = (0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = 5.2 unit

Priority Scheduling example 2



Consider the following set of processes, assumed to have arrived at time 0, in the order P1, P2, P3, P4, P5, with the length of the CPU burst given in milliseconds:

Process ID	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Using Priority Scheduling, we would schedule these processes according to the following Gantt Chart:

```
0    1             6                     16         18     19
| P2 |     P5      |         P1          |    P3    |  P4  |
```

Waiting Time for P1 = 6 ms

Waiting Time for P2 = 0 ms

Waiting Time for P3 = 16 ms

Waiting Time for P4 = 18 ms

Waiting Time for P5 = 1 ms

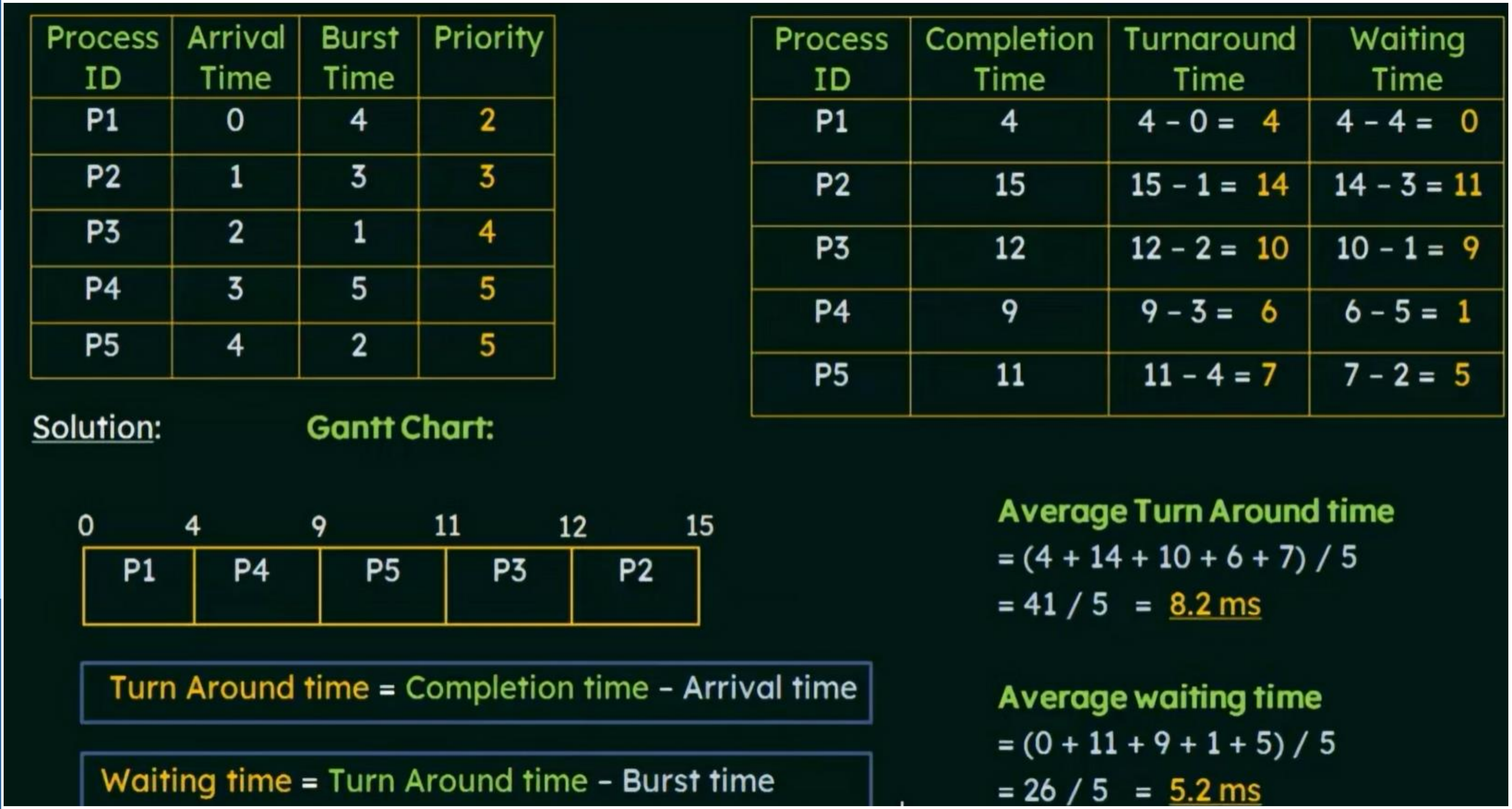
Average Waiting Time

= (6 + 0 + 16 + 18 + 1) / 5

= 41 / 5 ms

= 8.2 ms

Priority Scheduling example 3



Process ID	Arrival Time	Burst Time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	4	4 - 0 = 4	4 - 4 = 0
P2	15	15 - 1 = 14	14 - 3 = 11
P3	12	12 - 2 = 10	10 - 1 = 9
P4	9	9 - 3 = 6	6 - 5 = 1
P5	11	11 - 4 = 7	7 - 2 = 5

Solution: Gantt Chart:

0 – 4	4 – 9	9 – 11	11 – 12	12 – 15
P1	P4	P5	P3	P2

Turn Around time = Completion time - Arrival time

Waiting time = Turn Around time - Burst time

Average Turn Around time
= (4 + 14 + 10 + 6 + 7) / 5
= 41 / 5 = 8.2 ms

Average waiting time
= (0 + 11 + 9 + 1 + 5) / 5
= 26 / 5 = 5.2 ms

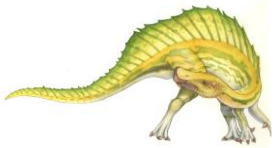
Priority Scheduling examples for practice

https://www.guru99.com/priority-scheduling-program.html

https://www.geeksforgeeks.org/priority-cpu-scheduling-with-different-arrival-time-set-2/

https://www.geeksforgeeks.org/preemptive-priority-cpu-scheduling-algortithm/

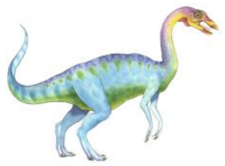
https://www.javatpoint.com/os-preemptive-priority-scheduling



Round Robin (RR) Scheduling

Round Robin (RR)

- Each process gets a small unit of CPU time (time quantum *q*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than $(n-1)q$ time units.
- Timer interrupts every quantum to schedule next process
- Performance
  - *q* large ⇒ FIFO
  - *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high



RR

Characteristics of Round robin:

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
- One of the most widely used methods in CPU scheduling as a core.
- It is considered preemptive as the processes are given to the CPU for a very limited time.

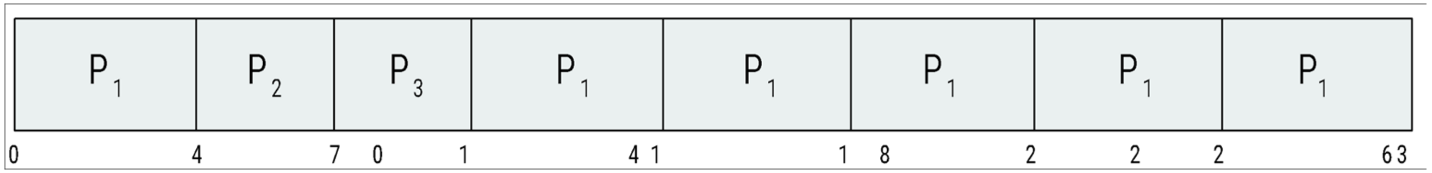
Advantages of Round robin:

- Round robin seems to be fair as every process gets an equal share of CPU.
- The newly created process is added to the end of the ready queue.

Example of RR with Time Quantum = 4

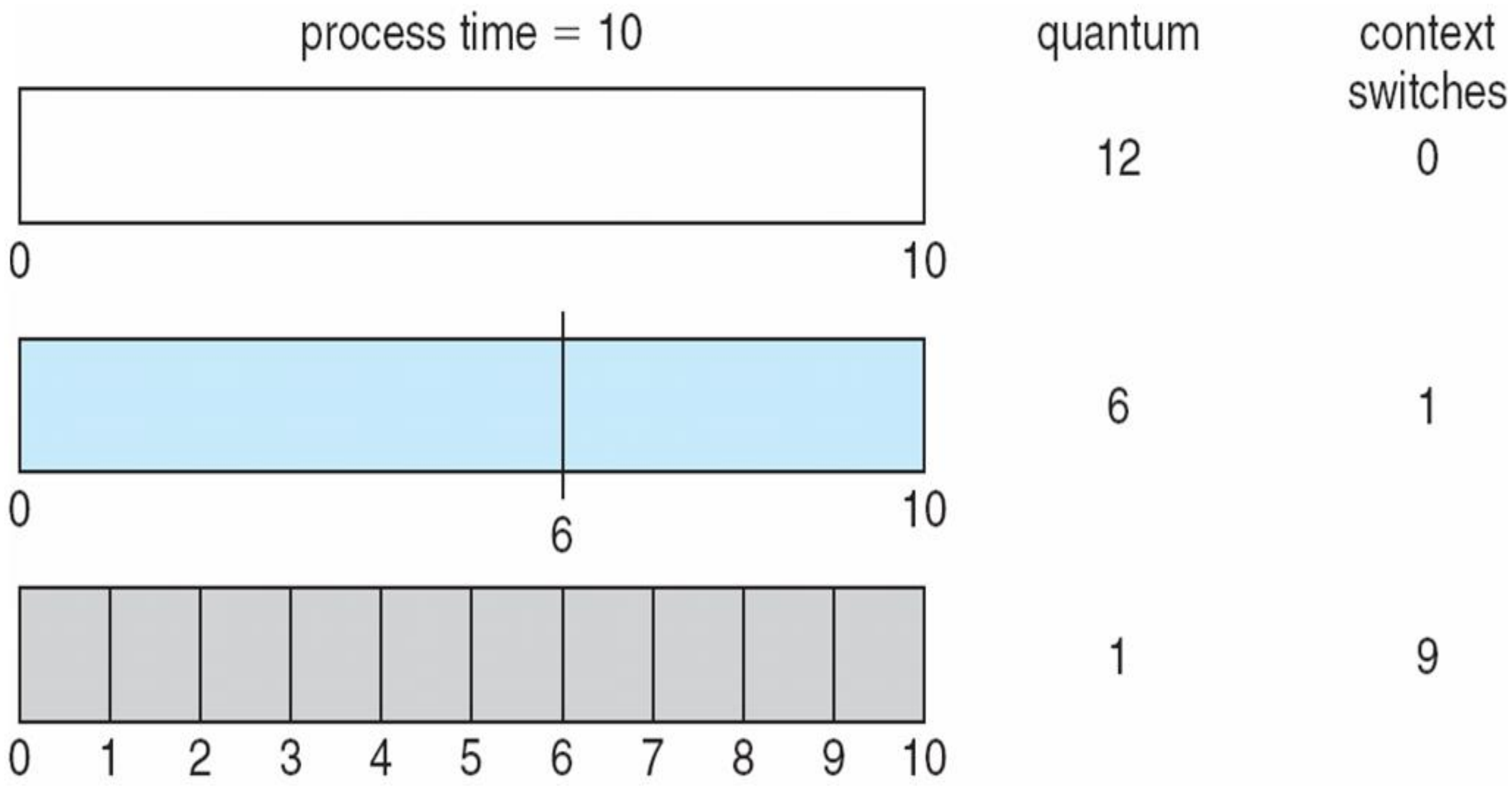
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:

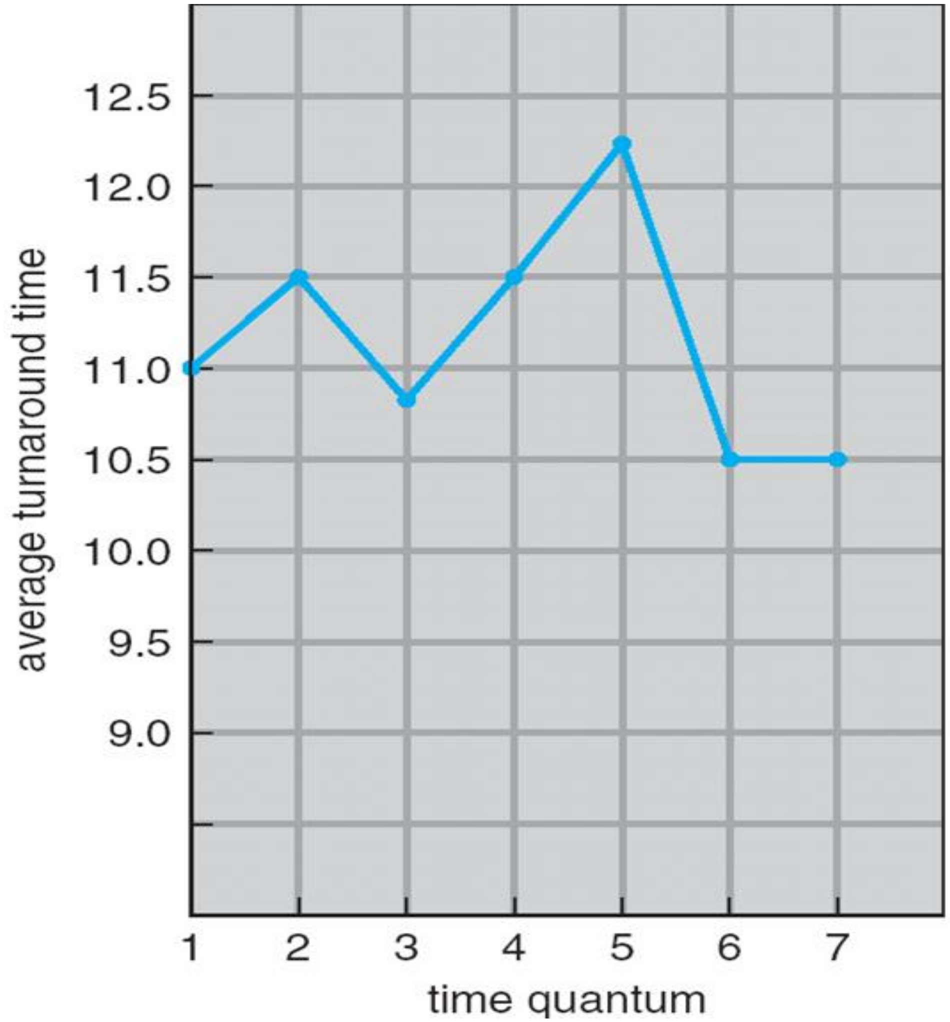


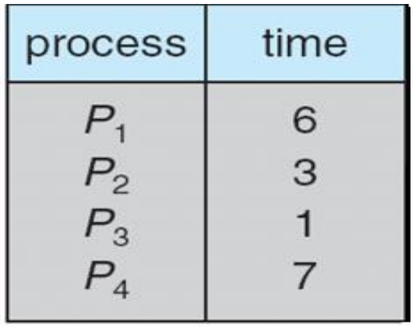
- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

Time Quantum and Context Switch Time



Turnaround Time Varies With The Time Quantum





process	time
P_1	6
P_2	3
P_3	1
P_4	7

80% of CPU bursts
should be shorter than q

Algorithm 5: Round Robin Example 1

Problem 1: Apply the Round Robin CPU Scheduling algorithm (time quantum = 6) considering the scenario and calculate the average waiting time and average turn-around time with the Gantt chart.

Process Id	Arrival time	Burst time
P1	0	8
P2	1	5
P3	2	10
P4	3	11

Algorithm 5: Round Robin Example 1

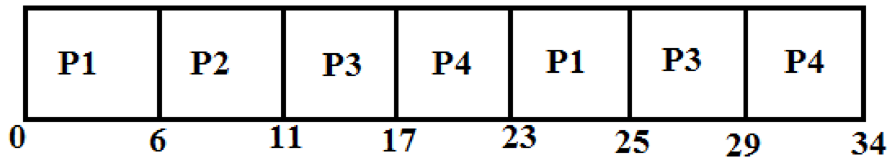
Solution:

* Turn-around time (TAT) = Completion time (CT) – Arrival time (AT)

* Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

Process Id	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT
P1	0	8	25	25	17
P2	1	5	11	10	5
P3	2	10	29	27	17
P4	3	11	34	31	20

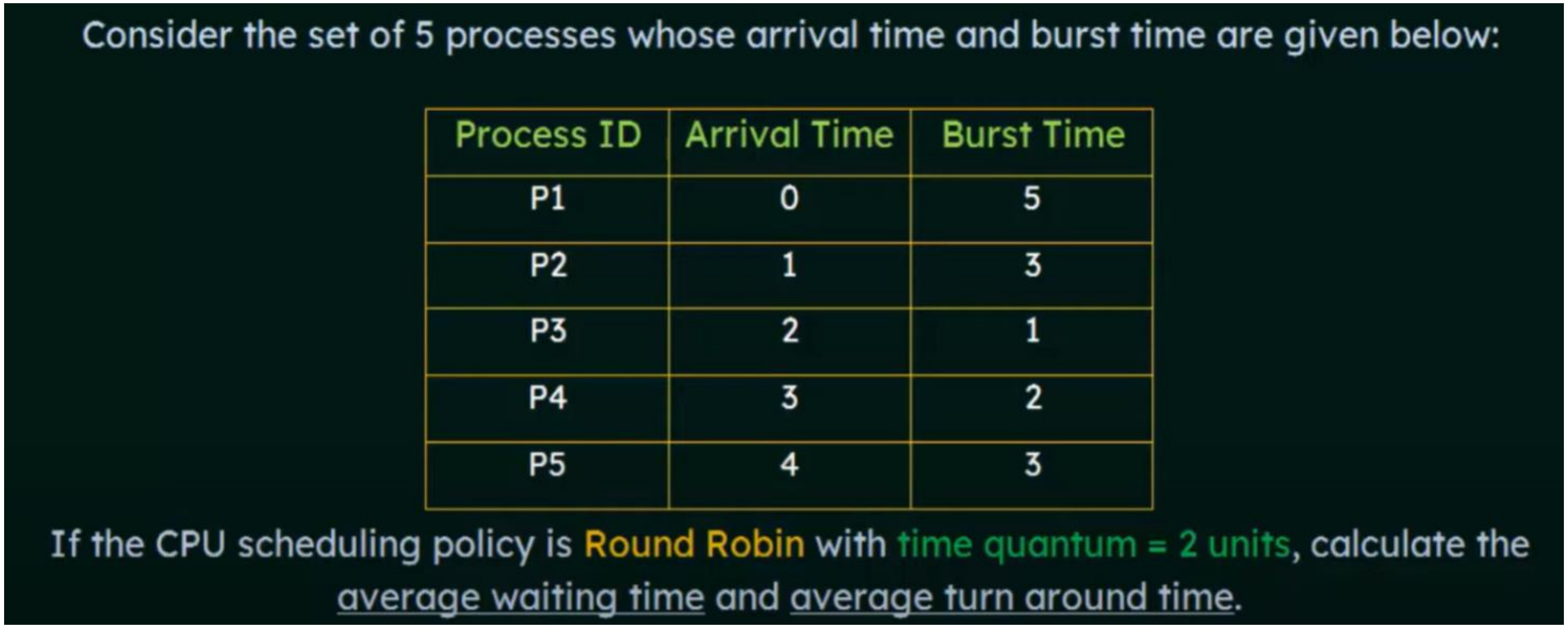
Gantt chart-



*quantum q = 6

•**Average Turn Around time =** (25+10+27+31)/4 = 23.25
•**Average waiting time =** (17+5+17+20)/4 = 59/4 = 14.75

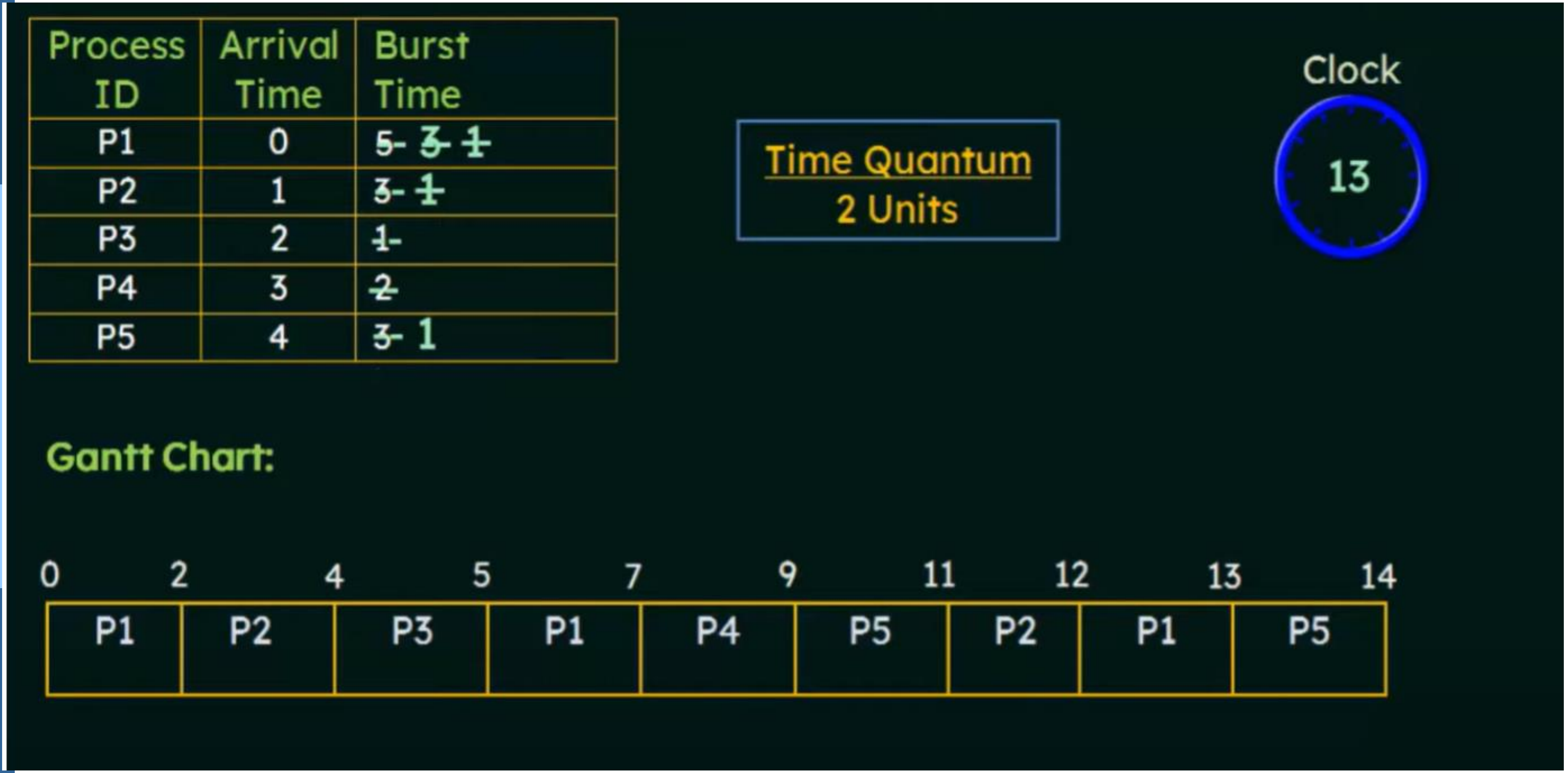
Round robin example 2



Consider the set of 5 processes whose arrival time and burst time are given below:

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

If the CPU scheduling policy is Round Robin with time quantum = 2 units, calculate the average waiting time and average turn around time.



Process ID	Arrival Time	Burst Time
P1	0	~~5~~- ~~3~~- ~~1~~
P2	1	~~3~~- ~~1~~
P3	2	~~1~~-
P4	3	~~2~~
P5	4	~~3~~- 1

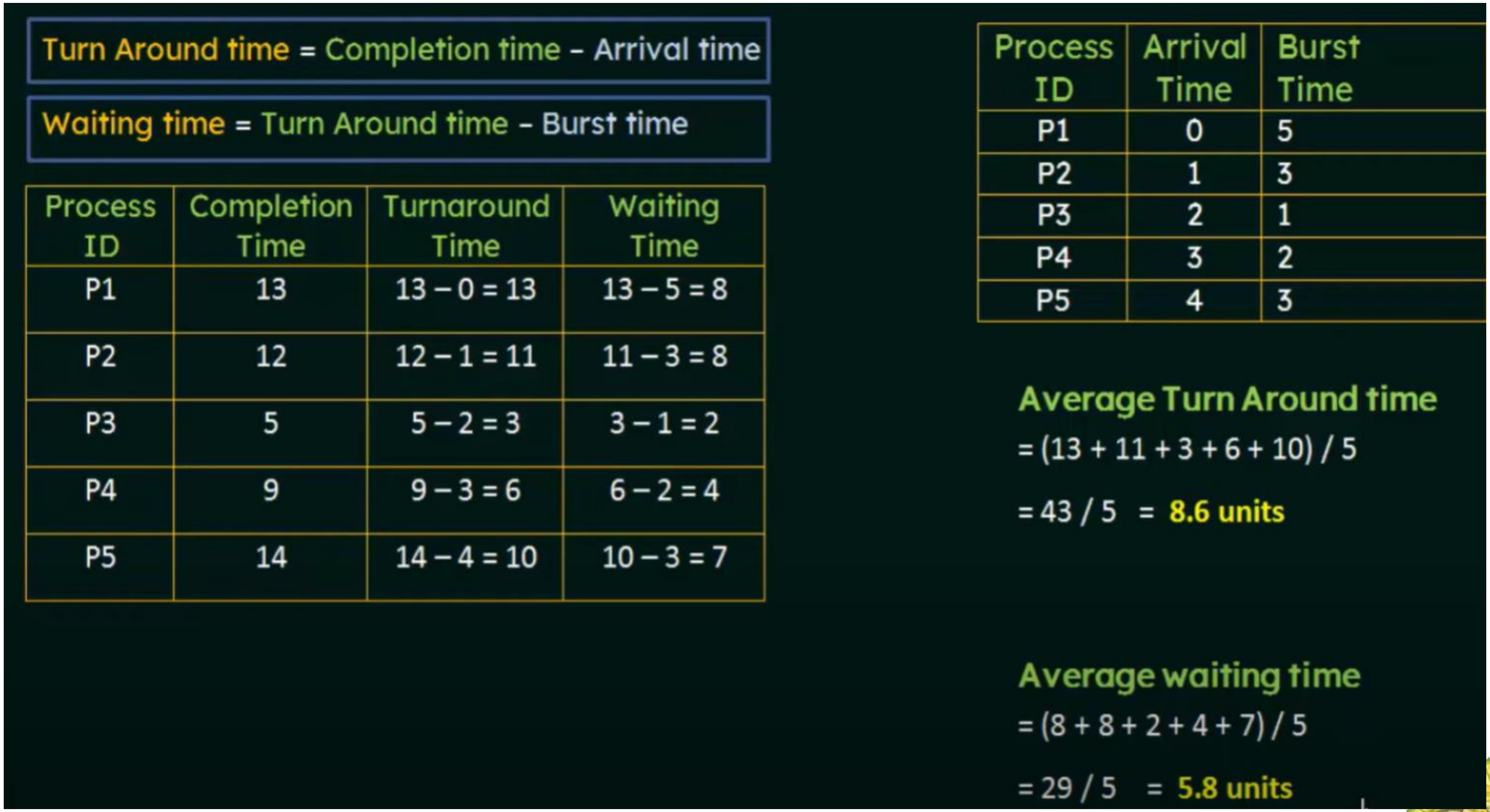
Time Quantum
2 Units

Clock
13

Gantt Chart:

0–2	2–4	4–5	5–7	7–9	9–11	11–12	12–13	13–14
P1	P2	P3	P1	P4	P5	P2	P1	P5

Also calculate average turnaround time and waiting time



Turn Around time = Completion time – Arrival time

Waiting time = Turn Around time – Burst time

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

Average Turn Around time

= (13 + 11 + 3 + 6 + 10) / 5

= 43 / 5 = **8.6 units**

Average waiting time

= (8 + 8 + 2 + 4 + 7) / 5

= 29 / 5 = **5.8 units**

Round robin example for practice

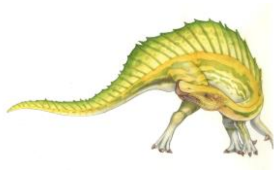
https://www.javatpoint.com/os-round-robin-scheduling-example

Multilevel Queue

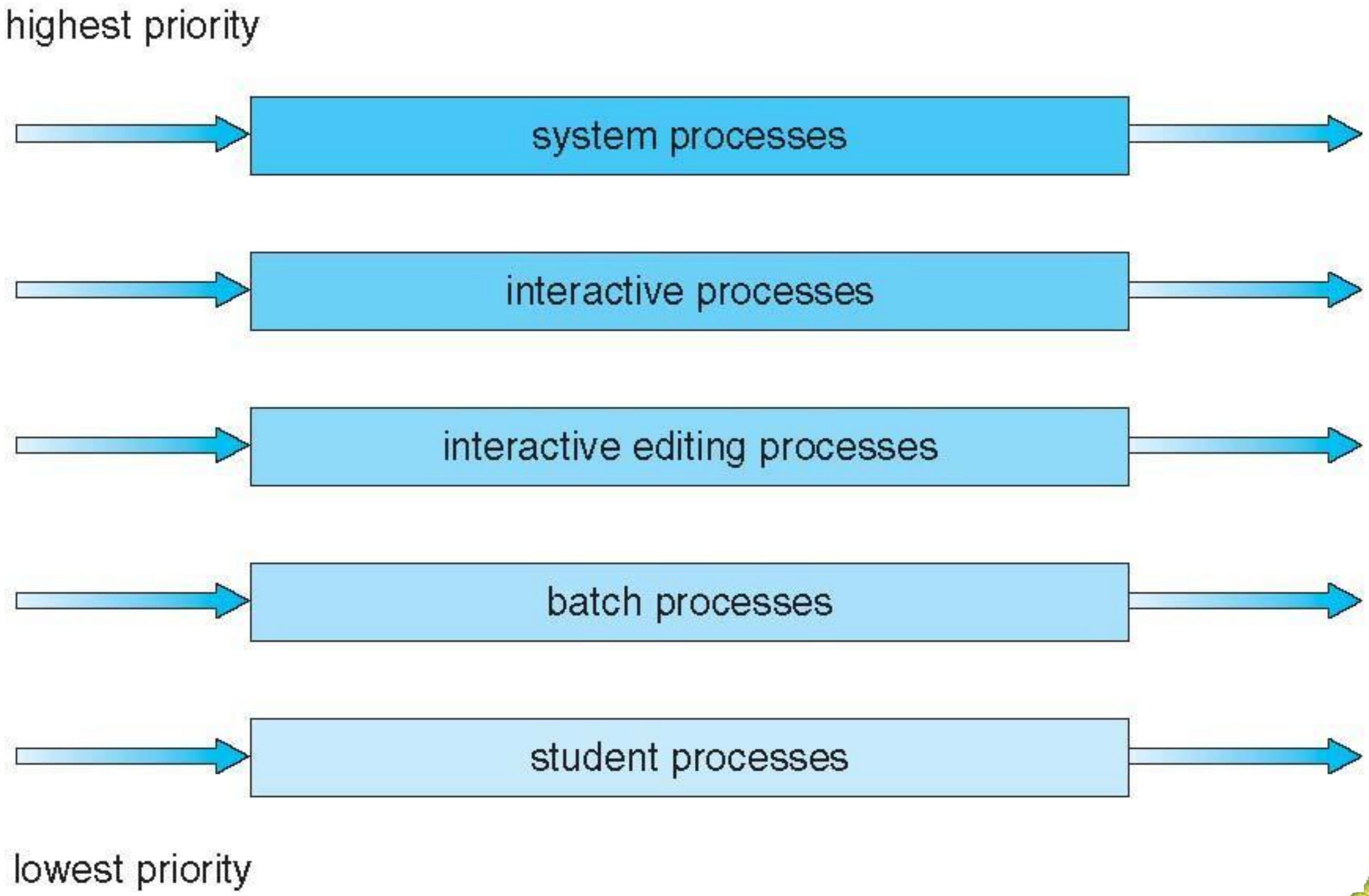
- Ready queue is partitioned into separate queues, eg:
  - foreground (interactive)
  - background (batch)

- Process permanently in a given queue
- Each queue has its own scheduling algorithm:
  - foreground – RR
  - background – FCFS

- Scheduling must be done between the queues:
  - Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
  - 20% to background in FCFS



Multilevel Queue Scheduling



Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

Multilevel Feedback Queue

Characteristics of Multilevel Feedback Queue Scheduling:

- In a **multilevel queue-scheduling** algorithm, processes are permanently assigned to a queue on entry to the system, and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand disadvantage of being inflexible.

Advantages of Multilevel feedback queue scheduling:

- It is more flexible
- It allows different processes to move between different queues

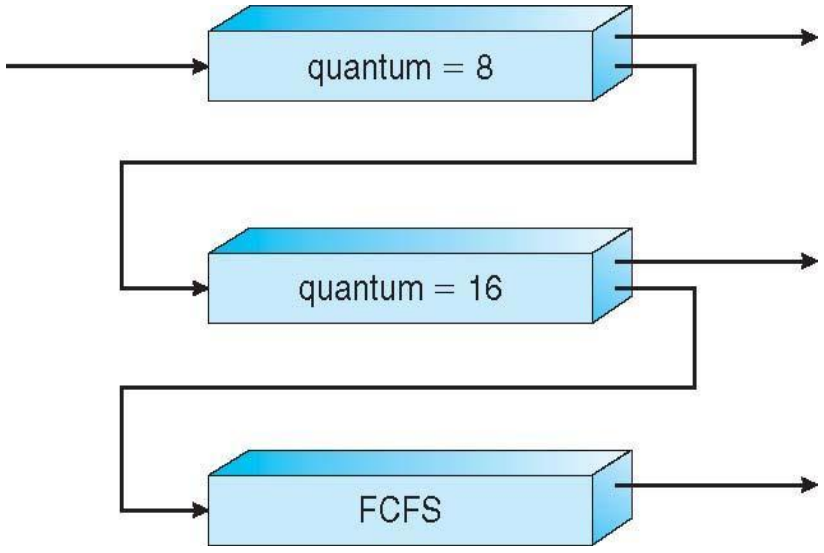
Disadvantages of Multilevel feedback queue scheduling:

- It also produces CPU overheads
- It is the most complex algorithm.

Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS

- Scheduling
  - A new job enters queue Q_0 which is served FCFS
    - When it gains CPU, job receives 8 milliseconds
    - If it does not finish in 8 milliseconds, job is moved to queue Q_1
  - At Q_1 job is again served FCFS and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q_2



Real-Time CPU Scheduling

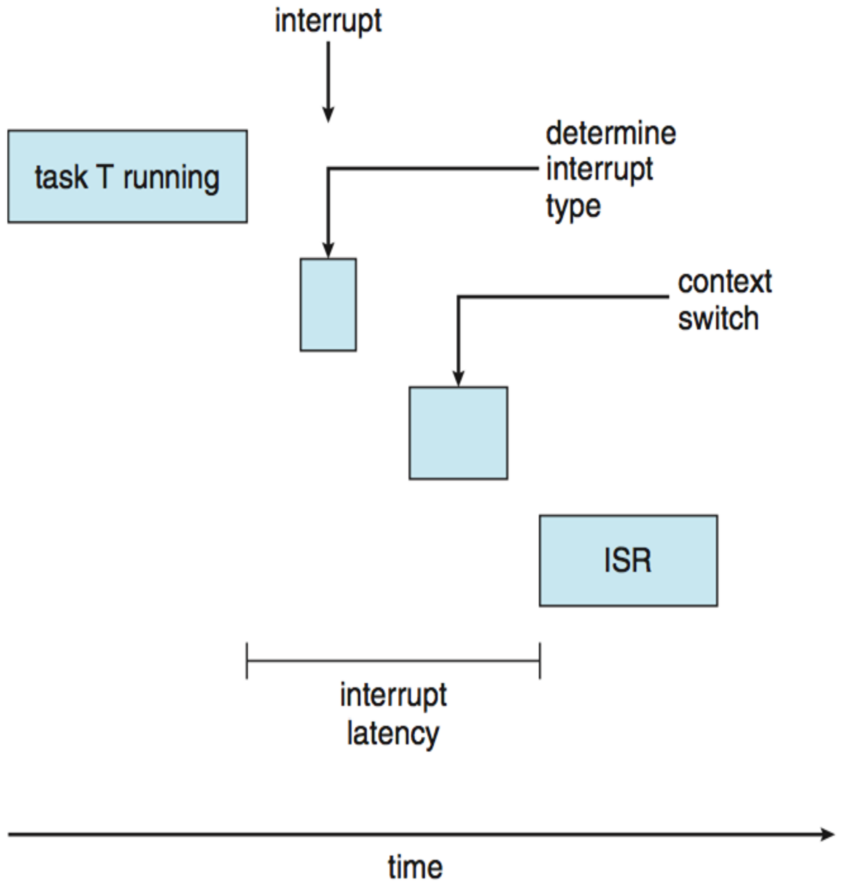
- Can present obvious challenges

- Soft real-time systems – no guarantee as to when critical real-time process will be scheduled

- Hard real-time systems – task

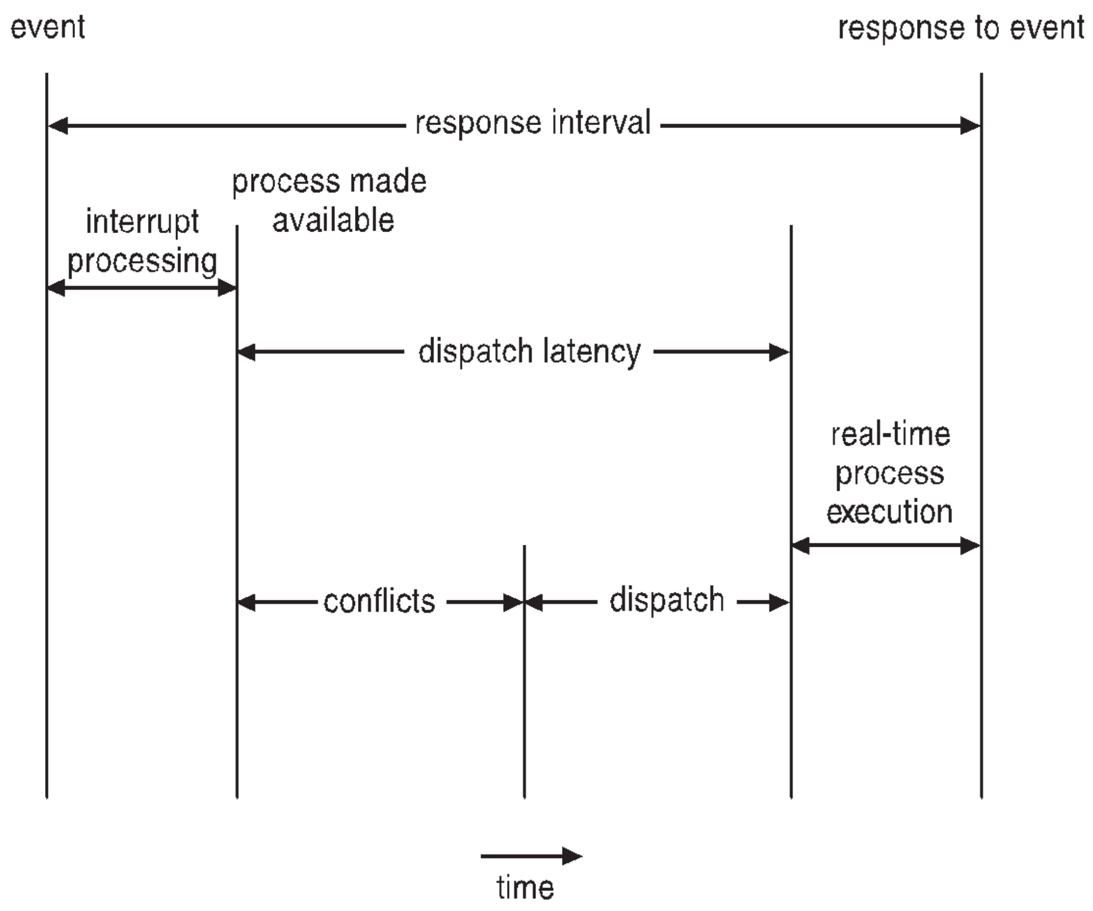
must be serviced by its deadline

- Two types of latencies affect performance
  1. Interrupt latency – time from arrival of interrupt to start of routine that services interrupt
  2. Dispatch latency – time for schedule to take current process off CPU and switch to another



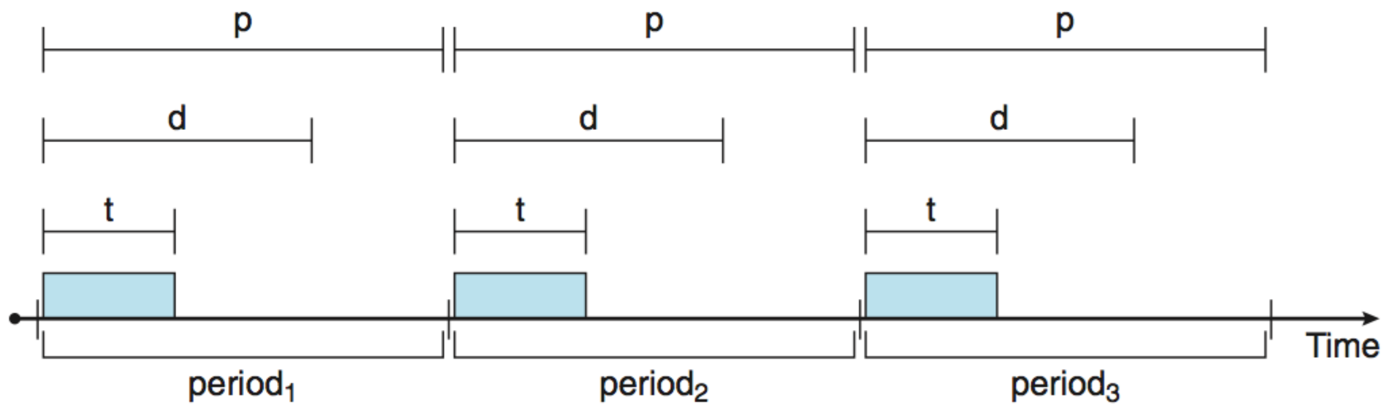
Real-Time CPU Scheduling (Cont.)

- Conflict phase of dispatch latency:
  1. Preemption of any process running in kernel mode
  2. Release by low-priority process of resources needed by high-priority processes



Priority-based Scheduling

- For real-time scheduling, scheduler must support preemptive, priority-based scheduling
  - But only guarantees soft real-time
- For hard real-time must also provide ability to meet deadlines
- Processes have new characteristics: periodic ones require CPU at constant intervals
  - Has processing time *t*, deadline *d,* period *p*
  - $0 \le t \le d \le p$
  - Rate of periodic task is 1/*p*

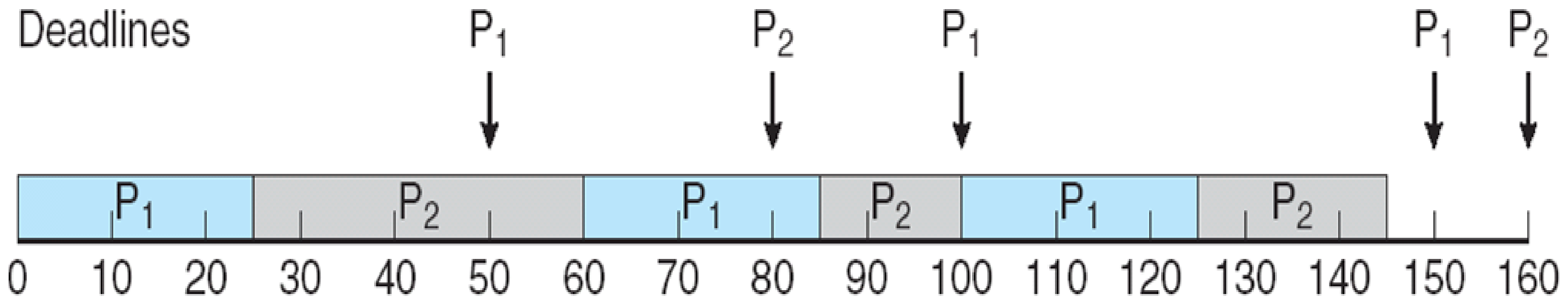


Earliest Deadline First Scheduling (EDF)

- Priorities are assigned according to deadlines:

  * the earlier the deadline, the higher the priority;

  * the later the deadline, the lower the priority



References

For Algorithm math solution:

https://www.javatpoint.com/cpu-scheduling-algorithms-in-operating-systems

End of Lecture 5