CS343: Operating System

Process Scheduling

Lect11 : 21st Aug 2023

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Process Scheduling

Shortest-Job-First (SJF) Scheduling

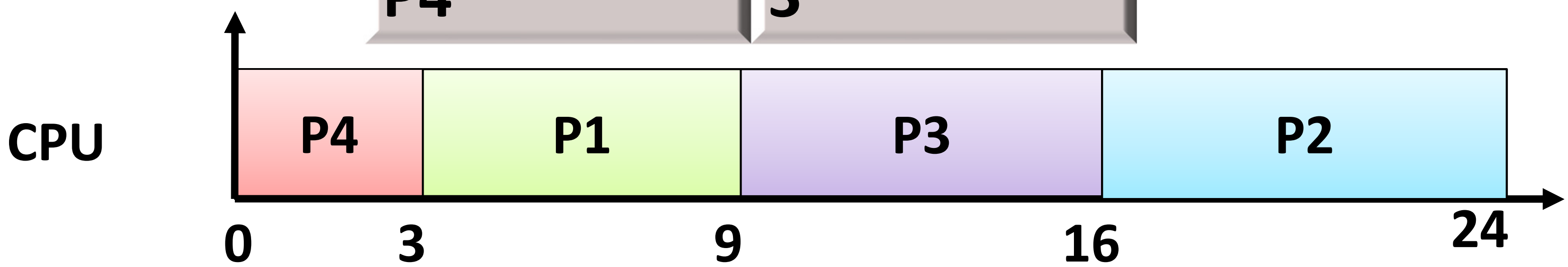
- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time

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
P1	6
P2	8
P3	7
P4	3



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

SJF-Preemptive: Shortest-remaining-time-first

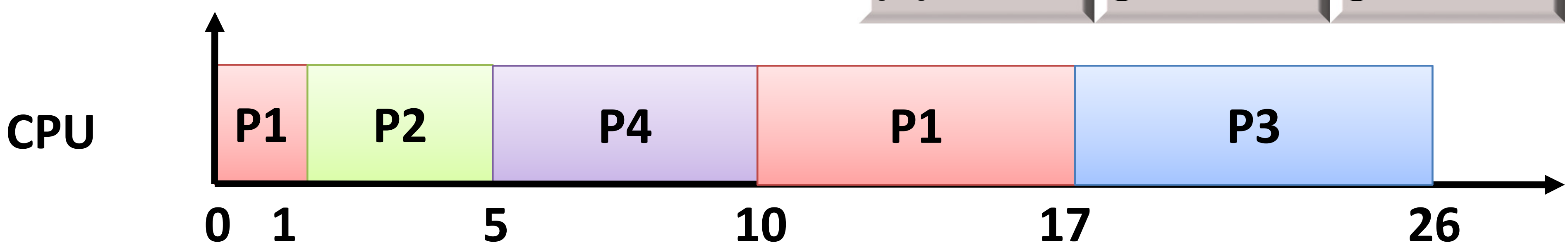
- Now we add the concepts of varying arrival times and preemption to the analysis
- **Dynamic Decision @Runtime**

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

SJF-Preemptive: Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis
- **Dynamic Decision @Runtime**

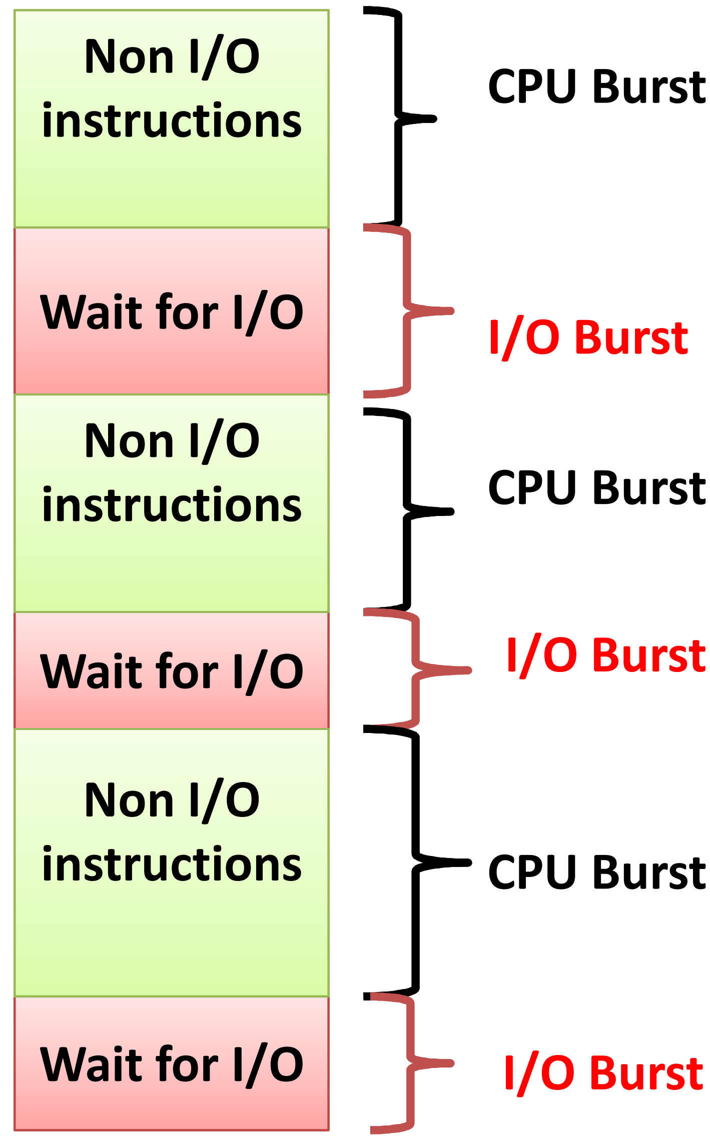
Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



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

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



Prediction: Next CPU Burst

- Can be done by using the length of previous CPU bursts, using exponential averaging
  - t_n is actual CPU Burst of nth CPU Burst
  - τ_{n+1} = predicted values of next cpu burst

$$t_3 = f(t_1, t_2, t_3, \tau_1, \tau_2, \tau_3);$$

$$= \text{may be } f(t_3, \tau_3);$$

Another funny Example: http file transfer

BW depends on chunk size C_n, Initial = 1+x,

$C_n = (1+x)^n$ where $x<1$

EWMA Prediction: Next CPU Burst

- α, $0 \leq \alpha \leq 1$

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

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

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 actual last CPU burst counts
- If we expand the formula, we get:

$$\tau_{n+1} = \alpha\, \mathbf{t}_n + (1 - \alpha)\alpha\, \boldsymbol{t}_{n-1} + \ldots$$
$$+ (1 - \alpha)^j \alpha\, \boldsymbol{t}_{n-j} + \ldots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

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

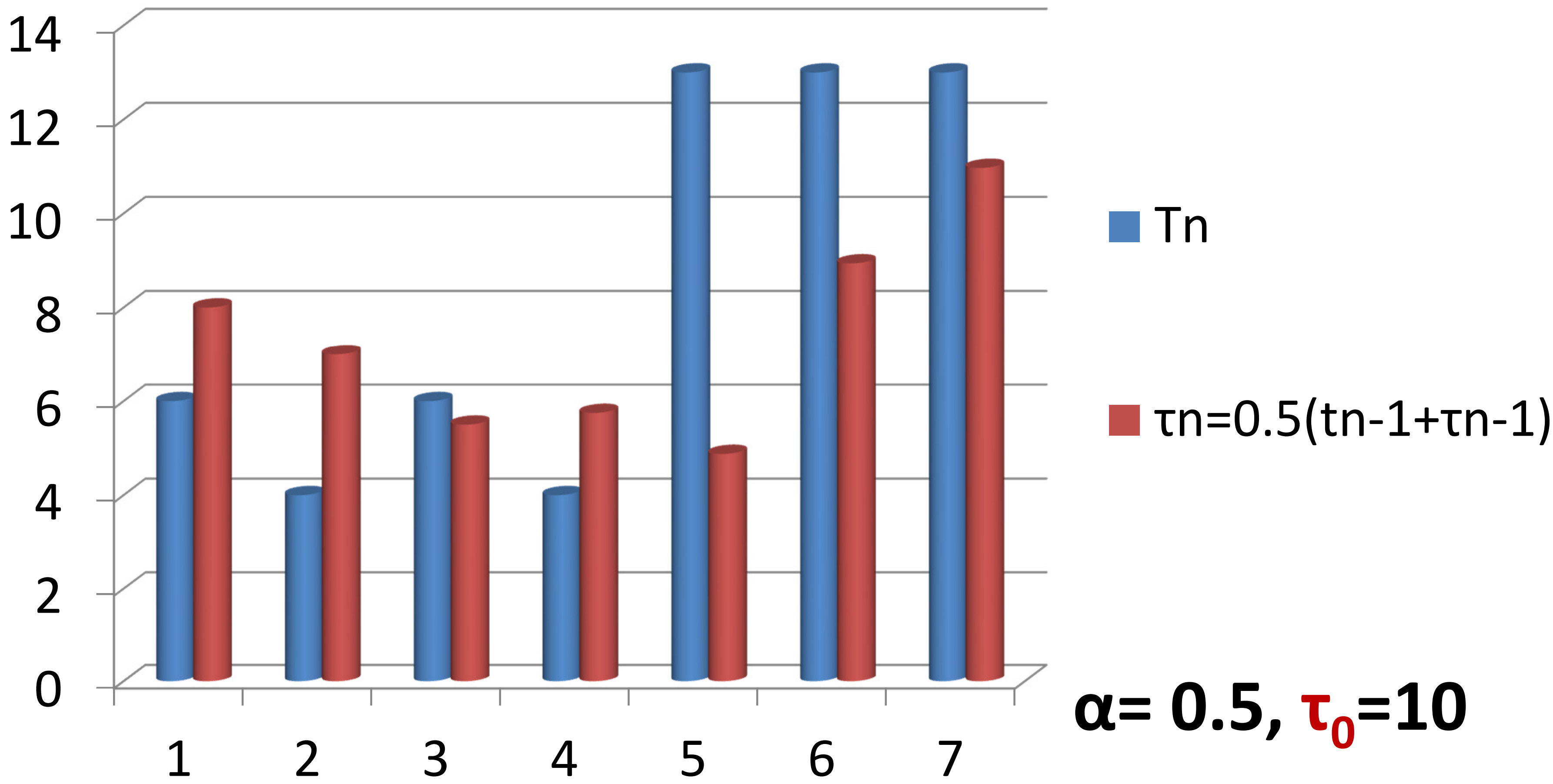
Examples of Exponential Averaging

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

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

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

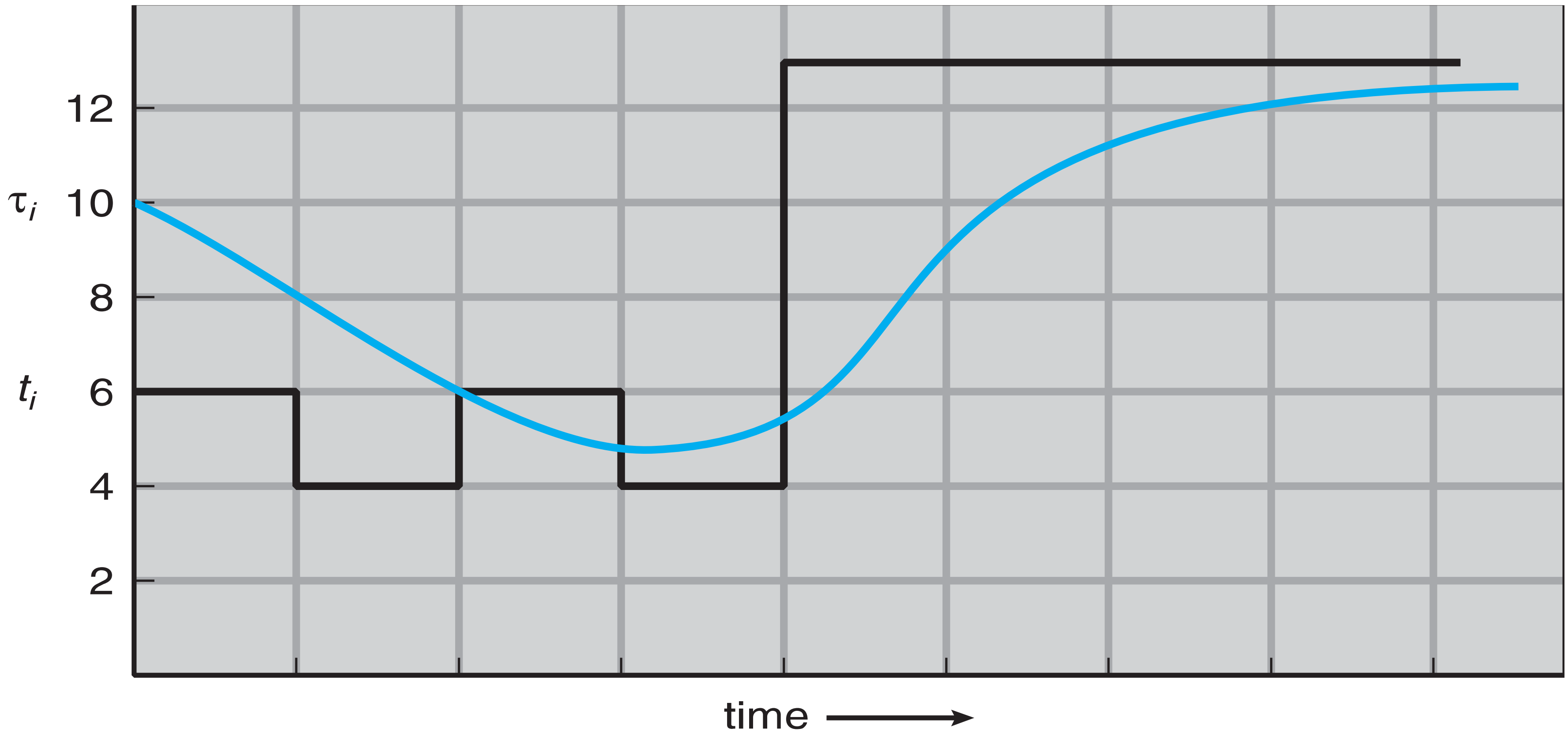
$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n$$



$\alpha = 0.5$, $\tau_0 = 10$

$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n$ with $\alpha = 0.5$

Prediction of the Length of the Next CPU Burst

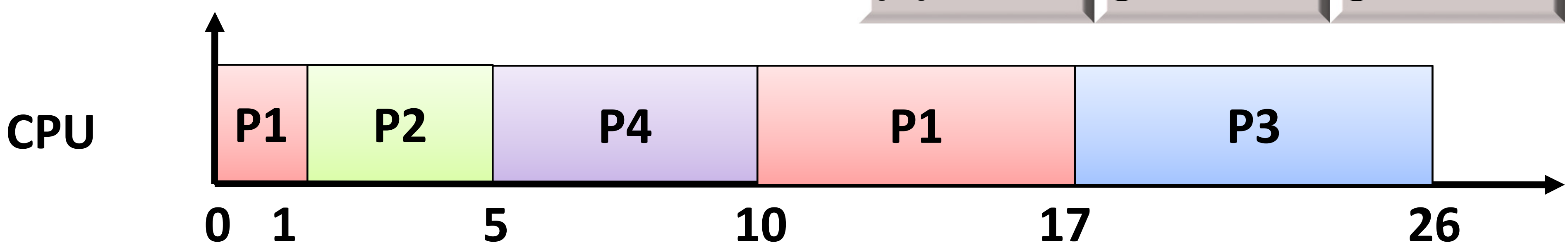


CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

SJF-Preemptive: Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis
- **Dynamic Decision @Runtime**

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



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

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, Non-preemptive
- SJF is priority scheduling
  - Where priority is 1/CPU Burst time

Starvation: Priority Scheduling

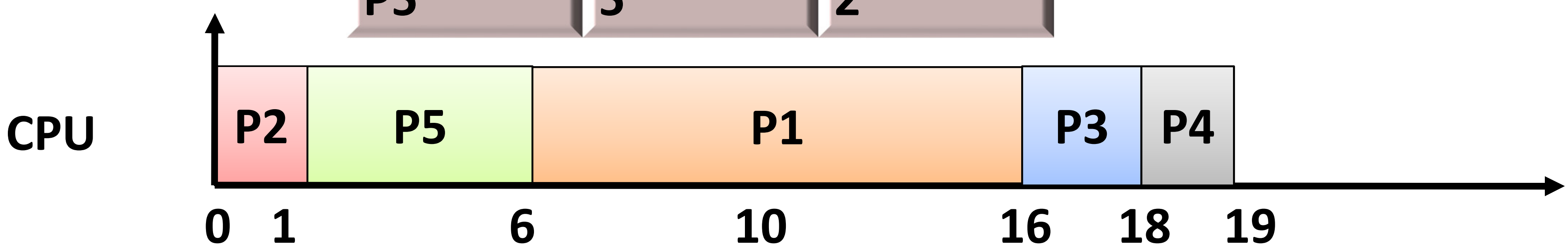
- **Starvation Problem**
  - low priority processes may never execute
  - Starve to execute…
- **Solution**
  - **Aging** – as time progresses increase the priority of the process

Example of Priority Scheduling

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

Example of Priority Scheduling

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

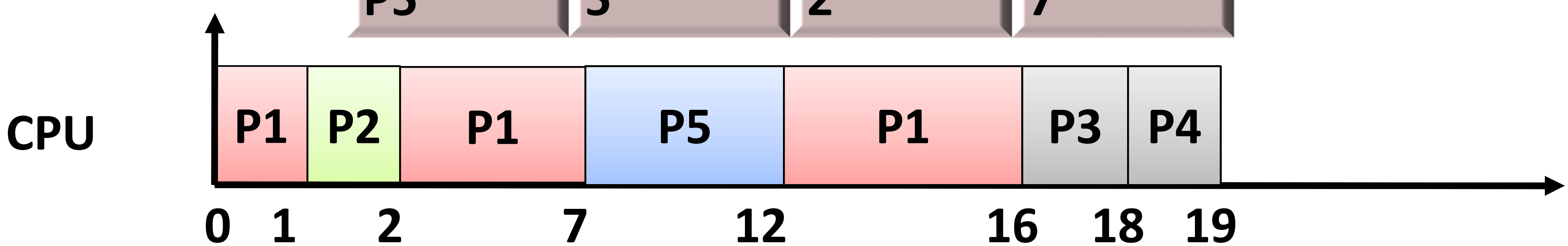


- Assume all arrived@ time 0
- Average waiting time (6+0+16+18+1=41)/

  =8.2 msec

Priority Scheduling Preemptive

Process	Burst Time	Priority	Arrival Time
P1	10	3	0
P2	1	1	1
P3	2	4	6
P4	1	5	6
P5	5	2	7



- **All arrived@ different time**
- Average waiting time (6+0+16+18+7=47)/5

=9.4 msec

Round Robin (RR)

- **Gol Gappe Wala Scheduling**
- 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.

Round Robin (RR)

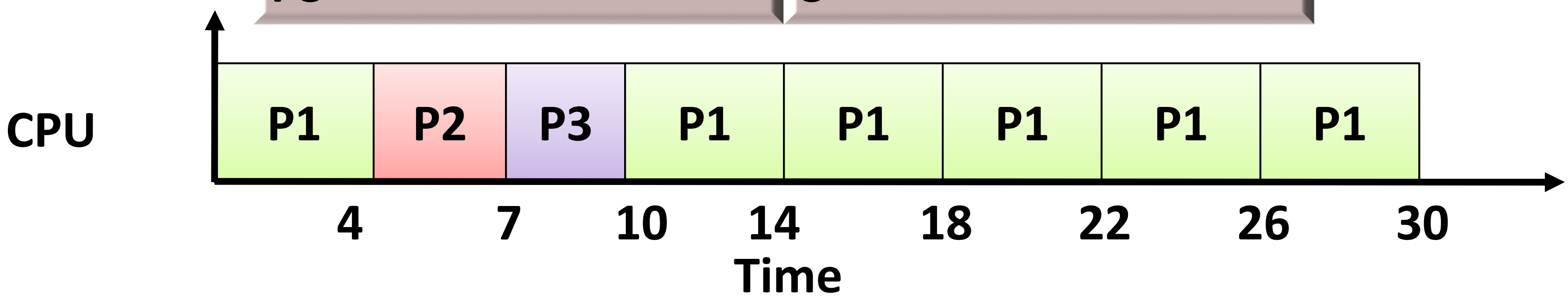
- If there are ***n*** processes in the ready queue and the time quantum is ***q***, then
  - Each process gets **1/*n*** of the CPU time in chunks of at most ***q*** time units at once.
- No process waits more than **(*n*-1)*q*** time units.

Round Robin (RR)

- Timer interrupts every quantum to schedule next process
  - **Timer: Hardware unit, similar to Alarm**
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

Example : RR (with q = 4)

Process	Burst Time
P1	24
P2	3
P3	3



- 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

Process time 10

Quantum	Context Switches
12	0
6	1
1	9

Thanks