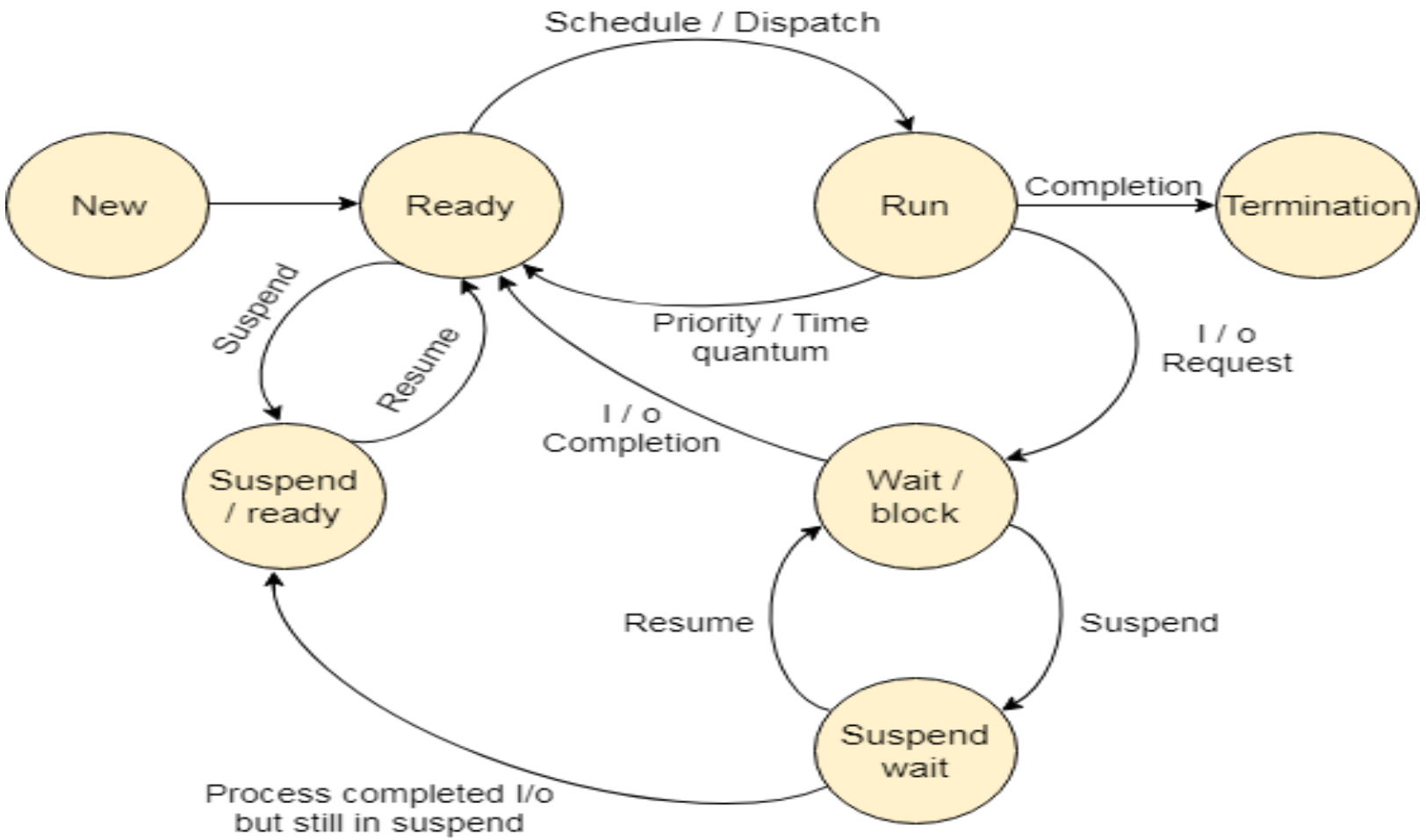
Process Scheduling

Process State Transition Diagram



CPU Scheduler

- A process migrates between the various scheduling queues throughout its lifetime.
- The OS must select, for scheduling purposes, processes from these queues in some fashion.
- The selection process is carried out by the appropriate **schedulers**

Types of Schedulers

- Long Term Scheduler
  - Selects processes that are ready to execute and loads them into ready queue.
- Short Term Scheduler
  - Selects process from the ready queue and allocates CPU to it
- Medium Term Scheduler
  - Selects which swapped out process is to be loaded next into the ready queue

Dispatcher

- The dispatcher is the module that gives control of the CPU to the process selected by the short term scheduler.

- ST → Dispatcher→running

Scheduling Criteria

- **Arrival Time**: When a process enters in a ready state
- **Burst Time/Execution Time:** It is the time required by the process to complete execution. It is also called running time.
- **Completion Time**: When process complete and exit from a system
- **Turn Around Time** : The interval from time of submission of a process to time of completion. TAT=CT-AT

Scheduling Criteria

- **Waiting Time**: Time spent by a process waiting in the ready queue.

  WT = TAT – Burst Time.

- **Response Time** : The time taken from submission of the process of request until the first response is produced.

Scheduling Criteria

- CPU Utilization
  - Keep the CPU as busy as possible.
- Throughput
  - Number of processes that are completed per unit time
- **Maximize** – CPU Utilization & Throughput
- **Minimize** – TAT, WT, RT
- Fair Allocation of CPU to processes.

CPU Scheduling

- CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.
- There are many different CPU scheduling algorithms are there.

CPU Scheduling

- Short Term Scheduler
- Ready to Running
- Calls when Process moves from
  - Running -> ready
  - Running -> Termination
  - Running -> Wait

Pre-emptive & Non Pre-emptive Scheduling

- Under non preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to wait state.
- A scheduling is said to be Preemptive, when a process returns the CPU if its time slice is over.

FCFS (First Come First Serve)

- Mode: Non Pre-emptive / Criteria: Arrival time
- The process requests the CPU first is allocated the CPU first .
- The implementation can be easily managed by maintaining a FIFO list.
  - When a process enters the ready queue, its PCB is linked onto the tail of the queue.
  - The running process is removed from the queue. When the CPU is free, it is allocated to the process at the head of the queue.

FCFS (First Come First Serve)

PNO	AT	BT	CT	TAT	WT
1	0	4			
2	1	3			
3	2	1			
4	3	2			
5	4	5			

TAT = CT – AT
WT= TAT - BT

Average TAT
Average Waiting Time:

If more processes with same arrival time , then it will take the process with smaller PID

FCFS (First Come First Serve)

PID	AT	BT	CT	TAT	WT
1	0	40			
2	1	3			
3	1	1			

Average TAT
Average Waiting Time:

FCFS (First Come First Serve)

- FCFS suffers from **Convoy effect**.
- Shorter processes get stuck waiting behind longer processes
- Inefficient use of system resources.
- Increased waiting times for all processes.
- The average waiting time is much higher than the other algorithms.

FCFS is very simple and easy to implement but efficient.

SHORTEST JOB FIRST (SJF) – NON PREEMPTIVE

- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.
- Objective is to minimize the average waiting time by prioritizing shorter processes over longer ones.

SJF – NON PREEMPTIVE

Given five processes with process id/ no. as P1, P2, P3, P4, P5 along with their respective arrival time (A.T.) and burst time (B.T.). We need to draw the Gantt chart and find the completion time for each process. The time is denoted in milliseconds(ms).

Process id/ no.	Arrival Time(A.T.)	Burst Time(B.T.)
P1	0 ms	2 ms
P2	1 ms	5 ms
P3	2 ms	3 ms
P4	3 ms	1 ms
P5	5 ms	4 ms

SJF – NON PREEMPTIVE

Process id/ no.	Arrival Time(A.T.)	Burst Time(B.T.)
P1	0 ms	2 ms
P2	1 ms	5 ms
P3	2 ms	3 ms
P4	3 ms	1 ms
P5	5 ms	4 ms

Completion time(C.T)	Turn around time(T.A.T) (C.T. - A.T.)	Waiting time(W.T.) (T.A.T. - B.T.)
2ms	2-0= 2ms	2-2=0ms
15ms	15-1= 14ms	14-5= 9ms
5ms	5-2= 3ms	3-3 = 0ms
6ms	6-3= 3ms	3-1= 2ms
10ms	10-5= 5ms	5-4= 1ms

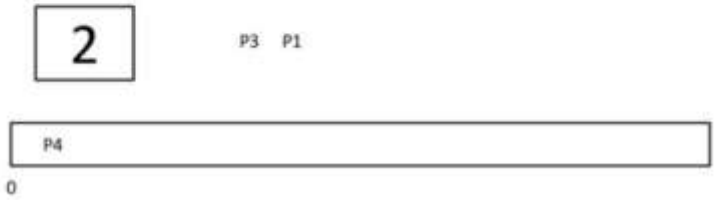
SJF - PREEMPTIVE

Consider the following five process:

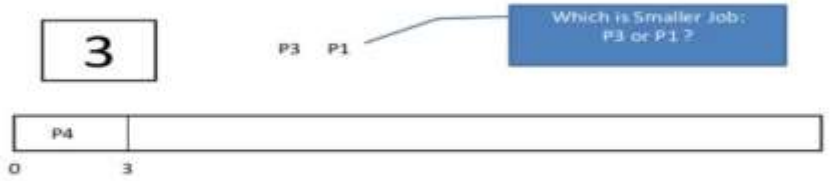
Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Step 1) At time= 1, Process P3 arrives. But, P4 has a shorter burst time. It will continue execution.

Step 2) At time = 2, process P1 arrives with burst time = 6. The burst time is more than that of P4. Hence, P4 will continue execution.

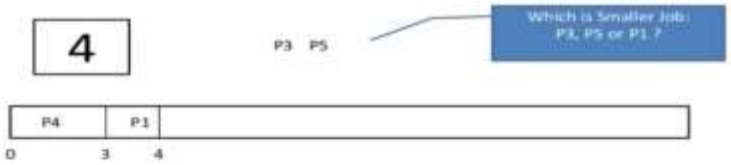


Step 3) At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is lower.



Step 4) At time = 4, process P5 will arrive. The burst time of P3, P5, and P1 is compared. Process P5 is executed because its burst time is lowest. Process P1 is preempted.

Process Queue	Burst time	Arrival time
P1	5 out of 6 is remaining	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4



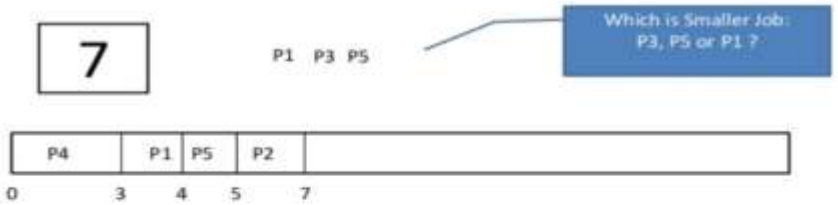
Step 5) At time = 5, process P2 will arrive. The burst time of P1, P2, P3, and P5 is compared. Process P2 is executed because its burst time is least. Process P5 is preempted.

Process Queue	Burst time	Arrival time
P1	5 out of 6 is remaining	2
P2	2	5
P3	8	1
P4	3	0
P5	3 out of 4 is remaining	4

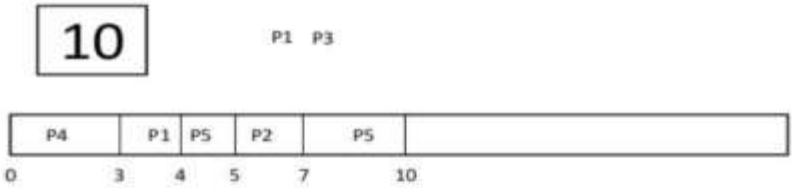
Step 6) At time =6, P2 is executing.

Step 7) At time =7, P2 finishes its execution. The burst time of P1, P3, and P5 is compared. Process P5 is executed because its burst time is lesser.

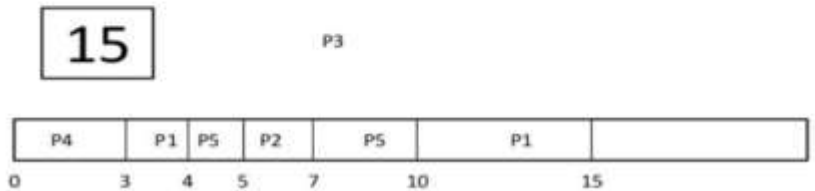
Process Queue	Burst time	Arrival time
P1	5 out of 6 is remaining	2
P2	2	5
P3	8	1
P4	3	0
P5	3 out of 4 is remaining	4

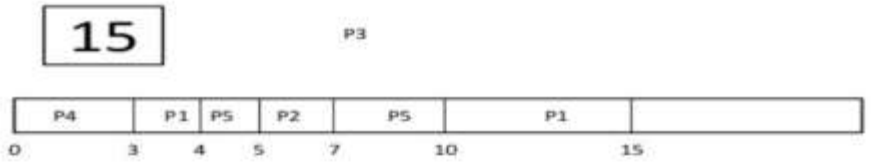


Step 8) At time =10, P5 will finish its execution. The burst time of P1 and P3 is compared. Process P1 is executed because its burst time is less.

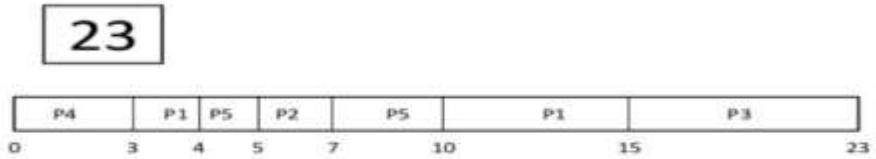


Step 9) At time =15, P1 finishes its execution. P3 is the only process left. It will start execution.





Step 10) At time =23, P3 finishes its execution.



Step 11) Let's calculate the average waiting time for above example.

```
Wait time
P4= 0-0=0
P1=   (3-2) + 6 =7
P2= 5-5 = 0
P5= 4-4+2 =2
P3= 15-1 = 14
```

```
Average Waiting Time =
0+7+0+2+14/5 = 23/5 =4.6
```

SJF

- **Optimal Average Waiting Time**
- **Efficient Resource Utilization**
- **Starvation:** Longer processes may suffer from starvation, where they wait indefinitely if shorter processes keep arriving.
- **Difficult to Implement:** SJF requires precise knowledge of the duration of each process, which is not always possible or practical to obtain.

PRIORITY SCHEDULING

- In Priority scheduling, there is a priority number assigned to each process.
- In some systems, the lower the number, the higher the priority. While, in the others, the higher the number, the higher will be the priority.
- The Process with the higher priority among the available processes is given the CPU.

PREEMPTIVE PRIORITY SCHEDULING

- The CPU is preempted if a new process arrives with a higher priority than the currently running process.
- Example: If Process A is running and Process B arrives with a higher priority, Process A will be stopped, and Process B will start execution.

NON-PREEMPTIVE PRIORITY SCHEDULING

- The running process is not interrupted until it finishes its CPU burst.
- Example: If Process A is running and Process B arrives with a higher priority, Process B will have to wait until Process A completes.

PRIORITY SCHEDULING – NON PREEMPTIVE

Process	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4

Gantt Chart –

PRIORITY SCHEDULING – PREEMPTIVE

Process	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4

PRIORITY SCHEDULING

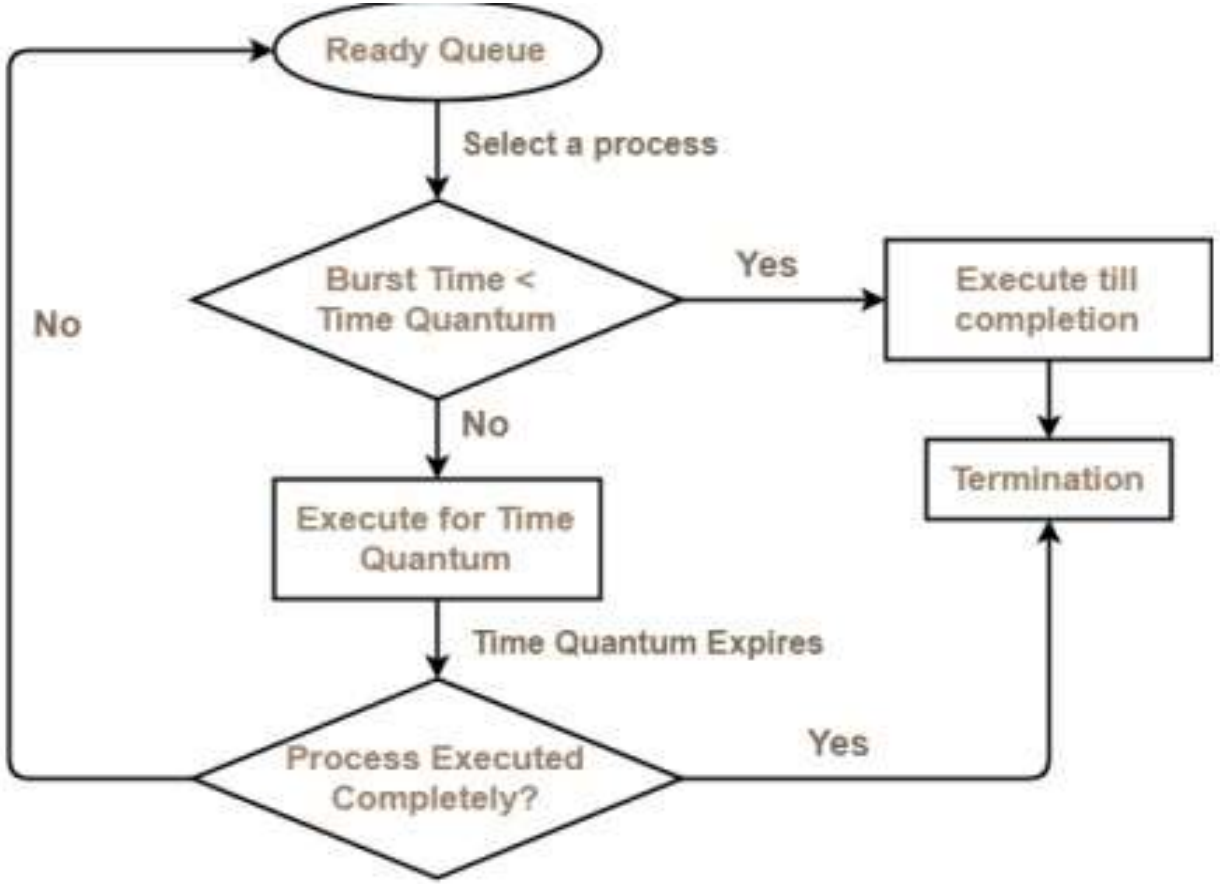
- ✓ **Starvation**: Lower-priority processes may never get executed if high-priority processes keep coming.

- ✓ **Complexity**: Managing different priorities can be complex.

ROUND ROBIN SCHEDULING

- Round Robin (RR) scheduling is a CPU scheduling algorithm that assigns a fixed time quantum (time slice) to each process in the queue.
- The CPU scheduler cycles through the processes, giving each one a turn to execute for a set amount of time before moving on to the next process in the queue.

ROUND ROBIN SCHEDULING



Round Robin Scheduling

ROUND ROBIN SCHEDULING

- Time Quantum: A small, fixed unit of time is defined, usually between 10-100 milliseconds.
- Process Execution: Each process in the queue gets to execute for a time equal to the time quantum.
- Context Switching: When a process's time quantum expires, a context switch occurs, and the CPU is given to the next process in the queue.
- Rotation: The process is then placed at the back of the queue if it hasn't finished execution. This cycle continues until all processes are complete.

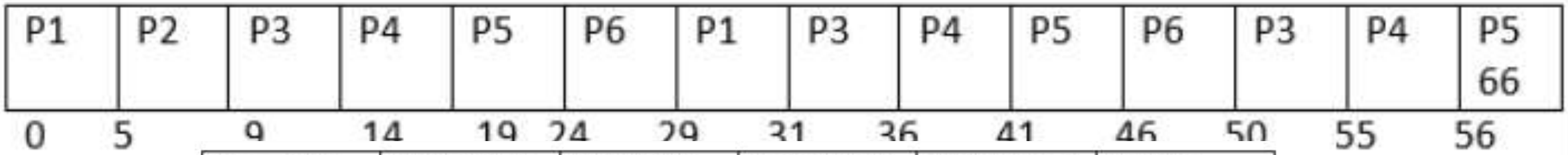
ROUND ROBIN SCHEDULING

Process ID	Arrival Time	Burst Time
P1	0	7
P2	1	4
P3	2	15
P4	3	11
P5	4	20
P6	4	9

Time slice = 5 ms

ROUND ROBIN SCHEDULING

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



Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	7	31	31	24
P2	1	4	9	8	4
P3	2	15	55	53	38
P4	3	11	56	53	42
P5	4	20	66	62	42
P6	4	9	50	46	37

Average Completion Time

Average Completion Time = (31 +9 + 55 +56 +66 + 50) / 6

Average Completion Time = 267 / 6

Average Completion Time = 44.5