# Course Name : Embedded System  EC5464

RA  PRANJALE
LECTURER IN ELECTRONICS
GOVERNMENT POLYTECHNIC AMRAVATI

# EMBEDDED SYSTEMS EC5464

## CO6 :Interpret the features of REAL TIME OPERATING SYSTEM

Topic 6.1 : Operating System: General and Real time operating system.

Topic 6.2: Characteristics of Real Time Operating System: Consistency, Reliability, scalability, Performance, Predictability

Topic 6.3 Functions of RTOS: Task management, Scheduling, Resource allocation and interrupt handling .

Topic 6.4 Task synchronization and Mutual Exclusion, Multitasking

Topic 6.5 : Features of  RTOS: Watchdog timer, Semaphore, Deadlock.  Starvation Deadlock , Multiple process

# Scheduling : ROUND ROBIN ALGORITHM

- The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.

- In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes.

- Round robin is a pre-emptive algorithm
- The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
- The process that is preempted is added to the end of the queue.
- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task that needs to be processed. However, it may differ OS to OS.
- It is a real time algorithm which responds to the event within a specific time limit.
- Round robin is one of the oldest, fairest, and easiest algorithm.
- Widely used scheduling method in traditional OS.

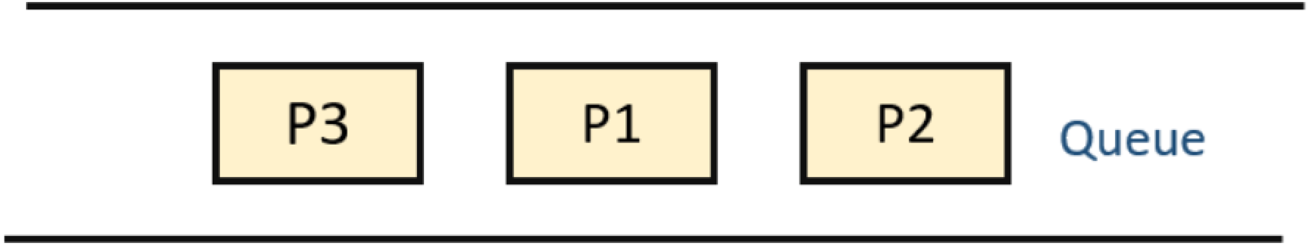
**Advantage of Round-robin Scheduling**

- It doesn't face the issues of starvation.
- All the jobs get a fair allocation of CPU.
- It deals with all process without any priority
- If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
- This scheduling method does not depend upon burst time. That's why it is easily implementable on the system.
- Once a process is executed for a specific set of the period, the process is preempted, and another process executes for that given time period.
- Allows OS to use the Context switching method to save states of preempted processes.
- It gives the best performance in terms of average response time.

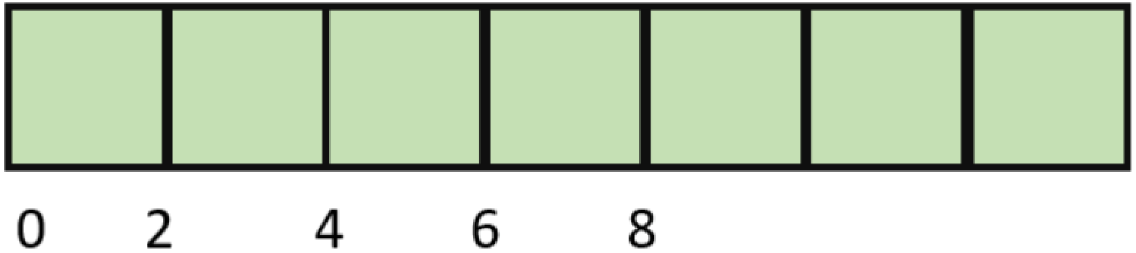
- **Disadvantages of Round-robin Scheduling**
- If slicing time of OS is low, the processor output will be reduced.
- This method spends more time on context switching
- Its performance heavily depends on time quantum.
- Priorities cannot be set for the processes.
- Round-robin scheduling doesn't give special priority to more important tasks.
- Decreases comprehension
- Lower time quantum results in higher the context switching overhead in the system.
- Finding a correct time quantum is a quite difficult task in this system.

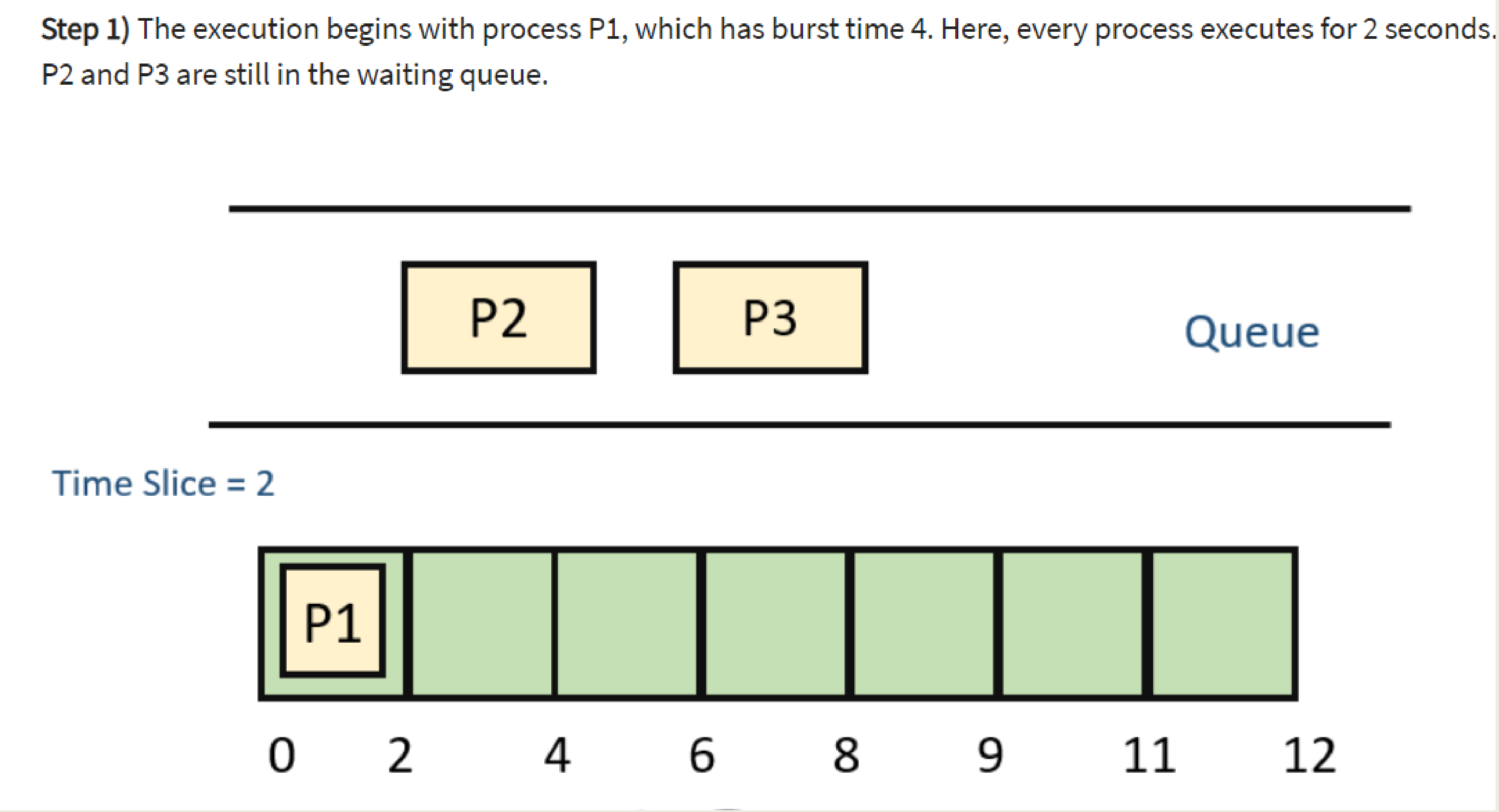
- The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns.
- Round robin is one of the oldest, fairest, and easiest algorithms and widely used scheduling methods in traditional OS.
- Round robin is a pre-emptive algorithm
- The biggest advantage of the round-robin scheduling method is that If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
- This method spends more time on context switching
- Worst-case latency is a term used for the maximum time taken for the execution of all the tasks.

| Process Queue | Burst time |
|---|---|
| P1 | 4 |
| P2 | 3 |
| P3 | 5 |

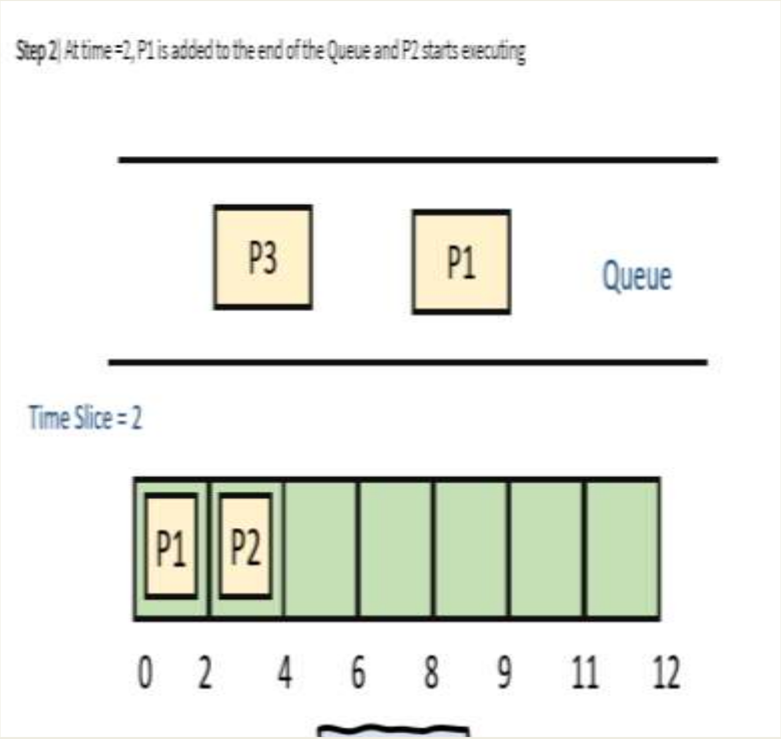


Time Slice = 2

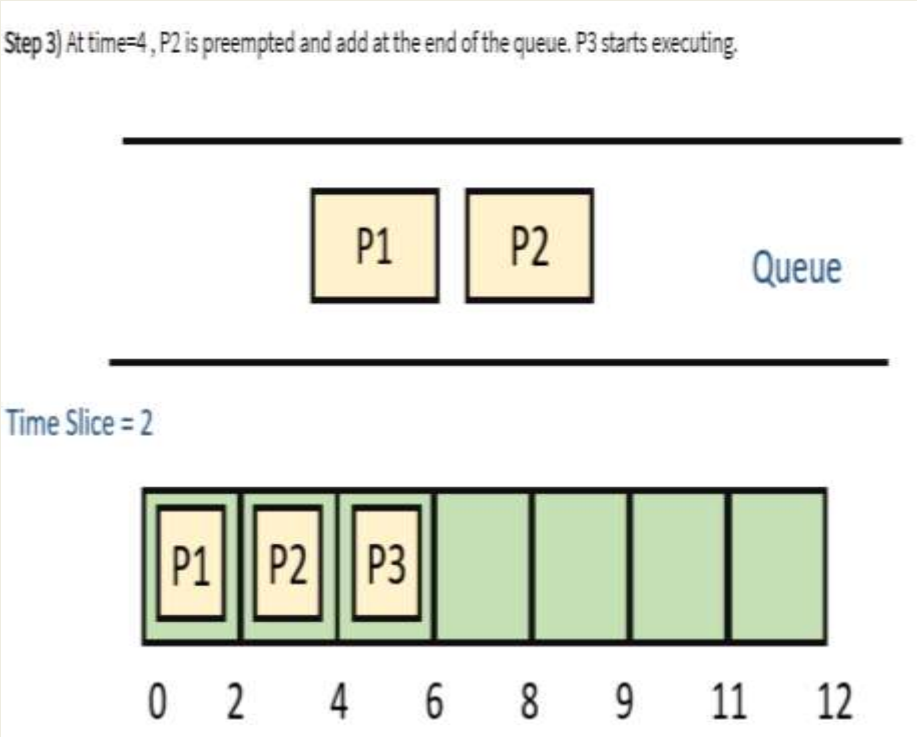




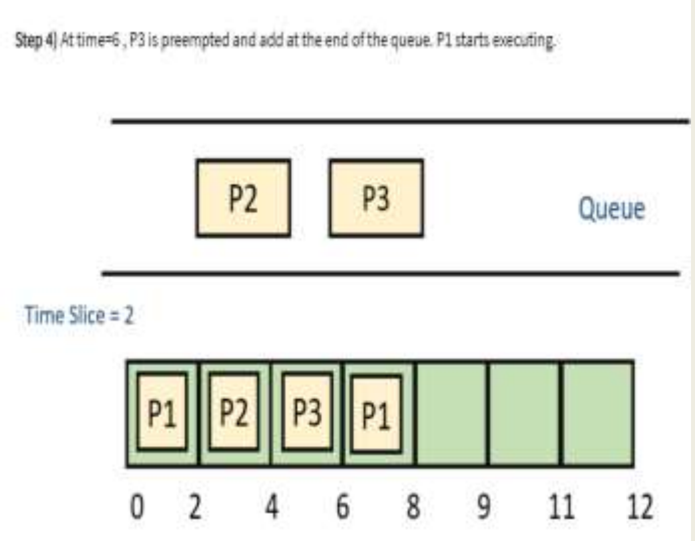
**Step 1)** The execution begins with process P1, which has burst time 4. Here, every process executes for 2 seconds. P2 and P3 are still in the waiting queue.



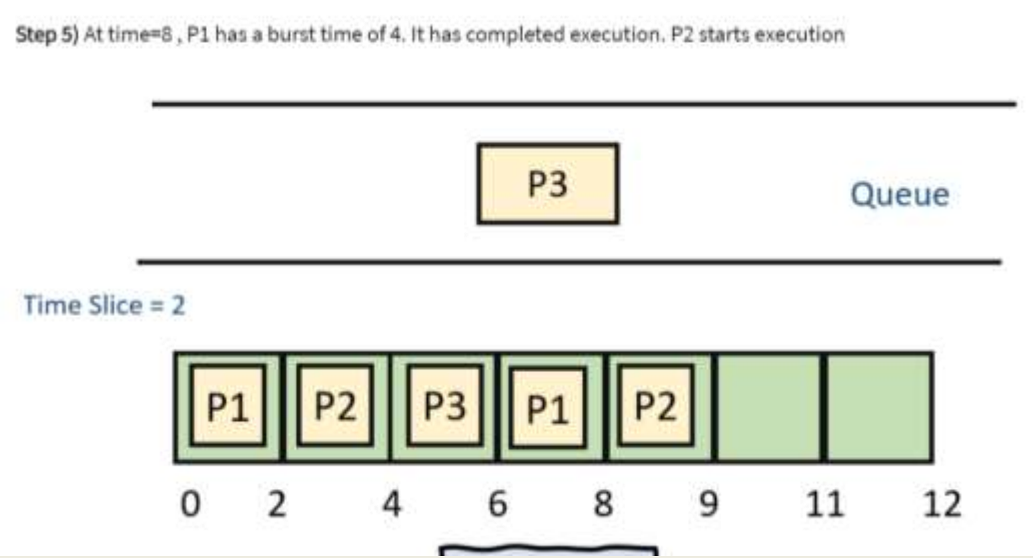
Step 2) At time =2, P1 is added to the end of the Queue and P2 starts executing



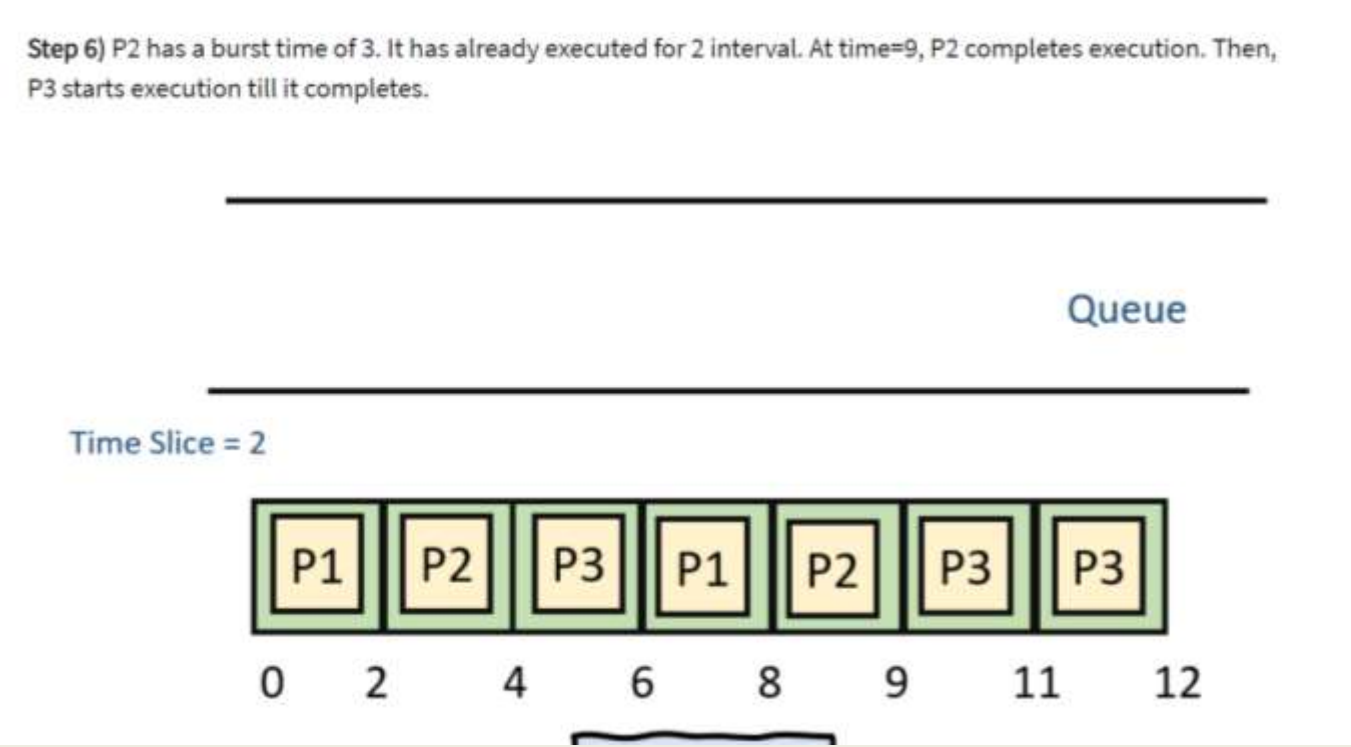
Step 3) At time=4 , P2 is preempted and add at the end of the queue. P3 starts executing.



Step 4) At time=6 , P3 is preempted and add at the end of the queue. P1 starts executing.



Step 5) At time=8 , P1 has a burst time of 4. It has completed execution. P2 starts execution



**Step 6)** P2 has a burst time of 3. It has already executed for 2 interval. At time=9, P2 completes execution. Then, P3 starts execution till it completes.

Queue

Time Slice = 2

| | | | | | | |
|---|---|---|---|---|---|---|
| P1 | P2 | P3 | P1 | P2 | P3 | P3 |

0 2 4 6 8 9 11 12