Unit-II
Lecture: 11
(Scheduling)
(Part-III)

Priority Scheduling:

In this type of algorithms, a priority is associated with each process and the processor is given to the process with the highest priority.

Equal priority processes are scheduled with FCFS method.

- Priority scheduling is a more general case of SJF.
    - SJF uses the inverse of the next expected burst time as its priority. The smaller the expected burst, the higher the priority.
- Priorities are generally some fixed range of numbers such as 0 to 7. (Here we assume that low numbers represent high priority).
- Priorities can be assigned either internally or externally.
    - Internal priorities are assigned by the OS using criteria such as average burst time, ratio of CPU to I/O activity, system resource use and other factors available to the kernel.
    - External priorities are assigned by users like based on the importance of the job etc.
- Priority scheduling can be either preemptive or non-preemptive.

e.g. Consider the following:

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

The average waiting time is: 8.2 ms.

Priority scheduling can **suffer from a major problem known as indefinite blocking or starvation**, in which a low-priority task can wait forever because there are always some other jobs around that have higher priority.

- If this problem is allowed to occur, then processes will either run eventually when the system load lightens or will eventually get lost when the system is shut down or crashes.

- One **common solution to this problem is aging**,
  - in which priorities of job increase the longer they wait.
  - Under this scheme, a low-priority job will eventually get its priority raised high enough that it gets run.

Round Robin Scheduling:

Round robin scheduling is similar to FCFS scheduling except that CPU bursts are assigned with limits called time quantum.

The round robbing (RR) scheduling is designed especially for time-sharing systems.

When a process is given to the CPU, a timer is set for whatever value has been set for a time quantum.

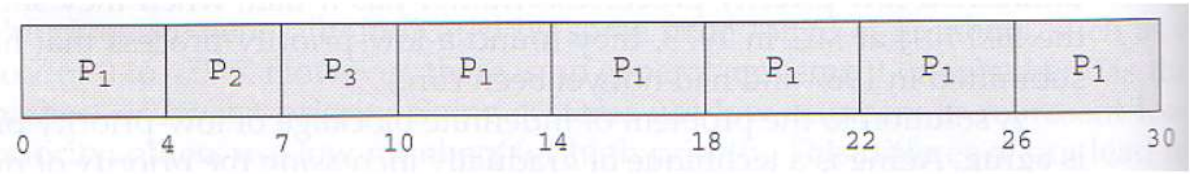
- If the process finishes its burst before the time quantum timer expires, then it is swapped out of the CPU just like the normal FCFS algorithm.
- If the timer goes off first, then the process is swapped out of the CPU and moved to the back end of the ready queue.

The ready queue is maintained as a circular queue, so when all processes have had a turn, then the scheduler gives the first process another run and so on.

A time quantum is generally from 10 to 100ms.

Consider the following set of processes that arrive at time 0. If the time quantum is 4ms, then the average waiting time is 5.66ms.

Process	Burst Time
P1	24
P2	3
P3	3



The performance of RR depends heavily on the size of the time quantum selected.

- At one extreme, if the quantum is very large, RR is same as the FCFS policy.
- If the time quantum is very small, it would appear each process among n processes has its own processor running at 1/n the speed of the real processor. Further, a smaller time quantum will increase context switches.

The average turnaround time of a set of processes does not necessarily improve as the time-quantum size increases.

In general, the average turnaround time can be improved if the most processes finish their next CPU burst in a single time quantum.

Example (smaller timeslice)

Time slice = 1

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	3	2
P4	9	1

Average Waiting time
= (7 + 6 + 3 + 1) / 4
= 4.25

Average Response Time
= (0 + 0 + 1 + 1) / 4
= 1/2

#Context Switches = 11

Arrival	P1		P2	P3						P4				
schedule	P1		P2	P1	P3	P2	P1	P3	P2	P1	P4	P2	P1	P1
FIFO			P1	P3 P2	P2 P1	P1 P3	P3 P2	P2 P1	P1	P4 P2	P2 P1	P1		

More context switches but quicker response time

Example (larger timeslice)

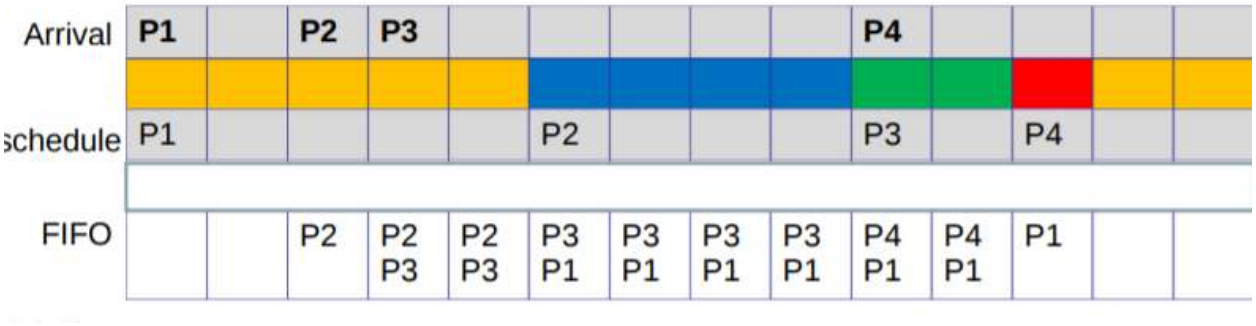
Time slice = 5

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	3	2
P4	9	1

Average Waiting time
= (7 + 3 + 6 + 2) / 4
= 4.25

Average Response Time
= (0 + 3 + 6 + 2) / 4
= 2.75

#Context Switches = 4



Lesser context switches but looks more like FCFS (bad response time)