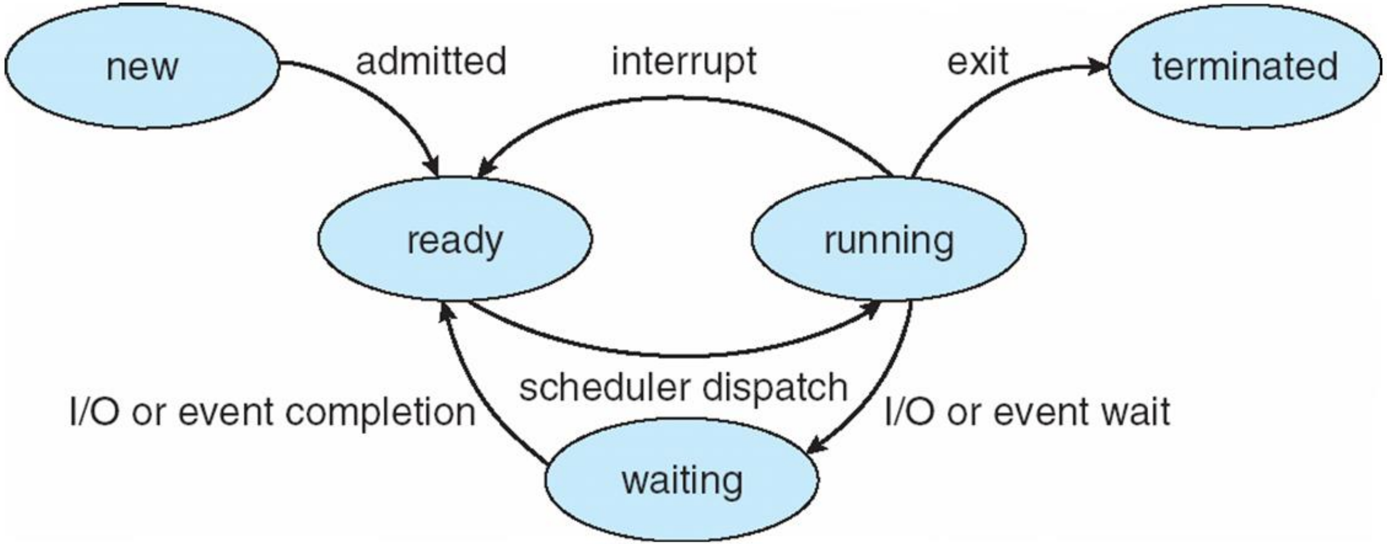
CS343 Operating Systems

Lecture 4

CPU Scheduling Algorithms

John Jose
Associate Professor
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Process State Diagram



Scheduling Criteria

- CPU Utilization – Percentage time CPU is busy executing process.⇧
- Throughput - Number of processes that are completed per time unit.⇧
- Turnaround time - The interval from the time of submission of a process to the time of completion.⇩
- Waiting Time - Amount of time that a process spends waiting in the ready queue. ⇩
- Response Time - Time from the submission of a request until the first response is produced.⇩

CPU Scheduling Algorithms

Batch Systems

- First-come first-served
- Shortest job first
- Shortest remaining Time next

Interactive Systems

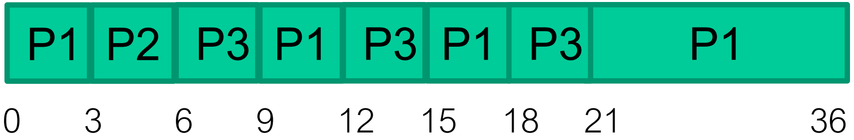
- Round-robin scheduling
- Priority scheduling
- Multiple queues
- Shortest process next
- Guaranteed scheduling
- Lottery scheduling
- Fair-share scheduling

Round Robin Scheduling

- Modified version of preemptive FCFS
- Each process gets a small unit of CPU time (time quantum)
- FIFO queue is used as input
- When a process enters/re-enters the ready queue, its PCB is linked the tail of the queue
- After quantum expires, the process is preempted and added to the tail of the ready queue (Hence, preemptive scheduling algorithm)
- CPU is allocated to the process at the head of the queue
- Longer process may have multiple context switch before completion

Round Robin Scheduling

- Consider the following process arriving at T0, time quantum of 3 units
  - P1 burst time: 24
  - P2 burst time: 3
  - P3 burst time: 9



- Waiting Time
  - P1: (6+3+3) =12, P2: 0, P3: (6+3+3)= 12
- Completion Time:
  - P1: 36, P2: 3, P3: 21
- Average Waiting Time: (0+12+12)/3 = 8
- Average Completion Time: (3+21+36)/3 = 20

Round Robin Scheduling

- RR scheduling is better for short jobs and fair
- Shorter response time, good for interactive jobs
- Context-switching time adds up for long jobs
- Context switching takes additional time and overhead
- If the chosen quantum is
  - too large, response time suffers
  - infinite, performance is the same as FIFO
  - too small, throughput suffers and percentage overhead grows

Priority Scheduling

- Each process has a priority number
- Highest priority process is scheduled first; if equal priorities, then FCFS
- Managed with a priority queue with priority value as input
- When a process enters the ready queue, its PCB is linked onto the priority queue at the appropriate entry
- CPU is allocated to the process at the head of the queue
- It can have 2 variants; non-preemptive and preemptive
- Arrival of a new process with a higher priority can preempt the currently running process.

Issues with Priority Scheduling

- Consider a scenario in which there are three processes, a high priority (H), a medium priority (M), and a low priority (L).
- Process L is running and successfully acquires a resource file.
- Process H begins; since we are using a preemptive priority scheduler, process L is preempted for process H.
- Process H tries to acquire L's resource, and blocks (held by L).
- Process M begins running, and, since it has a higher priority than L, it is the highest priority ready process. It preempts L and runs, thus starving high priority process H.
- This is known as priority inversion. What can we do?

Priority Inversion

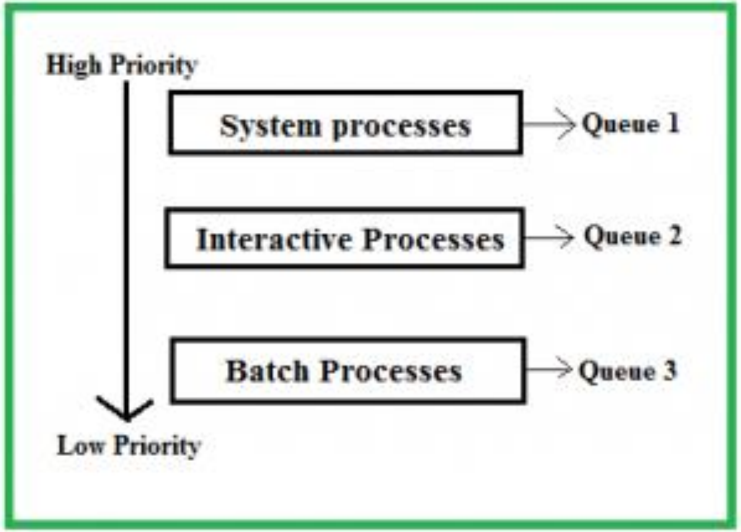
- Process L should, in fact, be temporarily of higher priority than process M, on behalf of process H.
- Process H can donate its priority to process L, which, in this case, would make it higher priority than process M.
- This enables process L to preempt process M and run.
- When process L is finished, process H becomes unblocked.
- Process H, now being the highest priority ready process, runs, and process M must wait until it is finished.

Multilevel Queue

- Ready queue is partitioned into separate queues:
  - Foreground, interactive process→ RR scheduling
  - Background, batch process→ FCFS scheduling
- A process is permanently assigned to one queue
- Each queue has its own scheduling algorithm
- Can be preemptive

Multilevel Feedback Queue Scheduling

- Scheduling must be done between the queues.
  - Fixed priority scheduling
  - 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 among its processes
    - i.e.: 80% Vs 20%

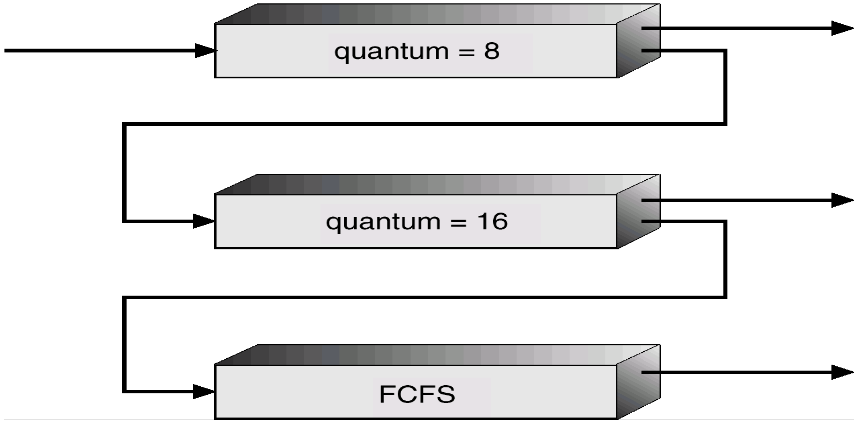


Multilevel Feedback Queue Scheduling

- **A process can move between the various queues. (Aging)**
- 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

Example of Multilevel Feedback Queue

- Three queues:
  - Q0 – time quantum 8 milliseconds, FCFS
  - Q1 – time quantum 16 milliseconds, FCFS
  - Q2 – FCFS



- A new job enters queue Q0 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 Q1.
- At Q1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q2.

Lottery Scheduling

- Each job some number of lottery tickets are issued
- On each time slice, randomly pick a winning ticket
- On average, CPU time is proportional to number of tickets given to each job over time
- How to assign tickets?
  - To approximate SRTF, short-running jobs get more, long running jobs get fewer
  - To avoid starvation, every job gets at least one ticket (everyone makes progress)

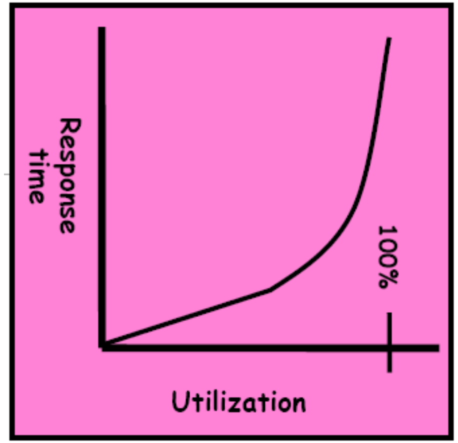
Example: Lottery Scheduling

- Assume short jobs get 10 tickets, long jobs get 1 ticket

# short jobs / # long jobs	% of CPU each short job gets	% of CPU each long job gets
1/1	91%	9%
0/2	N/A	50%
2/0	50%	N/A
10/1	9.9%	0.99%
1/10	50%	5%

Conclusion

- Scheduling: selecting a waiting process from the ready queue and allocating the CPU to it
- When do the details of the scheduling policy and fairness really matter?
  - When there aren't enough resources to go around



Conclusion

- FCFS scheduling, FIFO Run Until Done:
  - Simple, but short jobs get stuck behind long ones
- RR scheduling:
  - Give each thread a small amount of CPU time when it executes, and cycle between all ready threads
  - Better for short jobs, but poor when jobs are the same length
- SJF/SRTF:
  - Run whatever job has the least amount of computation to do / least amount of remaining computation to do
  - Optimal (average response time), but unfair; hard to predict the future

Conclusion

- Multi-Level Feedback Scheduling:
  - Multiple queues of different priorities
  - Automatic promotion/demotion of process priority to approximate SJF/SRTF
- Lottery Scheduling:
  - Give each thread a number of tickets (short tasks get more)
  - Every thread gets tickets to ensure forward progress / fairness
- Priority Scheduling:
  - Preemptive or Non-preemptive
  - Priority Inversion

johnjose@iitg.ac.in

http://www.iitg.ac.in/johnjose/