OPERATING SYSTEMS

Scheduling Algorithms

Chandravva Hebbi
Department of Computer Science

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

OPERATING SYSTEMS

Priority and Round Robin Scheduling

Suresh Jamadagni

Department of Computer Science

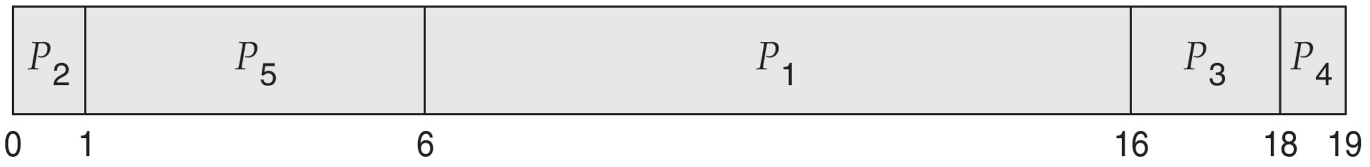
Priority Scheduling

- 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
  - Nonpreemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

Example of Priority Scheduling

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority Scheduling Gantt chart



- Average waiting time = (6 + 0 + 16 + 18 + 1) / 5 = 41/5 = 8.2

Round-Robin Scheduling

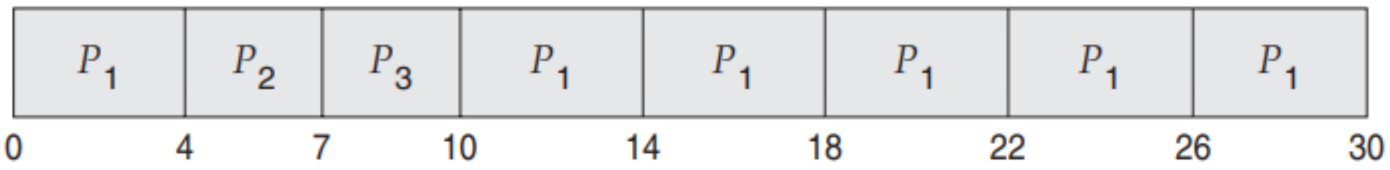
- Round-robin (RR) scheduling algorithm is designed especially for timesharing systems.
- It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes
- A small unit of time, called a time quantum or time slice, is defined.
- A time quantum is generally from 10 to 100 milliseconds in length
- The ready queue is treated as a circular queue
- The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum

Round-Robin Scheduling Example

- Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	24
P_2	3
P_3	3

- RR scheduling Gantt chart using a time quantum of 4 milliseconds



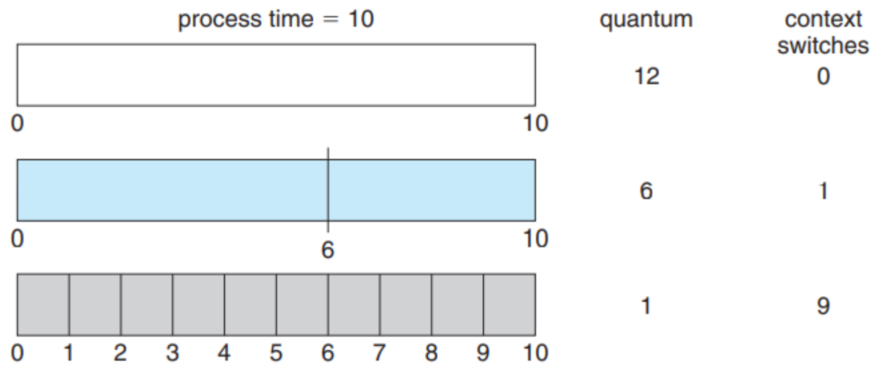
  - P1 wa
  - P2 waits for 4 milliseconds
  - P3 waits for 7 milliseconds.
- The average waiting time = (6 + 4 + 7)/3 = 5.66 milliseconds

Round-Robin Scheduling Performance

- 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.
- Each process must wait no longer than $(n - 1) \times q$ time units until its next time quantum.
  - For example, with five processes and a time quantum of 20 milliseconds, each process will get up to 20 milliseconds every 100 milliseconds.
- Performance of the RR algorithm depends heavily on the size of the time quantum
- If the time quantum is extremely large, the RR policy is the same as the FCFS policy
- If the time quantum is extremely small, the RR approach can result in a large number of context switches

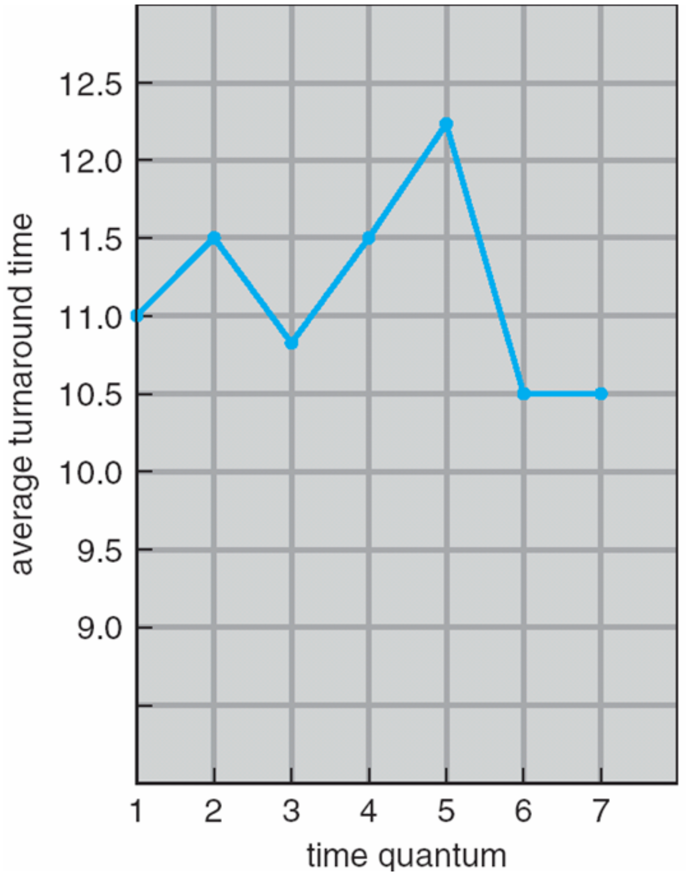
Example of Round-Robin Scheduling Performance

- Consider only one process of 10 time units.
- If the quantum is 12 time units, the process finishes in less than 1 time quantum, with no overhead.
- If the quantum is 6 time units, the process requires 2 quanta, resulting in one context switch.
- If the time quantum is 1 time unit, then nine context switches will occur, slowing the execution of the process accordingly



- In practice, most modern systems have time quanta ranging from 10 to 100 milliseconds.
- The time required for a context switch is typically less than 10 microseconds. Thus, the context-switch time is a small fraction of the time quantum.

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

80% of CPU bursts should be shorter than the time quantum

© copyright Silberschatz, Galvin and Gagne, 2013

THANK YOU

Suresh Jamadagni
Department of Computer Science Engineering
sureshjamadagni@pes.edu