OPERATING SYSTEMS

Scheduling Algorithms

Chandravva Hebbi
Department of Computer Science

OPERATING SYSTEMS

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

SJF, SRTF, Priority and RR Scheduling

Chandravva Hebbi

Department of Computer Science

Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Example of SJF Scheduling

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart

P_4	P_1	P_3	P_2
0–3	3–9	9–16	16–24

- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Note: If using FCFS scheduling, average waiting time = (0 + 6 + 14 + 21) / 4 = 10.25 ms.

Determining Length of Next CPU Burst

- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst
- Can be done by using the length of previous CPU bursts, using exponential averaging

  1. $t_n =$ actual length of n^{th} CPU burst
  2. $\tau_{n+1} =$ predicted value for the next CPU burst
  3. $\alpha, 0 \leq \alpha \leq 1$
  4. Define : $\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\tau_n$.
- Commonly, α set to ½
- Preemptive version called **shortest-remaining-time-first**

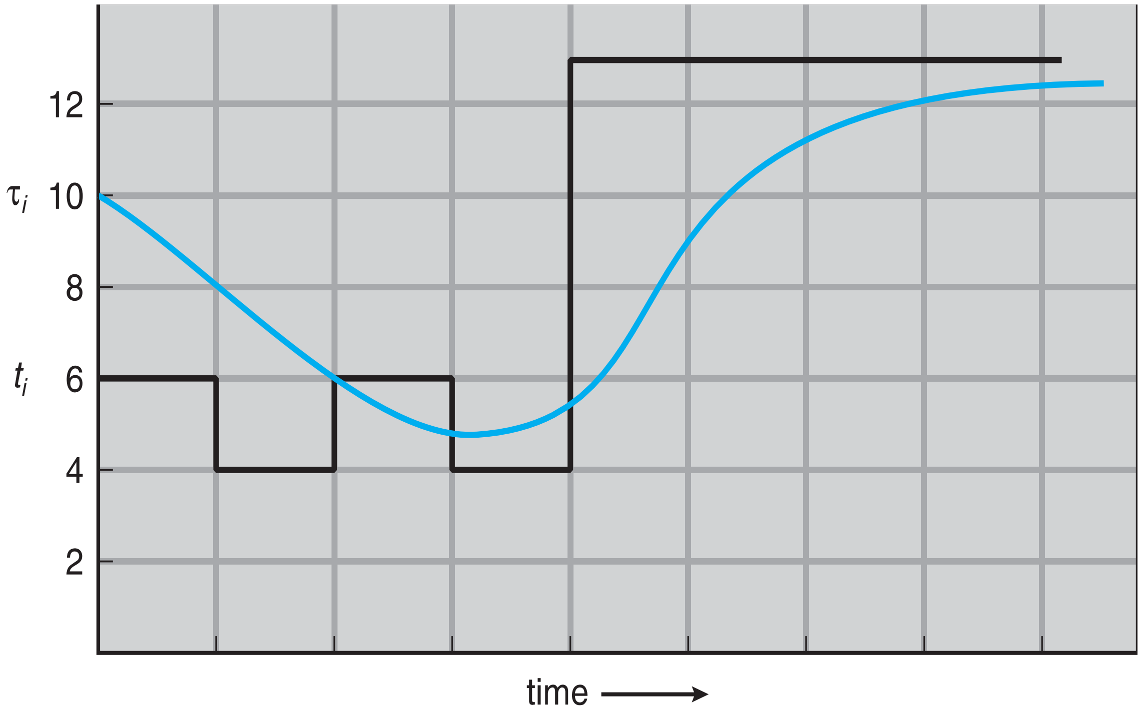
Determining Length of Next CPU Burst

- Calculate the exponential averaging with T1 = 10, α = 0.5 and the algorithm is SJF with previous runs as 8, 7, 4, 16.

Initially T1 = 10 and α = 0.5 and the run times given are 8, 7, 4, 16 as it is shortest job first,
So the possible order in which these processes would serve will be 4, 7, 8, 16 since SJF is a non-preemptive technique.
So, using formula: $T2 = \alpha * t1 + (1-\alpha)T1$
so we have,
$T2 = 0.5*4 + 0.5*10 = 7$, here $t1 = 4$ and $T1 = 10$
$T3 = 0.5*7 + 0.5*7 = 7$, here $t2 = 7$ and $T2 = 7$
$T4 = 0.5*8 + 0.5*7 = 7.5$, here $t3 = 8$ and $T3 = 7$
So the future prediction for 4th process will be $T4 = 7.5$

Src: https://www.geeksforgeeks.org/shortest-job-first-cpu-scheduling-with-predicted-burst-time/

Prediction of the Length of the Next CPU Burst



CPU burst (t_i)		6	4	6	4	13	13	13 …
"guess" (τ_i)	10	8	6	6	5	9	11	12 …

Examples of Exponential Averaging

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_{n-1} + \ldots$$
$$+ (1 - \alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

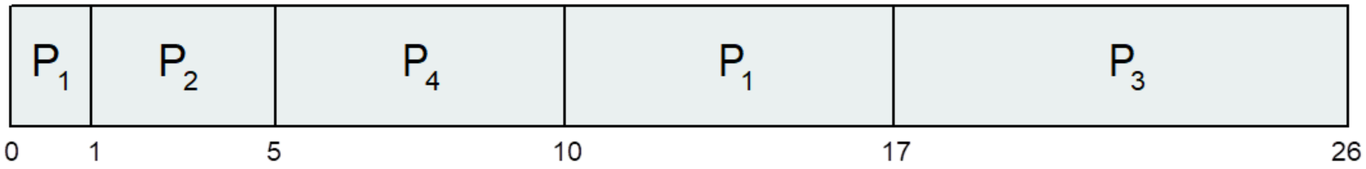
- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

Example of Shortest-remaining-time-first

- Preemptive SJF Scheduling is sometimes called SRTF
- Now we add the concepts of varying arrival times and preemption to the analysis

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

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

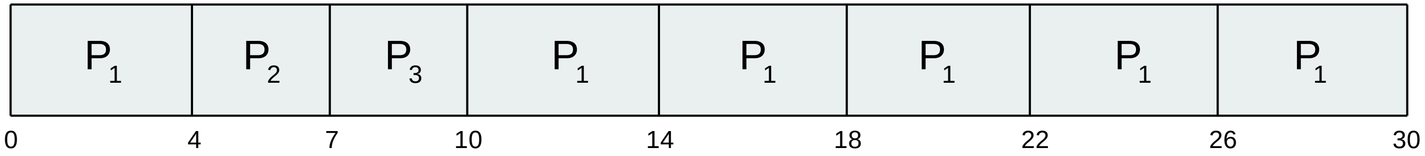
Round Robin (RR)

- Each process gets a small unit of CPU time (**time quantum** *q*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- 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 at once. No process waits more than $(n-1)q$ time units.
- Timer interrupts every quantum to schedule next process
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

Example of RR with Time Quantum = 4

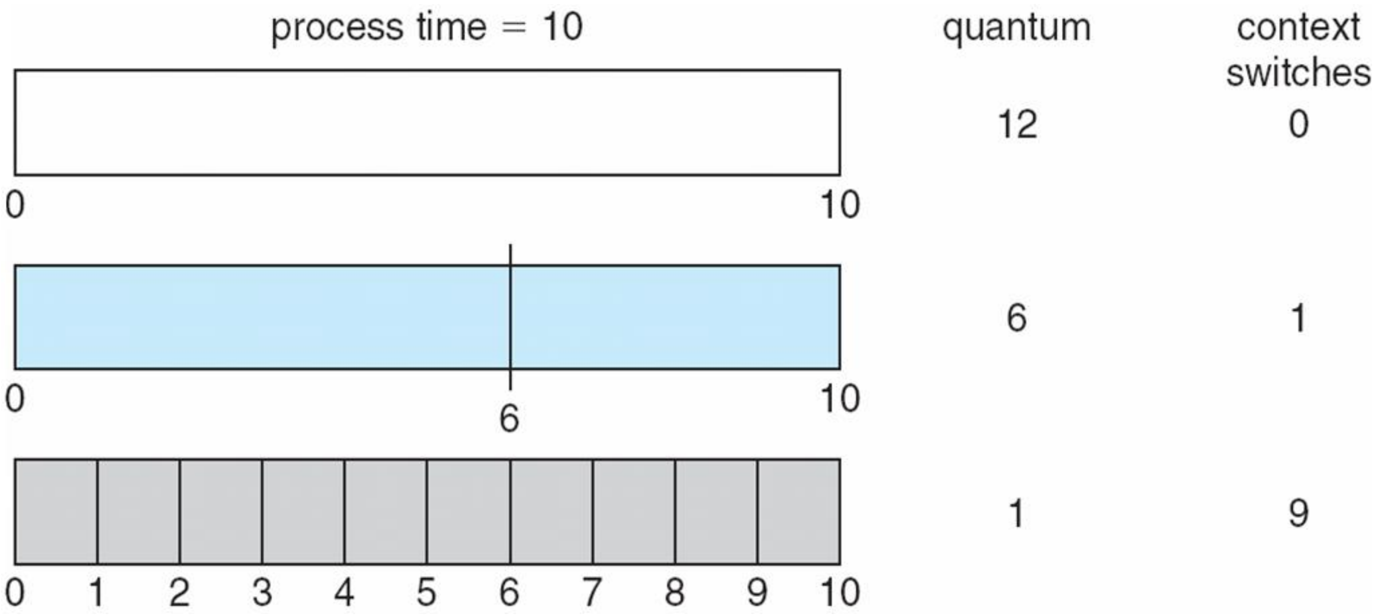
<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- The Gantt chart is:

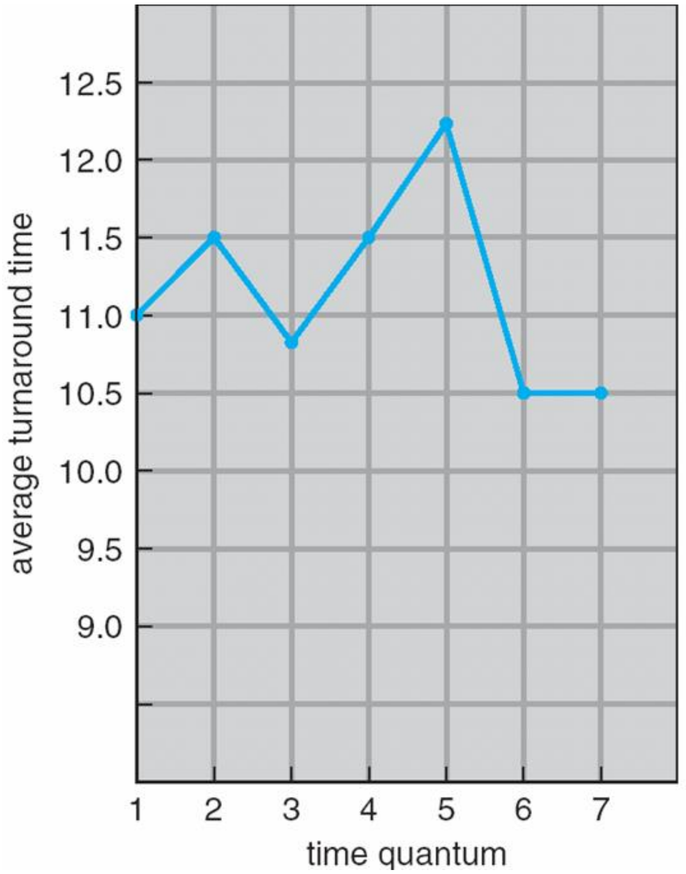


- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

Time Quantum and Context Switch Time



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

OPERATING SYSTEMS

Take Home assignment

5.11

Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?

a. = 0 and τ_0 = 100 milliseconds

b. = 0.99 and τ_0 = 10 milliseconds

THANK YOU

Venkatesh Prasad
Department of Computer Science Engineering

venkateshprasad@pes.edu