Quiz

Process ID	Arrival Time	Priority	Burst Time
A	0	4	15
B	2	1	5
C	8	3	10
D	15	5	1
E	16	2	5

Question: Assuming a **preemptive shortest job first** algorithm is in effect,

a) Draw the Gantt chart for the above processes.
b) Find the response time for each process
c) Find the waiting time for each process
d) Find the turnaround time for each process

CSE 421/521 - Operating Systems
Fall 2013

LECTURE - VI

CPU SCHEDULING - II

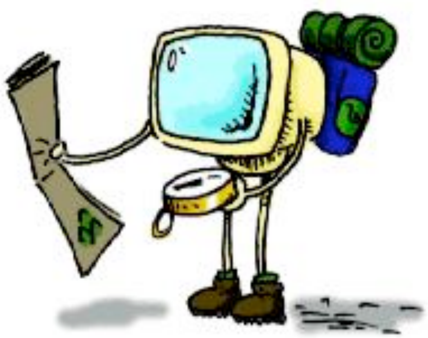
Tevfik Koşar

University at Buffalo

September 19th, 2013

Roadmap

- CPU Scheduling
  - Round-Robin Scheduling
  - Multilevel Feedback Queues
  - Estimating CPU bursts

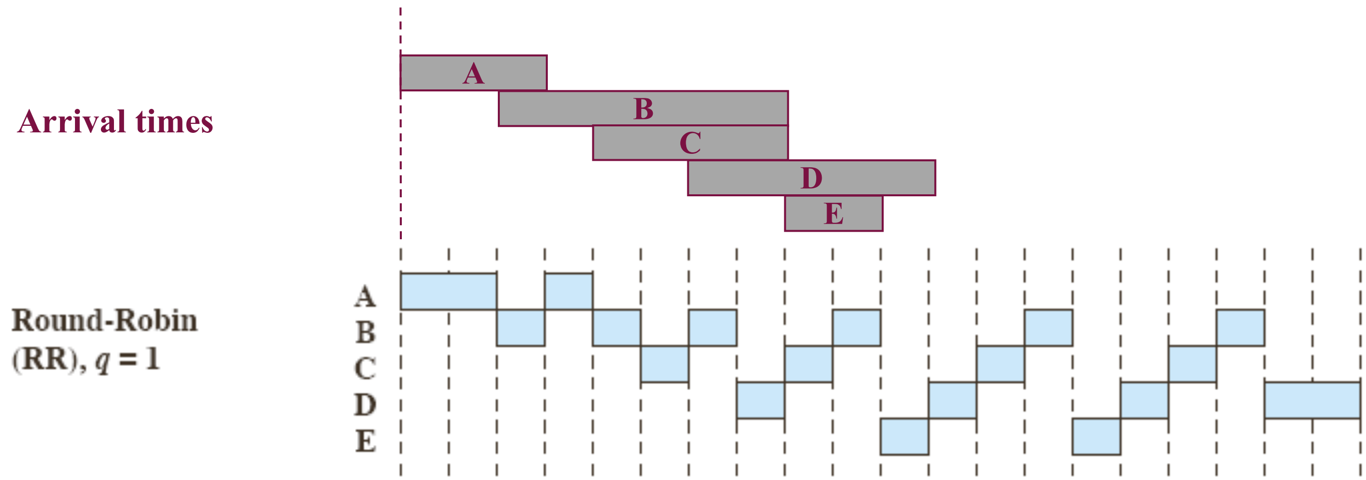


Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), 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.
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

Round Robin (RR)

- ✓ <u>preemptive</u> FCFS, based on a timeout interval, the **quantum** *q*
- ✓ the running process is interrupted by the clock and put last in a FIFO "Ready" queue; then, the first "Ready" process is run instead



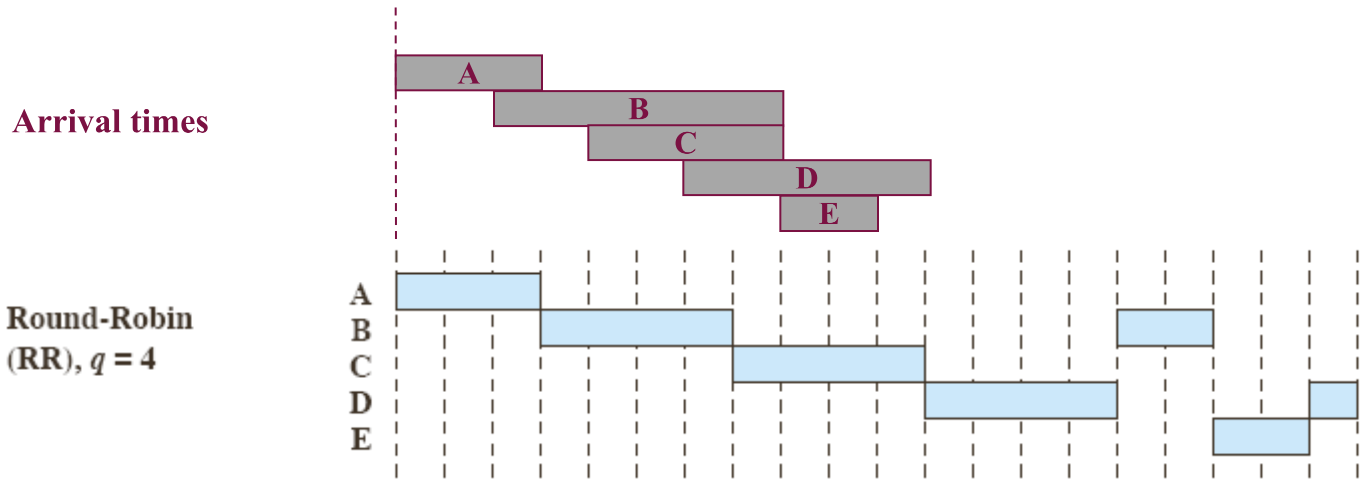
RR $q = 1$							
	Finish Time	A 4	B 18	C 17	D 20	E 15	Mean
	Turnaround Time (T_r)	4	16	13	14	7	10.80
	T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71

RR (q = 1) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Round Robin (RR)

- ✓ a crucial parameter is the quantum *q* (generally ~10–100ms)
  - *q* should be big compared to context switch latency (~10μs)
  - *q* should be less than the longest CPU bursts, otherwise RR degenerates to FCFS



RR $q = 4$	Finish Time Turnaround Time (T_r) T_r/T_s	A 3 3 1.00	B 17 15 2.5	C 11 7 1.75	D 20 14 2.80	E 19 11 5.50	Mean 10.00 2.71

RR (q = 4) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Example of RR with Time Quantum = 20

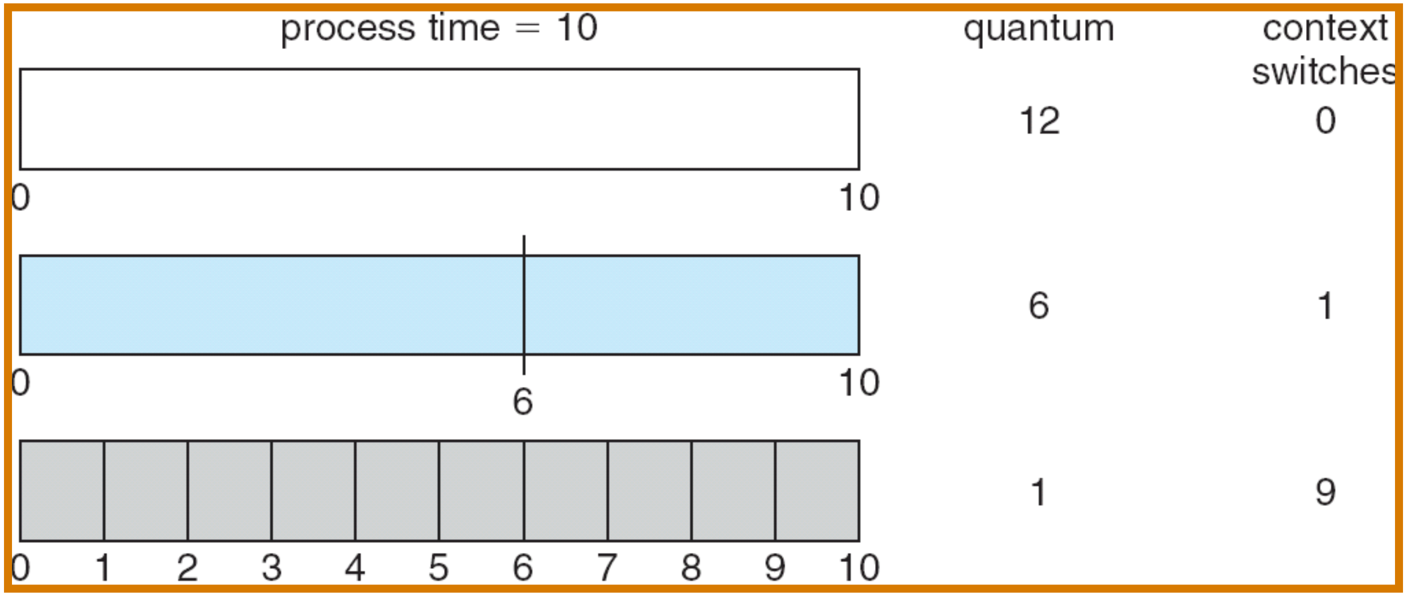
Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- For q=20, the Gantt chart is:

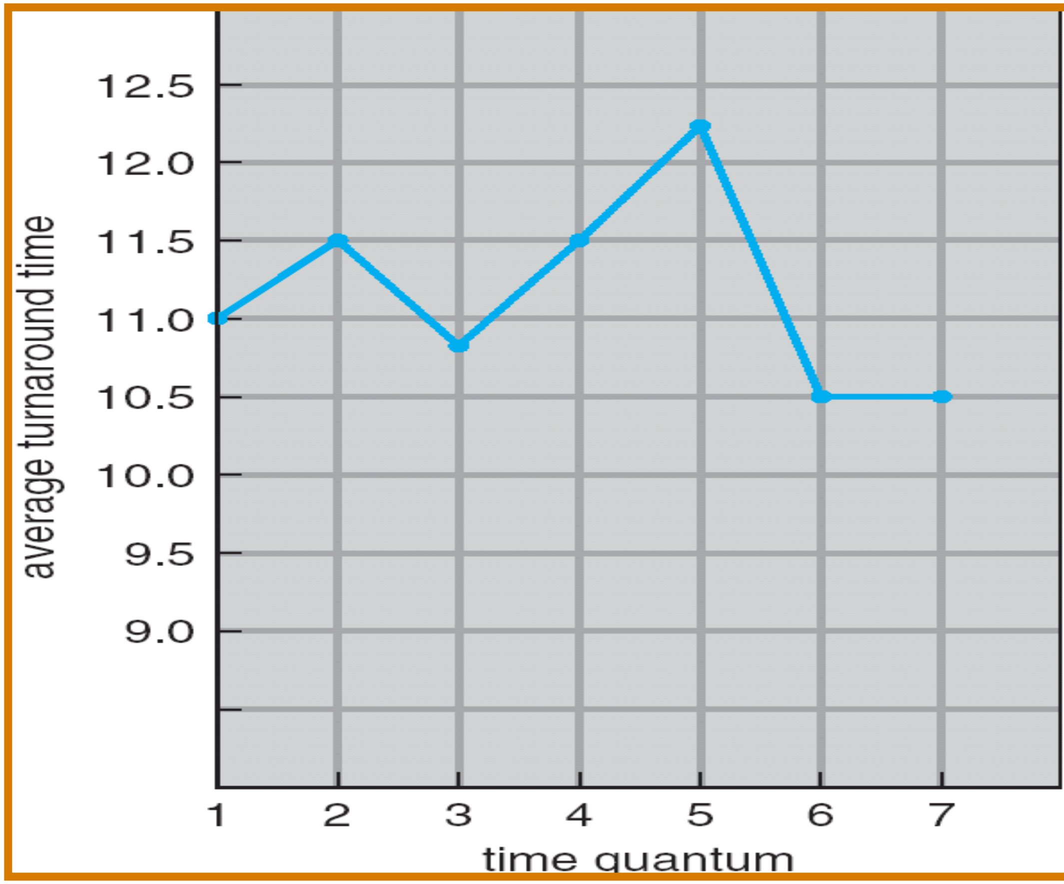
P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3	
0	20	37	57	77	97	117	121	134	154	162

Typically, higher average turnaround than SJF, but better *response*

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

Comparison of Scheduling Algorithms

FCFS

PROS:

- It is a fair algorithm
  - schedule in the order that they arrive

CONS:

- Average response time can be lousy
  - small requests wait behind big ones
- May lead to poor utilization of other resources
  - FCFS may result in poor overlap of CPU and I/O activity
    - E.g., a CPU-intensive job prevents an I/O-intensive job from doing a small bit of computation, thus preventing it from going back and keeping the I/O subsystem busy

SJF

PROS:

- Provably optimal with respect to average response time
  - prevents convoy effect (long delay of short jobs)

CONS:

- Can cause starvation of long jobs
- Requires advanced knowledge of CPU burst times
  - this can be very hard to predict accurately!

SJF

PROS:

- Guarantees early completion of high priority jobs

CONS:

- Can cause starvation of low priority jobs
- How to decide/assign priority numbers?

RR

PROS:

- Great for timesharing
  - no starvation
- Does not require prior knowledge of CPU burst times

CONS:

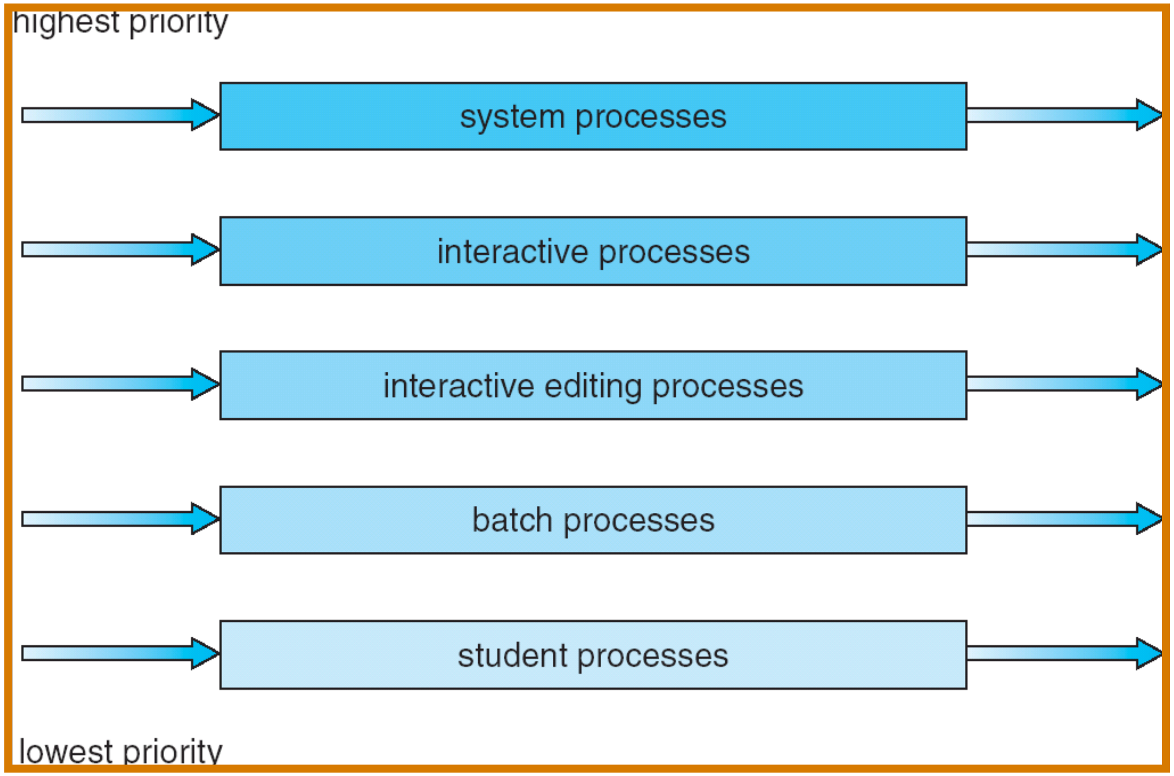
- What if all jobs are almost time same length?
- How to set the “best” time quantum?
  - if small, then context switch often, incurring high overhead
  - if large, then response time degrades

Multilevel Queue

- Ready queue is partitioned into separate queues:
  foreground (interactive)
  background (batch)
- Each queue has its own scheduling algorithm
  - foreground – RR
  - background – FCFS
- Scheduling must be done between the queues
  - Fixed priority scheduling; (i.e., 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 amongst its processes; i.e., 80% to foreground in RR, 20% to background in FCFS

Multilevel Queues

Multilevel Queue Scheduling



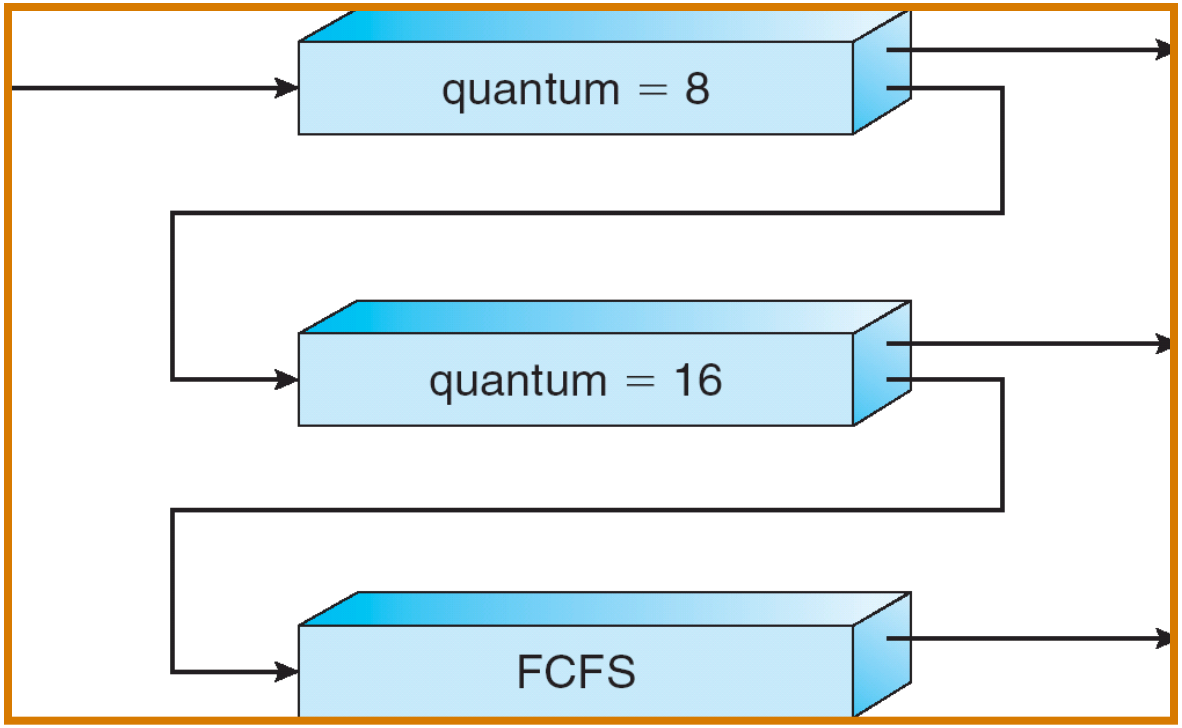
Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- 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:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 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 Q_1.
  - At Q_1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.

Multilevel Feedback Queues



How to estimate CPU burst time?

Determining Length of Next CPU Burst

- Can only estimate the length
- Can be done by using the length of previous CPU bursts, using exponential averaging

$$\tau_{n+1} = \alpha \, t_n + (1 - \alpha) \tau_n .$$

1. $t_n =$ actual lenght of n^{th} CPU burst
2. $\tau_{n+1} =$ predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Define :

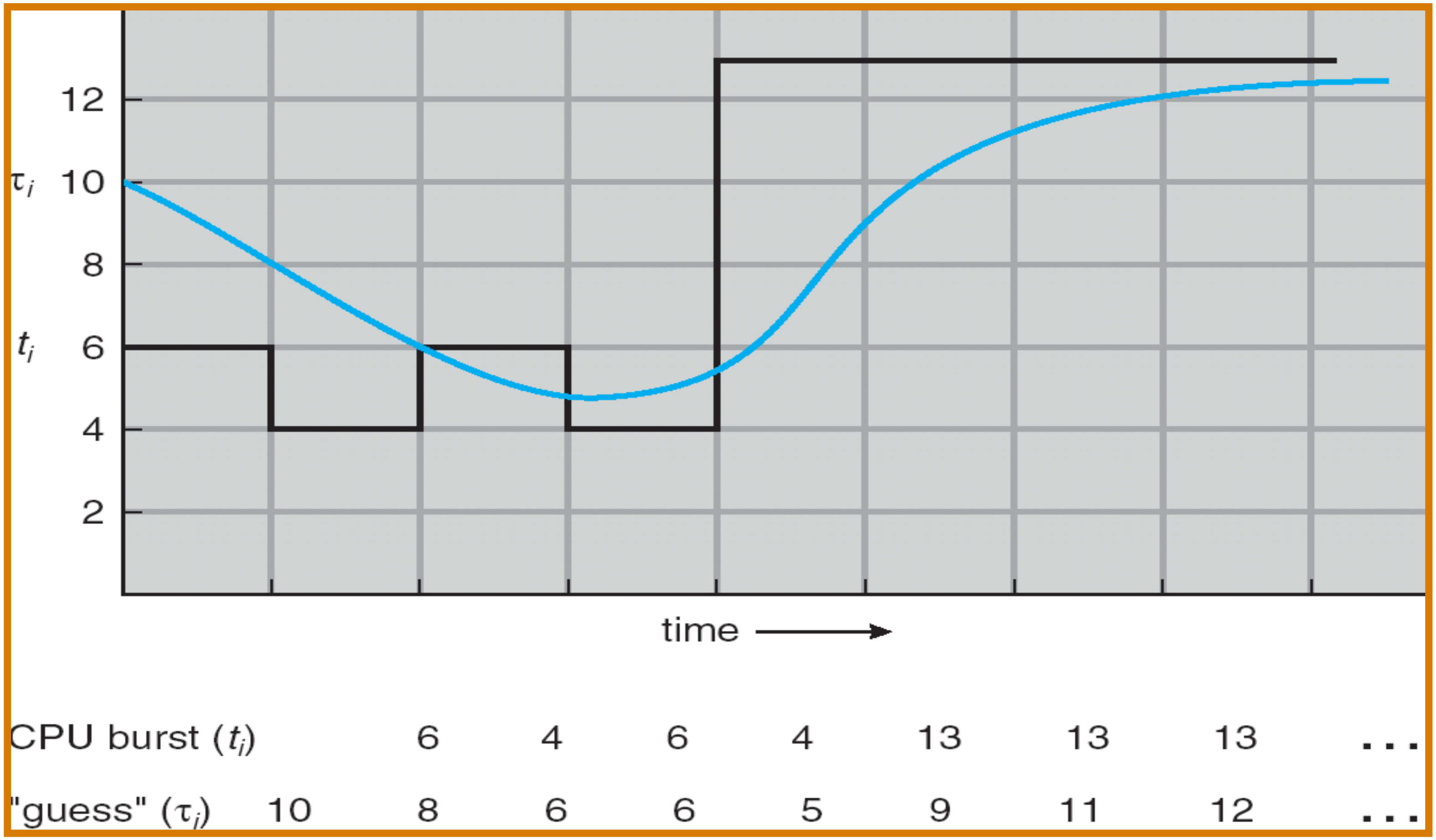
Examples of Exponential Averaging

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

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

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

Prediction of the Length of the Next CPU Burst



Alpha = 1/2, T0 = 10

Exercise

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?

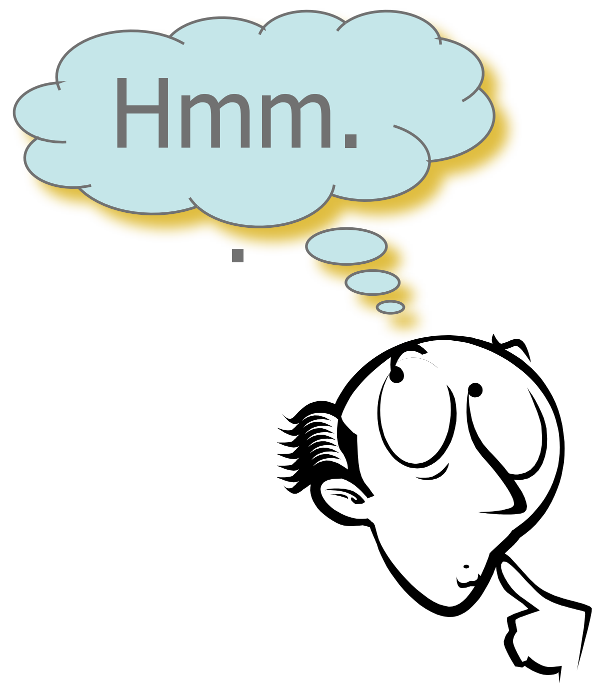
a. $\alpha = 0$ and $\tau_0 = 100 milliseconds$

b. $\alpha = 0.99$ and $\tau_0 = 10 milliseconds$

Answer: When $\alpha = 0$ and $\tau_0 = 100 milliseconds$, the formula always makes a prediction of 100 milliseconds for the next CPU burst. When $\alpha = 0.99$ and $\tau_0 = 10 milliseconds$, the most recent behavior of the process is given much higher weight than the past history associated with the process. Consequently, the scheduling algorithm is almost memory-less, and simply predicts the length of the previous burst for the next quantum of CPU execution.

Summary

- CPU Scheduling
  - Round-Robin Scheduling
  - Multilevel Feedback Queues
  - Estimating CPU bursts



- Next Lecture: Project-1 Discussion

Acknowledgements

- "Operating Systems Concepts" book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- "Operating Systems: Internals and Design Principles" book and supplementary material by W. Stallings
- "Modern Operating Systems" book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR, Ed Lazowska from UWashington