Introduction to Operating Systems
CS 1550

Spring 2023

Sherif Khattab

ksm73@pitt.edu

(Some slides are from **Silberschatz, Galvin and Gagne ©2013)**

Announcements

- Upcoming deadlines
  - Homework 8 is due **this Friday**
  - Quiz 1 and Lab 2 due on Tuesday 2/28 at 11:59 pm
  - Project 2 is due Friday 3/17 at 11:59 pm
- Talk by candidate faculty
  - This Wednesday 3/15 @ 10 am at 5317 Sennott Square
  - Donuts will be served!

Last Lecture …

- How to implement Condition Variables and Locks using semaphores
- CPU scheduling
  - SJF
  - Priority
  - RR

Today …

- CPU scheduling
  - Multi-Level Feedback Queues
  - CPU burst estimation

Multilevel Feedback Scheduling

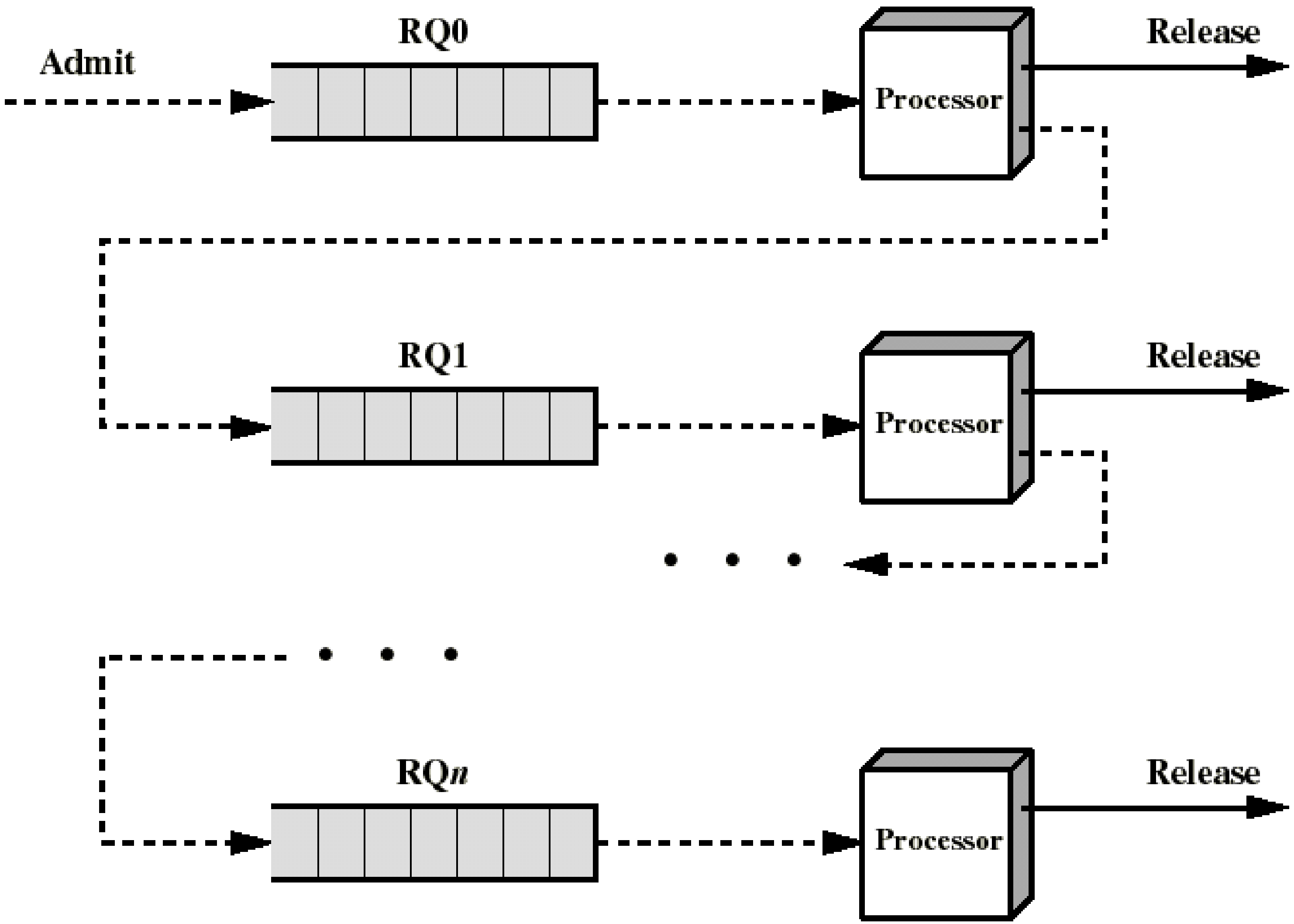
- Preemptive scheduling with dynamic priorities
- N ready to execute queues with decreasing priorities:
  - $P(RQ_0) > P(RQ_1) > \ldots > P(RQ_{N-1})$
- Dispatcher selects a process for execution from RQ_i only if RQ_{i-1} to RQ_0 are empty

Multilevel Feedback Scheduling

- New process are placed in RQ_0
- After the first quantum, they are moved to RQ_1, and to RQ_2 after the second quantum, … and to RQ_{N-1} after the Nth quantum
- I/O-bound processes remain in higher priority queues.
  - CPU-bound jobs drift downward.
  - Hence, long jobs may starve

Multiple Feedback Queues

Different RQs may have different quantum values



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

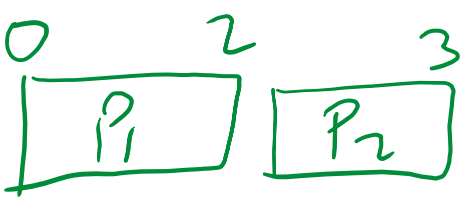
0 2

P1

Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

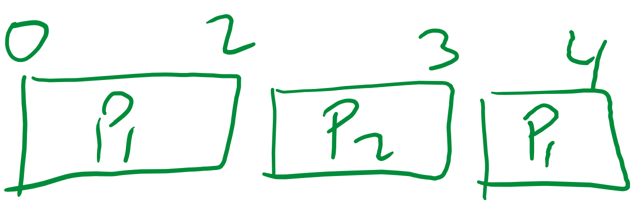
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

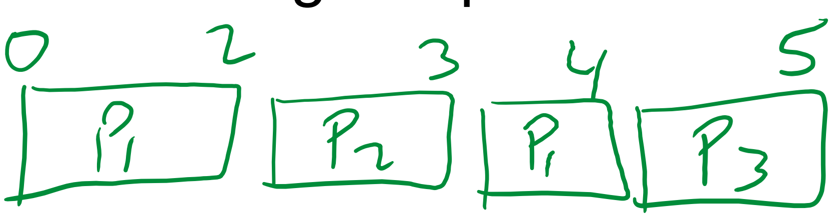
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

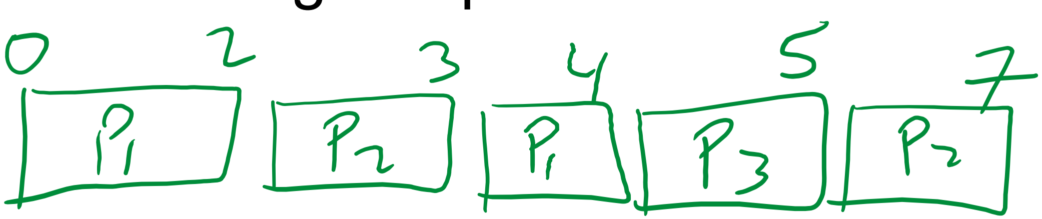
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

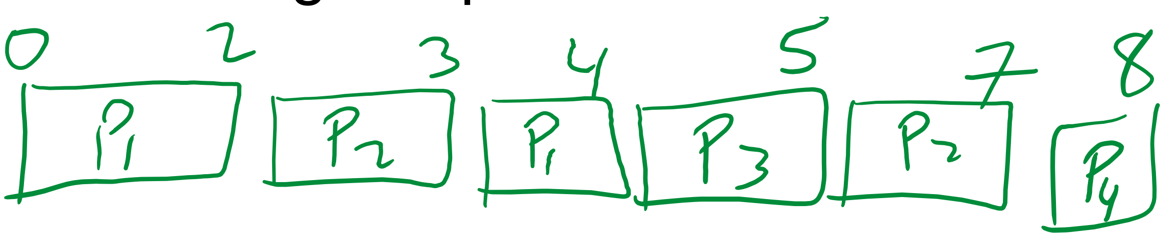
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

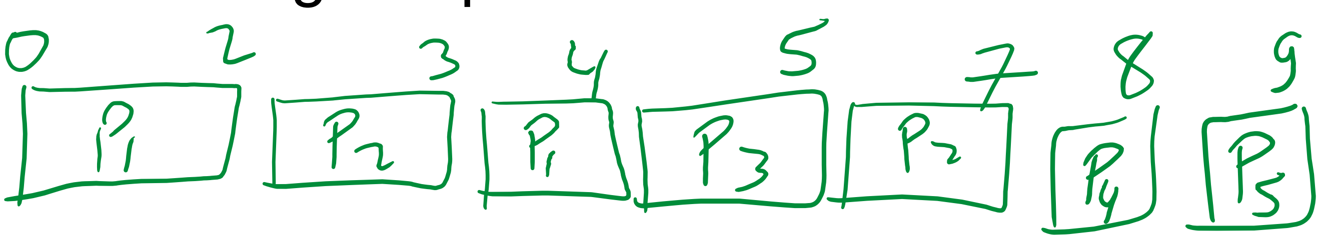
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

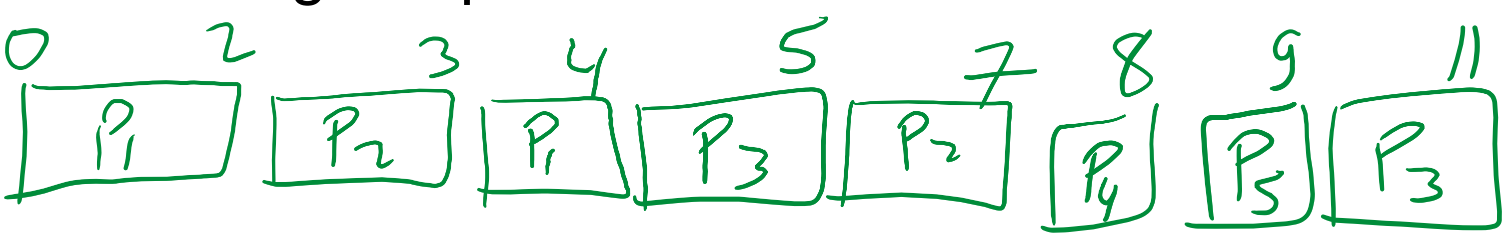
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

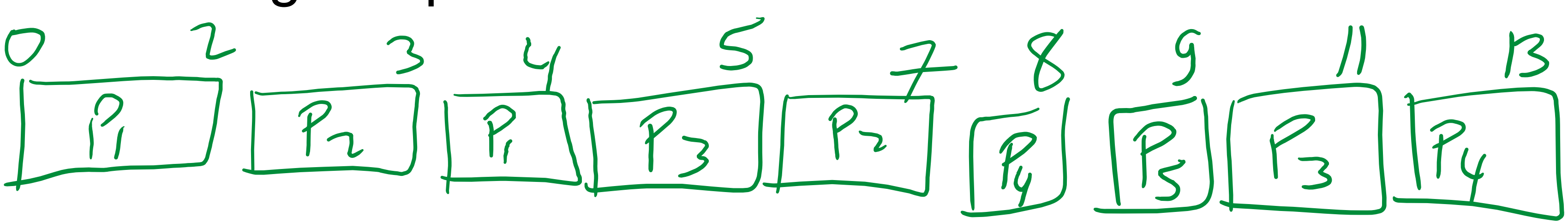
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

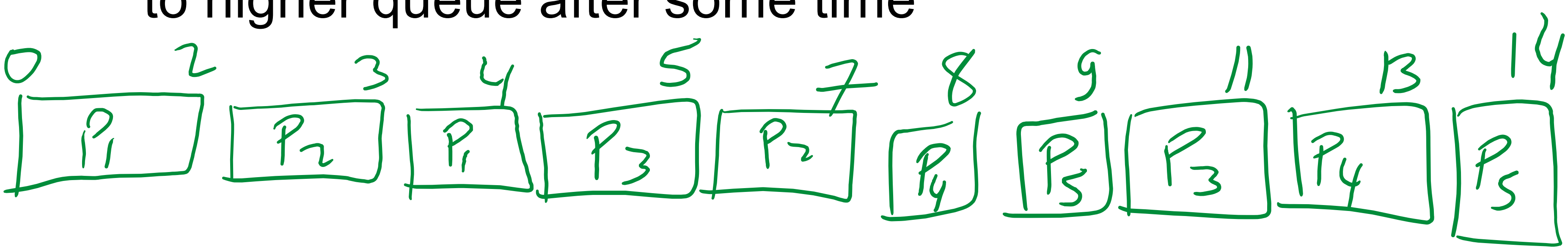
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

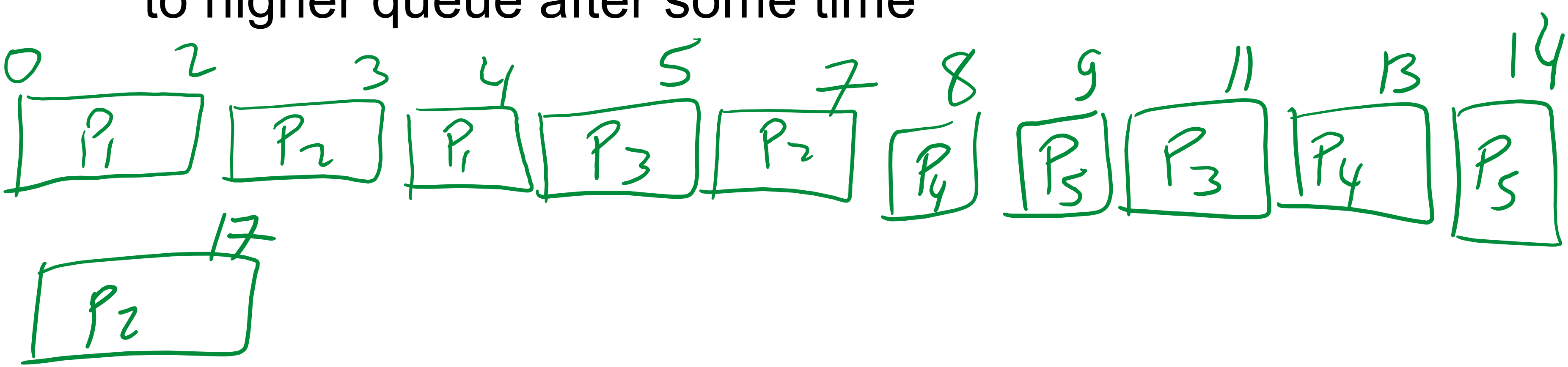
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
    - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
    - Possible fix is to promote a process to higher queue after some time

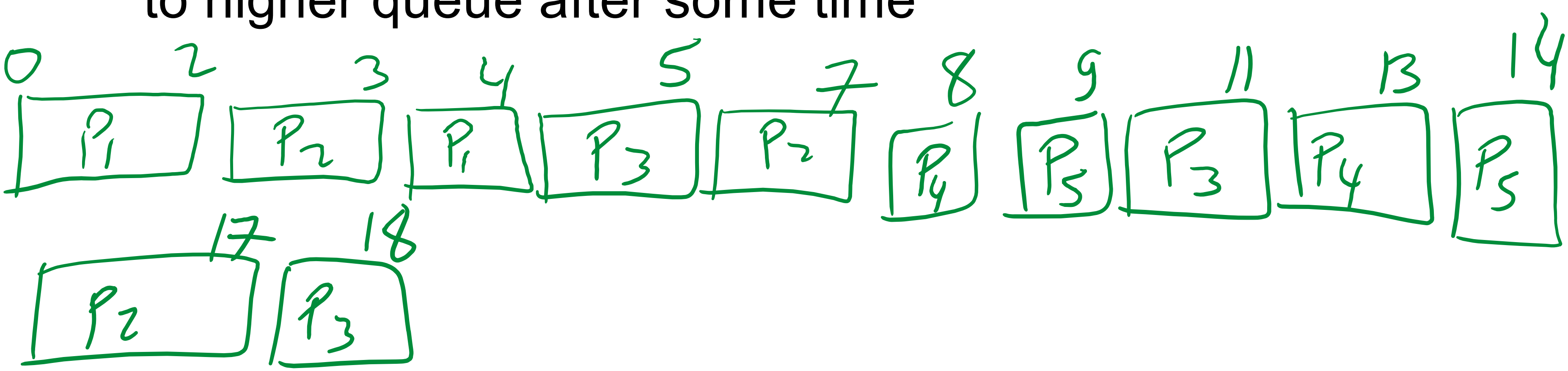
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
    - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
    - Possible fix is to promote a process to higher queue after some time

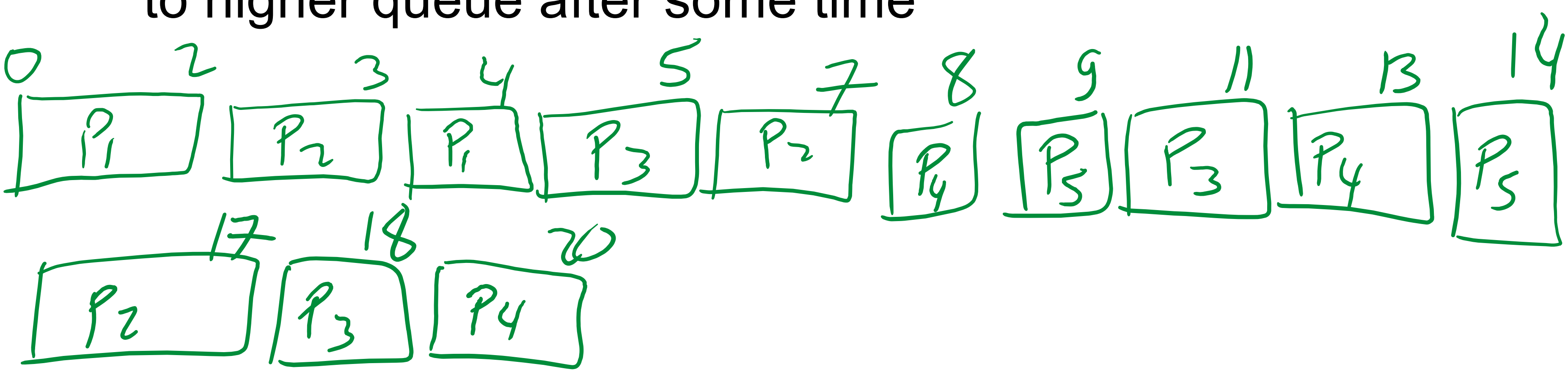
Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Time Quantum for feedback Scheduling

- With a fixed quantum time, the turn-around time of longer processes can be high
- To alleviate this problem, the time quantum can be increased based on the depth of the queue
  - Time quantum of $RQ_i = 2^{i-1}$
- May still cause longer processes to suffer starvation.
  - Possible fix is to promote a process to higher queue after some time

Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Algorithm Comparison

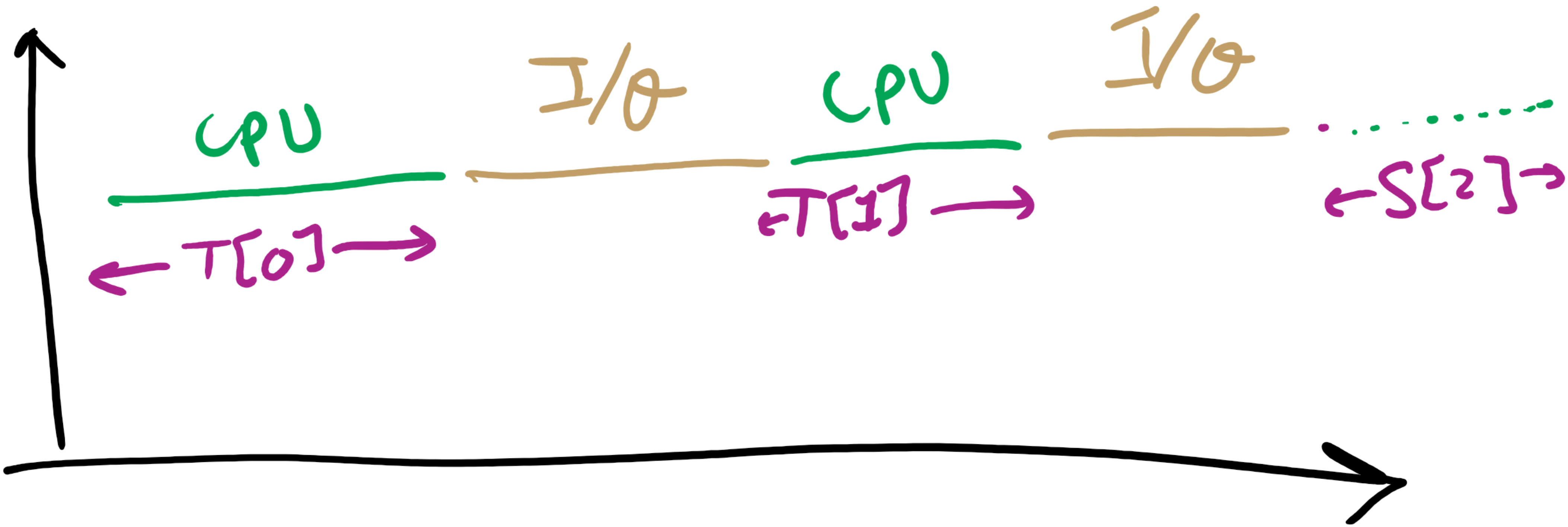
- Which one is the best?
- The answer depends on many factors:
  - the system workload (extremely variable)
  - hardware support for the dispatcher
  - relative importance of performance criteria (response time, CPU utilization, throughput...)
  - The evaluation method used (each has its limitations...)

Back to SJF: CPU Burst Estimation

- Let T[i] be the execution time for the ith instance of this process: the actual duration of the ith CPU burst of this process

- Let S[i] be the predicted value for the ith CPU burst of this process. The simplest choice is:
  - $S[n+1] = (1/n)(T[1]+\ldots+T[n]) = (1/n) \Sigma_{\{i=1 \text{ to } n\}} T[i]$
- This can be more efficiently calculated as:
  - $S[n+1] = (1/n)\ T[n] + ((n-1)/n)\ S[n]$
- This estimate, however, results in equal weight for each instance

CPU Burst Estimation



Estimating the required CPU burst

- Recent instances are more likely to better reflect future behavior
- A common technique to factor the above observation into the estimate is to use exponential averaging :

  - $S[n+1] = \alpha\, T[n] + (1- \alpha)\, S[n]$; $0 < \alpha < 1$

CPU burst Estimate (Exponential Average)

- Recent instances have higher weights, whenever $\alpha > 1/n$

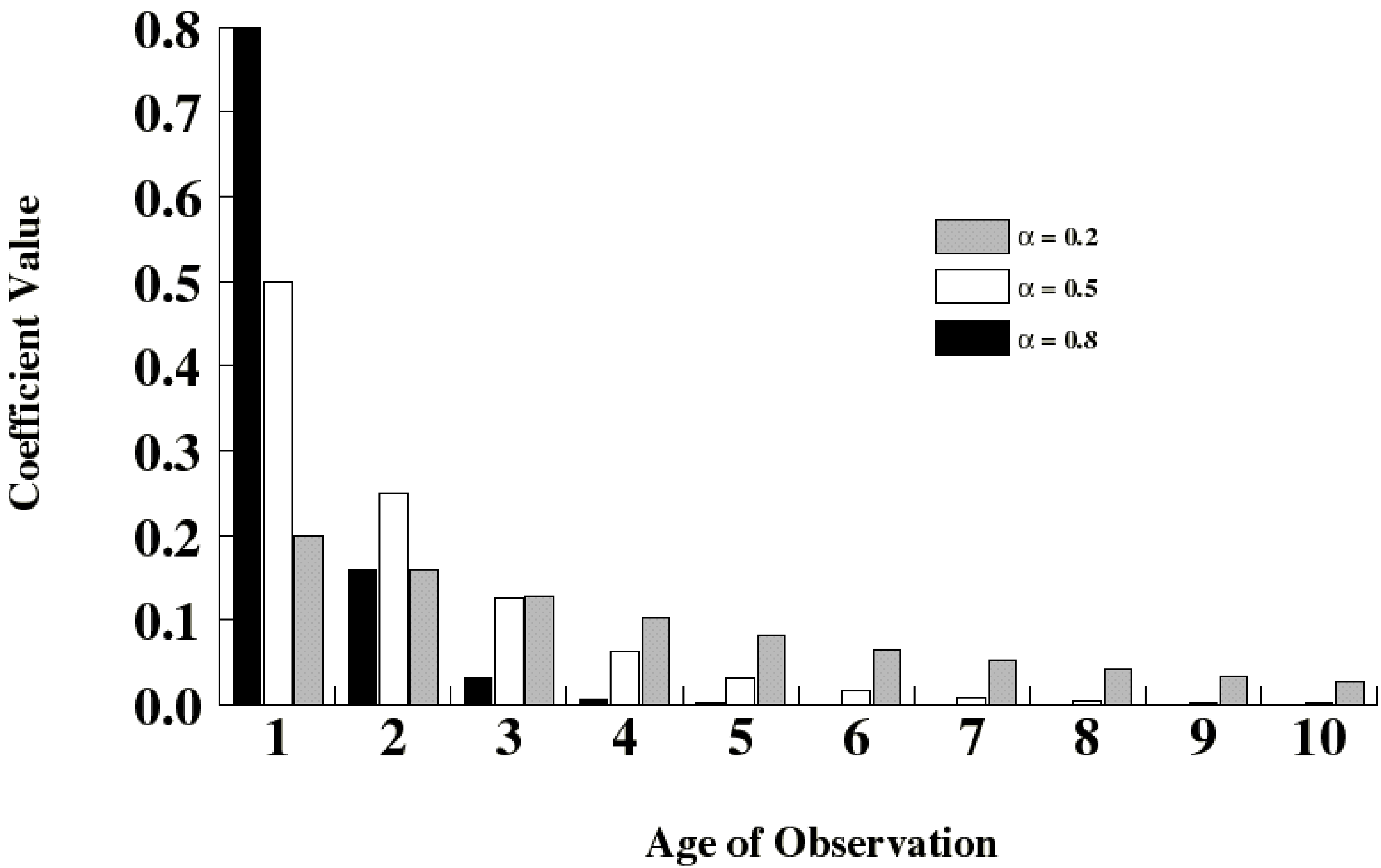
- Expanding the estimated value shows that the weights of past instances decrease exponentially

  - $S[n+1] = \alpha\, T[n] + (1-\alpha)\, \alpha\, T[n-1] + \ldots (1-\alpha)^{i}\, \alpha\, T[n-i] +$

    $\ldots + (1-\alpha)^{n} S[1]$

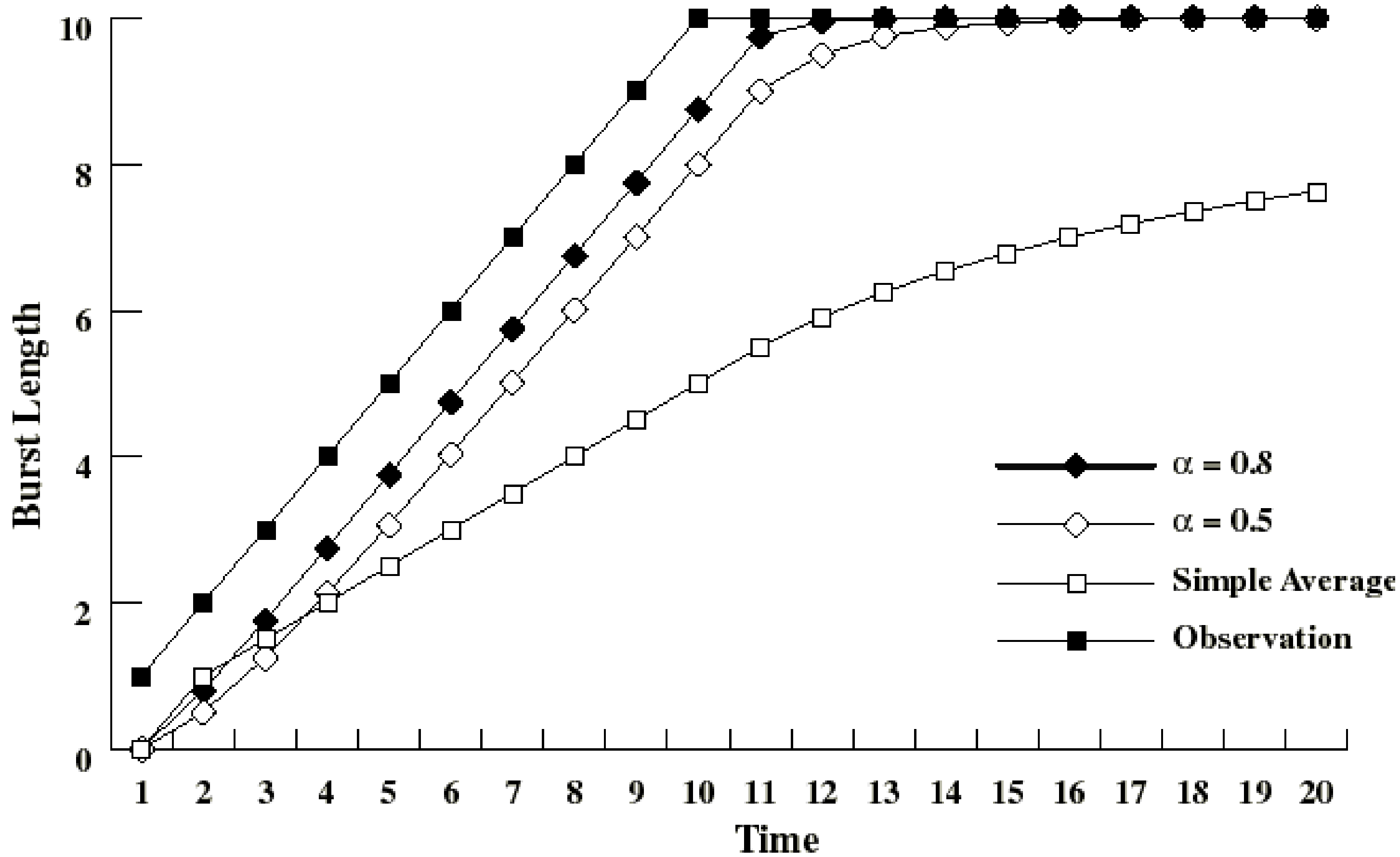
  - The predicted value of 1st instance, S[1], is usually set to 0 to give priority to new processes

Exponentially Decreasing Coefficients

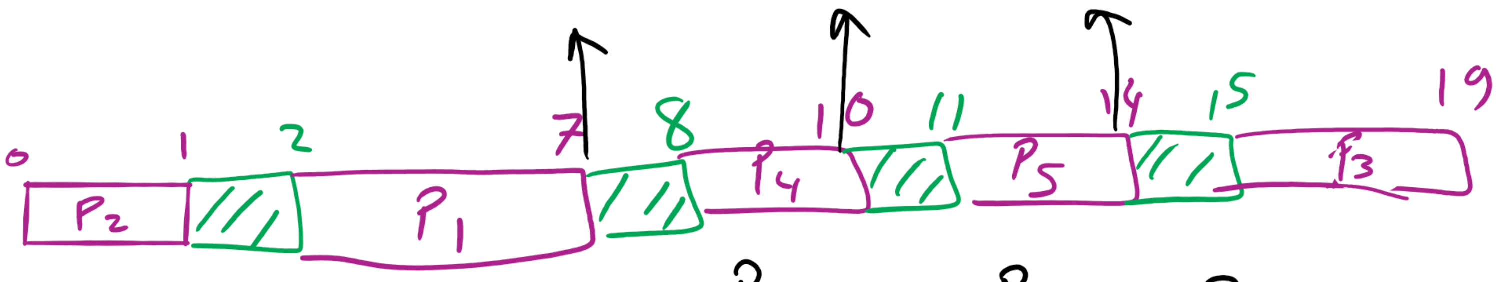


Exponentially Decreasing Coefficients

- S[1] = 0 to give high priority to new processes
- Exponential averaging tracks changes in process behavior much faster than simple averaging



FCFS Problem in HW7



$$A.R.T. = \frac{\overset{P_1}{(7-2)} + \overset{P_2}{(1-0)} + \overset{P_3}{(19-8)} + \overset{P_4}{(10-4)} + \overset{P_5}{(14-7)}}{5}$$

$$= \frac{5+1+11+6+7}{5} = 6ms$$

$$CPU\ util = \frac{19-4}{19} = \frac{15}{19}$$

CPU Burst Estimation

$$S[n+1] = \alpha T[n] + (1-\alpha) S[n]$$

$$0 < \alpha < 1$$