Operating System

Chapter 9. Uniprocessor Scheduling

Lynn Choi

School of Electrical Engineering

Processor Scheduling

Scheduling

- Assign system resource (CPU, IO device, etc.) to processes/threads to meet system objectives (response time, turnaround time, throughput, or fairness..)
  - In practice, these goals often conflict

Three types of processor scheduling

- Long-term scheduling (*admission scheduler*)
  - Decide which jobs/processes to be admitted to the system
    - Control the degree of multiprogramming
  - Once admitted, a job or user program becomes a process
    - The process may be added to the ready queue for the short-term scheduler or to a queue for the medium-term scheduler in a swapped-out condition
- Mid-term scheduling (*swapper*)
  - Remove processes from main memory and place them on secondary memory, or vice versa
  - Swap in/out processes
- Short-term scheduling (*dispatcher*)
  - Decide which of the ready, in-memory processes to be executed by the processor following a clock interrupt, IO interrupt, or OS call
  - Execute most frequently

Scheduling and Process State Transitions

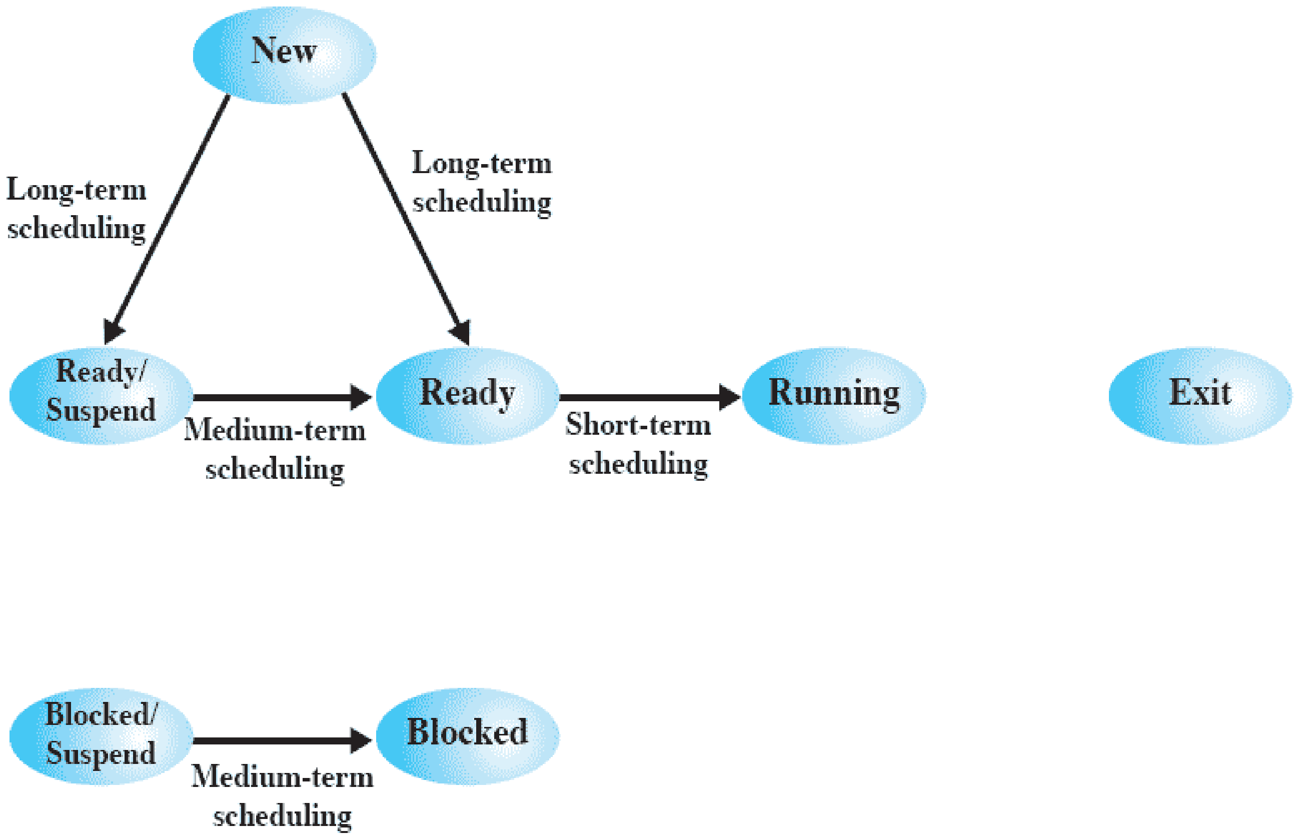
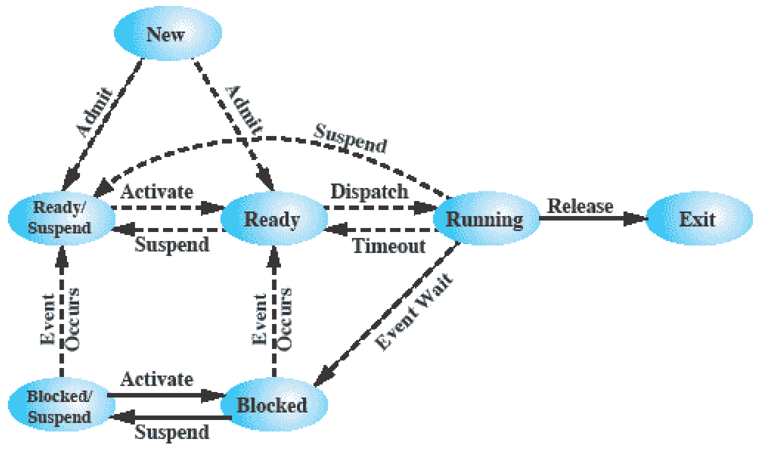


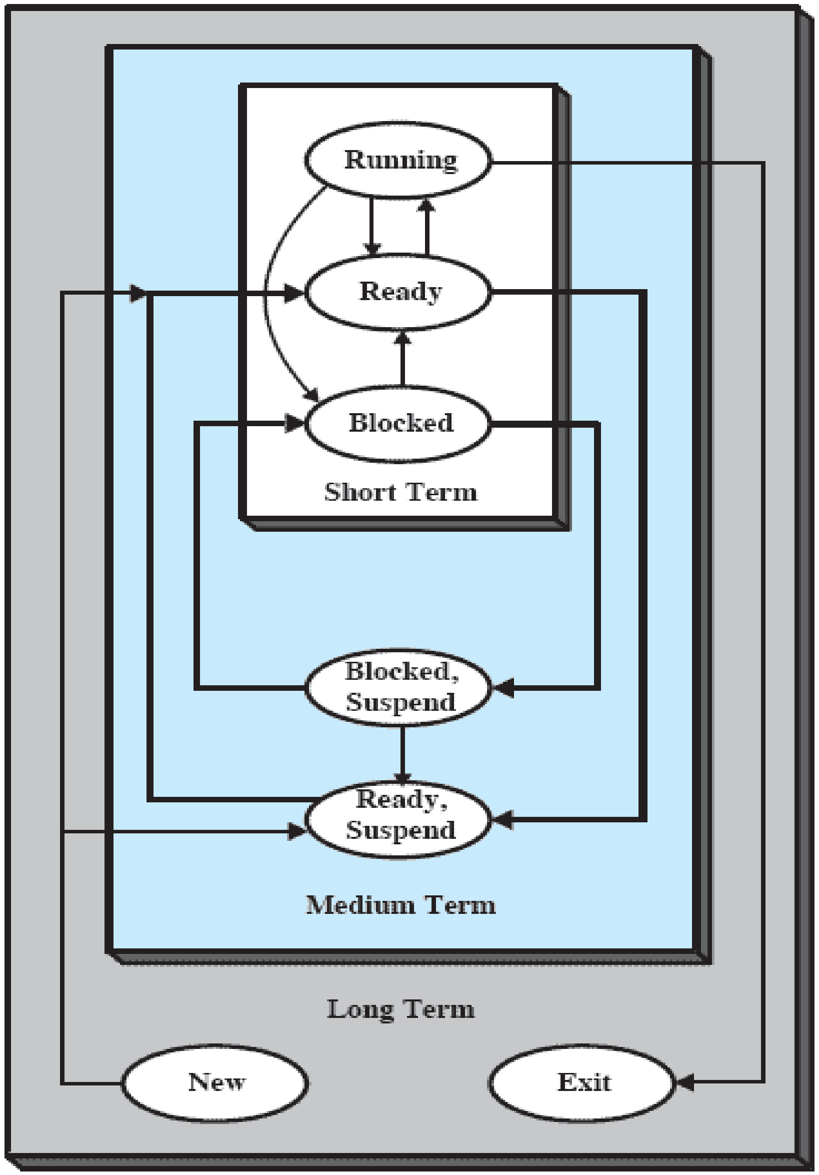
Figure 9.1 Scheduling and Process State Transitions

Source: Pearson

Nesting of Scheduling Functions

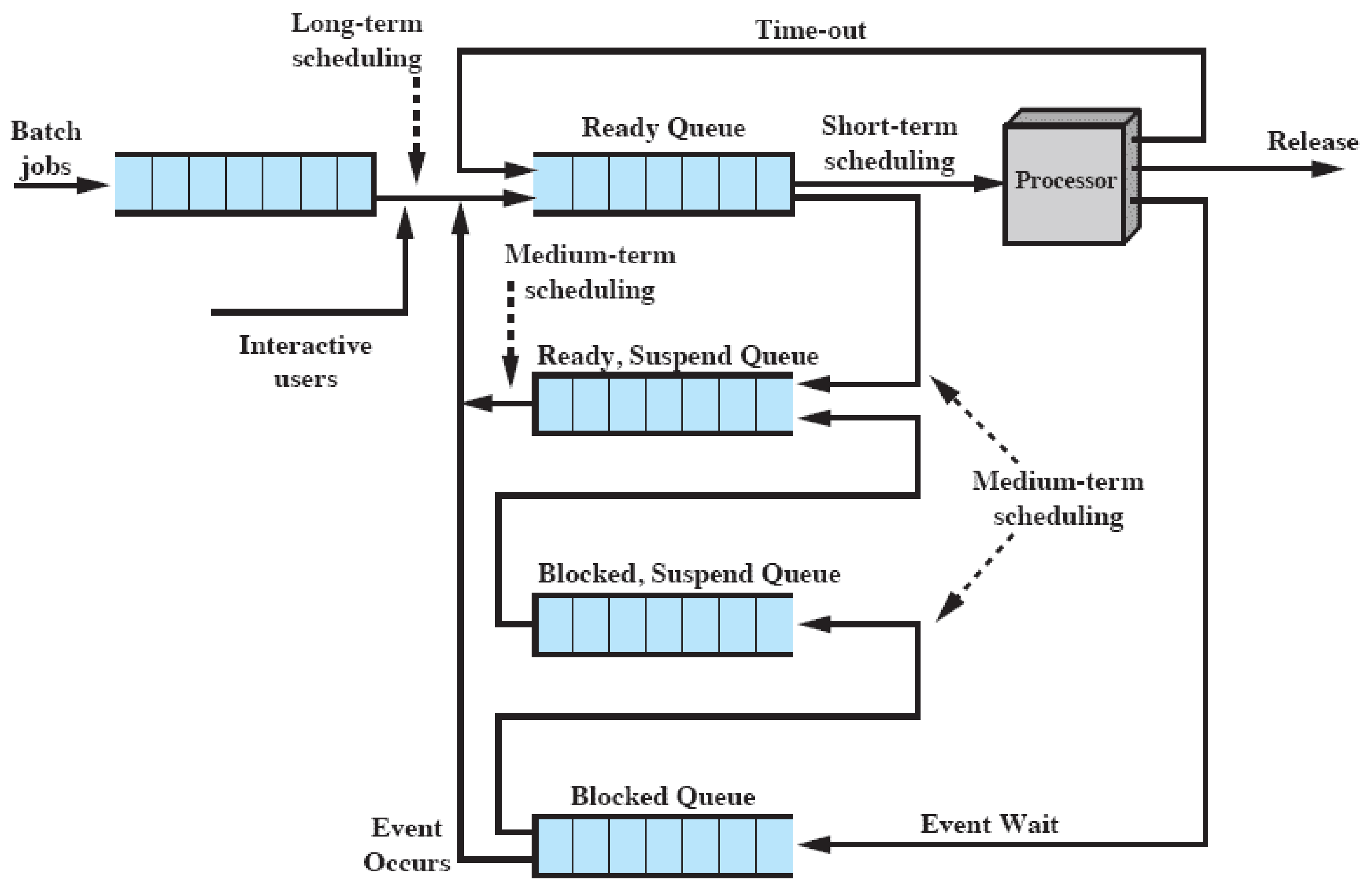


(b) With Two Suspend States



Source: Pearson

Queuing Diagram



Source: Pearson

Figure 9.3 Queuing Diagram for Scheduling

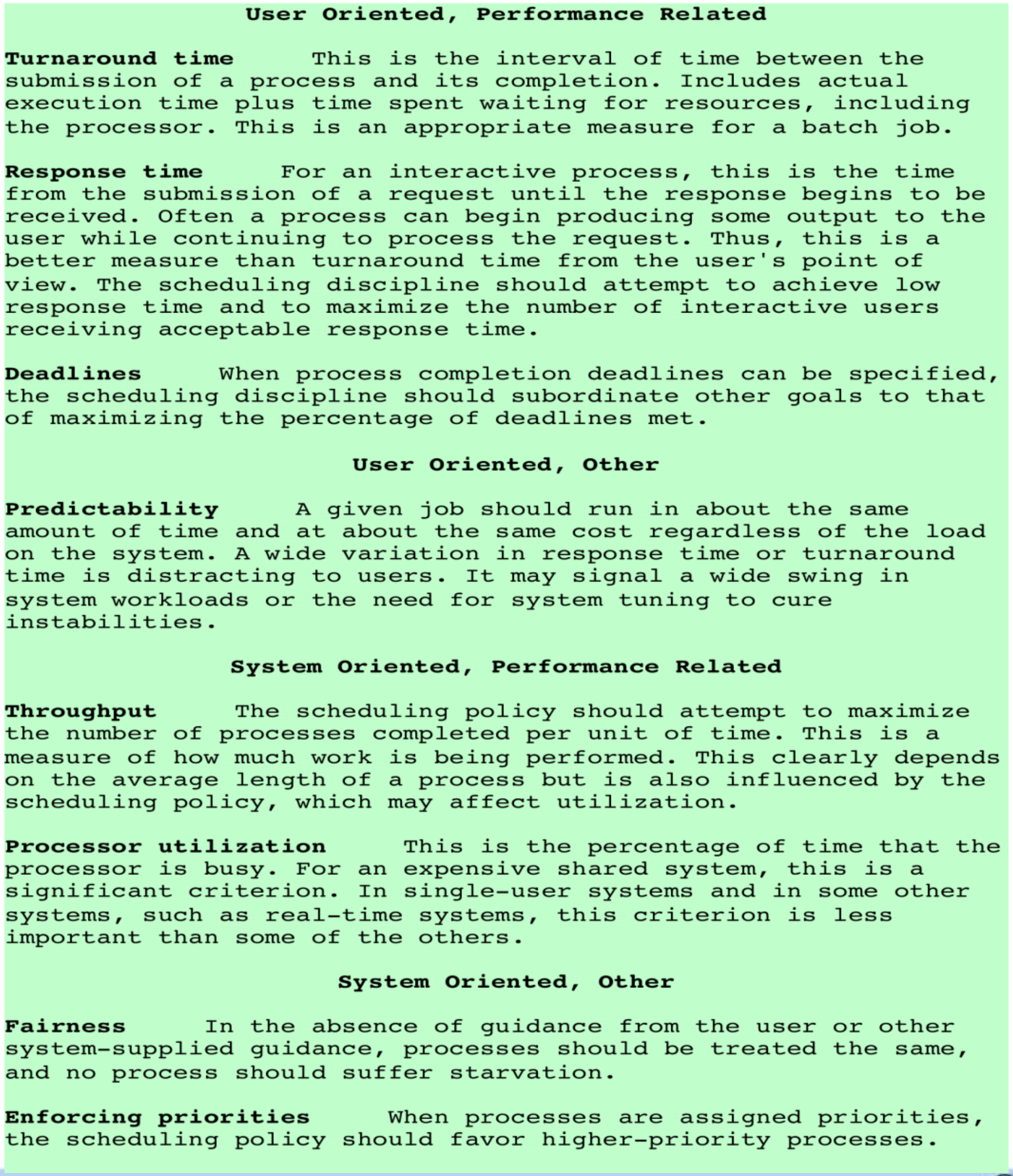
Long-Term Scheduler

- **Determine which programs are admitted to the system for processing**
  - Once admitted a user program becomes a process
- **Control the degree of multiprogramming**
  - The more processes that are created, the smaller the percentage of time that each process can be executed
    - May limit the degree of multiprogramming to provide satisfactory service to the current set of processes
    - Or, may increase the degree of multiprogramming if CPU is idle too long
- **Which jobs to admit next can be**
  - First come, first served (FCFS), or
  - Priority, expected execution time, I/O requirements
- **For interactive programs in a time-sharing system**
  - OS will accept all authorized comers until the system is saturated

Short-Term Scheduling Criteria

- **The main objective of short-term scheduling is to allocate processor time to optimize system behaviour**
- **Can be categorized into two dimensions**
  - User-oriented criteria
    - Relate to the behaviour of the system as perceived by the individual user or process (such as response time in an interactive system)
    - Important on virtually all systems
  - System-oriented criteria
    - Focus on efficient utilization of the processor such as throughput
    - Generally of minor importance on single-user systems
  - Performance-related criteria
    - Quantitative and can be measured
    - Example: response time, throughput
  - Non-performance-related criteria
    - Qualitative and hard to measure
    - Example: predictability, priority

Scheduling Criteria



User Oriented, Performance Related

Turnaround time This is the interval of time between the submission of a process and its completion. Includes actual execution time plus time spent waiting for resources, including the processor. This is an appropriate measure for a batch job.

Response time For an interactive process, this is the time from the submission of a request until the response begins to be received. Often a process can begin producing some output to the user while continuing to process the request. Thus, this is a better measure than turnaround time from the user's point of view. The scheduling discipline should attempt to achieve low response time and to maximize the number of interactive users receiving acceptable response time.

Deadlines When process completion deadlines can be specified, the scheduling discipline should subordinate other goals to that of maximizing the percentage of deadlines met.

User Oriented, Other

Predictability A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system tuning to cure instabilities.

System Oriented, Performance Related

Throughput The scheduling policy should attempt to maximize the number of processes completed per unit of time. This is a measure of how much work is being performed. This clearly depends on the average length of a process but is also influenced by the scheduling policy, which may affect utilization.

Processor utilization This is the percentage of time that the processor is busy. For an expensive shared system, this is a significant criterion. In single-user systems and in some other systems, such as real-time systems, this criterion is less important than some of the others.

System Oriented, Other

Fairness In the absence of guidance from the user or other system-supplied guidance, processes should be treated the same, and no process should suffer starvation.

Enforcing priorities When processes are assigned priorities, the scheduling policy should favor higher-priority processes.

Source: Pearson

Priority Queuing

Ready queue with the highest priority

Ready queue with the lowest priority

May suffer from starvation!

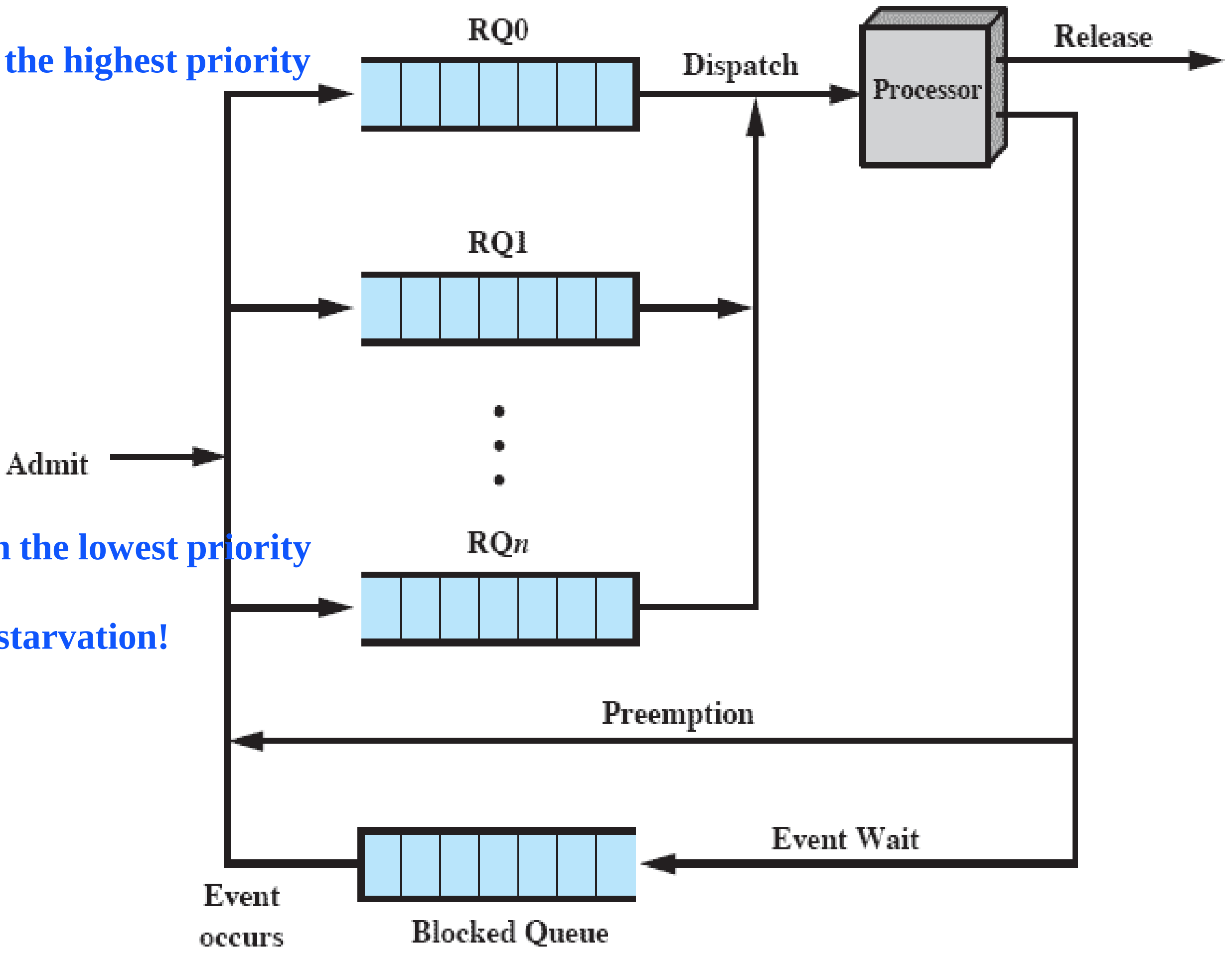
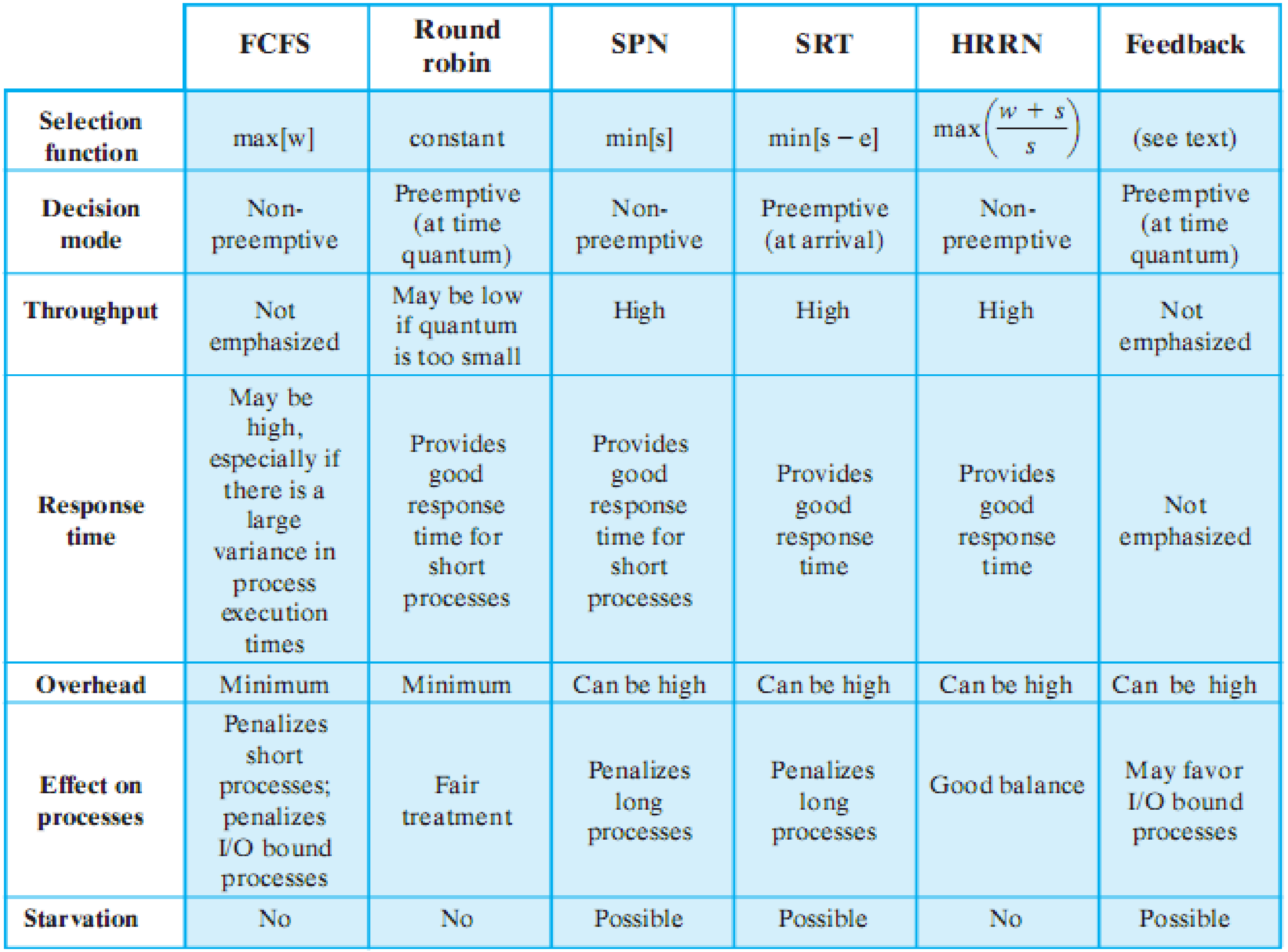


Figure 9.4 Priority Queuing

Source: Pearson

Various Scheduling Policies

Table 9.3 Characteristics of Various Scheduling Policies



	FCFS	Round robin	SPN	SRT	HRRN	Feedback
Selection function	max[w]	constant	min[s]	min[s − e]	$\max\left(\frac{w+s}{s}\right)$	(see text)
Decision mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive	Preemptive (at time quantum)
Throughput	Not emphasized	May be low if quantum is too small	High	High	High	Not emphasized
Response time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time	Provides good response time	Not emphasized
Overhead	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
Effect on processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes	Good balance	May favor I/O bound processes
Starvation	No	No	Possible	Possible	No	Possible

Source: Pearson

Selection Function

- **Determine which process, among ready processes, is selected next for execution**
- **May be based on priority, resource requirements, or the execution characteristics of the process**
- **If based on execution characteristics then important quantities are**
  - ***w*** = time spent in system so far (waiting)
  - ***e*** = time spent in execution so far
  - ***s*** = total service time required by the process, including *e*; generally, this quantity must be estimated or supplied by the user

Decision Mode

- Specifies the instant in time at which the selection function is exercised
- Two categories
  - Nonpreemptive
    - Once a process is in the running state, it will continue until it terminates or blocks itself for I/O
  - Preemptive
    - Currently running process may be interrupted by the scheduler and moved to ready state
    - Preemption may occur anytime (when new process arrives, when an interrupt occurs that places a blocked process in the Ready queue, or periodically based on a clock interrupt)

Process Scheduling Example

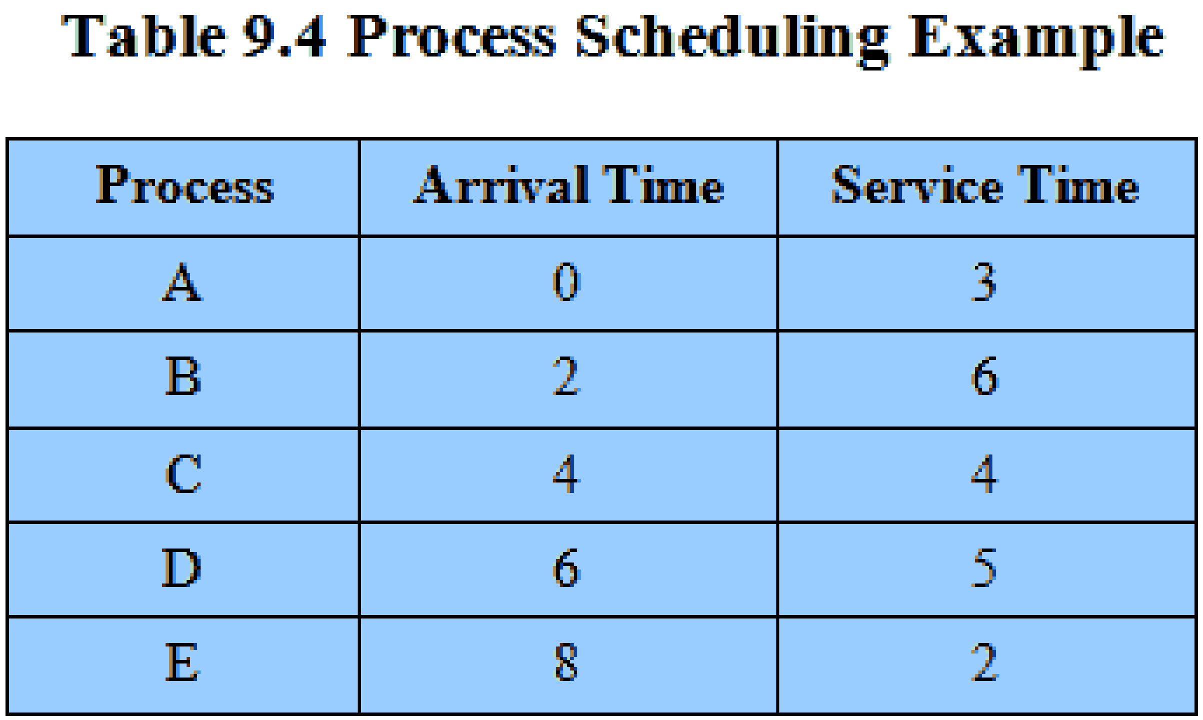
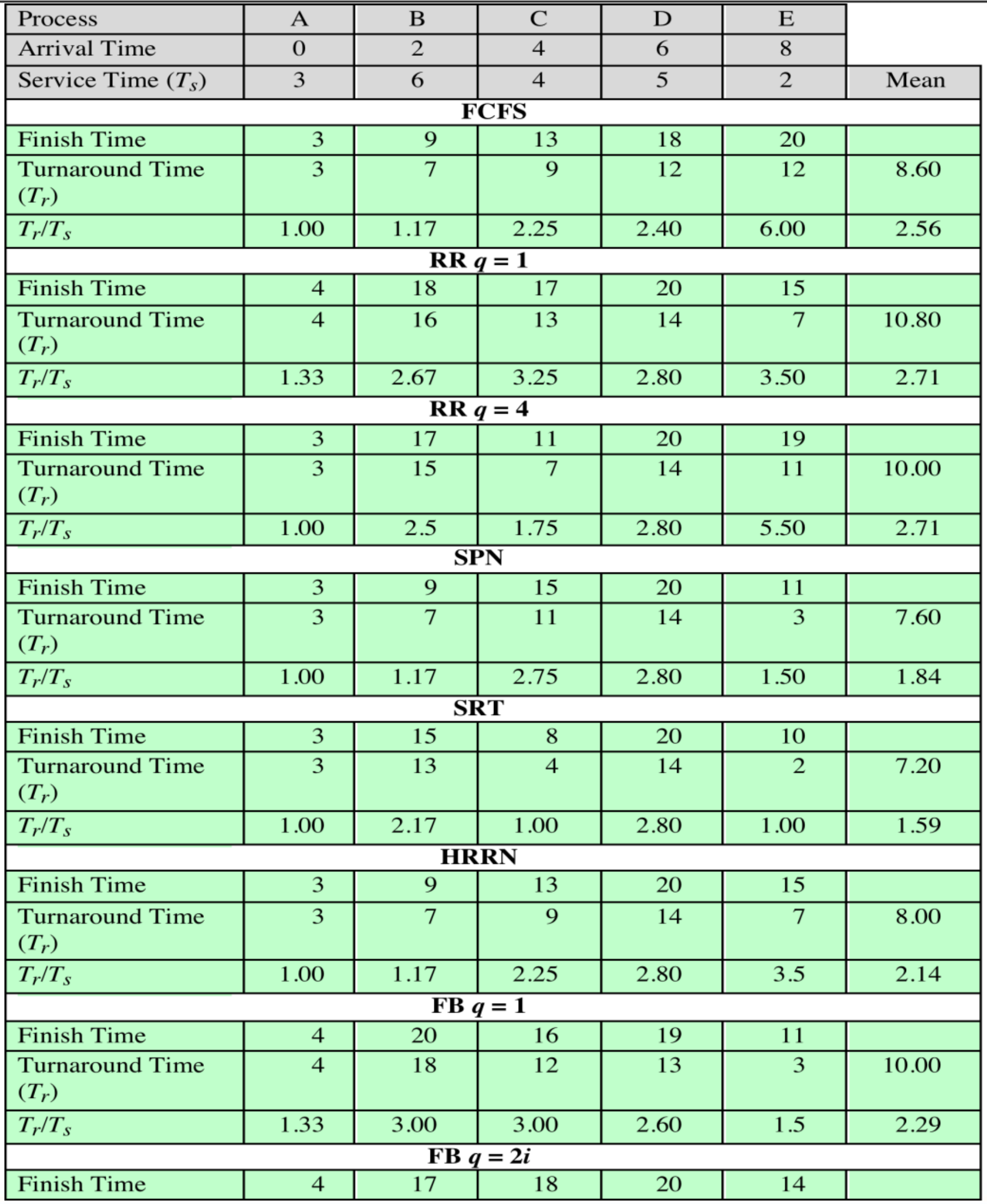


Table 9.4 Process Scheduling Example

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Source: Pearson

Comparison of Scheduling Policies



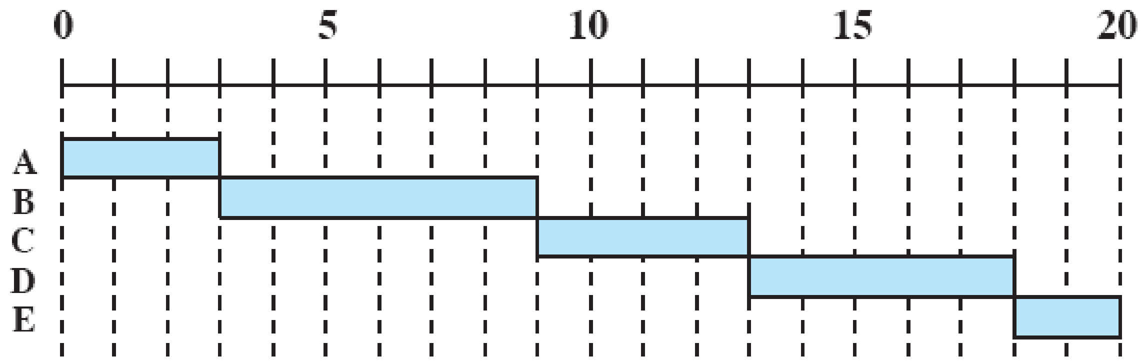
Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71
RR $q = 4$						
Finish Time	3	17	11	20	19	
Turnaround Time (T_r)	3	15	7	14	11	10.00
T_r/T_s	1.00	2.5	1.75	2.80	5.50	2.71
SPN						
Finish Time	3	9	15	20	11	
Turnaround Time (T_r)	3	7	11	14	3	7.60
T_r/T_s	1.00	1.17	2.75	2.80	1.50	1.84
SRT						
Finish Time	3	15	8	20	10	
Turnaround Time (T_r)	3	13	4	14	2	7.20
T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59
HRRN						
Finish Time	3	9	13	20	15	
Turnaround Time (T_r)	3	7	9	14	7	8.00
T_r/T_s	1.00	1.17	2.25	2.80	3.5	2.14
FB $q = 1$						
Finish Time	4	20	16	19	11	
Turnaround Time (T_r)	4	18	12	13	3	10.00
T_r/T_s	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2i$						
Finish Time	4	17	18	20	14	

Source: Pearson

First-Come First-Served (FCFS)

- Simplest scheduling policy
- Also known as first-in-first-out (FIFO)
- When the current process ceases to execute, the oldest process in the Ready queue is selected
- Performs much better for long processes than short ones
  - Whenever a short process arrives just after a long one, it waits too long
- Tends to favor CPU-bound processes over IO-bound ones
  - When a CPU-bound process is running, all the IO-bound processes must wait
- May result in inefficient use of both CPU and IO devices

First-Come-First Served (FCFS)

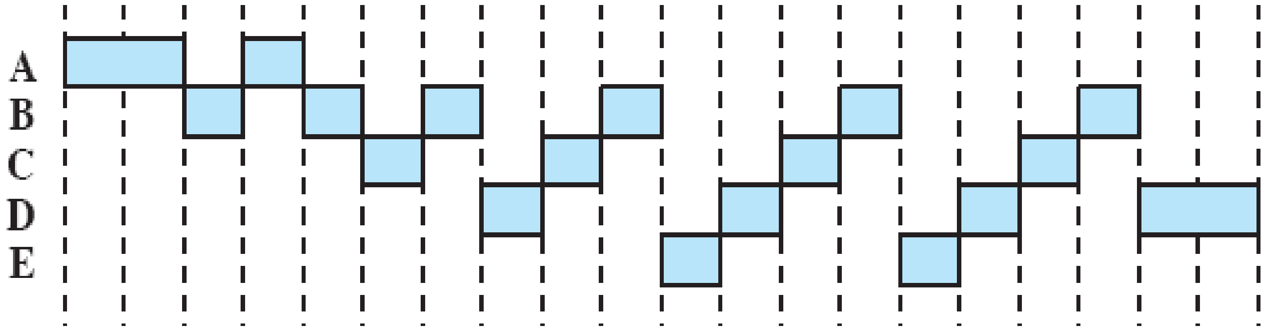


Source: Pearson

Round Robin

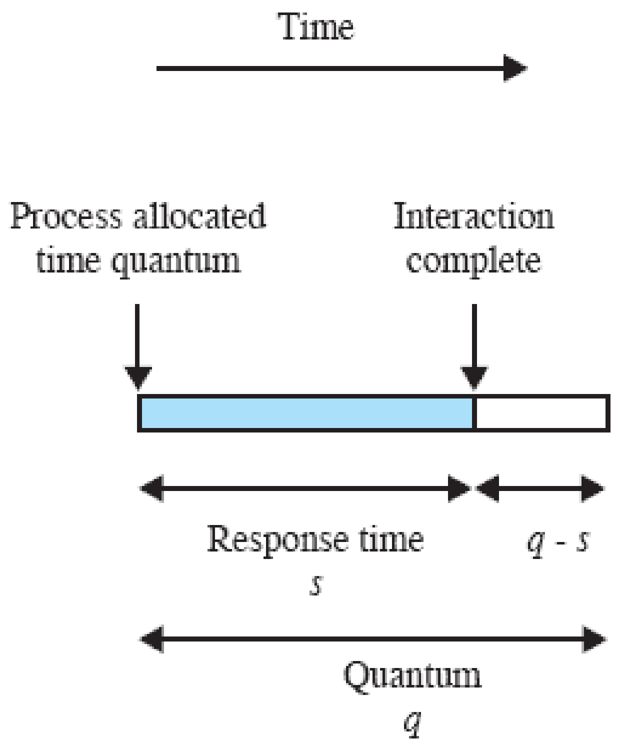
- **Uses preemption based on a clock interrupt**
- **Also known as** *time slicing* **because each process is given a slice of time before being preempted**
- **Principal design issue is the length of the time quantum, or slice, to be used**
  - Time quantum should be slightly longer than a typical interaction
- **Particularly effective in a general-purpose time-sharing system or transaction processing system**
- **One drawback: CPU-bound processes receive a complete quantum while IO-bound ones may not**

Round-Robin (RR), $q = 1$



Source: Pearson

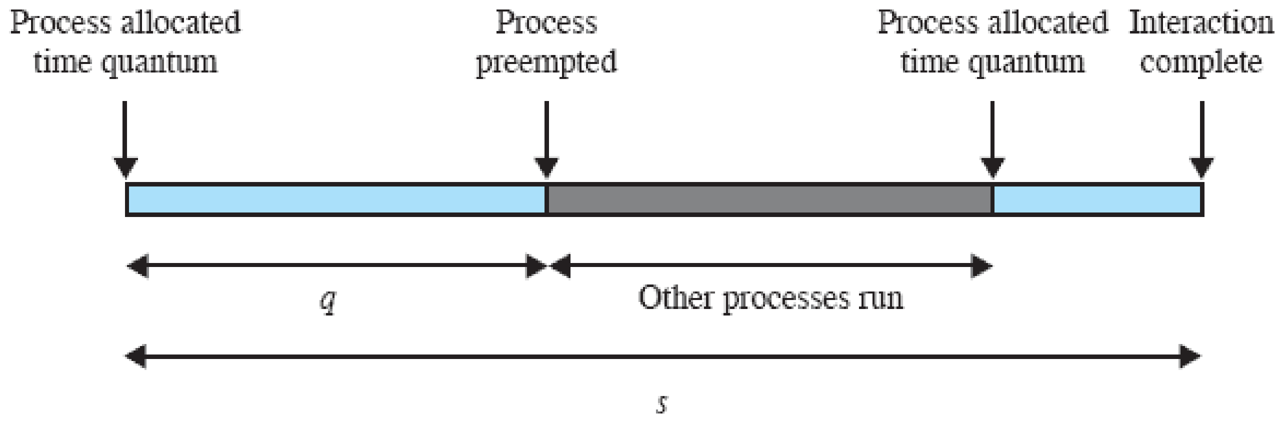
Effect of Preemption Time Quantum Size



(a) Time quantum greater than typical interaction

Source: Pearson

Effect of Preemption


Process allocated time quantum
Process preempted
Process allocated time quantum
Interaction complete
q
Other processes run
s

(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum

Source: Pearson

Virtual Round Robin (VRR)

- Avoid the unfairness of IO-bound processes
- When an IO-bound process is released from IO block, it is moved to the auxiliary queue
- Processes in the auxiliary queue get preference over those in the ready queue

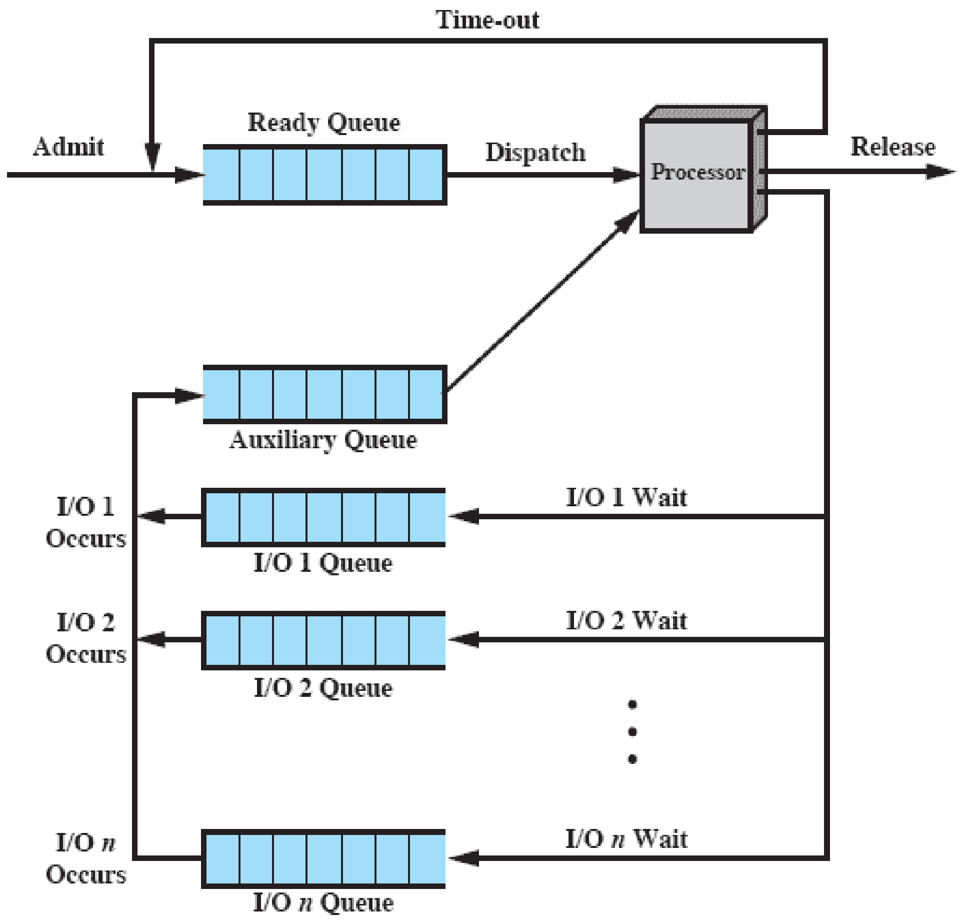
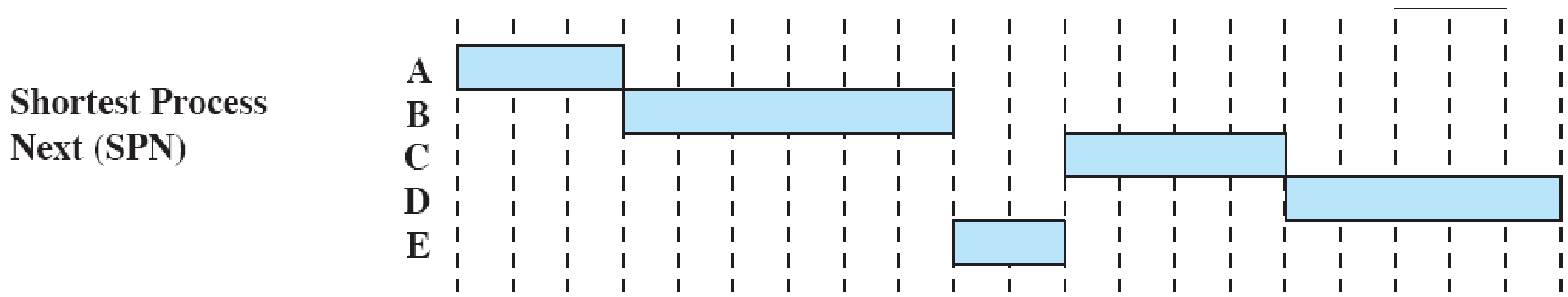


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

Shortest Process Next (SPN)

- **Also called SJF (Shortest Job First)**
  - Nonpreemptive policy in which the process with the shortest expected processing time is selected next
  - A short process will jump to the head of the queue
  - Possibility of starvation for longer processes
  - One difficulty is the need to know, or at least estimate, the required processing time of each process
  - If the programmer's estimate is substantially under the actual running time, the system may abort the job



- **Exponential averaging**
  - For interactive processes, OS may compute a running average of each process run

Source: Pearson

Exponential Smoothing Coefficients

Simple Averaging: $S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i \quad = \frac{1}{n} T_n + \frac{n-1}{n} S_n$

Exponential Averaging: $S_{n+1} = \alpha T_n + (1\text{-}\alpha) S_n$

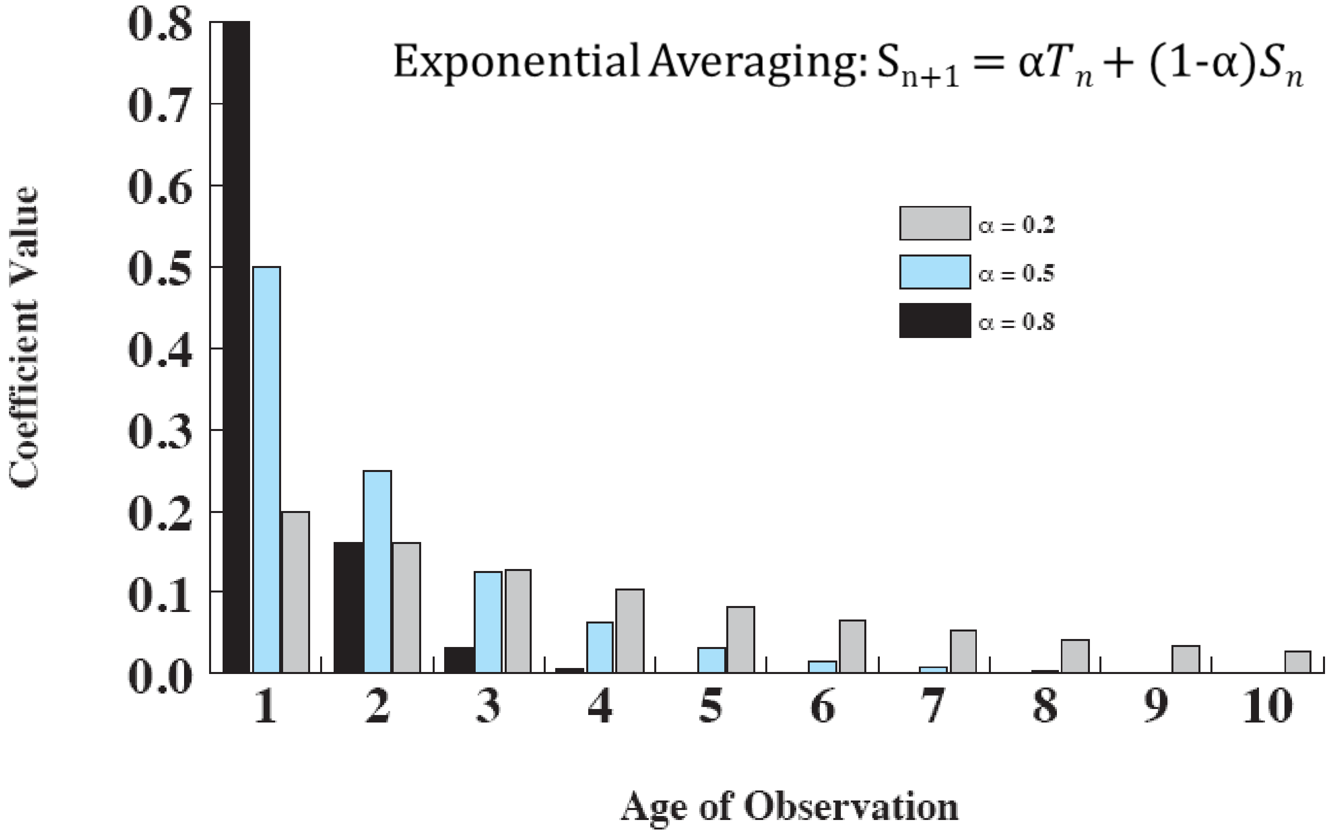
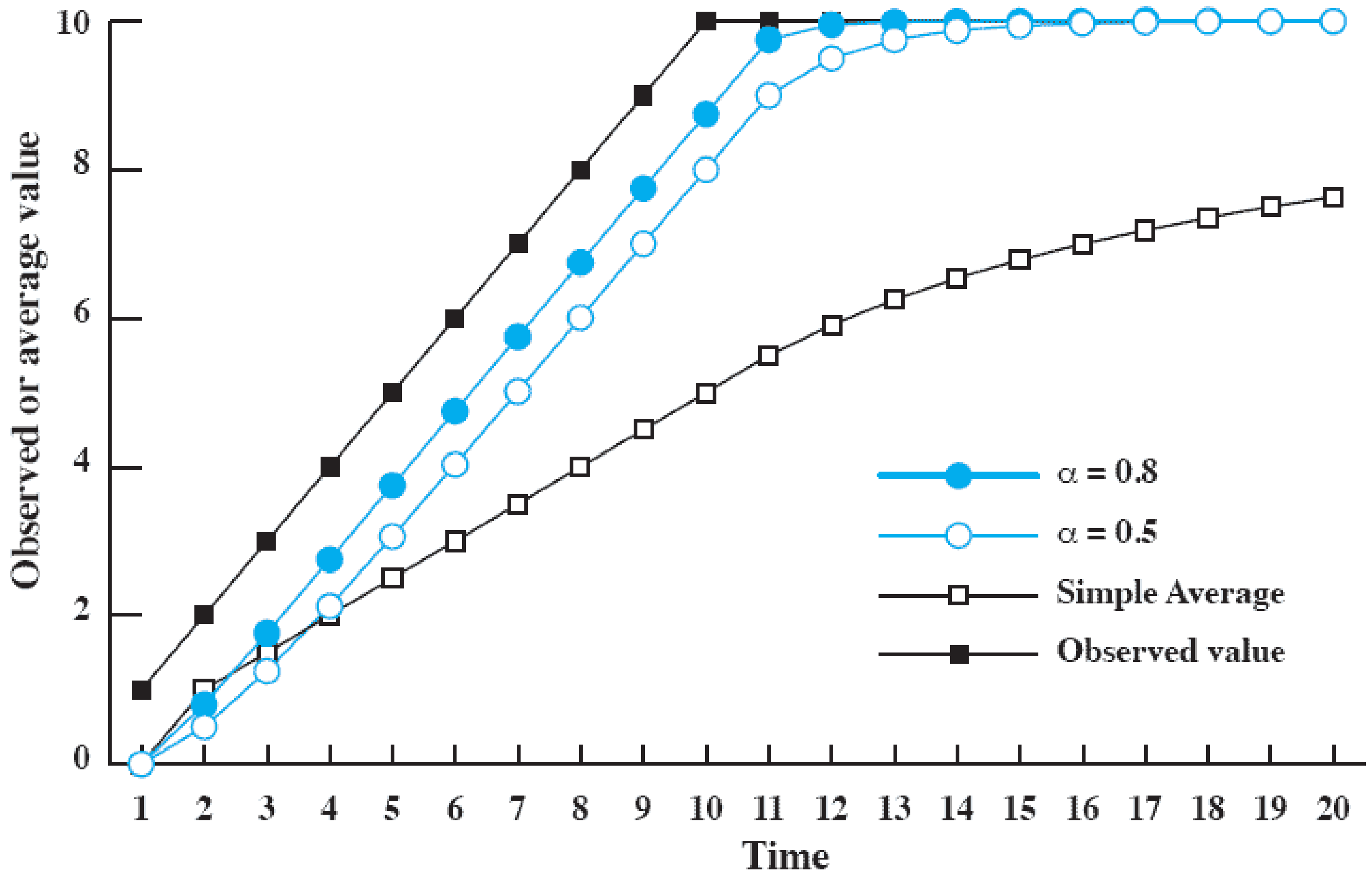


Figure 9.8 Exponential Smoothing Coefficients

Source: Pearson

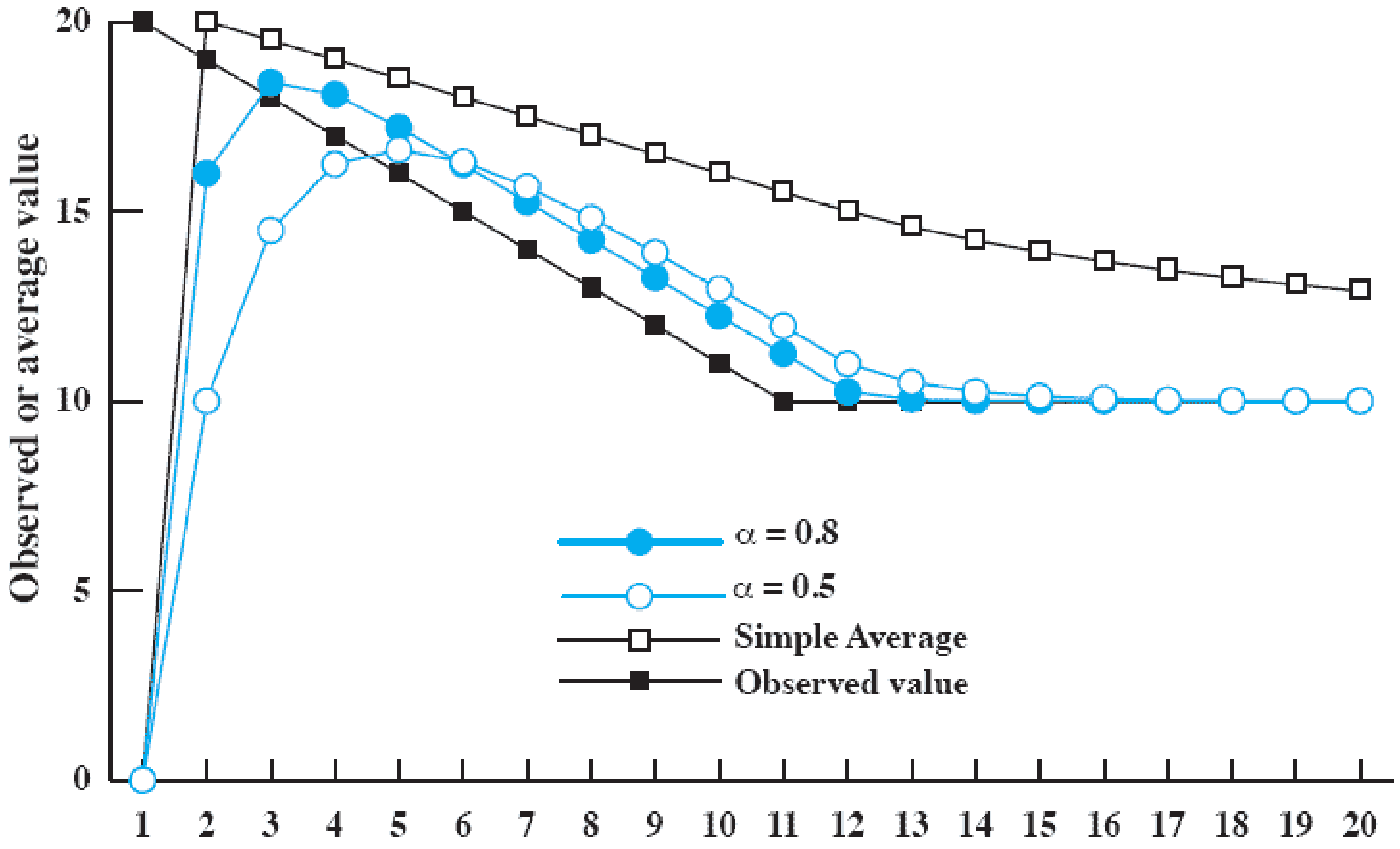
Use Of Exponential Averaging



(a) Increasing function

Source: Pearson

Use Of Exponential Averaging



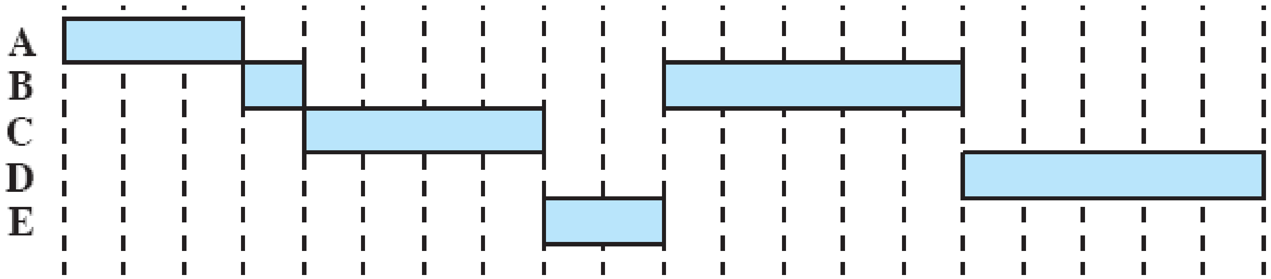
(b) Decreasing function

Source: Pearson

Shortest Remaining Time (SRT)

- **Preemptive version of SPN**
  - Scheduler always chooses the process that has the shortest expected remaining time
    - The scheduler may preempt the current process when a new process with a short expected processing time becomes ready
  - Risk of starvation for longer processes
  - Should give superior turnaround time performance to SPN because a short job is given immediate preference to a running longer job

Shortest Remaining Time (SRT)



Source: Pearson

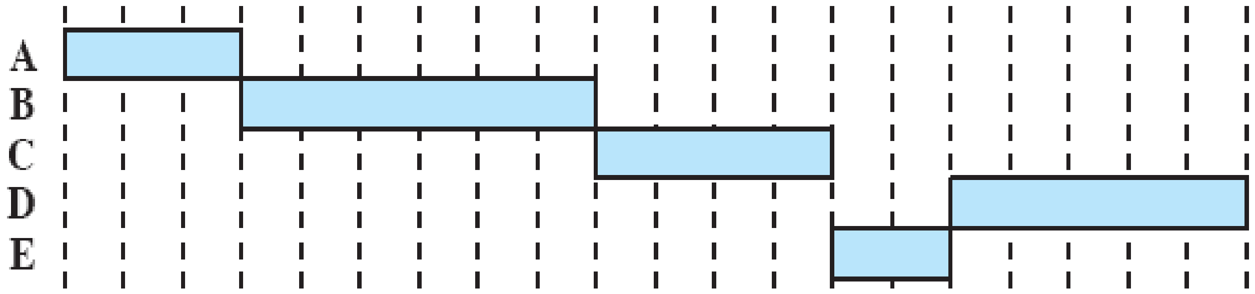
Highest Response Ratio Next (HRRN)

- When the current process completes or is blocked (non-preemptive policy), choose the next process with the greatest ratio (normalized turnaround time)

$$Ratio = \frac{time\ spent\ waiting + expected\ service\ time}{expected\ service\ time}$$

- Attractive because it considers the aging of a process
- While shorter jobs are favored, aging without service increases the ratio so that a long process will eventually win the competition against shorter jobs

Highest Response Ratio Next (HRRN)



Source: Pearson

Feedback Scheduling

- Also known as ***multilevel feedback queue***
  - Penalize jobs that have been running longer by placing these jobs into lower priority queues
  - When a process enters the system, it is placed in RQ0. After its first preemption, it is demoted to RQ1.
  - Short processes will complete quickly while long processes may starve
  - To avoid starvation and long turnaround time, RQi may be assigned 2^i time units

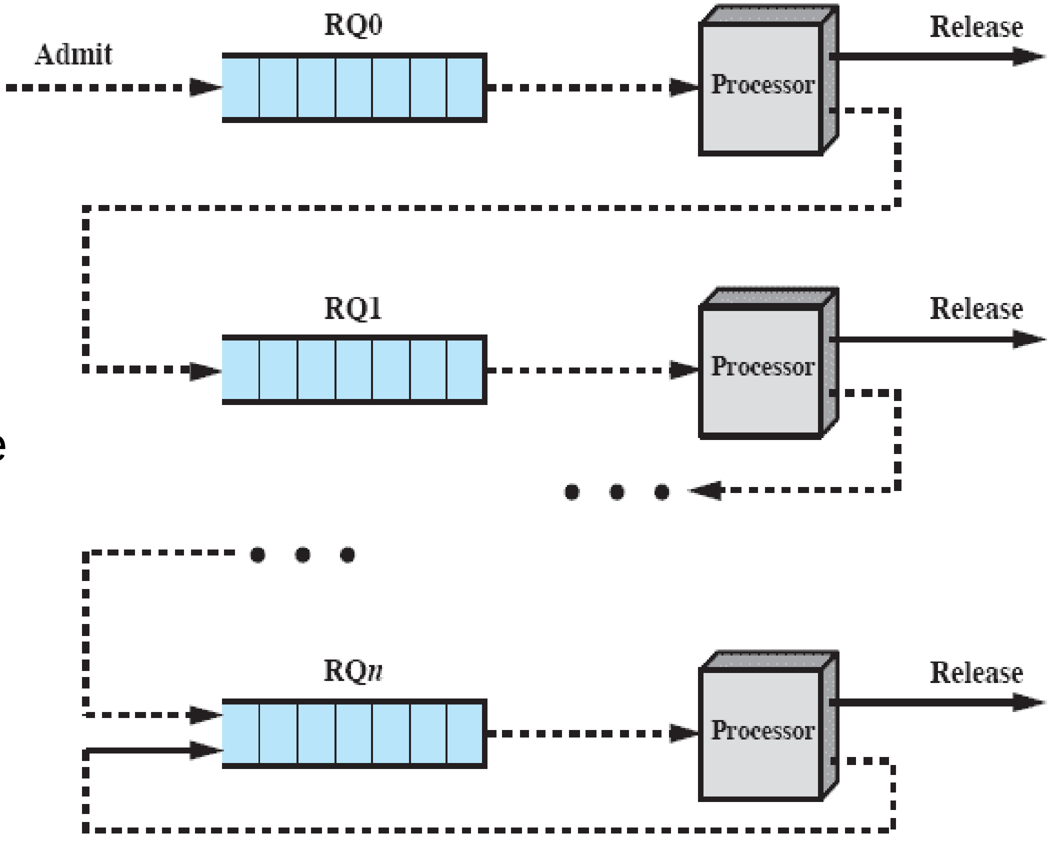
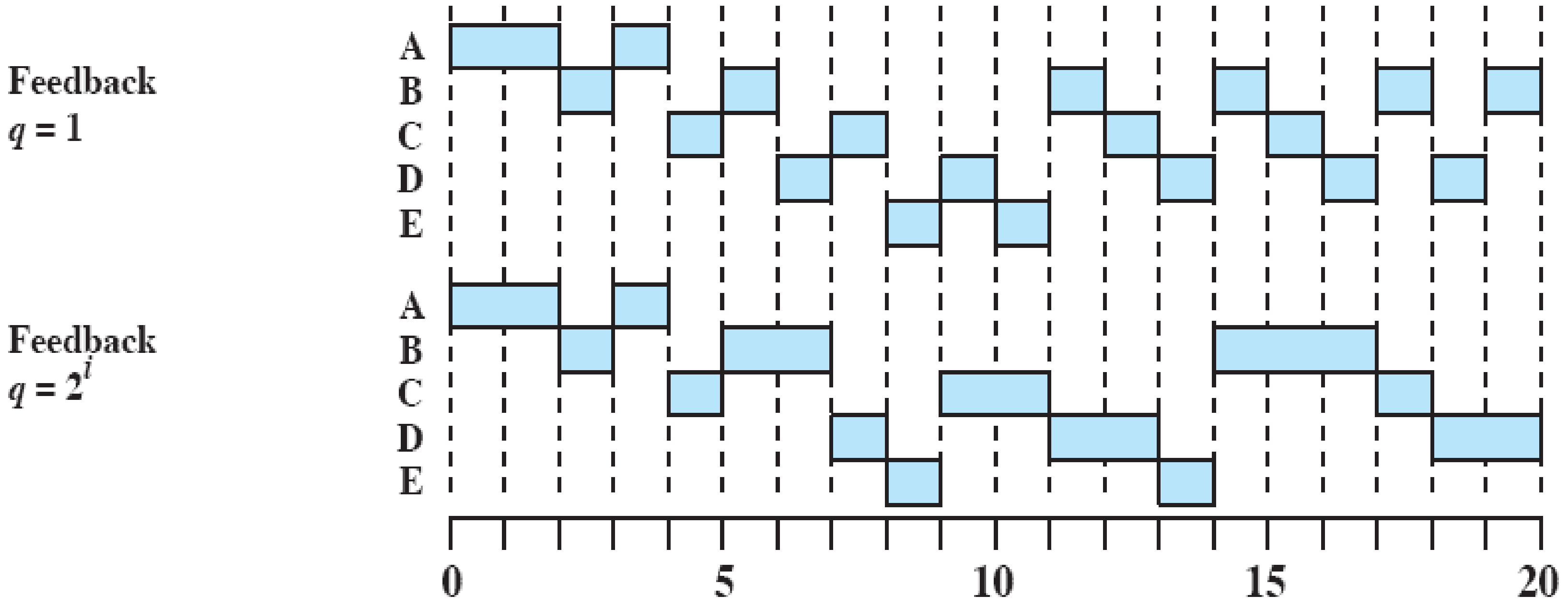


Figure 9.10 Feedback Scheduling

Source: Pearson

Feedback Performance



Source: Pearson

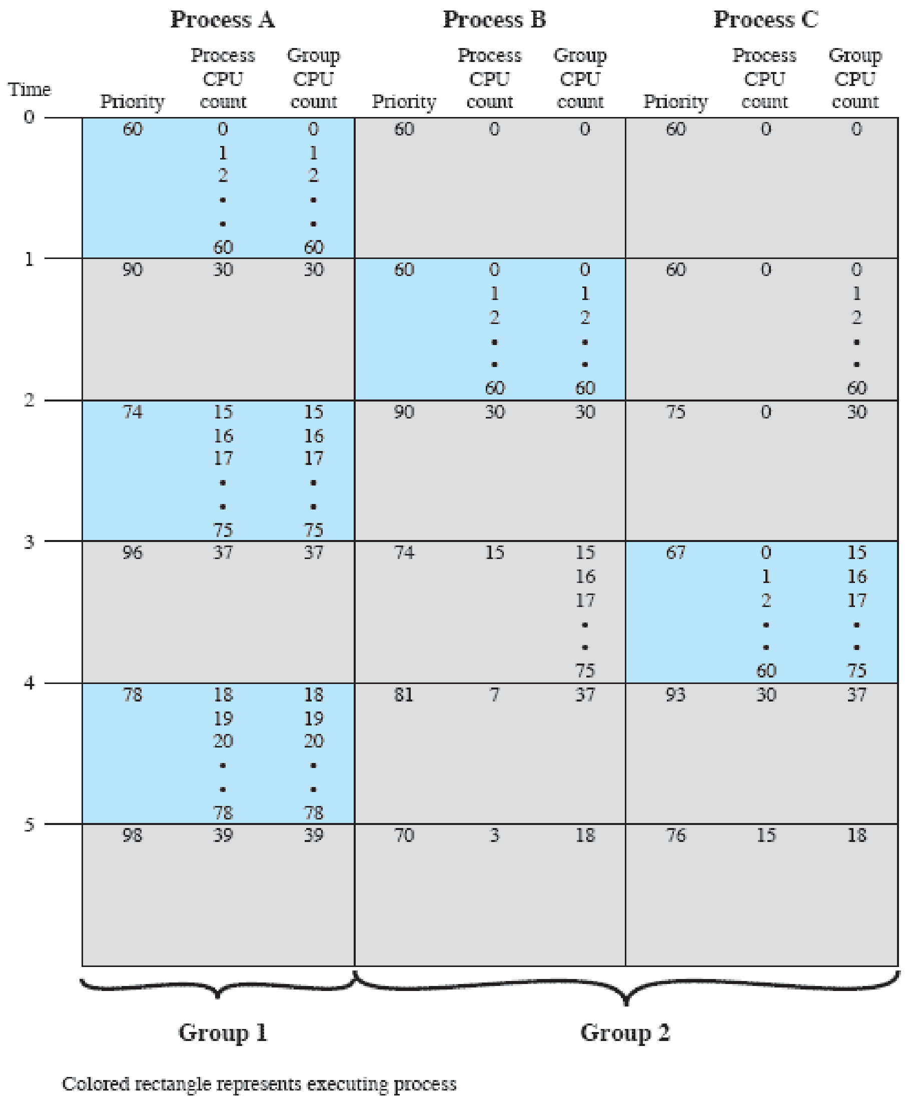
Fair-Share Scheduling

- **Scheduling decisions are made based on sets of processes rather than each individual process**
  - An individual application may be organized as multiple processes/threads
  - From the user perspective, the concern is not how a particular process performs but rather how his/her application (set of processes) performs
- **Each user is assigned a share of the processor**
- **Objective is to monitor usage to give fewer resources to users who have had more than their fair share and more to those who have had less than their fair share**
- **Scheduling is done on the basis of priority, which considers the process priority, its recent processor usage, and the recent processor usage of the group it belongs.**

Fair-Share Scheduler

- $CPU_j(i) = CPU_j(i-1)/2$
  - CPU utilization by process j during interval i
- $GCPU_k(i) = GCPU_k(i-1)/2$
  - CPU utilization of group k during interval i
- $P_j(i) = Base_i + CPU_j(i)/2 + GCPU_k(i)/(4 * W_k)$
  - $Base_i$: base priority of process j
  - W_k : weight assigned to group k

Source: Pearson



	Process A			Process B			Process C		
Time	Priority	Process CPU count	Group CPU count	Priority	Process CPU count	Group CPU count	Priority	Process CPU count	Group CPU count
0	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0	60	0	0
1	90	30	30	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0 1 2 • • 60
2	74	15 16 17 • • 75	15 16 17 • • 75	90	30	30	75	0	30
3	96	37	37	74	15	15 16 17 • • 75	67	0 1 2 • • 60	15 16 17 • • 75
4	78	18 19 20 • • 78	18 19 20 • • 78	81	7	37	93	30	37
5	98	39	39	70	3	18	76	15	18

Group 1 Group 2

Colored rectangle represents executing process

Figure 9.16 Example of Fair Share Scheduler—Three Processes, Two Groups

Traditional Unix Scheduling

- **Used in both SVR3 and 4.3 BSD UNIX**
  - These systems are primarily targeted at the time-sharing interactive environment
- **Designed to provide good response time for interactive users while ensuring that low-priority background jobs do not starve**
  - Employs multilevel feedback queue using round robin within each of the priority queues
  - If a running process does not block or complete within one second, it is preempted.
  - Priority is based on process type and execution history

Scheduling Formula

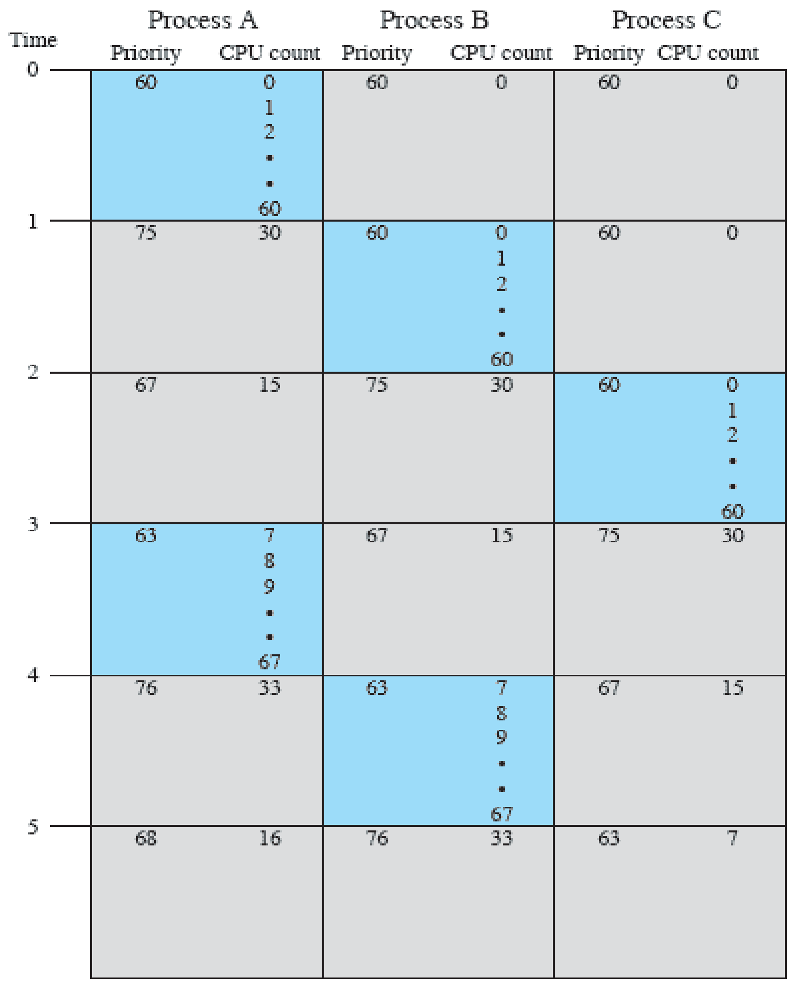
$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + nice_j$$

where

$CPU_j(i)$ =	measure of processor utilization by process j through interval i
$P_j(i)$ =	priority of process j at beginning of interval i; lower values equal higher priorities
$Base_j$ =	base priority of process j
$nice_j$ =	user-controllable adjustment factor

Examples of Traditional UNIX Process Scheduling



Time	Process A		Process B		Process C	
	Priority	CPU count	Priority	CPU count	Priority	CPU count
0	60	0 1 2 • • 60	60	0	60	0
1	75	30	60	0 1 2 • • 60	60	0
2	67	15	75	30	60	0 1 2 • • 60
3	63	7 8 9 • • 67	67	15	75	30
4	76	33	63	7 8 9 • • 67	67	15
5	68	16	76	33	63	7

Colored rectangle represents executing process

Source: Pearson

Figure 9.17 Example of Traditional UNIX Process Scheduling

Homework 8

- Exercise 9.1
- Exercise 9.2
- Exercise 9.3
- Exercise 9.7
- Exercise 9.9