

*Operating  
Systems:  
Internals  
and  
Design  
Principles*

# Chapter 9 Uniprocessor Scheduling

Eighth Edition  
By William Stallings

## Table 9.1

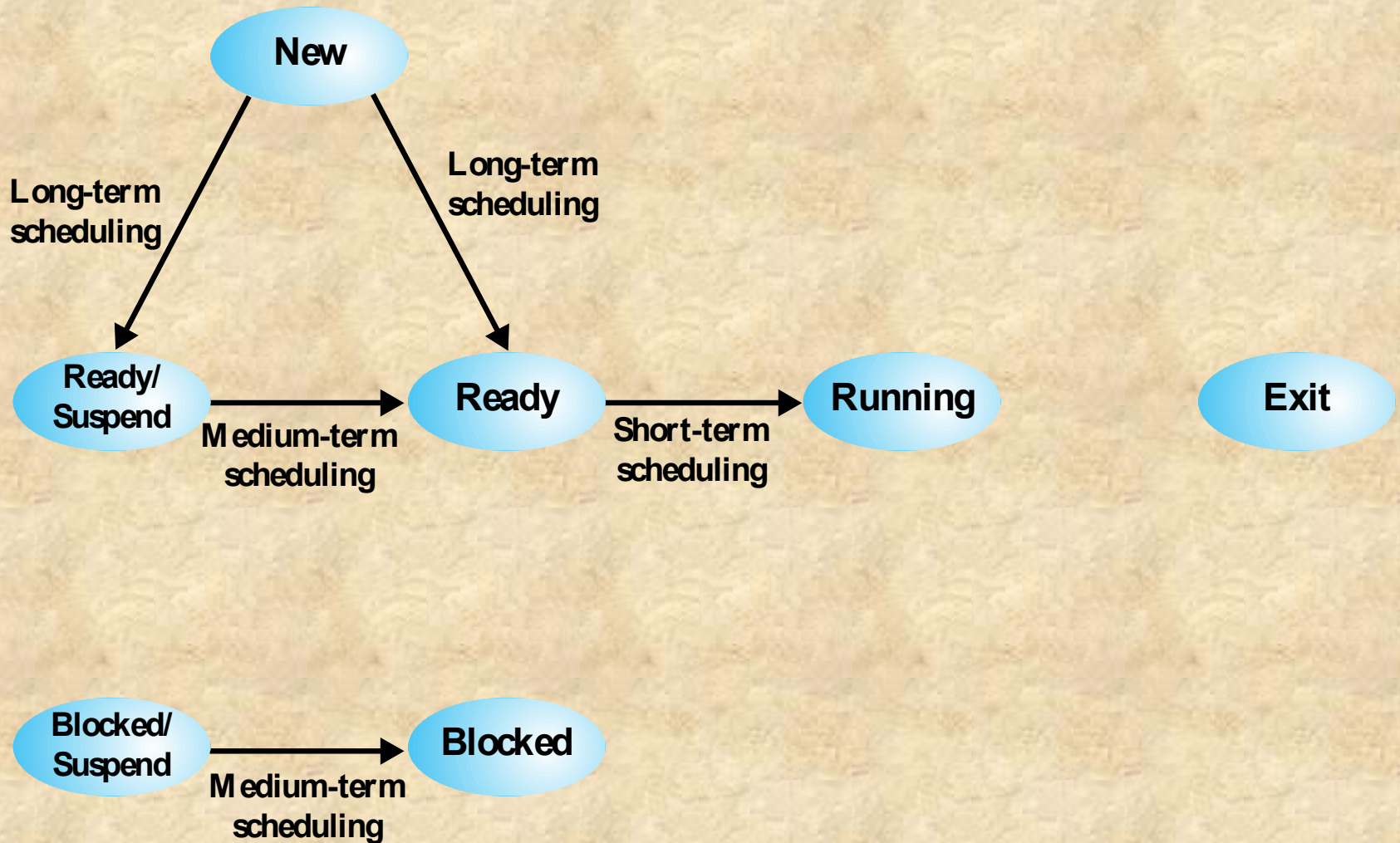
# Types of Scheduling

<b>Long- t e r m schedul i n g</b>	The decision to add to the pool of processes to be executed
<b>M e d i u m t e r m schedul i n g</b>	The decision to add to the number of processes that are partially or fully in main memory
<b>Short - t e r m schedul i n g</b>	The decision as to which available process will be executed by the processor
<b>I / O schedul i n g</b>	The decision as to which process's pending I/O request shall be handled by an available I/O device

# Processor Scheduling

- Aim is to assign processes to be executed by the processor in a way that meets system objectives, such as response time, throughput, and processor efficiency
- Broken down into three separate functions:





**Figure 9.1 Scheduling and Process State Transitions**



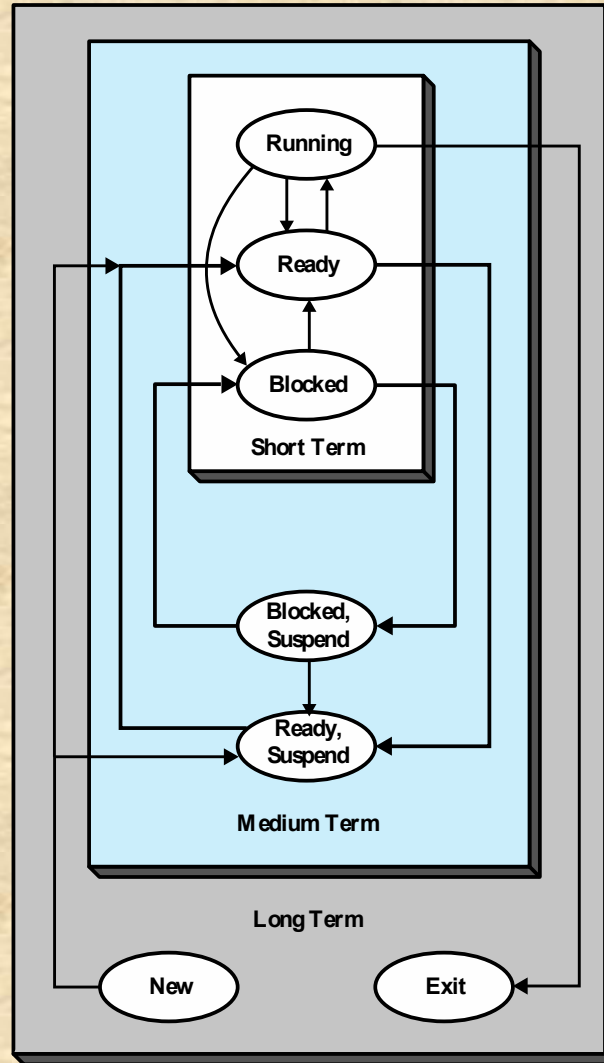
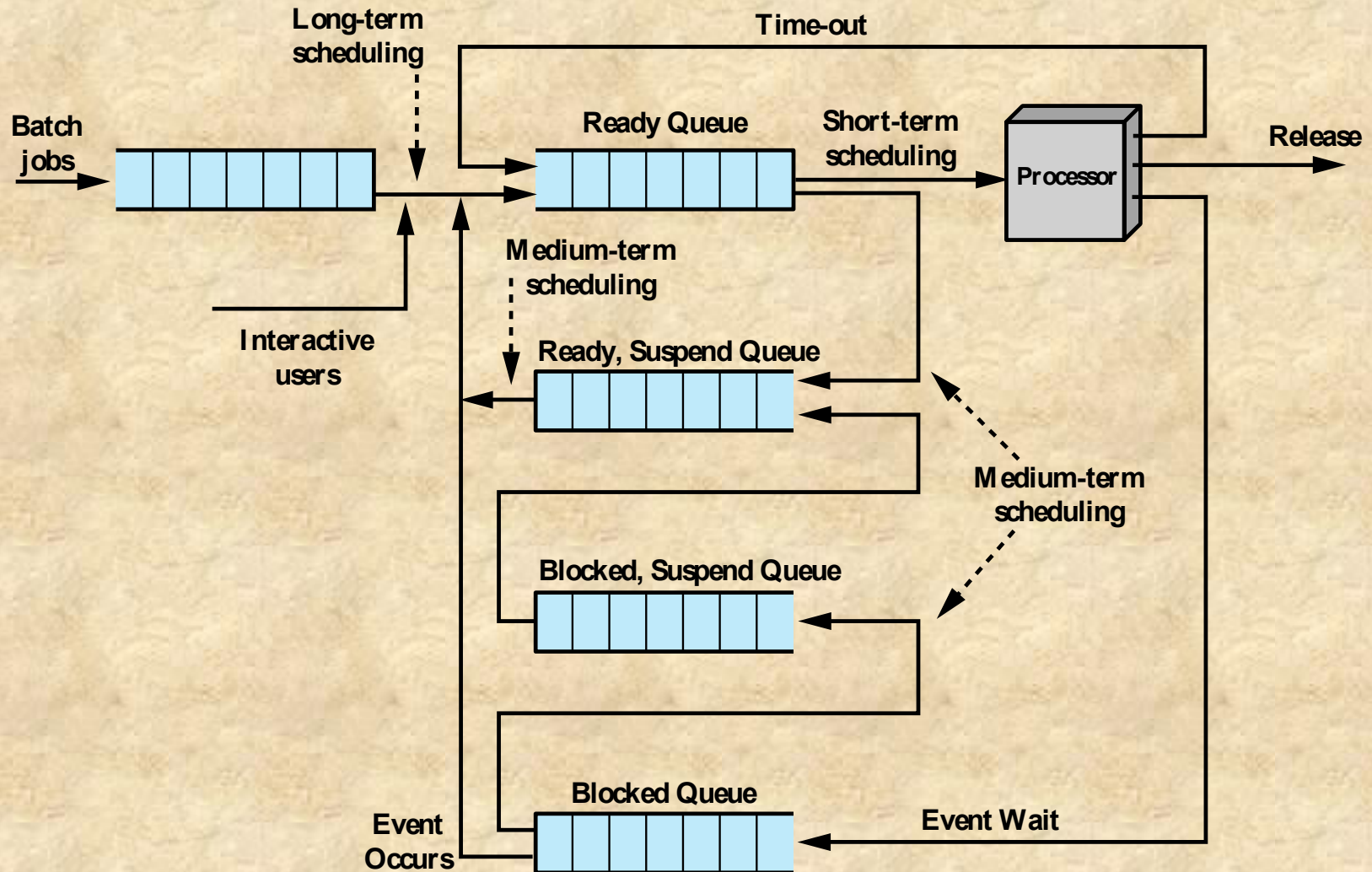


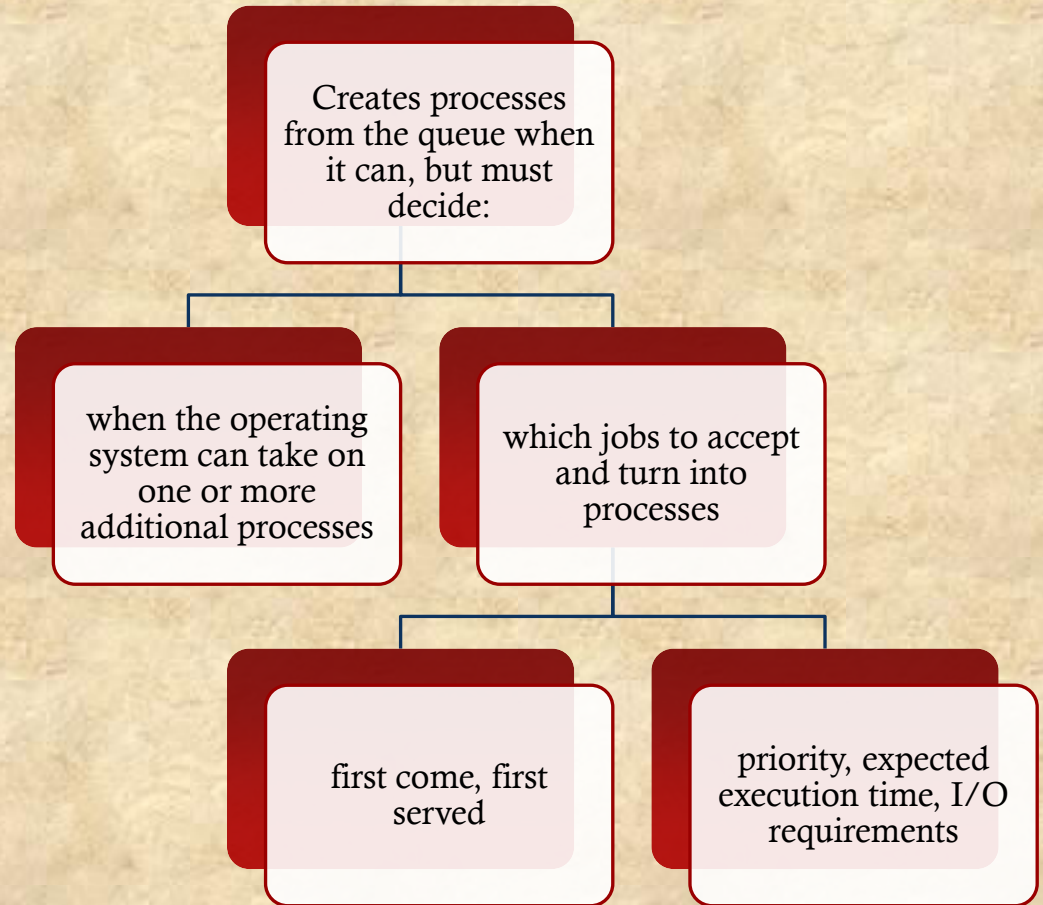
Figure 9.2 Levels of Scheduling



**Figure 9.3 Queuing Diagram for Scheduling**

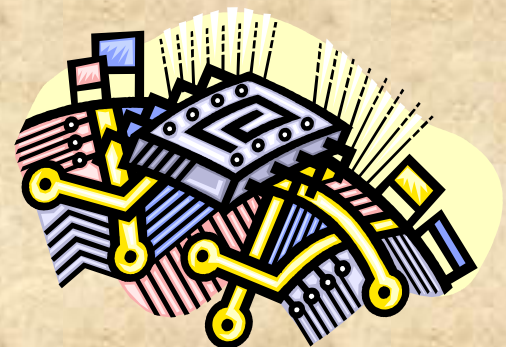
# Long-Term Scheduler

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
  - the more processes that are created, the smaller the percentage of time that each process can be executed
  - may limit to provide satisfactory service to the current set of processes



# Medium-Term Scheduling

- Part of the swapping function
- Swapping-in decisions are based on the need to manage the degree of multiprogramming
  - considers the memory requirements of the swapped-out processes





# Short-Term Scheduling

- Known as the dispatcher
- Executes most frequently
- Makes the fine-grained decision of which process to execute next
- Invoked when an event occurs that may lead to the blocking of the current process or that may provide an opportunity to preempt a currently running process in favor of another

## Examples:

- Clock interrupts
- I/O interrupts
- Operating system calls
- Signals (e.g., semaphores)

# Short Term Scheduling Criteria

- Main objective is to allocate processor time to optimize certain aspects of system behavior
- A set of criteria is needed to evaluate the scheduling policy

## User-oriented criteria

- relate to the behavior 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 in on effective and efficient utilization of the processor (rate at which processes are completed)
- generally of minor importance on single-user systems

# Short-Term Scheduling Criteria: Performance

examples:

- response time
- throughput

Criteria can  
be classified  
into:

example:

- predictability

Performance-related

quantitative

easily  
measured

Non-performance  
related

qualitative

hard to  
measure



## Table 9.2

# 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.

**Balancing resources** The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

(Table can be found on page 403 in textbook)



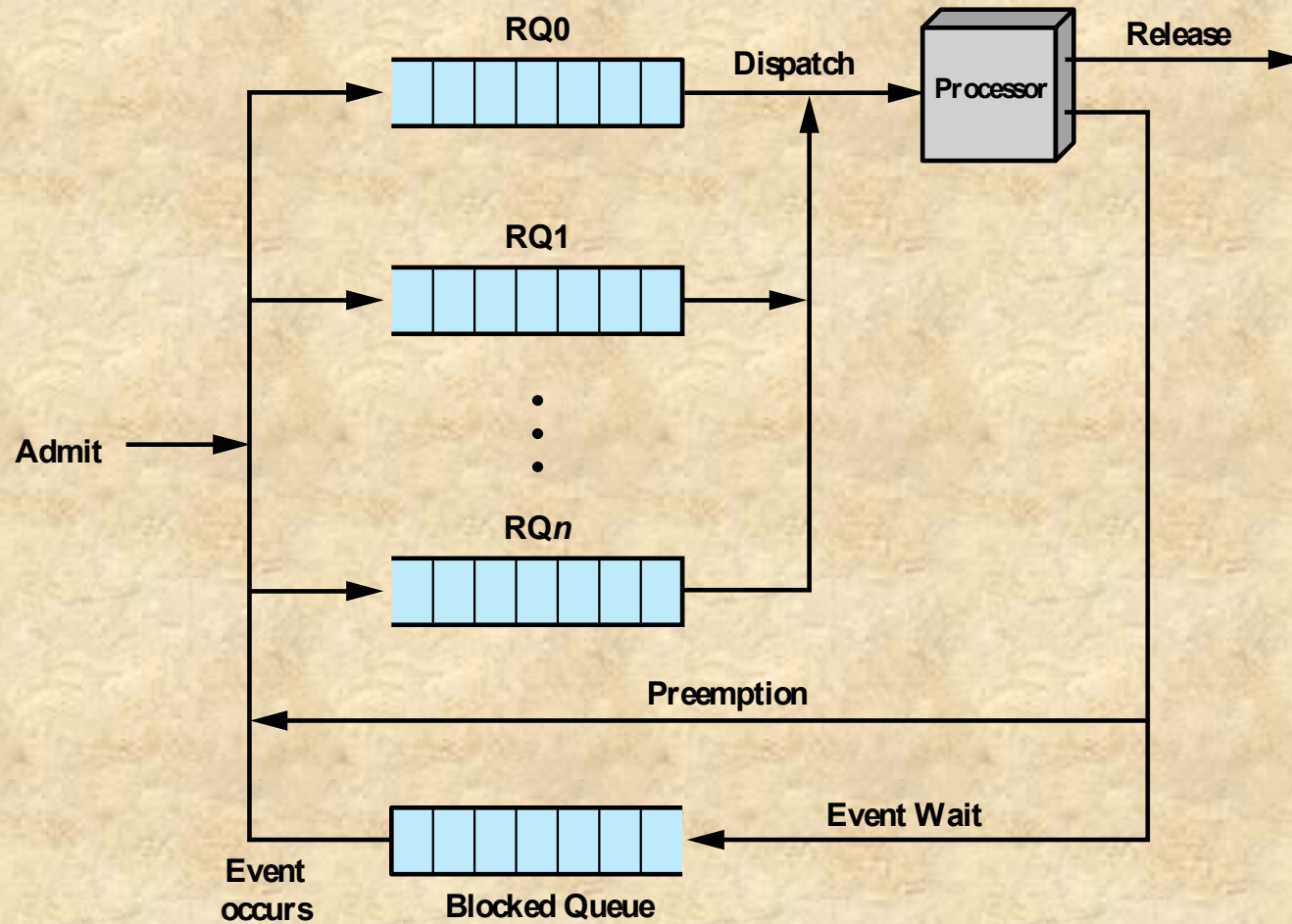


Figure 9.4 Priority Queuing

	FCFS	Round robin	SPN	SRT	HRRN	Feedback
<b>Selection function</b>	$\max [w]$	constant	$\min [s]$	$\min [s - e]$	$\max \frac{w + s}{s}$	(see text)
<b>Decision mode</b>	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive	Preemptive (at time quantum)
<b>Through-Put</b>	Not emphasized	May be low if quantum is too small	High	High	High	Not emphasized
<b>Response time</b>	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
<b>Overhead</b>	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
<b>Effect on processes</b>	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes	Good balance	May favor I/O bound processes
<b>Starvation</b>	No	No	Possible	Possible	No	Possible

**Table 9.3**

Characteristic  
s of Various  
Scheduling  
Policies

(Table can be found  
on page 405 in  
textbook)

# Selection Function

- Determines 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 instants in time at which the selection function is exercised
- Two categories:
  - Nonpreemptive
  - Preemptive





# Nonpreemptive vs Preemptive

## 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 and moved to ready state by the OS
- preemption may occur when new process arrives, on an interrupt, or periodically

---

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

**Table 9.4**  
**Process Scheduling Example**

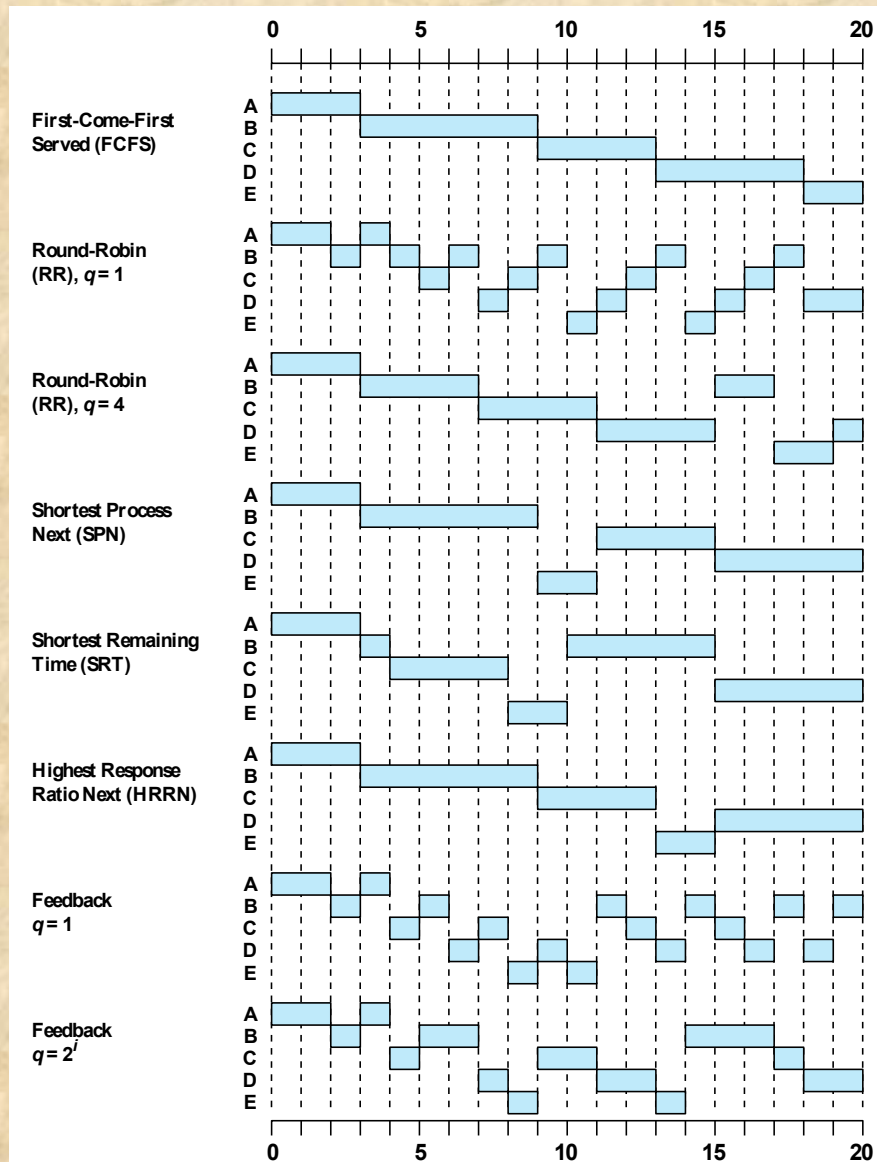


Figure 9.5 A Comparison of Scheduling Policies

# First-Come-First-Served (FCFS)

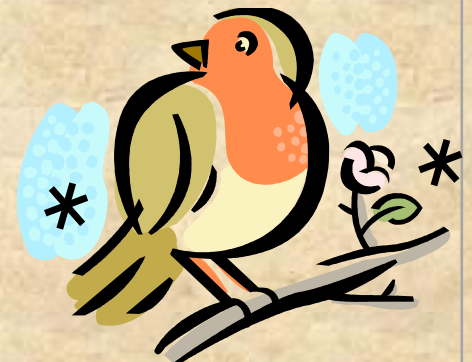
- Simplest scheduling policy
- Also known as first-in-first-out (FIFO) or a strict queuing scheme
- When the current process ceases to execute, the longest process in the Ready queue is selected
- Performs much better for long processes than short ones
- Tends to favor processor-bound processes over I/O-bound processes

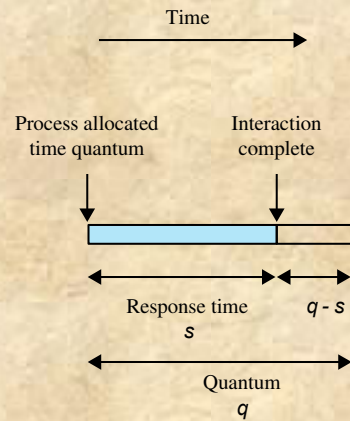




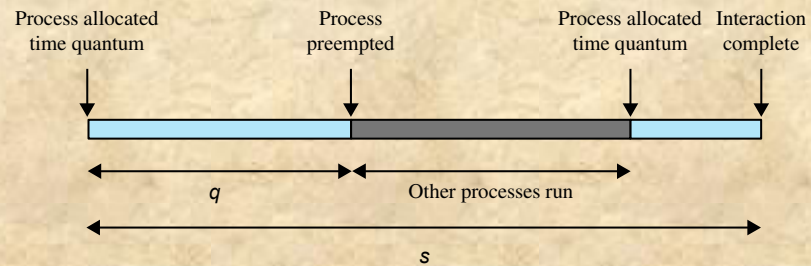
# Round Robin

- Uses preemption based on a clock
- 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
- Particularly effective in a general-purpose time-sharing system or transaction processing system
- One drawback is its relative treatment of processor-bound and I/O-bound processes





(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum

Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time ( $T_S$ )	3	6	4	5	2	Mean
<b>FCFS</b>						
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
<b>RR <math>q=1</math></b>						
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
<b>RR <math>q=4</math></b>						
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
<b>SPN</b>						
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
<b>SRT</b>						
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
<b>HRRN</b>						
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
<b>FB <math>q=1</math></b>						
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
<b>FB <math>q=2i</math></b>						
Finish Time	4	17	18	20	14	
Turnaround Time ( $T_R$ )	4	15	14	14	6	10.60
$T_R/T_S$	1.33	2.50	3.50	2.80	3.00	2.63

**Table 9.5**  
**A Comparison**  
**of Scheduling**  
**Policies**

(Table is on page 408 in textbook)

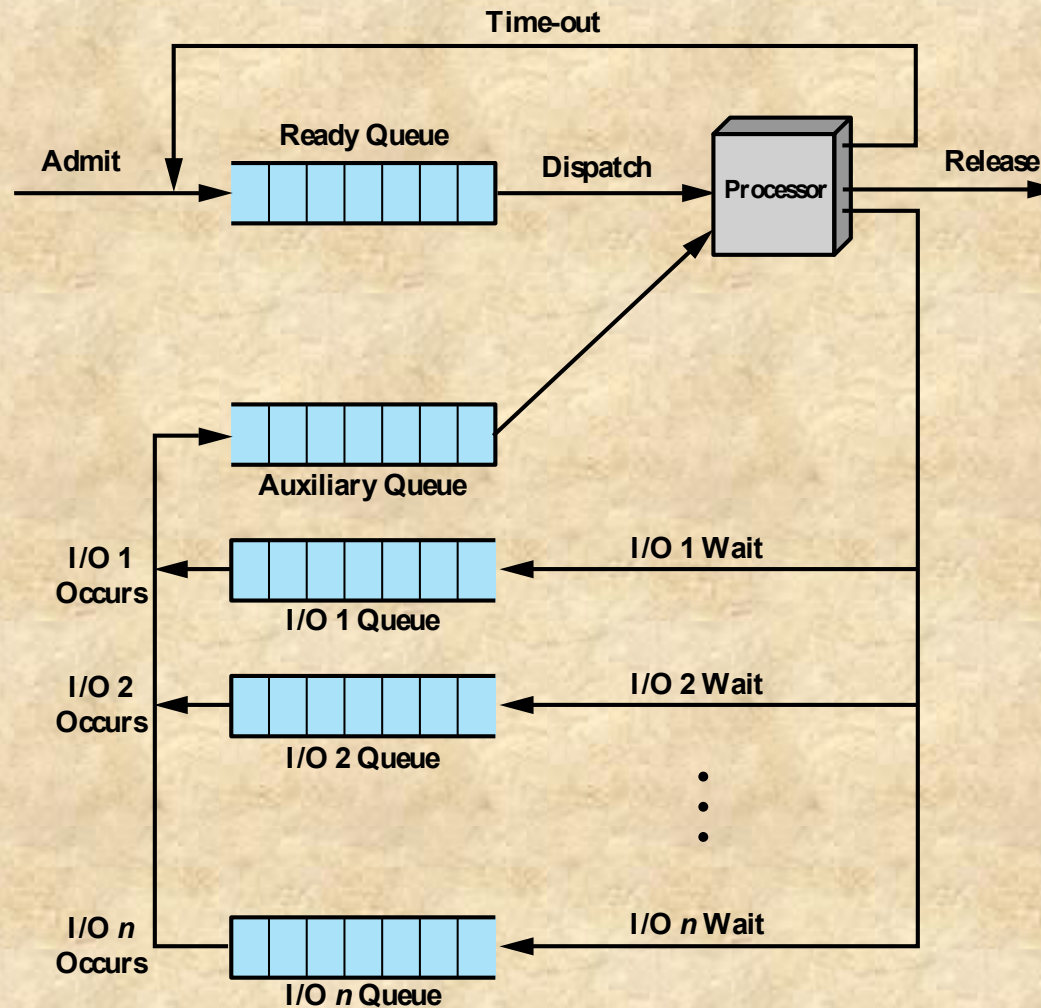
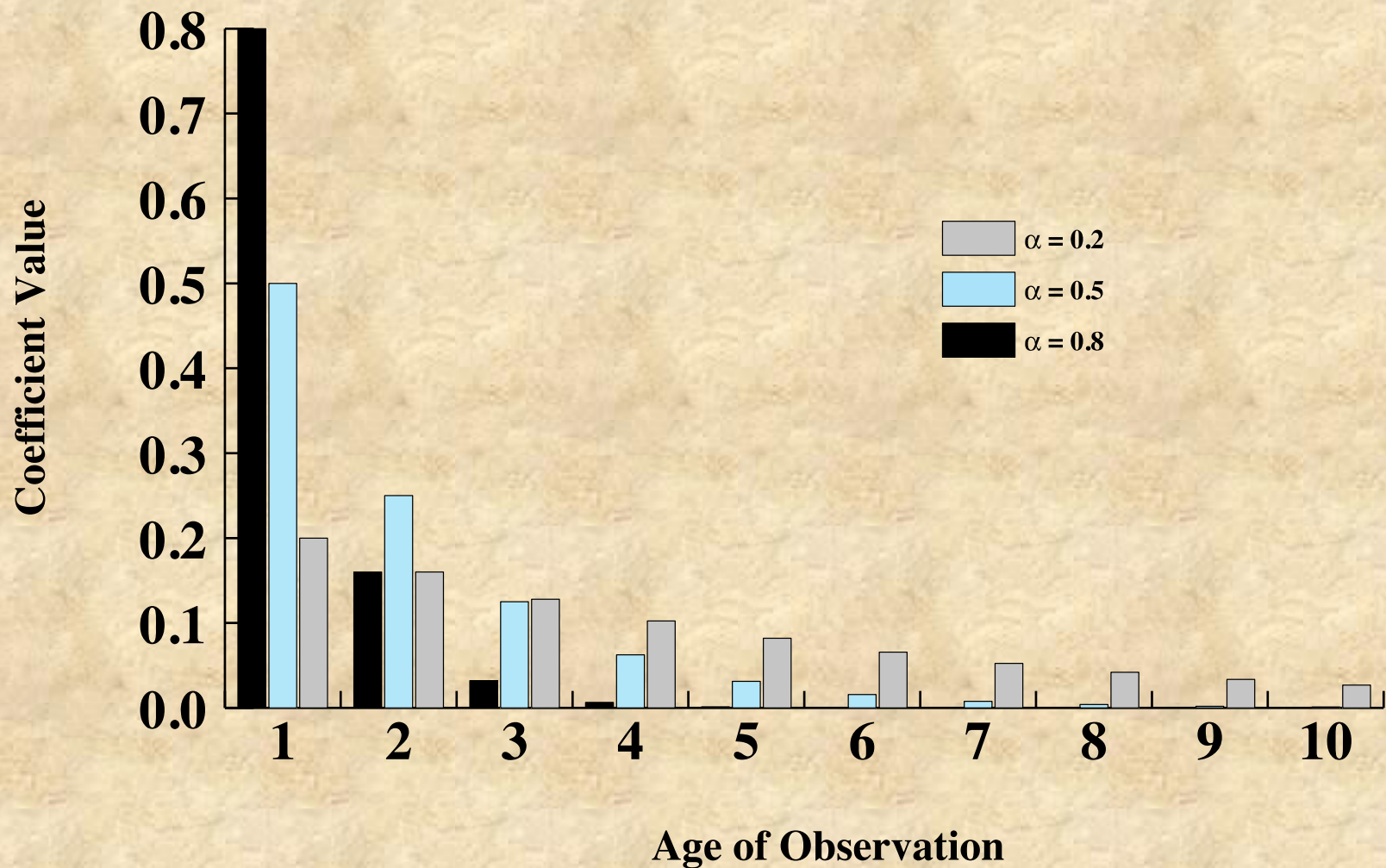


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

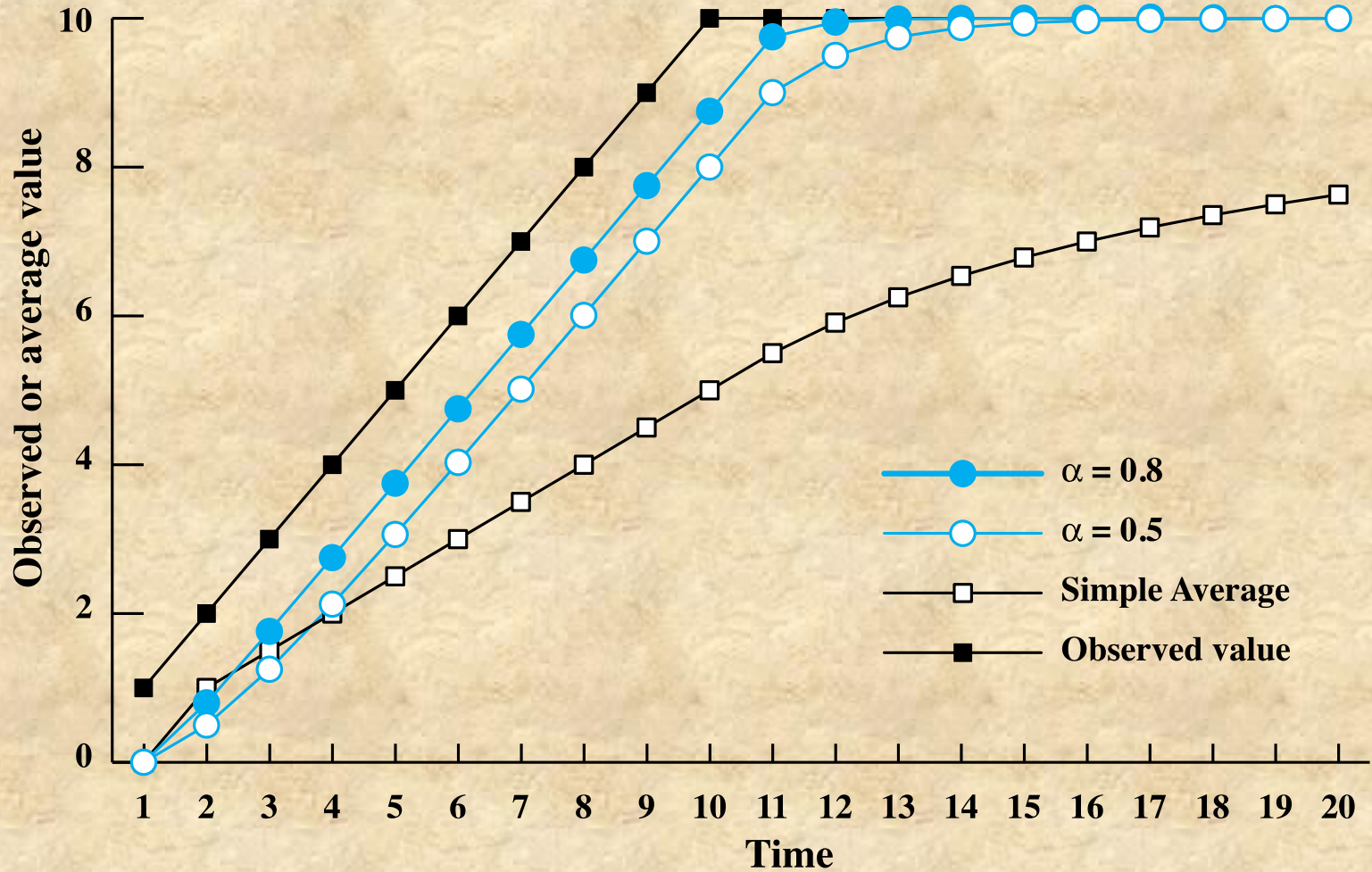


# Shortest Process Next (SPN)

- 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

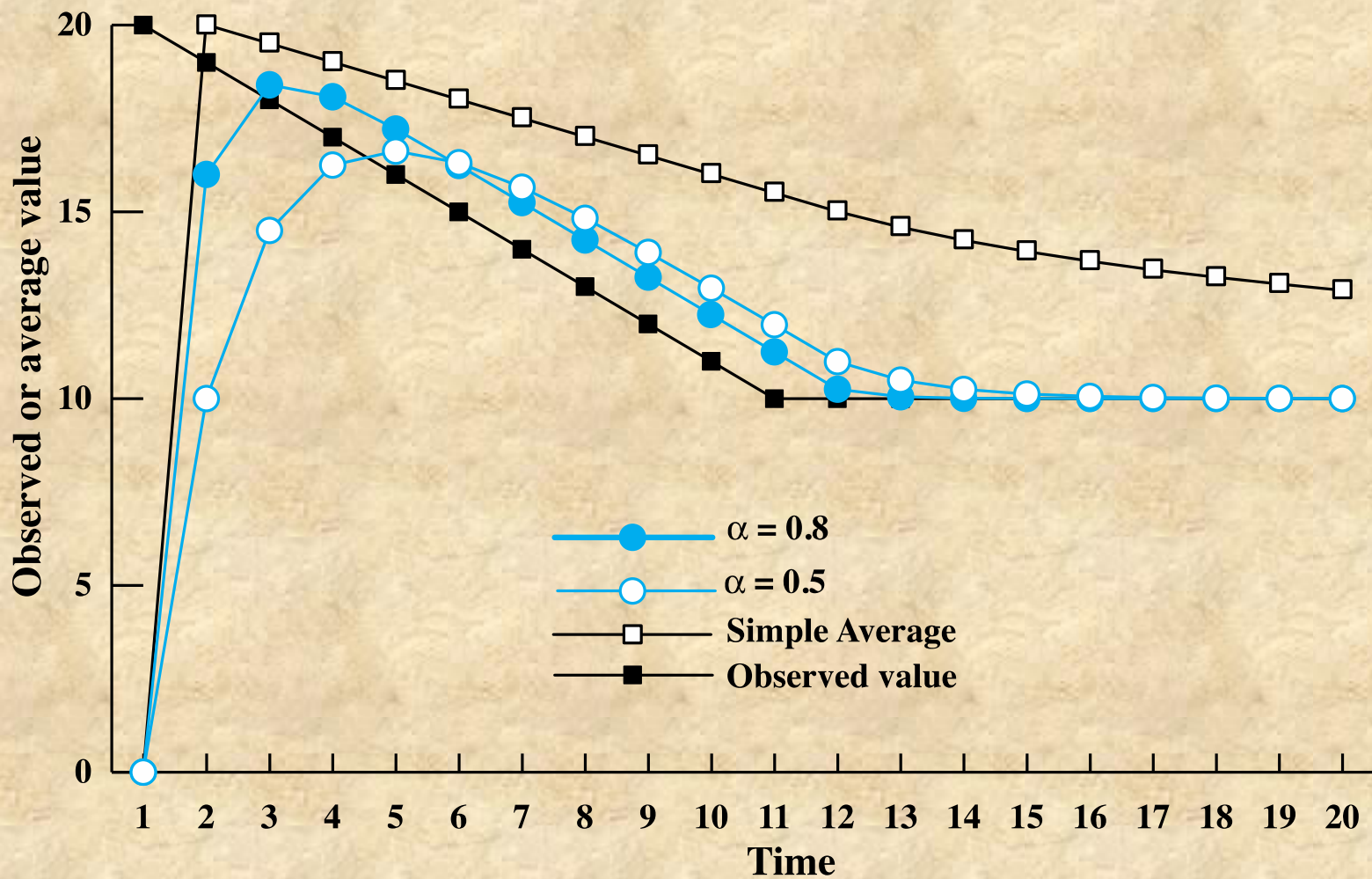


**Figure 9.8 Exponential Smoothing Coefficients**



(a) Increasing function

Figure 9.9 Use of Exponential Averaging



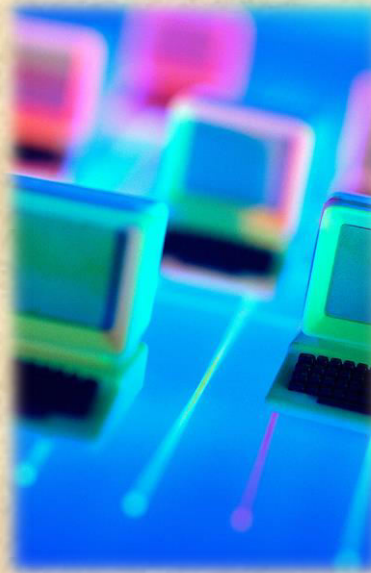
(b) Decreasing function

**Figure 9.9 Use of Exponential Averaging**



# Shortest Remaining Time (SRT)

- Preemptive version of SPN
- Scheduler always chooses the process that has the shortest expected remaining processing time
- Risk of starvation of longer processes
- Should give superior turnaround time performance to SPN because a short job is given immediate preference to a running longer job



# Highest Response Ratio Next (HRRN)

- Chooses next process with the greatest ratio
- Attractive because it accounts for the age of the process
- While shorter jobs are favored, aging without service increases the ratio so that a longer process will eventually get past competing shorter jobs

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

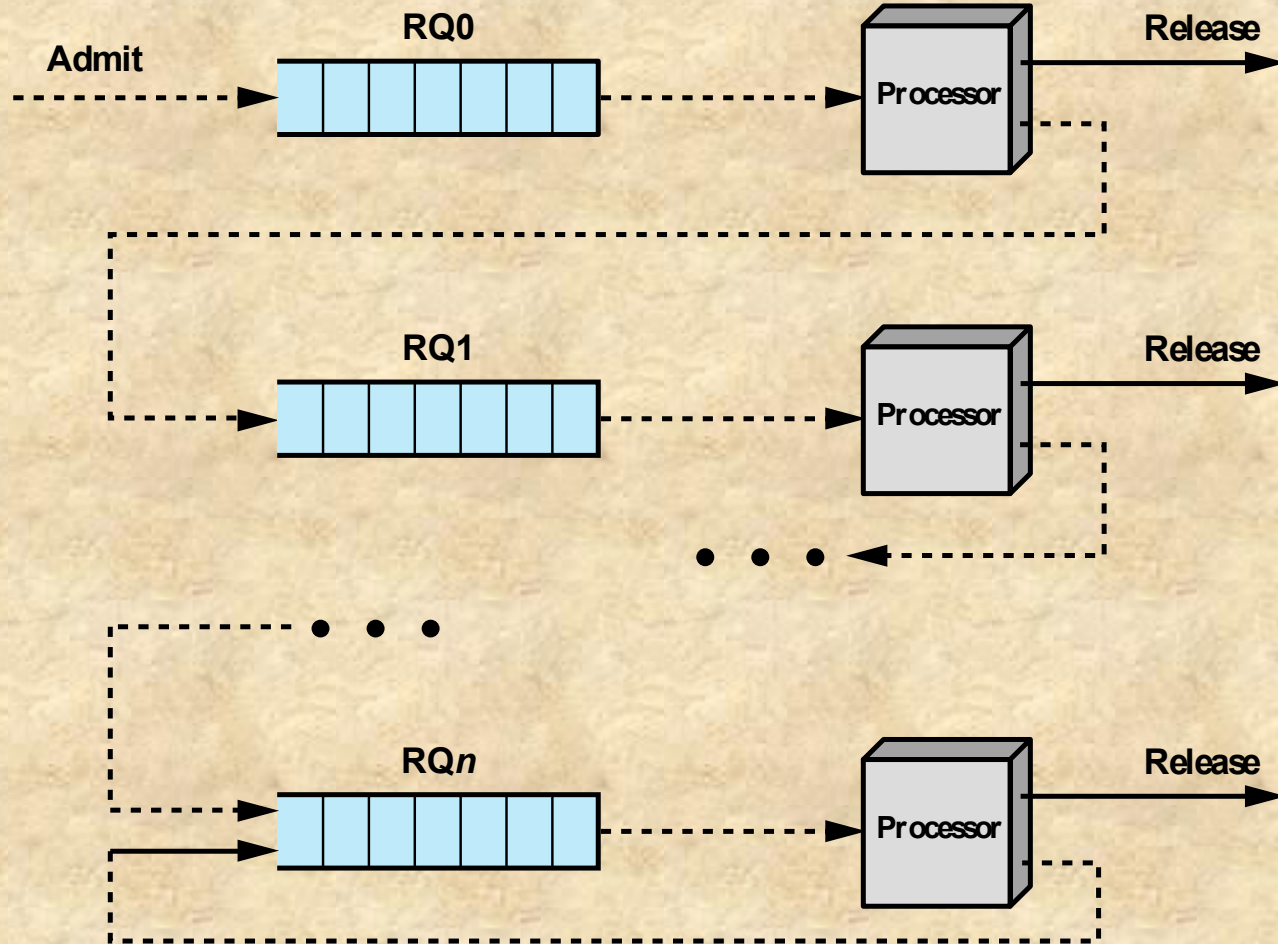


Figure 9.10 Feedback Scheduling



# Performance Comparison

- Any scheduling discipline that chooses the next item to be served independent of service time obeys the relationship:

$$\frac{T_r}{T_s} = \frac{1}{1 - \rho}$$

where

$T_r$  = turnaround time or residence time; total time in system, waiting plus execution

$T_s$  = average service time; average time spent in Running state

$\rho$  = processor utilization



# Table 9.6

## Formulas for Single- Server Queues with Two Priority Categories

- Assumptions:
1. Poisson arrival rate.
  2. Priority 1 items are serviced before priority 2 items.
  3. First-come-first-served dispatching for items of equal priority.
  4. No item is interrupted while being served.
  5. No items leave the queue (lost calls delayed).



### (a) General formulas

$$\begin{aligned}\lambda &= \lambda_1 + \lambda_2 \\ \rho_1 &= \lambda_1 T_{s1}; \quad \rho_2 = \lambda_2 T_{s2} \\ \rho &= \rho_1 + \rho_2 \\ T_s &= \frac{\lambda_1}{\lambda} T_{s1} + \frac{\lambda_2}{\lambda} T_{s2} \\ T_r &= \frac{\lambda_1}{\lambda} T_{r1} + \frac{\lambda_2}{\lambda} T_{r2}\end{aligned}$$

### (b) No interrupts; exponential service times

$$\begin{aligned}T_{r1} &= T_{s1} + \frac{\rho_1 T_{s1} + \rho_2 T_{s2}}{1 - \rho_1} \\ T_{r2} &= T_{s2} + \frac{T_{r1} - T_{s1}}{1 - \rho}\end{aligned}$$

### (c) Preemptive-resume queuing discipline; exponential service times

$$\begin{aligned}T_{r1} &= T_{s1} + \frac{\rho_1 T_{s1}}{1 - \rho_1} \\ T_{r2} &= T_{s2} + \frac{1}{1 - \rho_1} \left( \rho_1 T_{s2} + \frac{\rho T_s}{1 - \rho} \right)\end{aligned}$$

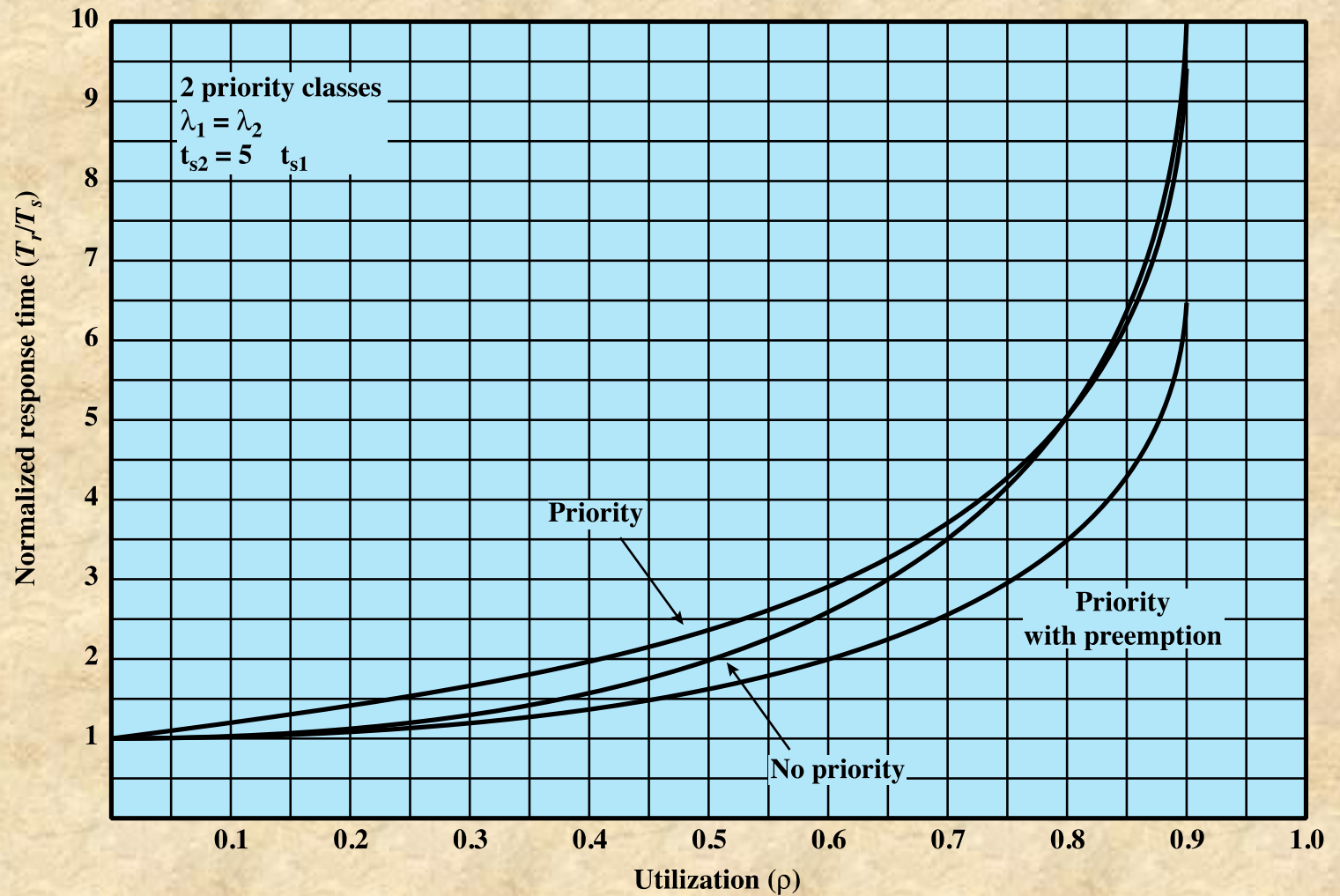


Figure 9.11 Overall Normalized Response Time



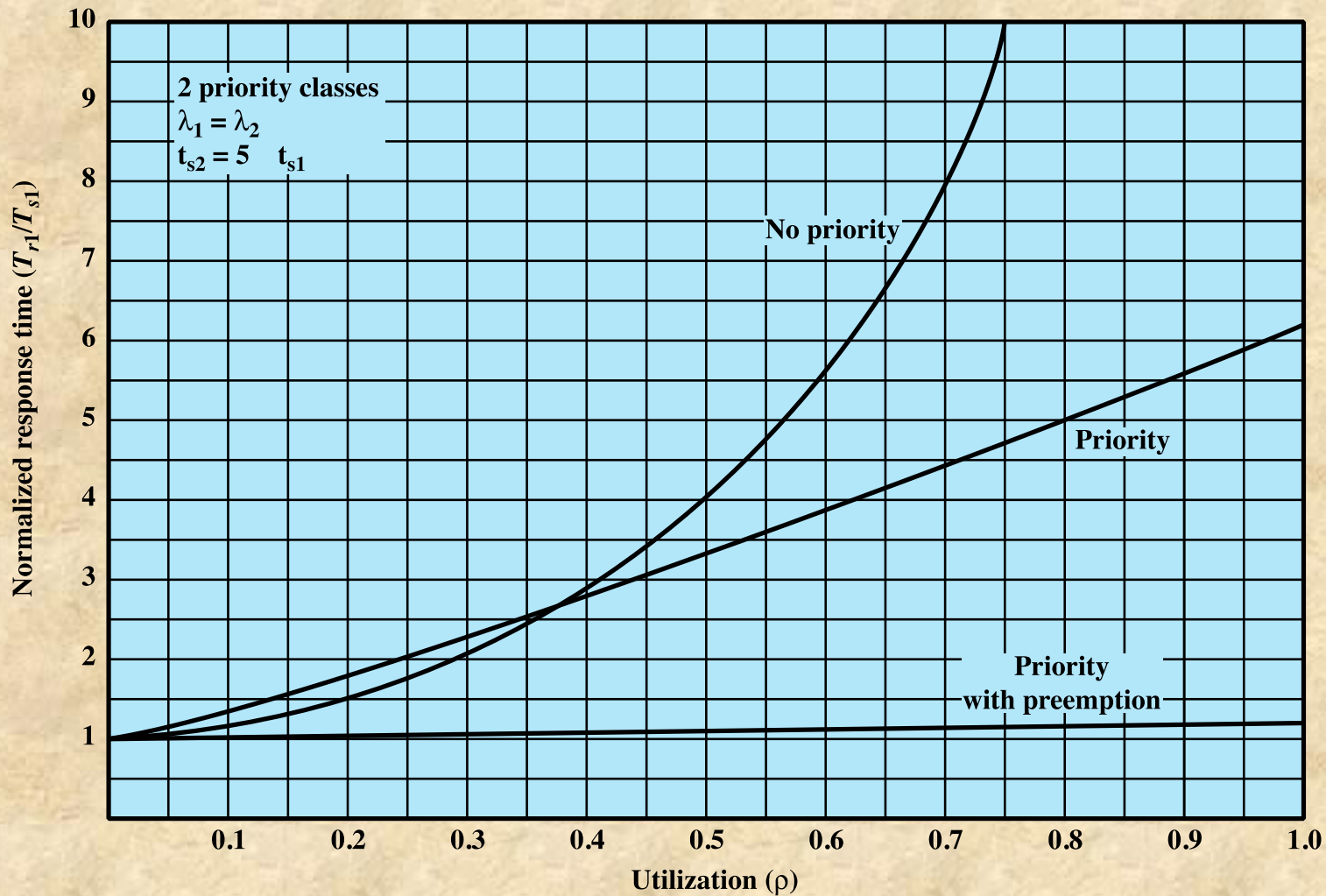
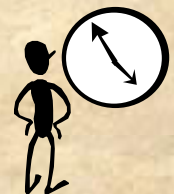


Figure 9.12 Normalized Response Time for Shorter Processes



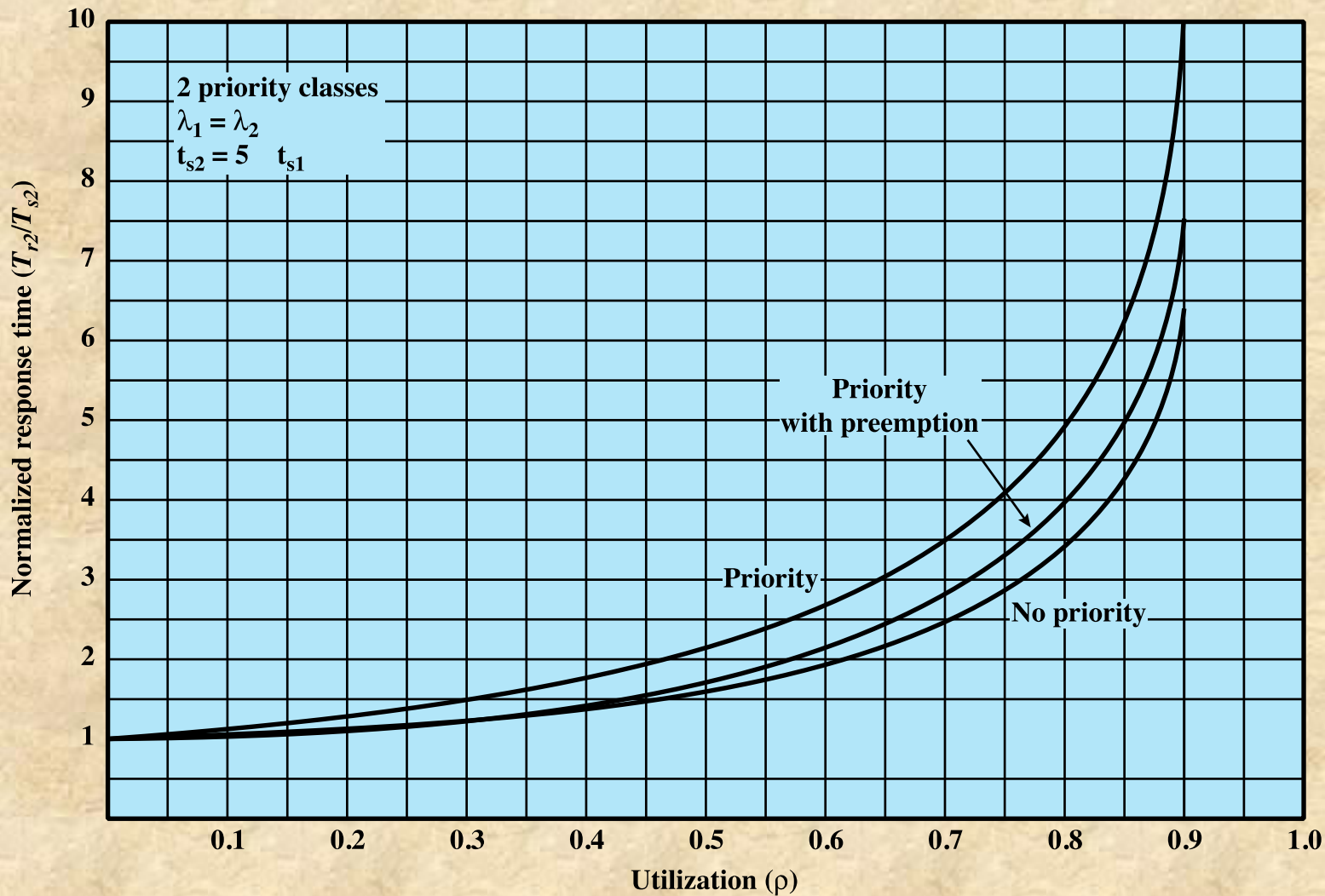
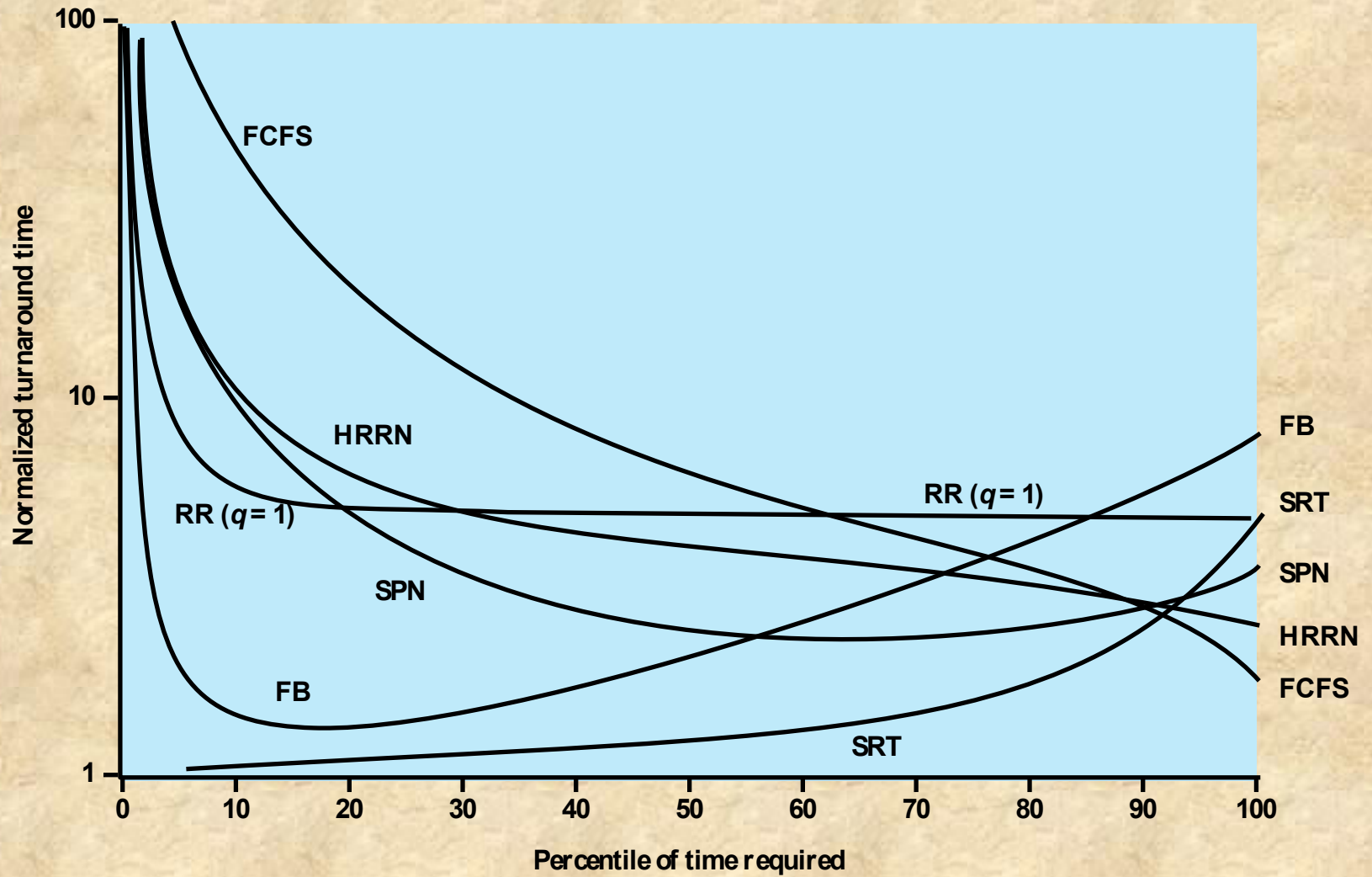


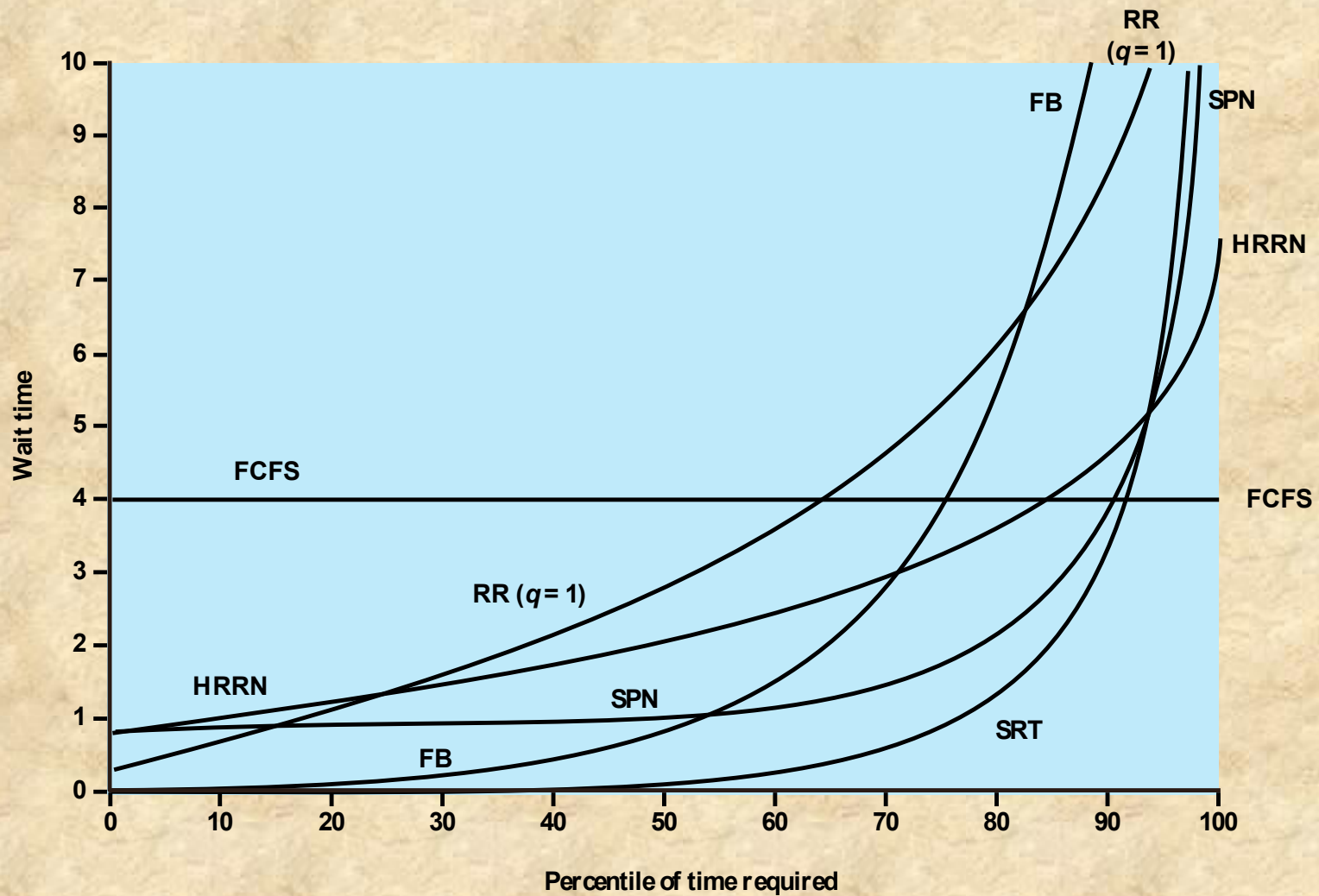
Figure 9.13 Normalized Response Time for Longer Processes







**Figure 9.14 Simulation Results for Normalized Turnaround Time**



**Figure 9.15 Simulation Results for Waiting Time**

# Fair-Share Scheduling

- Scheduling decisions based on the process sets
- 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







# 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 using round robin within each of the priority queues
- Makes use of one-second preemption
- 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  $i$

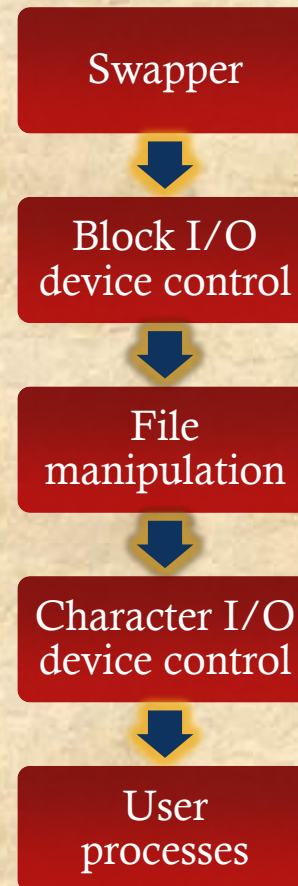
$P_j(i)$  = priority of process  $j$  at beginning of interval  $i$ ; lower values equal higher priorities

$Base_j$  = base priority of process  $j$

$nice_j$  = user-controllable adjustment factor

# Bands

- Used to optimize access to block devices and to allow the operating system to respond quickly to system calls
- In decreasing order of priority, the bands are:





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 1 2 • 60	0	60	0
2	67	15	75	30	60 1 2 • • 60	0
3	63 7 8 9 • • 67	7	67	15	75	30
4	76	33	63 7 8 9 • • 67	7	67	15
5	68	16	76	33	63	7

Colored rectangle represents executing process

**Figure 9.17 Example of Traditional UNIX Process Scheduling**



# Summary

- Types of processor scheduling
  - Long-term scheduling
  - Medium-term scheduling
  - Short-term scheduling
- Traditional UNIX scheduling
- Scheduling algorithms
  - Short-term scheduling criteria
  - The use of priorities
  - Alternative scheduling policies
  - Performance comparison
  - Fair-share scheduling