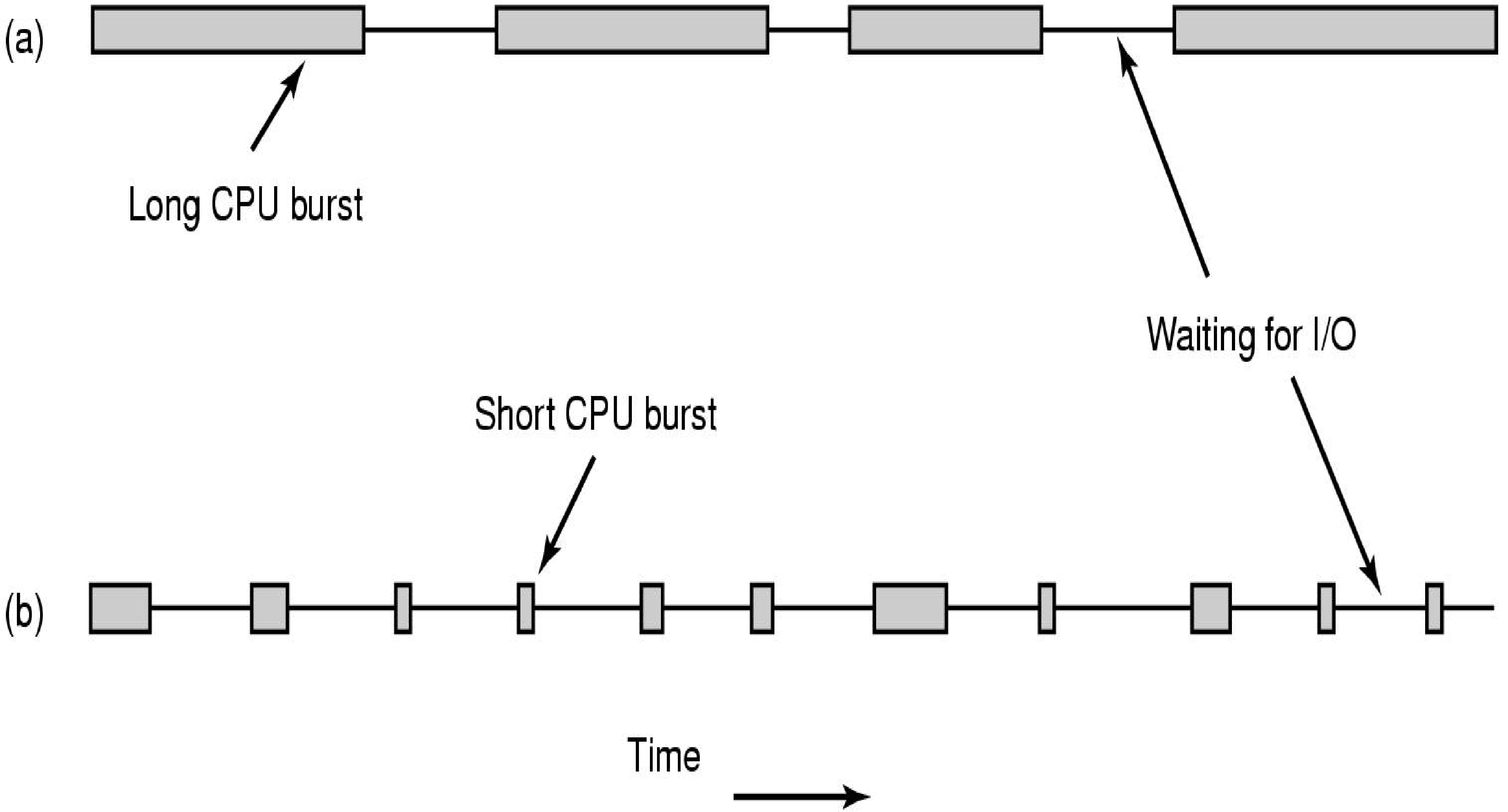
Uniprocessor Scheduling

Chapter 9

Operating Systems:
Internals and Design Principles, 9/E
William Stallings

CPU- and I/O-bound processes



Bursts of CPU usage alternate with periods of I/O wait

Goals of Scheduling

All systems

Fairness - giving each process a fair share of the CPU
Policy enforcement - seeing that stated policy is carried out
Balance - keeping all parts of the system busy

Batch systems

Throughput - maximize jobs per hour
Turnaround time - minimize time between submission and termination
CPU utilization - keep the CPU busy all the time

Interactive systems

Response time - respond to requests quickly
Proportionality - meet users' expectations

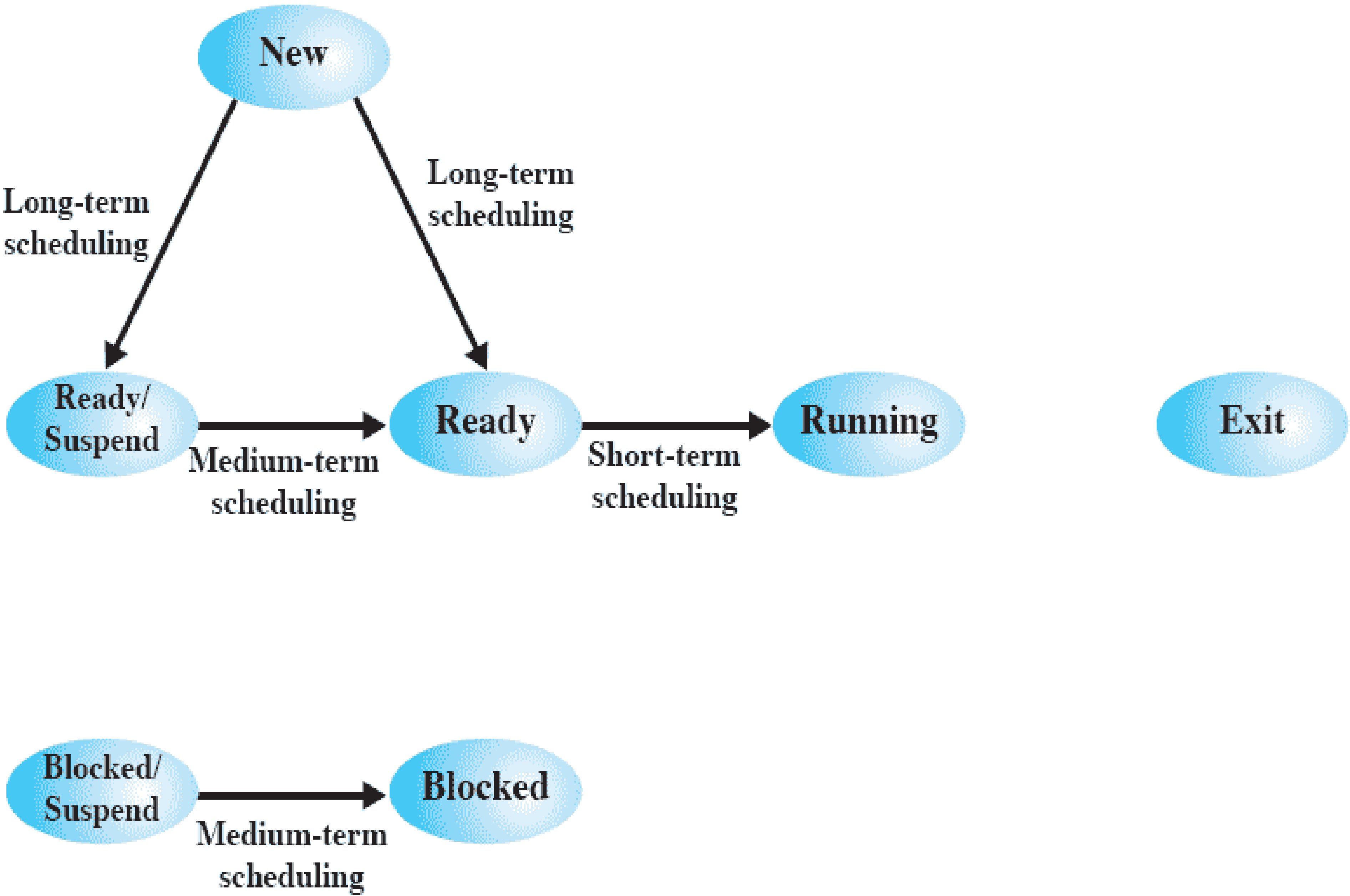
Real-time systems

Meeting deadlines - avoid losing data
Predictability - avoid quality degradation in multimedia systems

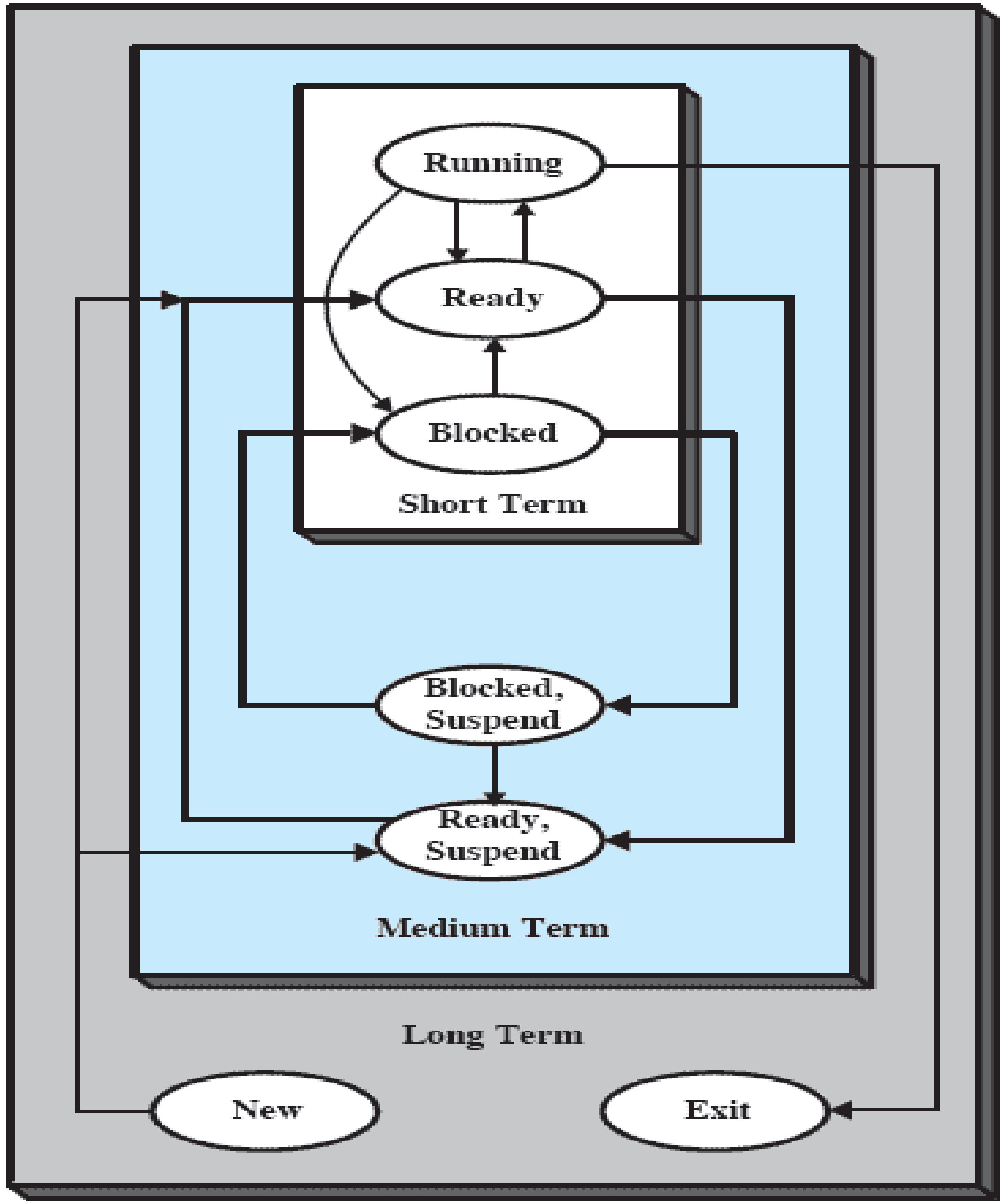
Types of Scheduling

- Long-term: admission into the system
- Medium-term: between main memory and hard disk
- Short-term/Dispatcher: between ready and running
- I/O

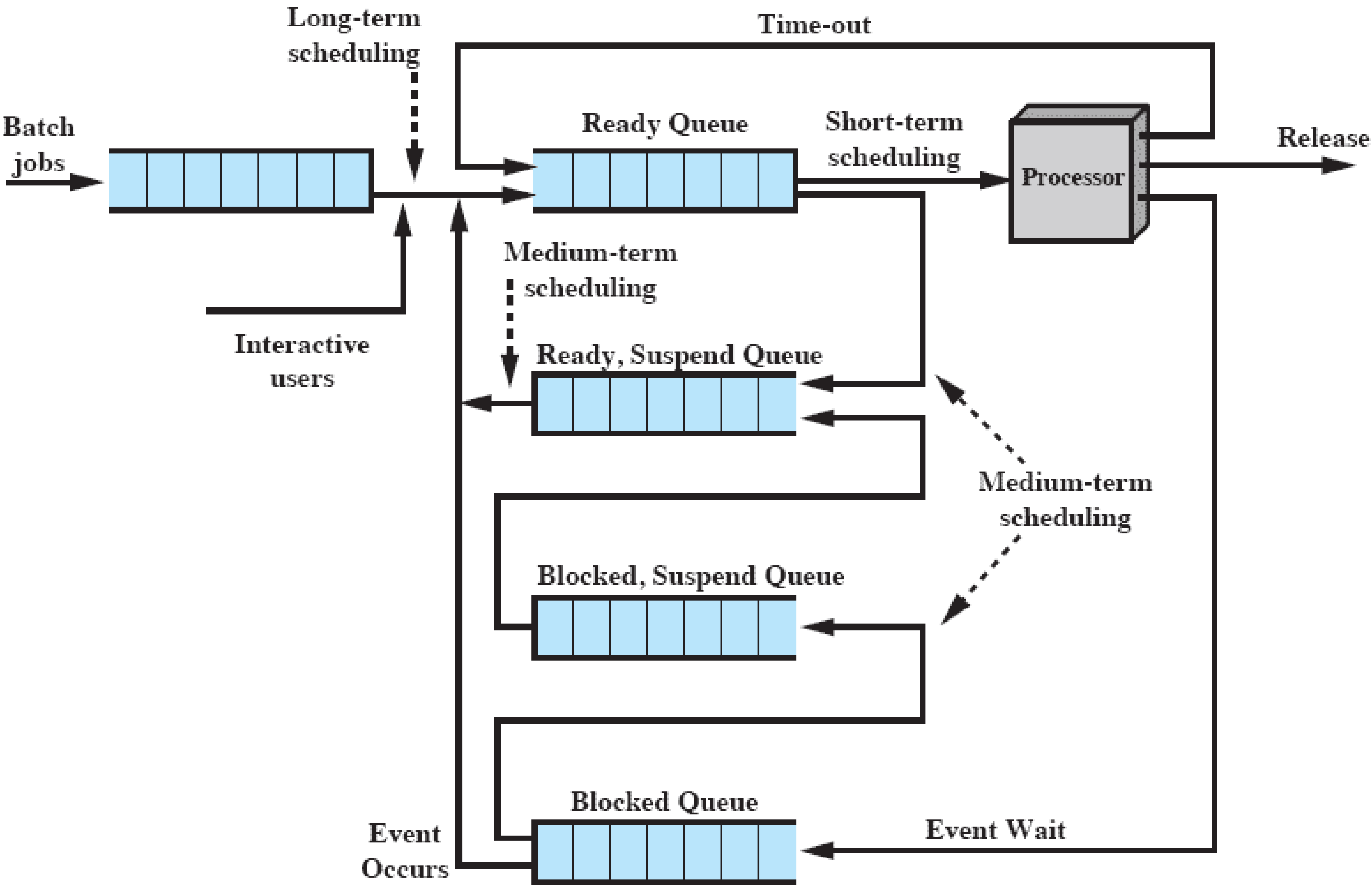
Scheduling and Process State Transitions



Levels of Scheduling



Queuing Diagram



Long-Term Scheduling

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming

Medium-term Scheduling

- Part of the swapping function
- Based on the need to manage the degree of multiprogramming

Short-Term Scheduling

- Known as the dispatcher
- Executes most frequently
- Invoked when an event occurs that triggers process switch
  – Clock interrupts
  – I/O interrupts
  – Operating system calls
  – Signals

Decision Mode

- Non-preemptive
  - Once a process is in the running state, it will continue until it terminates or blocks for I/O
- Preemptive
  - Currently running process may be interrupted and moved to the Ready state by the OS
  - Allows for better service, since no process can monopolize the processor for very long

Priorities

- Scheduler chooses a process of higher priority over one of lower priority
- Have multiple ready queues to represent each level of priority
- Lower-priority may suffer starvation
  – Change process priority based on its age or execution history – *dynamic* allocation of priorities

Priority Queuing

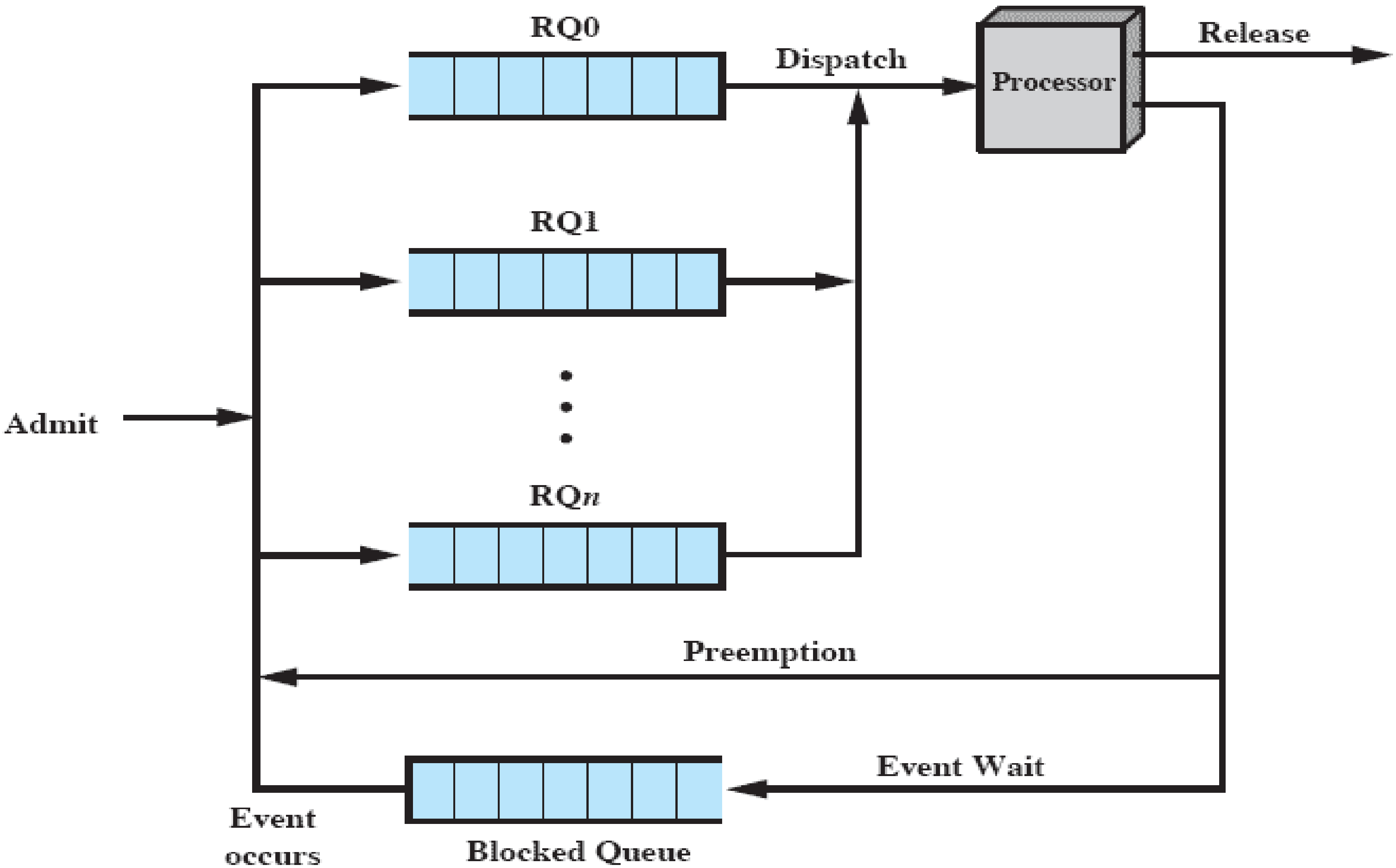
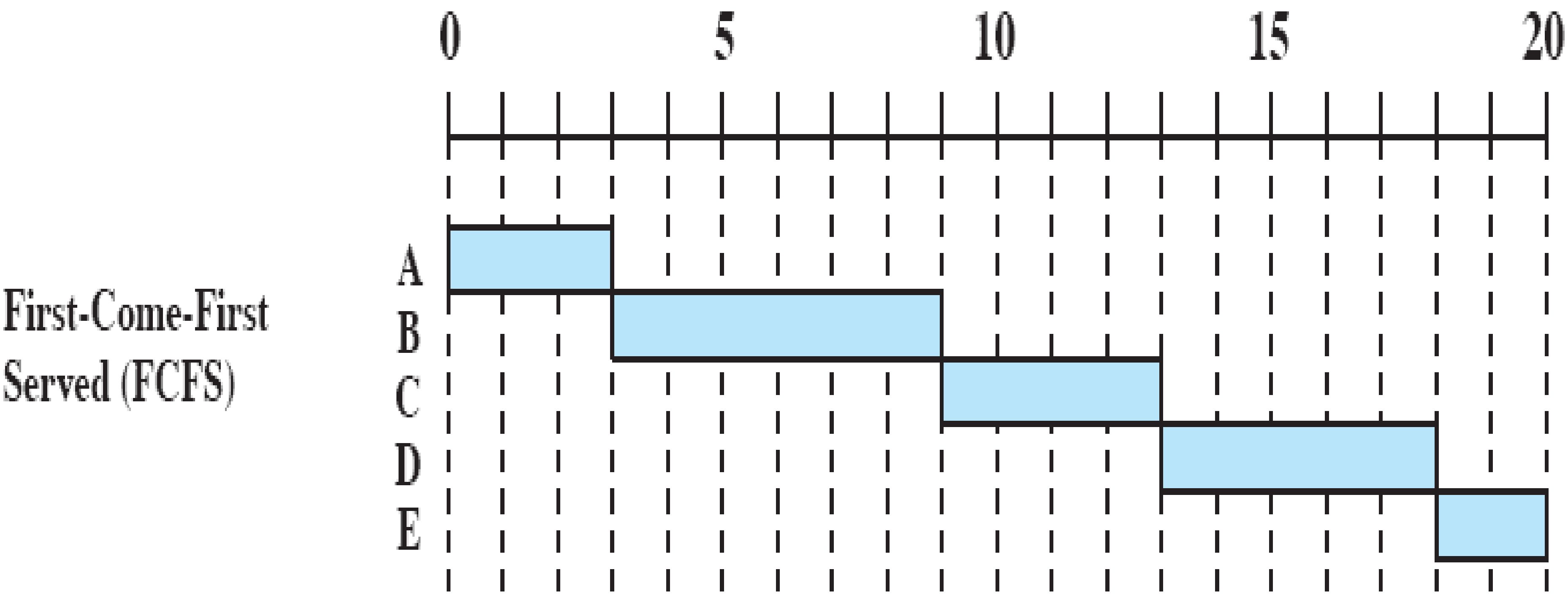


Table 9.4 Process Scheduling Example

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

First-Come-First-Served (non-preemtive)

- Each process joins the Ready queue
- When the current process ceases to execute, the *oldest* process in the Ready queue is selected

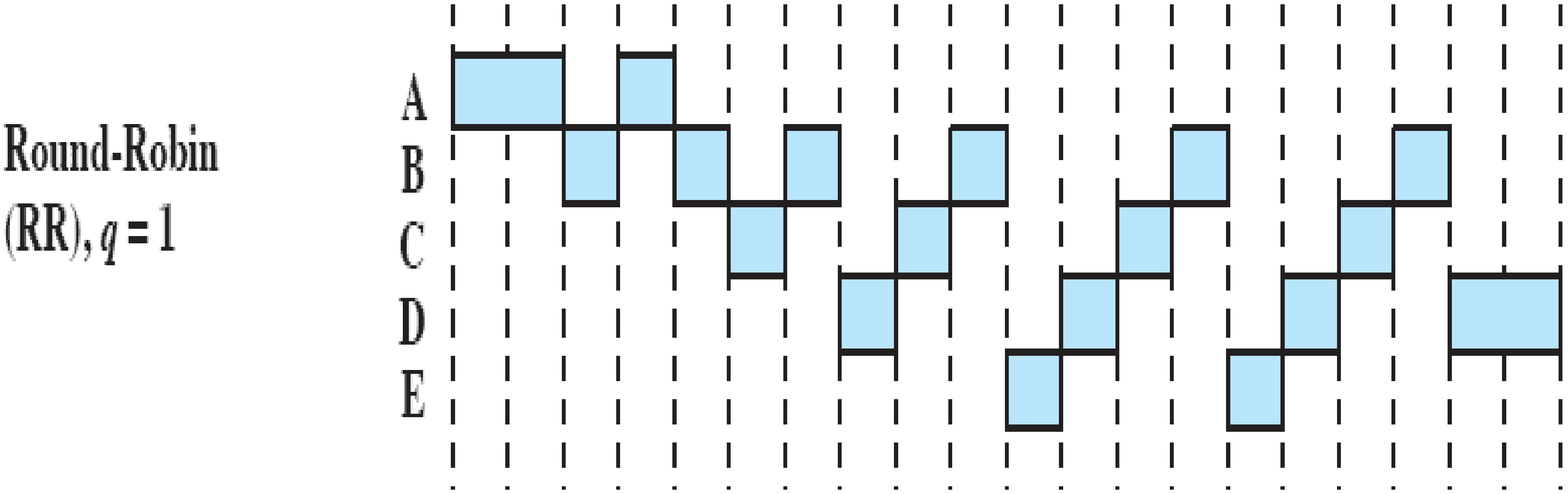


First-Come-First-Served (2)

- A short process may have to wait a very long time before it can execute
- Favours CPU-bound processes
  - I/O processes have to wait until CPU-bound process completes

Round Robin

- Uses preemption based on a clock
- Importance of the length of the quantum!!!



Round Robin (2)

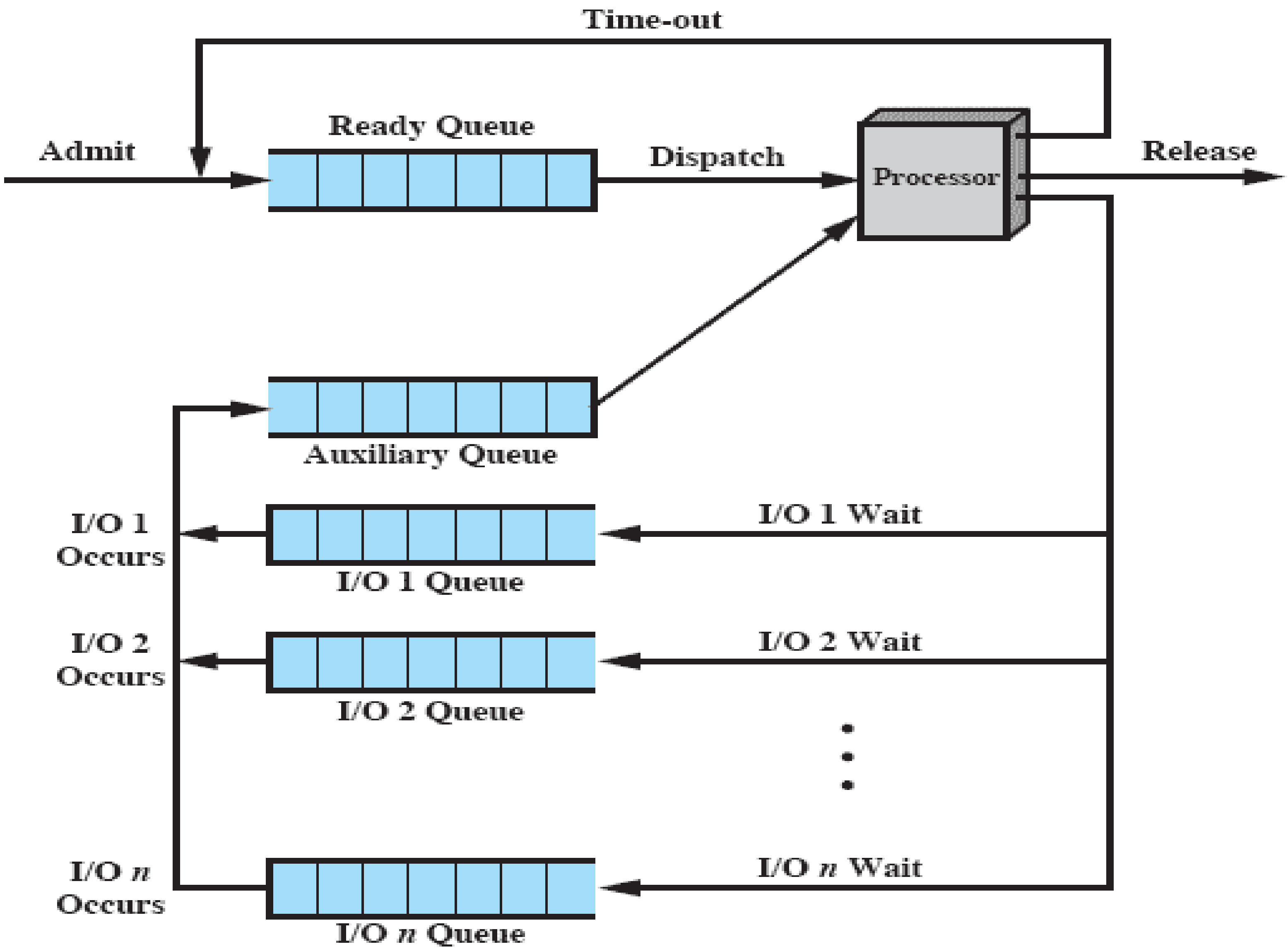
- Clock interrupt is generated at periodic intervals (dozens of msec)
- When an interrupt occurs,

the currently running process is placed in the ready queue

next ready job is selected

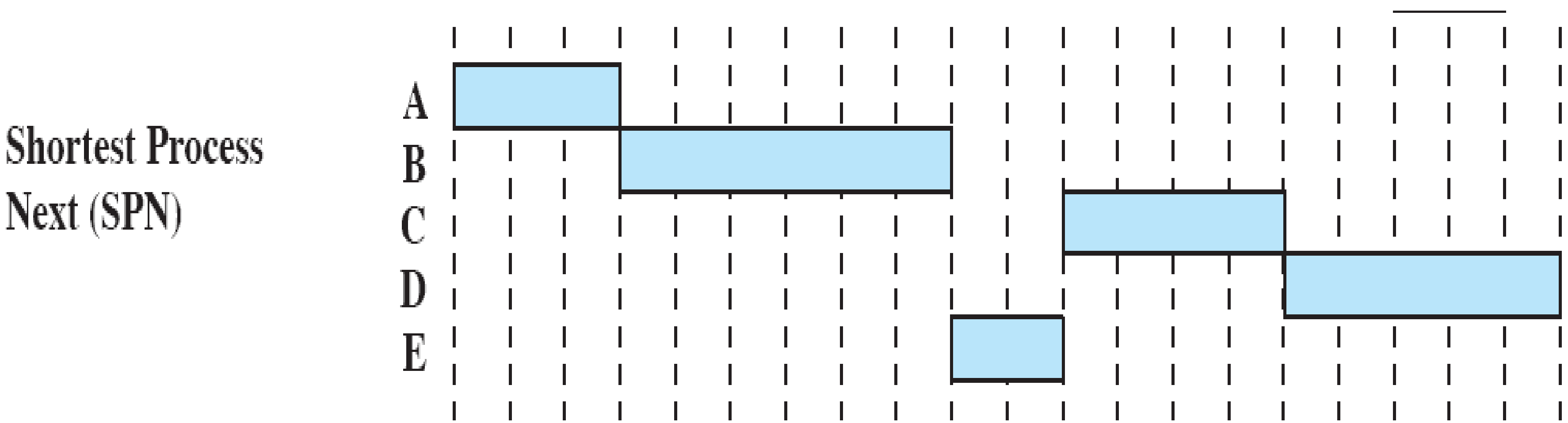
- Known as *time slicing*
- Importance of quantum – responsiveness
- Bad service for I/O-bound processes

Virtual Round-Robin



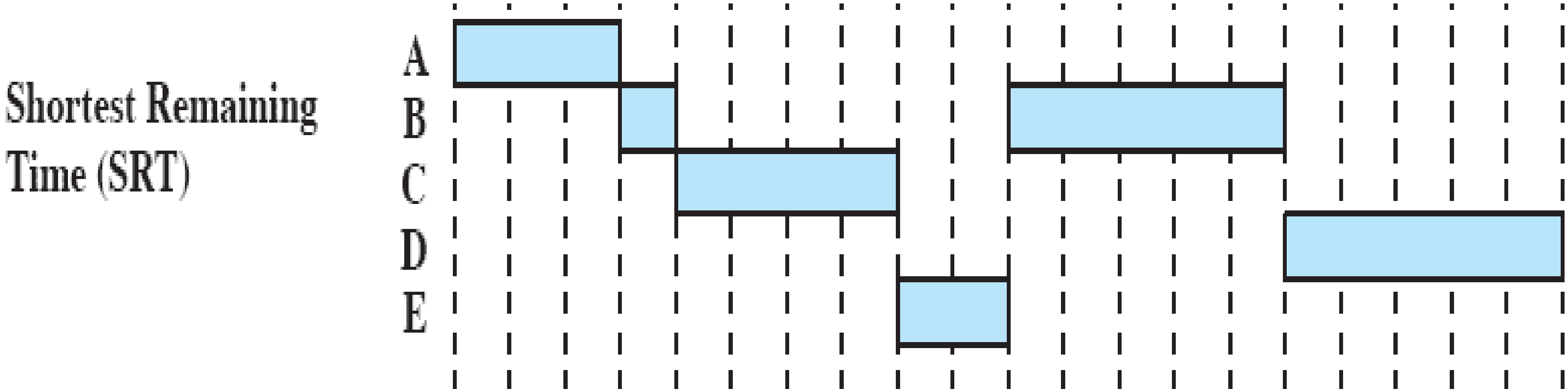
Shortest Process Next

- Non-preemptive policy
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes



Shortest Remaining Time

- Preemptive version of shortest process next policy
- Must estimate processing time



Predicting running times

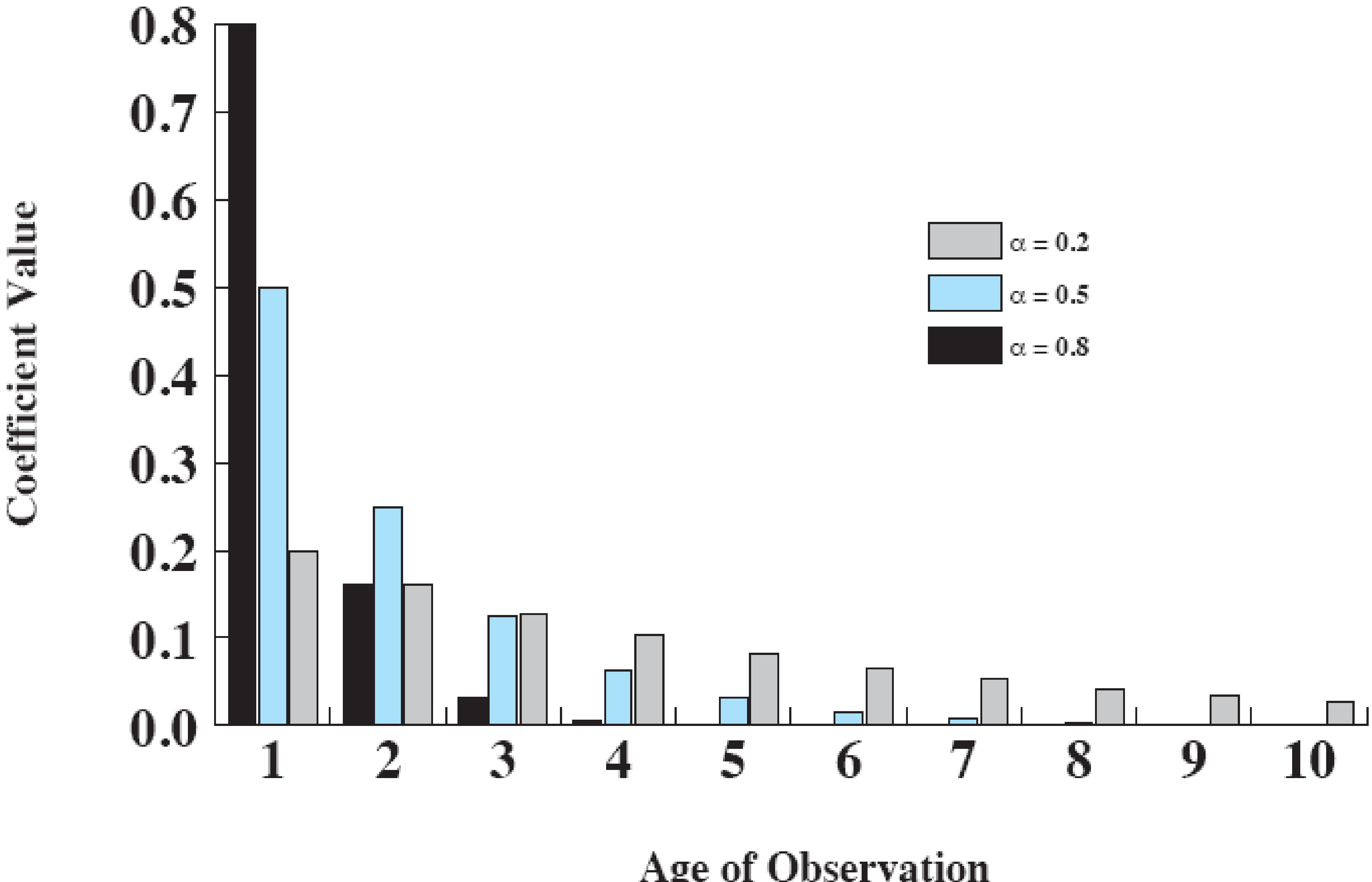
- OS has to estimate remaining running time or length of next CPU burst
- Simple or *weighted average*: for some $0<\alpha<1$

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

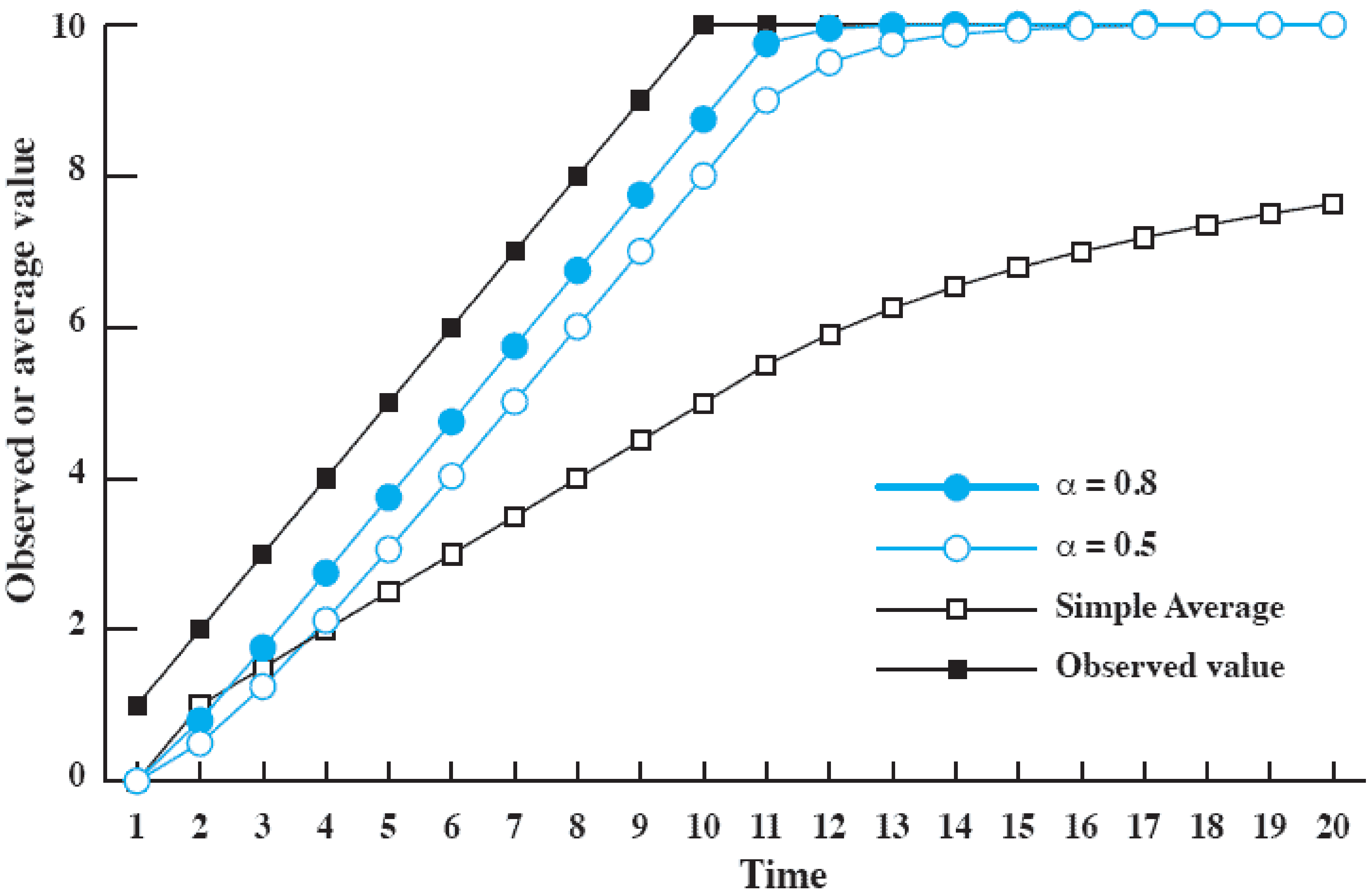
$$=\alpha T(n)+(1-\alpha)\alpha T(n-1)+...+(1-\alpha)^n S(1)$$

- Possibility of starvation for longer processes

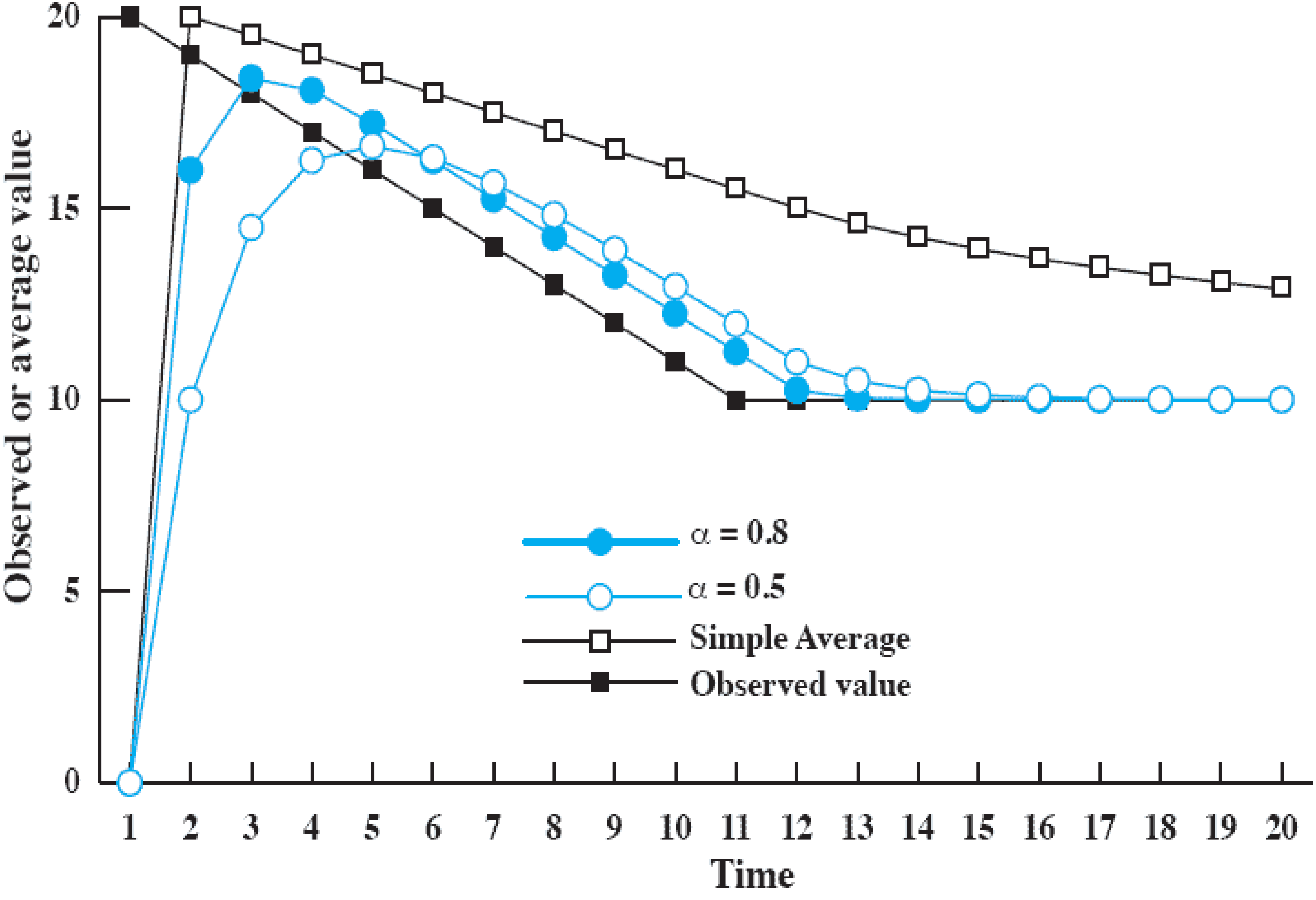
Exponential Smoothing Coefficients



Use Of Exponential Averaging



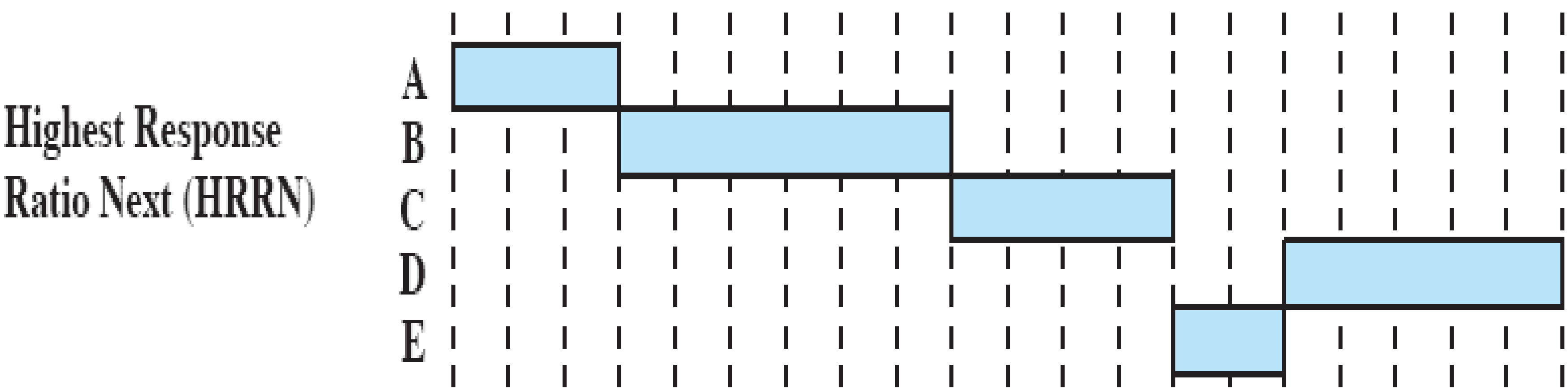
Use Of Exponential Averaging (2)



Highest Response Ratio Next

- Choose next process with the greatest ratio

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



Feedback Scheduling

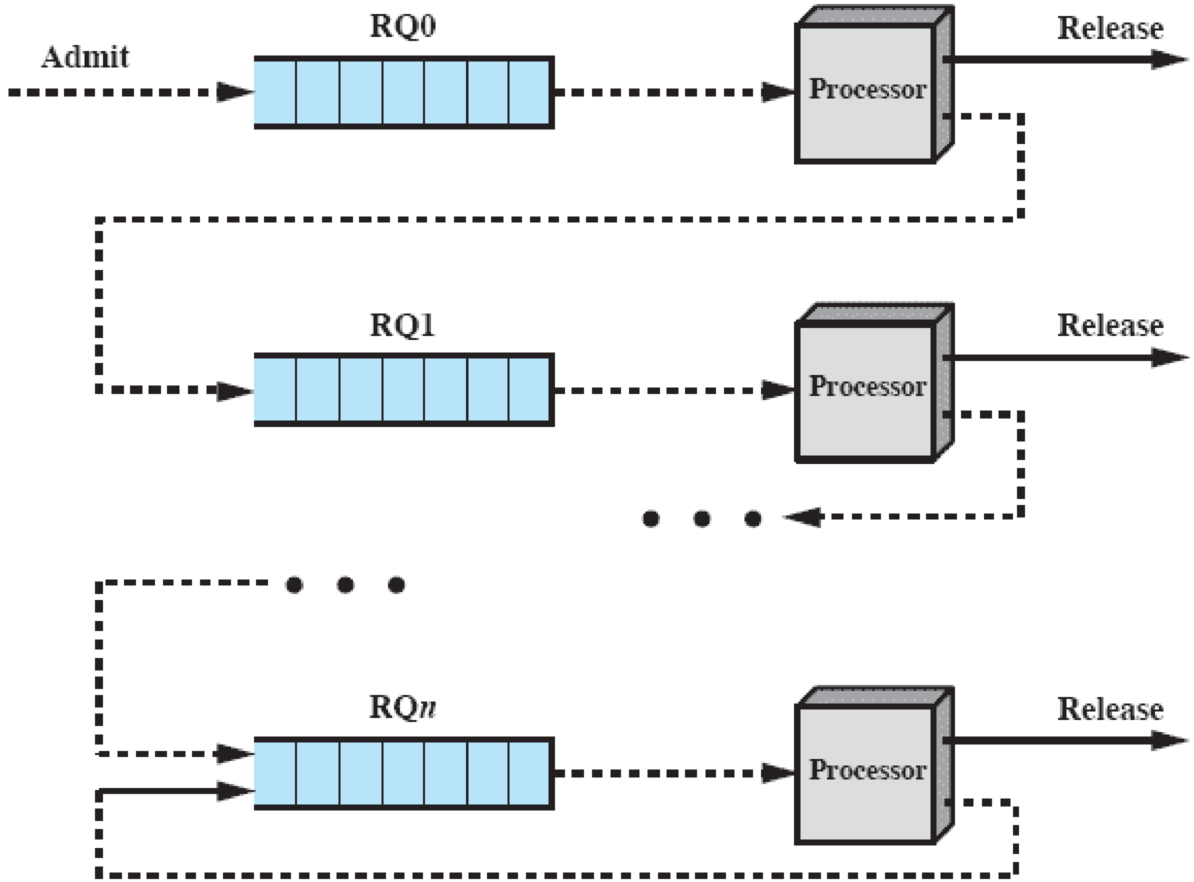
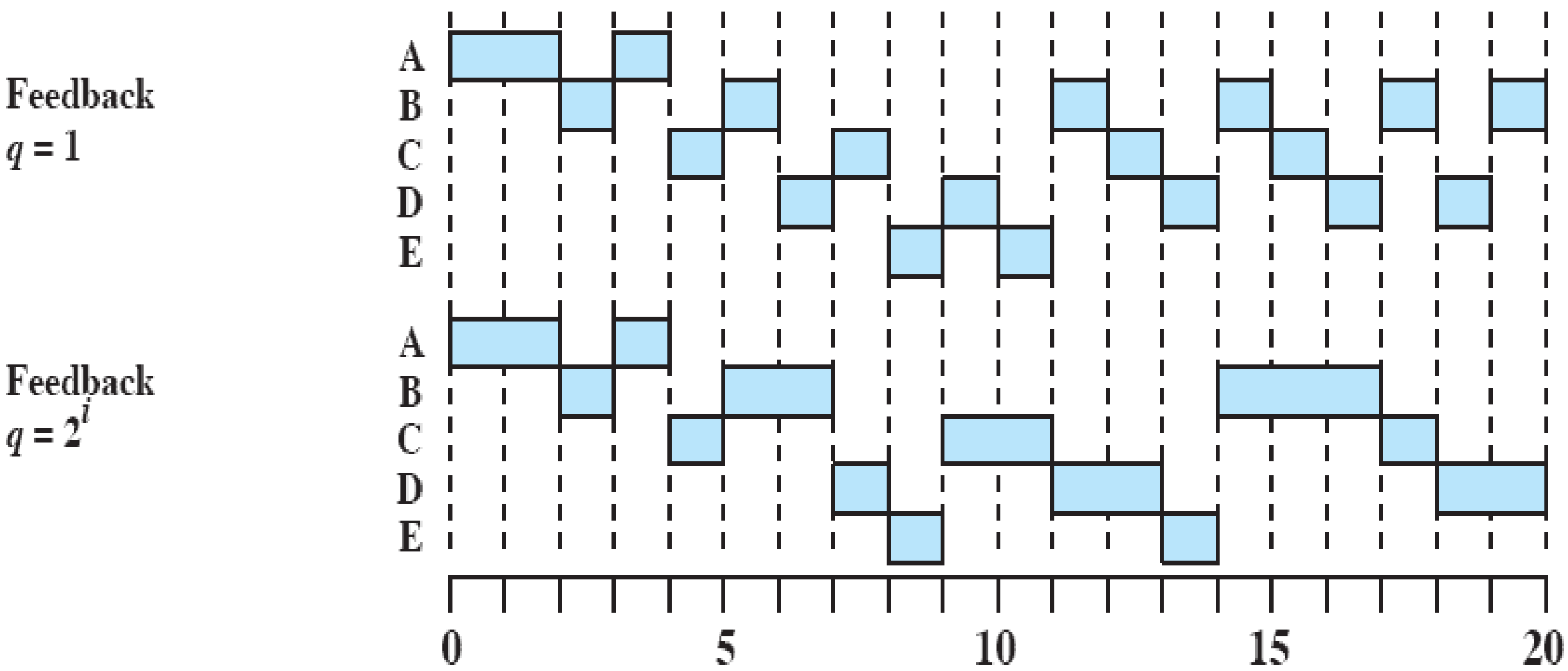


Figure 9.10 Feedback Scheduling

Feedback

- Penalize jobs that have been running longer
- Don't know remaining time process needs to execute



	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)
Through-Put	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

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 $q = 2^i$						
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

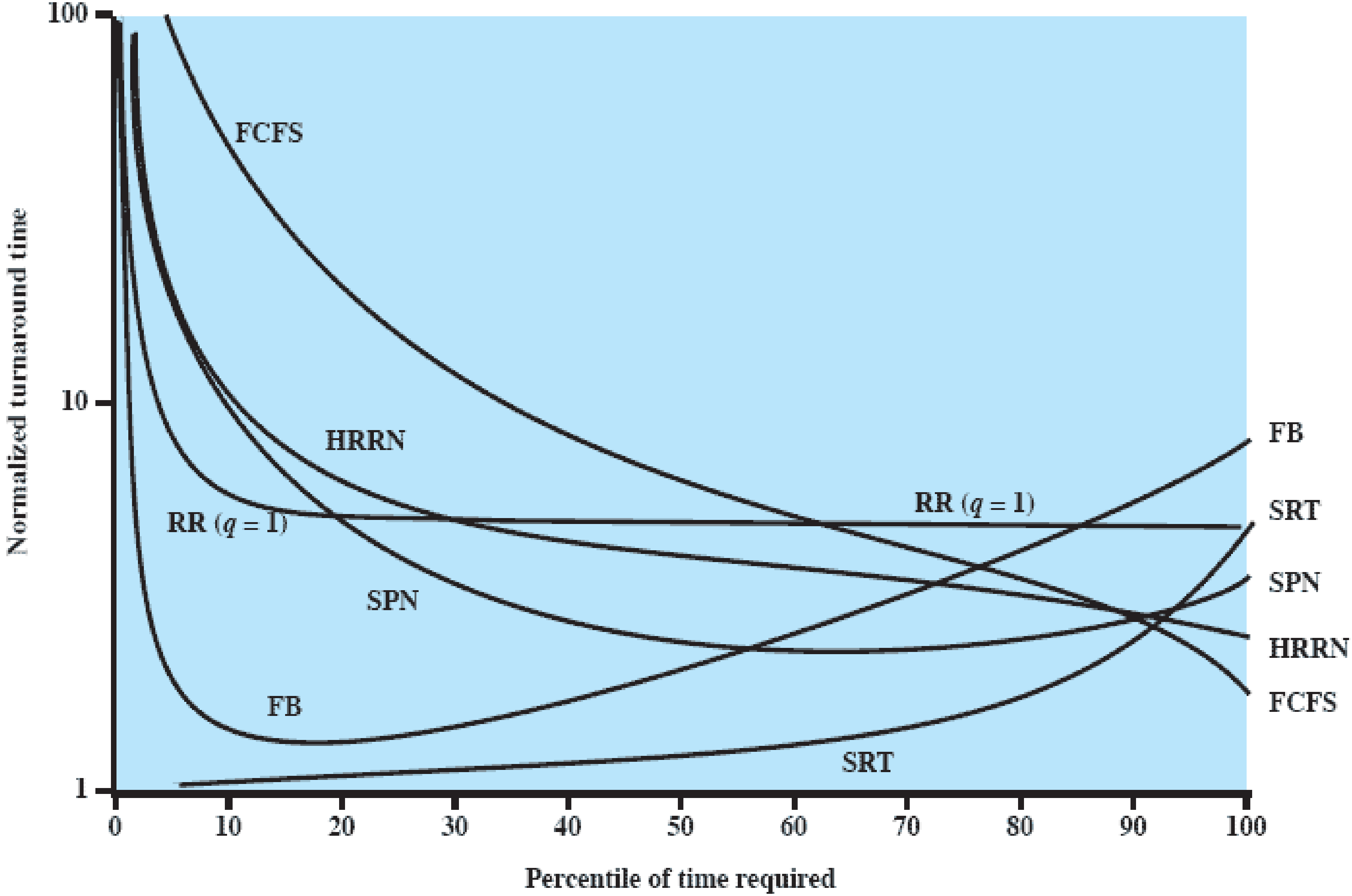


Figure 9.14 Simulation Results for Normalized Turnaround Time

Fair-Share Scheduling

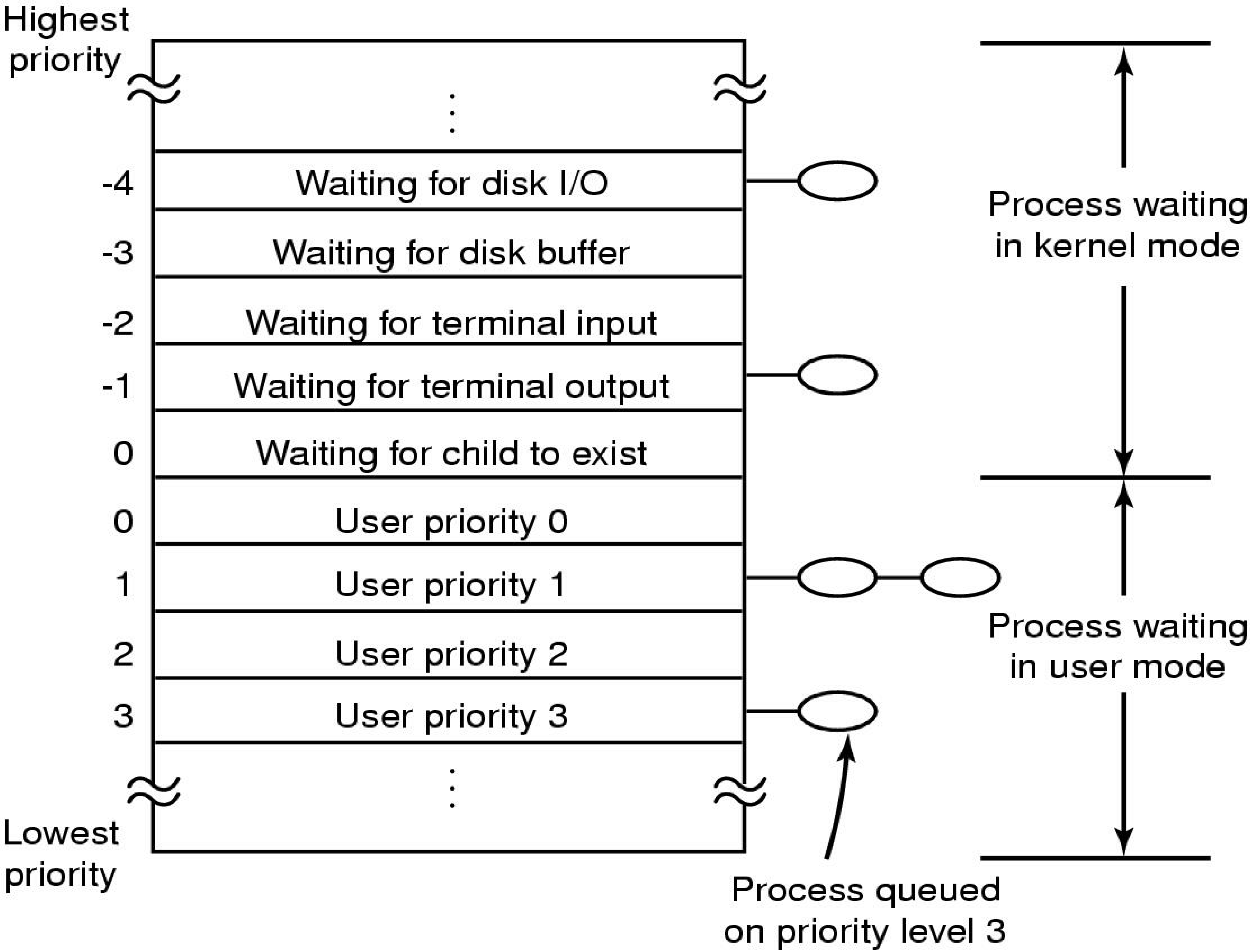
- User’s application runs as a collection of processes (threads)
- User is concerned about the performance of the application
- Need to make scheduling decisions based on process sets

Policy versus Mechanism

- Separate what is allowed to be done from how it is done
  - a process knows which of its children threads are important and need priority
- Scheduling algorithm parameterized
  - mechanism in the kernel
- Parameters filled in by user processes
  - policy set by user process for its threads
- Lottery scheduling

Traditional UNIX Scheduling

- Multilevel feedback using round robin within each of the priority queues
- If a running process does not block or complete within the quantum, it is preempted
- Priorities are recomputed once per second (e.g. starving processes get a boost)
- Processes coming back from I/O-wait get a boost



The UNIX scheduler is based on a multilevel queue structure