Operating Systems: Internals and Design Principles

Chapter 9 Uniprocessor Scheduling

Ninth Edition
By William Stallings

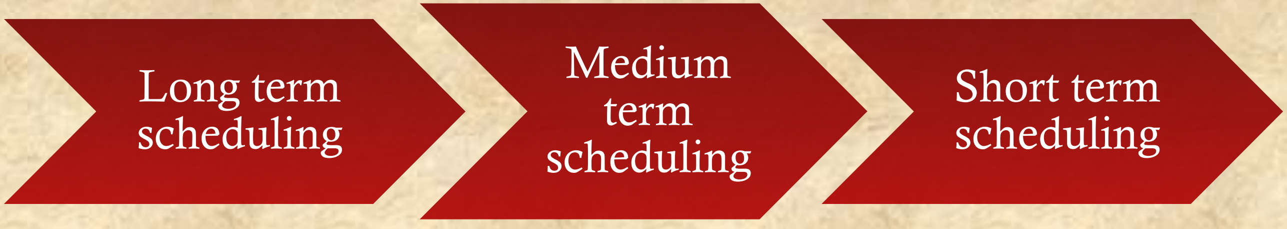
Table 9.1

Types of Scheduling

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	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:



Long term scheduling

Medium term scheduling

Short term scheduling

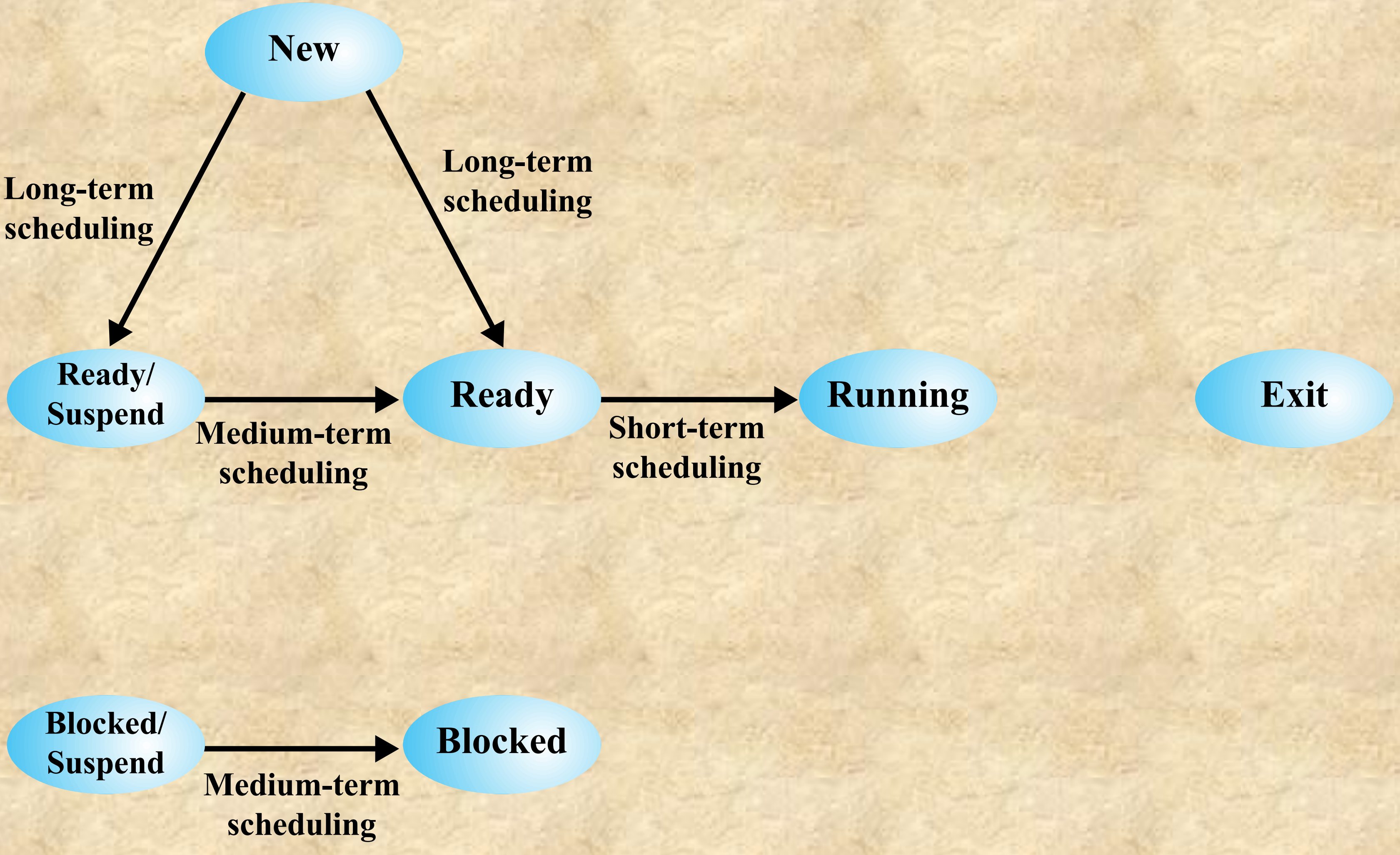


Figure 9.1 Scheduling and Process State Transitions

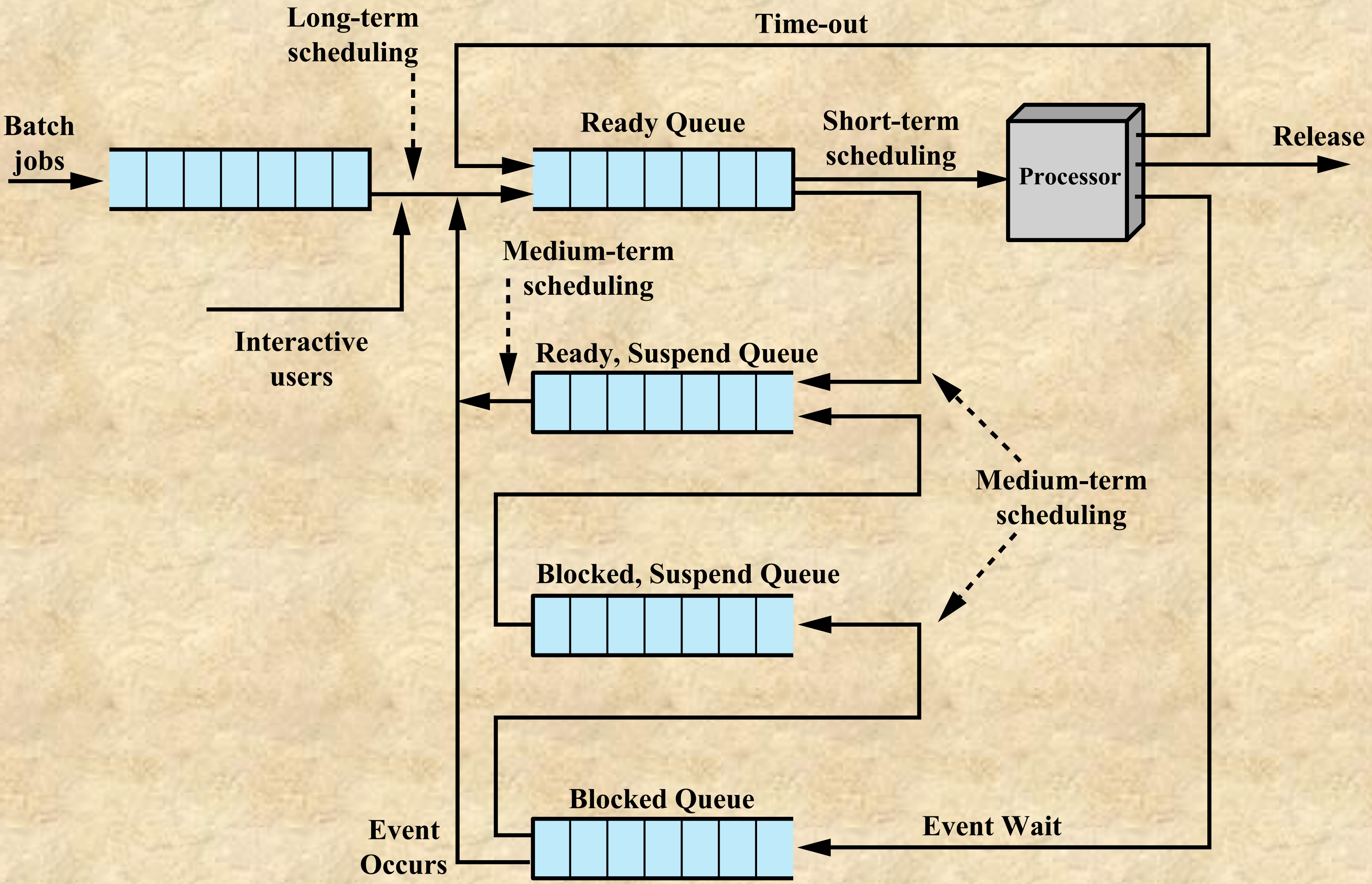


Figure 9.3 Queuing Diagram for Scheduling

Short-Term Scheduler

- 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

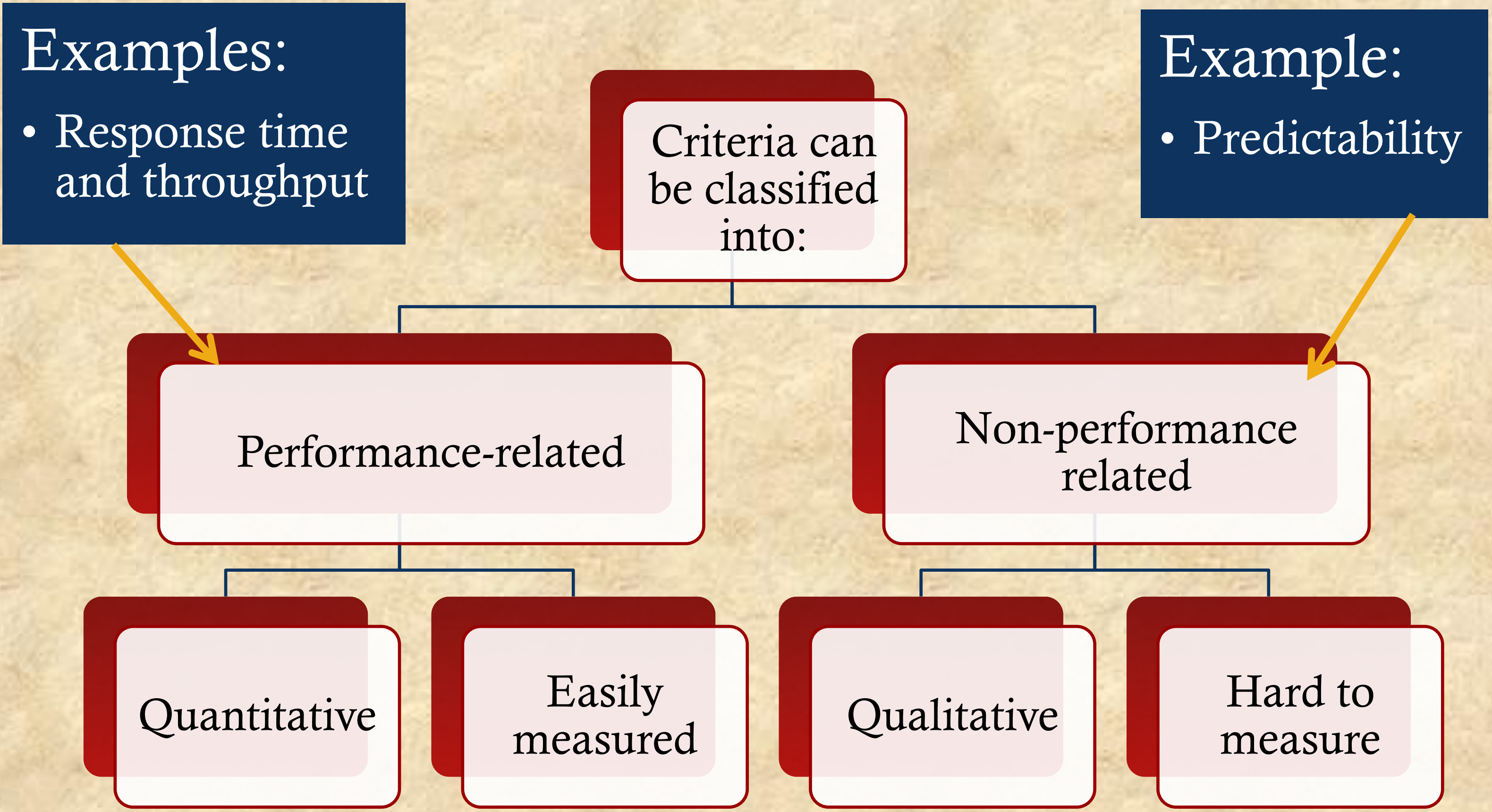
User-oriented criteria

- 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 is 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 and throughput

Criteria can be classified into:

Performance-related

Quantitative

Easily measured

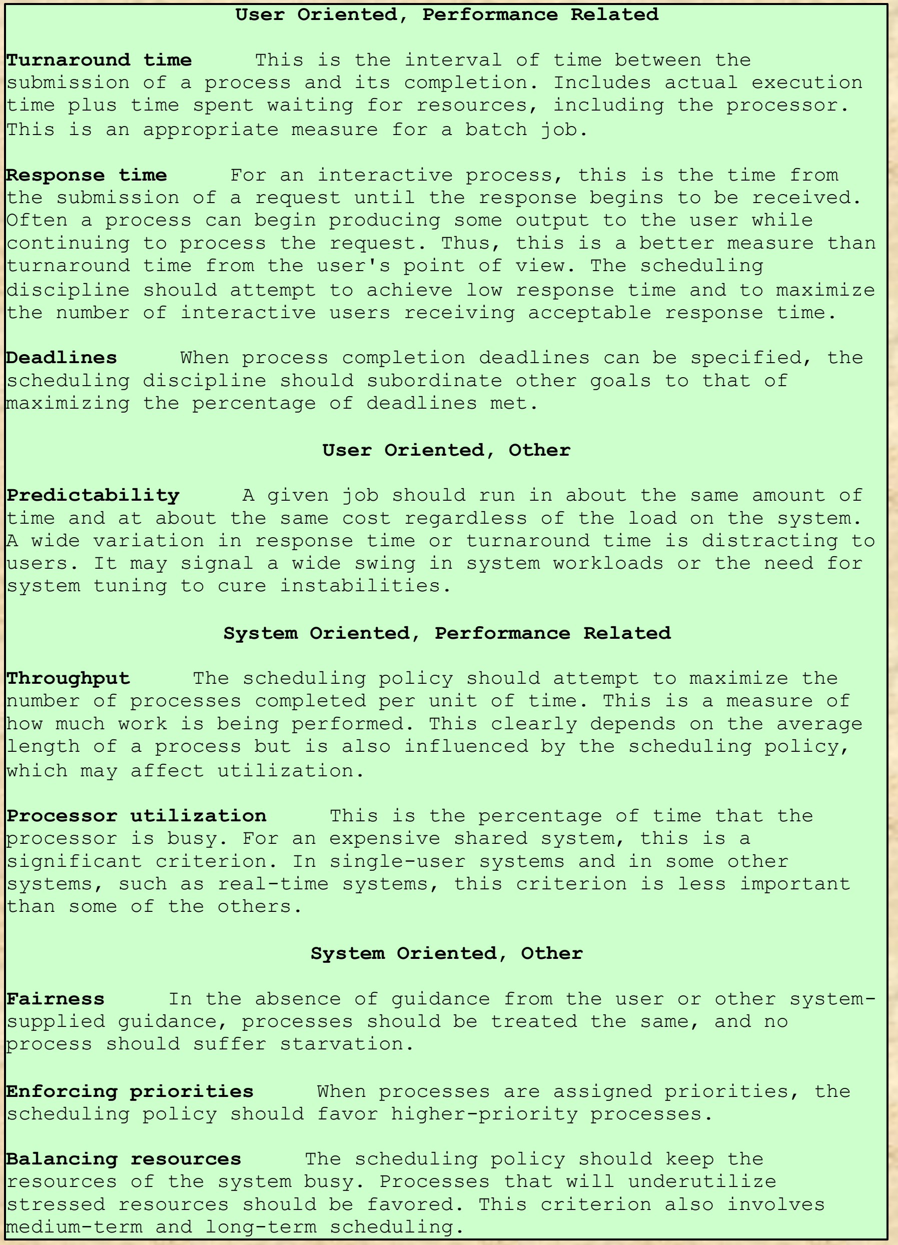
Non-performance related

Qualitative

Hard to measure

Example:

- Predictability



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 9.2

Scheduling Criteria

(Table can be found on page 403 in textbook)

Use of priority

- In most systems, a process can be assigned with some priority
- System will choose to execute process with higher priority
- Problems?

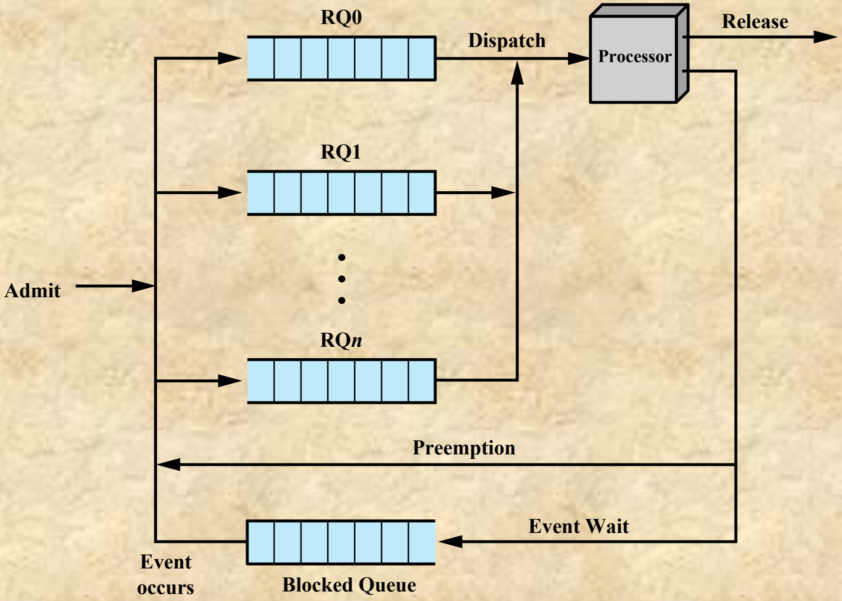


Figure 9.4 Priority Queuing

Starvation in pure priority-based scheduling

- Starvation: Lower priority processes might not be selected for execution if there are higher priority processes available for execution.
- Scheduling based on just priority may not be a good idea.
- Although the notion of priority is important in a system as well.

Scheduling Policies

- Next, we will look at several scheduling policies that are aimed at optimizing different objectives
- First, we must look at selection functions and decision modes
- **Selection functions** determines which process is to be selected next.
- **Decision modes** specifies the instants in time at which the selection function is exercised.

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

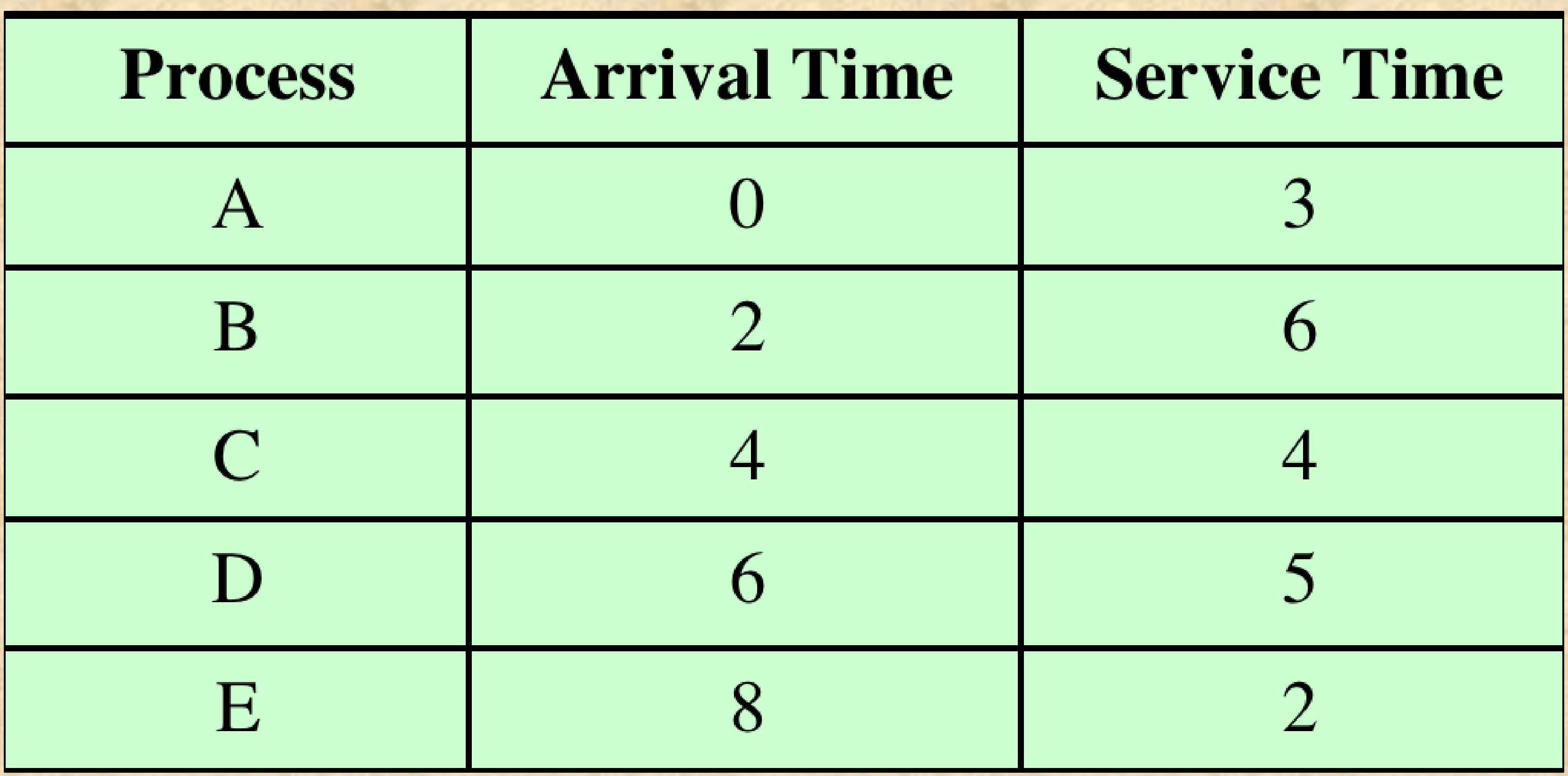
- Once a process is in the running state, it will continue until it terminates or blocks itself for I/O

Preemptive

- Currently running process may be interrupted and moved to ready state by the OS
- Decision to preempt may be performed when a new process arrives, when an interrupt occurs that places a blocked process in the Ready state, or periodically, based on a clock interrupt

Nonpreemptive vs Preemptive

- Preemptive policies incur greater overhead.
- Provides better services to the total population of processes.
- No monopolizing the CPU by one process
- Efficient process switching, hardware support, larger main memory can help preemtive scheduling policies to run efficiently.



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

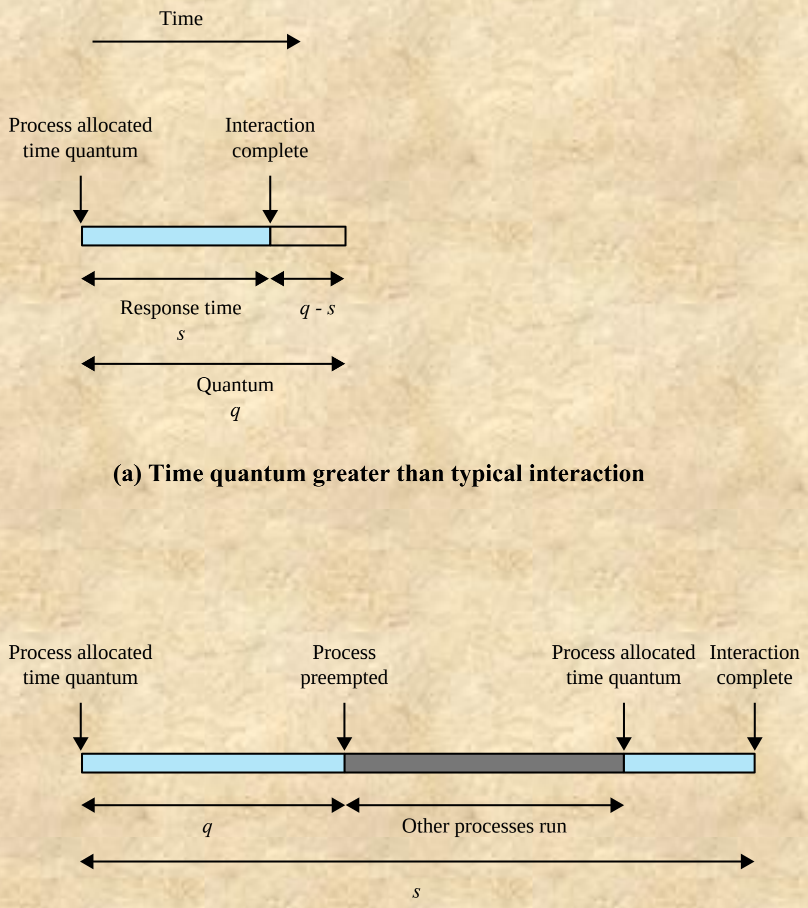
First-Come-First-Served (FCFS)

- Simplest scheduling policy
- Also known as first-in-first-out (FIFO) or a strict queuing scheme
- As each process becomes ready, it joins the ready queue
- When the currently running process ceases to execute, the process that has been in the ready queue the longest is selected for running

- Performs much better for long processes than short ones
- Tends to favor processor-bound processes over I/O-bound processes
- **Selection function: max[w]**
- **Decision mode: Non-preemptive**

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 **(q)**
- 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
- **Selection function: constant**
- **Decision mode: preemptive (at a given time quantum : q)**



(a) Time quantum greater than typical interaction

(b) Time quantum less than typical interaction

Figure 9.6 Effect of Size of Preemption Time Quantum

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
- Variability of response time is increased (less predictable)

- **Difficulty**: 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
- **Selection function: min(s)**
- **Decision mode: Non-Preemptive**

Shortest Remaining Time (SRT)

- Preemptive version of SPN
- Scheduler always chooses the process that has the shortest expected remaining processing time (at arrival)
- Risk of starvation of longer processes
- **Selection function: min(s-e)**
- **Decision mode: Non-Preemptive**
- Should give superior turnaround time performance to SPN because a short job is given immediate preference to a running longer job
- Elapsed service time must be recorded.

Highest Response Ratio Next (HRRN)

- Chooses next process with the greatest ratio
- Attractive because it accounts for the age of the process
- **Selection function:** max$\left(\frac{w+s}{s}\right)$
- **Decision mode: Non-Preemptive**

- While shorter jobs are favored, aging without service increases the ratio so that a longer process will eventually get past competing shorter jobs
- Smaller process are preferred in this approach.

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

Feedback

- If we do not know the service time, then SPN, SRT, HRRN does not work.
- Then focus on time spent in the execution so far.
- Scheduling done at time quantum.
- Dynamic priority mechanism is used.
- Decision mode: preemptive at time quantum

- While shorter jobs are favored, aging without service increases the ratio so that a longer process will eventually get past competing shorter jobs
- Smaller process are preferred in this approach.

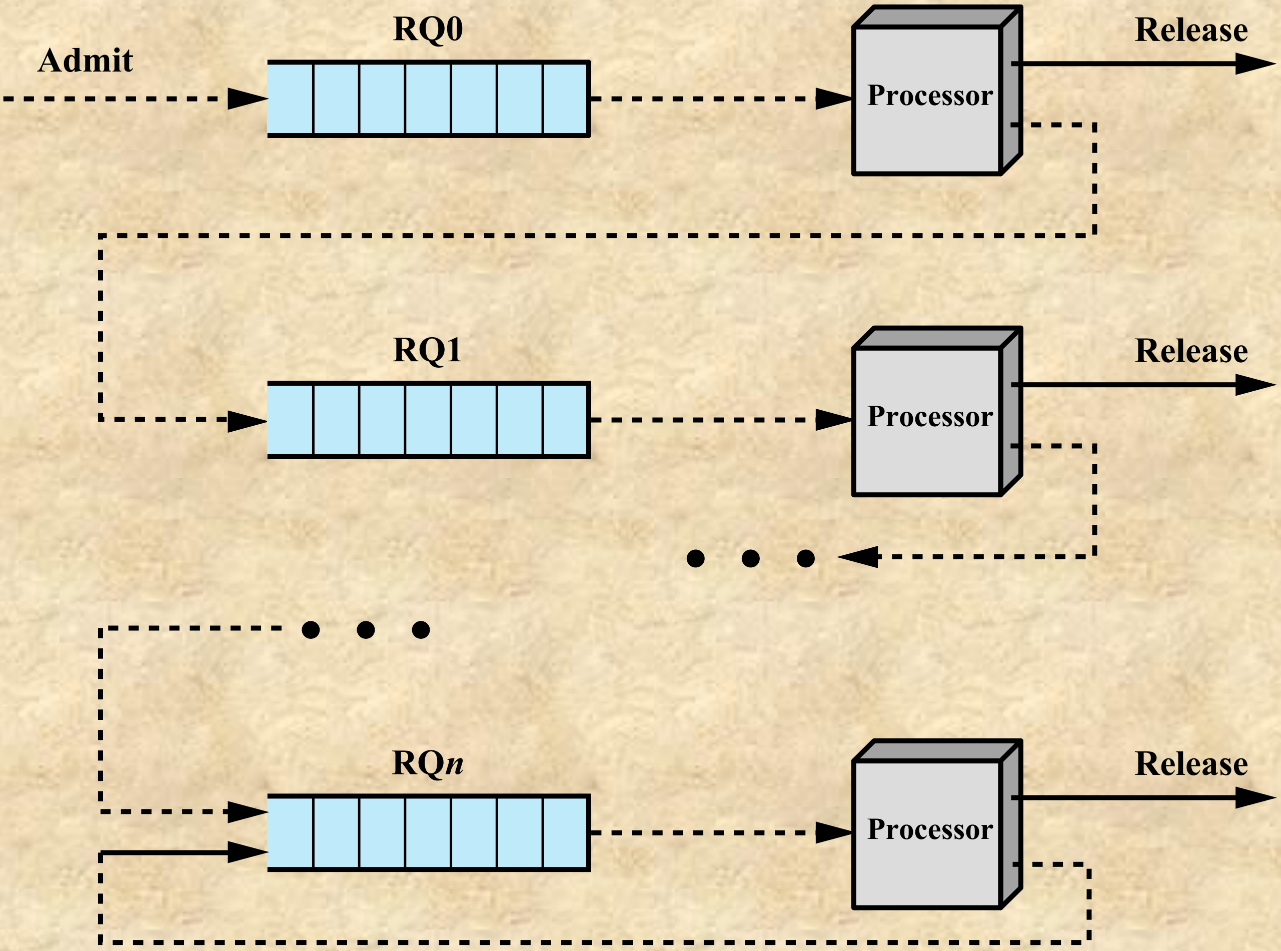


Figure 9.10   Feedback Scheduling

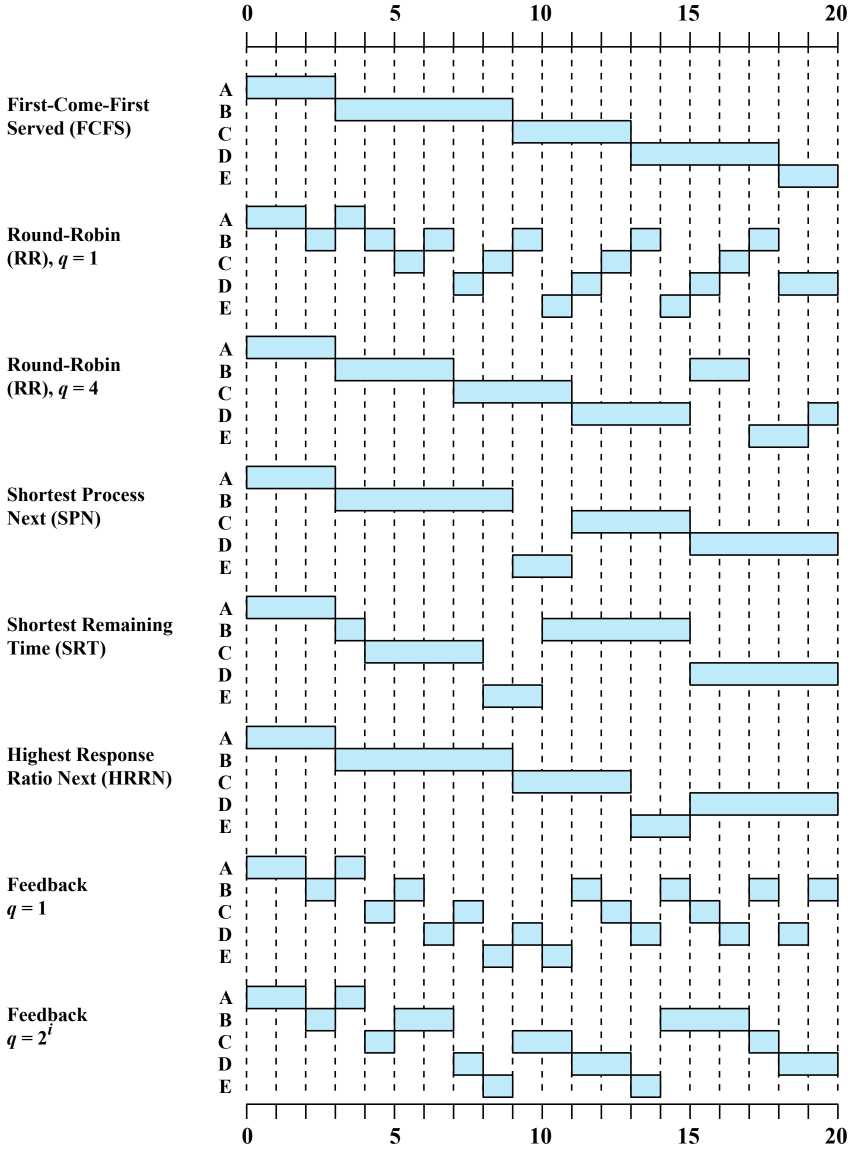


Figure 9.5 A Comparison of Scheduling Policies

Table 9.5

A 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* = 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

(Table is on page 408 in textbook)

	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

Table 9.3

Characteristics of Various Scheduling Policies

(Table can be found on page 405 in textbook)