

*Operating
Systems:
Internals
and
Design
Principles*

Chapter 9
Uniprocessor
Scheduling

Eighth Edition
By William Stallings

Processor Scheduling

- The 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
- Affects the performance of the system by managing queue to minimize queuing delays
- Broken down into three separate functions:





Figure 9.1 Scheduling and Process State Transitions

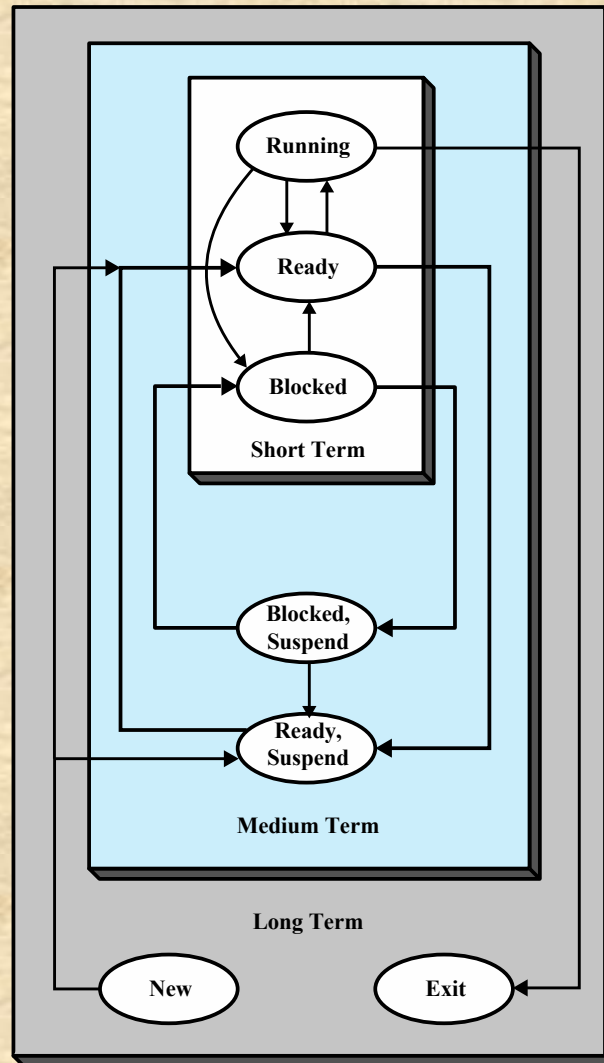


Figure 9.2 Levels of Scheduling

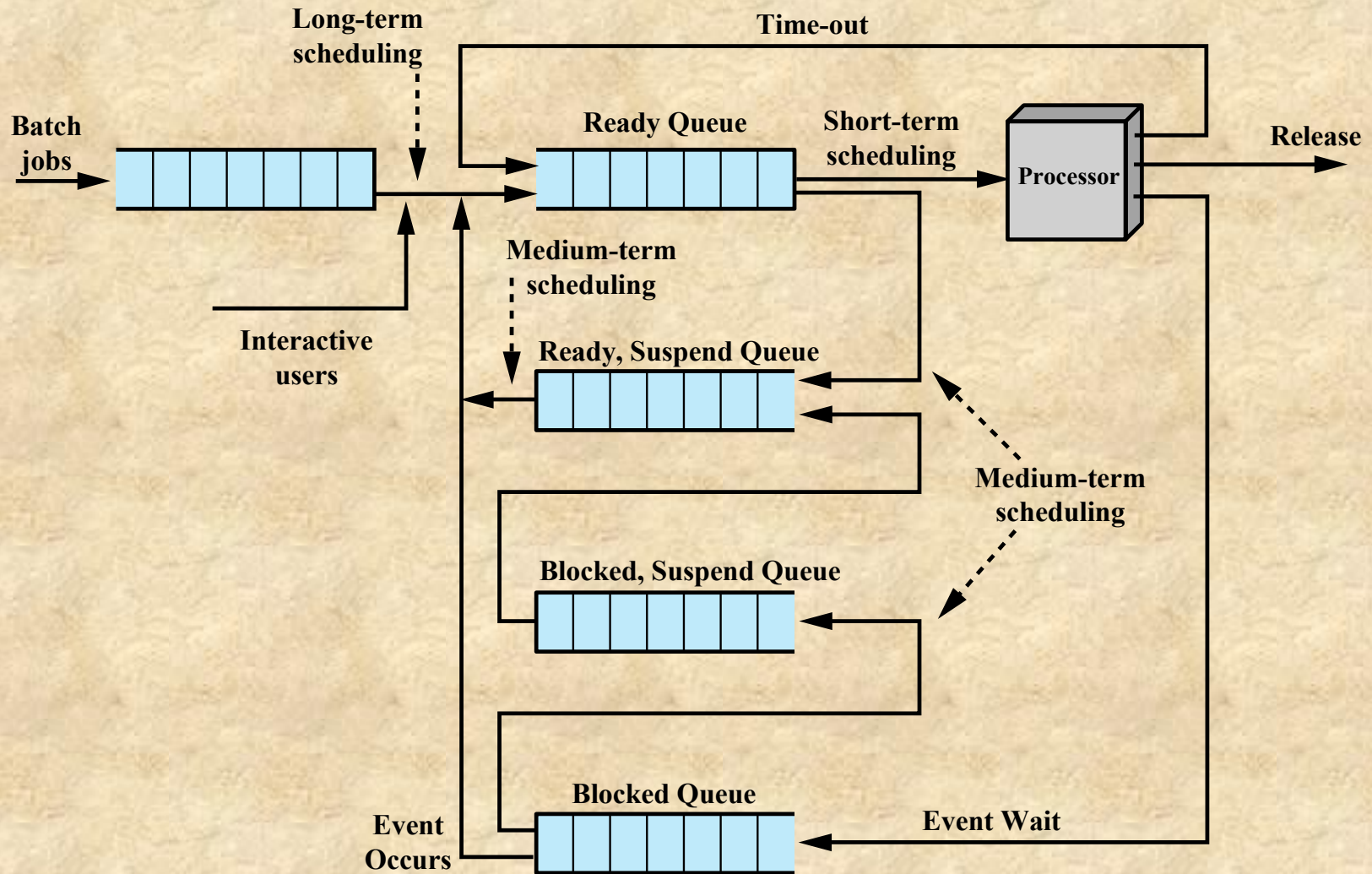
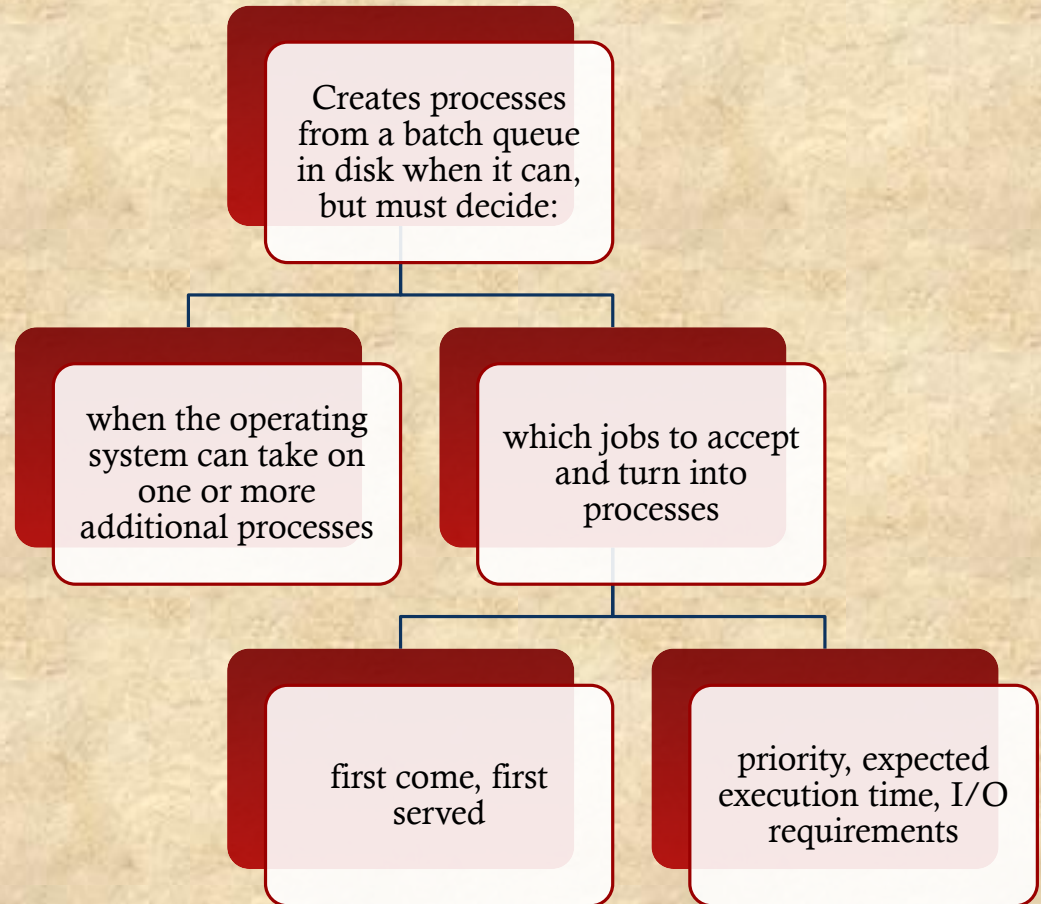


Figure 9.3 Queuing Diagram for Scheduling

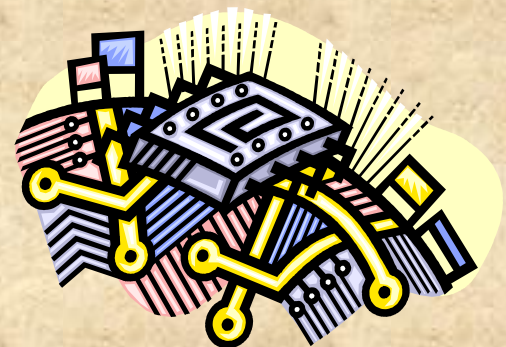
Long-Term Scheduler

- Performed when a new process is created
- Determines whether to add and 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 of Events:

- 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(say 2 sec.) in an interactive system)
- important on virtually all systems

System-oriented criteria

- focus on effective and efficient utilization of the processor (throughput: the rate at which processes are completed)
- focus on system performance rather than service provided to the user
- 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 &
analyse

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.

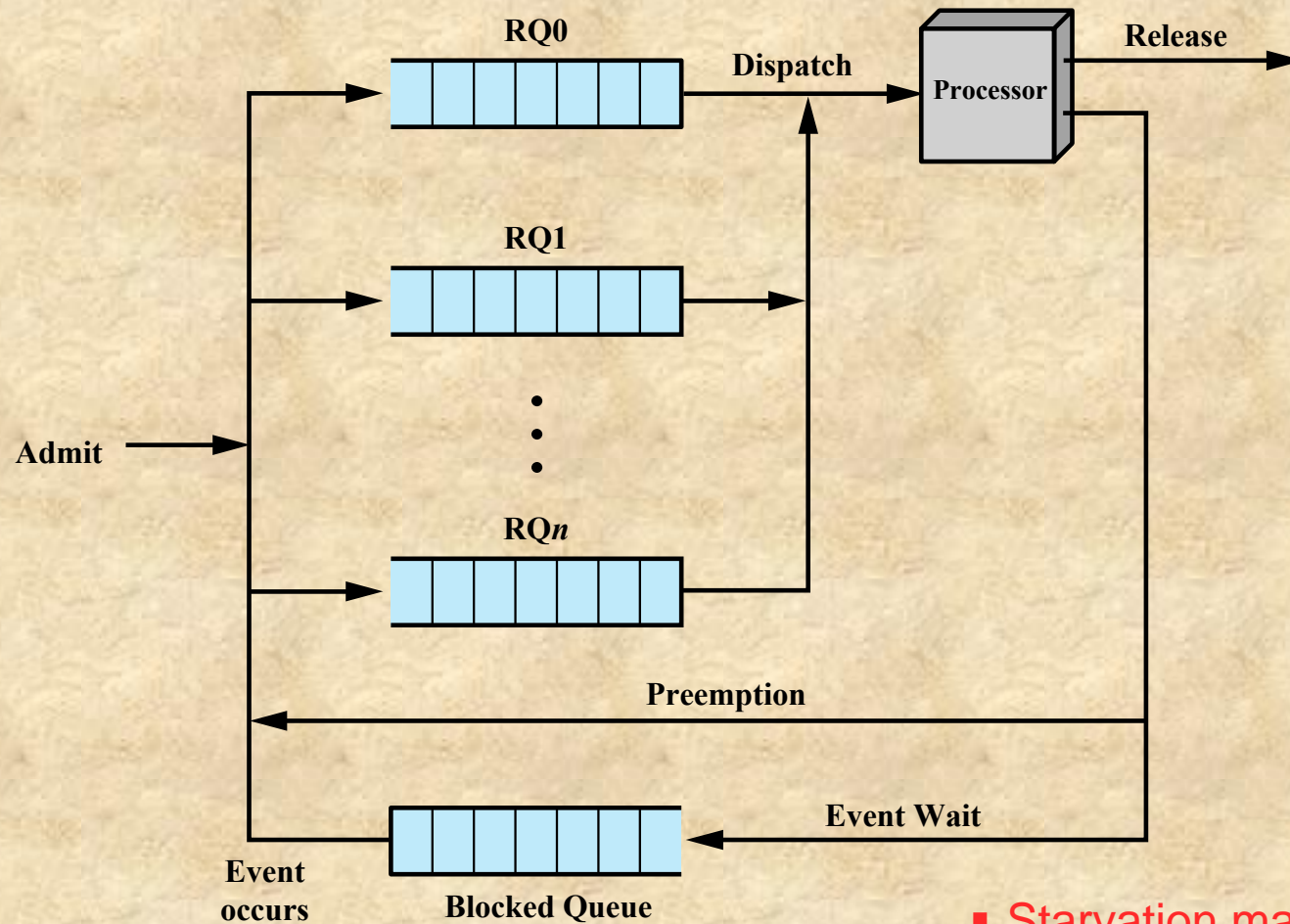
Balancing resources The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

Table 9.2

Scheduling Criteria

- Independent
- Impossible to optimize all of them simultaneously
- Need to compromise among conflicting requirements(e.g., response time vs throughput)

(Table can be found on page 433 in textbook)



- Starvation may occur
- Why occur?
- How to solve?

Figure 9.4 Priority Queuing

| | FCFS | Round robin | SPN | SRT | HRRN | Feedback |
|----------------------------|---|---|---|-----------------------------|--|-------------------------------|
| Selection function | $\max [w]$ | constant | $\min [s]$ | $\min [s - e]$ | $\max \left\{ \frac{w + s_0}{s}, 0 \right\}$ | (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 435 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

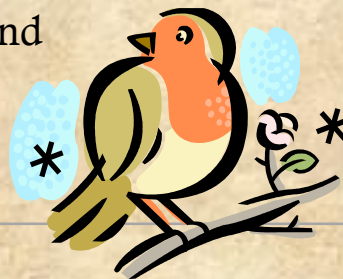
First-Come-First-Served (FCFS)

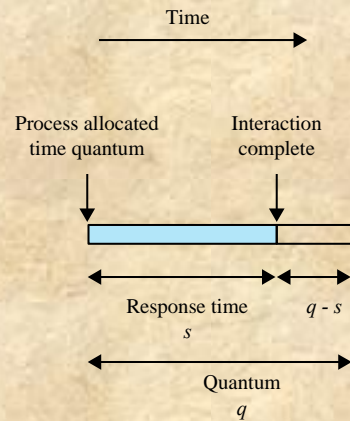
- Simplest scheduling policy
- Also known as first-in-first-out (FIFO) or a strict queuing scheme
- When the currently running process ceases to execute, the process that has been in the Ready queue the longest is selected for running
- Performs much better for long processes than short ones
- Tends to favor processor-bound processes over I/O-bound processes
- May result in inefficient use of both the processor and the I/O devices



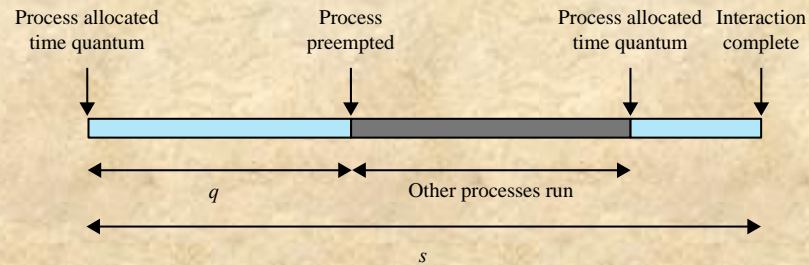
Round Robin

- Uses preemption based on a clock to reduce the penalty that short jobs suffer with FCFS
- Also known as **time slicing** because each process is given a slice of time before being preempted
- The principal design issue is the length of the time quantum, or slice, to be used
 - if very short, short processes will move through the system relatively quickly
 - there is processing overhead involved in handling the clock interrupt and performing the scheduling and dispatching function
- 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
 - processor-bound processes tend to receive an unfair portion of processor time
 - unfair for I/O-bound processes and inefficient use of I/O devices
 - an increase in the variance of response time





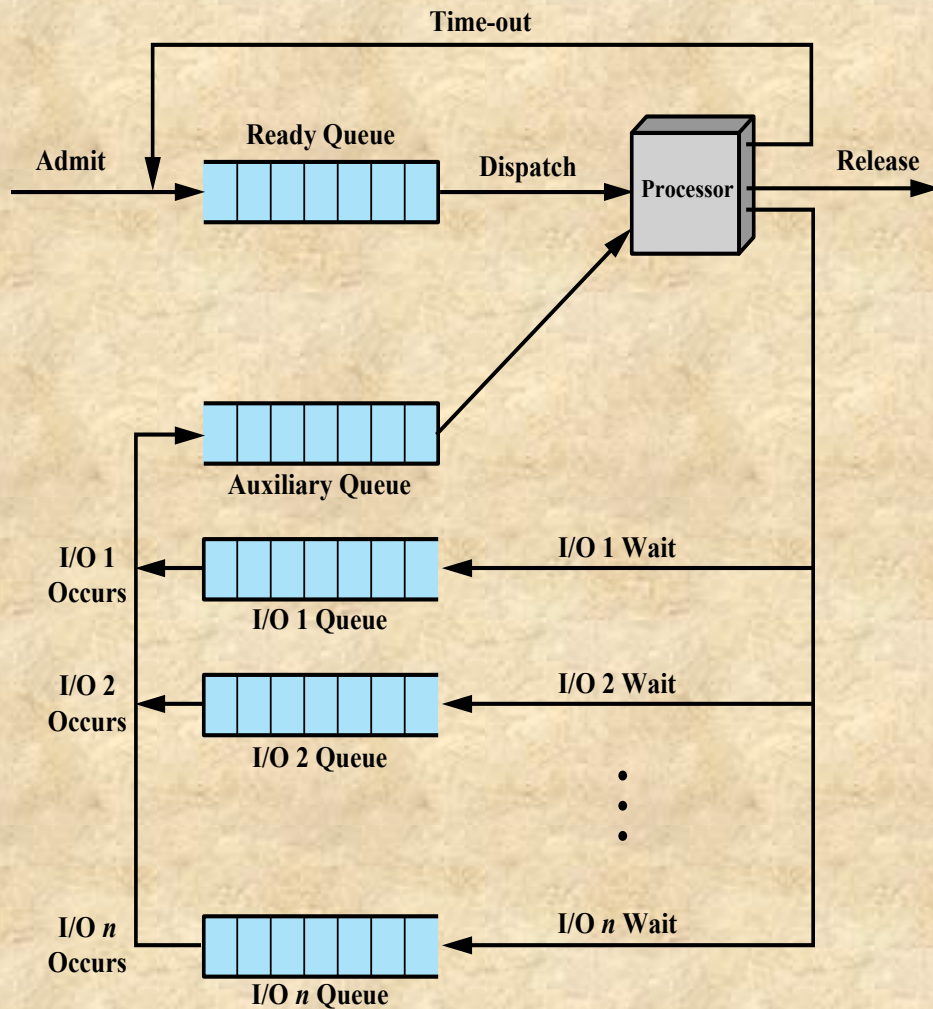
(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

- q should be slightly greater than the time required for a typical interaction or process function
- If it is less, then most processes will require at least two time quanta

Figure 9.6 Effect of Size of Preemption Time Quantum



- An FCFS auxiliary queue to which processes are moved after being released from an I/O block
- Processes in the auxiliary queue get preference over those in the main ready queue
- When a process is dispatched from the auxiliary queue, it runs no longer than a time equal to the basic time quantum minus the total time spent running since it was last selected from the main ready queue
- Superior to round robin in terms of fairness

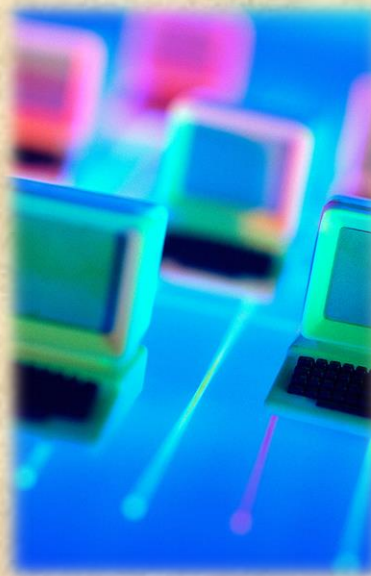
Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

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

Shortest Remaining Time (SRT)

- A preemptive version of SPN
- The scheduler always chooses the process that has the shortest expected remaining processing time
- A 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 the ready process with the greatest value of the response 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

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

Feedback

- A preemptive scheduling based on a dynamic priority mechanism
- Demoted to the next lower-priority queue whenever being preempted
- Newer, shorter processes are favored over older, longer processes
- Within each queue, FCFS
 - in the lowest-priority queue, RR
- When the process blocks or is preempted, feeds it back into one of several priority queues
- The turnaround time of longer processes can stretch out alarmingly, even starvation may occur
 - Set $q = 2^i$ for RQ_i
 - Promote a process to a higher-priority queue after it spends a certain amount of time waiting for service in its current queue

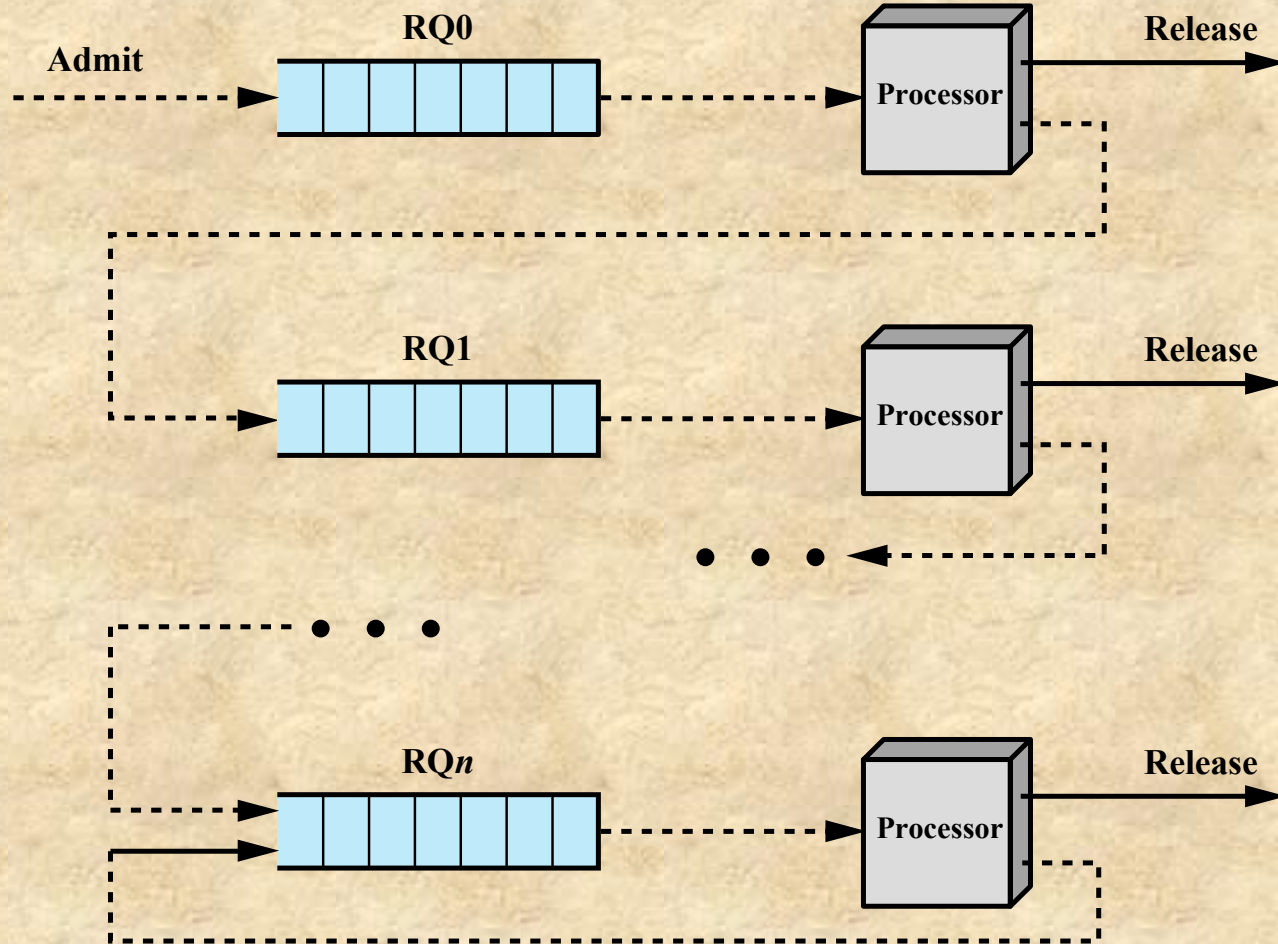


Figure 9.10 Feedback Scheduling

| 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

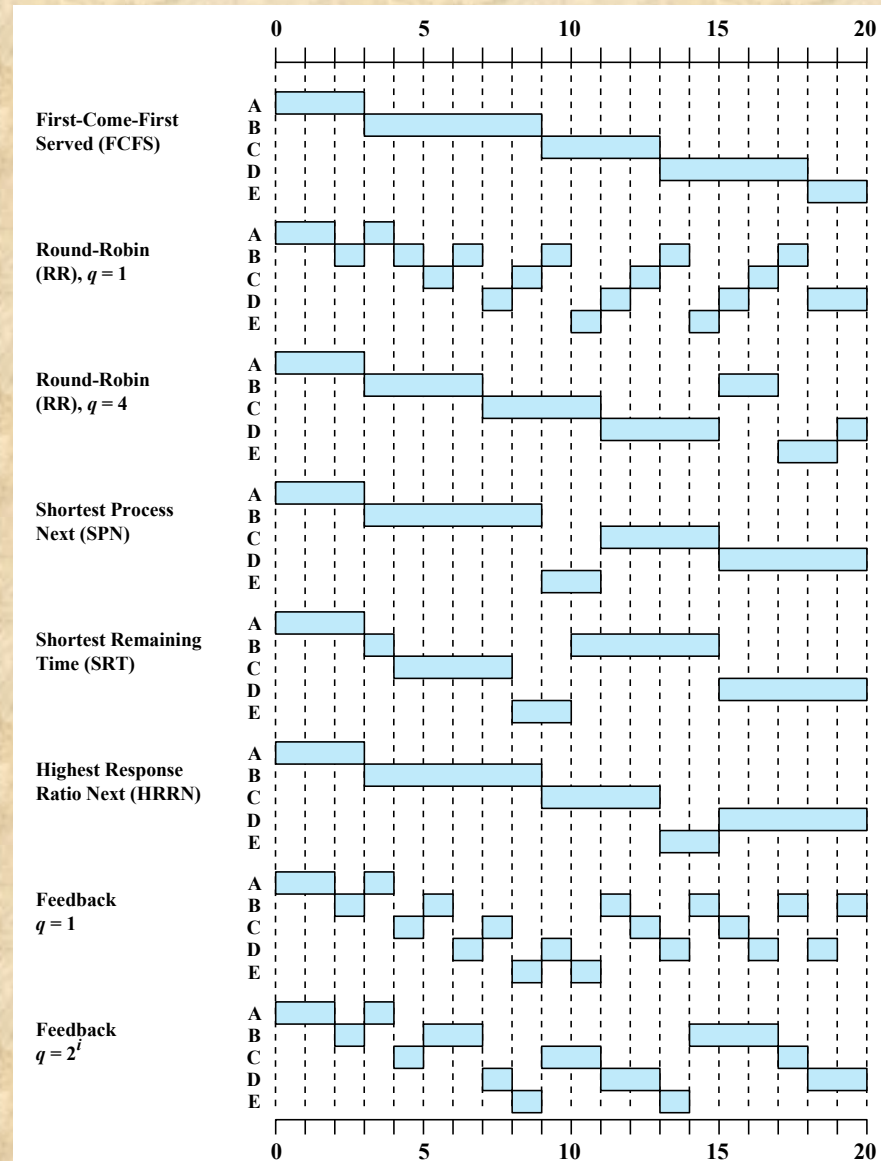


Figure 9.5 A Comparison of Scheduling Policies

| 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 | |
| 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 438 in textbook)

Performance Comparison

- No definitive comparisons is possible, since relative performance will depend on a variety of factors:
 - the probability distribution of service times of the various processes
 - the efficiency of the scheduling and context switching mechanisms
 - the nature of the I/O demand and the performance of the I/O subsystem
- Any scheduling discipline that chooses the next item to be served independent of service time obeys the relationship, assuming that Poisson arrivals and exponential service times:

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

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

ρ = processor utilization

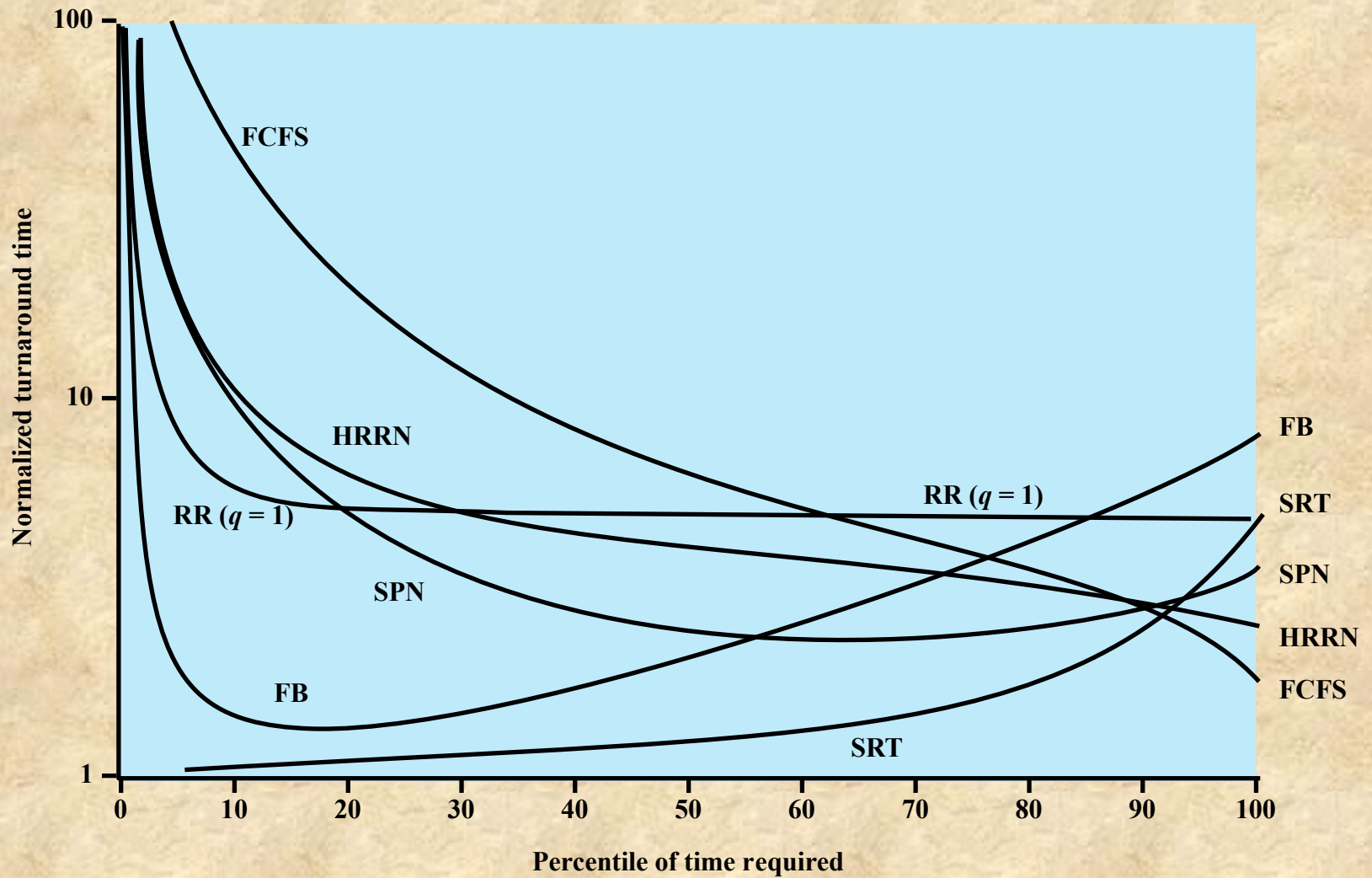


Figure 9.14 Simulation Results for Normalized Turnaround Time

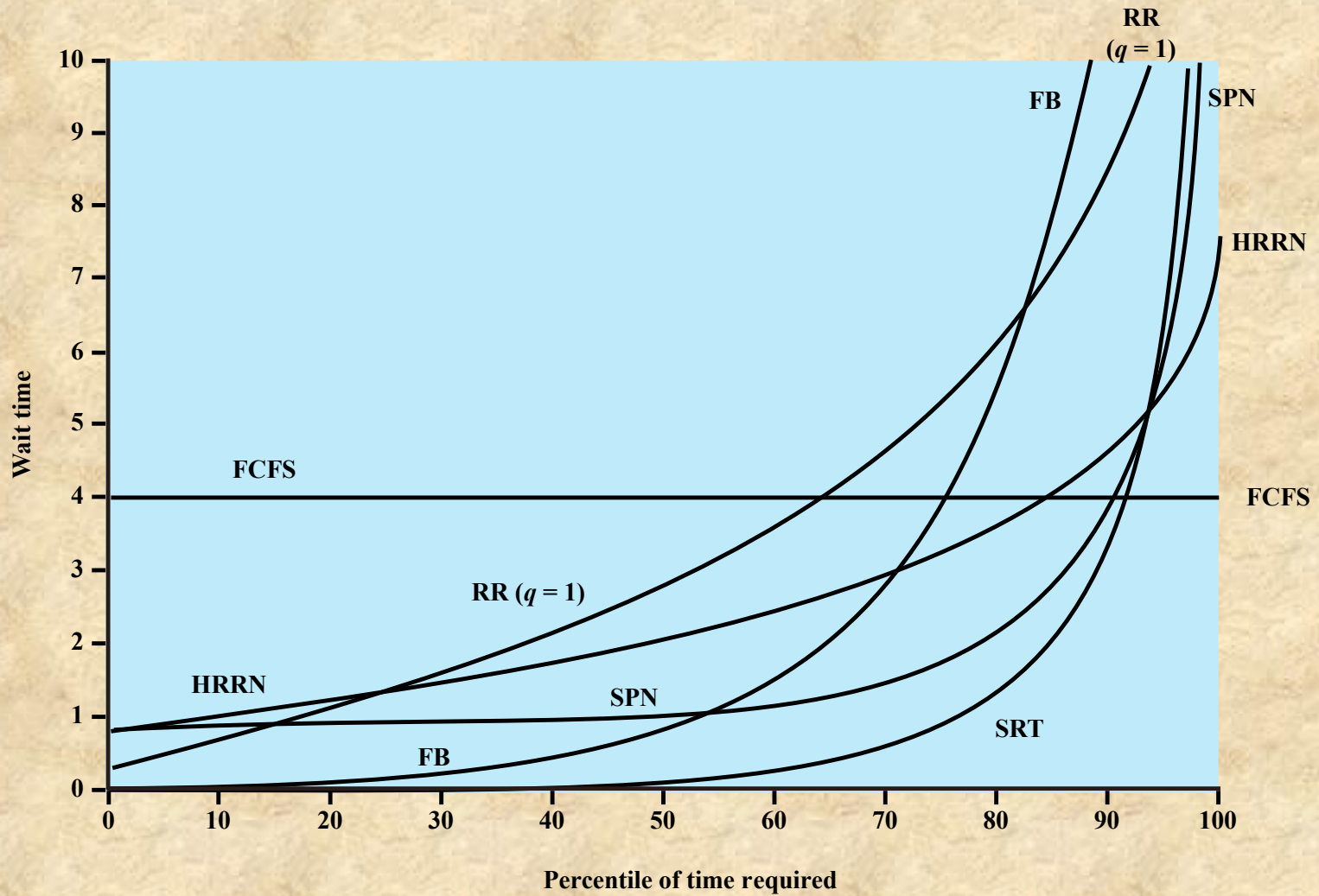


Figure 9.15 Simulation Results for Waiting Time

Simulation Modeling

- Normalized turnaround time
 - FCFS is very unfavorable, with one-third of the processes having a normalized TT > 10 times the service time
 - Except for the shortest processes, RR treats all fairly
 - SPN performs better than RR, except for the shortest processes
 - SRT performs better than SPN except for the longest 7% of all processes
- FCFS favors long processes
- SPN favors short ones
- HRRN compromises two effects
- FB performs quite well for short processes

Fair-Share Scheduling

- Scheduling decisions based on the process sets rather than how a particular process performs
- The 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



Fair-Share Scheduling

- Consider the execution history of a related group of processes, along with the individual execution history of each process
- Scheduling is done on the basis of priority
 - underlying priority of a process
 - its recent process usage
 - recent process usage of its group

Fair-Share Scheduling

- The formulas below apply for process j in group k

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

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

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

where

$CPU_j(i)$ = measure of processor utilization by process j through interval i

$GCPU_k(i)$ = measure of processor utilization of group k 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

W_k = weighting assigned to group k , with the constraint that $0 < W_k \leq 1$

and $\sum_k W_k = 1$

- The greater the weight assigned to the group, the less its utilization will affect its priority

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

Colored rectangle represents executing process

- Group1 = {process A}
- Group2 = {process B, process C}
- Each group's weight = 0.5
- Base priority = 60
- The processor usage field is incremented on every interrupt
- Once per second, priorities are recalculated

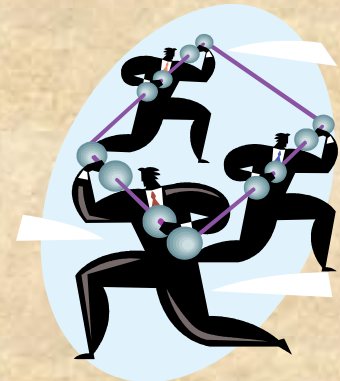


Figure 9.16 Example of Fair Share Scheduler—Three Processes, Two Groups

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 using round robin within each of the priority queues
- Makes use of one-second preemption
- Priority is based on process type and execution history and recomputed once per second

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

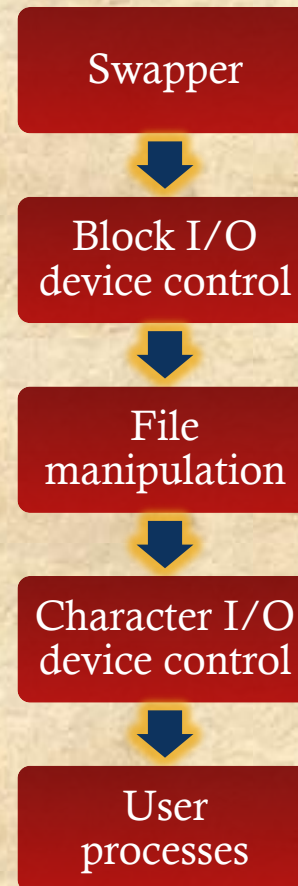
- The purpose of the base priority is to divide all processes into fixed bands of priority levels
- The CPU and $nice$ components are restricted to prevent a process from migrating out of its assigned band (assigned by the base priority level)

Scheduling Formula

- The purpose of the base priority is to divide all processes into fixed bands of priority levels.
- The *CPU* and *nice* components are restricted to prevent a process from migrating out of its assigned band (assigned by the base priority level).

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:
- Should provide the most efficient use of the I/O devices





| 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