Uniprocessor Scheduling

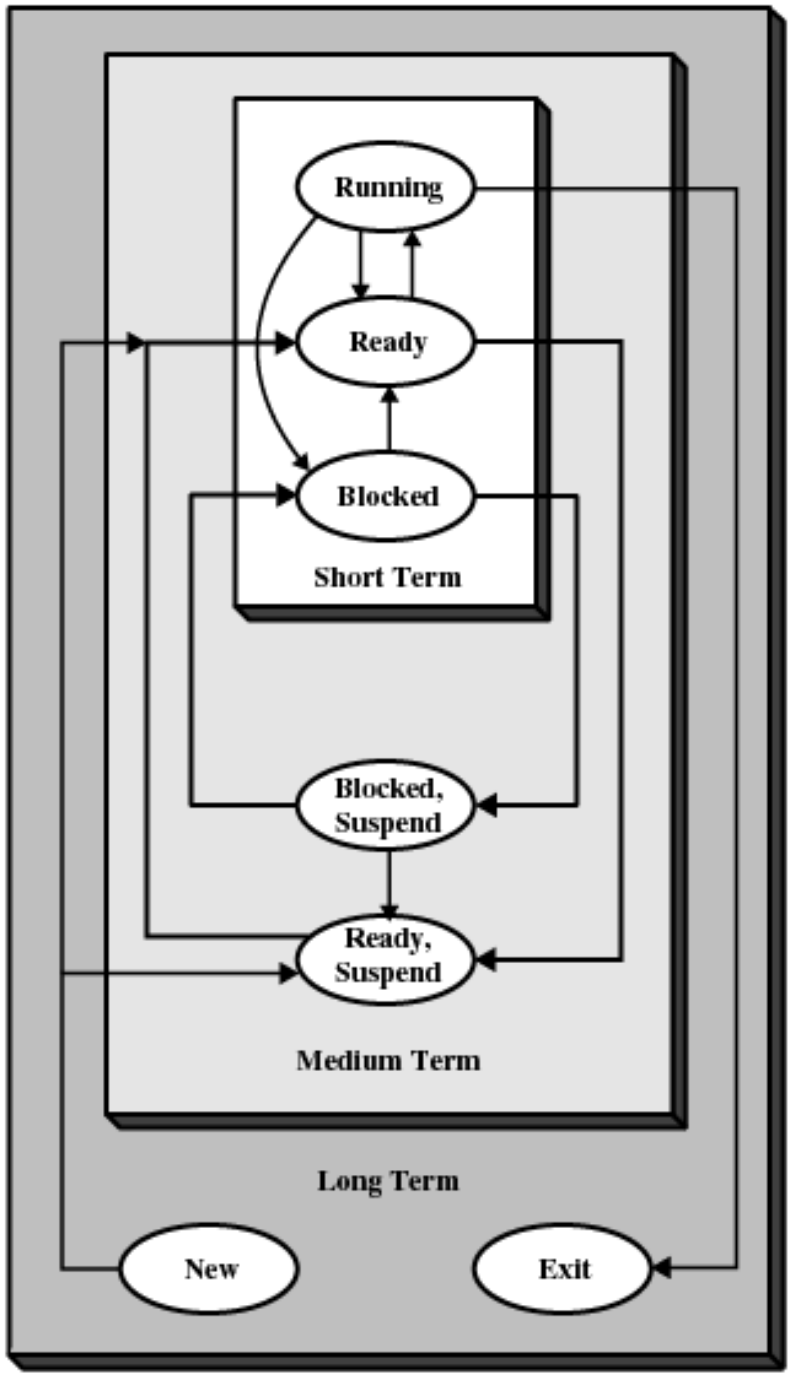
Chapter 9

Aim of Scheduling

- Assign processes to be executed by the processor(s)
- Response time
- Throughput
- Processor efficiency

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


Running
Ready
Blocked
Short Term
Blocked,
Suspend
Ready,
Suspend
Medium Term
Long Term
New
Exit

Long-Term Scheduling

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
- More processes, smaller percentage of time each process is executed

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
  - Clock interrupts
  - I/O interrupts
  - Operating system calls
  - Signals

Dispatcher Criteria

- User-oriented
  - Response Time
    - Elapsed time between the submission of a request until there is output.
- System-oriented
  - Effective and efficient utilization of the processor
- Performance-related
  - Quantitative
  - Measurable such as response time and throughput
- Predictability (same behavior in time)

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

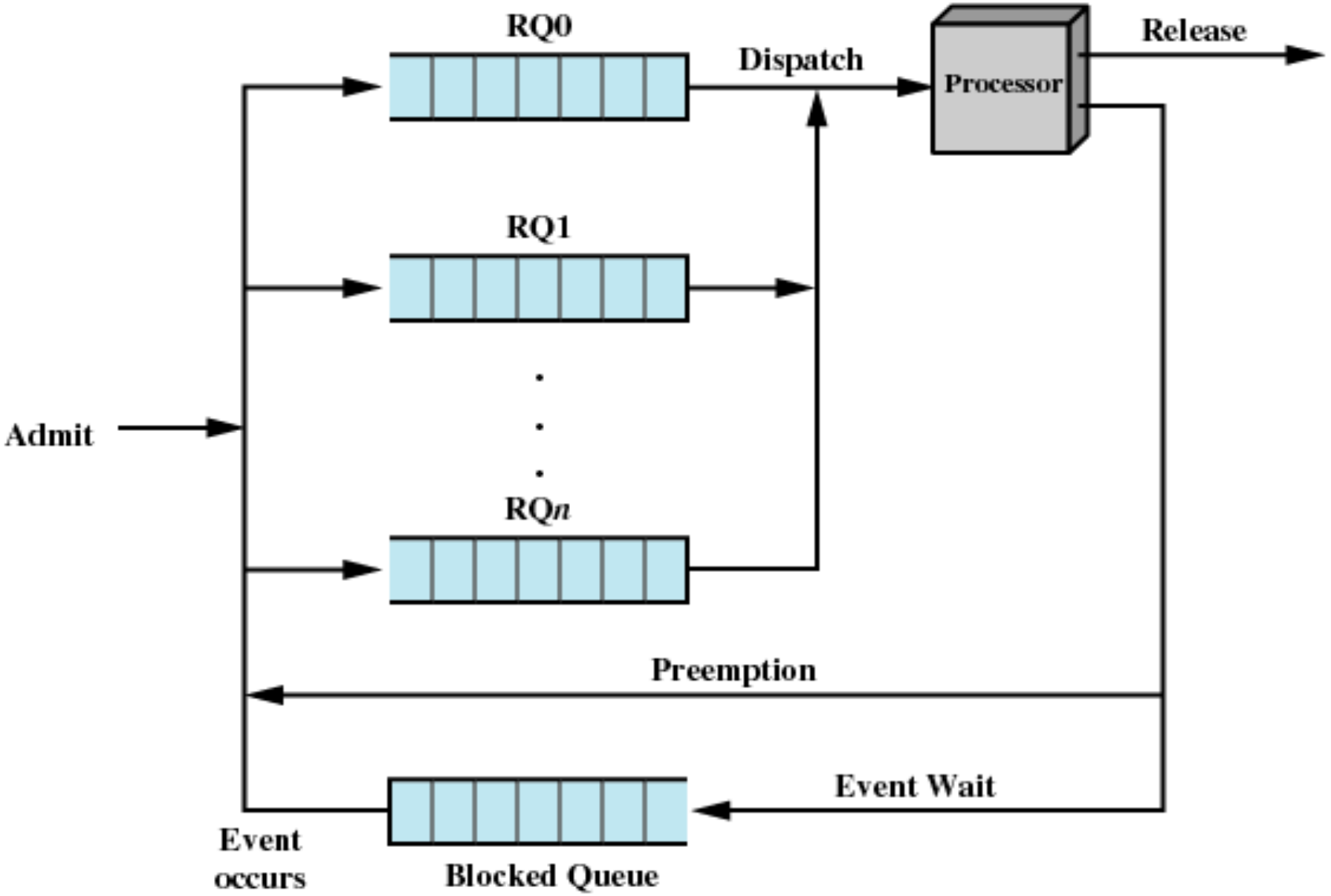
Fairness In the absence of guidance from the user or other system-supplied guidance, processes should be treated the same, and no process should suffer starvation.

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.

Priorities

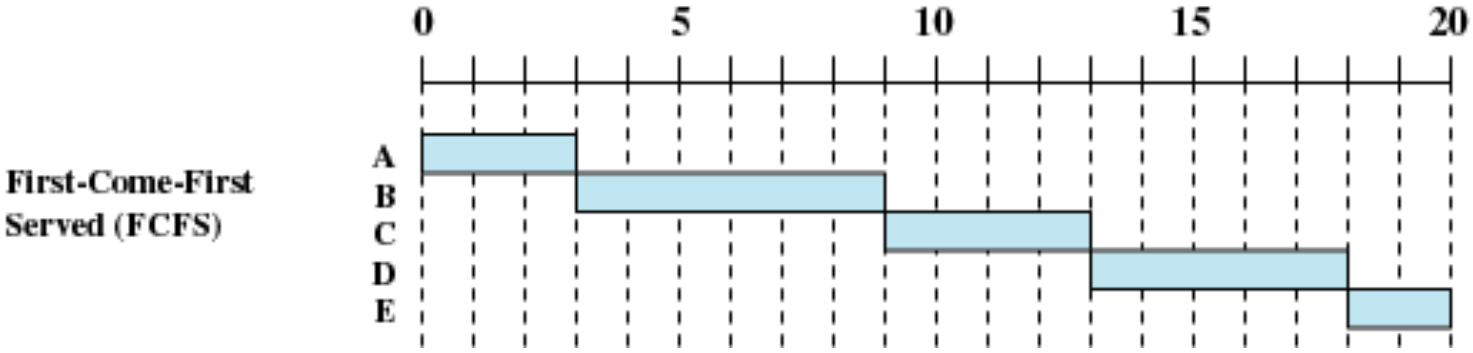
- Scheduler will always choose a process of higher priority over one of lower priority
- Have multiple ready queues to represent each level of priority
- Lower-priority may suffer starvation
  - Allow a process to change its priority based on its age or execution history


RQ0
Dispatch
Processor
Release
RQ1
Admit
RQn
Preemption
Event Wait
Event
occurs
Blocked Queue

Decision Mode

- 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 the Ready state by the operating system
  - Allows for better service since any one process cannot monopolize the processor for very long

First-Come-First-Served (FCFS)



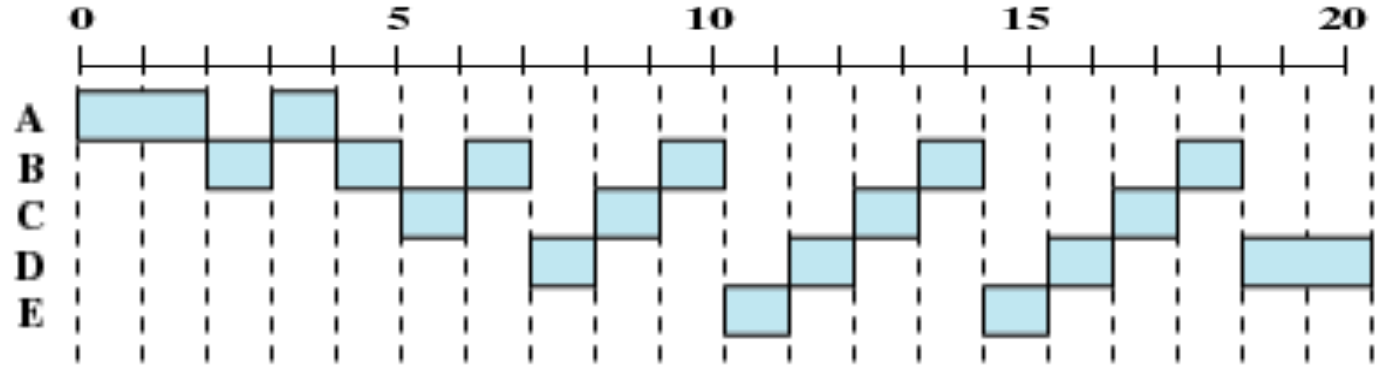
- Each process joins the Ready queue
- When the current process ceases to execute, the oldest process in the Ready queue is selected

First-Come-First-Served (FCFS)

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

Round-Robin

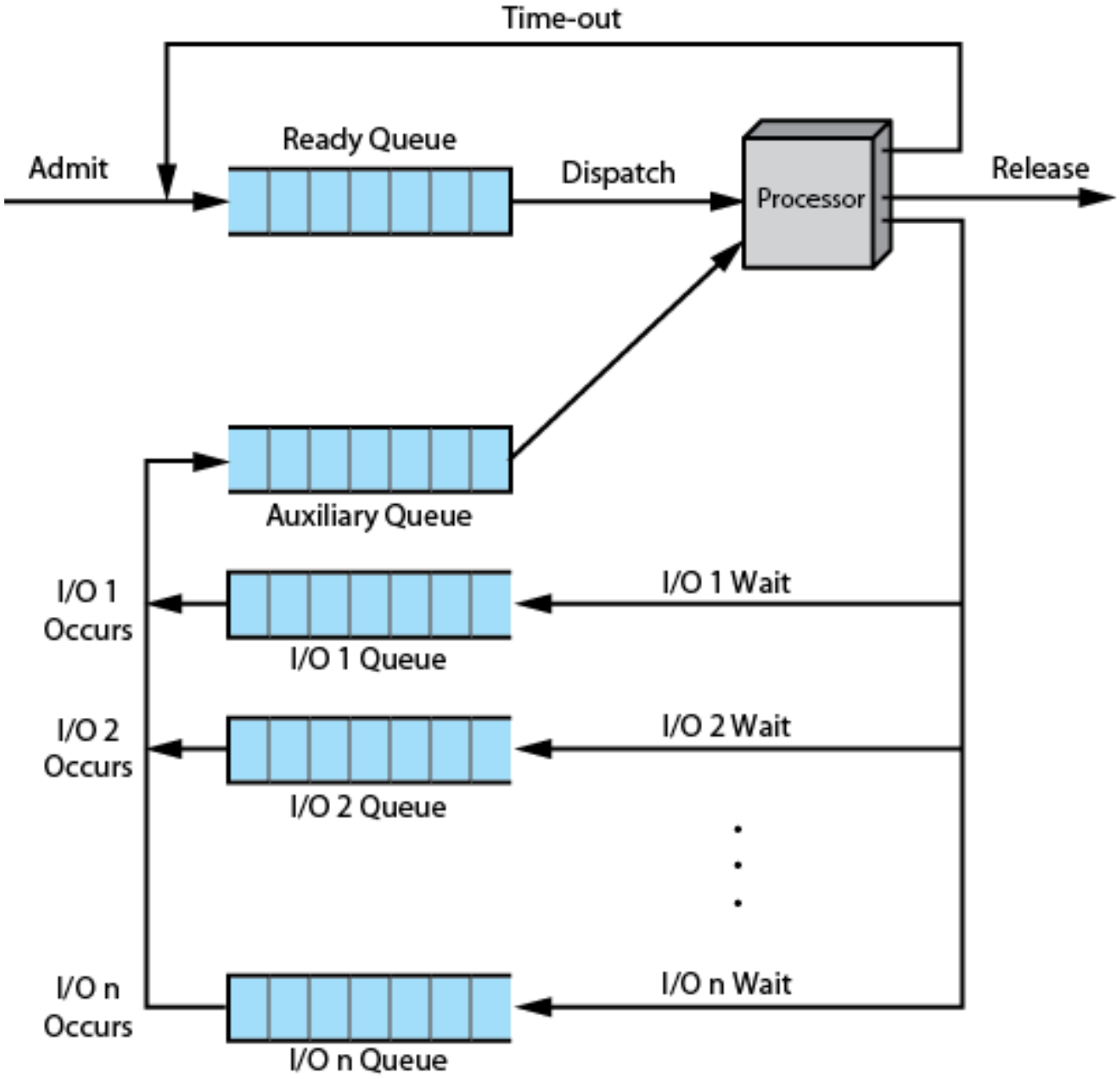
Round-Robin
(RR), $q = 1$



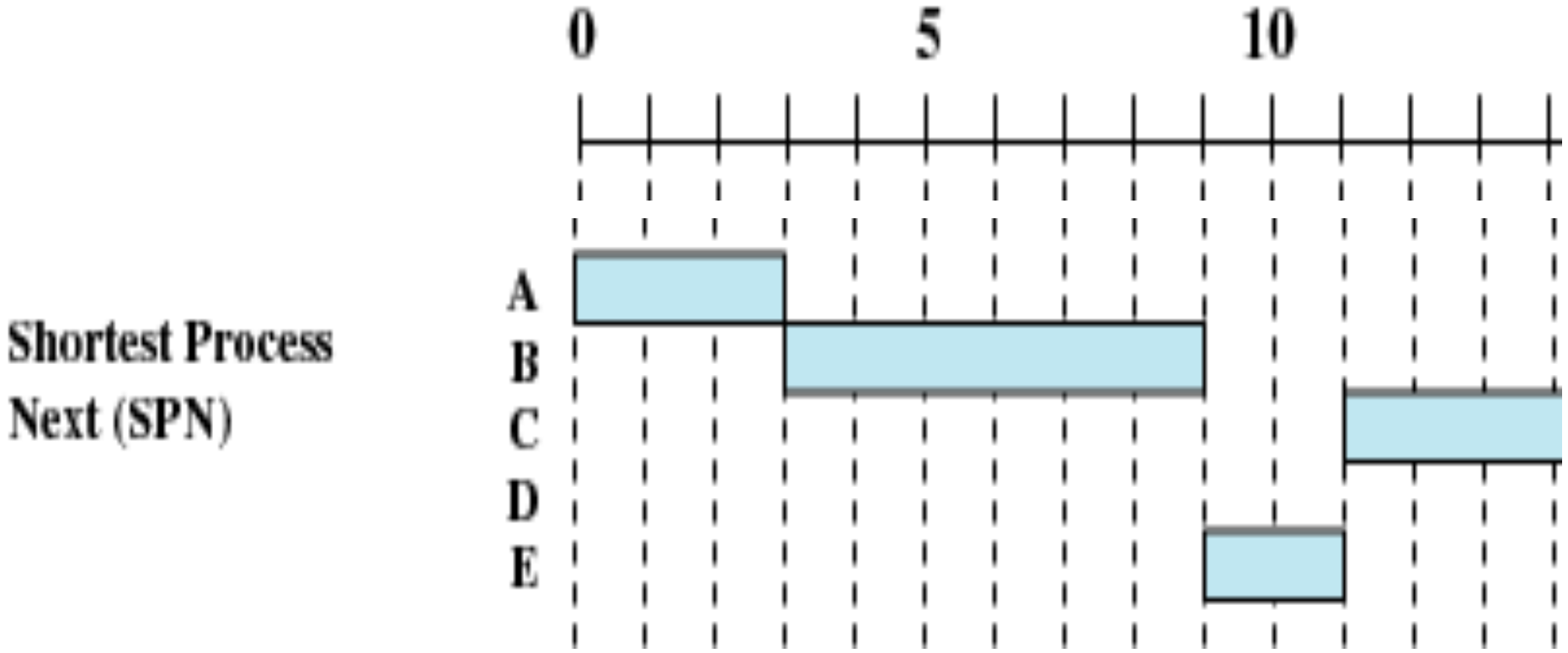
- Uses preemption based on a clock
- An amount of time is determined that allows each process to use the processor for that length of time

Round-Robin (Time Slicing)

- Clock interrupt is generated at periodic intervals
- When an interrupt occurs, the currently running process is placed in the ready queue
  - Next ready job is selected
- Better for processor-bound processes
- Worse for I/O bound processes
- => Virtual Round Robin (Aux. Queue)
  - Runtime = timeslice – run of last time


Time-out
Admit
Ready Queue
Dispatch
Processor
Release
Auxiliary Queue
I/O 1
Occurs
I/O 1 Wait
I/O 1 Queue
I/O 2
Occurs
I/O 2 Wait
I/O 2 Queue
I/O n
Occurs
I/O n Wait
I/O n Queue

Shortest Process Next

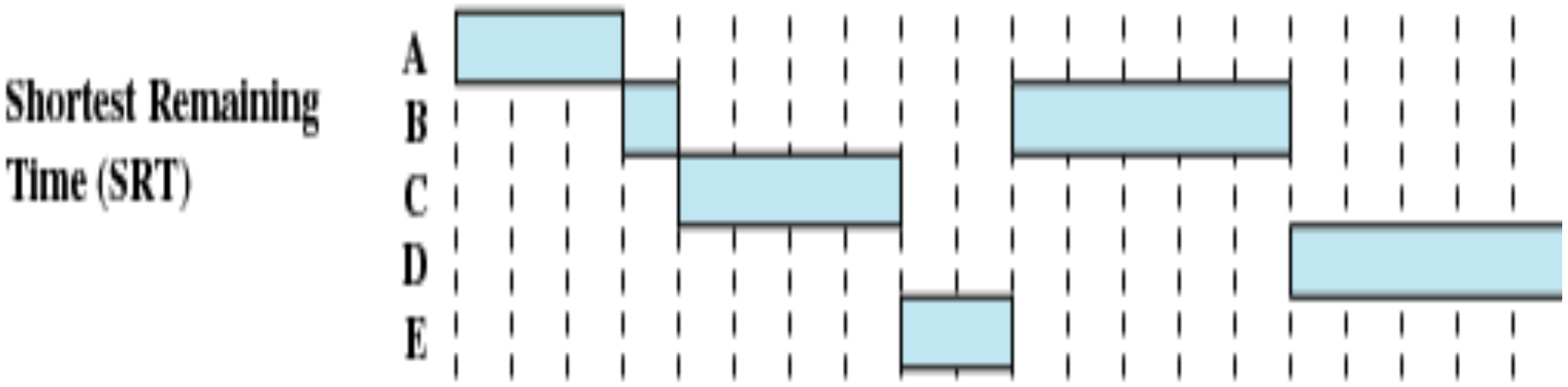


- Short process jumps ahead of longer processes

Shortest Process Next

- No preemption
- Predictability of longer processes is reduced
- If estimated time for process not correct, the operating system may abort it
- Possibility of starvation for longer processes
- => preemptive version next

Shortest Remaining Time



- Preemptive version of shortest process next policy
- Must estimate processing time
- Must keep track of elapsing time
- Less interrupts w.r.t. Round Robin

Highest Response Ratio Next (HRRN)

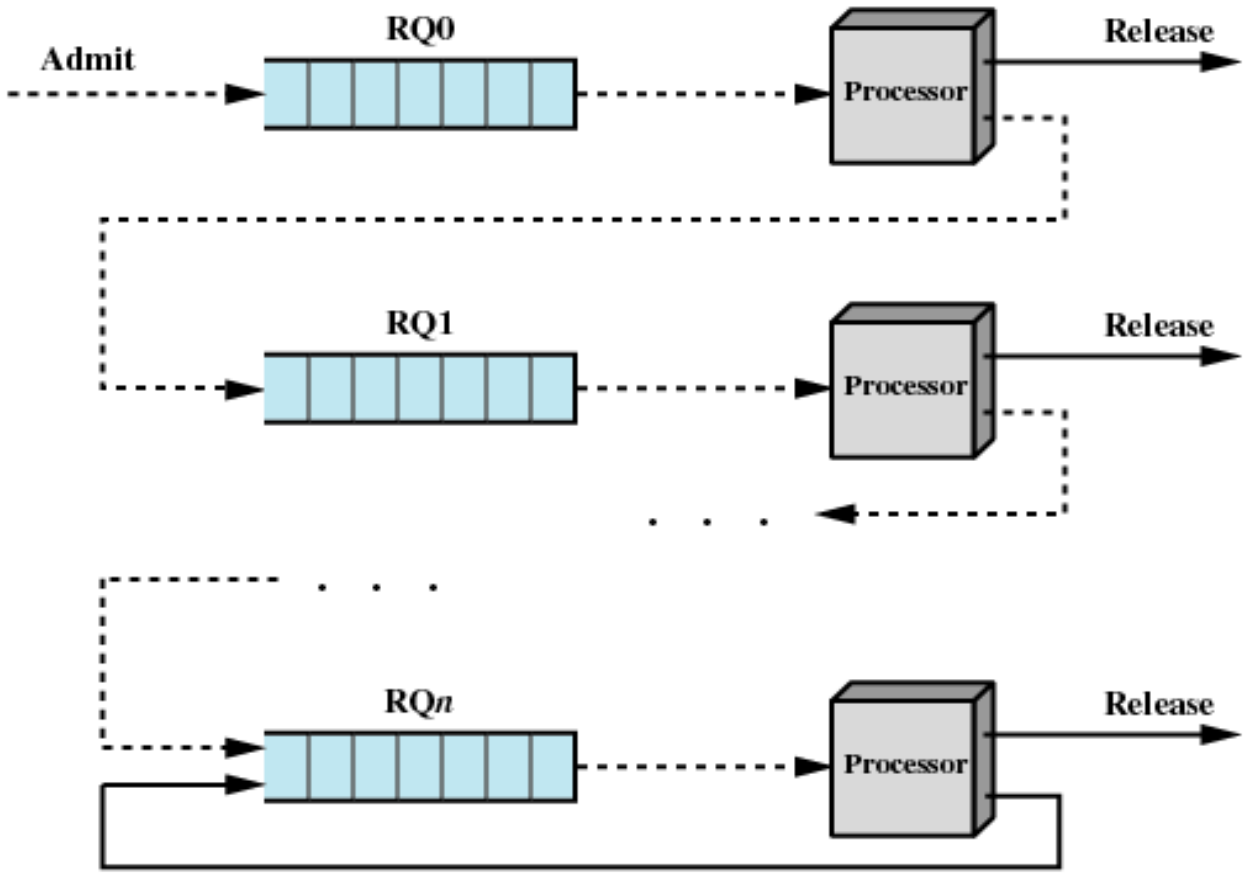
- No preemption
- W = waiting time
- S = expected service time (statistics)
- Choose next process with the greatest ratio

$$R = (W + S)/S$$

- Fair to short (small S) and aged jobs (large W)

Feedback (Preemptive)

- Difficult to know remaining time
- =>Penalize jobs that have been running longer



Feedback (Preemptive)

- Problem for long processes: starvation
- Strategies:
  - Use timeslices of different length for different queues
  - Shorter for upper-level queues
  - Promote a process to an upper-level queue after has waited for some time on a given queue

Traditional UNIX Scheduling (Fair Share)

- Multilevel feedback using round robin within each of the priority queues
- Process preempted after 1 sec running
- Priorities are recomputed every second

  $CPU_j(i) = CPU_j(i-1)/2$

  $P_j(i) = Base_j + CPU_j(i)/2 + nice_j$
- Penalizes processor-bound processes
- Base priority divides all processes into fixed bands of priority levels

Bands

- Decreasing order of priority
  - Swapper
  - Block I/O device control
  - File manipulation
  - Character I/O device control
  - User processes