Uniprocessor Scheduling

types of scheduling in OS

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

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
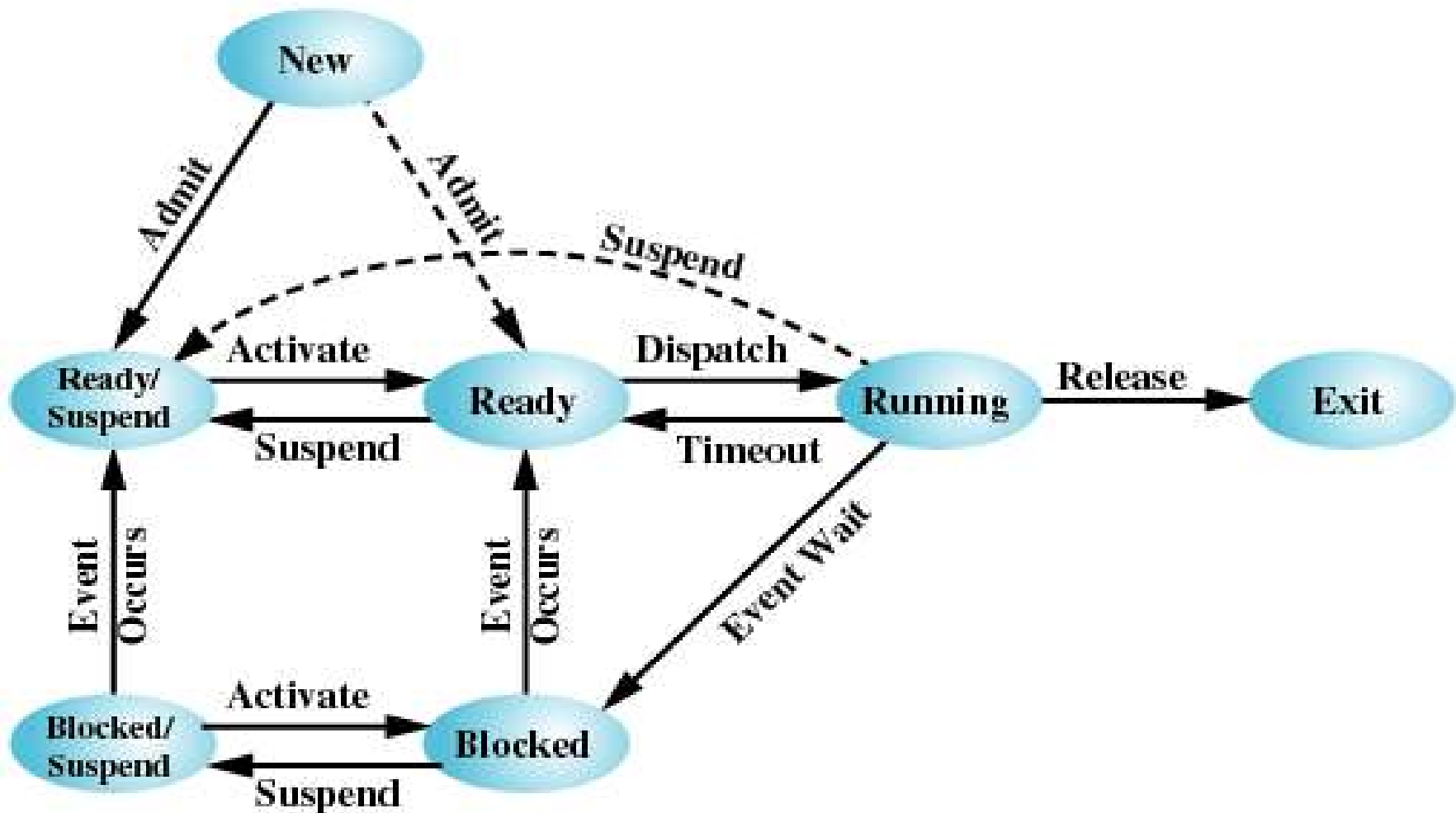
Medium-Term Scheduling

- i.e. swapping
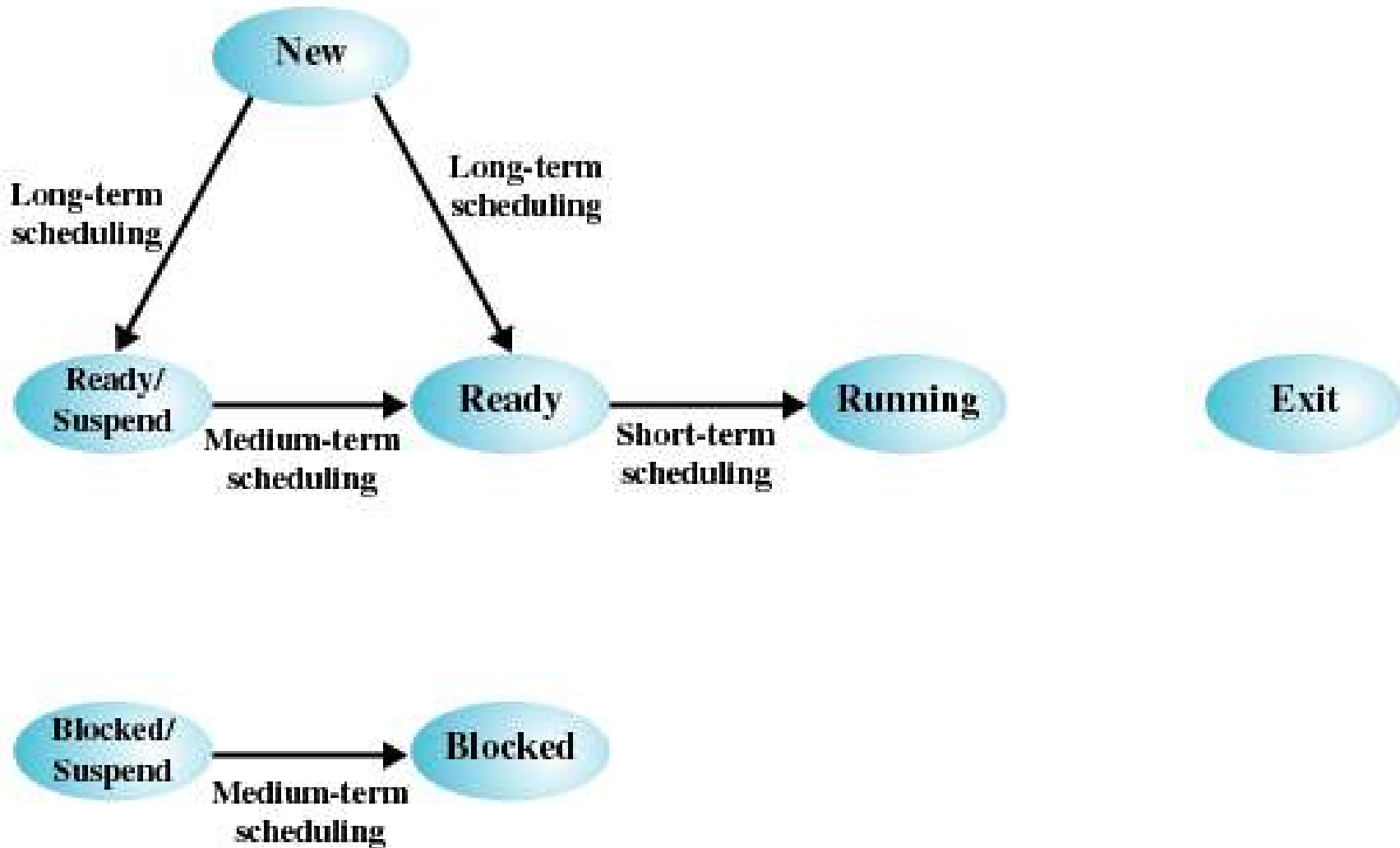- Based on the need to manage the degree of multiprogramming

Short-Term (cpu) Scheduling

- a cpu scheduling policy decides for each cpu...
  - which process should be executed

  - how long it will be executed

- implemented in the dispatcher, a.k.a. scheduler

- the scheduler executes very frequently
  - Invoked when an event occurs

    - Timer interrupts (time quantum expired)

    - I/O interrupts
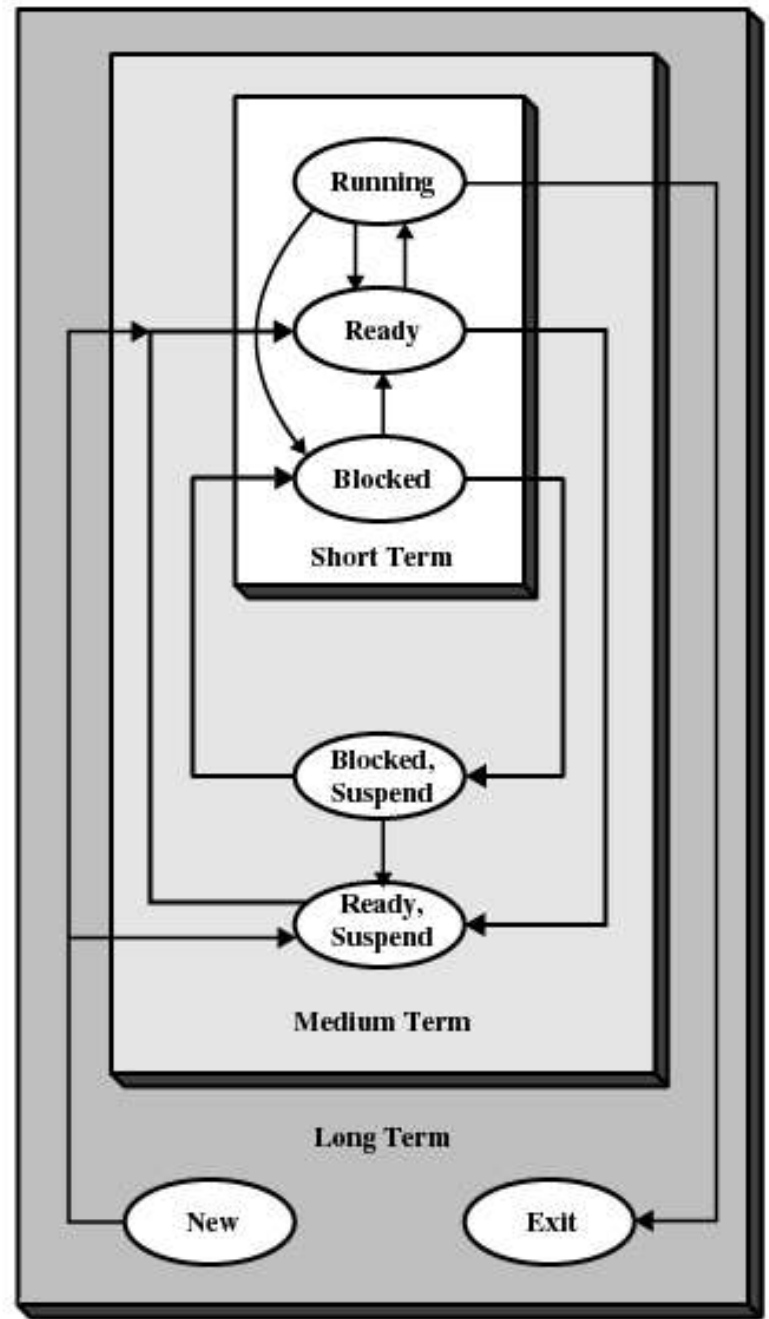
    - traps

    - Operating system calls
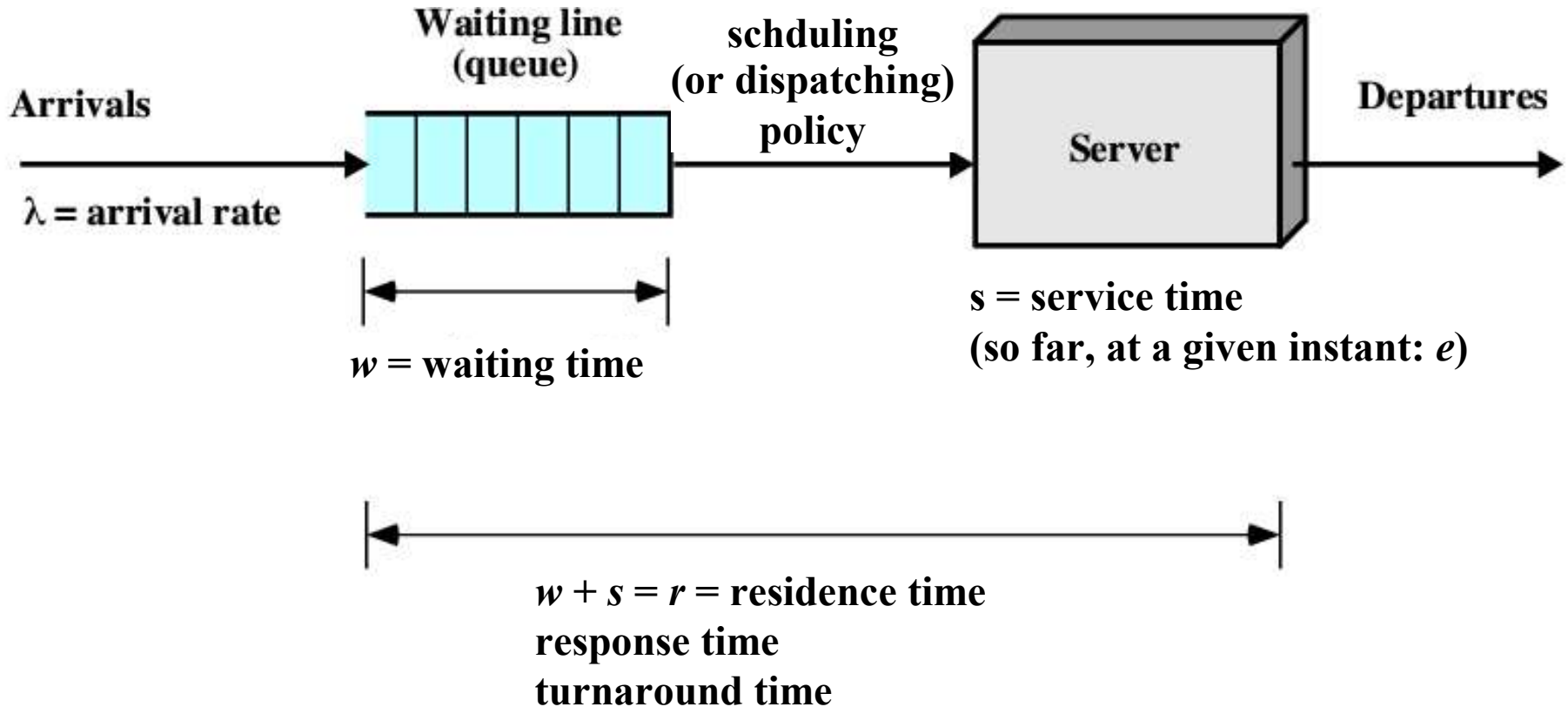
process states

scheduling and process state transitions

New

Long-term scheduling

Long-term scheduling

Ready/ Suspend

Medium-term scheduling

Ready

Short-term scheduling

Running

Exit

Blocked/ Suspend

Medium-term scheduling

Blocked

levels of scheduling

Running

Ready

Blocked

Short Term

Blocked, Suspend

Ready, Suspend

Medium Term

Long Term

New

Exit

8

short term cpu scheduling

queue

Arrivals

λ = arrival rate

Waiting line (queue)

schduling (or dispatching) policy

Server

Departures

w = waiting time

s = service time
(so far, at a given instant: *e*)

w + *s* = *r* = residence time
response time
turnaround time

$$\lambda \leqslant \frac{1}{s}$$ **(statistically)**

for cpu scheduling the server is the cpu

10

Optimality Criteria

- ## User-oriented
  - ### Response Time
    - Elapsed time from submission to begin of the service.
  - ### Normalized Response Time w.r.t. service time, that is $(w+s)/s$

- ## System-oriented
  - ### Effective and efficient utilization of the processor

Optimality Criteria

- Performance-related
  - Quantitative
  - Measurable
    - such as response time and throughput

- other (not performance-related)
  - predictability
  - fairness
  - ecc.

optimality 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.

13

scheduling criteria

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.

scheduling queuing diagram
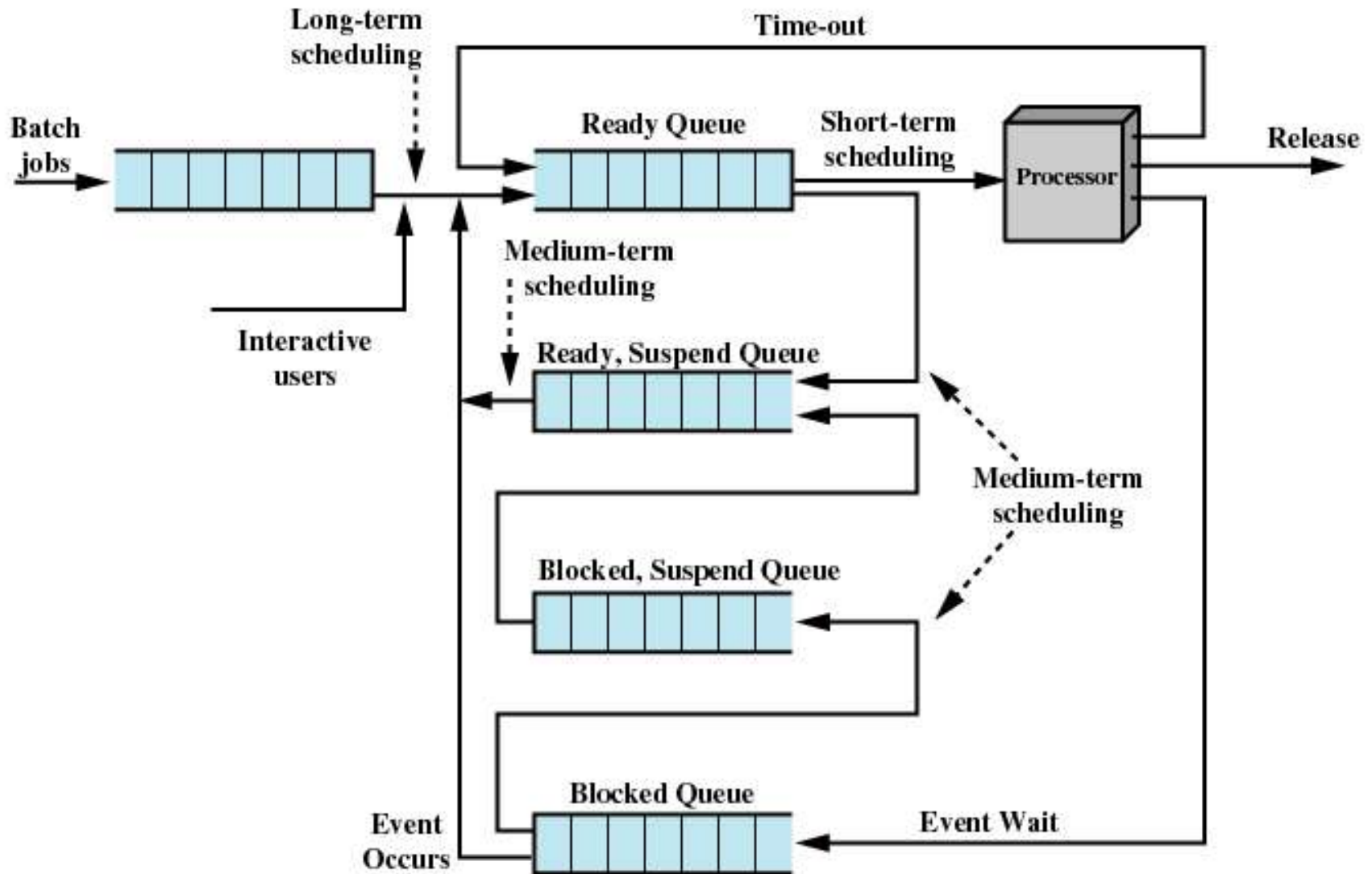
Priorities

- a priority is assigned to each process
- a ready process queue for each priority
- Scheduler will always choose a process of higher priority over one of lower priority
- Lower-priority may suffer starvation
  - Allow a process to change its priority based on its age or execution history

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

priority queuing and preemption

RQ0

Dispatch

Processor

Release

RQ1

RQn

Admit

Preemption

Event Wait

Event occurs

Blocked Queue

18
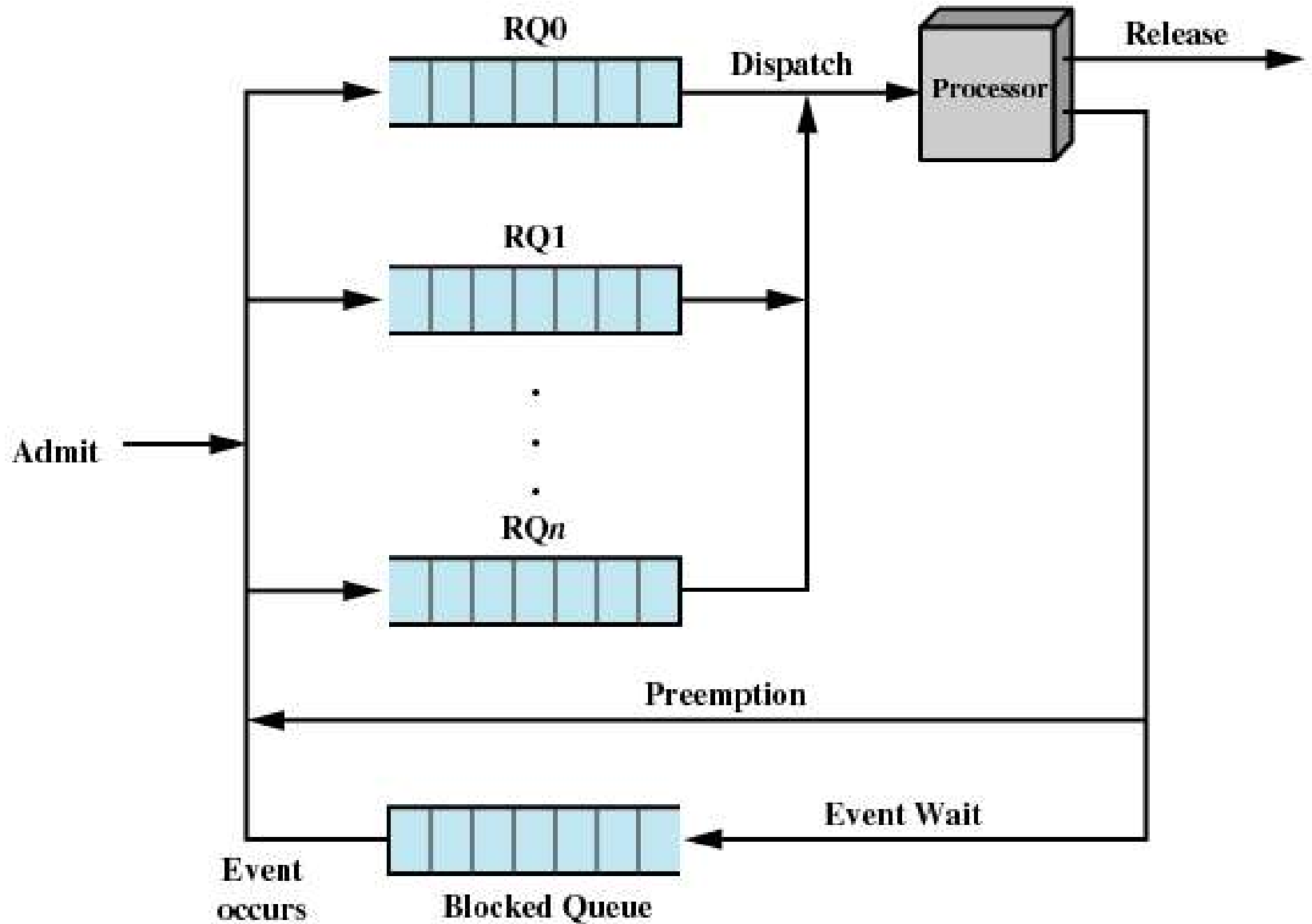
Process Scheduling Example

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

- arrival time: when the process enter the ready queue

- service time: the process virtual time elapsed till the next blocking operation

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

First-Come-First-Served (FCFS)

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2
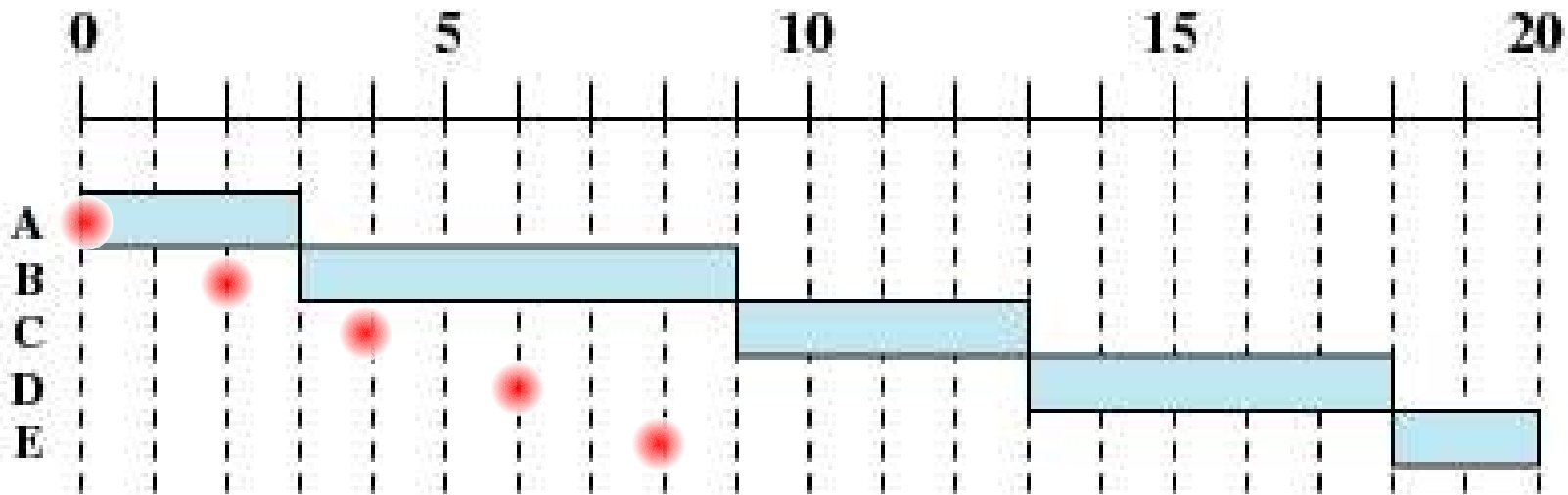
- When the current process ceases to execute, the oldest process in the Ready queue is selected

21

First-Come-First-Served (FCFS)

| Processo	Tempo di arrivo	Tempo di servizio (T_s)	Tempo di inizio	Tempo di fine	Tempo di turnaround (T_r)	T_r / T_s
A	0	1	0	1	1	1
B	1	100	1	101	100	1
C	2	1	101	102	100	100
D	3	100	102	202	199	1.99
Media					100	26

Round-Robin

- Clock interrupt is generated at periodic intervals
- When an interrupt occurs, the currently running process is placed in the read queue (preempted)
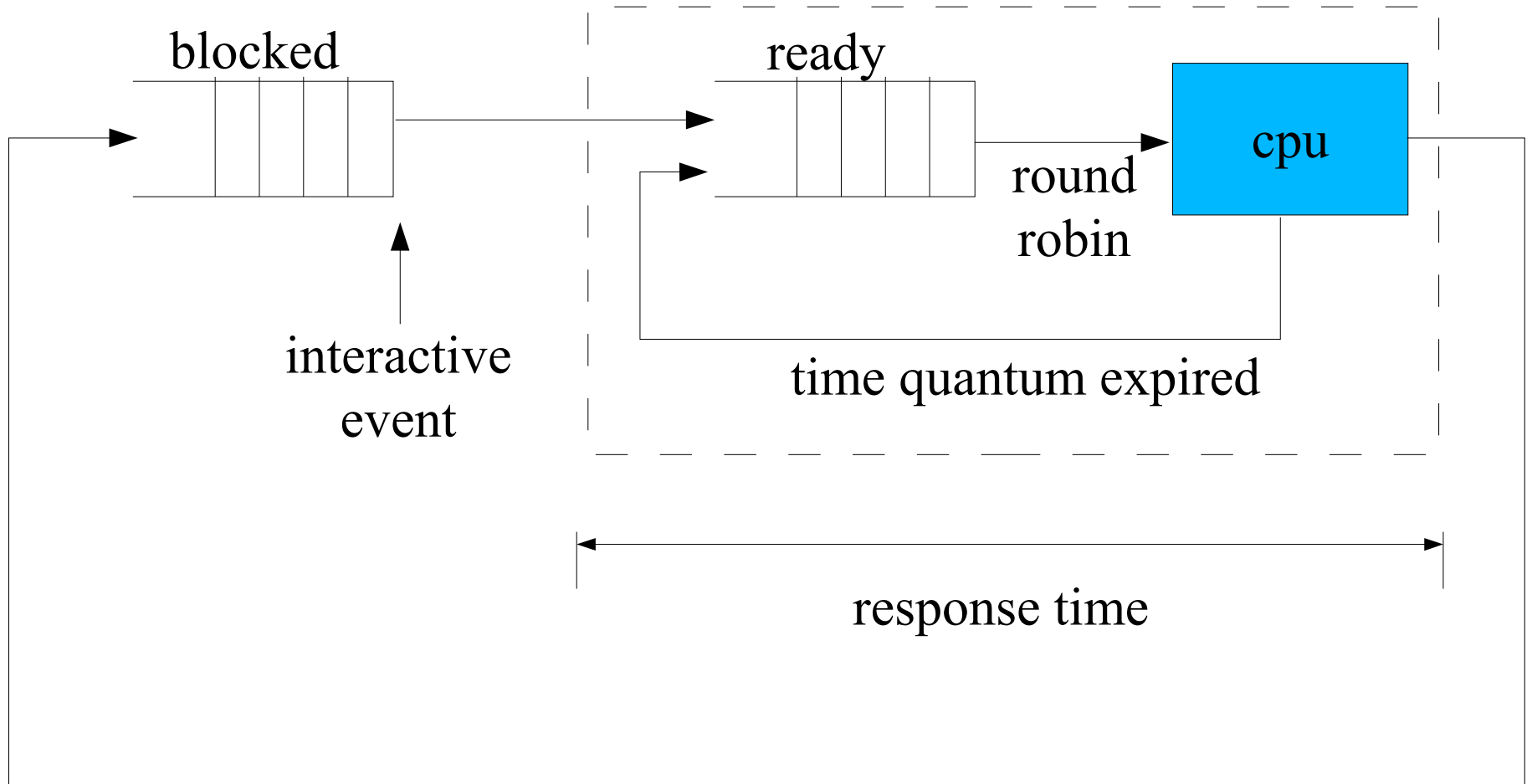 - Next ready job is selected
- a.k.a. time slicing

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

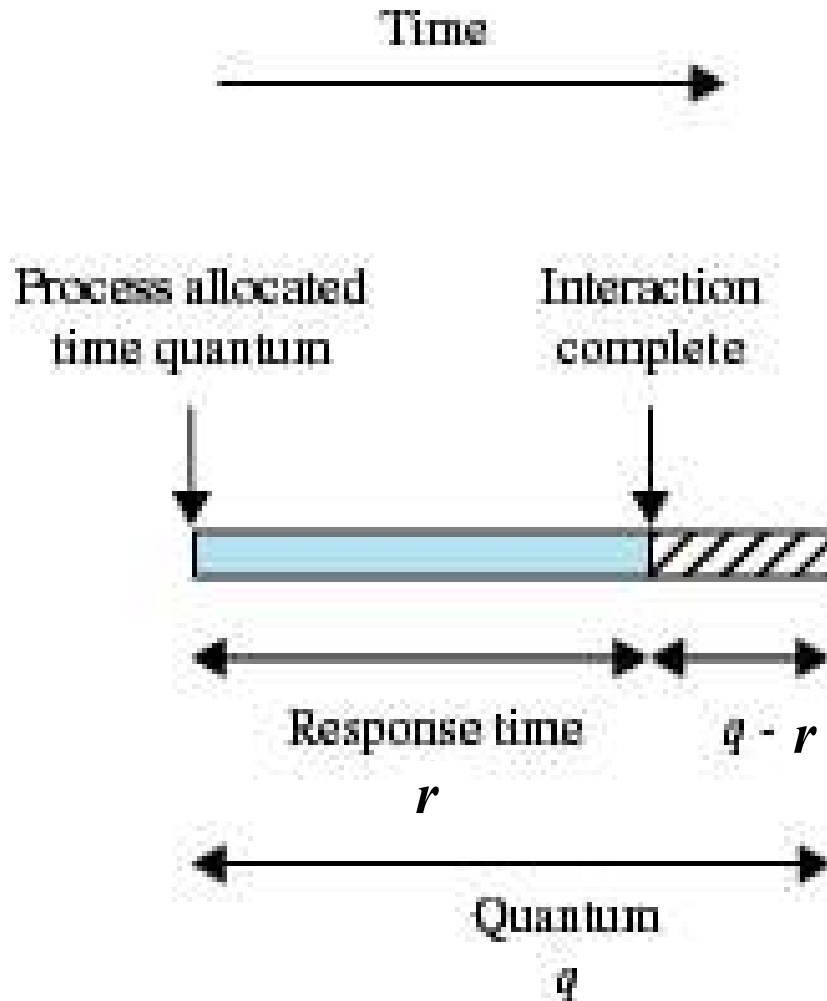| Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2
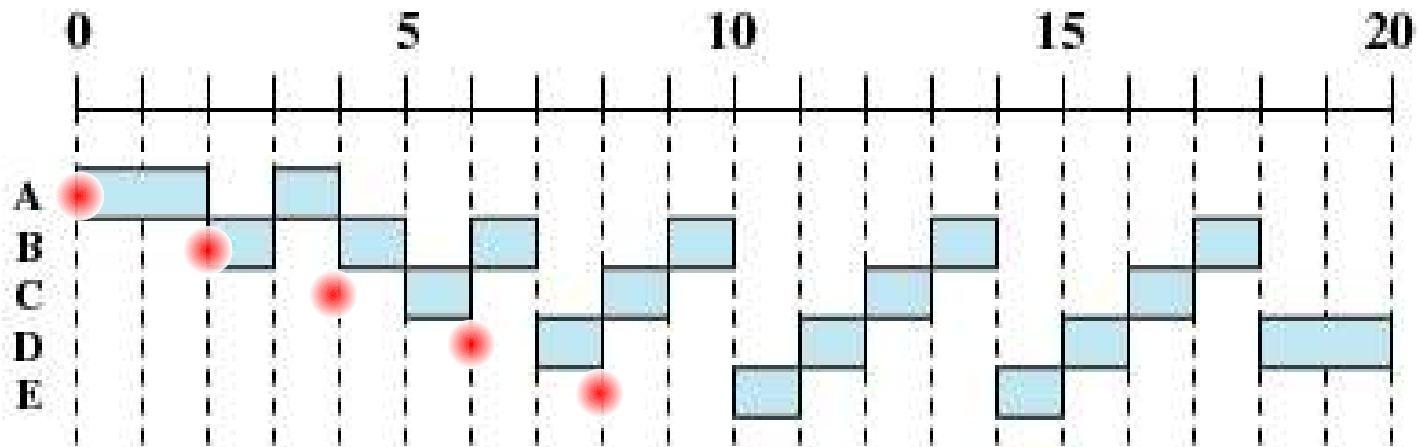
- **preemption** based on a timer
- **time quantum** *q*: each process is allowed to use the processor for the time quantum and then preempted
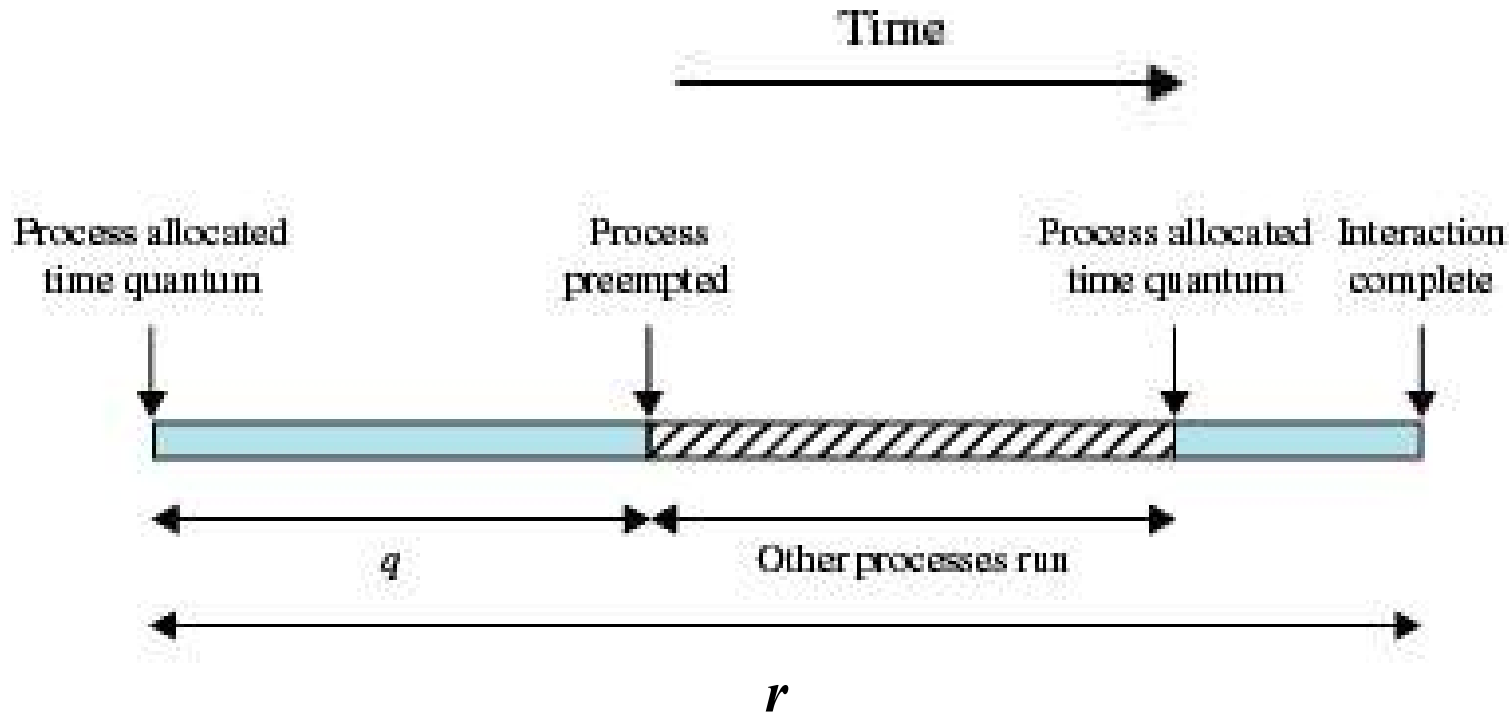
24

RR and queues

blocked

ready

cpu

round
robin

interactive
event

time quantum expired

response time

effect of size of preemption time quantum

Time

Process allocated
time quantum

Interaction
complete

Response time
r

$q - r$

Quantum
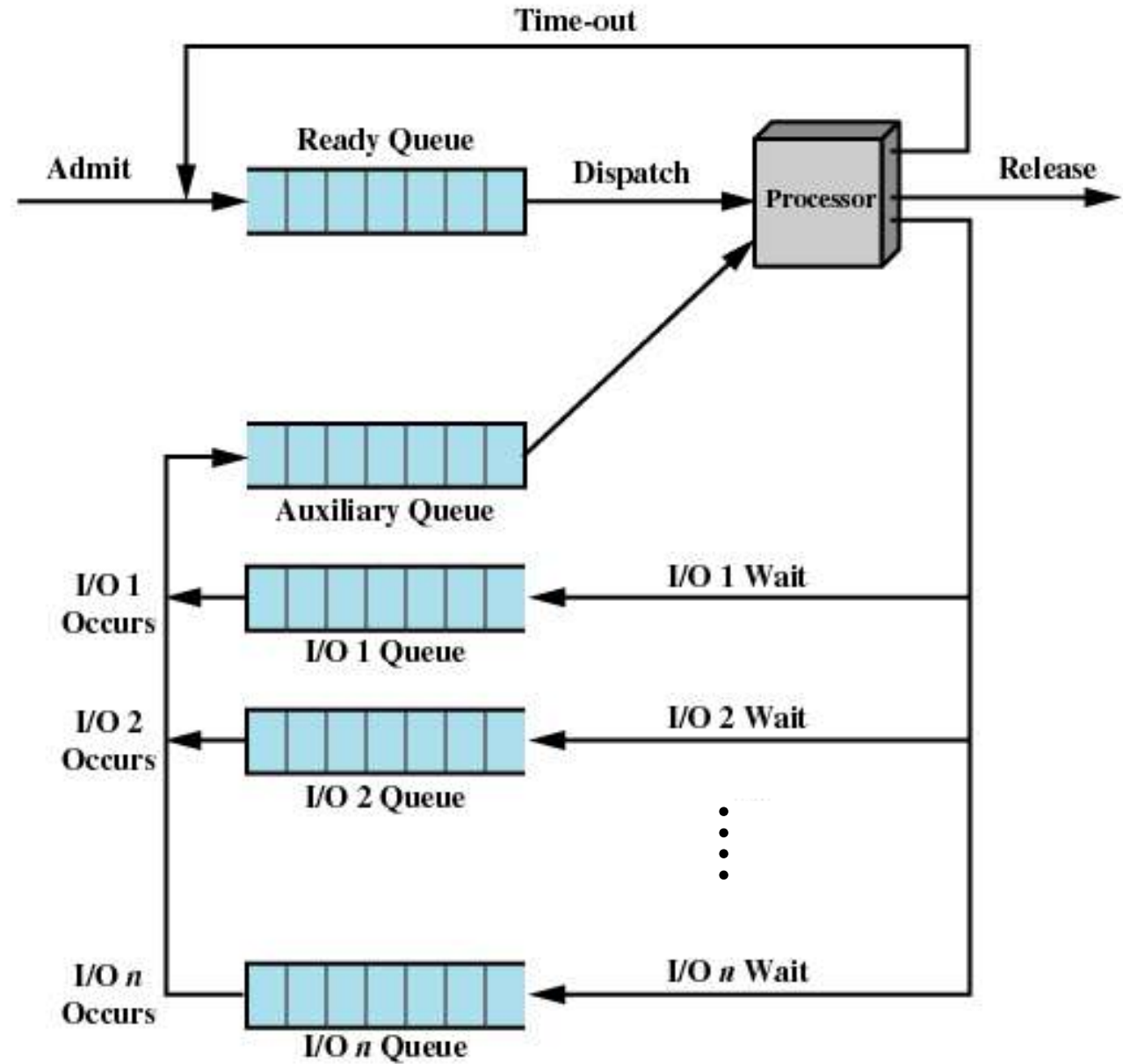q

effect of size of preemption time quantum

(b) Time quantum less than typical interaction

unfairness of RR

- I/O-bound processes usually release cpu before expiration of their quantum

- cpu-bound processes run for the whole quantum

- RR prefers cpu-bound processes

virtual round robin

- aux queue is preferred over ready queue

- processes from aux queue run for their remaining quantum fraction

Time-out

Admit → Ready Queue → Dispatch → Processor → Release

Auxiliary Queue

I/O 1 Occurs ← I/O 1 Queue ← I/O 1 Wait

I/O 2 Occurs ← I/O 2 Queue ← I/O 2 Wait

I/O n Occurs ← I/O n Queue ← I/O n Wait

29

Shortest Process Next (SPN, SJF)

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2
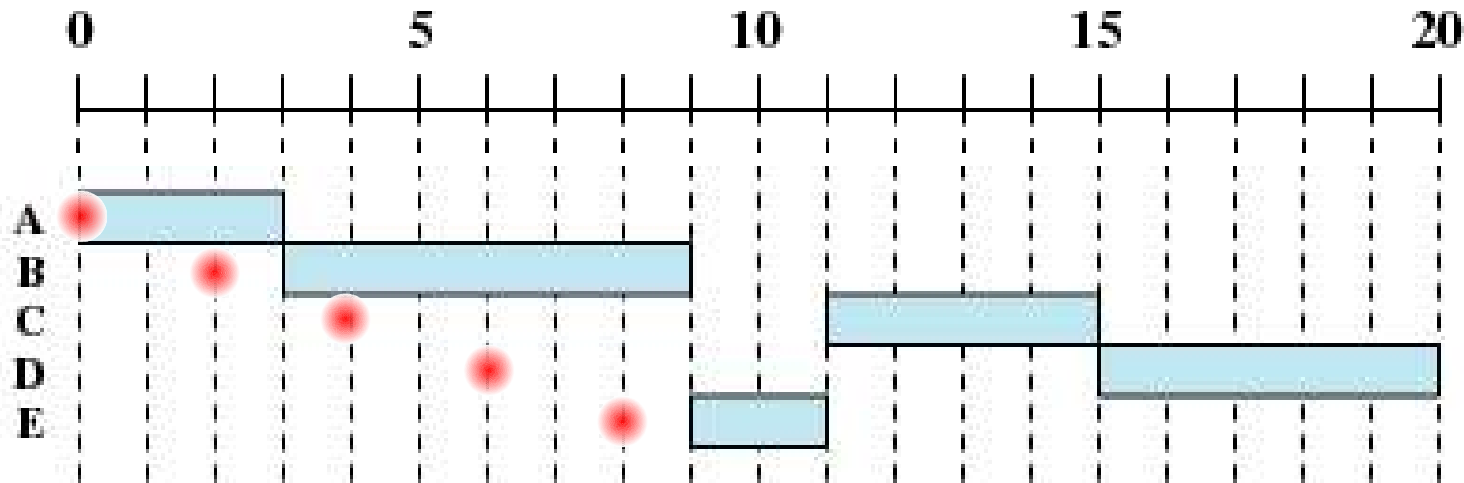
- process with shortest expected processing time is selected next (expected processing time to the next blocking i/o operation)

- need to know future! approximated.

- non-preemptive policy optimal w.r.t. minimum total waiting time

30

Shortest Process Next

- Predictability of longer processes is reduced
- Possibility of starvation for longer processes
- exstimation of time length of the next cpu-bust may be done by exponential averaging

$$S_{n+1} = \alpha T_n + (1-\alpha) S_n$$
$$\alpha \in (0,1]$$

Shortest Remaining Time (SRT)

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2
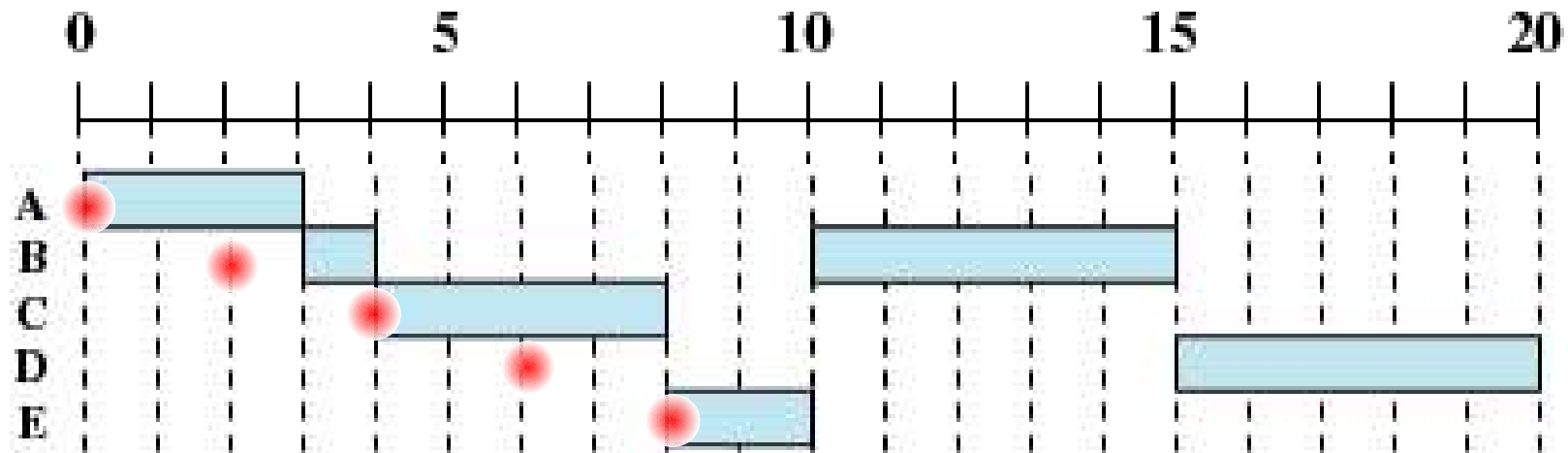
- Preemptive version of shortest process next policy
- Must estimate processing time

feedback

- does not need service time estimation

- preemptive as in RR

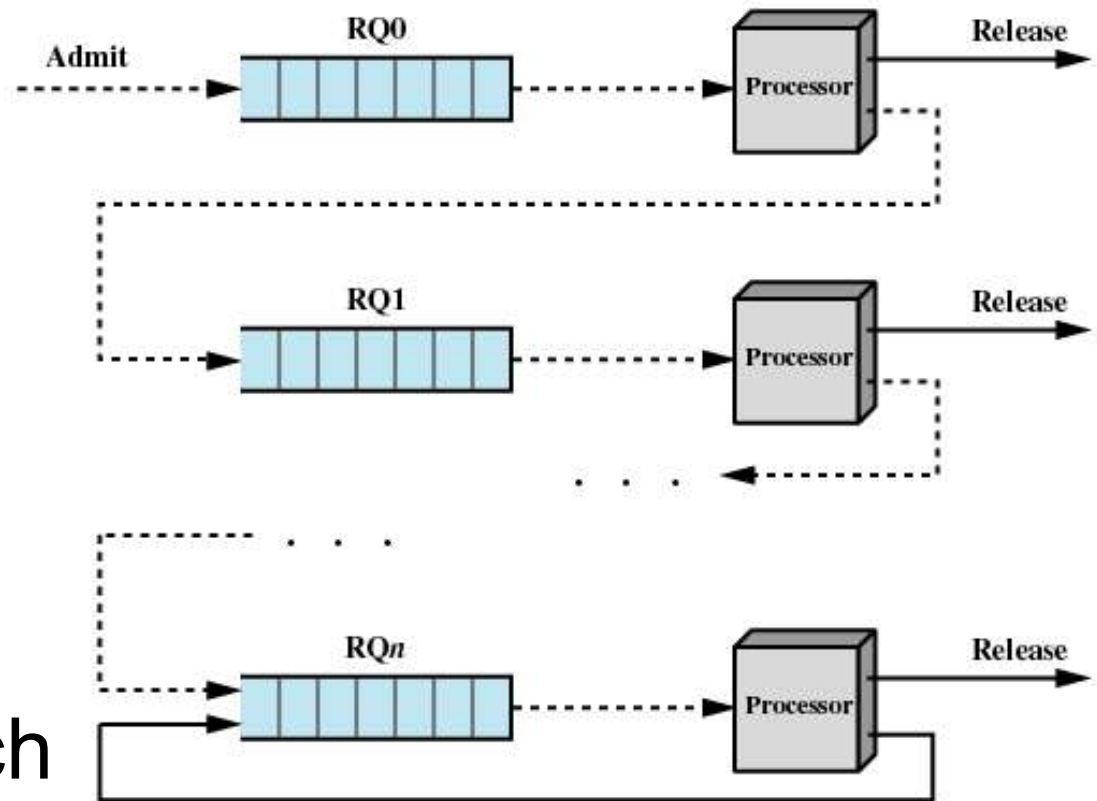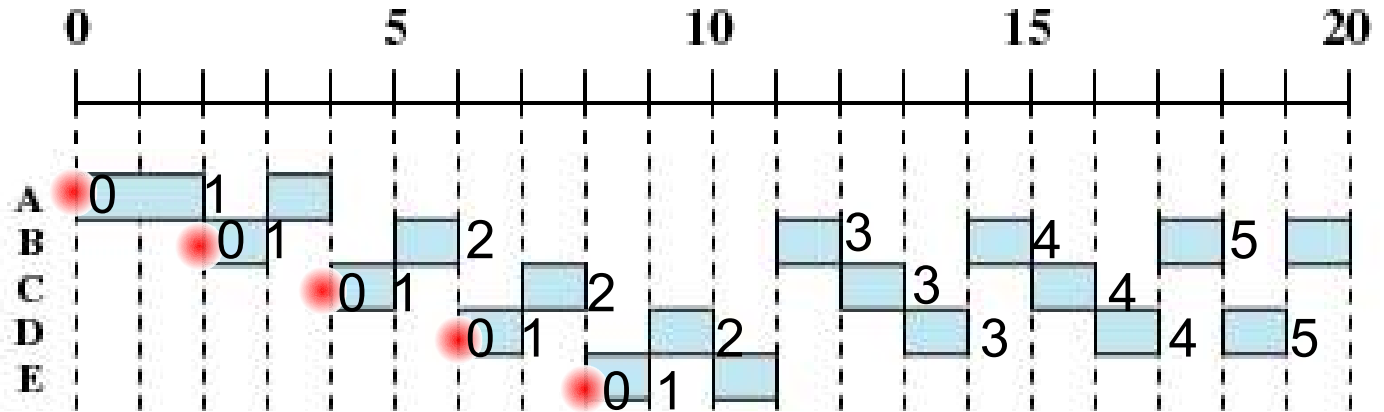- demotes processes at each expired time quantum into lower priority queues

Figure 9.10   Feedback Scheduling

Feedback

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

Feedback
$q = 1$

34

feedback: varianti

- un processo scala di priorita'...

  – sempre quando scade il suo quanto di tempo

  oppure

  – quando scade il quanto e c'e` almeno una altro processo nel sistema (Stallings)

feedback: varianti

- un processo aumenta di priorita' quando va in blocco...

  – aumento fisso ogni volta che va in blocco

  oppure

  – aumento dipende dal tempo speso in blocco

# Table 9.3 Characteristics of Various Scheduling Policies

	Selection Function	Decision Mode	Throughput	Response Time	Overhead	Effect on Processes	Starvation
FCFS	max[w]	Nonpreemptive	Not emphasized	May be high, especially if there is a large variance in process execution times	Minimum	Penalizes short processes; penalizes I/O bound processes	No
Round Robin	constant	Preemptive (at time quantum)	May be low if quantum is too small	Provides good response time for short processes	Minimum	Fair treatment	No
SPN	min[s]	Nonpreemptive	High	Provides good response time for short processes	Can be high	Penalizes long processes	Possible
SRT	min[s − e]	Preemptive (at arrival)	High	Provides good response time	Can be high	Penalizes long processes	Possible
HRRN	$\max\left(\dfrac{w+s}{s}\right)$	Nonpreemptive	High	Provides good response time	Can be high	Good balance	No
Feedback	(see text)	Preemptive (at time quantum)	Not emphasized	Not emphasized	Can be high	May favor I/O bound processes	Possible

w = time spent waiting
e = time spent in execution so far
s = total service time required by the process, including e

simulation results
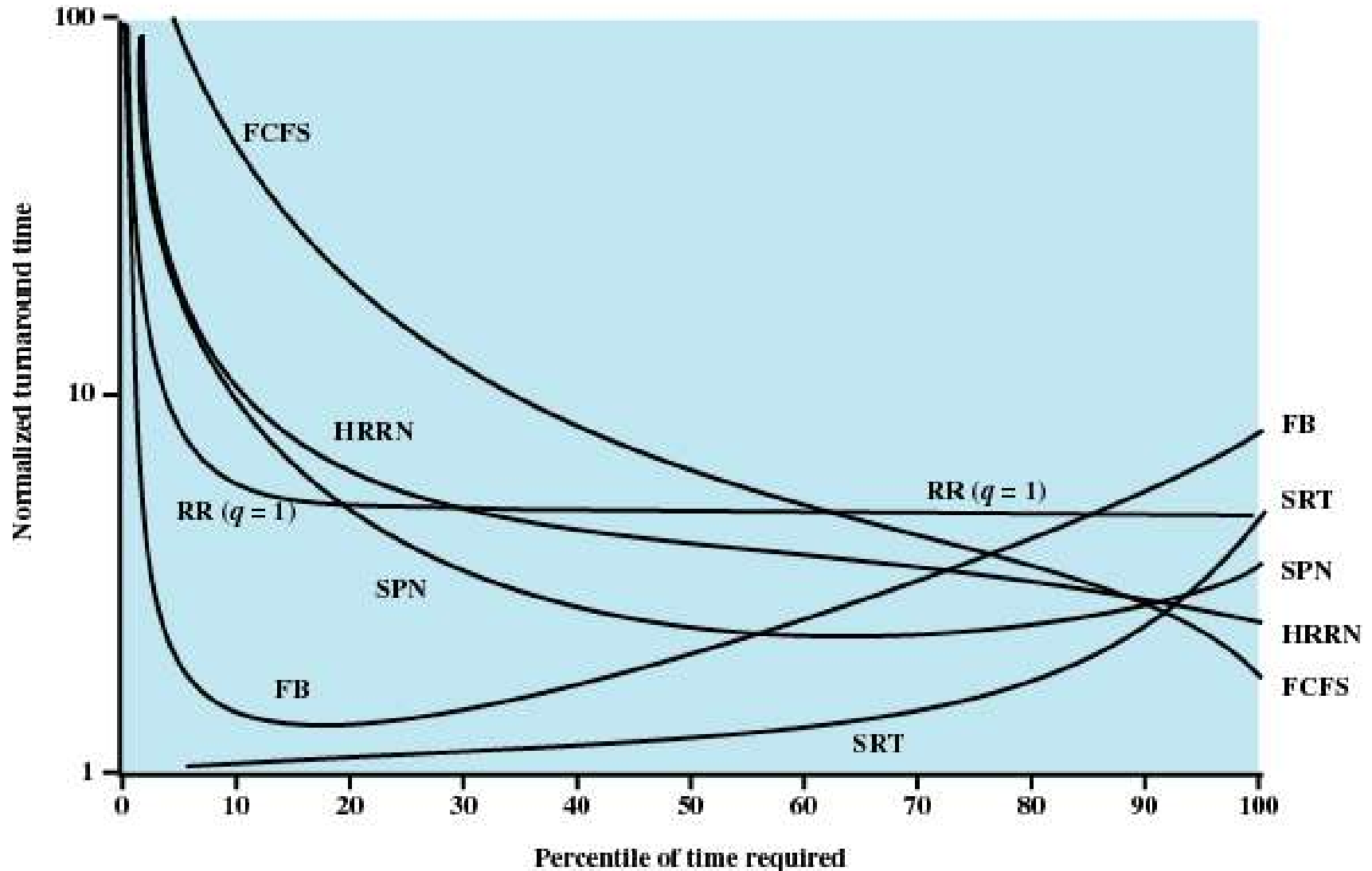
linux scheduling policies in kernel 2.6

- conventional processes

  - FB with preemption (also in kernel mode)

  - "estimation" of the cpu-burst

  - higher priority for processes with shorter cpu-burst

  - user defined parameter (command: nice)

- real time processes

  - FCFS

  - RR (user defined time quantum)

  - priorities and preemption

sistemi operativi a.a. 2005-2006

© 2005-2006 maurizio pizzonia