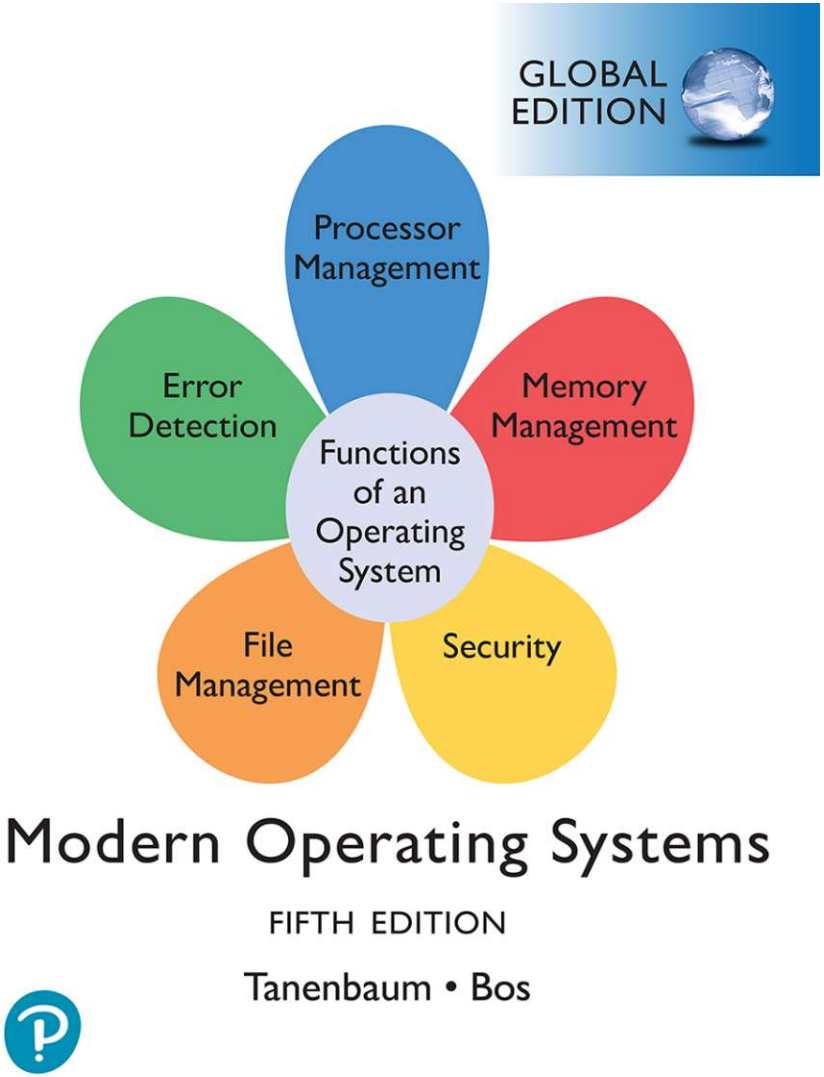
Modern Operating Systems

Fifth Edition, Global Edition



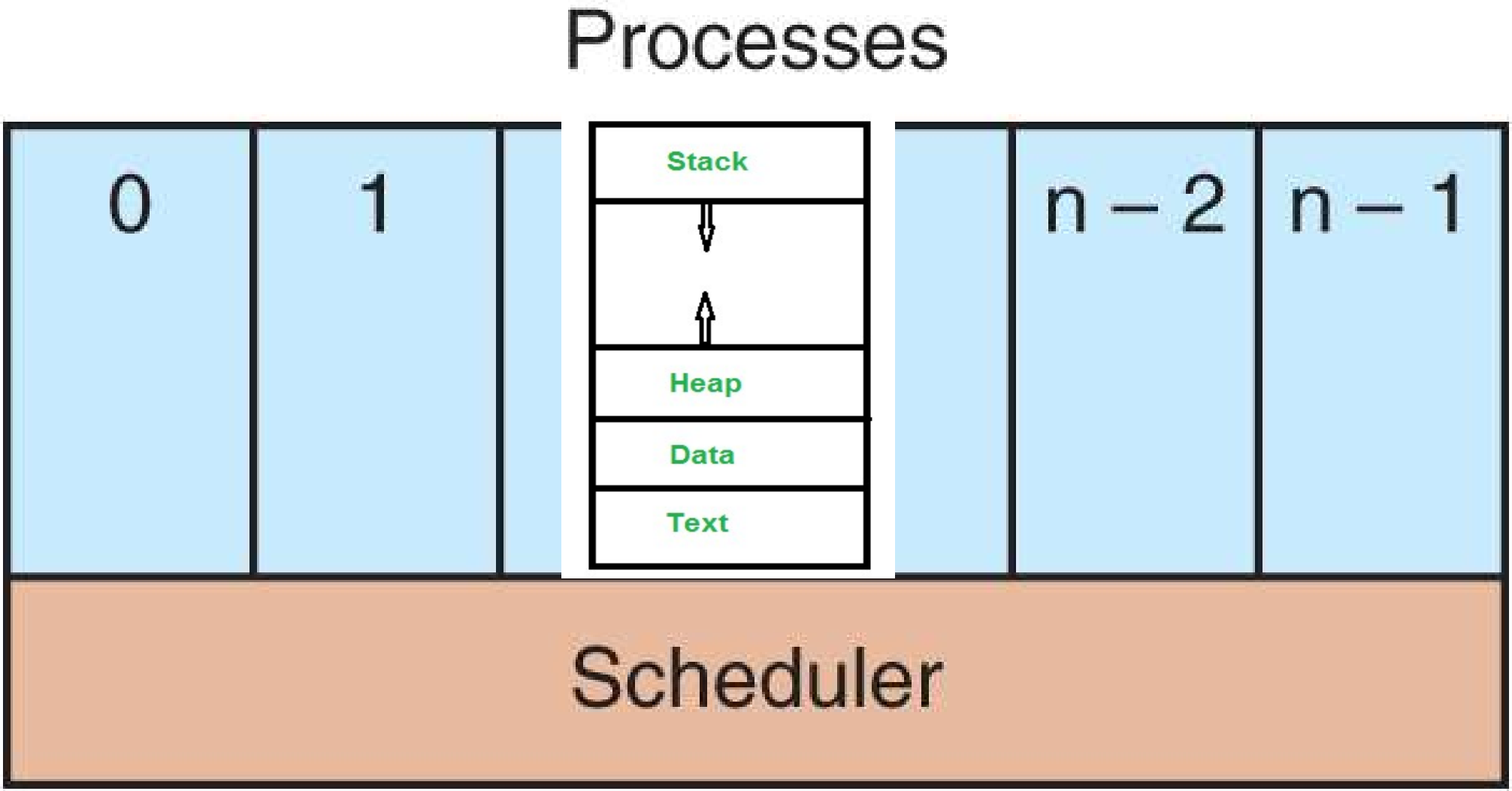
Lecture 4

Job Scheduling

Pearson

Job Scheduling

The lowest layer of a process-structured operating system handles interrupts and scheduling. Above that layer are sequential processes.



Job Scheduling

Allocate CPU resources (time sharing)

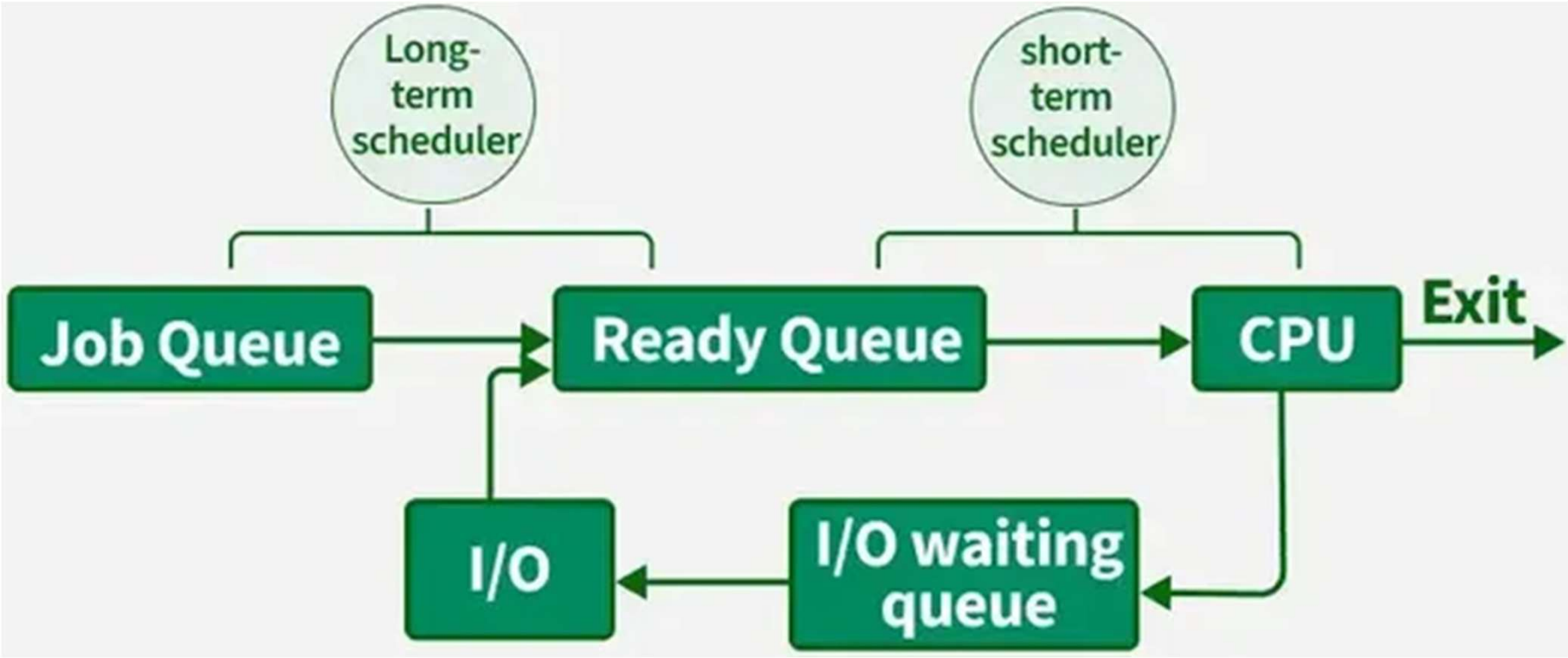
- Which process next?
- How long should it have the CPU?
  - process behavior

Critical when CPU resources are limited

Algorithms that optimize scheduling

- fairness –"fair" share for each process
- policy enforcement
- balance –avoid unused resources

Lifecycle of a Process



Types of Schedulers

Admissions scheduler – when a process can start?

- Not commonly used on consumer operating systems
- Never used for interactive processes

Memory scheduler – suspends/reloads processes when performance degrades

- Not usually used with virtual memory
- Not usually used with interactive processes

Short-term scheduler – which process runs next

Scheduling Algorithm Characteristics

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

Performance Criteria - Metrics

Non-interactive processes

- max CPU load (light: 40% to heavy: 90%)
- max throughput: # completed jobs per unit time
- min turnaround (time from submission to completion)
- min waiting time (time in queue, waiting queue?)

Interactive systems

- min response time (time from job start to first response)
- proportional to expectations

Short-Term Scheduler

First come first served (FCFS)

- Non-preemptive
- Ready processes are stored in an FCFS ready queue
- Processes get CPU until CPU burst expires (in a multiprocessing system)

One job with a high CPU burst can greatly impact the average turn around time.

Batch Systems: Example

No multiprocessing, all jobs started at the same time

Turnaround time:

first come, first serve			
P	start	complete	turnaround
P_A	0	9	9
P_B	9	13	13
P_C	13	19	19

Shortest job first			
P	start	complete	turnaround
P_B	0	4	4
P_C	4	10	10
P_A	10	19	19

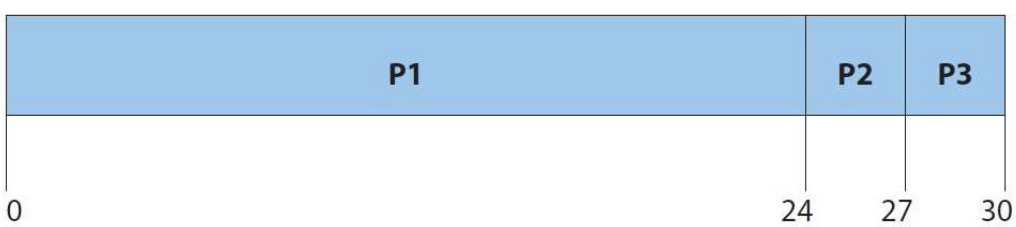
- Mean turnaround?

$$FCFS: \frac{9+13+19}{3} \approx 13.7 \quad SJF: \frac{4+10+19}{3} = 11$$

First come first served (FCFS)

P1	24
P2	3
P3	3

P1, P2, P3



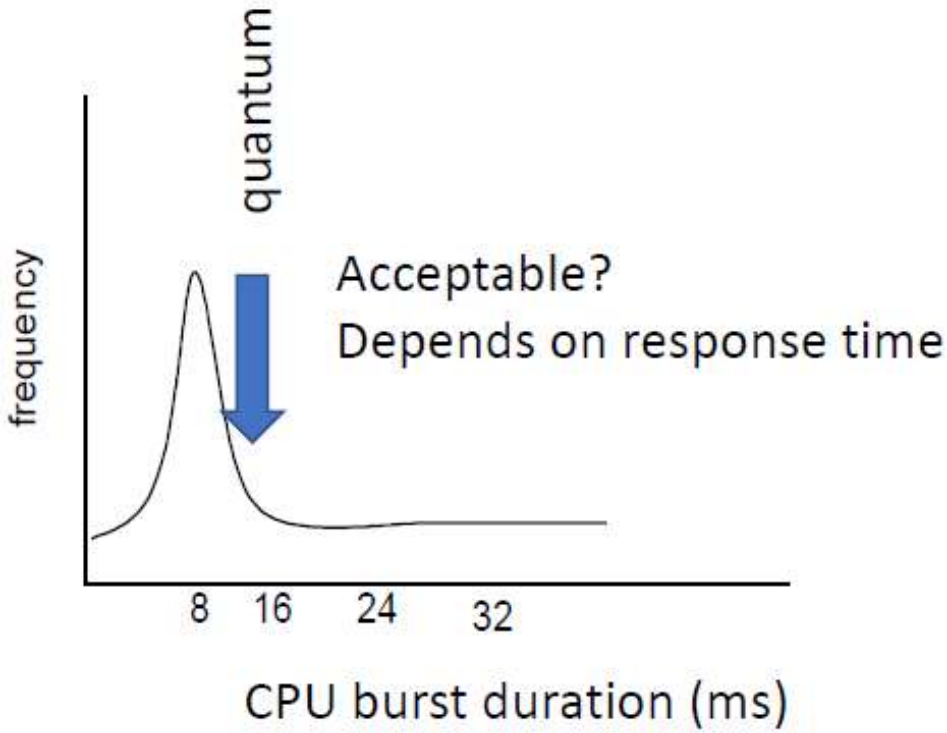
P1	24
P2	3
P3	3

P2, P3, P1

- Average Waiting Time: (0+24+27)/3 = 17 vs (0+3+6)/3 = 3
- FCFS avoid convoy effect
  - small processes behind large ones

Interactive Short-Term Scheduler: Round Robin

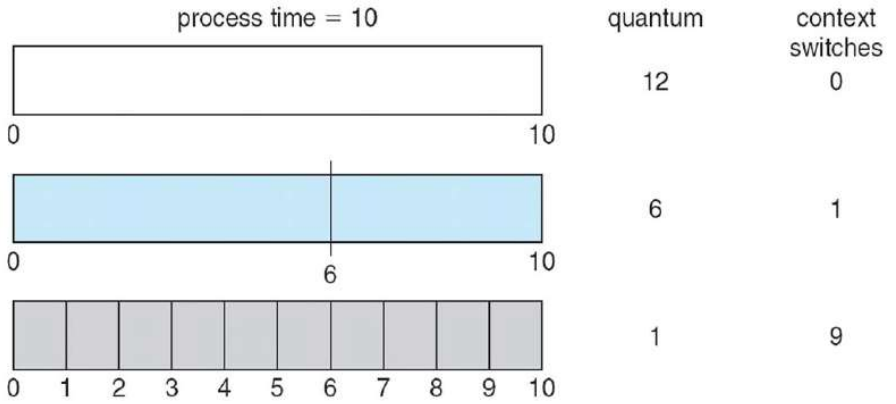
- Preemptive scheduler, each process given a time quantum
- Process executes until:
  - CPU burst terminates, or
  - a quantum timer expires



How long should a quantum be?

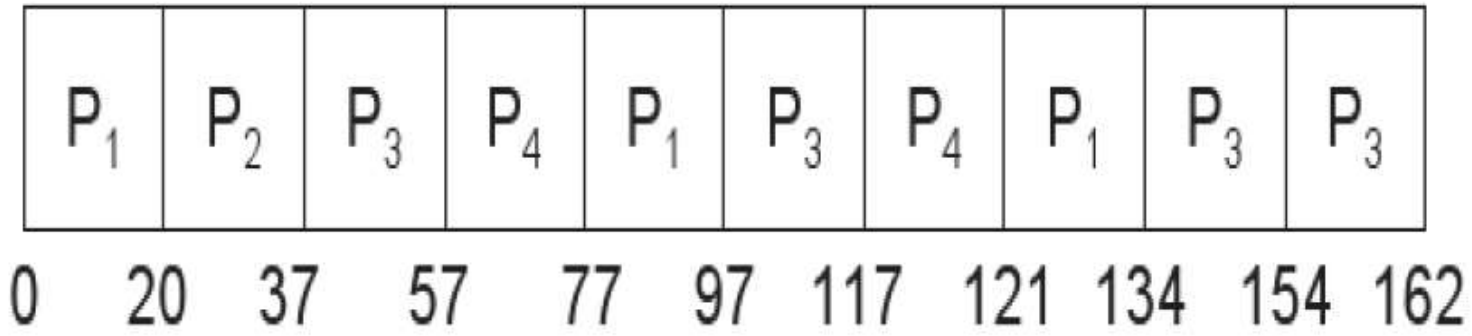
- Too long: Poor response time
- Too short: Waste time in context switches

Usually it is 10 to 200ms



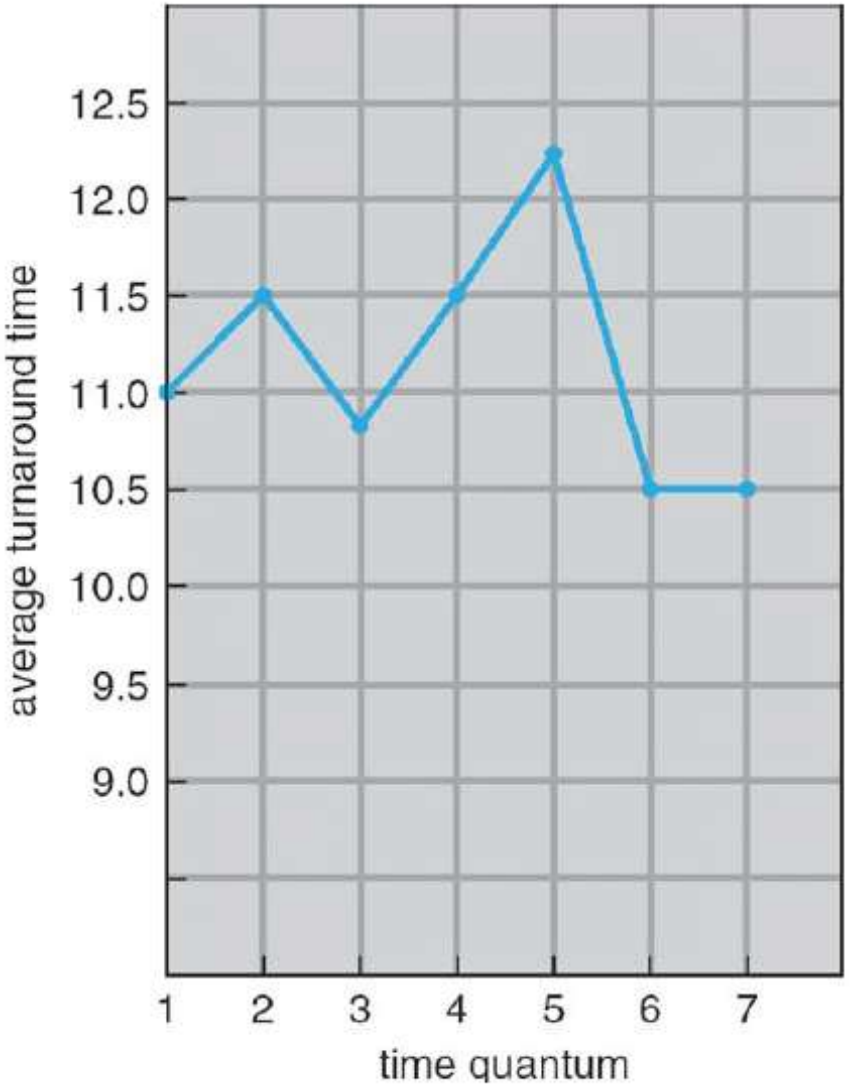
Round Robin

	Duration
P1	53
P2	17
P3	68
P4	24



- Generally, it results in a longer turnaround time than SJF, but provides better response time.

Round Robin and Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

- A large quantum does not always increase turnaround time

Interactive Short-Term Schedulers: Priority Scheduler

- Each process assigned a priority level (number)
- Processes in ready queue scheduled by priority
- Ties broken by policy rule, e.g., FCFS
- Preemptive: Arrival of a higher priority process can displace an executing process (Starvation -> Aging)

Assignment of priority is a policy decision.

- Navigation -> sensor reading
- Take into account deadlines
- Set priority depending on the duration of the last CPU burst

Priority Scheduler

Each process is assigned a priority. At each CPU scheduling decision, the process with the highest priority is selected

- SJF (Shortest Job First) is a special case
- Preemptive vs Non-Preemptive Scheduling

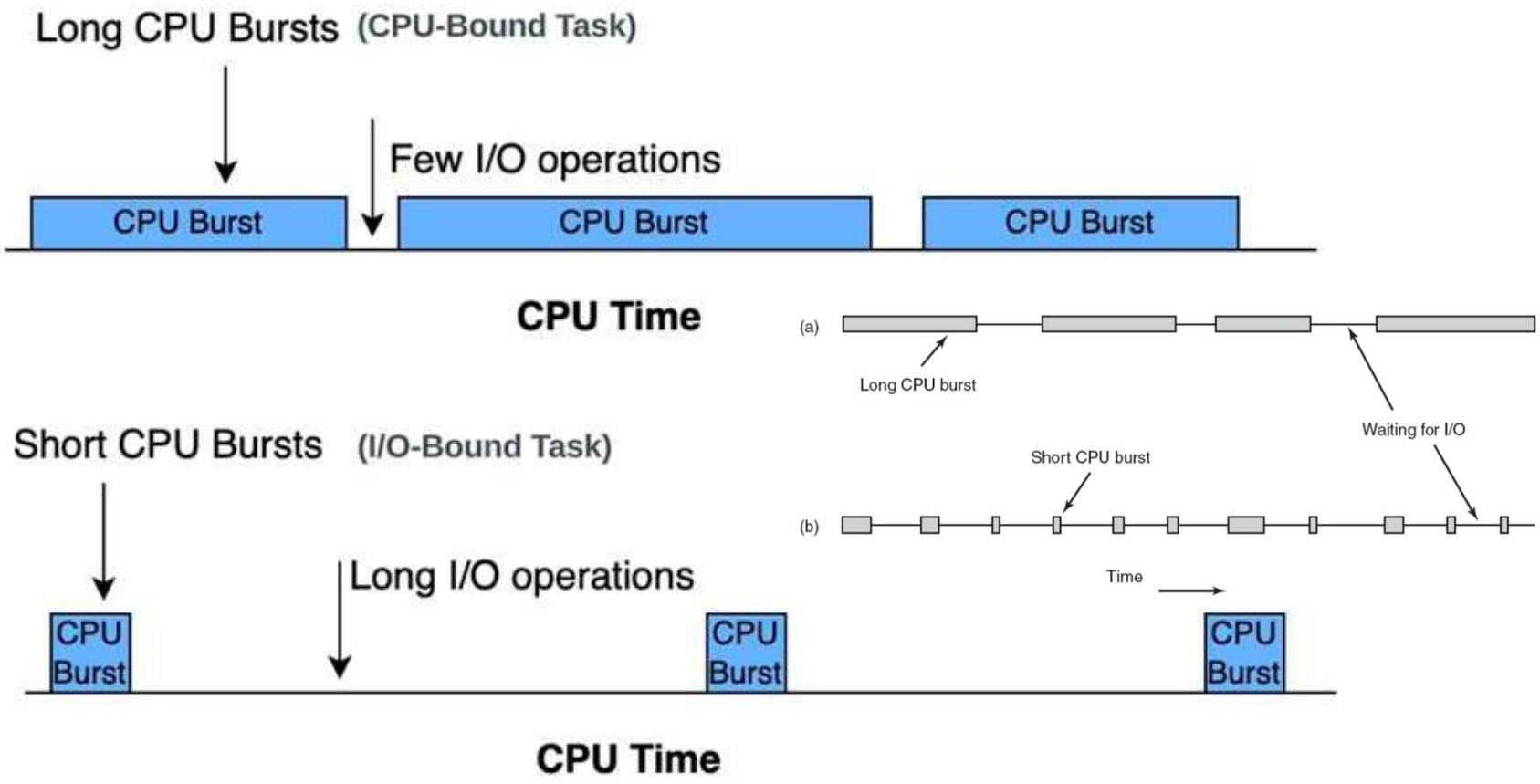
Starvation problem

- Processes with low priority may never execute
- Solution: Aging
  - As time passes, the priority of a waiting process increases so they eventually get executed.

Interactive Short-Term Scheduler: Shortest Job Next

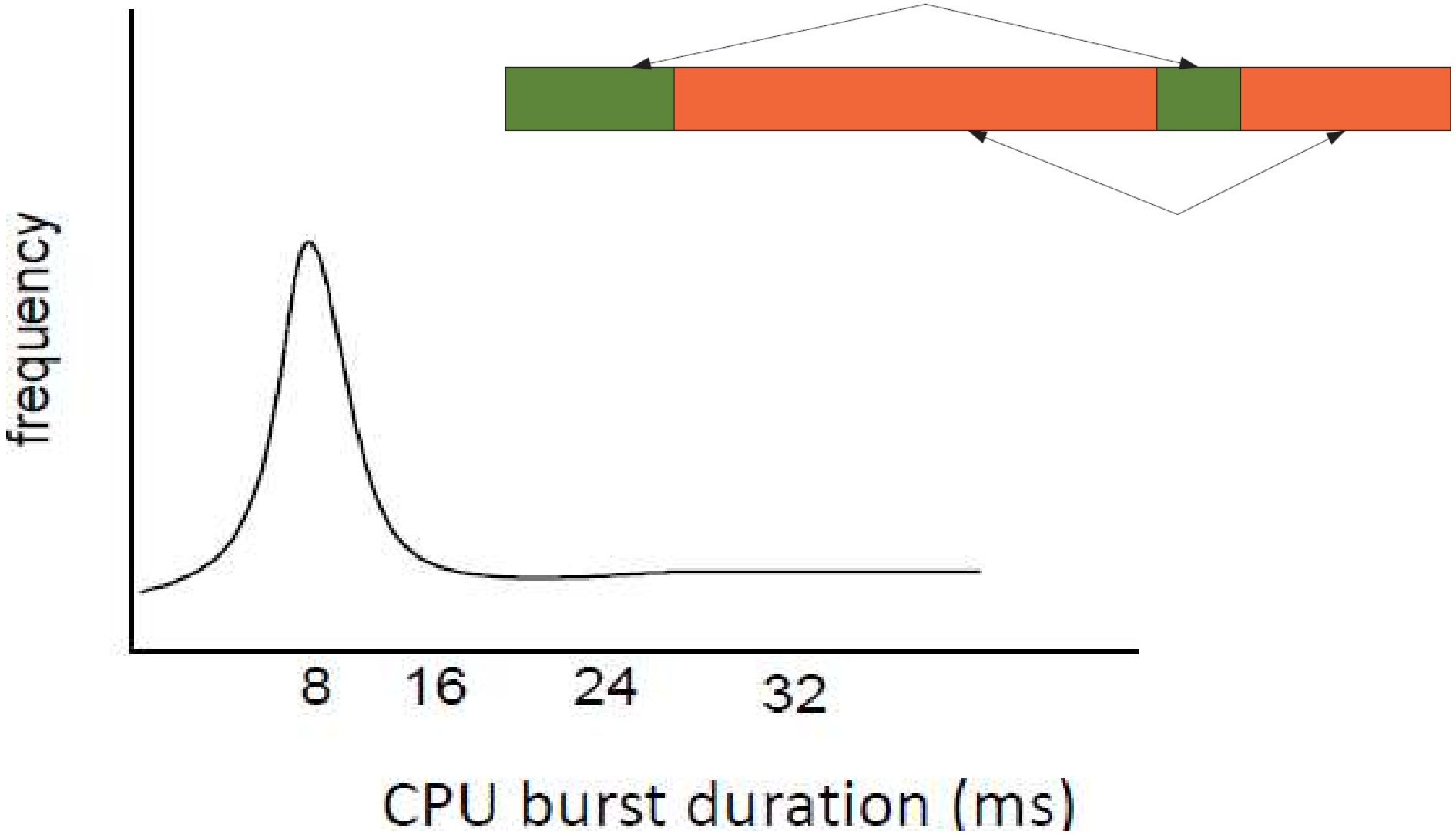
- Special case of priority scheduling
- Priority inversely proportional to the *next CPU burst*
- Minimize average waiting time
- Predicting CPU burst using EWMA

CPU and I/O Bursts



- CPU burst: Amount of time a process executes before it requires I/O
  - depends on the application, but typically follows a distribution
- I/O burst: Amount of time a process spends waiting for I/O (DMA)

CPU Bursts

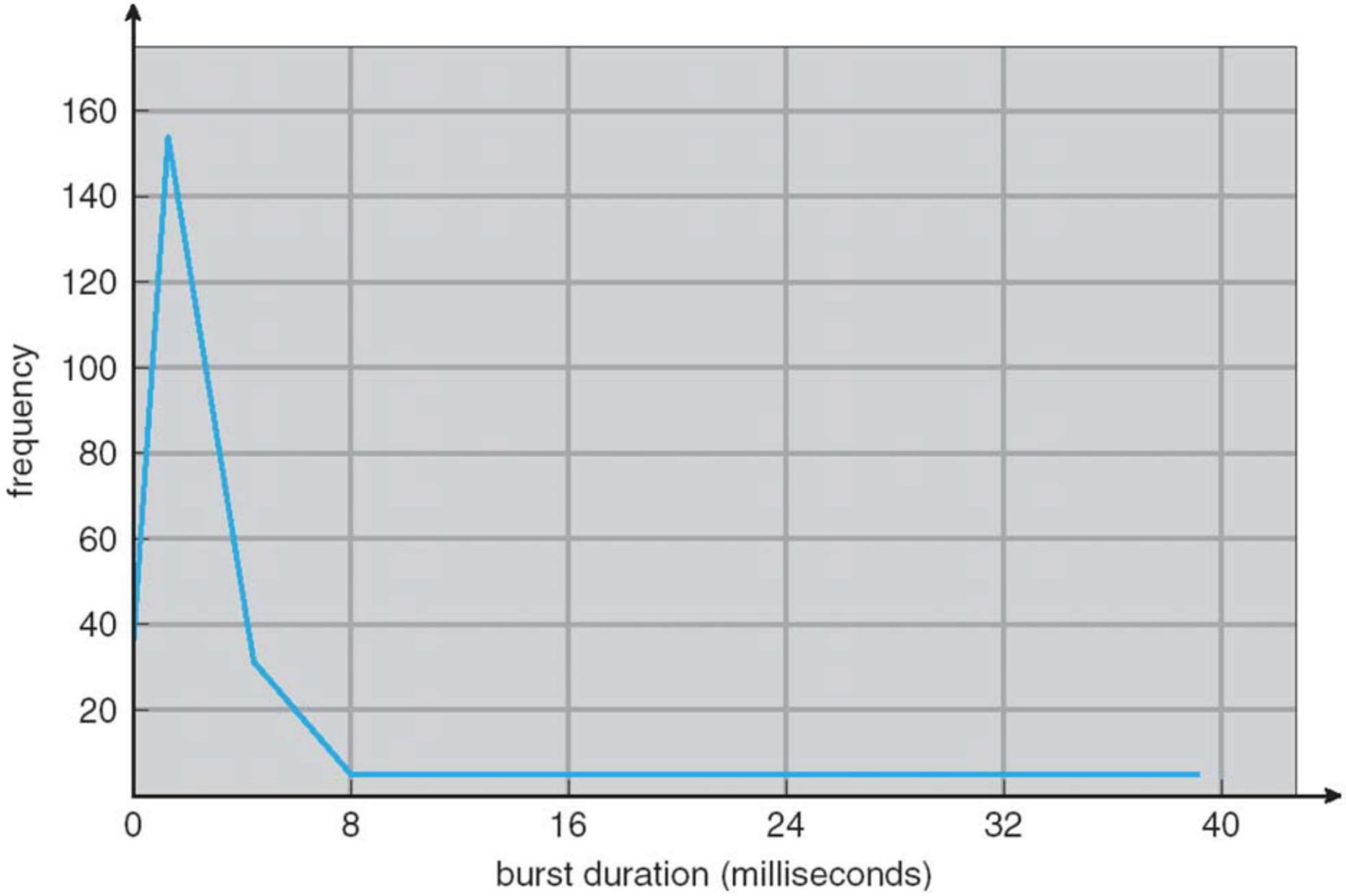


Interactive Short-Term Scheduler: Shortest Job Next

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\, \tau_n$$

- τ: prediction
- t: actual value
- $\alpha \in [0,1]$: relative weight of t and τ
  - $\alpha = 0$: Measurements of actual delay values do not affect the estimate
  - $\alpha = 1$: Only the most recent actual delay is considered

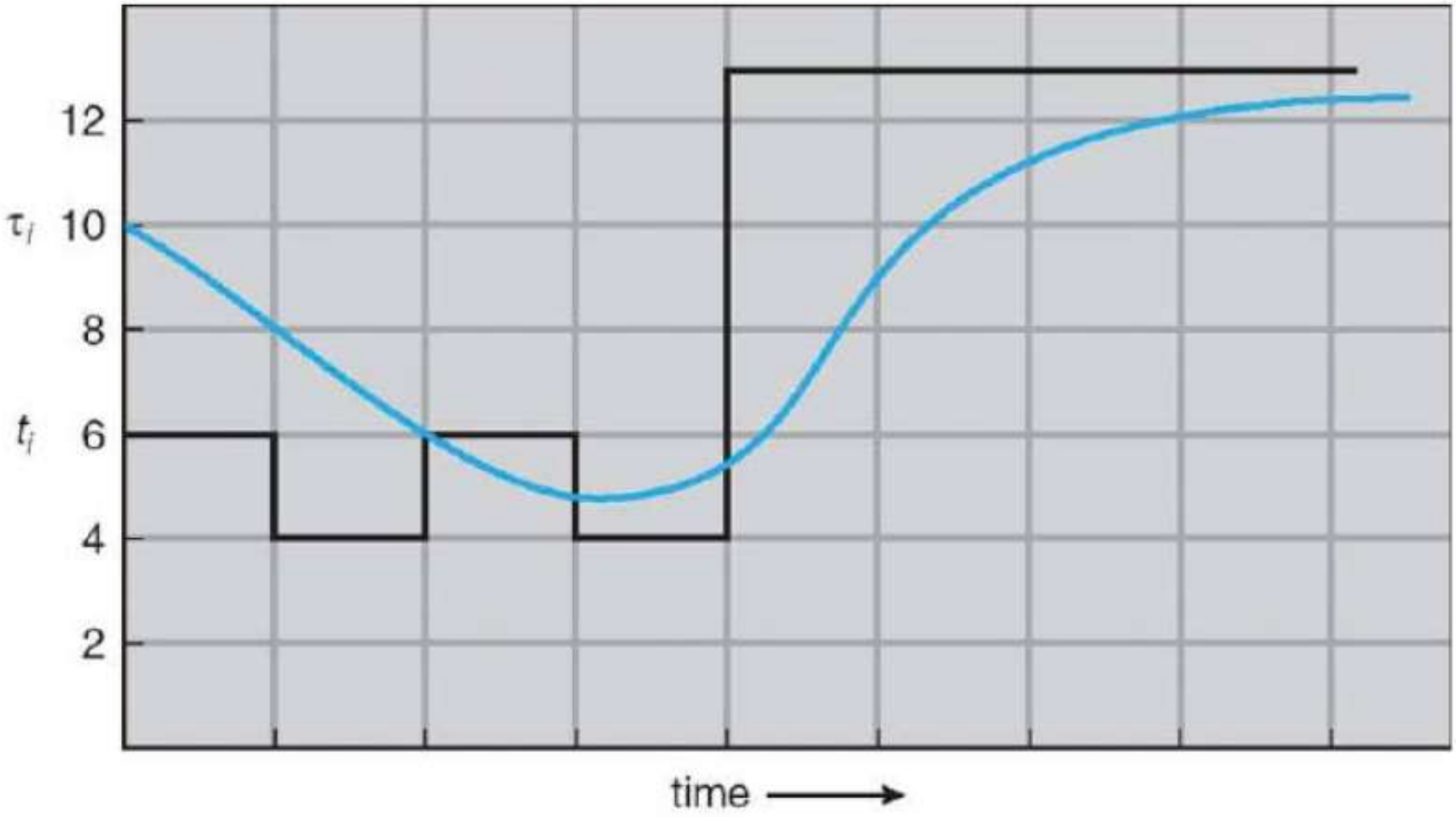
CPU Bursts


frequency
160
140
120
100
80
60
40
20
0
8
16
24
32
40
burst duration (milliseconds)

Interactive Short-Term Scheduler:

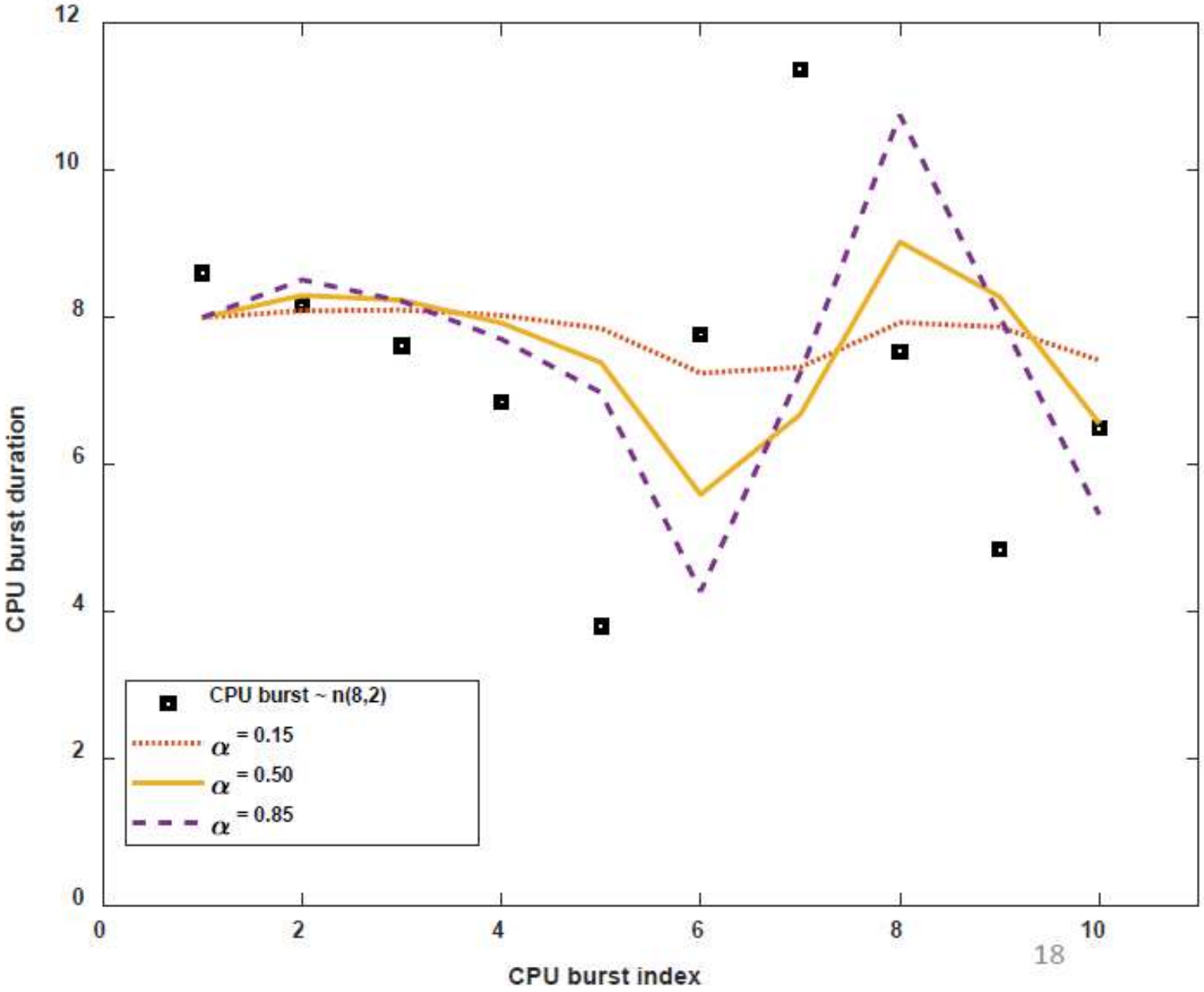
$a = 1/2$

$\tau_0 = 10$



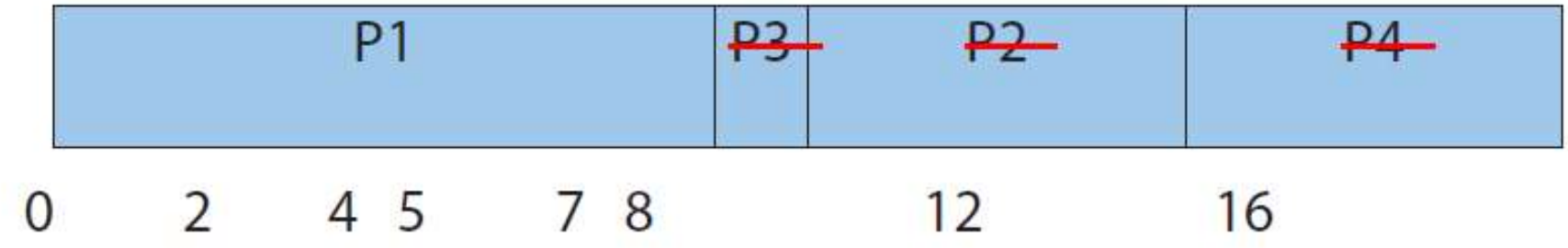
CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Interactive Short-Term Schedulers: Shortest Job First



Shortest Job First

	Arrival Time	Duration
P1	0	7
P2	2	4
P3	4	1
P4	5	4



Average Waiting Time: (0 + (8-2) + (7-4) +(12-5)) / 4 = 4

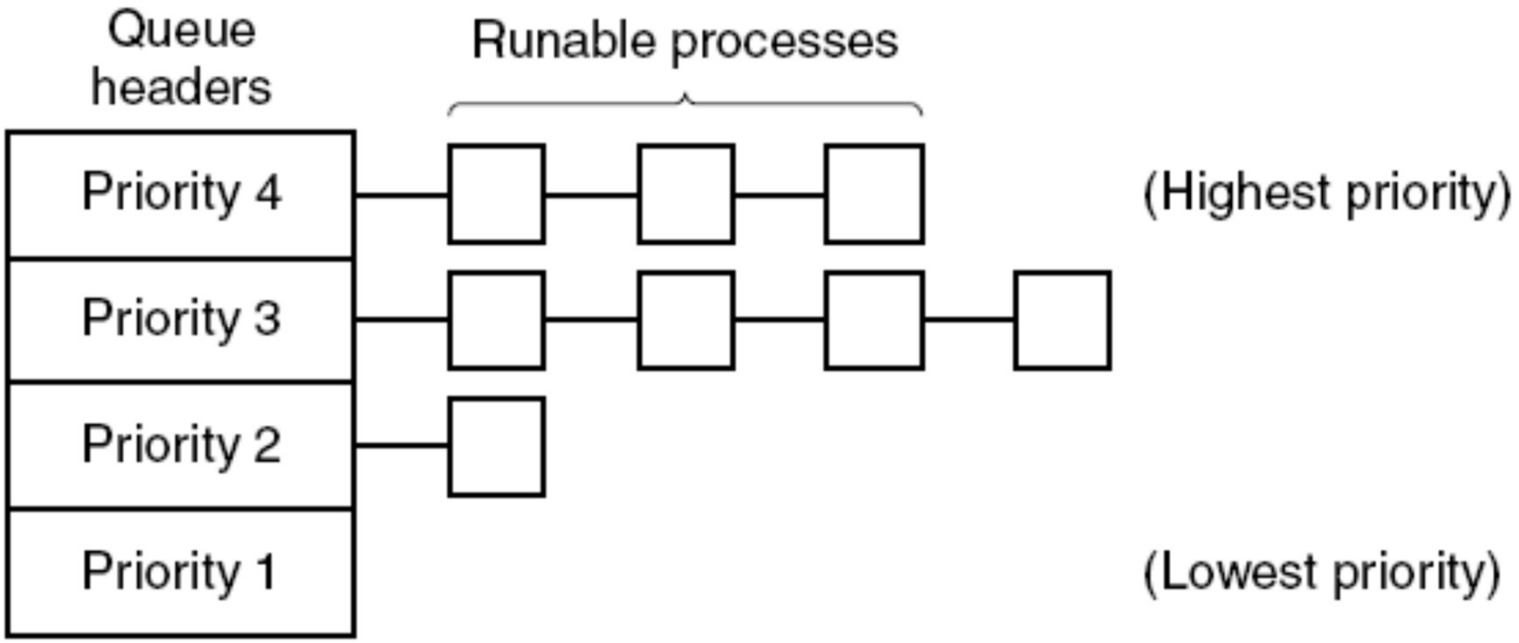
Shortest Job First (Preemptive)

	Arrival Time	Duration
P1	0	7
P2	2	4
P3	4	1
P4	5	4

t		
0-2	(P1,7)	P1
2-4	(P1,5) (P2,4)	P2
4-5	(P1,5) (P2,2) (P3,1)	P3
5-7	(P1,5) (P2,2) (P4,4)	P2
7-11	(P1,5) (P4,4)	P4
11-16	(P1,5)	P1

Average Waiting Time: ((11-2)+ (5-4) + (4-4) +(7-5)) / 4 = 3

Interactive Short-Term Schedulers: Multiple Queues



Each queue has a possibly different scheduling policy

Queue scheduling

- Only when higher priority queues empty, or
- Time share the queues.
- Common to allow longer scheduling for low priority queues.

Interactive Short-Term Schedulers

Guaranteed scheduling (preemptive)

- Processes will receive 1/nth of the CPU
- Must track history of process and schedule appropriately

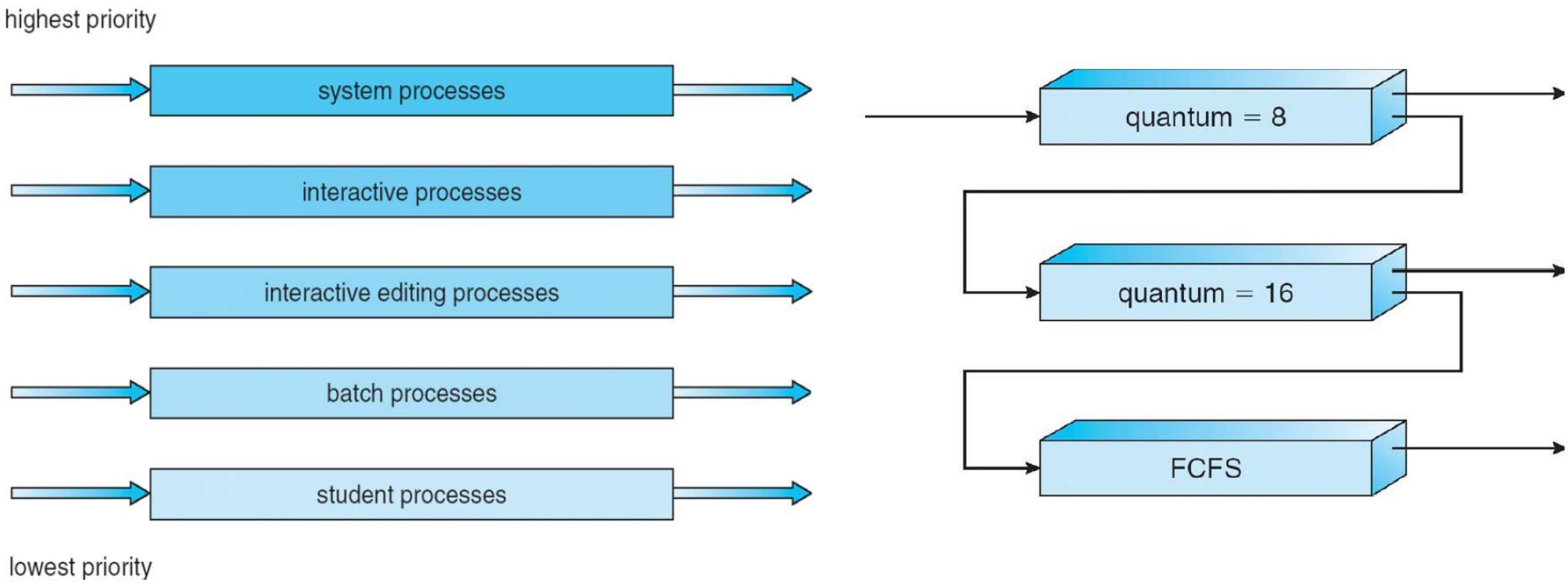
Fair share scheduling (preemptive)

- Switches fairness from process-centric to user-centric
- Each user gets 1/n'thof CPU that is shared amongst their processes

Lottery scheduling

- Stochastic scheduling algorithm –Non-deterministic
- Each process given f(N) lottery tickets
- Winner gets scheduled
- Processes can share tickets

Hierarchical Scheduling



Scheduling in Multicore Systems

Asymmetric

- The scheduler runs on one processor
- The other processors are used to execute user code

Symmetric

- The scheduler runs on all processors
- Each scheduler selects a process to execute on its corresponding processor
- Synchronization is required

Scheduling Issues in Multicores

Processor Affinity

- Cache performance

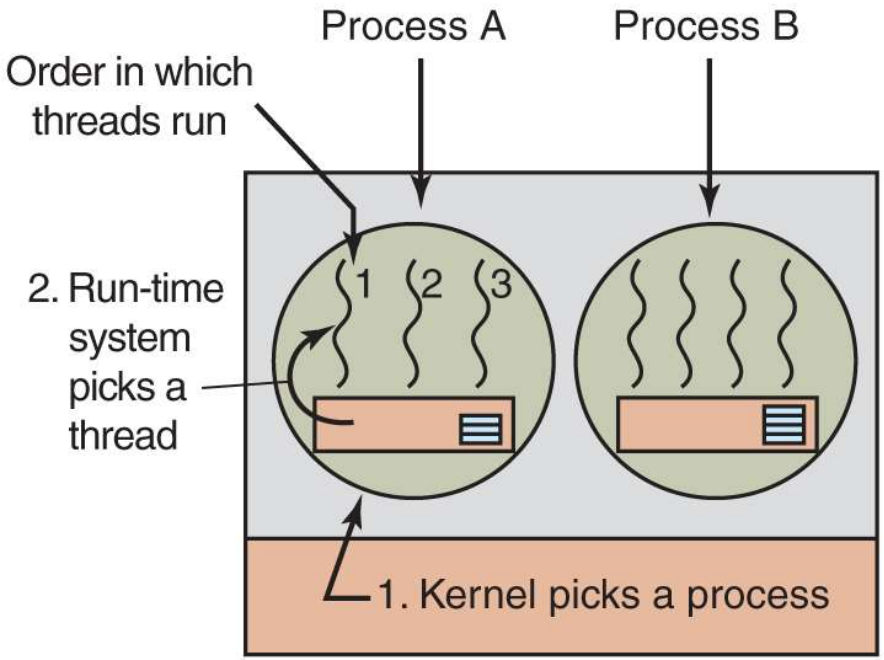
Load Balancing

- Requires migration of processes
  - Push migration
  - Pull migration

Awareness of the system's topology, e.g., NUMA, SMT

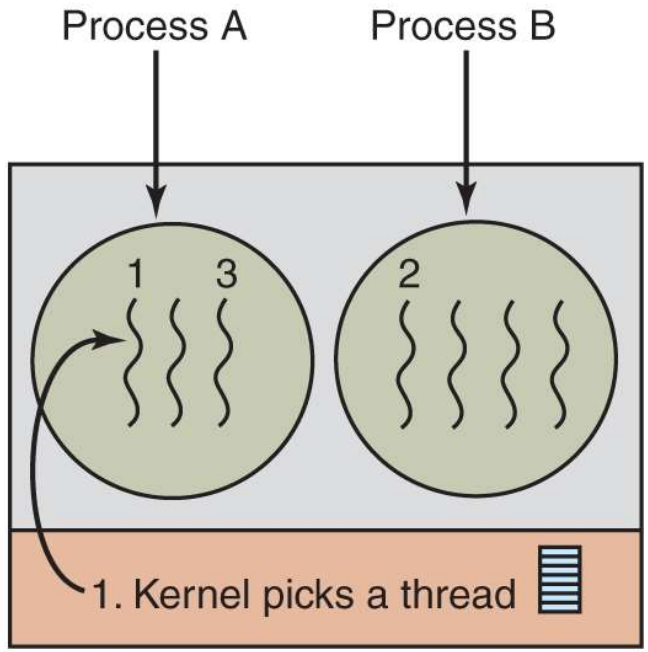
User- & Kernel-Level Thread Schedules

(a) Possible scheduling of user-level threads with a 50-msec process quantum and threads that run 5 msec per CPU burst. (b) Possible scheduling of kernel-level threads with the same characteristics as (a).



Possible: A1, A2, A3, A1, A2, A3
Not possible: A1, B1, A2, B2, A3, B3

(a)



Possible: A1, A2, A3, A1, A2, A3
Also possible: A1, B1, A2, B2, A3, B3

(b)