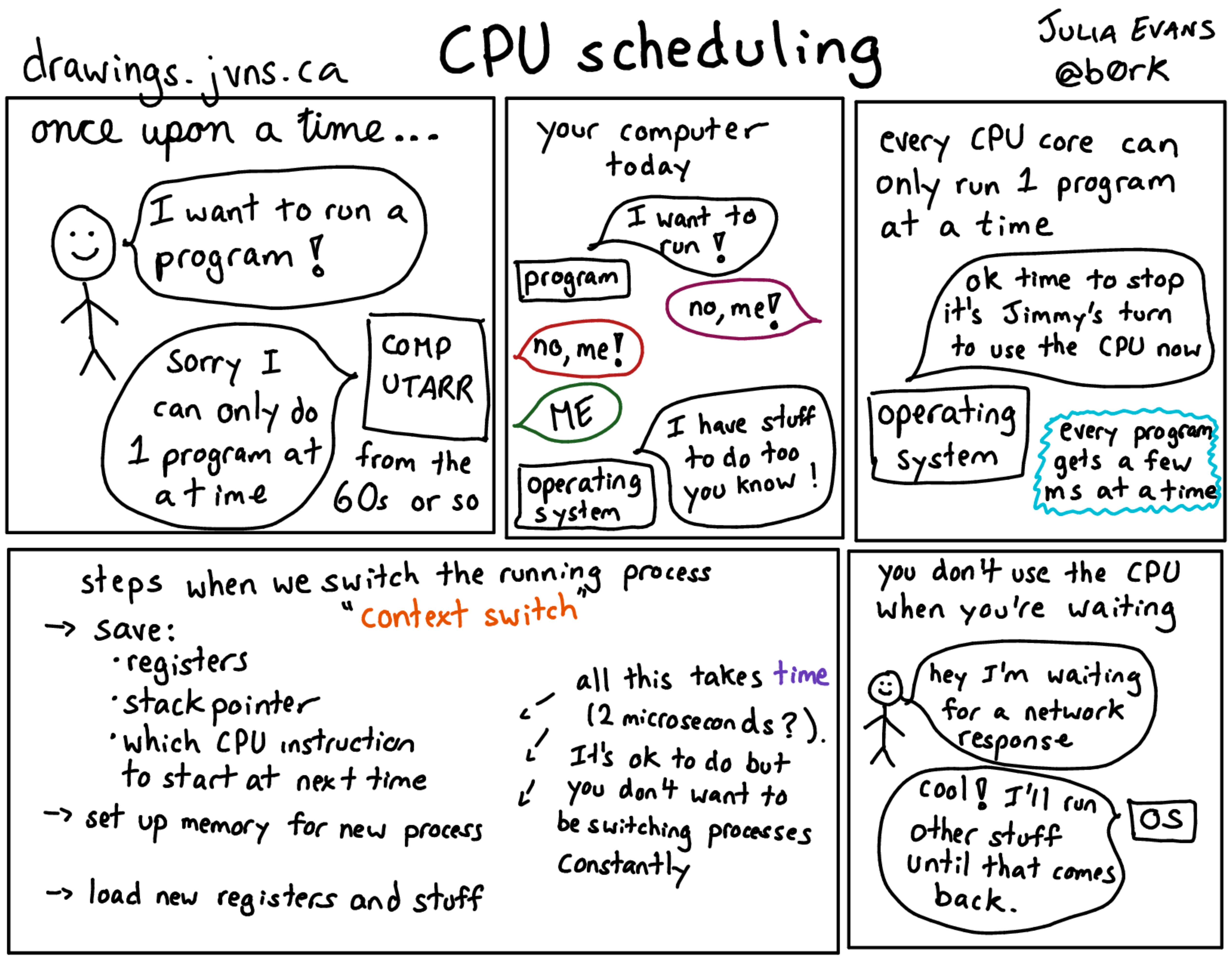
Scheduling (Part I)

Professor Travis Peters
CSCI 460 Operating Systems
Fall 2019

Some slides & figures adapted from Stallings instructor resources.

Some slides adapted from Adam Bates's F'18 CS423 course @ UIUC
https://courses.engr.illinois.edu/cs423/sp2018/schedule.html



https://drawings.jvns.ca/scheduling/

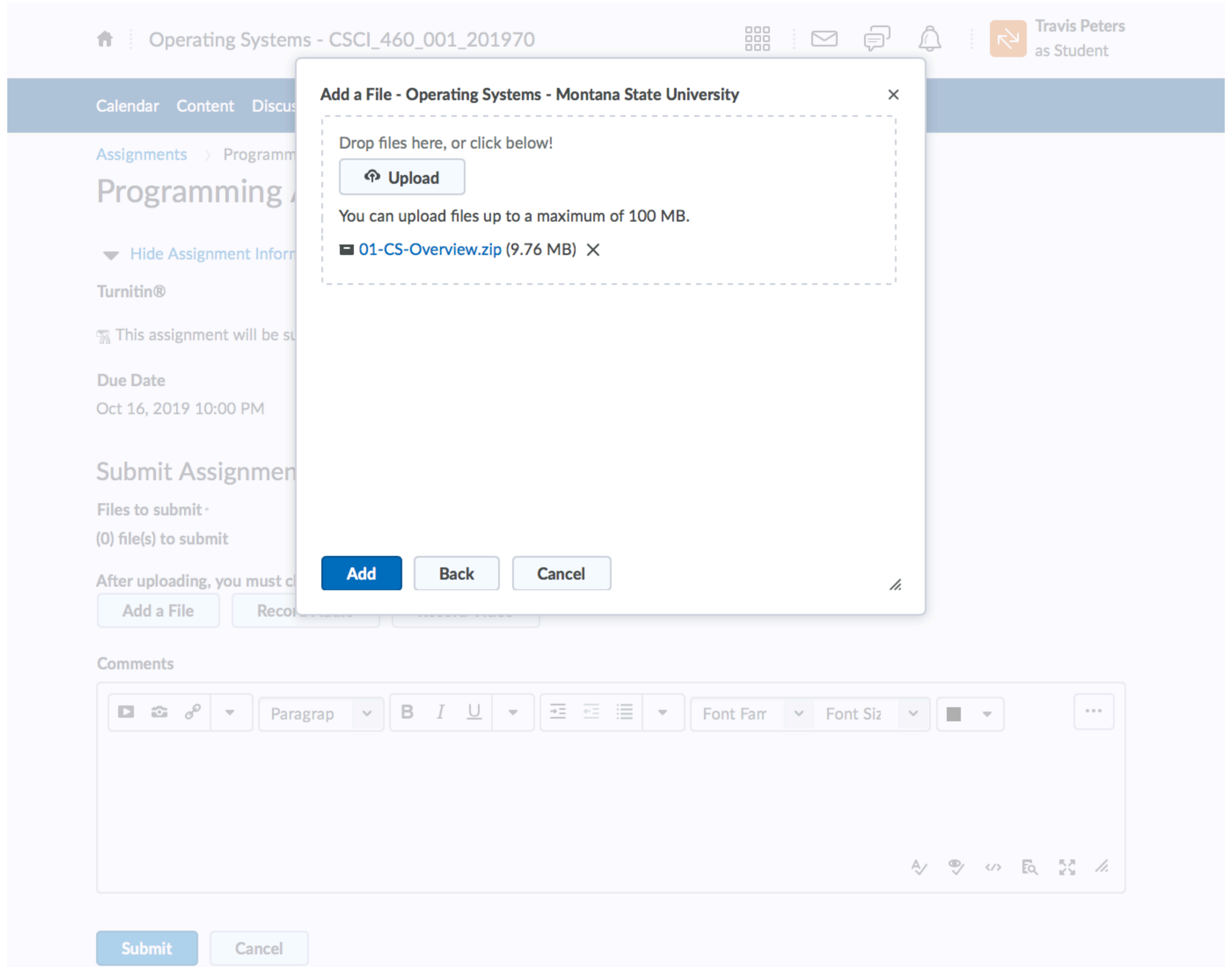
Goals for Today

Learning Objectives

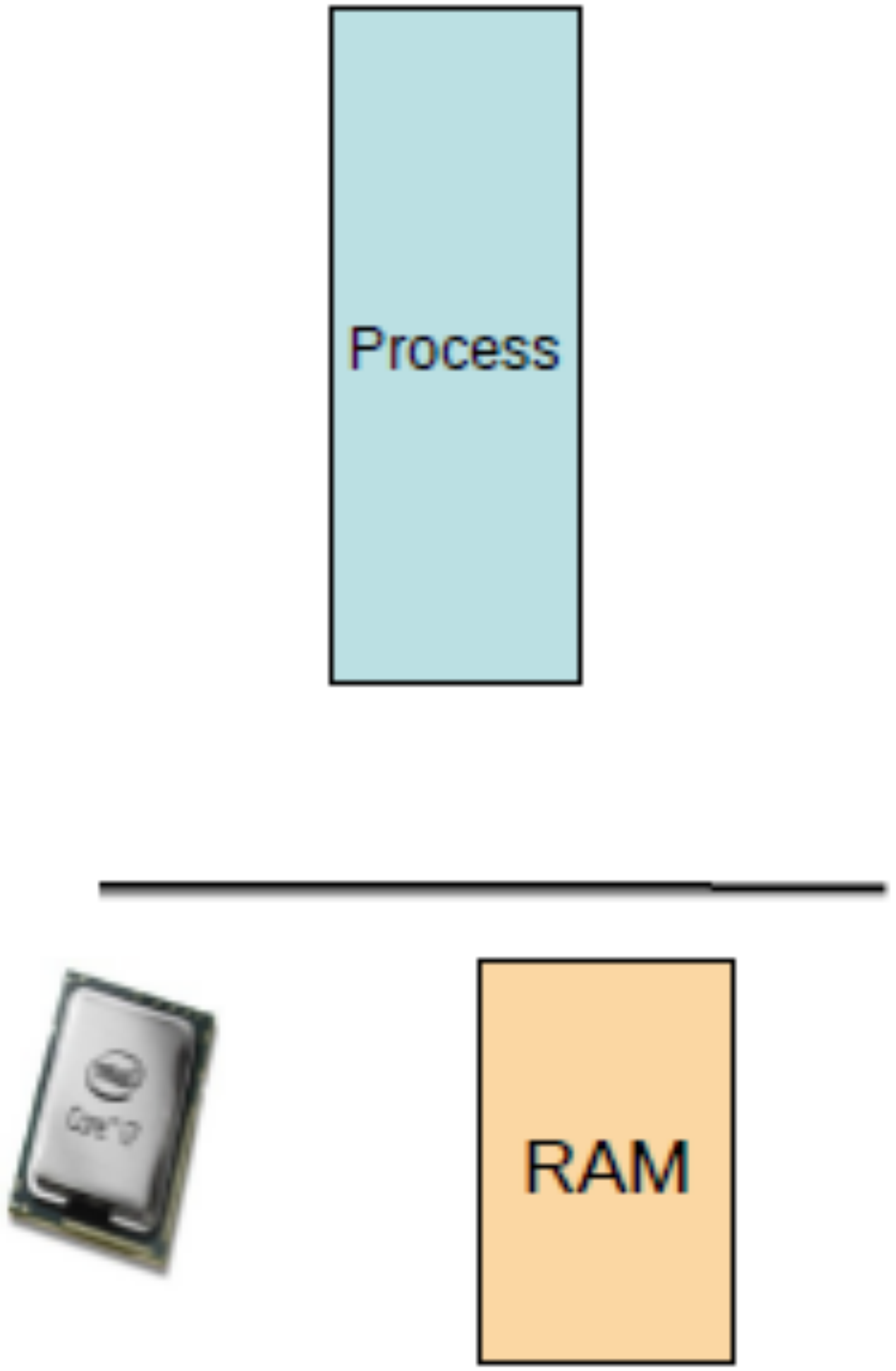
- Explain the differences among long-, medium-, and short-term scheduling.
- Assess the performance of different scheduling policies.
- Understand the scheduling technique used in traditional UNIX.

Announcements

- Programming Assignment 1 Due Wednesday @10pm
  - *Multiple files?*
    Please see submission reqs. on syllabus
    + upload as a zipped folder…



Scheduling



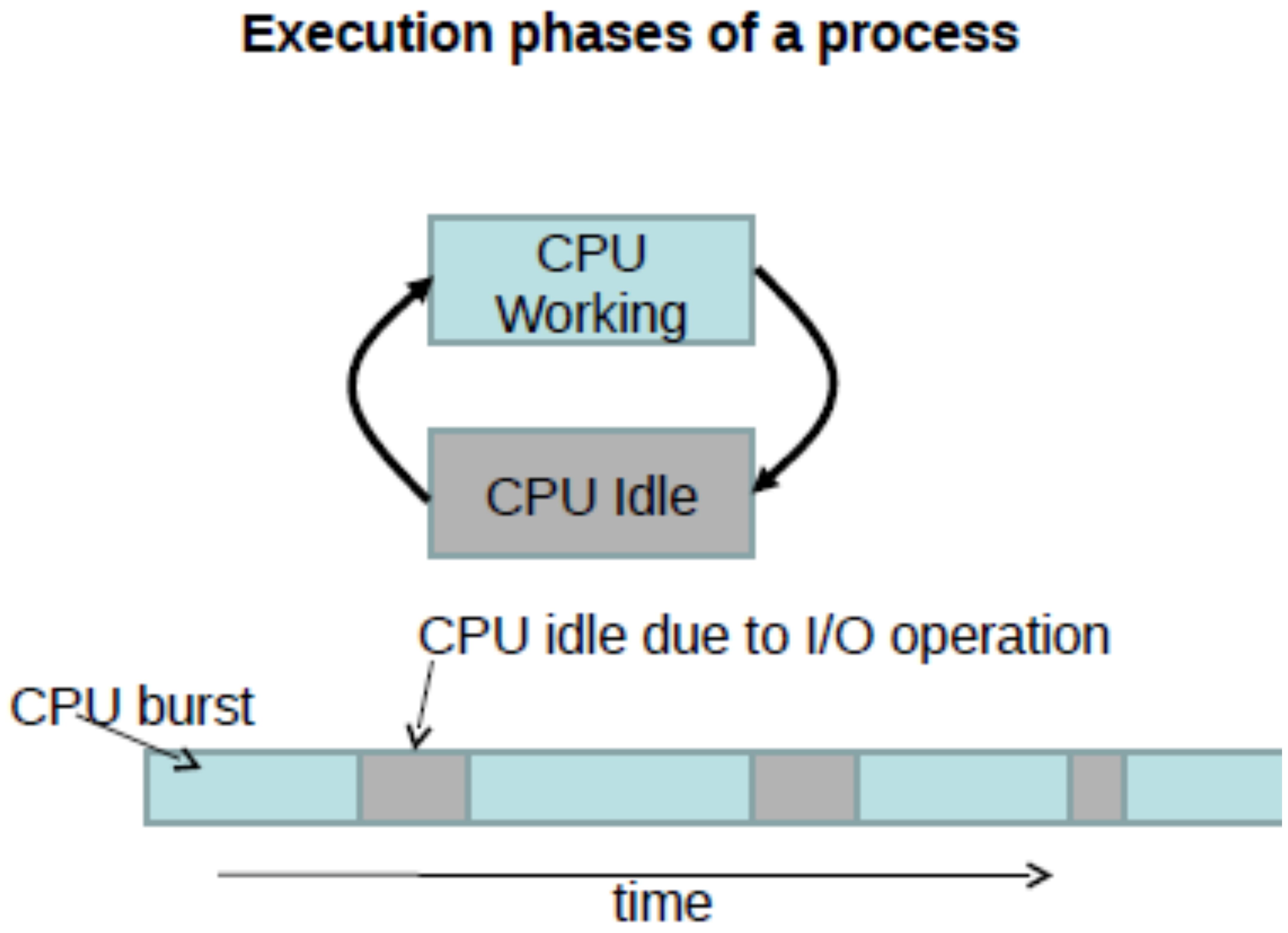


Photo Credit: https://www.cse.iitm.ac.in/~chester/courses/15o_os/slides/8_Scheduling.pdf

Types of Processes

I/O Bound

- most work is I/O-related
- small bursts of CPU activity, then waits for I/O
- a process that primarily depends (and wait on) I/O operations
- affects user interaction → should run with high-priority

Processor-Bound

- most work is CPU-related
- hardly any I/O
- a process that primarily performs computational work
- e.g., gcc, scientific modeling, 3D rendering
- suitable for running with lower-priority

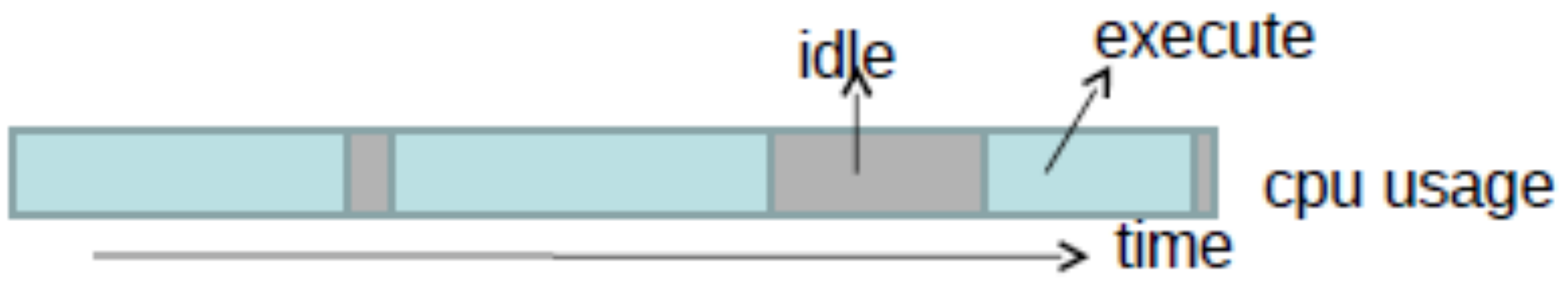


Photo Credit: https://www.cse.iitm.ac.in/~chester/courses/15o_os/slides/8_Scheduling.pdf

CPU Scheduler

- Scheduler triggered to run when timer-based interrupt occurs, or when process is blocked on I/O
- Scheduler picks another process from the Ready Queue (RQ)
  - *Switch to new process via context switch*

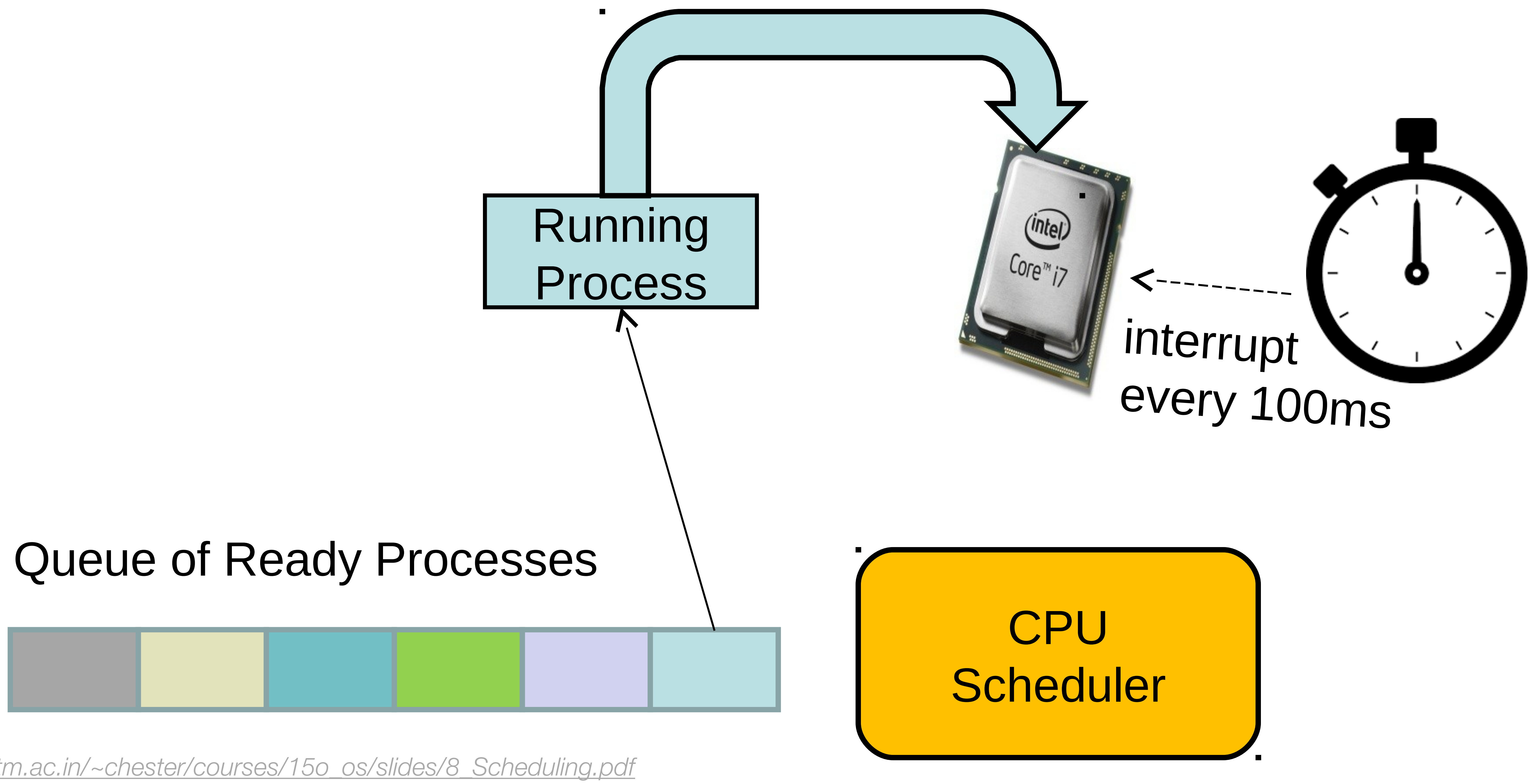
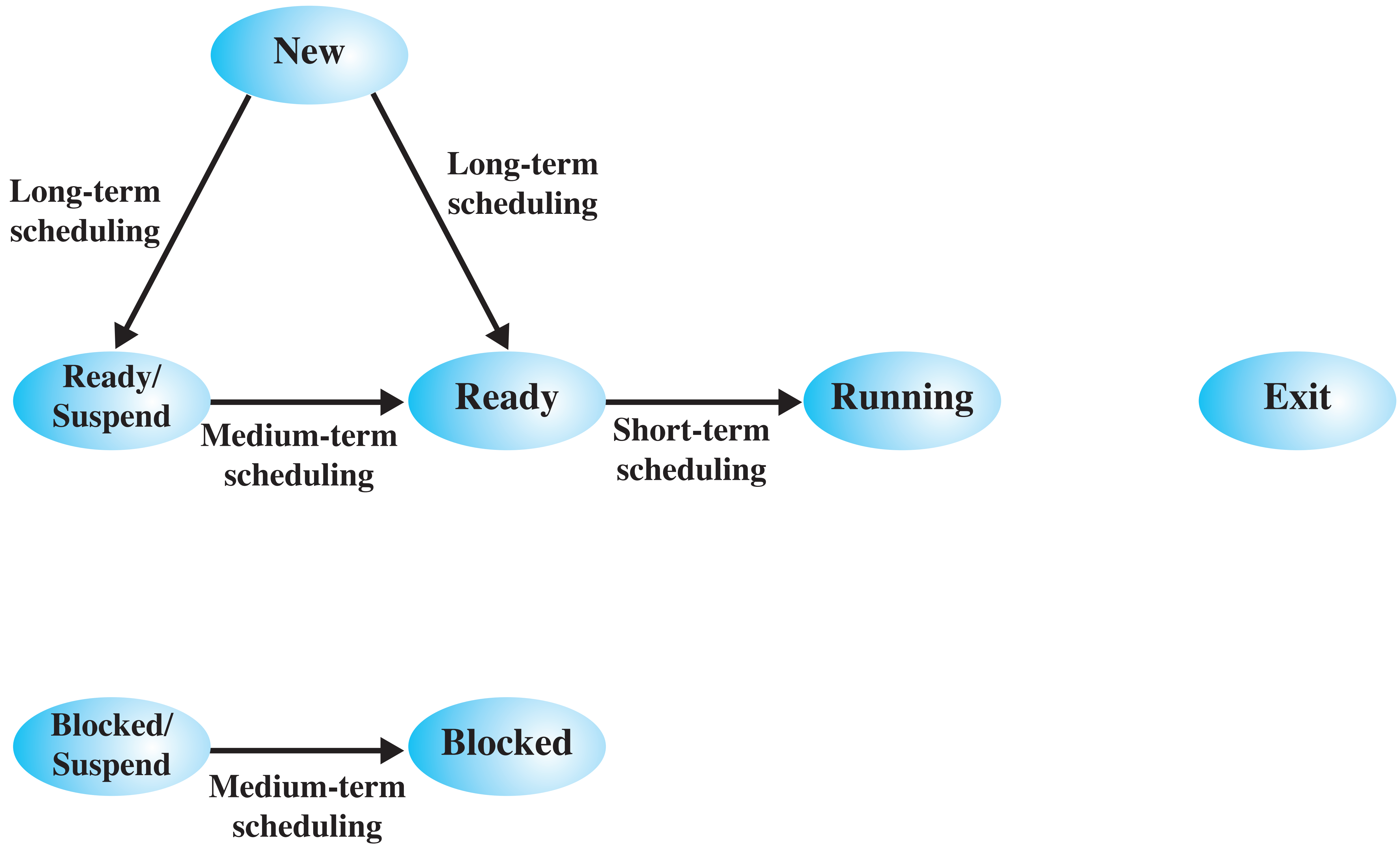
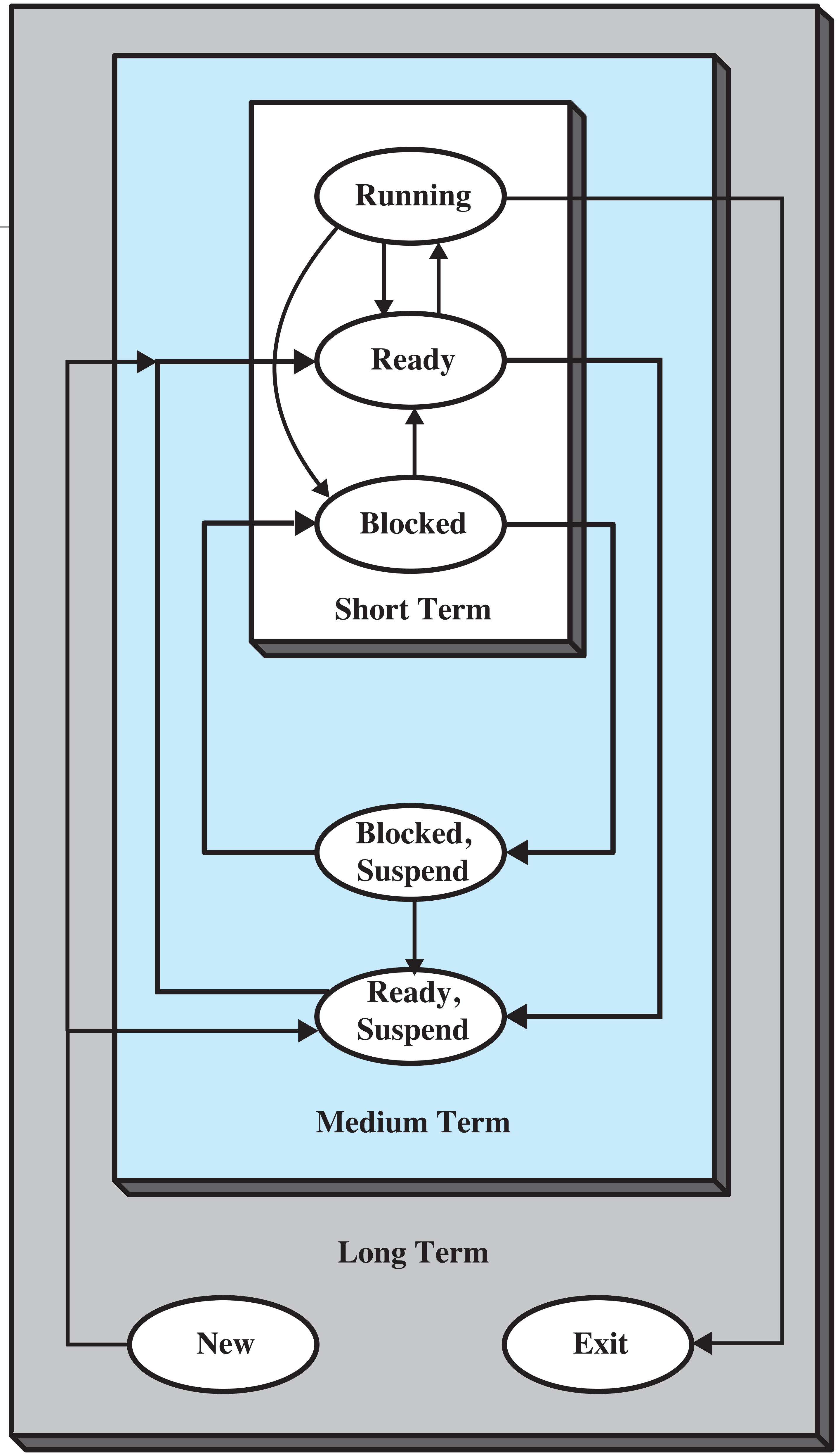


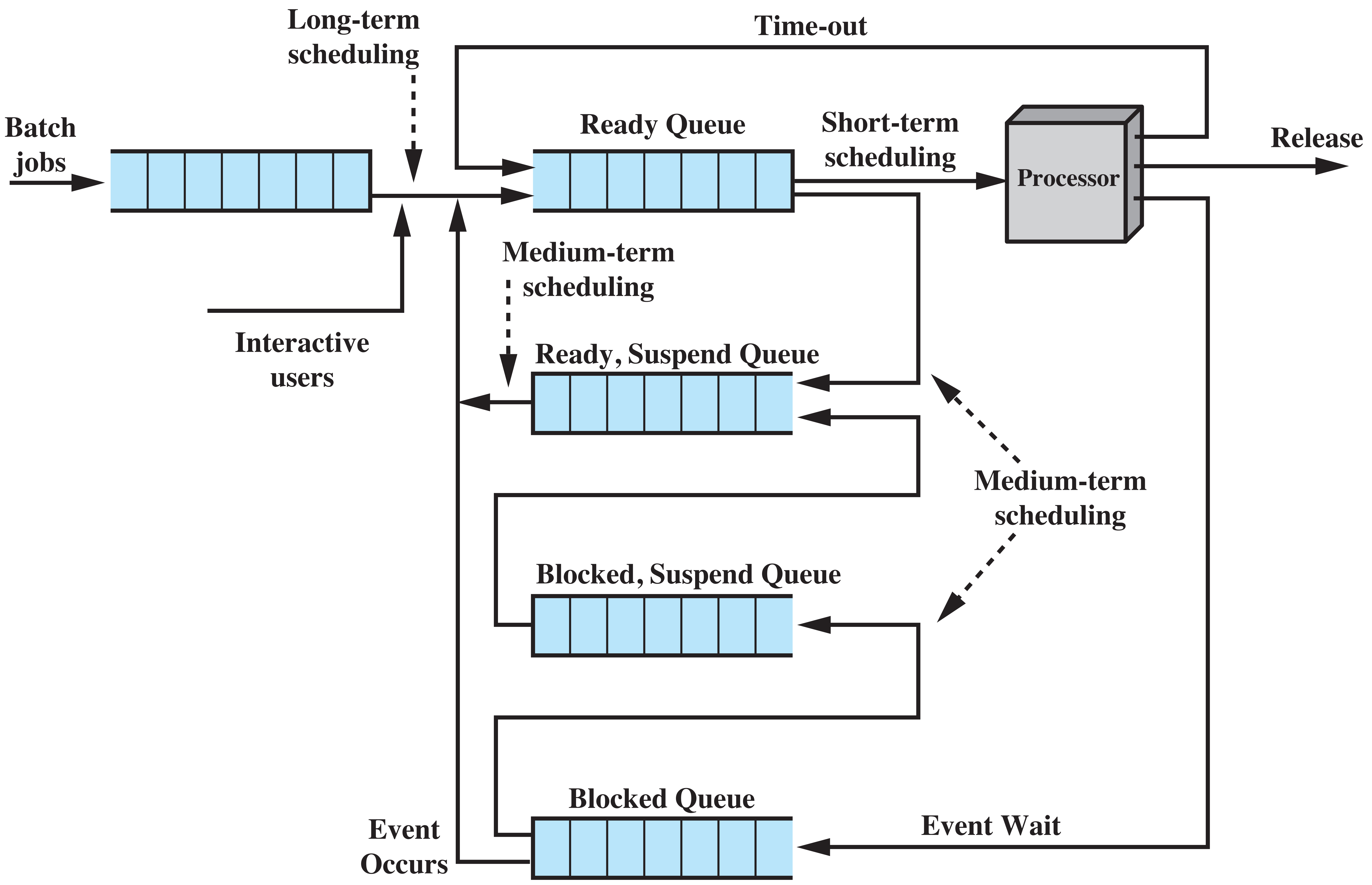
Photo Credit: https://www.cse.iitm.ac.in/~chester/courses/15o_os/slides/8_Scheduling.pdf

Levels of Scheduling





Levels of Scheduling



Scheduling Algorithms

- Criteria for scheduling policies
  - Short-term scheduling
  - ***user***-oriented vs. ***system***-oriented criteria / performance-oriented vs. non-performance-oriented crtiteria
- ***user-***oriented criteria

  define avg. threshold; maximize # of users who experience avg. response time
- ***system-***oriented criteria

  throughput = the rate at which processes are completed
- ***performance-***oriented criteria

  quantitive / measurable (e.g., response time, throughput)
- ***non-performance-***oriented criteria

  qualitative / not easily measurable (e.g., predictability)

  → *calculating/measuring X as a function of workload*

Scheduling Algorithms *(cont.)*

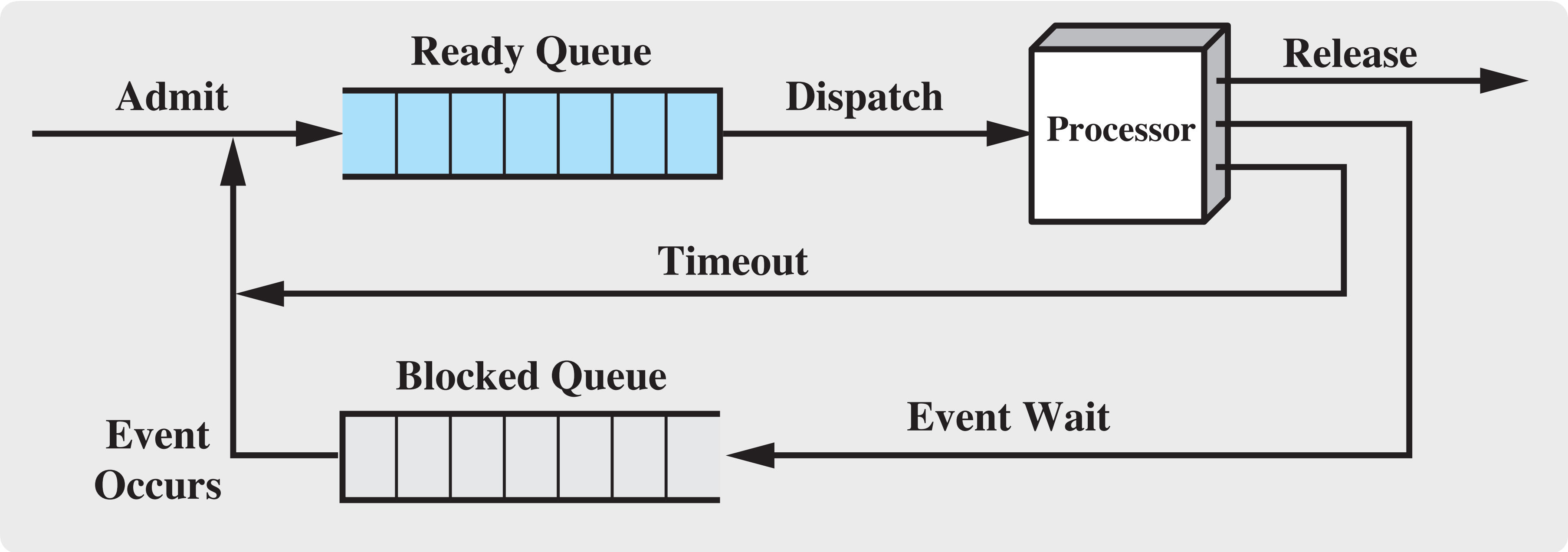
Summary of Scheduling Metrics

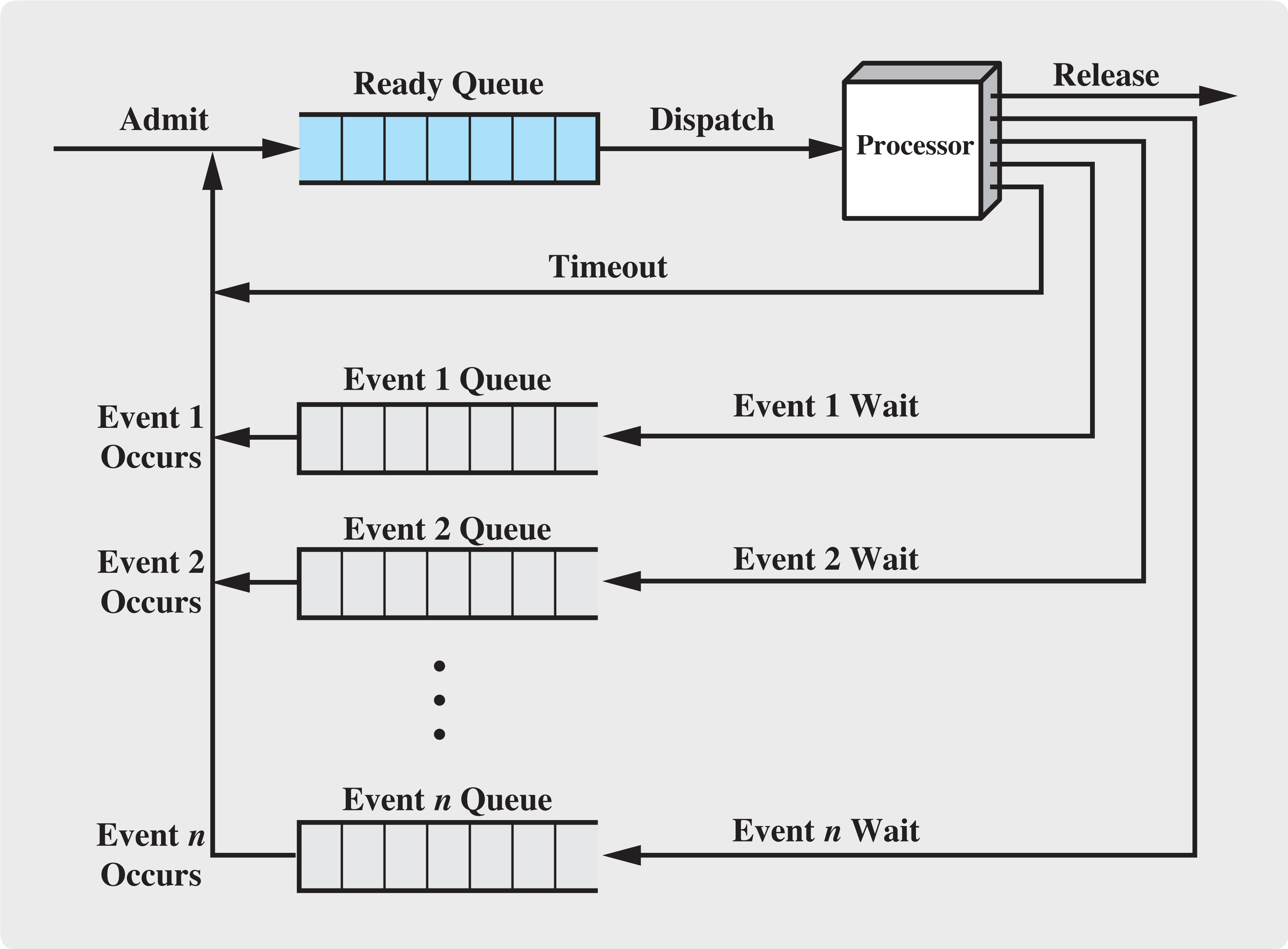
CPU Utilization Amount of time the CPU spends doing productive work *(utilization = 1 – idle_time)*	
Throughput # of processes that are completed per time unit *(e.g., 54 procs per second)*	
Turnaround Time Amount of time from process arrival to process completion *(TT = completion - arrival)*	
Response Time How quickly a process starts running after submission *(response_time = start - arrival)*	

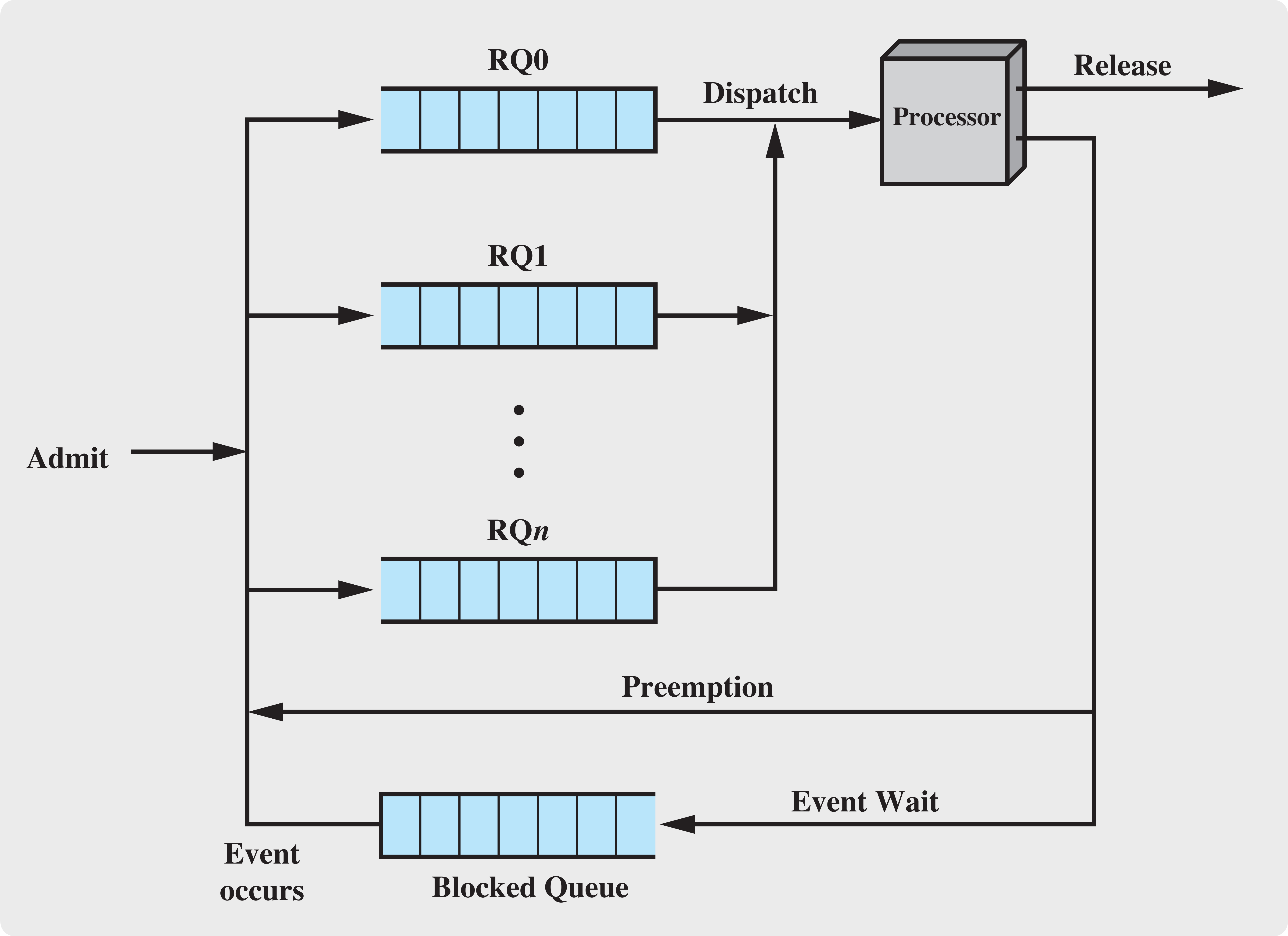
Scheduling Terms

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.	
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.	
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.	
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.	

Using Priorities for Scheduling







Characteristics of Various Scheduling Policies

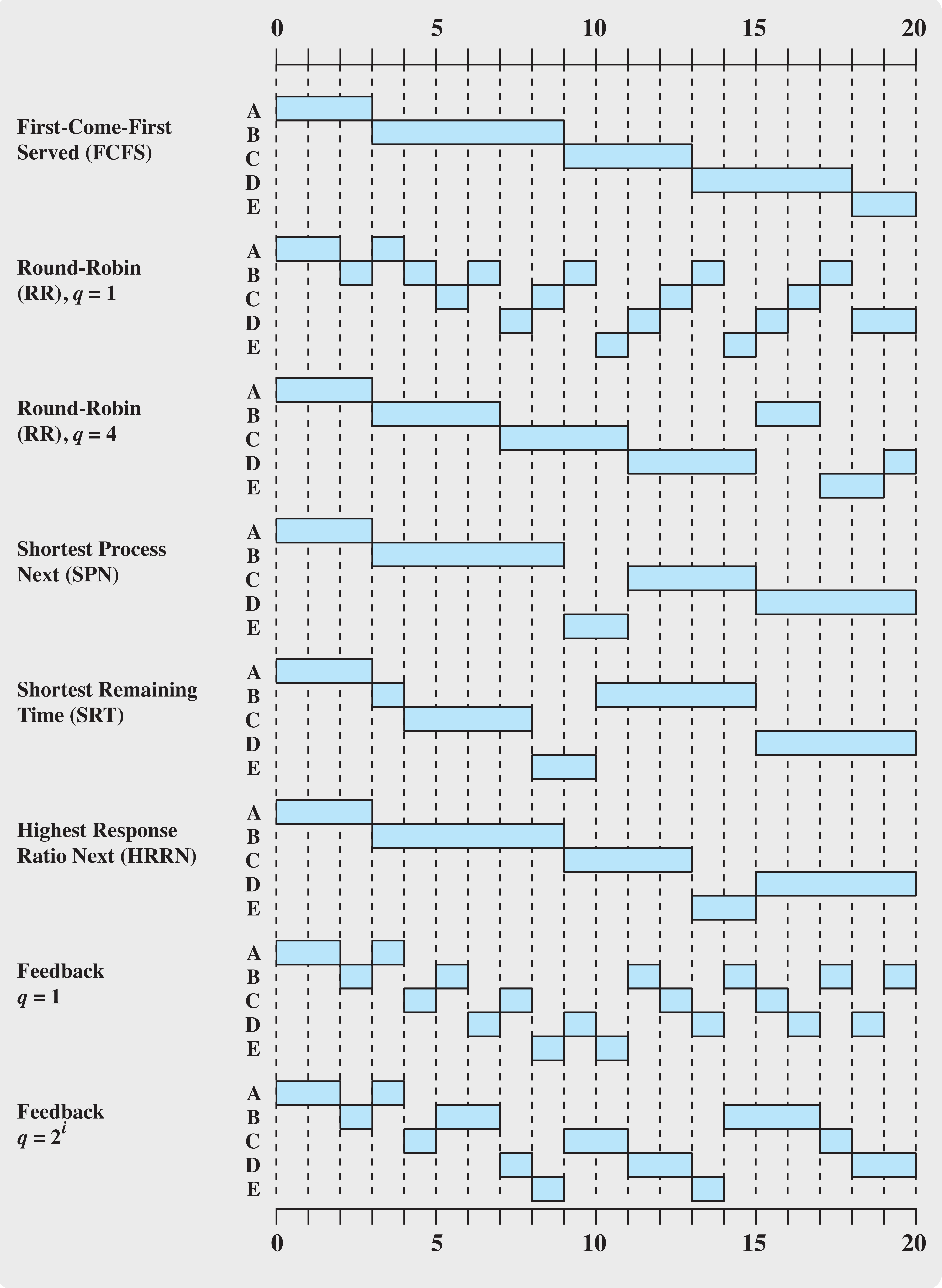
key
w = time spent waiting
e = time spent executing
s = total service time

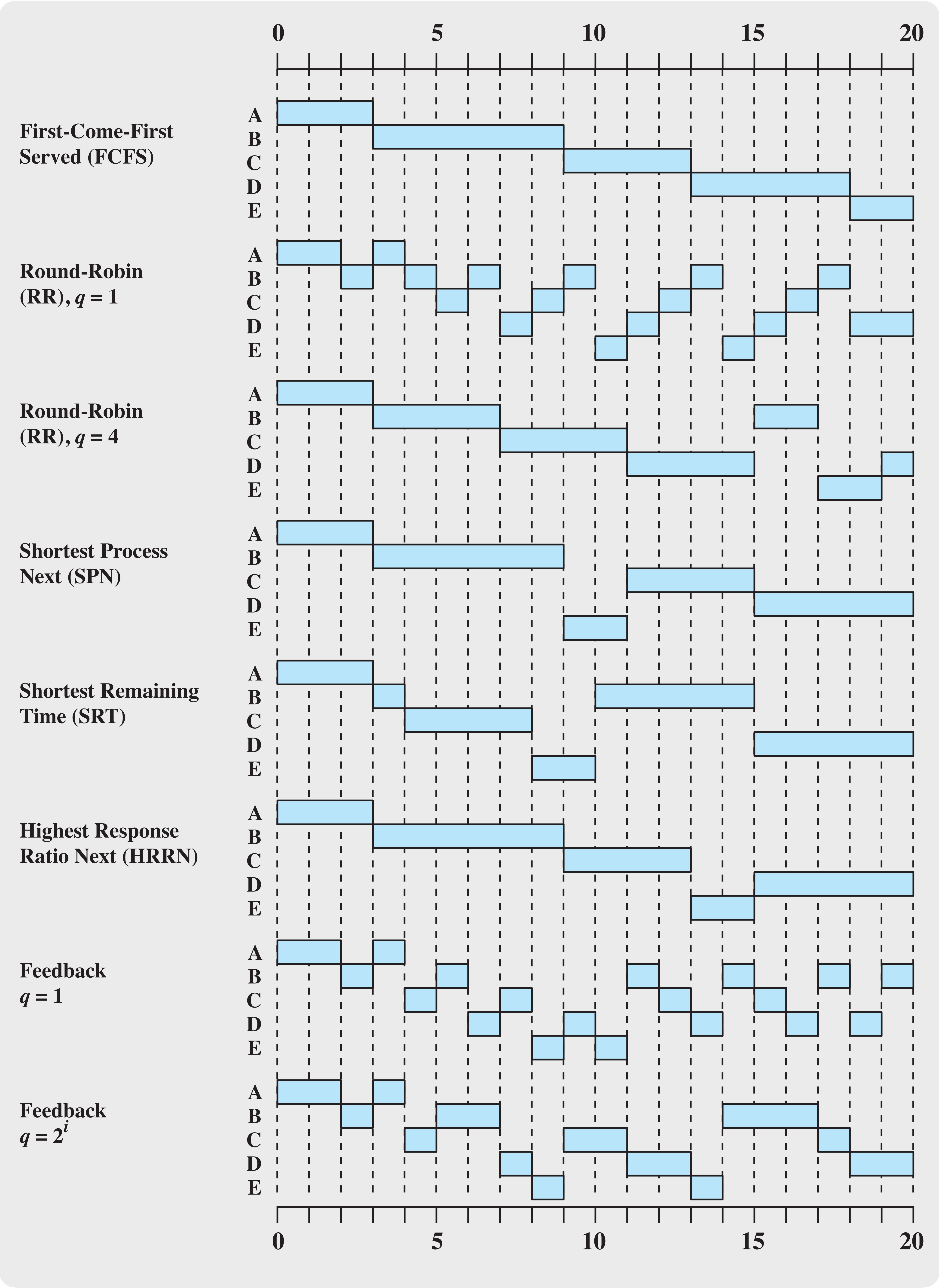
	FCFS	Round robin	SPN	SRT	HRRN	Feedback
Selection Function	max$[w]$	constant	min$[s]$	min$[s - e]$	max$\left(\frac{w+s}{s}\right)$	(see text)
Decision Mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive	Preemptive (at time quantum)
Throughput	Not emphasized	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

Comparing Scheduling Algorithms

- Example
  - 5 processes
  - service times *or* 1 cycle of ongoing processes

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

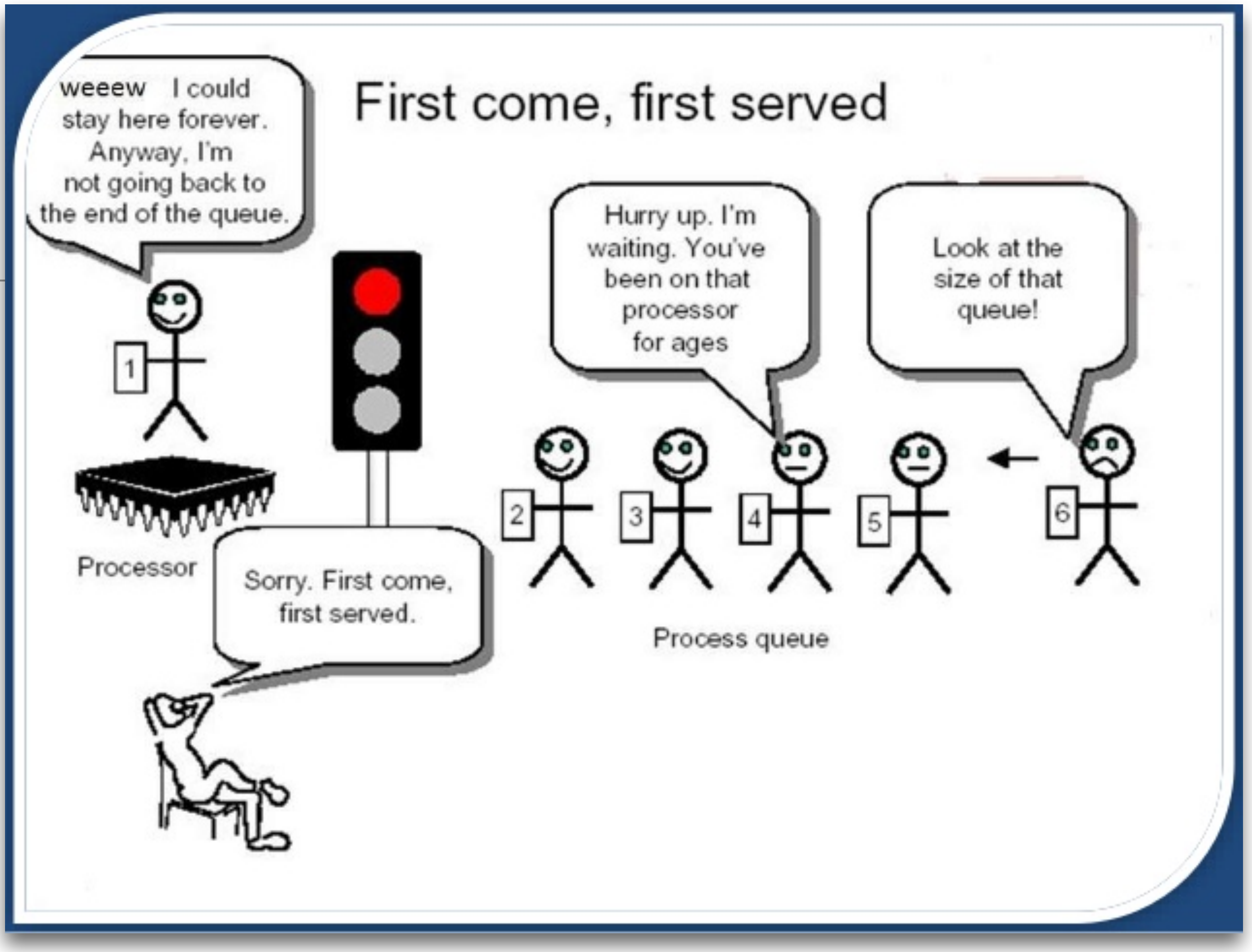




Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	**Mean**
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	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

First-Come-First-Served (FCFS)

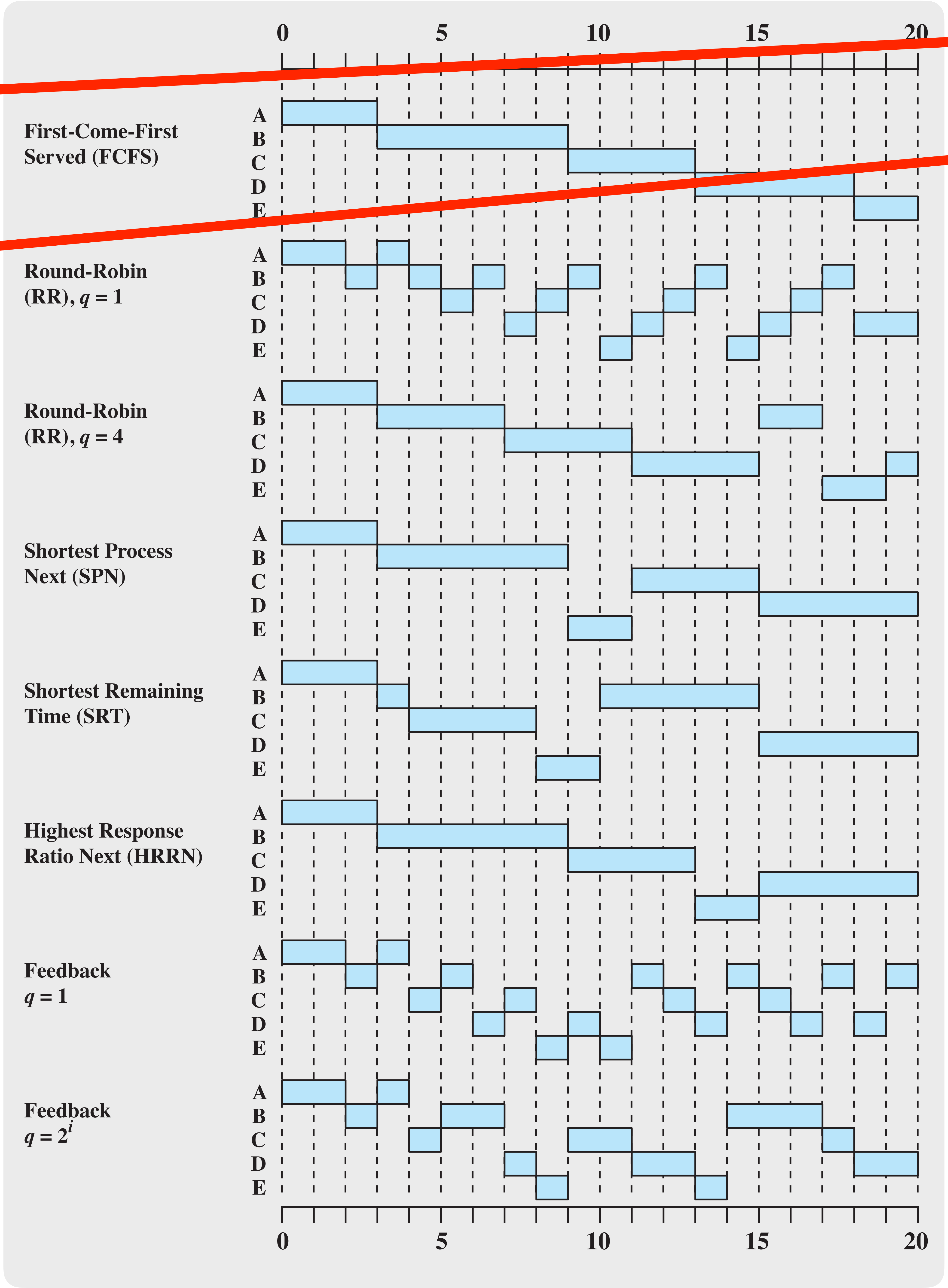
Select the process that has been waiting the longest for service



- Strict queueing scheme (a.k.a. FIFO)
- The process that has been in the ready queue the longest is selected next

- **Q:** What happens when a short process arrives just after a long process?

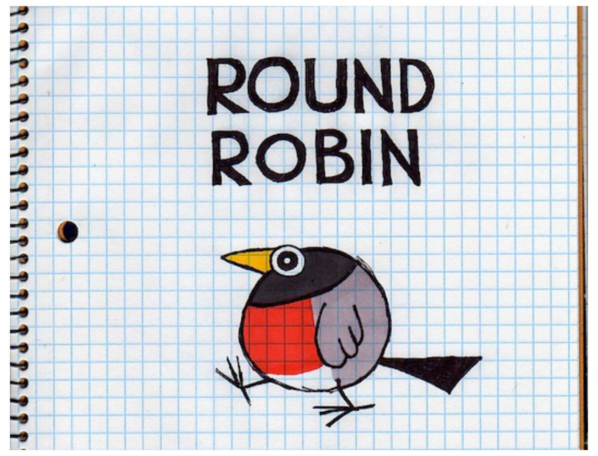
- Not a great solution on its own for a uniprocessor system
  - Tends to favor processor-bound processes over I/O-bound processes
  - Often combined with a priority scheme!
    E.g., a number of queues for different priorities; dispatch within each queue on a FCFS basis



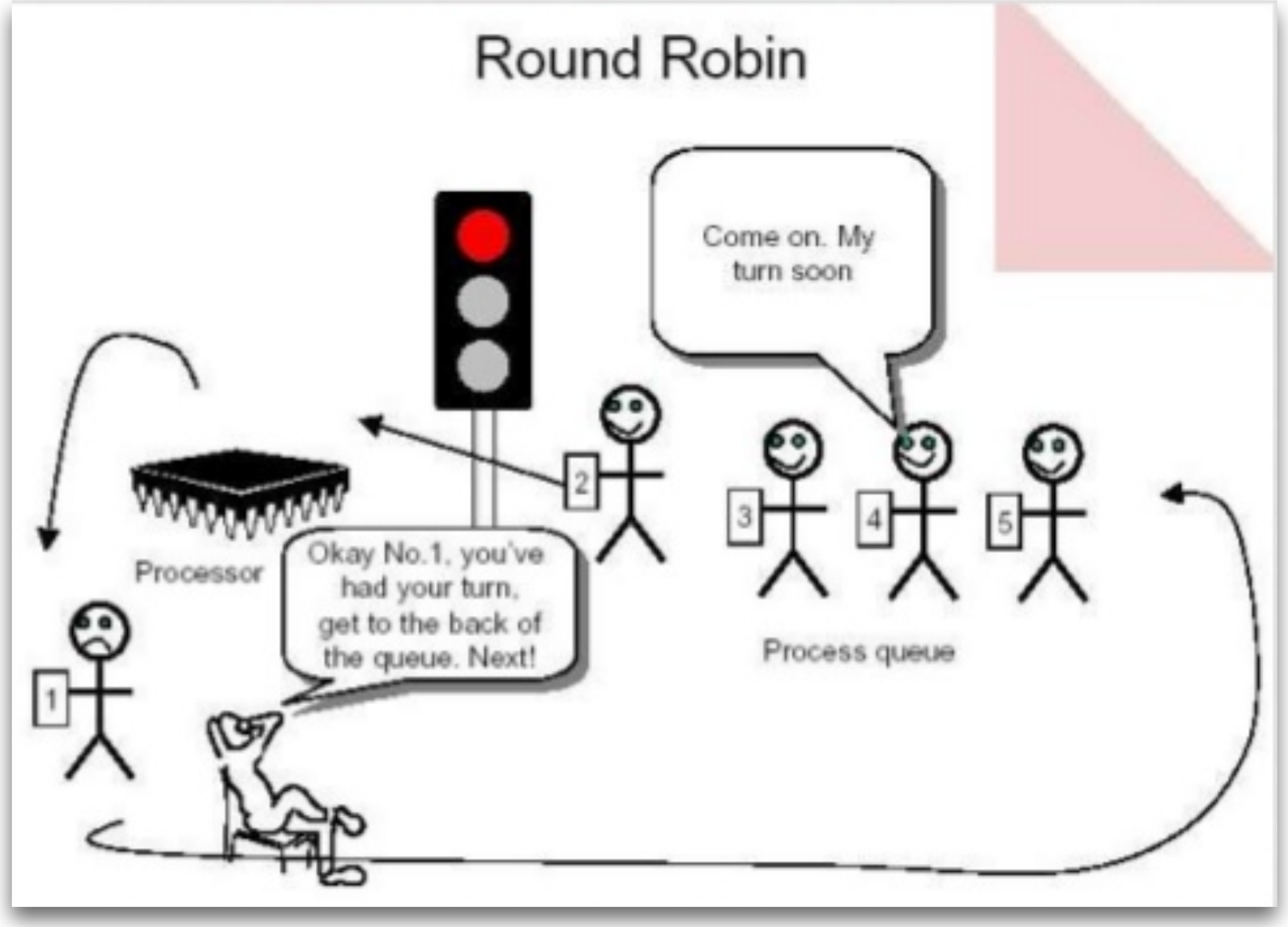
Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	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

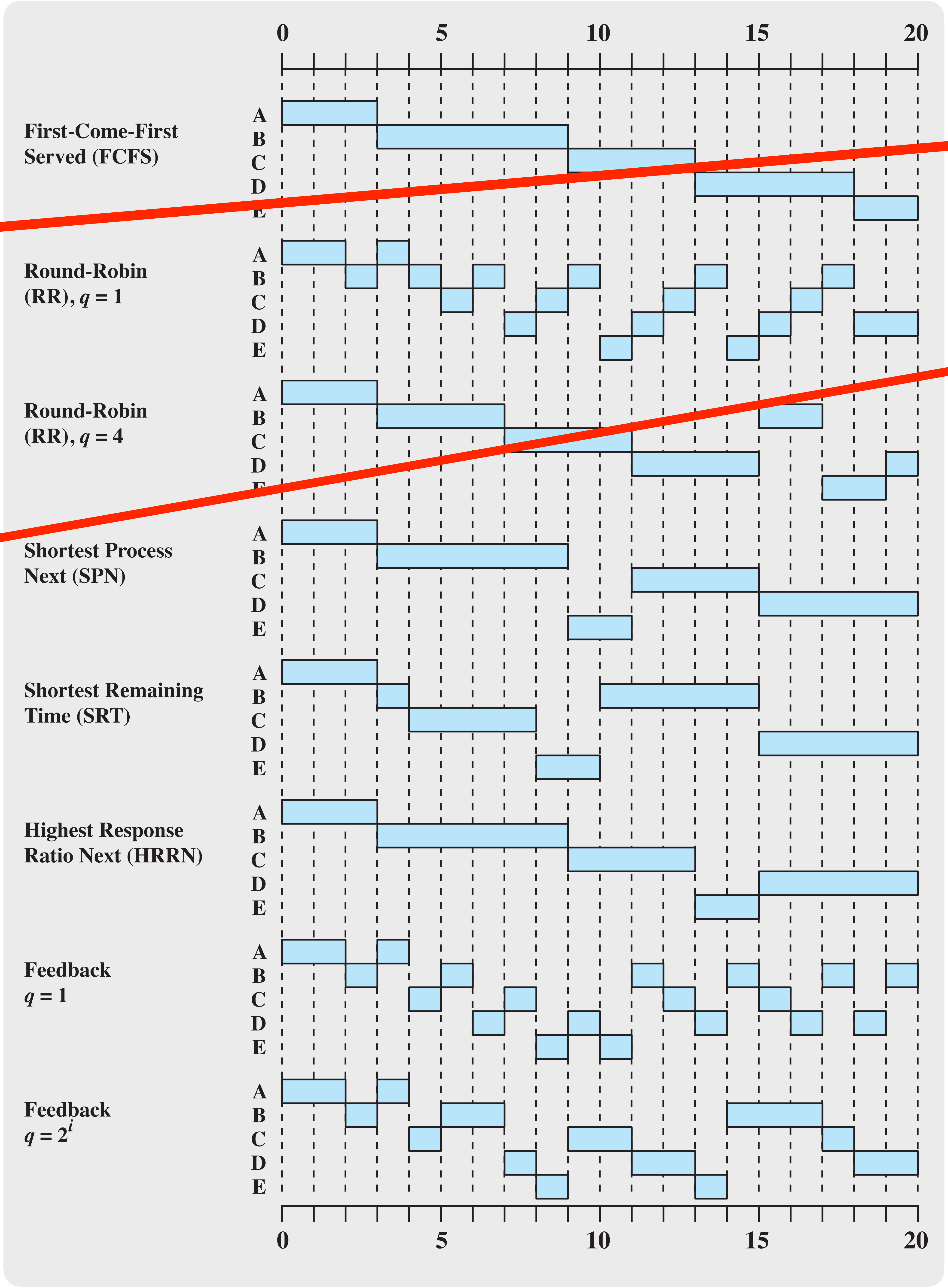
Round Robin (RR)

Use time slicing to limit any running process to a short burst of processor time; rotate among all processes



- Use preemption based on a clock!
  - clock-based interrupt == periodic intervals of execution
  - preempt a process → put it on the ready queue → select next ready proc. on a FCFS basis
  - a.k.a ***time-slicing***
- Principle design choice for RR → length of the ***time quantum (slice)***
- **Q:** Should we prefer small slices or big slices?
- Quite effective in general purpose time-sharing systems!
  - Any Drawbacks? *(Re: processor-bound vs. I/O-bound processes?)*

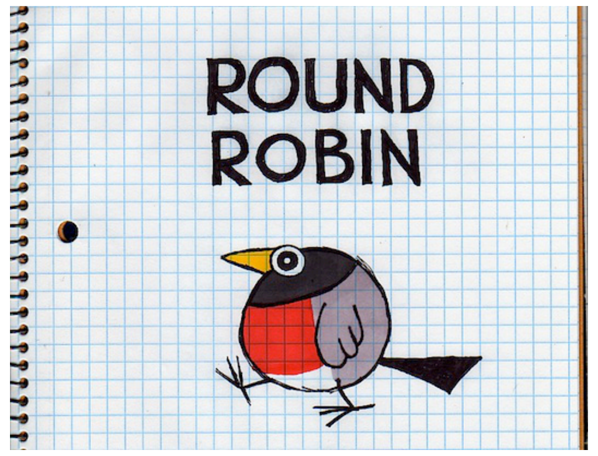




Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	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

Refinement: Virtual Round Robin (VRR)

Use time slicing to limit any running process to a short burst of processor time; rotate among all processes



- Same as before… plus…
  - Introduce **Auxiliary Queue (AQ)** to hold recently un-blocked I/O-bound processes.
  - Processes in AQ get preference over processes in normal **Ready Queue (RQ).**
  - Each proc gets a chance to complete its time quanta.
    i.e., time quanta - total time spent running since it was last selected.