Principles of Operating Systems

CS 446/646

4. CPU Scheduling

René Doursat

Department of Computer Science & Engineering
University of Nevada, Reno

Spring 2006

Principles of Operating Systems

CS 446/646

0. Course Presentation
1. Introduction to Operating Systems
2. Processes
3. Memory Management
4. **CPU Scheduling**
5. **Input/Output**
6. **File System**
7. **Case Studies**

Principles of Operating Systems

CS 446/646

4. CPU Scheduling

a. **Concepts of Scheduling**

b. **Scheduling Algorithms**

c. **Queuing Analysis**

d. **Thread Scheduling**

Principles of Operating Systems

CS 446/646

4. CPU Scheduling

a. Concepts of Scheduling

- ✓ Three-level scheduling
- ✓ Purpose of CPU scheduling

b. Scheduling Algorithms

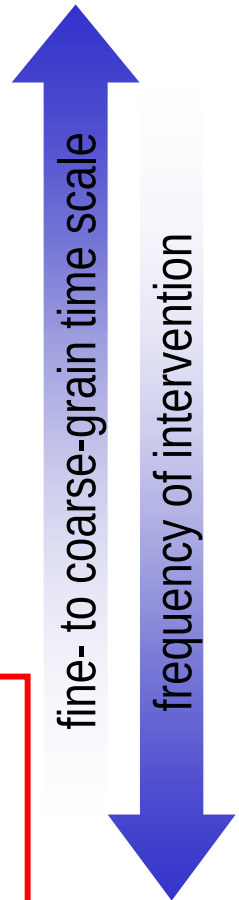
c. Queuing Analysis

d. Thread Scheduling

4.a Concepts of Scheduling

Three-level scheduling

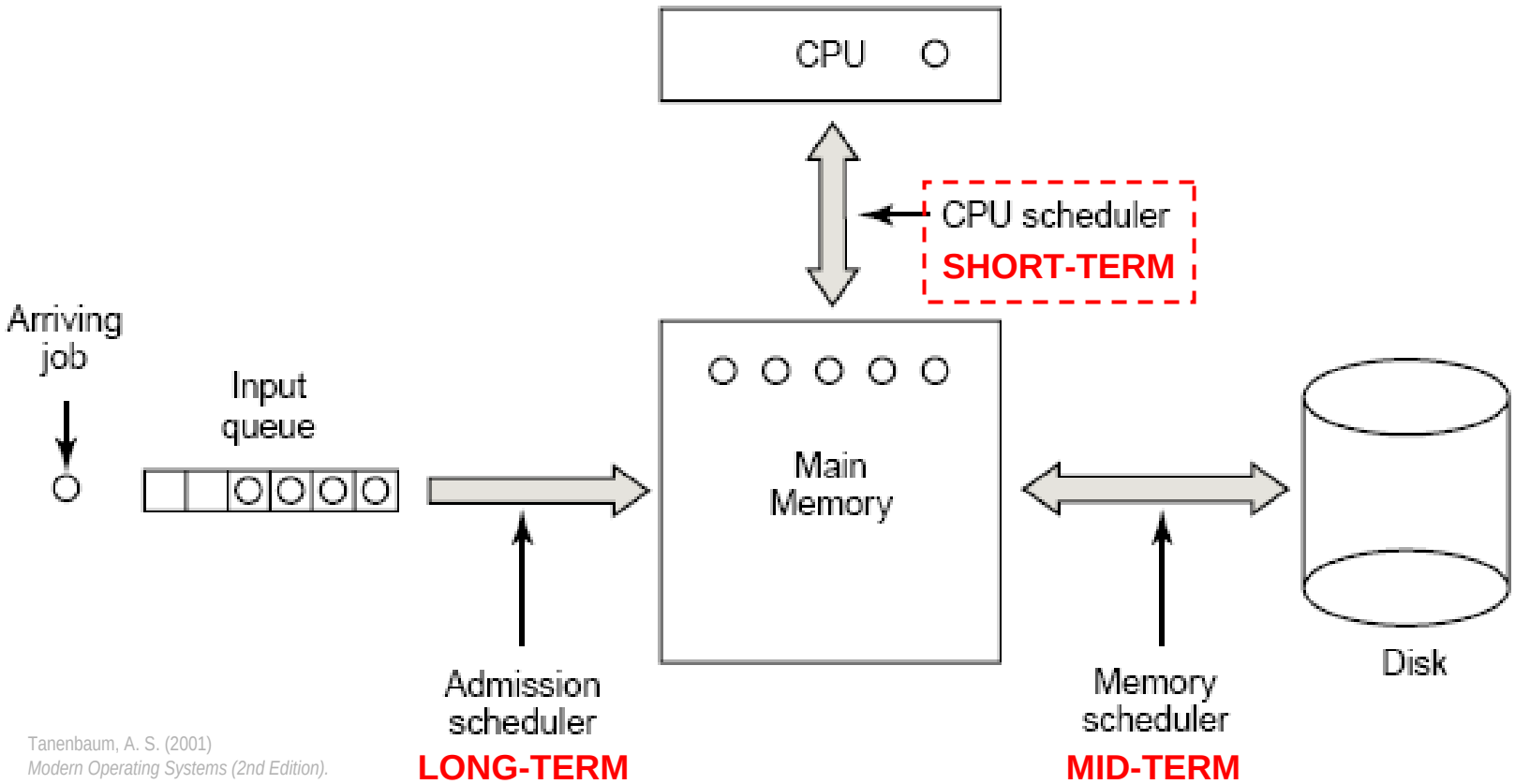
- Long-term scheduling (mostly in batch)
  - ✓ the decision to add a program to the pool of processes to be executed: controls the degree of multiprogramming
- Medium-term scheduling
  - ✓ the decision to add to the number of processes that are partially or fully in main memory ("swapping")
  - ✓ *not* the same as paging: swapping out means removing all the pages of a process
- Short-term scheduling = CPU scheduling
  - ✓ the decision as to which available processes in memory are to be executed by the processor ("dispatching")



4.a Concepts of Scheduling

Three-level scheduling

- Three-level scheduling



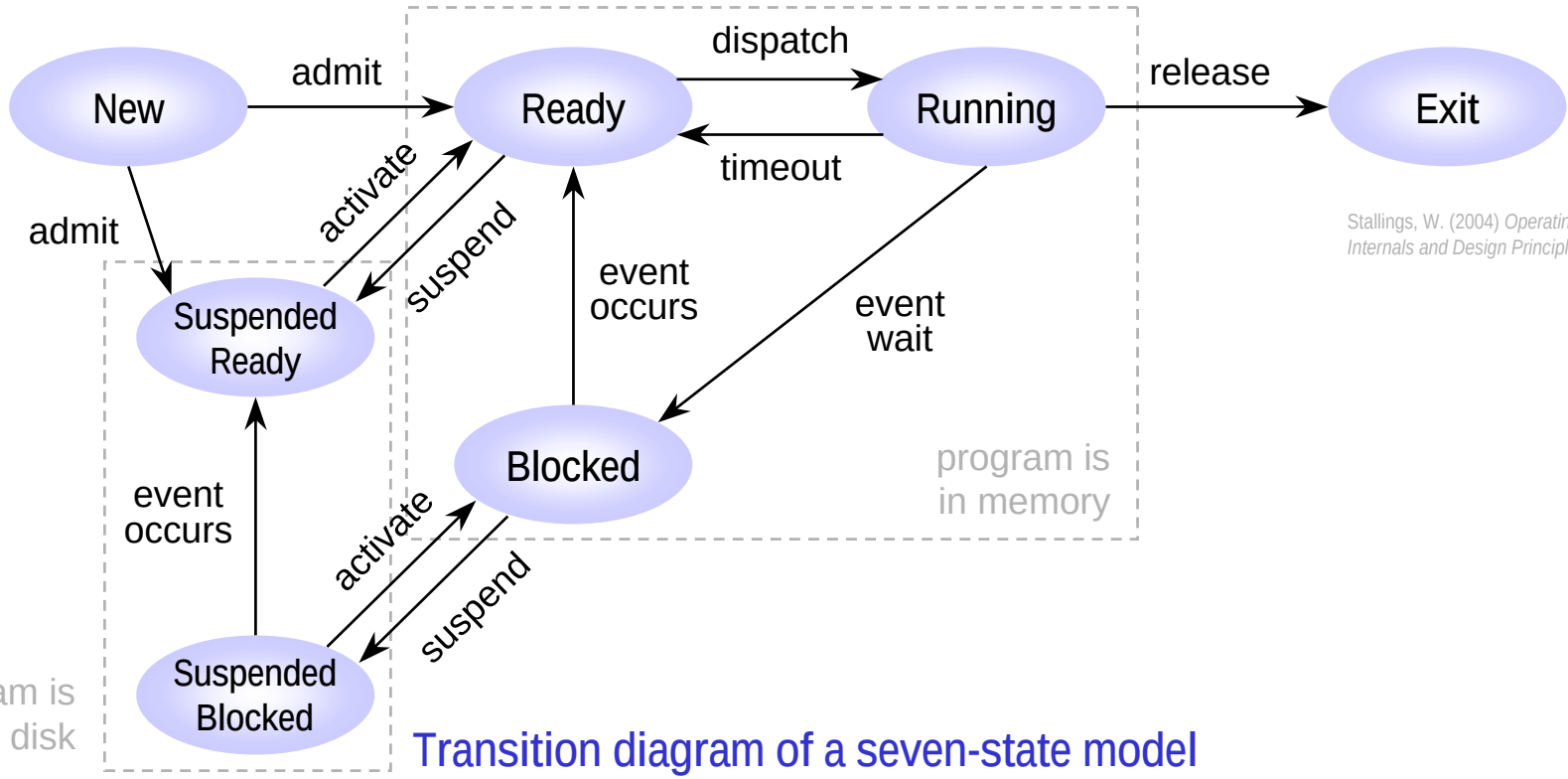
Tanenbaum, A. S. (2001)
Modern Operating Systems (2nd Edition).

Three-level scheduling

4.a Concepts of Scheduling

Three-level scheduling

- Reminder: process states



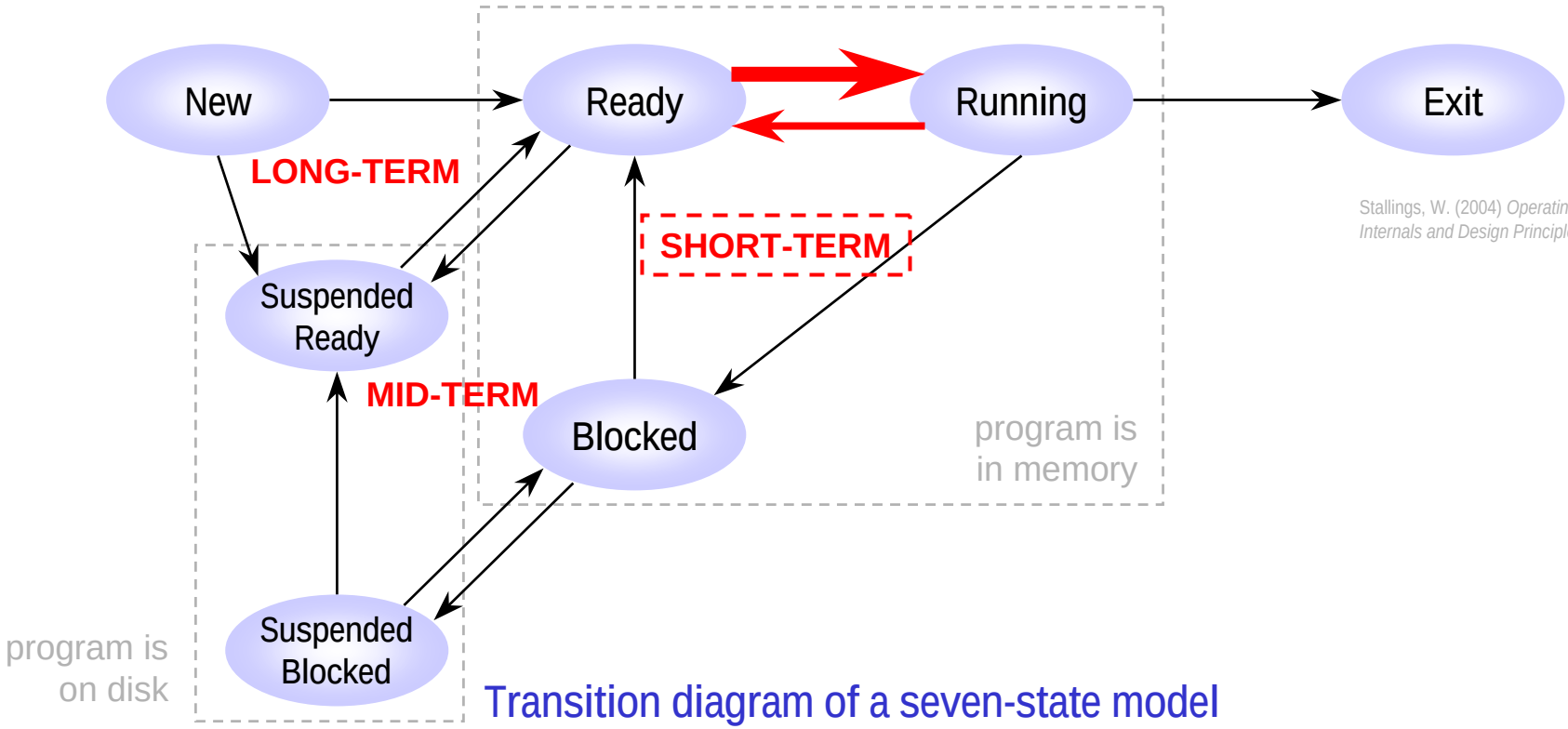
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Transition diagram of a seven-state model

4.a Concepts of Scheduling

Three-level scheduling

- In the O/S, the CPU scheduler decides which "Ready" process to run next (and which "Running" to time out)
  - ✓ the discipline it follows is the **scheduling algorithm**



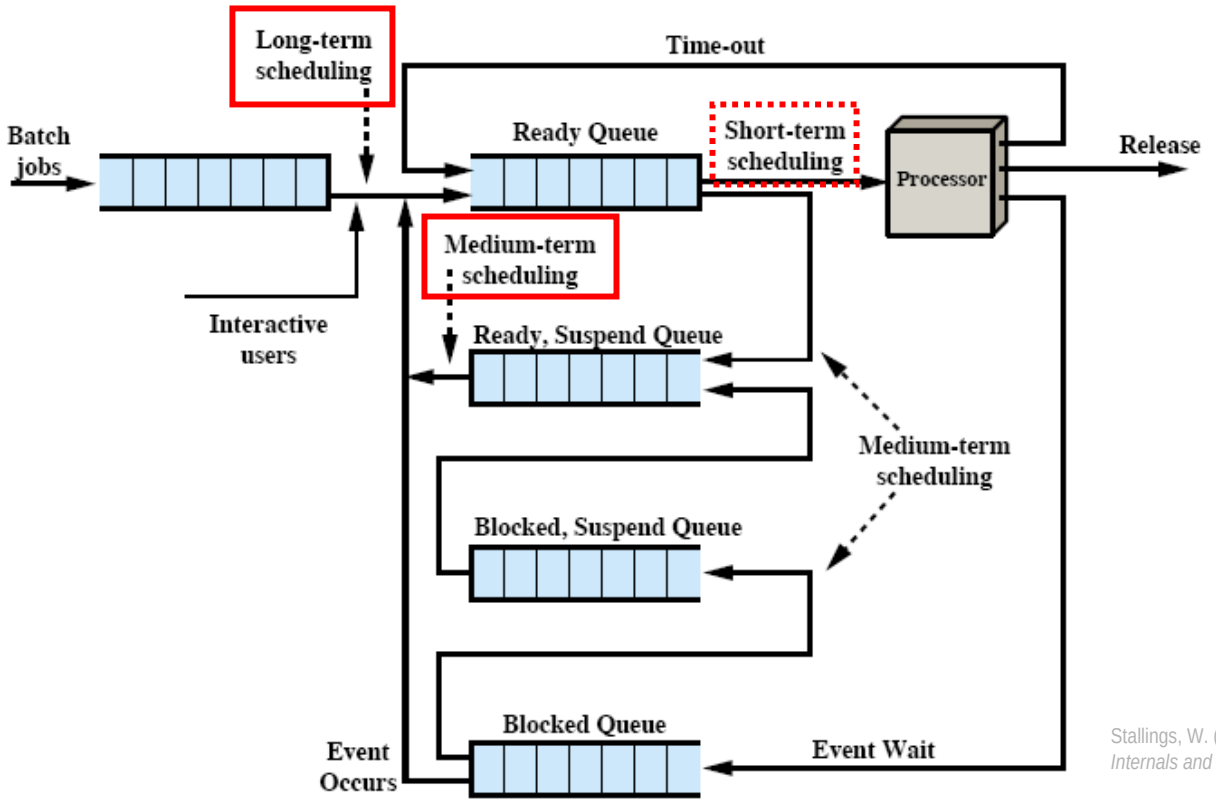
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Transition diagram of a seven-state model

4.a Concepts of Scheduling

Three-level scheduling

- General queuing system for scheduling
  - ✓ in most algorithms, queues are not strictly FIFO: rather “pools”



Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Queuing diagram for scheduling

4.a Concepts of Scheduling

Purpose of CPU scheduling

- Why scheduling matters: user service response
  - ✓ example: choosing between
    - a process that updates the screen after the user has closed a window
    - a process that sends out queued email
  - ✓ taking 2 seconds to close the window while sending the email would be unacceptable
  - ✓ on the other hand, delaying the email while closing the window would hardly be noticed
  - → *schedule wisely to match user's expectations*

4.a Concepts of Scheduling

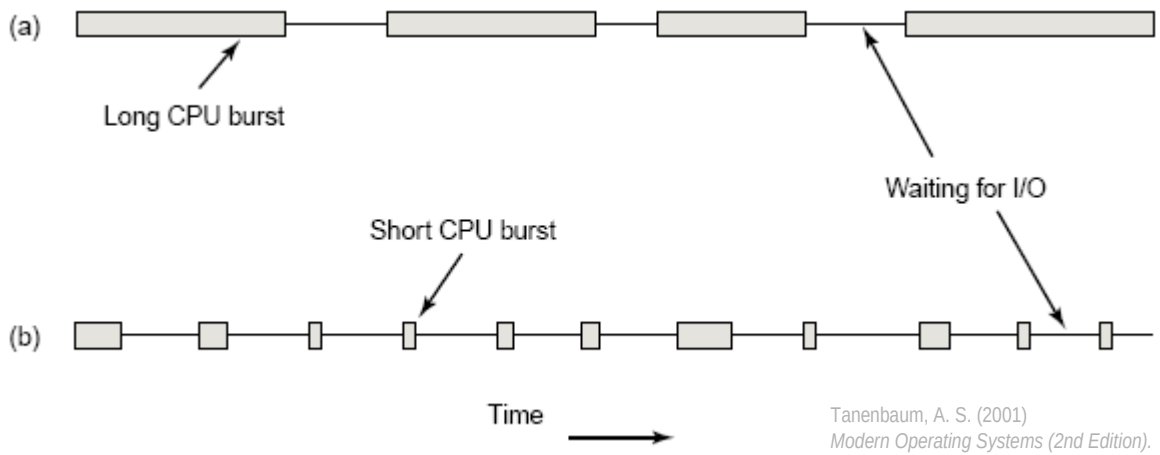
Purpose of CPU scheduling

- Why scheduling matters: CPU usage
  - ✓ switching processes (contexts) is heavy
    - switch from user mode to kernel mode
    - CPU state must be saved
    - process state must be saved
    - pages and page bits must be saved
    - MMU must be reloaded with new page table
    - etc.
  - → *to maximize CPU utilization, interleave but at the same time minimize process switches*

4.a Concepts of Scheduling

Purpose of CPU scheduling

- Types of process behavior: CPU-I/O burst cycle
  - ✓ processes alternate CPU usage with I/O wait
    - compute-bound processes have long CPU bursts and infrequent I/O
    - I/O-bound processes have short CPU bursts and frequent I/O

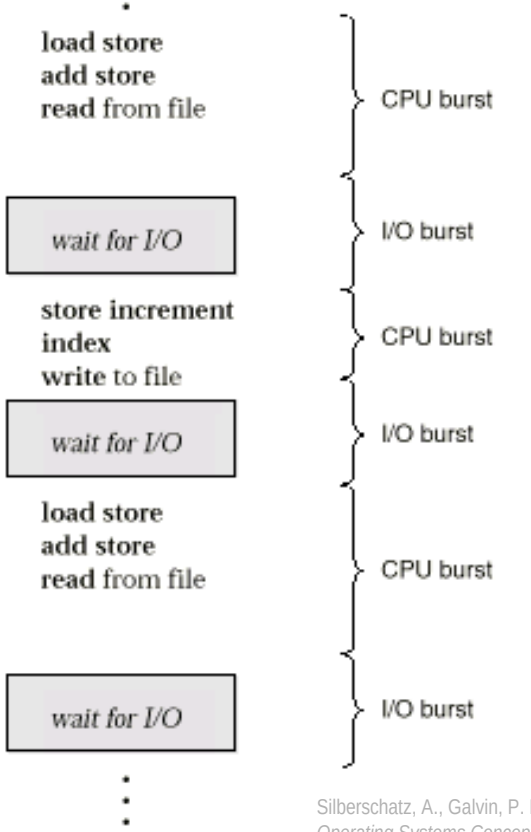


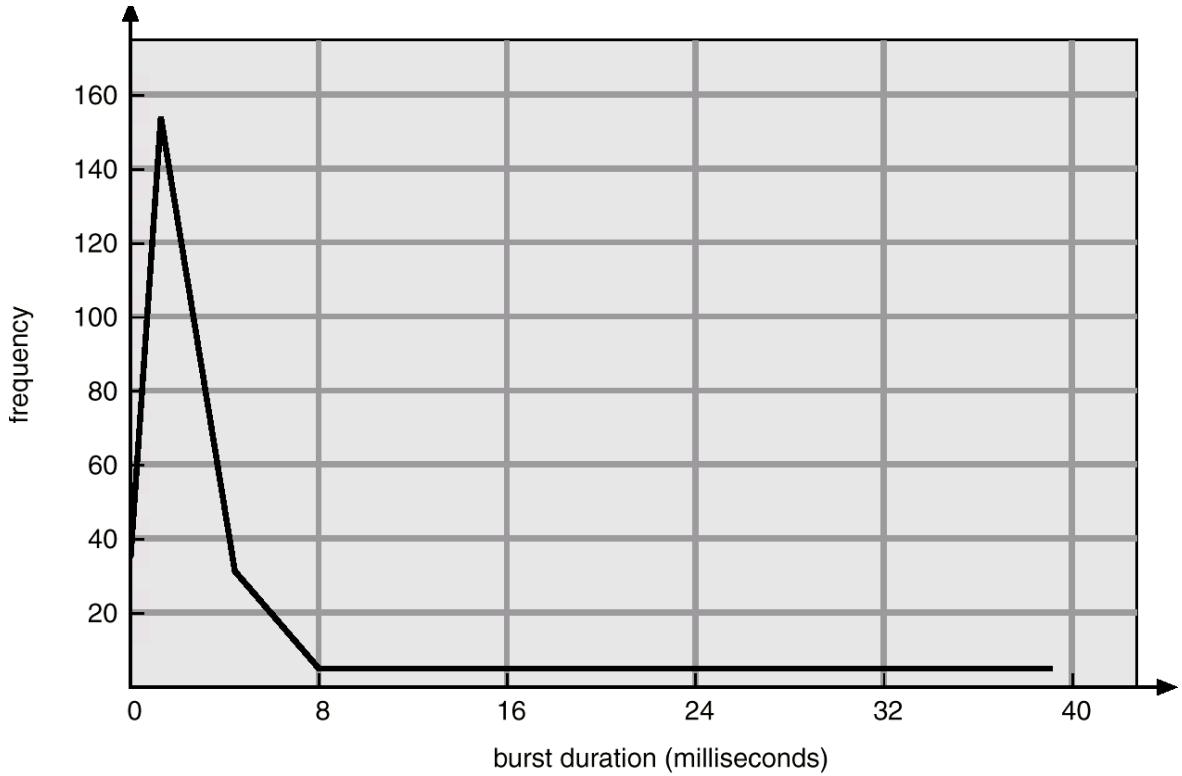
(a) Compute-bound process vs. (b) I/O-bound process

4.a Concepts of Scheduling

Purpose of CPU scheduling

- Types of process behavior: CPU-I/O burst cycle
  - ✓ power-law: large # of short CPU bursts, small # of large bursts





Silberschatz, A., Galvin, P. B. and Gagne. G. (2003) *Operating Systems Concepts with Java (6th Edition).*

Typical histogram of CPU-burst times

4.a Concepts of Scheduling

Purpose of CPU scheduling

- I/O-bound processes
  - ✓ as CPU speeds increase, processes generally tend to become more and more I/O-bound
  - ✓ the scheduling of I/O-bound processes will likely become an important subject in the future
  - → *basic idea: an I/O-bound process that is "Ready" to run should get the CPU quickly so it can keep the disk busy*

4.a Concepts of Scheduling

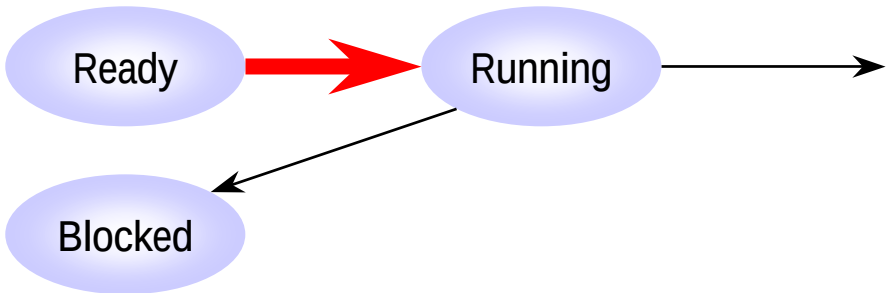
Purpose of CPU scheduling

- When scheduling decisions must be made
  - ✓ when a new process is created — run the child or the parent?
  - ✓ when a process exits — who's next?
  - ✓ when an I/O interrupt occurs upon finishing an I/O task — should the waiting process be rescheduled? or let the currently running process continue? or pick another process? etc.
  - ✓ when a timeout (clock interrupt) occurs — who's next?

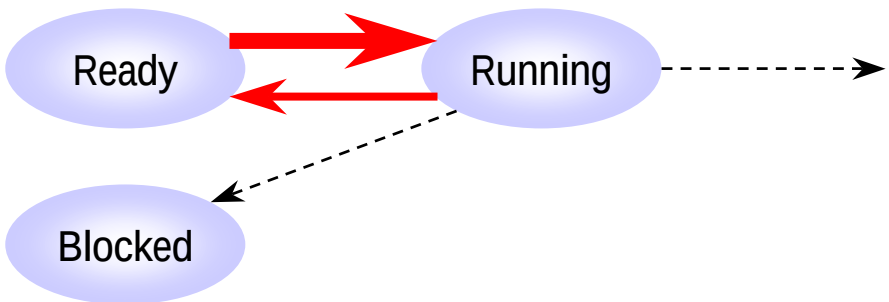
4.a Concepts of Scheduling

Purpose of CPU scheduling

- Two kinds of CPU-scheduling algorithms
  - ✓ **cooperative** scheduling — let a process run until it blocks on I/O, terminates or voluntarily releases the CPU (system call)



  - ✓ **preemptive** scheduling — follow clock interrupts (ex: 50Hz) to forcibly switch processes (demote the “Running” to “Ready”)



4.a Concepts of Scheduling

Purpose of CPU scheduling

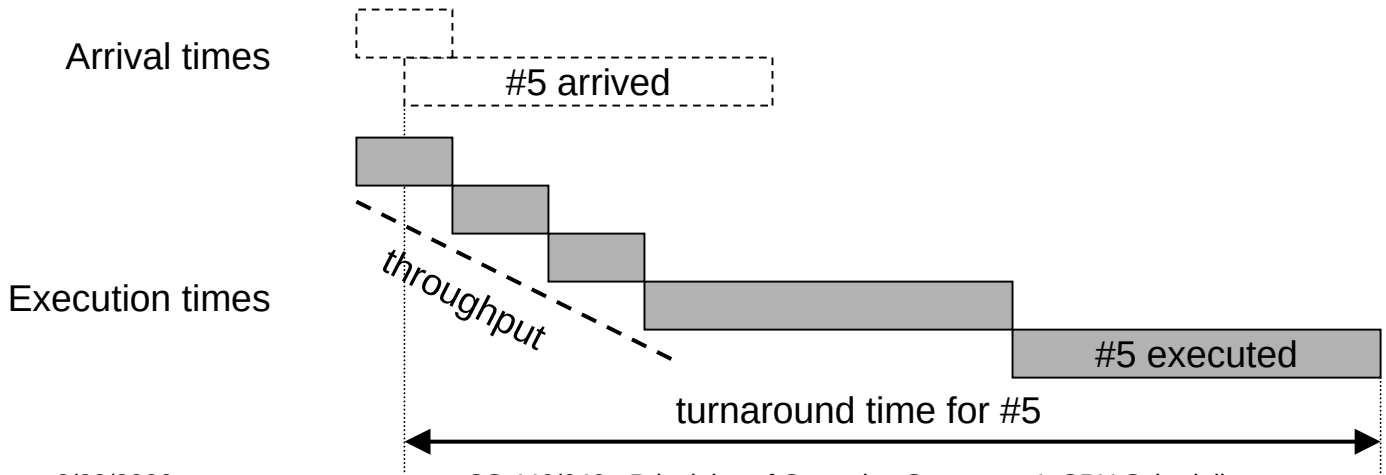
- Scheduling algorithm goals — all systems (batch & int.)
  - ✓ **fairness** — comparable processes get comparable service
  - ✓ **compliance to system's policy** — ex: high-priority override low-priority processes (ex: safety control vs. payroll in a nuclear plant)
  - ✓ **keep system busy** — CPU and I/O devices should be utilized fully
    - if all CPU-bound were run first: fight for CPU, I/O idle
    - then all I/O-bound were run: fight for I/O, CPU idle

    → *keep a well-balanced mix of CPU-bound and I/O-bound processes, so they can fill in for each other*

4.a Concepts of Scheduling

Purpose of CPU scheduling

- Scheduling algorithm goals — batch systems
  - ✓ **throughput** — maximize # of completed jobs per time unit
  - ✓ **turnaround time (latency)** — minimize time between submission and termination of job
    - high throughput and low turnaround are rarely compatible
    - for ex: supply of short jobs scheduled in front of long jobs: good throughput, bad turnaround time for long jobs



4.a Concepts of Scheduling

Purpose of CPU scheduling

- Scheduling algorithm goals — interactive systems
  - ✓ **response time** — respond to requests quickly: minimize time between issuing command and getting result
    - ex: a user request to start a new program should take precedence over background work
    - having interactive requests go first will be perceived as good service
  - ✓ **proportionality time** — meet users' expectation, even if irrational
    - ex: 45 seconds to establish a modem connection is *perceived* as acceptable, yet 45 seconds to hang up is not
    - whenever possible, take this into account when scheduling

4.a Concepts of Scheduling

Purpose of CPU scheduling

- Scheduling algorithm goals — summary

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

Tanenbaum, A. S. (2001)
Modern Operating Systems (2nd Edition).

Real-time systems

Meeting deadlines - avoid losing data
Predictability - avoid quality degradation in multimedia systems

Some goals of CPU scheduling under different circumstances

Principles of Operating Systems

CS 446/646

4. CPU Scheduling

a. Concepts of Scheduling

- ✓ Three-level scheduling
- ✓ Purpose of CPU scheduling

b. Scheduling Algorithms

c. Queuing Analysis

d. Thread Scheduling

Principles of Operating Systems

CS 446/646

4. CPU Scheduling

a. Concepts of Scheduling

b. Scheduling Algorithms

- ✓ Scheduling in batch systems
- ✓ Scheduling in interactive systems

c. Queuing Analysis

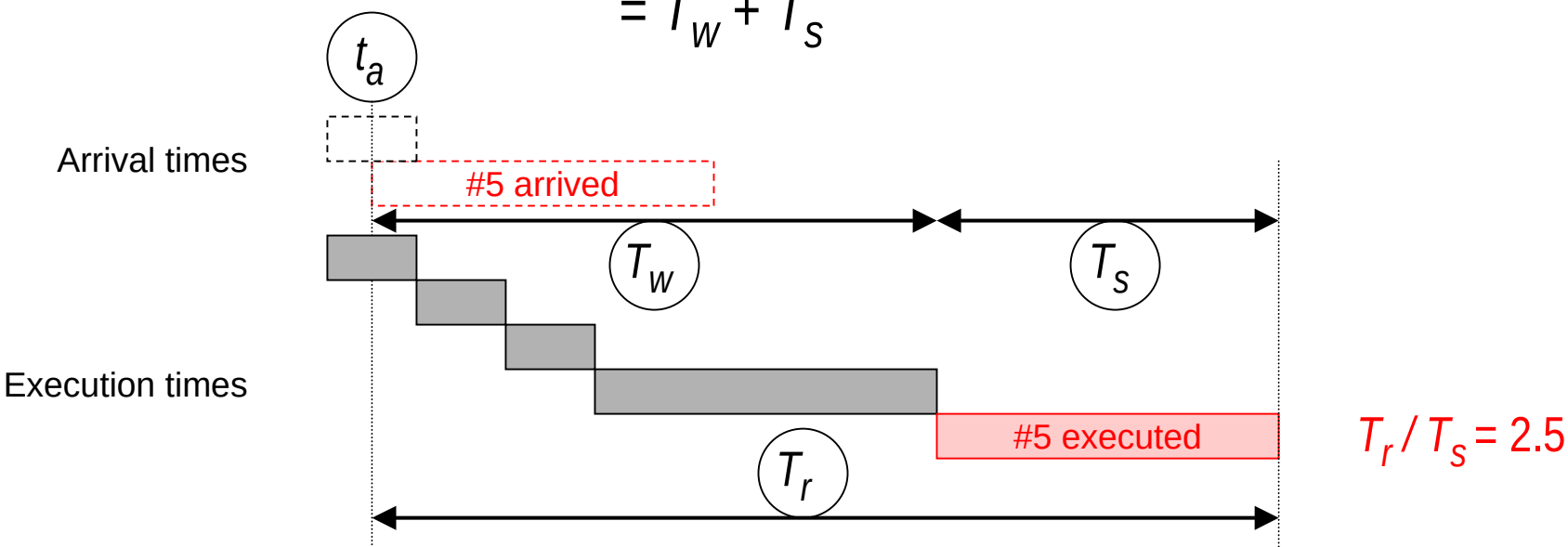
d. Thread Scheduling

4.b Scheduling Algorithms

Scheduling in batch systems

- Scheduling metrics
  - ✓ arrival time t_a = time the process became "Ready" (again)
  - ✓ wait time T_w = time spent waiting for the CPU
  - ✓ service time T_s = time spent executing in the CPU
  - ✓ turnaround time T_r = total time spent waiting and executing

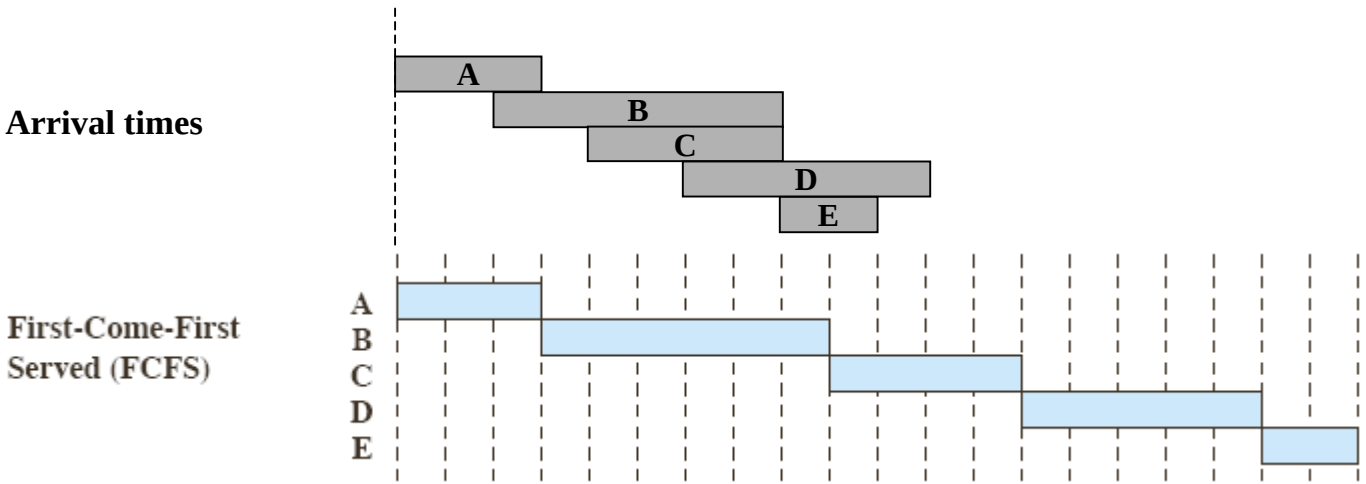
$$= T_w + T_s$$



4.b Scheduling Algorithms

Scheduling in batch systems

- First-Come-First-Served (FCFS)
  - ✓ processes are assigned the CPU in the order they request it
  - ✓ when the running process blocks, the first "Ready" is run next
  - ✓ when a process gets "Ready", it is put at the end of the queue



FCFS	Finish Time	A 3	B 9	C 13	D 18	E 20	Mean
	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

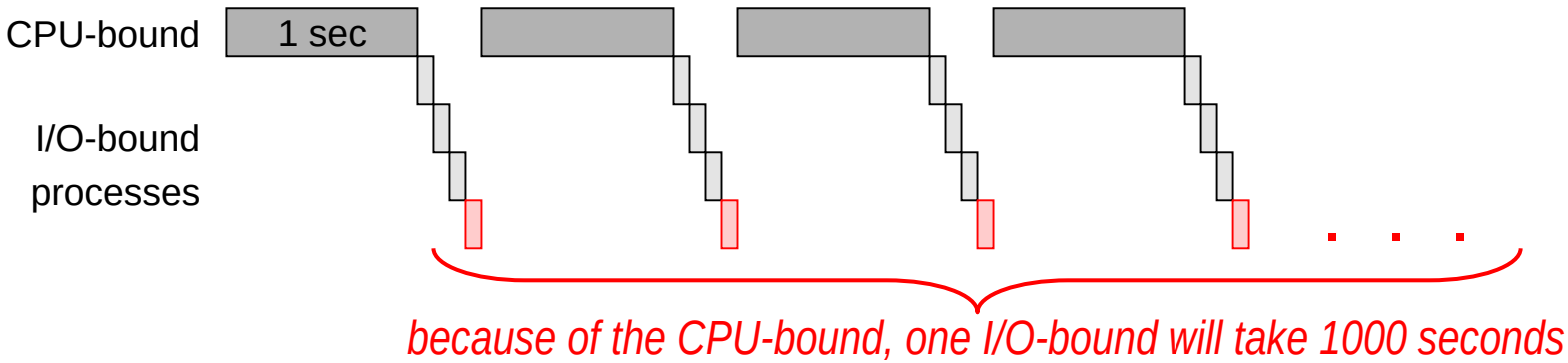
FCFS scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in batch systems

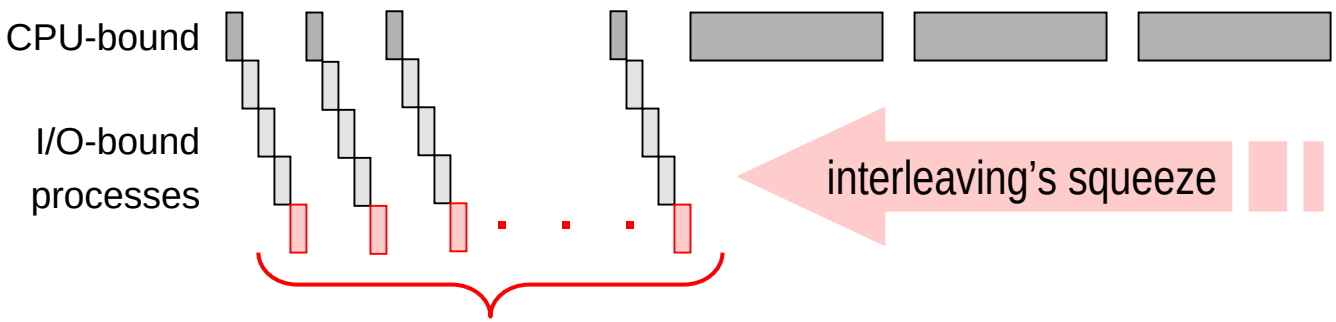
- First-Come-First-Served (FCFS)
  - ✓ nonpreemptive, oldest and simplest to program
  - ✓ apparently "fair" but very inefficient; example:
    - a CPU-bound process runs 1 sec, then reads 1 disk block
    - several I/O-bound processes run little CPU, but must read 1000 disk blocks



→ *preempt the CPU-bound more often to let the I/O-bound progress*

4.b Scheduling Algorithms

Scheduling in batch systems



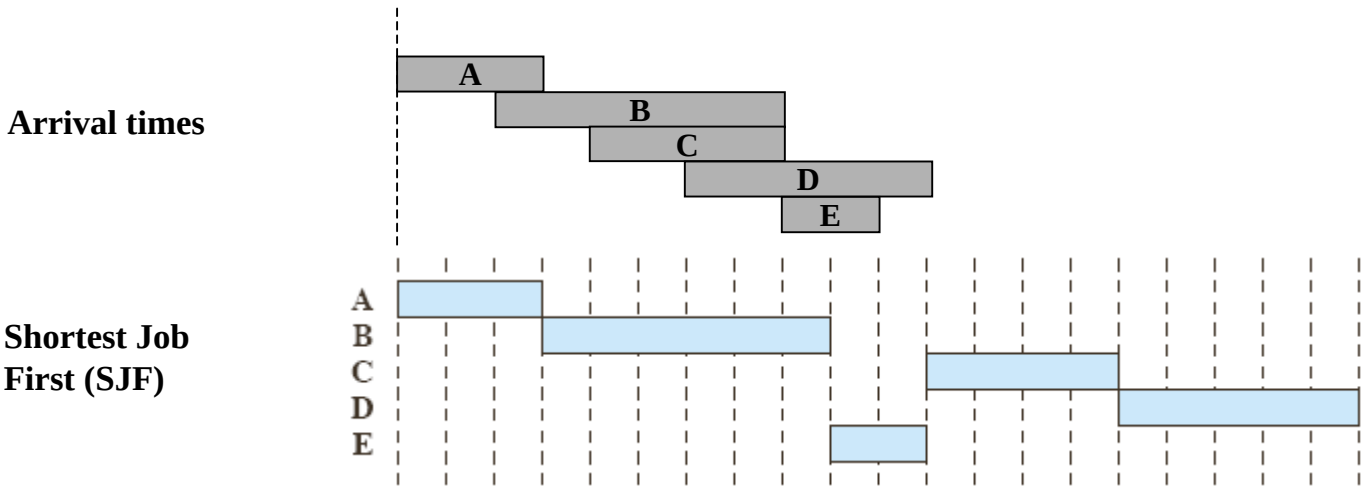
by preempting the CPU-bound every 10ms (100 Hz), each I/O-bound now takes only 10 seconds (without bothering the CPU-bound too much ~10s)

→ *see preemptive algorithms (Round-Robin, etc.) in later sections*

4.b Scheduling Algorithms

Scheduling in batch systems

- Shortest Job First (SJF)
  - ✓ nonpreemptive, assumes the run times are known in advance
  - ✓ among several equally important "Ready" jobs (or CPU bursts), the scheduler picks the one that will finish the earliest



SJF	Finish Time	**A**	3	**B**	9	**C**	15	**D**	20	**E**	11	**Mean**
	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

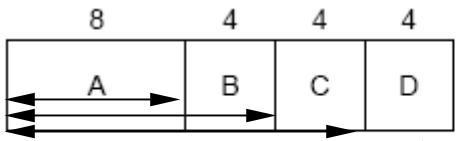
SJF scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

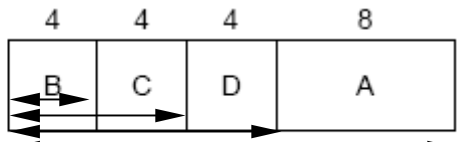
4.b Scheduling Algorithms

Scheduling in batch systems

- Shortest Job First (SJF)
  - ✓ example:



(a) no SJF



(b) SJF

a) turnaround times T_r = 8, 12, 16, 20 → mean T_r = 14

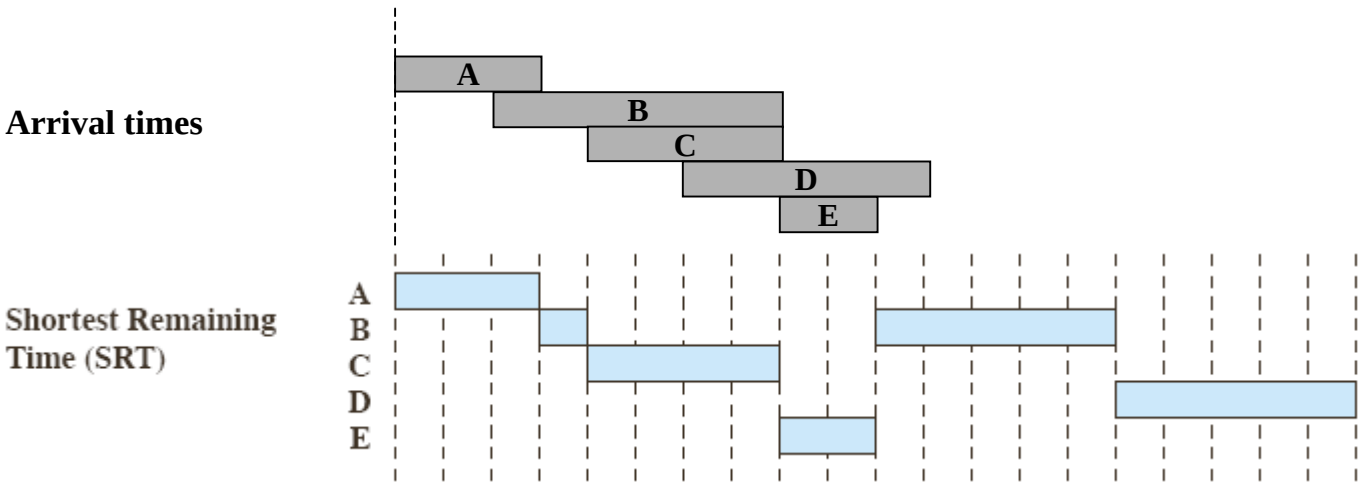
b) turnaround times T_r = 4, 8, 12, 20 → mean T_r = 11

  - ✓ SJF is optimal among jobs available immediately; proof:
    - generally, with service times T_s = a, b, c, d the mean turnaround time is: T_r = (4a + 3b + 2c + d) / 4, therefore it is always better to schedule the longest process (d) last
  - ✓ however, being non-preemptive, SJF does not deal well with jobs arriving subsequently (ex: 2,4,1,1,1 arriving at 0,0,3,3,3)

4.b Scheduling Algorithms

Scheduling in batch systems

- Shortest Remaining Time (SRT)
  - ✓ preemptive version of SJF, also assumes known run time
  - ✓ choose the process whose remaining run time is shortest
  - ✓ allows new short jobs to get good service



SRT	Finish Time	A	B	C	D	E	Mean
	Turnaround Time (T_r)	3	15	8	20	10	
		3	13	4	14	2	7.20
	T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59

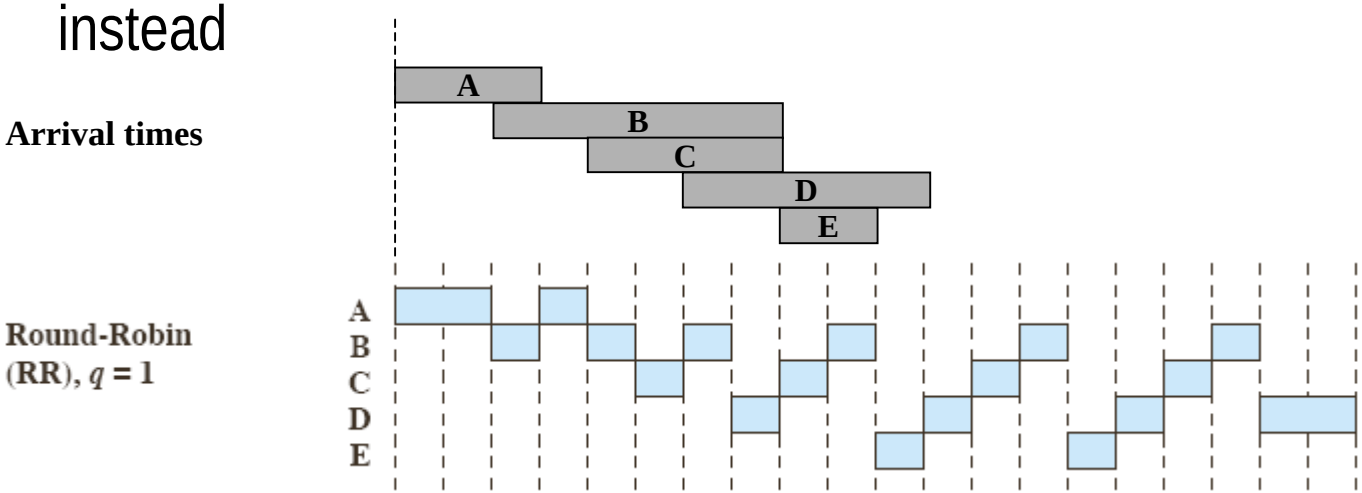
SRT scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- Round-Robin (RR)
  - ✓ preemptive FCFS, based on a timeout interval, the **quantum** *q*
  - ✓ the running process is interrupted by the clock and put last in a FIFO "Ready" queue; then, the first "Ready" process is run instead



RR $q = 1$	Finish Time	A 4	B 18	C 17	D 20	E 15	Mean
	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

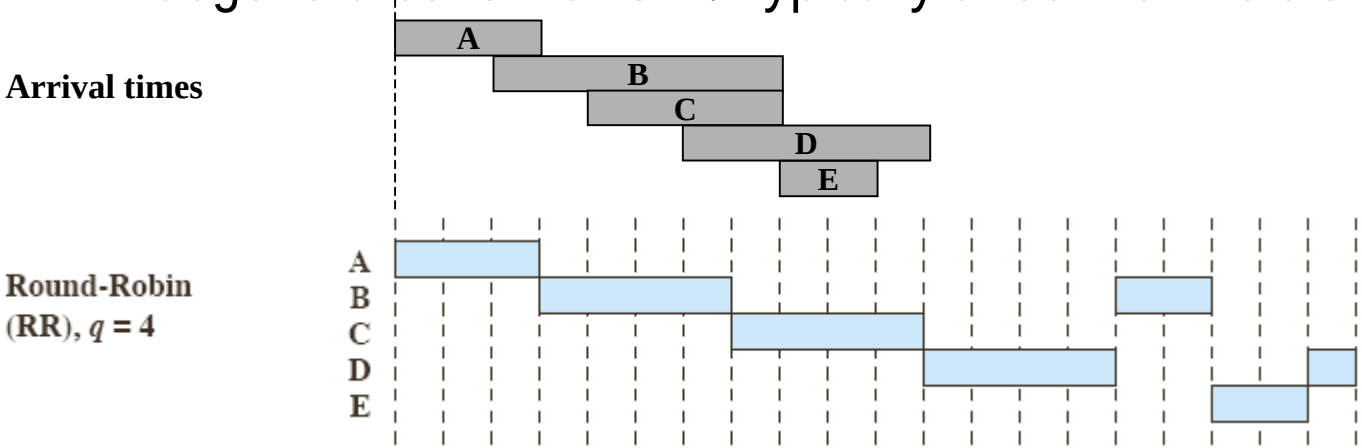
RR (q = 1) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- Round-Robin (RR)
  - ✓ a crucial parameter is the quantum q (generally ~10–100ms)
    - q should be big compared to context switch latency (~10μs)
    - q should be less than the longest CPU bursts, otherwise RR degenerates to FCFS → typically at 80% of the distrib. tail



RR $q = 4$	Finish Time	A 3	B 17	C 11	D 20	E 19	Mean
	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

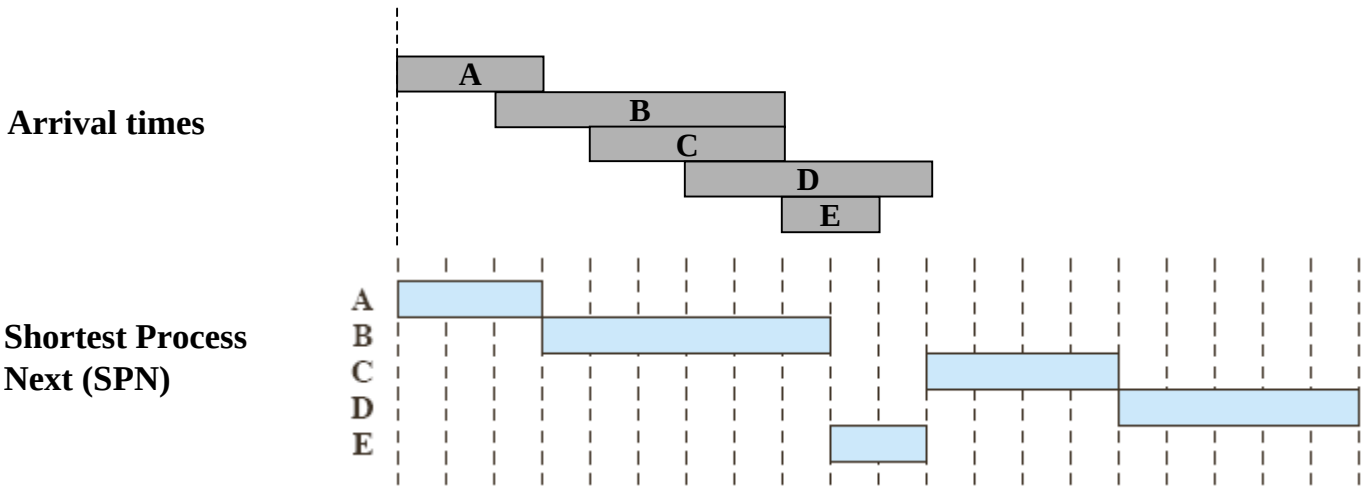
RR (q = 4) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- Shortest Process Next (SPN)
  - ✓ same as SJF: pick the one that should finish the earliest
  - → *difference in the interactive system: the prediction about future duration is not known but estimated from past durations*



SPN	Finish Time	A 3	B 9	C 15	D 20	E 11	Mean
	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

SPN scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

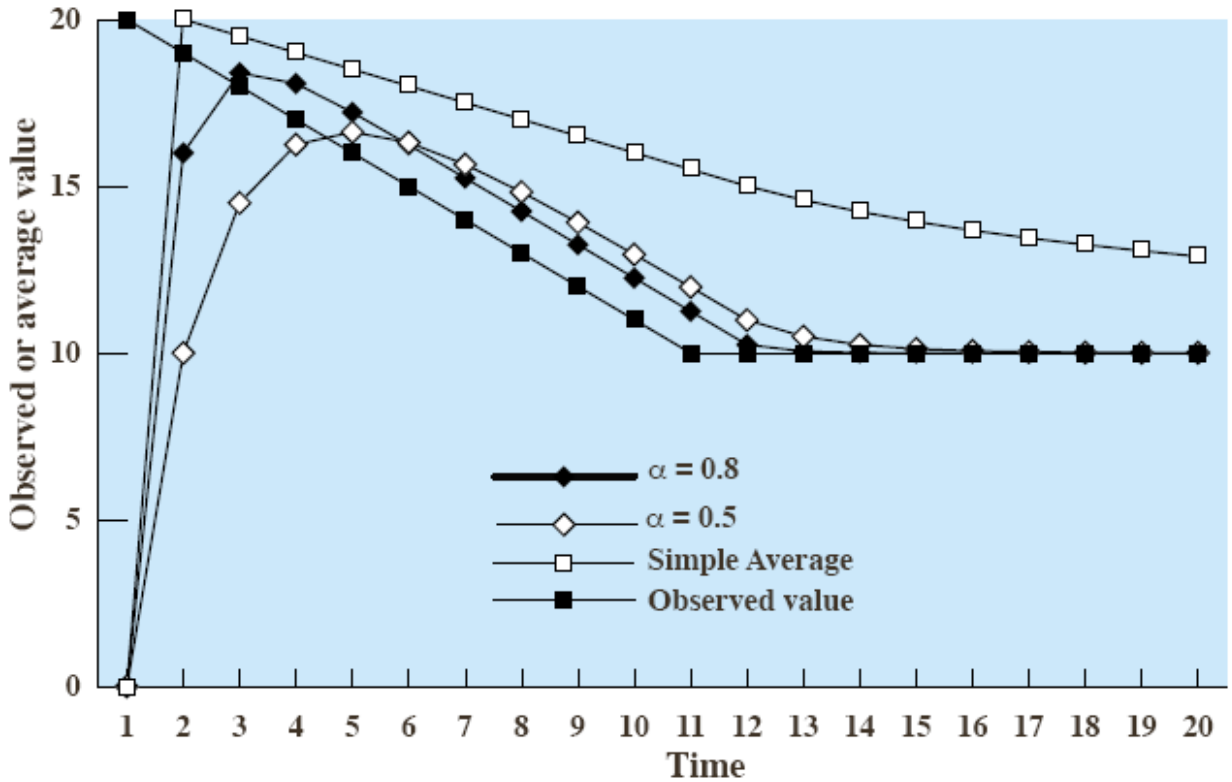
- Estimation of processing time from past
  - ✓ predicted service time = simple averaging of past run times
    - $S(n+1) = (1/n) \sum T(i)$

      $\Leftrightarrow S(n+1) = T(n)/n + (1 - 1/n)\, S(n)$
  - ✓ exponential averaging, also called “aging”
    - $S(n+1) = \alpha\, T(n) + (1 - \alpha)\, S(n), \quad 0 < \alpha \leq 1$
    - high α forgets past runs quickly
    - low α remembers past runs for a long time

4.b Scheduling Algorithms

Scheduling in interactive systems

- Estimation of processing time from past
  - ✓ "aging" tracks changes in process behavior faster than the mean



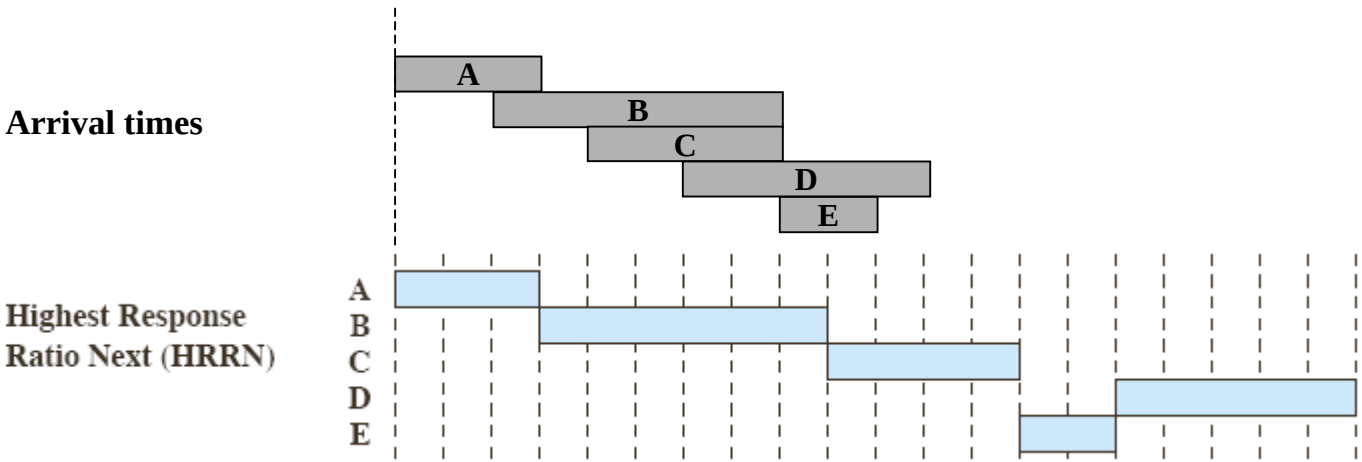
Example of exponential averaging in duration estimation

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- **Highest Response Ratio Next (HRRN)**
  - ✓ minimize the normalized turnaround time T_r / T_s
  - → *compromise between FCFS, which favors long processes, and SPN, which favors short processes*



HRRN	Finish Time	A 3	B 9	C 13	D 20	E 15	Mean
	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

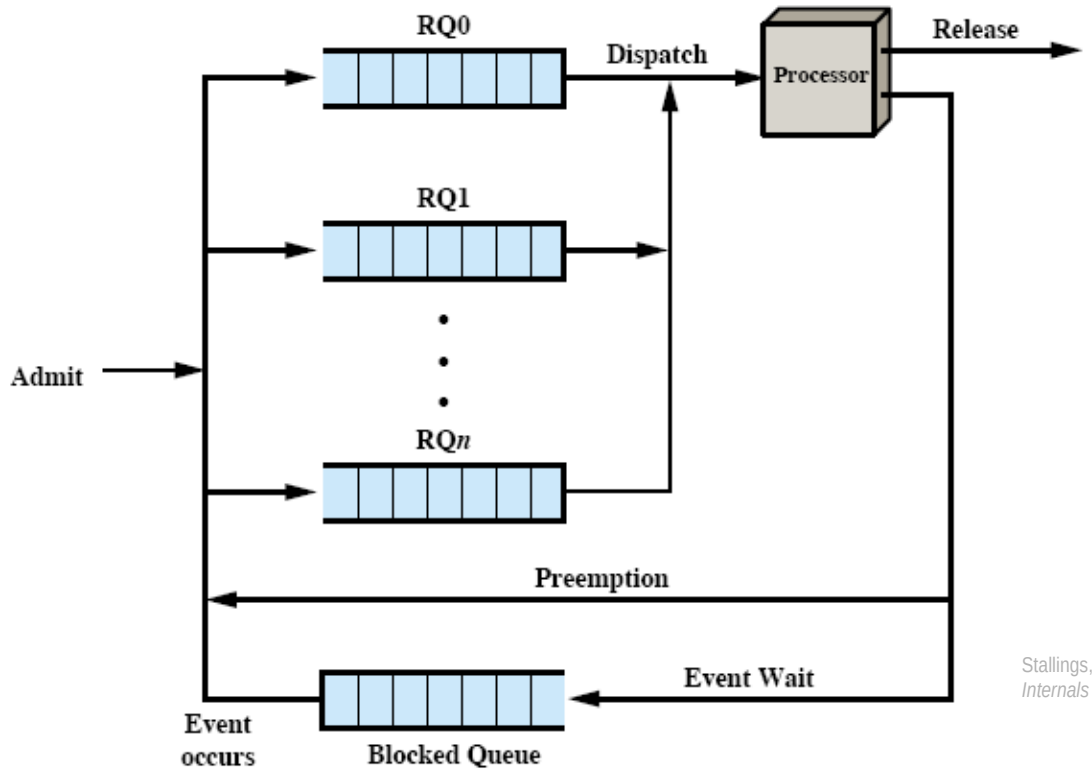
HRRN scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- Priority Scheduling
  - ✓ several “Ready” process queues, with different priorities



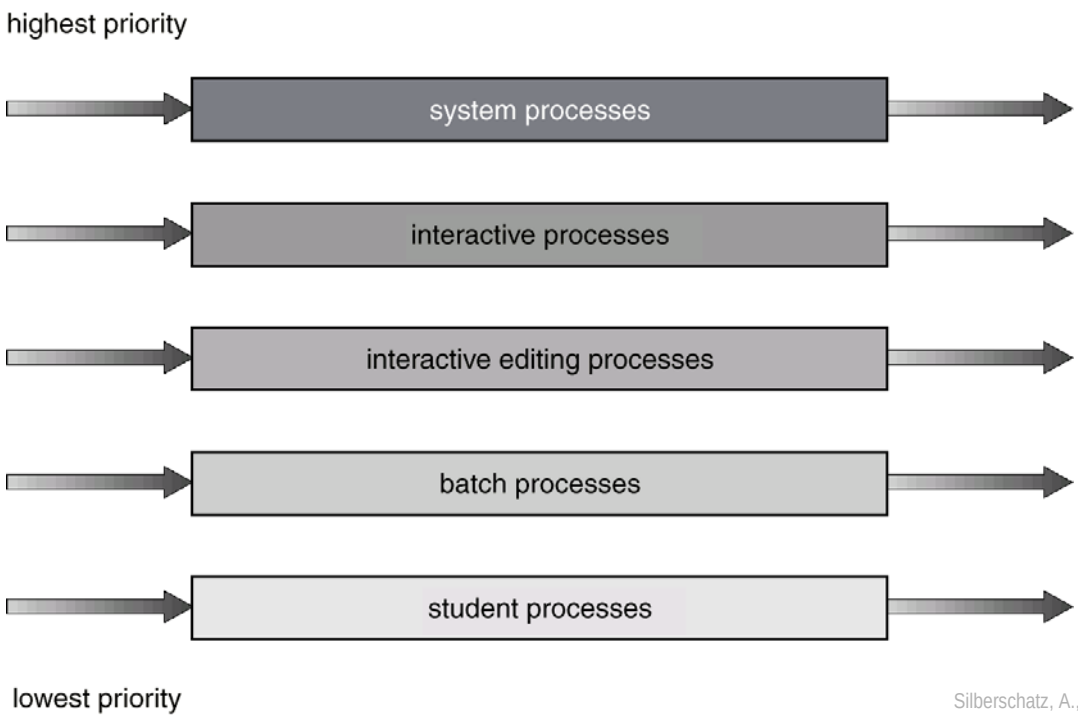
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Priority queuing

4.b Scheduling Algorithms

Scheduling in interactive systems

- Priority Scheduling
  - ✓ processes are assigned to queues based on their properties (memory size, priority, bound type, etc.)



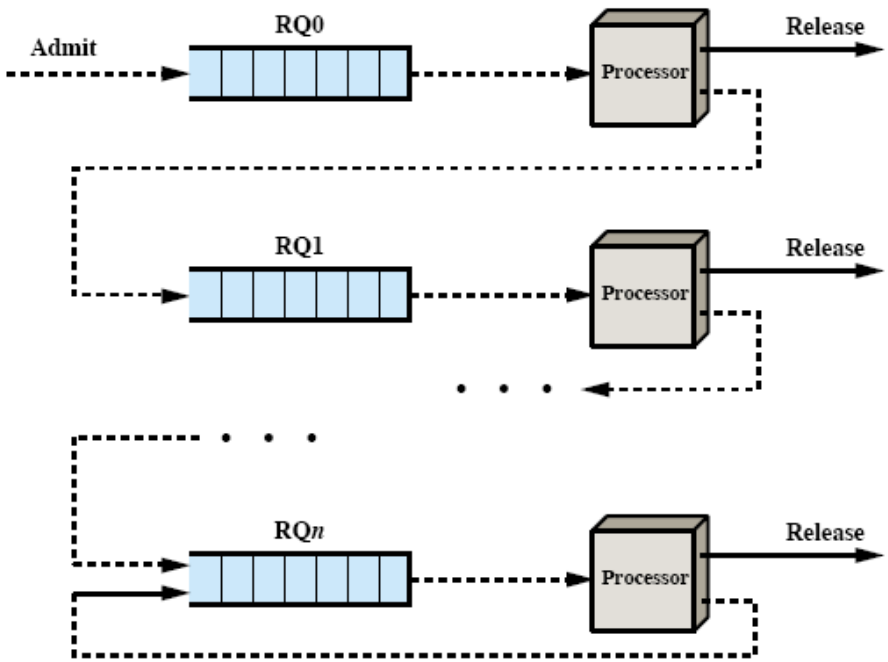
Multilevel queue scheduling

Silberschatz, A., Galvin, P. B. and Gagne. G. (2003) *Operating Systems Concepts with Java (6th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- Priority Scheduling with Feedback (FB)
  - ✓ processes can be moved among queues
  - ✓ each queue has its own policy, generally RR with variable $q(Q_i)$



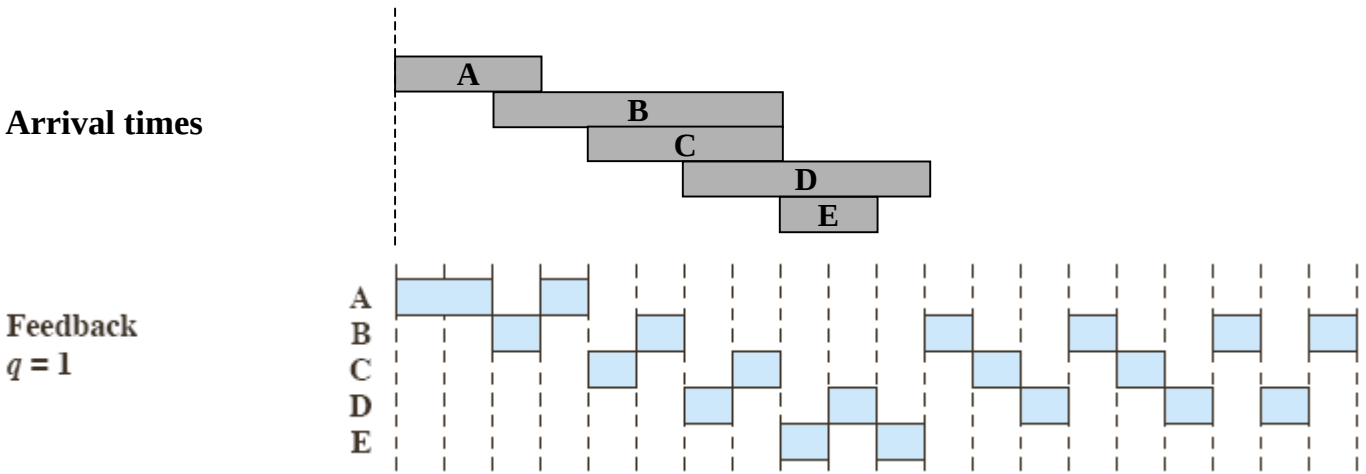
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Priority queuing

4.b Scheduling Algorithms

Scheduling in interactive systems

- Priority Scheduling with Feedback (FB)
  - ✓ each time a process is preempted, it is demoted to a lower-level queue
  - ✓ tends to leave I/O-bound in higher priority queues, as desired



FB $q = 1$	Finish Time	A 4	B 20	C 16	D 19	E 11	Mean
	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

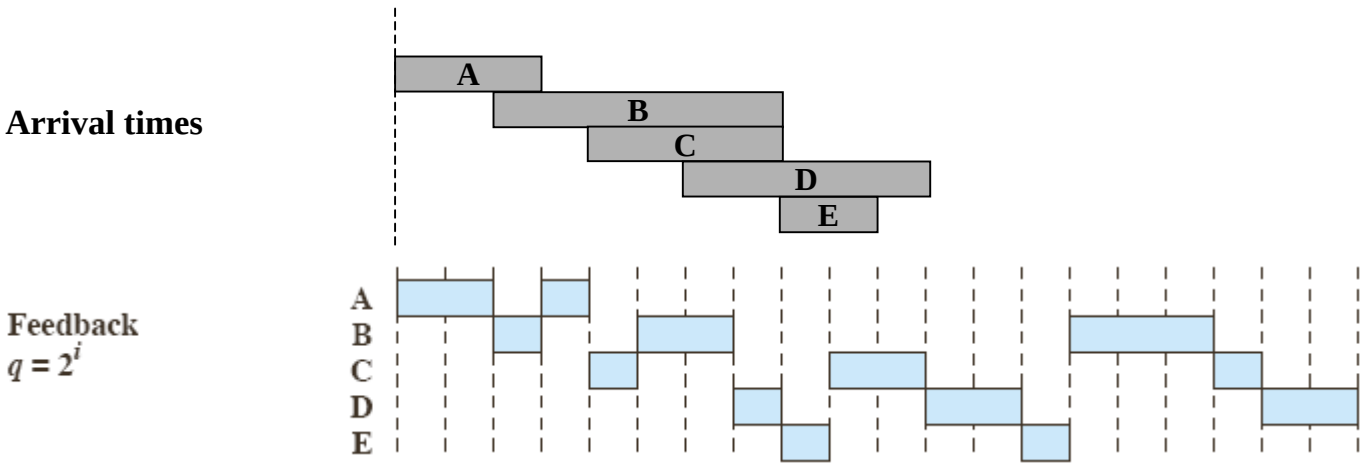
FB (q = 1) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

- Priority Scheduling with Feedback (FB)
  - ✓ a uniform RR quantum for all queues might create starvation
  - ✓ to compensate for increasing wait times in lower queue, increase q, too; for example $q = 2^i$



FB $q = 2^i$	Finish Time	A 4	B 17	C 18	D 20	E 14	Mean
	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

FB ($q = 2^i$) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

4.b Scheduling Algorithms

Scheduling in interactive systems

	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

4.b Scheduling Algorithms

Scheduling in interactive systems

- Traditional UNIX scheduling
  - ✓ multilevel feedback using RR within each of the priority queues
  - ✓ typically 1-second preemption timeout
  - ✓ system of integer priorities recomputed once per second
  - ✓ a base priority divides processes into fixed bands of priority levels; in decreasing order:
    - swapper
    - block I/O device control
    - file manipulation
    - character I/O device control
    - user processes

Principles of Operating Systems

CS 446/646

4. CPU Scheduling

a. Concepts of Scheduling

b. Scheduling Algorithms

- ✓ Scheduling in batch systems
- ✓ Scheduling in interactive systems

c. Queuing Analysis

d. Thread Scheduling