Scheduling (Part II)

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

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 TONIGHT @10pm!
  - →*Please upload as a zipped folder… upload issue was fixed ;)*

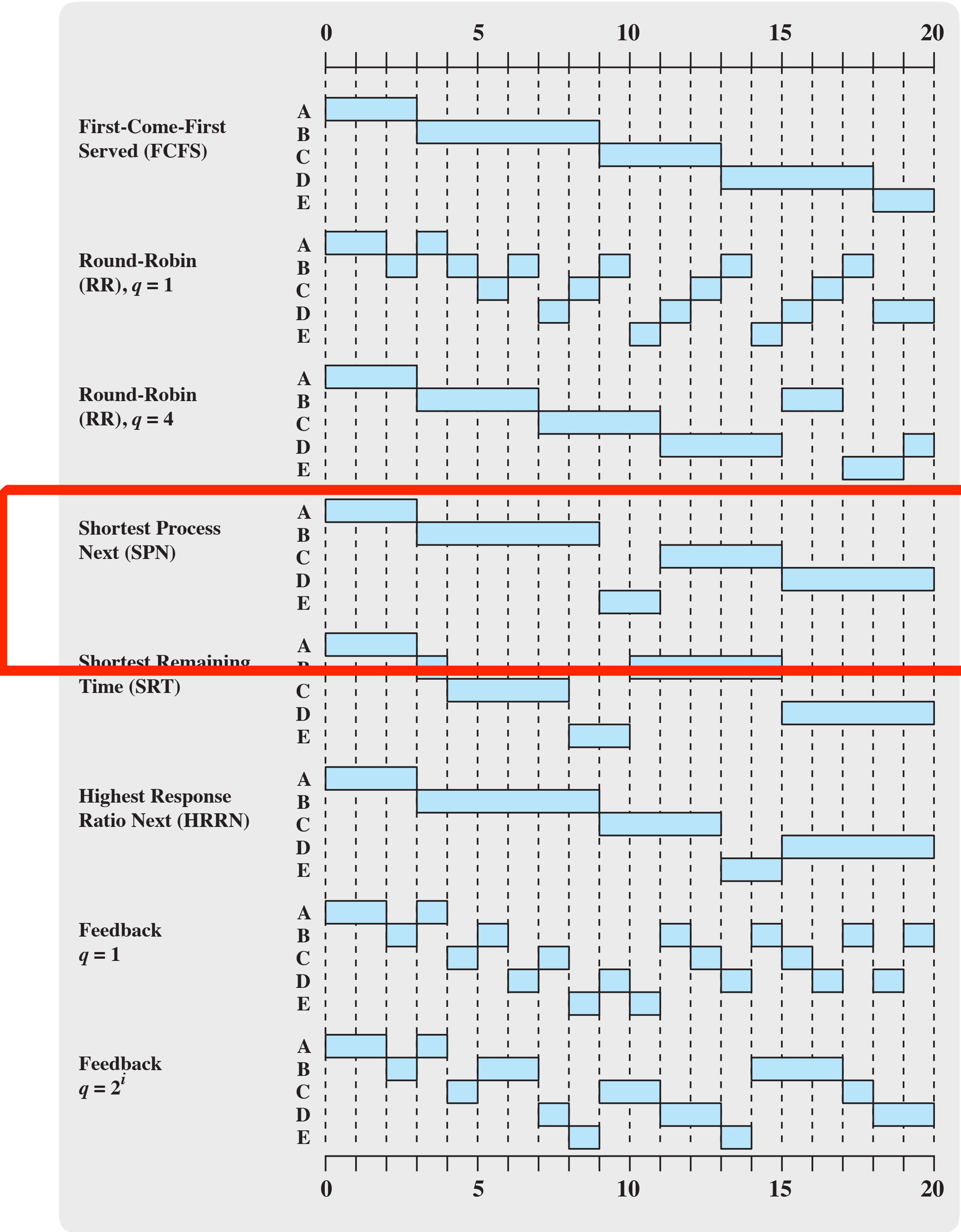*https://drawings.jvns.ca/scheduling/*
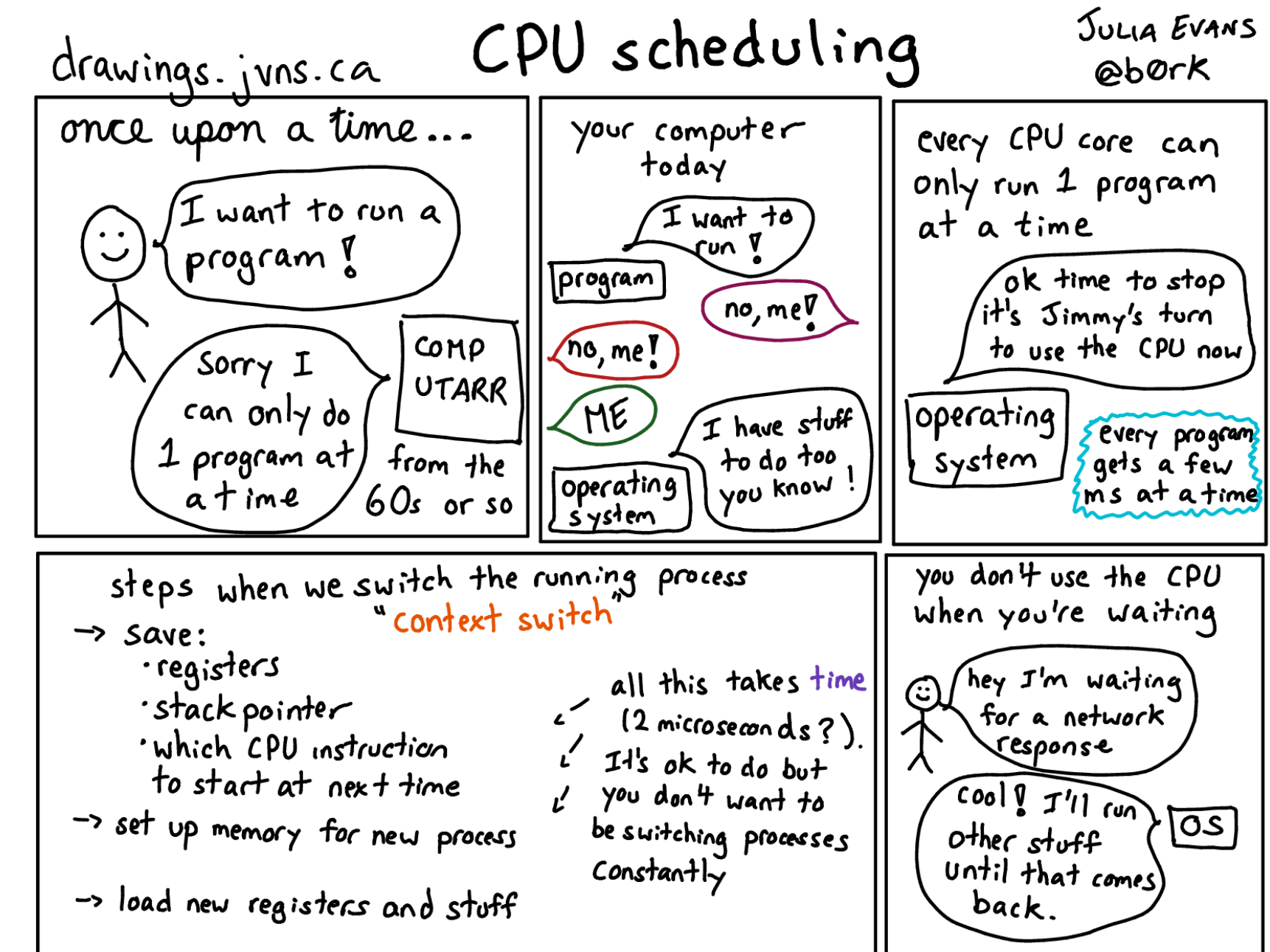
Shortest Process Next (SPN)

Select the process with the shortest expected processing time, and do not preempt the process

- No preemption allowed…
- Process w/ shortest (expected) processing time is selected next
  - i.e., "short" processes cut to the head of a queue
- Need to know (or estimate) the required processing time…
  - How?

- **Q:** Potential issues?

- Some pretty undesirable characteristics…
  - Starvation for longer processes is possible (assuming shorter processes keep arriving)
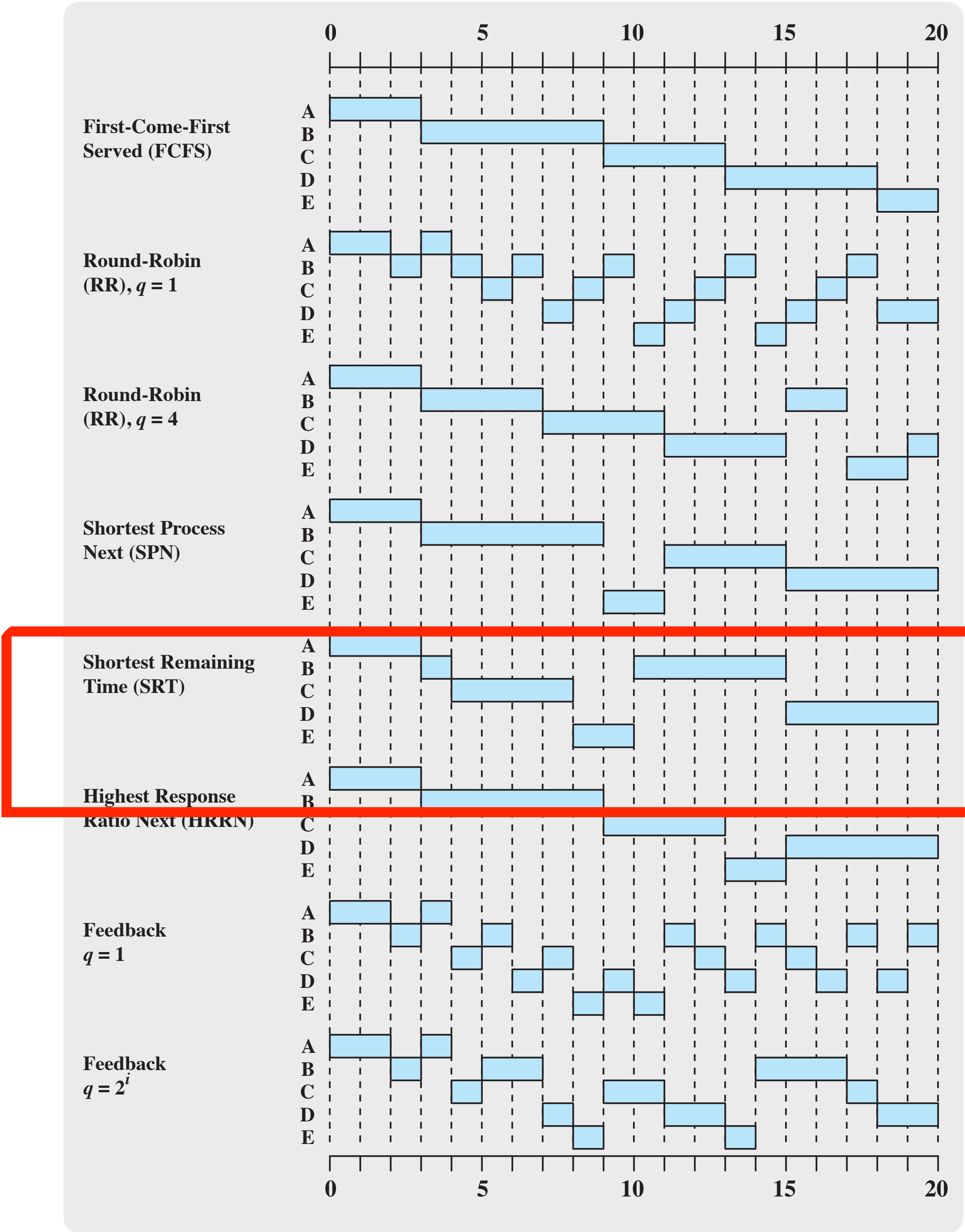 - Lacks preemption… not good for time-sharing systems

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

First-Come-First Served (FCFS)

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

Round-Robin (RR), $q = 4$

Shortest Process Next (SPN)

Shortest Remaining Time (SRT)

Highest Response Ratio Next (HRRN)

Feedback $q = 1$

Feedback $q = 2^i$

Shortest Remaining Time (SRT)

Select the process with the shortest expected remaining process time; preemption allowed

- Preemption allowed! *(a preemptive version of SPN)*
- Process w/ shortest (expected) **remaining** processing time is selected next
  - i.e., processes w/ less remaining time can preempt those w/ longer remaining times

- **Q:** Advantages? Disadvantages?

- ***Still…***
  - need to know (or estimate) the required processing time…
  - have a risk of starving longer-running processes…
  - incur overhead for recording service times, etc.
- ***BUT…***
  - SRT yields superior turnaround time as compared to SPN
    (i.e., a short task is given immediate preference to running a longer job)

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

Highest Response Ratio Next (HRRN)

Base scheduling decision on an estimate of normalized turnaround time (TAT)
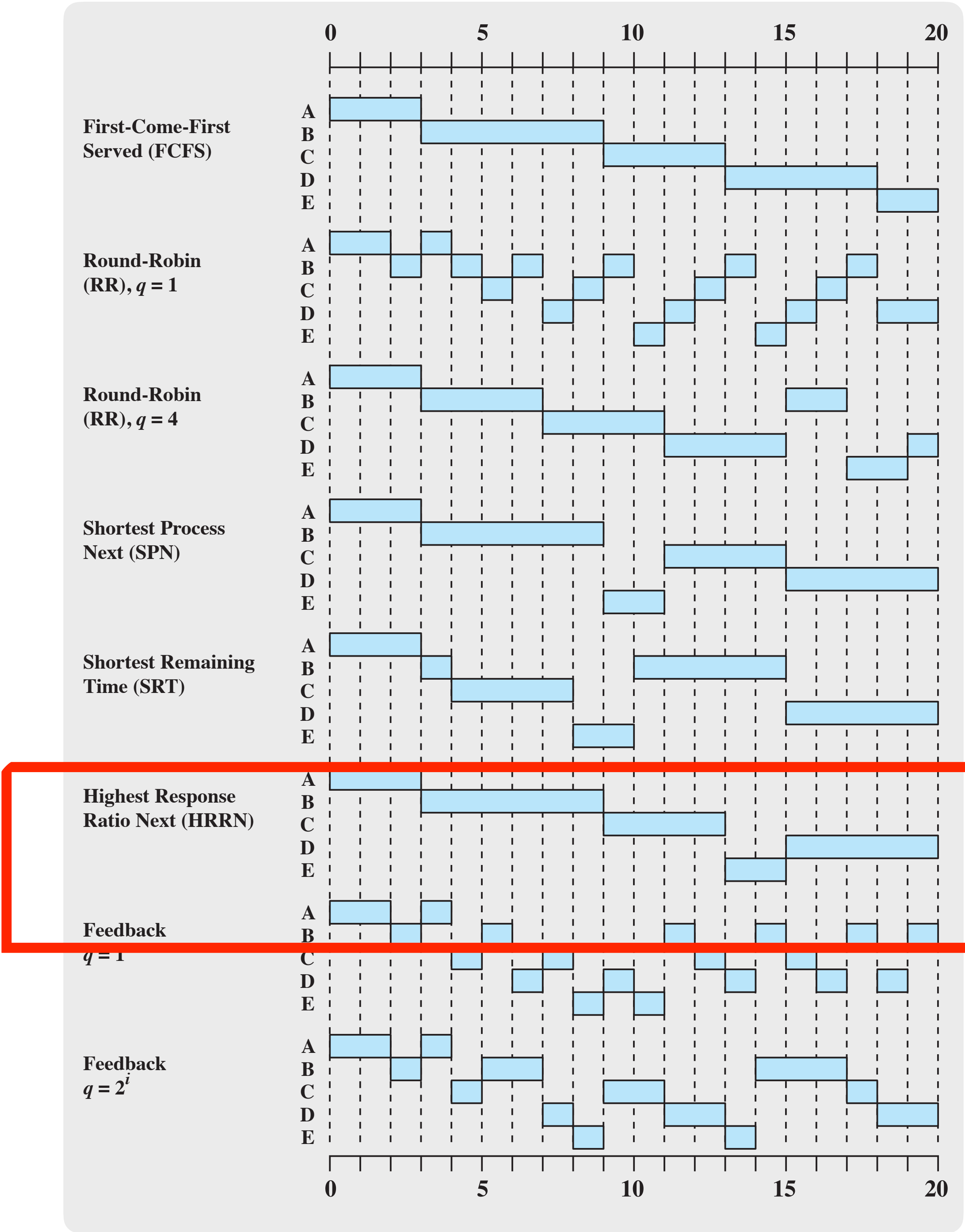
key
R = response ratio
w = time spent waiting
s = expected service time

- When the current process blocks

  → choose a ready process w/ the greatest **normalized turnaround time (R)**

$$R = \frac{w + s}{s}$$

- Accounts for the age of the process!
  - shorter jobs are favored…
  - but an aging process without service increases the ratio, so a longer process will eventually get past competing shorter jobs.

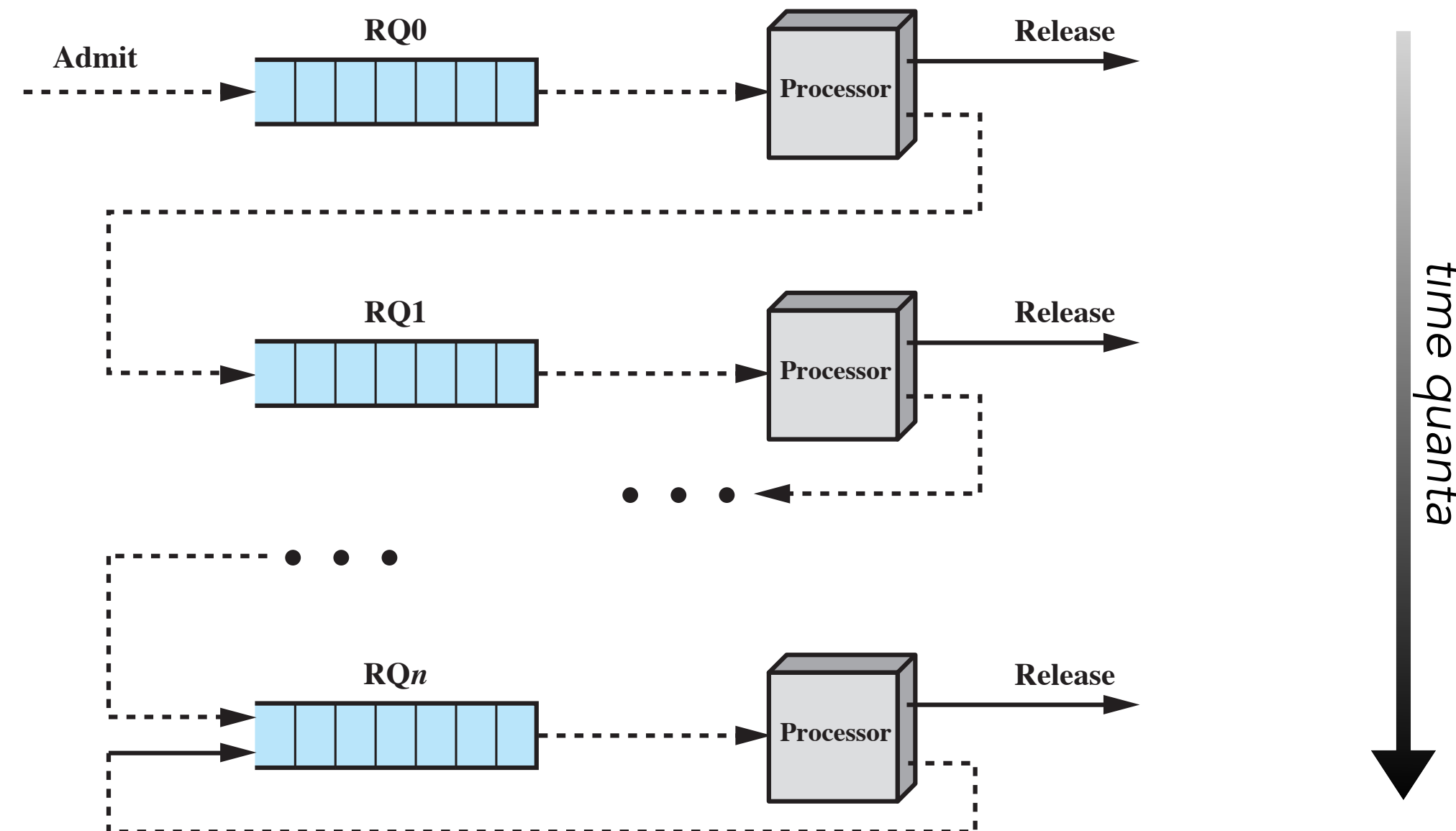| Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25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

(Multilevel) Feedback

Establish a set of scheduling queues and allocate processes to queues based on execution history and other criteria

- SPN, SRT, HRRN are tricky — need reasonable estimates for execution time
- What if we instead "penalize" jobs that have been running longer?

If we can't get reliable measures of
time remaining to execute
let us focus on
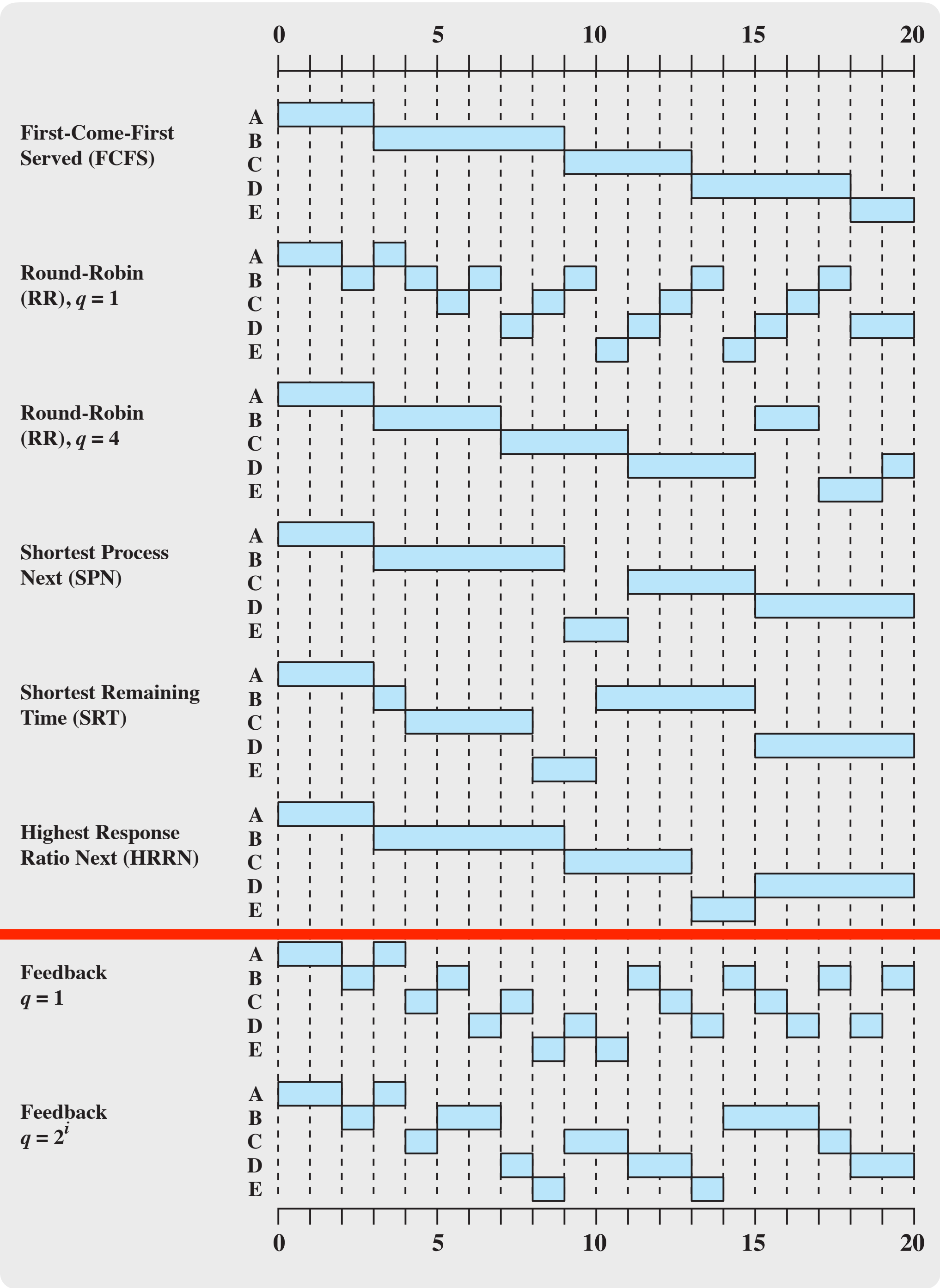***time spent executing so far!***

(Multilevel) Feedback

Establish a set of scheduling queues and allocate processes to queues based on execution history and other criteria

The Idea:

→ *Use some time quanta (re: [V]RR) to periodically interrupt a running process*

→ *"Demote" process to lower priority Q each time it returns to the RQ*

→ *Short processes still finish relatively quickly (few demotions)*

→ *Use FCFS in all Qs (except lowest-priority Q, which acts as RR; cannot demote any further)*

Any issues?

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

So… which approach is best?

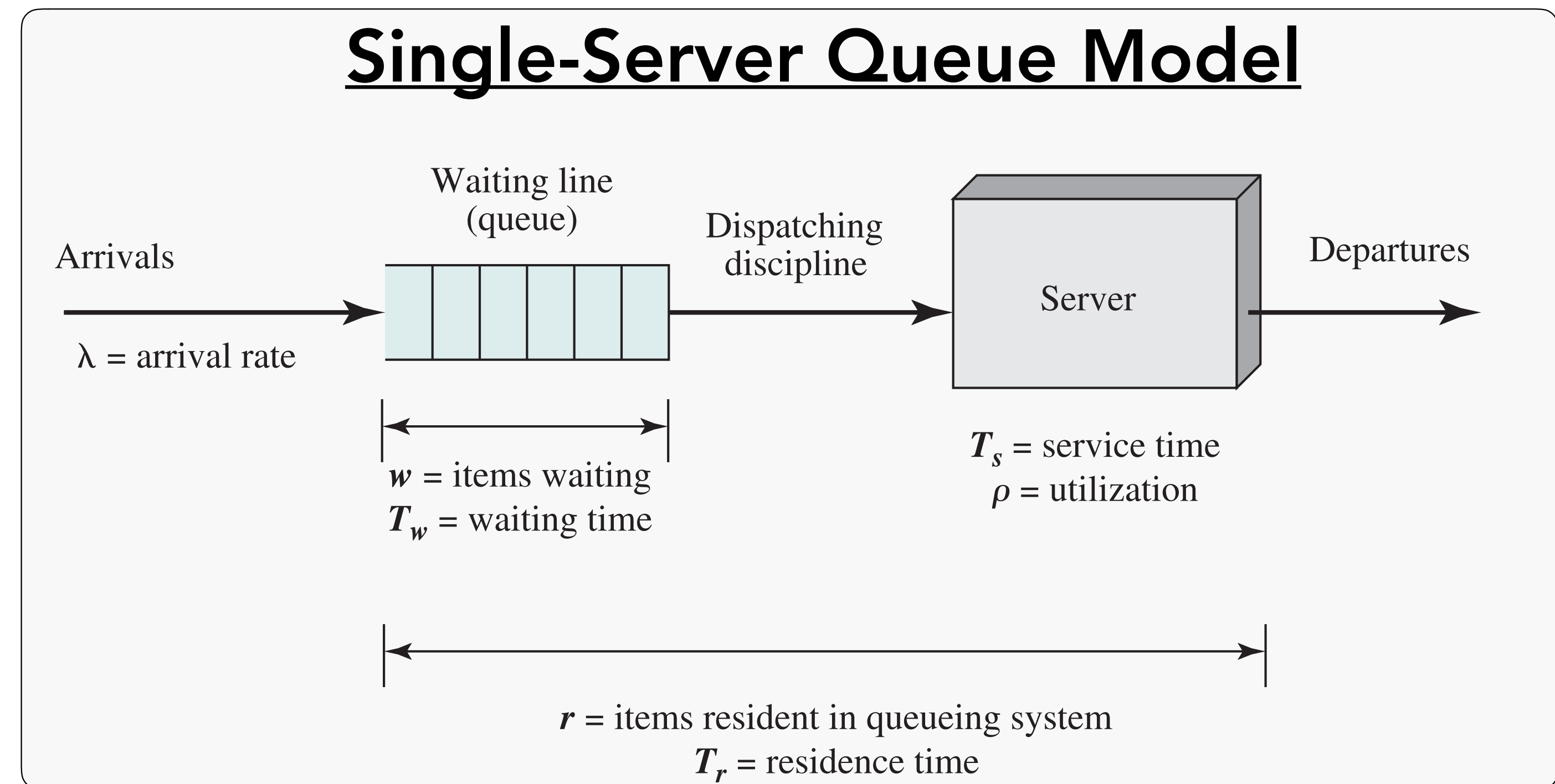# Performance Comparisons & General Take-Aways

- Difficult to develop closed analytic models for comparisons*

  → *All approaches (except RR and FCFS) depend on **expected service times**, which really complicates things*

- Relative performance depends on a variety of factors

  - **probability distribution** of service times of processes
  - **efficiency** of scheduling mechanisms
  - **efficiency** of context switching mechanisms
  - **demand** on, and efficiency of, I/O subsystem(s)
  - …

Performance Comparisons & General Take-Aways (Queuing Analysis)

- Difficult to develop closed analytic models for comparisons*

  → *All approaches (except RR and FCFS) depend on **expected service times**, which really complicates things*

- Relative performance depends on a variety of factors

  - probability distribution of service times of processes
  - efficiency of scheduling mechanisms
  - efficiency of context switching mechanisms
  - demand on, and efficiency of, I/O subsystem(s)

- Enter ***Queueing Analysis***

  - Assume: Poisson arrivals

    → *random arrival times*

  - Assume: exponential service times

    → *more demand == longer service times*

  - Assume: priority-based scheduling…

    → *higher priority items served first*

    → *FCFS for items of equal priority*

Single-Server Queue Model

Arrivals

λ = arrival rate

Waiting line (queue)

Dispatching discipline

Departures

Server

T_s = service time
ρ = utilization

w = items waiting
T_w = waiting time

r = items resident in queueing system
T_r = residence time

*Extensions also exist for **multiple servers** and **multiple queues***

Performance Comparisons & General Take-Aways (Simulations)

- Difficult to develop closed analytic models for comparisons*

  → *All approaches (except RR and FCFS) depend on **expected service times**, which really complicate*

- Relative performance depends on a variety of factors

  - probability distribution of service times of processes
  - efficiency of scheduling mechanisms
  - efficiency of context switching mechanisms
  - demand on, and efficiency of, I/O subsystem(s)

- # Enter *Simulation Modeling*

  - **Pros:** discrete event simulations allow a variety of models to be simulated & compared
    - Able to visualize direct comparisons between policies
    - Compare **wait times**, **normalized turnaround times**, …
  - **Cons:** results only really hold for a given "run"
    - a particular collection of processes…
    - a particular set of assumptions & constraints…
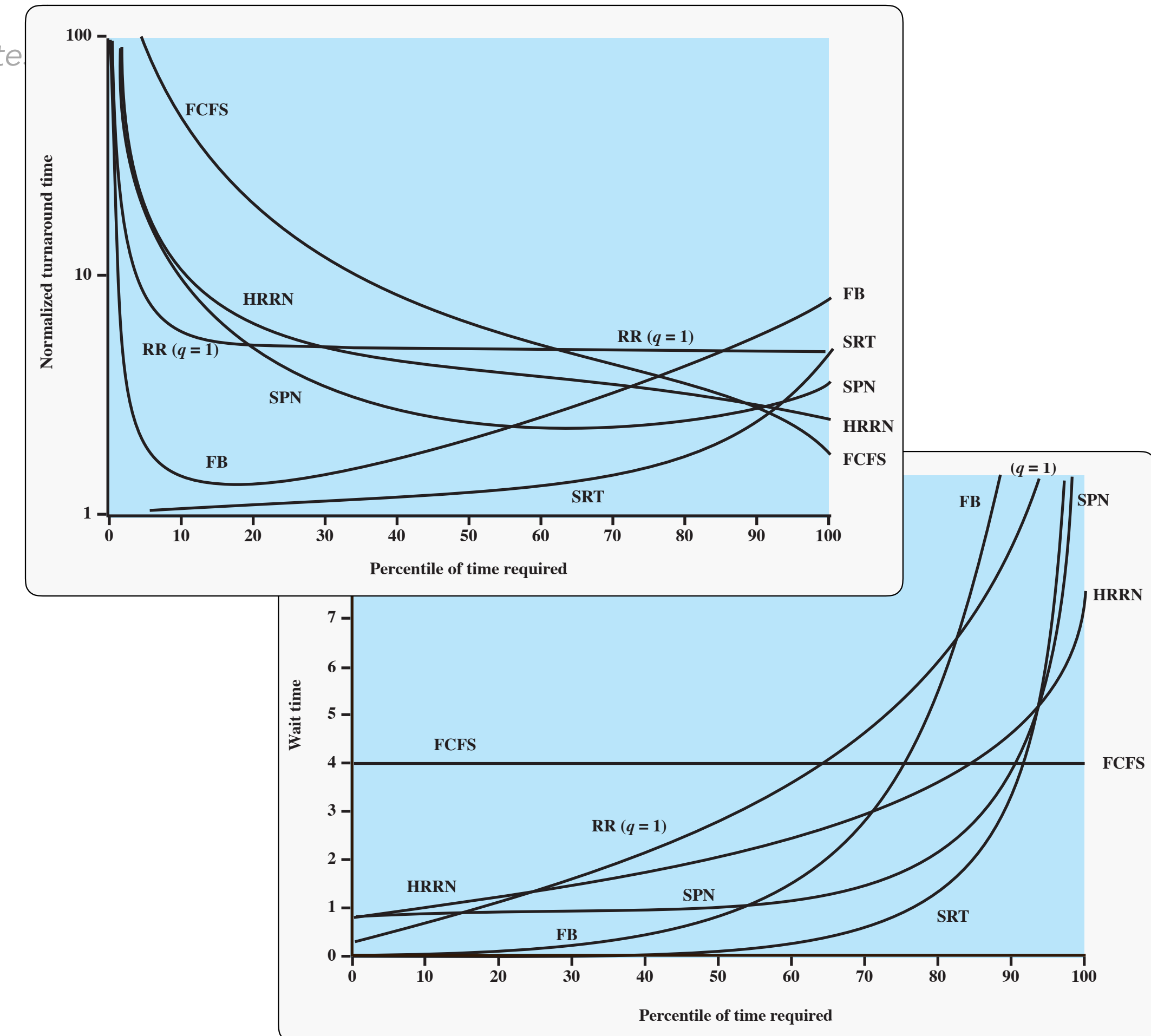
Fair-Share Scheduling

Establish a set of scheduling queues and allocate processes to queues based on execution history and other criteria

- Approaches so far all treat processes as a fairly homogeneous collection of ready-to-execute items
  - Many independent processes
  - Can be sub-divided by priority
  - Otherwise, treated the same
- **Q:** Can we see a potential issue with this?
  - What if there are 2 users, Alice and Bob, where each run 1 "app"
  - … and Alice's app is made up of 1 process
  - … and Bob's app is made up of 100 processes
- Fair-Share Scheduling
  - assign each user/group a "share" of the processor
  - monitor resource usage…
    - → give **fewer resources** to user/group that has had **more than their fair share**
    - → give **more resources** to user/group that has had **less than their fair share**

Note:
"fair-share"
doesn't necessarily mean
"equal share"

Fair-Share Scheduling (*example*)

Time	**Process A**			**Process B**			**Process C**		
	Priority	Process CPU count	Group CPU count	Priority	Process CPU count	Group CPU count	Priority	Process CPU count	Group CPU count
0	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0	60	0	0
1	90	30	30	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0 1 2 • • 60
2	74	15 16 17 • • 75	15 16 17 • • 75	90	30	30	75	0	30
3	96	37	37	74	15 16 17 • • 75	15 16 17 • • 75	67	0 1 2 • • 60	15 16 17 • • 75
4	78	18 19 20 • • 78	18 19 20 • • 78	81	7	37	93	30	37
5	98	39	39	70	3	18	76	15	18

Group 1 ⎵ (Process A, Process B) Group 2 ⎵ (Process C)

Colored rectangle represents executing process