Shortest Remaining Time (SRT)

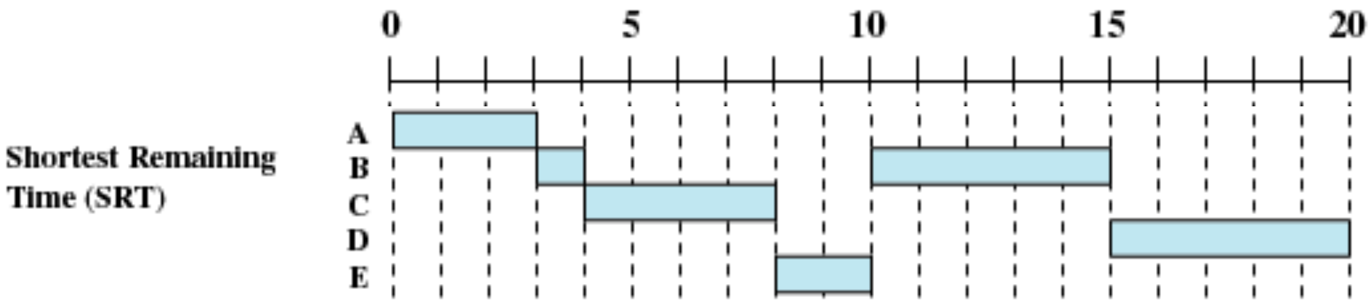


Table 9.4 Process Scheduling Example

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

Response Time and Ratio

- Response Ratio R is
  - total time spent waiting and executing normalized to the execution time
  - w: waiting time (waiting for a processor)
  - s: expected service (execution) time

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

  - Note: In scheduling theory response time is called **flow time** $F_i = C_i - r_i$
    - i.e., completion time minus ready time
    - this is the sum of waiting and processing times

Highest Response Ratio Next (HRRN)

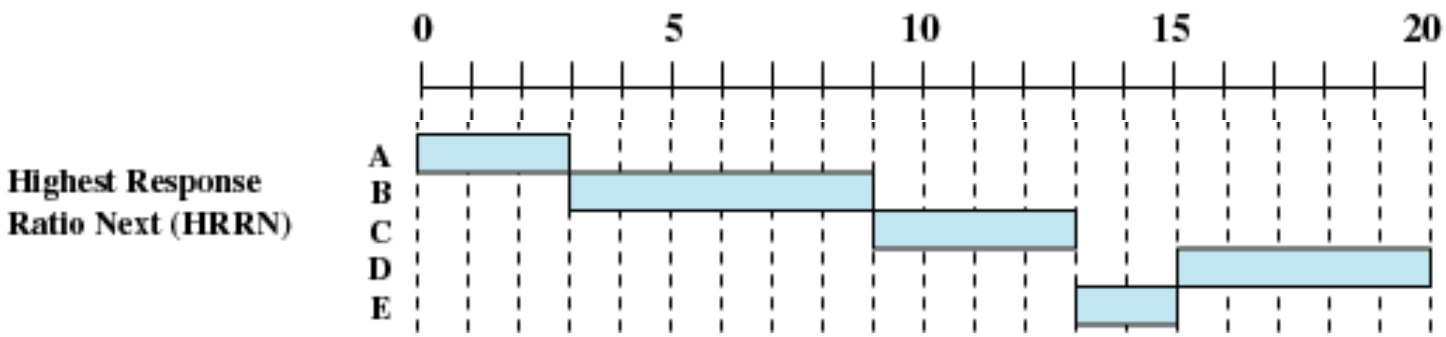


Table 9.4 Process Scheduling Example

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

- Choose next process with the greatest response ratio

Feedback

- SPN, SRT and HRRN require that something is known about the execution times
  - e.g., expected execution time
- Alternative policies
  - give preference to shorter tasks by penalizing tasks that have been running longer

Use multiple queues, pushing tasks to the next queue after each preemption

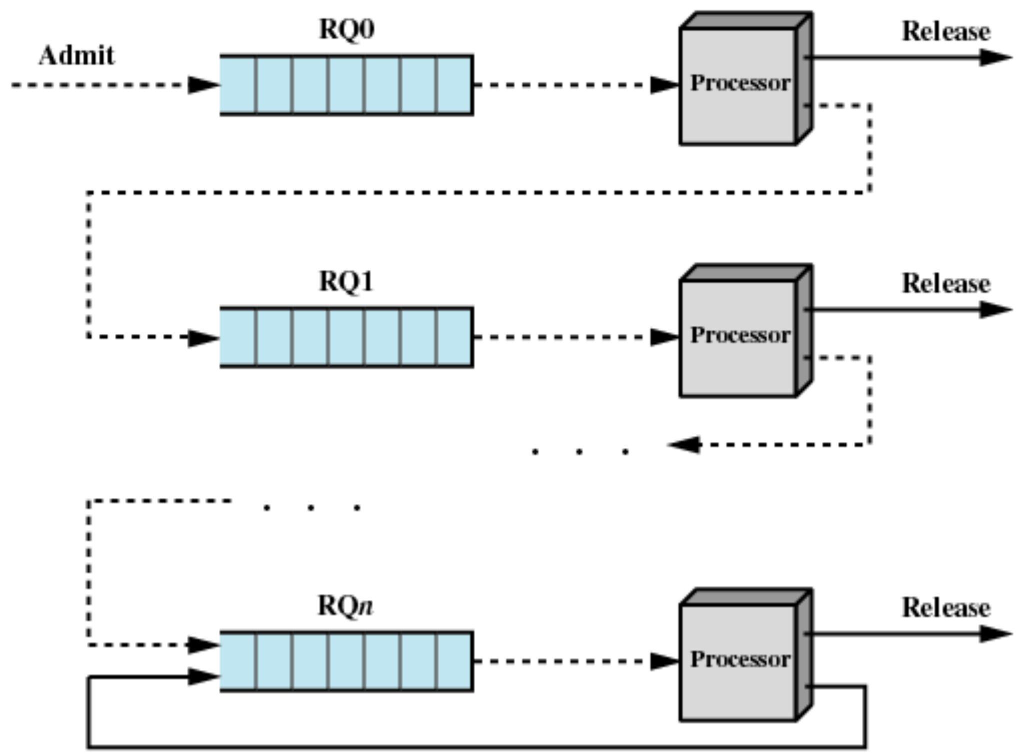
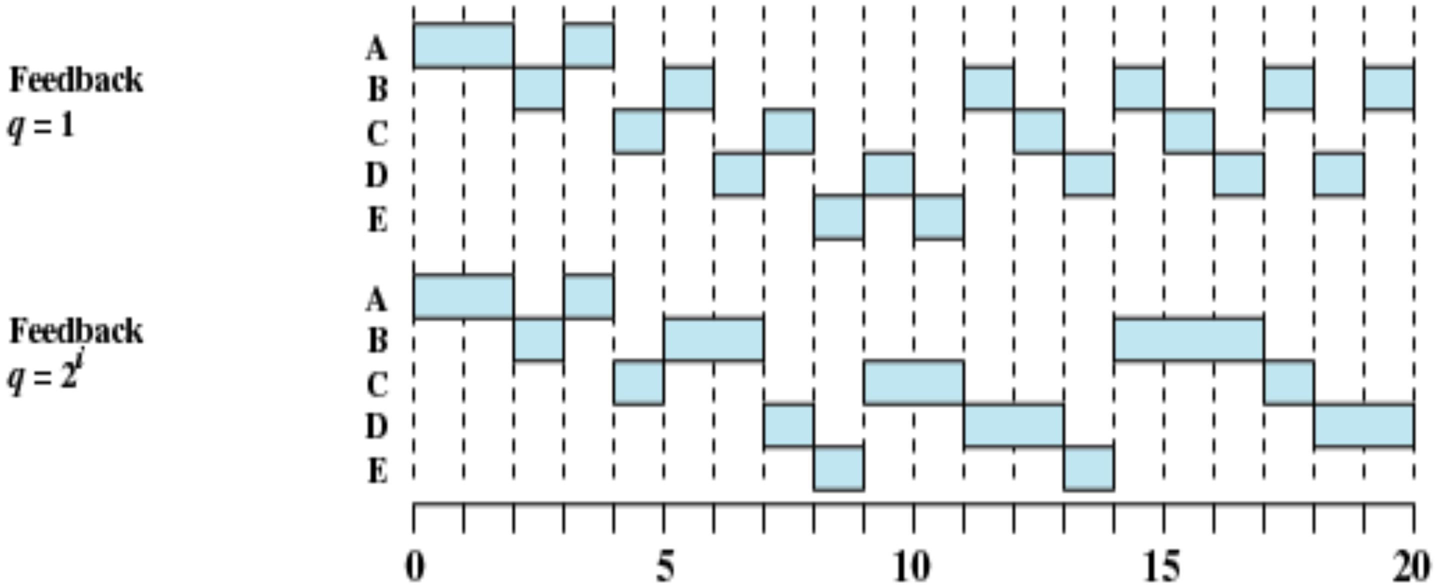


Figure 9.10 Feedback Scheduling

Feedback

- Potential problems
  - starvation
  - low response times for longer tasks
  - many solutions exists, e.g.,
    - use fixed quantum
      - $q = 1$
    - use different quantum in consequent queues
      - $q = 2^i$ for queue i
      - starvation still possible though
        - » solution: "promote" jobs to higher queue after some time

Feedback



- Don’t know remaining time process needs to execute

Table 9.4 Process Scheduling Example

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

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	$\max\left(\frac{w+s}{s}\right)$	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

Table 9.5 A Comparison of Scheduling Policies

	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

Table 9.6 Formulas for Single-Server Queues with Two Priority Categories

Assumptions:
1. Poisson arrival rate.
2. Priority 1 items are serviced before priority 2 items.
3. First-in-first-out dispatching for items of equal priority.
4. No item is interrupted while being served.
5. No items leave the queue (lost calls delayed).

(a) General Formulas

$\lambda = \lambda_1 + \lambda_2$	arrival rate
$\rho_1 = \lambda_1 T_{s1};\ \rho_2 = \lambda_2 T_{s2}$ $\rho = \rho_1 + \rho_2$	utilization
$T_s = \frac{\lambda_1}{\lambda} T_{s1} + \frac{\lambda_2}{\lambda} T_{s2}$	average service time
$T_r = \frac{\lambda_1}{\lambda} T_{r1} + \frac{\lambda_2}{\lambda} T_{r2}$	turnaround time

b) No interrupts; exponential service times

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1} + \rho_2 T_{s2}}{1 - \rho_1}$$

$$T_{r2} = T_{s2} + \frac{T_{r1} - T_{s1}}{1 - \rho}$$

(c) Preemptive-resume queuing discipline; exponential service times

$$T_{r1} = T_{s1} + \frac{\rho_1 T_{s1}}{1 - \rho_1}$$

$$T_{r2} = T_{s2} + \frac{1}{1 - \rho_1}\left(\rho_1 T_{s2} + \frac{\rho T_s}{1 - \rho}\right)$$

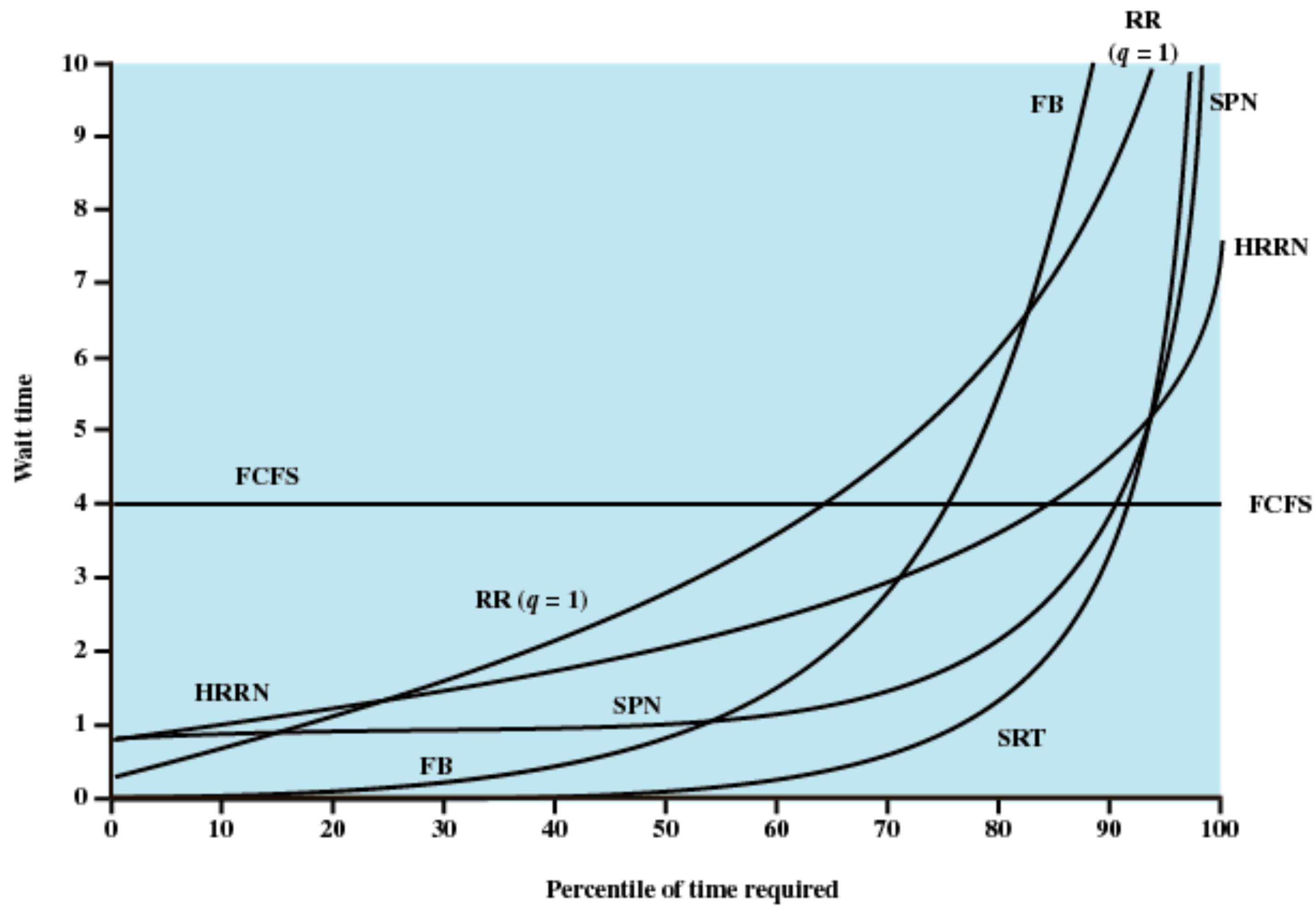


Figure 9.15 Simulation Results for Waiting Time

Fair-Share Scheduling

- All previous approaches treat collection of ready processes as single pool
- User's application runs as a collection of processes (threads)
  - concern about the performance of the application, not single process; (this changes the game)
  - need to make scheduling decisions based on process sets

Fair-Share Scheduling

- Philosophy can be extended to groups
  - e.g. time-sharing system,
    - all users from one department treated as group
    - the performance of that group should not affect other groups significantly
      - e.g. as many people from the group log in performance degradation should be primarily felt in that group

Fair-Share Scheduling

- Fair share
  - each user is assigned a weight that corresponds to the fraction of total use of the resources
  - scheme should operate approximately linear
    - e.g. if user A has twice the weight of user B, then (in the long run), user A should do twice the work than B.

Traditional UNIX Scheduling

- Multilevel feedback using round robin within each of the priority queues
- If a running process does not block or complete within 1 second, it is preempted
- Priorities are recomputed once per second
- Base priority divides all processes into fixed *bands* of priority levels

Bands

- Decreasing order of priority
  - Swapper
  - Block I/O device control
  - File manipulation
  - Character I/O device control
  - User processes