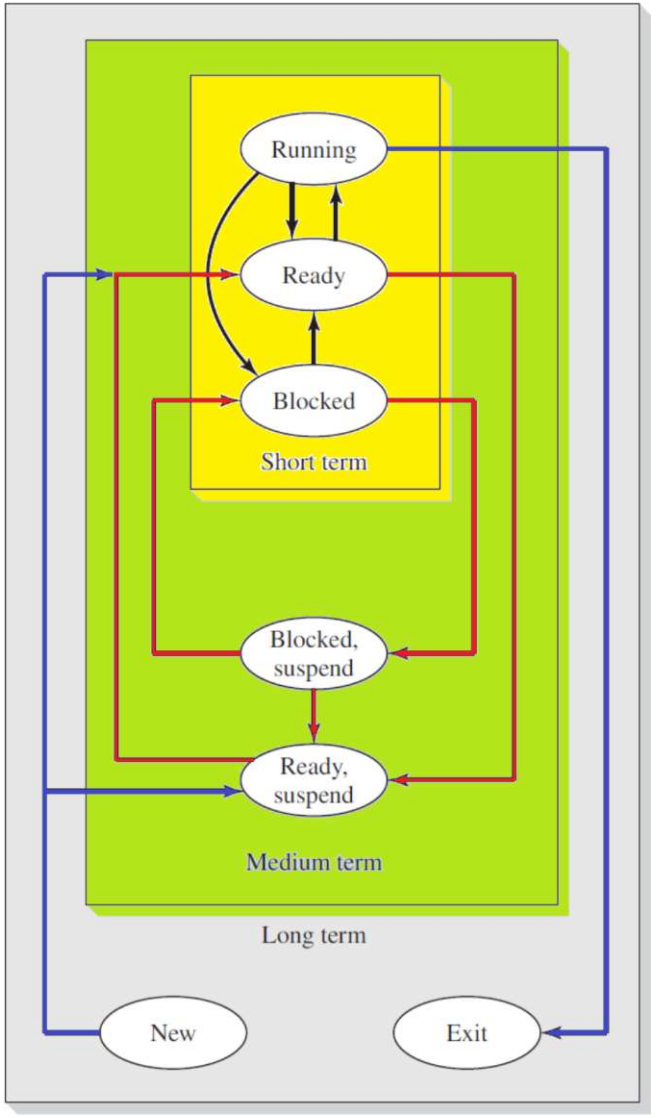
CSE 306 Operating Systems
Scheduling

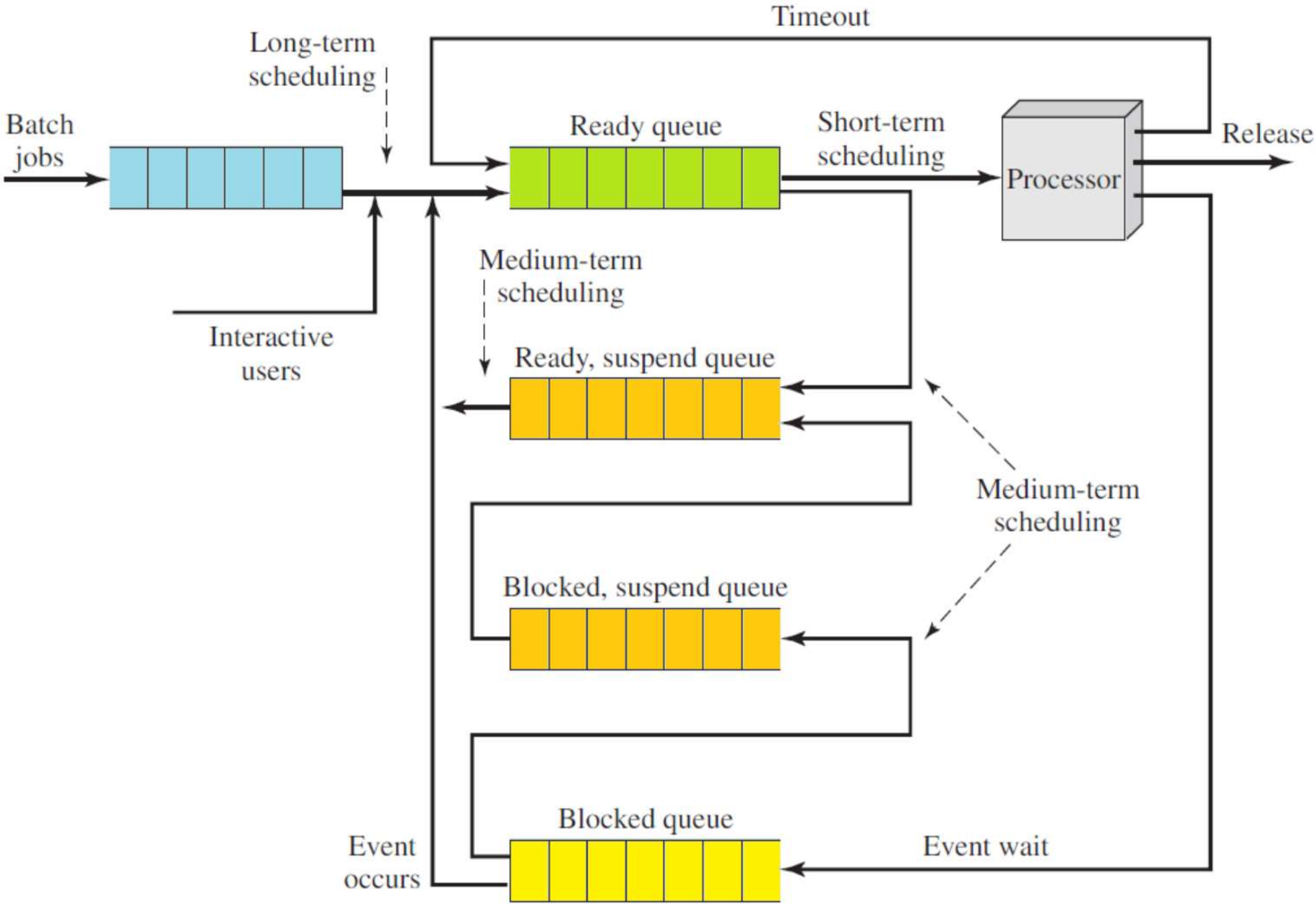
YoungMin Kwon

Types of Scheduling

- Long-term scheduling
  - Decide whether to add new processes to the pool of processes to be executed
- Medium-term scheduling
  - Decide the number of processes that are partially or fully in memory (swapping function)
- Short-term scheduling
  - Which available process will be executed by the processor



Queueing Diagram for Scheduling



Long-Term Scheduling

- Decides the degree of multiprogramming
- Batch system
  - When to take one or more additional processes based on
    - Percentage of time each process is executing
    - Fraction of time the processor is idle
  - Which job to admit next
    - Simple FCFS (First Come First Served)
    - Based on Priority, Execution time, or I/O requirements
- Interactive programs
  - Accept tasks from authorized users until the system is saturated

Medium-Term Scheduling

- Swapping function
  - Medium-term scheduling is a part of swapping function
  - Depends on the degree of multiprogramming
  - Depends on memory requirements

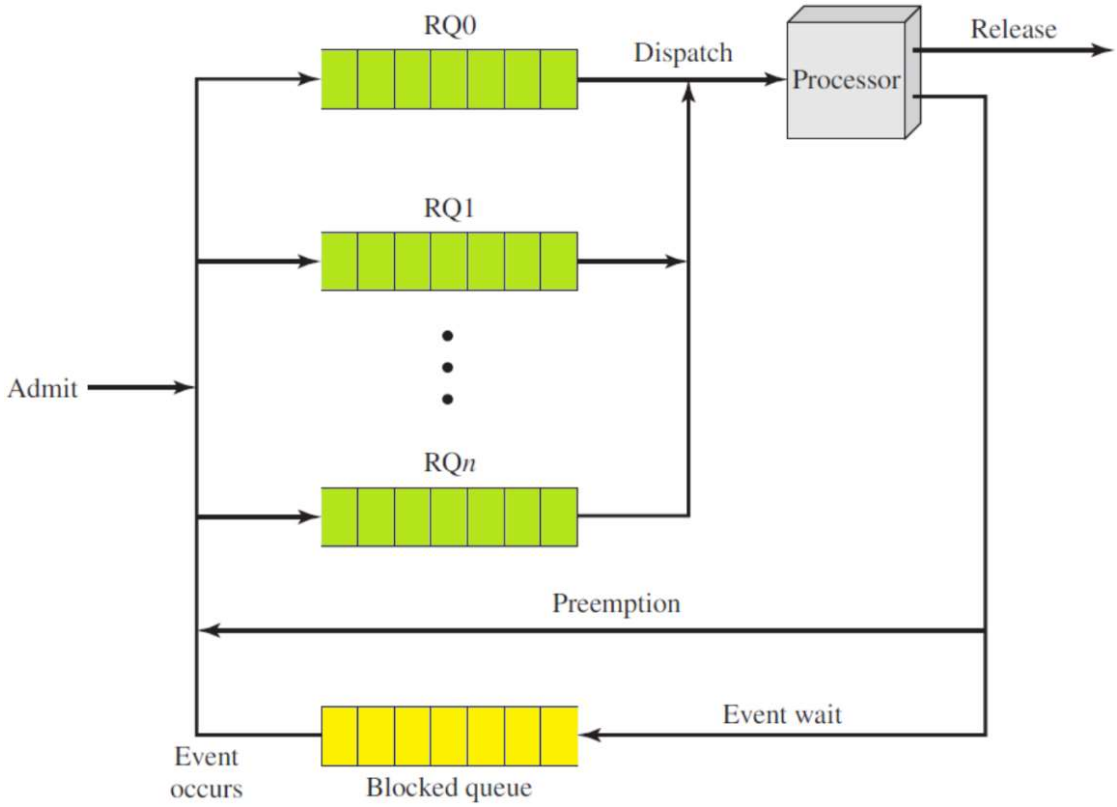
Short-Term Scheduling

- Dispatcher
  - Fine grained decision on which process runs next
- Short-term schedulers are invoked by
  - Clock interrupts
  - I/O interrupts
  - System calls
  - Signals (semaphores)

Short-term Scheduling Criteria

- User-oriented criteria
  - Turnaround time: submission ~ completion,
  - Response time: submission ~ response begins to show,
  - Deadline: meeting the deadline
  - Predictability: same job → same amount of time, same cost
- System oriented criteria
  - Throughput: number of processes completed per time,
  - Processor utilization: % of time the processor is busy,
  - Fairness: processes should be treated the same, no starvation,
  - Enforcing priorities: favor the higher priority processes,
  - Balancing resources: keep the resources busy

Use of Priorities



- Instead of a single ready queue, a set of queues with different priorities is used
- Lower-priority processes may suffer starvation

Scheduling Policies

- First Come First Served (FCFS)
- Round Robin (RR)
- Shortest Process Next (SPN)
- Shortest Remaining Time (SRT)
- Highest Response Ratio Next (HRRN)
- Feedback (FB)

Scheduling Policies

- Selection function
  - Which process in the ready queue is selected next
  - Based on priority, resource requirements, execution characteristics

  - Execution characteristics
    - w: time spent in system so far, waiting
    - e: time spent in execution so far
    - s: total service time required by the process (estimated by the user)

Scheduling Policies

- Decision mode: how the selection function is exercised
  - Non-preemptive: once a process is in the Running state, it continues to execute
    - It terminates or
    - It blocks itself for I/O or some OS service
  - Preemptive: the current running process may be interrupted and moved to the Ready state by OS
    - When a new process arrives
    - When an interrupt occurs
    - Periodically by a clock interrupt

First Come First Served

- The process that has been in the ready queue the longest is selected next
- Non-preemptive
- Favors long processes over short ones

Process	Arrival Time	Service Time (T_s)	Start Time	Finish Time	Turnaround Time (T_r)	T_r/T_s
W	0	1	0	1	1	1
X	1	100	1	101	100	1
Y	2	1	101	102	100	100
Z	3	100	102	202	199	1.99
Mean					100	26

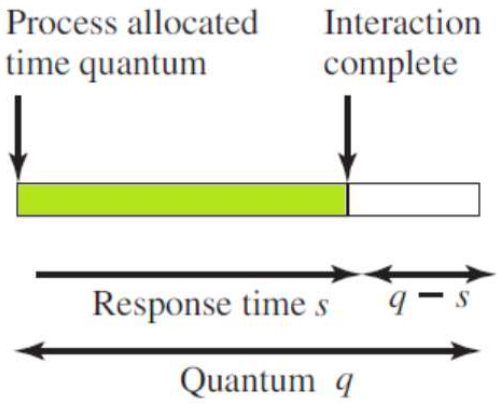
First Come First Served

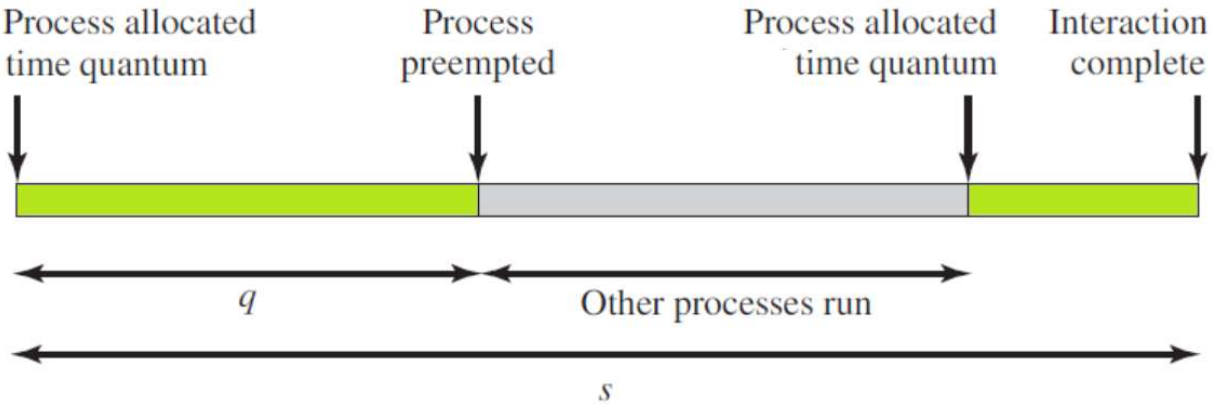
- FCFS favors processor-bound processes over I/O bound processes
  - When a processor-bound process is running, all I/O bound processes must wait
  - When the currently running process leaves the Running state, an I/O bound process runs quickly and becomes blocked again

Round Robin

- Time slicing
  - On a periodic clock interrupt, tasks in the ready queue are selected in FIFO manner
  - Preemptive
  - Unlike FCFS, short tasks do not suffer
  - Scheduling overhead is large for a short quantum
- Favors processor-bound processes over I/O-bound processes
  - I/O-bound processes may not fully use a time quantum
  - Processor-bound processes will fully use a time quantum

Round Robin

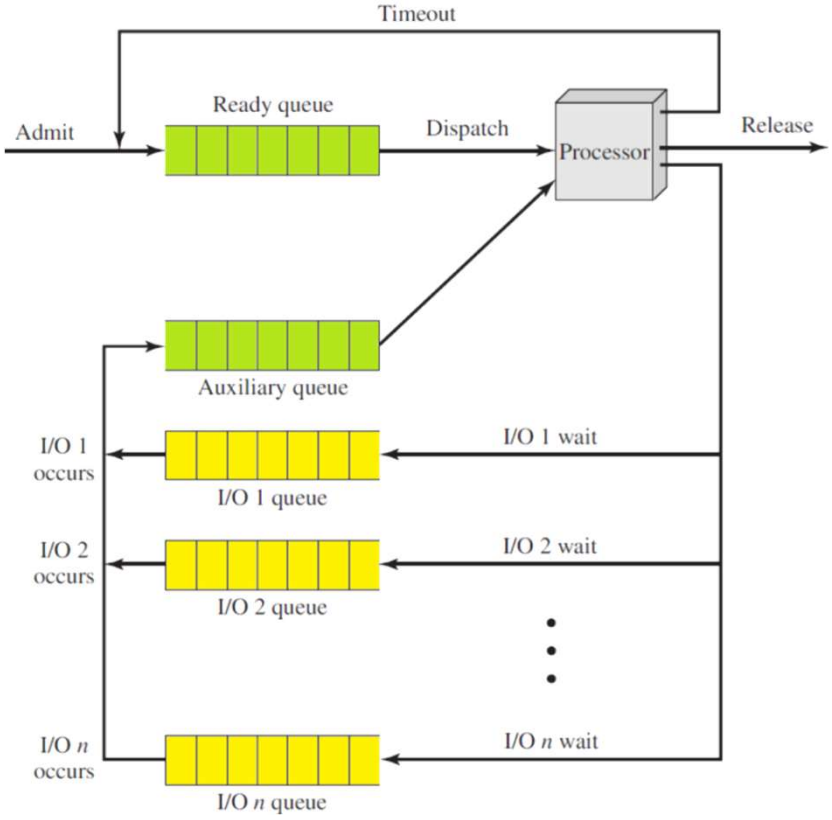




- Time quantum should be slightly larger than the time required for a typical interaction
  - Left: time quantum > typical interaction
  - Right: time quantum < typical interaction

Virtual Round Robin

- Auxiliary queue
  - Has higher priority than Ready queue
  - When unblocked from an I/O operation, processes are moved to Auxiliary queue
  - After dispatched from the Auxiliary queue, a process runs for its remaining time quantum
    - time quantum – time ran before being blocked



  - Solves the unfairness between processor-bound and I/O bound processes

Shortest Process Next

- The process with the shortest expected processing time is selected next
- Non-preemptive
- Reduce the bias in favor of long processes in FCFS
- Possibility of starvation
  - If there is a steady supply of short processes, long processes will starve

Shortest Process Next

- Estimating the processing time (average)

$$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i$$

  - T_i: processor execution time for the i^{th} instance of this process
  - S_i: predicted value for the i^{th} instance

  - To avoid the remembering all T_i's

$$S_{n+1} = \frac{1}{n}T_n + \frac{n-1}{n}S_n$$

- To give greater weight to more recent instances (exponential averaging)

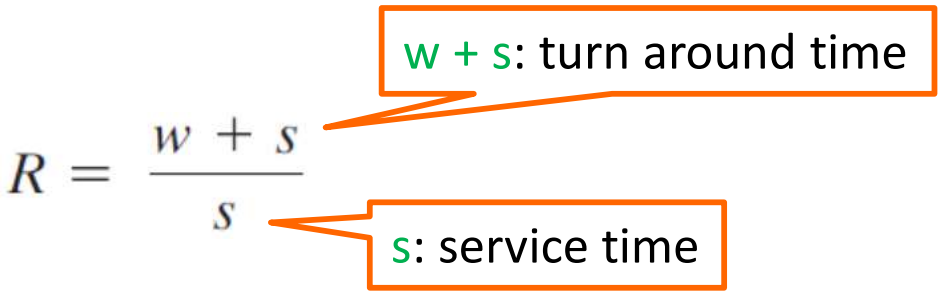
$$S_{n+1} = \alpha T_n + (1-\alpha)S_n$$

Shortest Remaining Time

- When a new process, with shorter remaining time, becomes ready, the scheduler preempt the current process
  - Preemptive version of SPN
  - SRT does not have the bias in favor of long processes found in FCFS
  - Unlike RR, no additional interrupts are generated
  - Elapsed service times must be recorded

Highest Response Ratio Next

- Ratio to use



  - R: response ratio
  - w: time spent waiting for the processor
  - s: expected service time

- Shorter jobs are favored

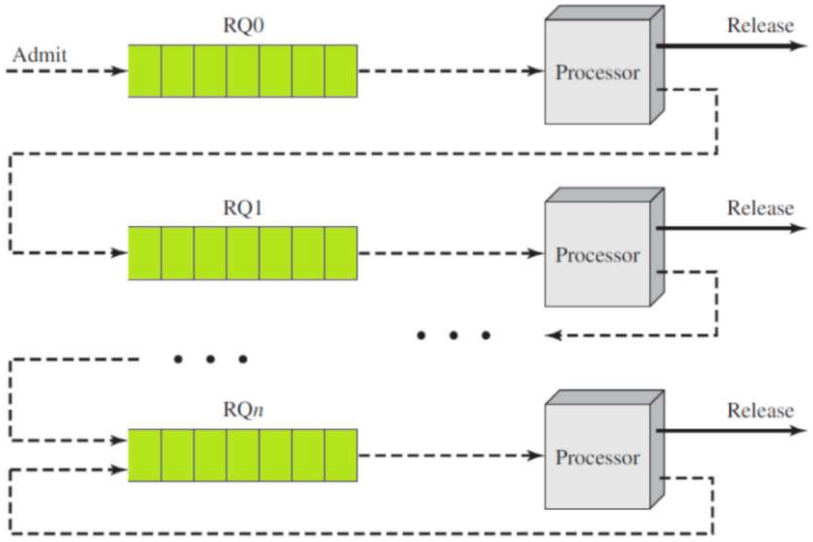
- Aging without service increases the ratio and longer jobs will eventually be competitive

- Non-preemptive

Feedback

- Another way to establish a preference for shorter jobs
  - Use time spent instead of time remaining
  - Works when no information about relative length of processes can be handled
  - Preemptive

Feedback



- Multilevel feedback
  - When a process is first admitted, it is placed in RQ0
  - After its first preemption, it goes to R1
  - On each subsequent preemption, it moves to a lower-priority queue

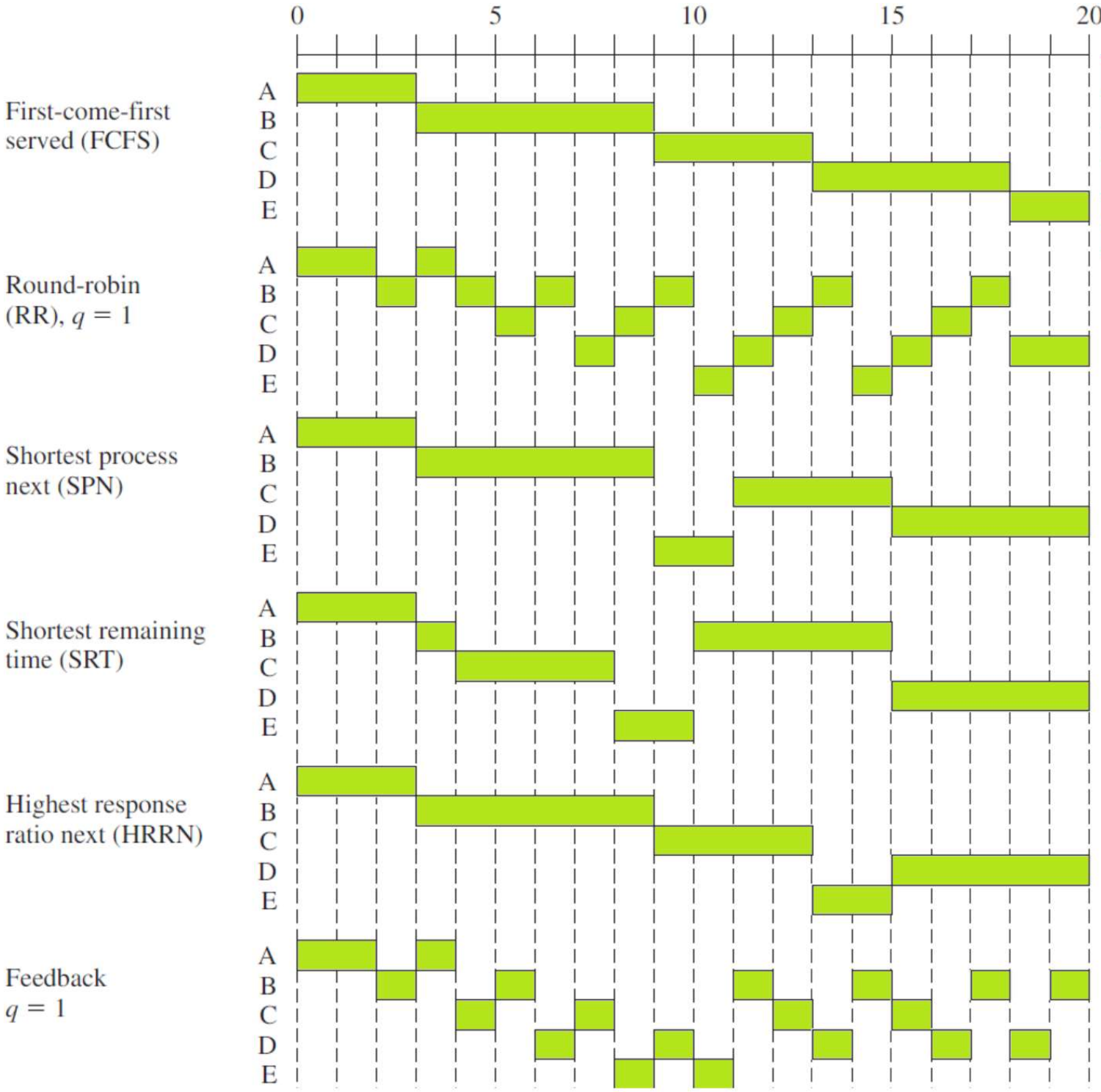
Feedback

- Starvation can occur if new jobs are frequently entering the system
  - Reduce the preemption time for higher-priority queues
  - Promote a process to a higher-priority queue after it waits for a service for a certain duration

	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

Process Scheduling Example

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



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

Fair-Share Scheduling

- In a multiuser system
  - individual jobs can be multi-processes or multi-threaded
  - From the user's perspective, the concern is
    - not how a particular process performs,
    - but how the set of processes perform
- Fair-share scheduling
  - Make schedule decisions based on process sets
  - The concept can be extended to group of users
  - Give fewer resources who have had more than their fair share, and more to those who had less than their fair share

Fair-Share Scheduling

- For a process j in group k at epoch i

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$GCPU_k(i) = \frac{GCPU_k(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

  - $CPU_j(i)$: measure of processor utilization by process j through interval i
  - $GCPU_k(i)$: measure of processor utilization of group k through interval i
  - $P_j(i)$: priority of process j (lower value means higher priority)
  - $Base_j$: base priority of process j
  - W_k: weight of group k: $0 < W_k \leq 1$ and $\sum_k W_k = 1$

Fair-Share Scheduling

- Example
  - Process A is in one group,
    Process B and C are in a second group
  - Weights W_k are 0.5 for each group
  - Base priority is 60
  - Processor utilization
    - The processor is interrupted 60 times per sec
    - The processor usage fields of the current process and the corresponding group are incremented
  - Once per a second, priorities are recalculated

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$GCPU_k(i) = \frac{GCPU_k(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

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	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			Group 2					

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$GCPU_k(i) = \frac{GCPU_k(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

$Base = 60$, $W = 0.5$