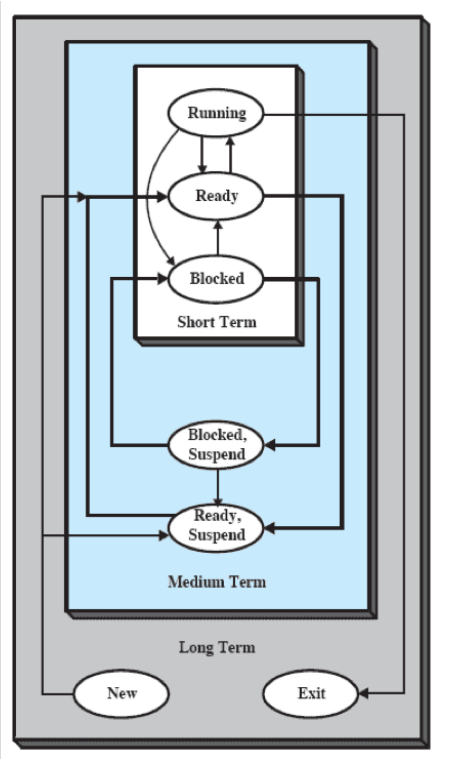
CS415 Module 6 Part B - Scheduling Policies

Athens State University

Outline

Contents

The Scheduling Process



- Schedulers
  - Long term, medium-term, and short-term
- Progress occurs in "bursts"
  - CPU bursts
  - I/O bursts

Priority Queuing

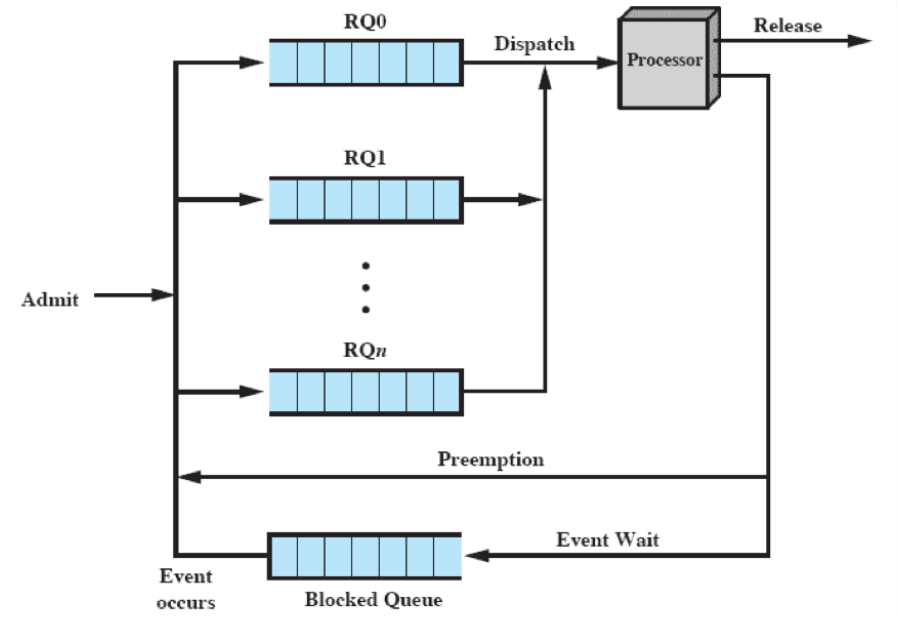


Figure 9.4 Priority Queuing

Selection Function

- Determines which ready thread is selected next for execution
- May be based on priority, resource requirements, or the execution characteristics of the process and thread
- Important execution characteristics
  - Time spent waiting in the system
  - Time spent in execution so far
  - Estimated total service time required by the process

Non-preemptive vs. Preemptive Scheduler

Nonpremptive

- Once a process is in the running state, it continues running until it terminates or must block itself to wait on I/O

Premptive

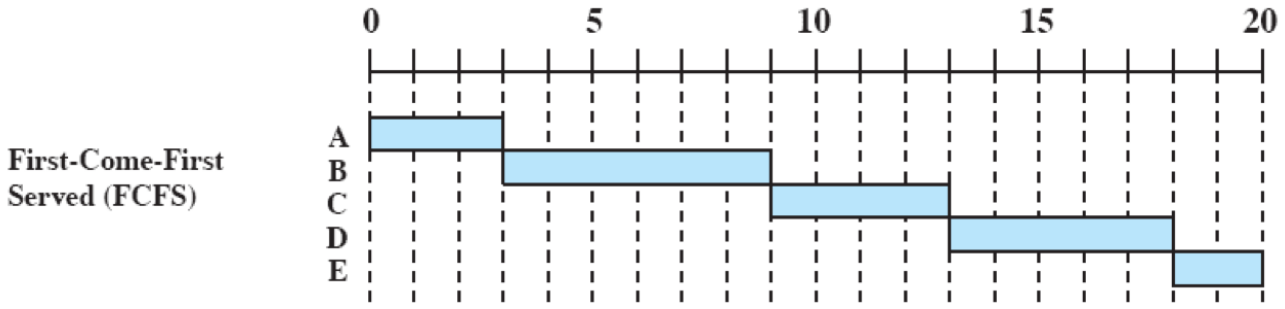
- Currently running process may be interrupted and moved to the ready state by the OS
- Preemption may occur when a new process arrives, on an interrupt, or periodically

1 Scheduling Policies

Thread Scheduling Example

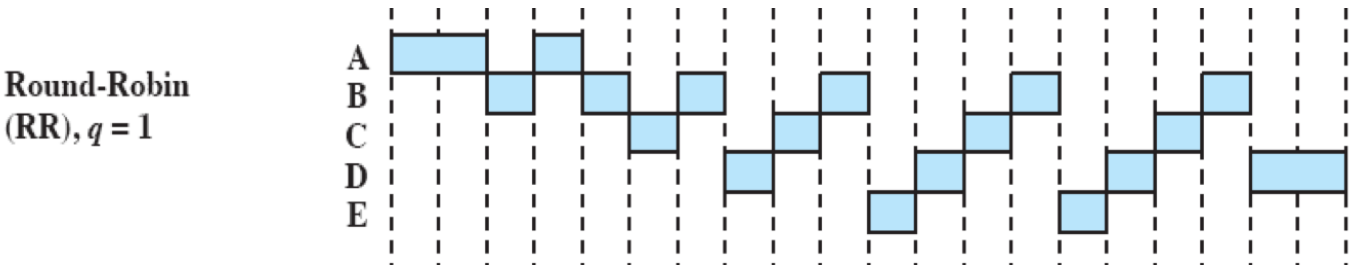
Thread	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

First Come - First Served (FCFS)



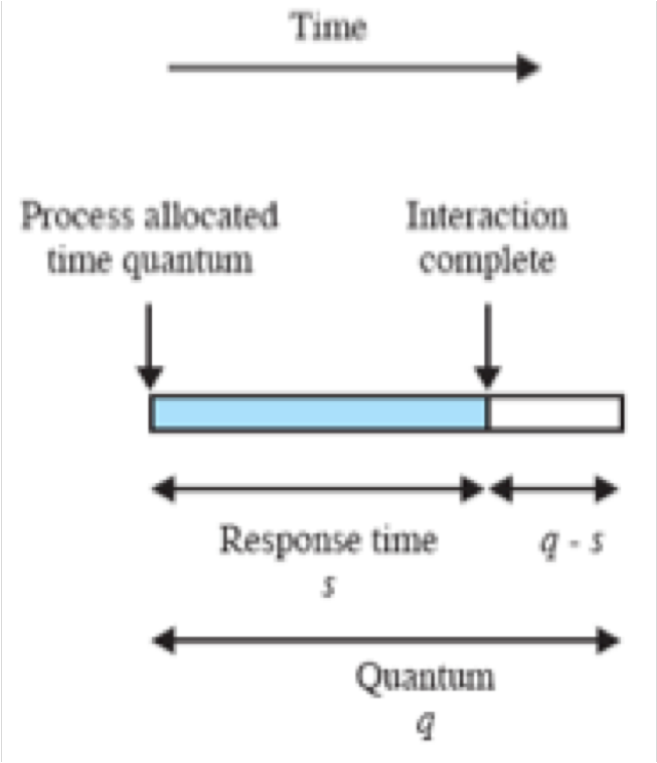
- Simplest policy
- A.k.a: FIFO or strict queuing
- When the current processes ceases to execute, the longest process in the Ready queue is selected
- Performs much better for long threads than short ones
- Tends to favor CPU-bound threads over I/O-bound threads

Round Robin



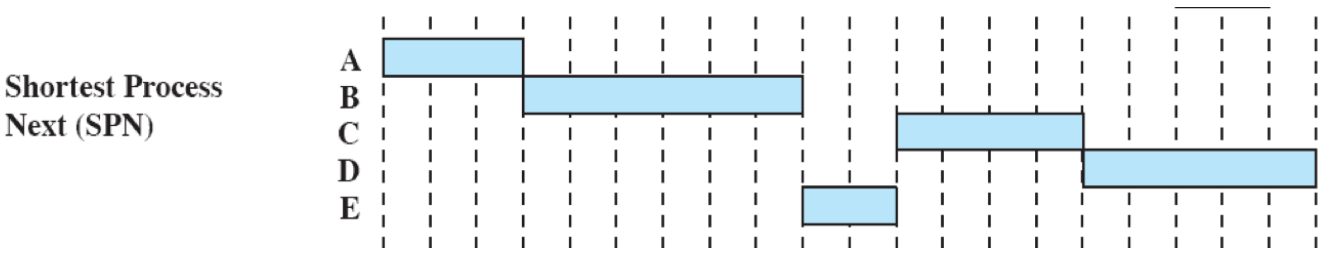
- Preemptive, based to time clock
- A.k.a: time slicing
- Principal design issue is the length of the time slice
- Effective for time-sharing or transaction processing systems
- Drawback is relative treatment of processor-bound and I/O threads

Quantum Size



- Time slice needs to be just slightly greater than length of a thread
  - Allows threads to be kept within a single time slice
  - If time slice is larger than longest running thread, then RR devolves into FCFS

Shortest Job First



- Nonpremptive, choose thread with smallest expected processing time
- Short processes jump to head of queue
- Possibility that longer-lived threads may starve
- Difficultly: need to know, or at least estimate, thread's required processing time
- System may abort a thread that should an estimate be much less than actual

Guessing The Length Of The Next CPU Burst

- Can only estimate the length, should be similar to the previous cycle
- Estimate using exponential averaging
  - t_n be the length of the n-th CPU burst
  - τ_n predicted value for next CPU burst
  - Set a constant $0 \leq \alpha \leq 1$, usually $\frac{1}{2}$
  - $\tau_{n+1} = \alpha * t_n + (1 - \alpha) * \tau_n$

Exponential Smoothing Coefficients

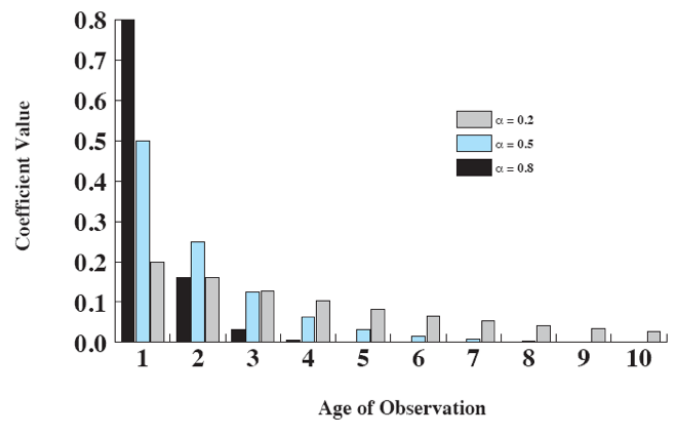
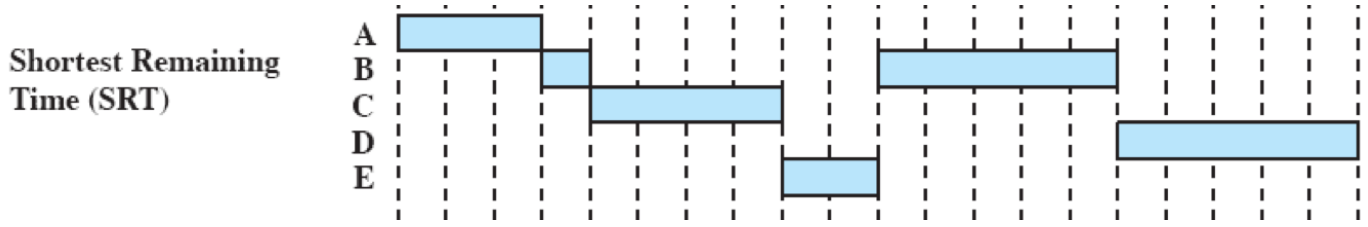


Figure 9.8 Exponential Smoothing Coefficients

The larger the value of the coefficient, the greater the weight assigned to the more recent observation

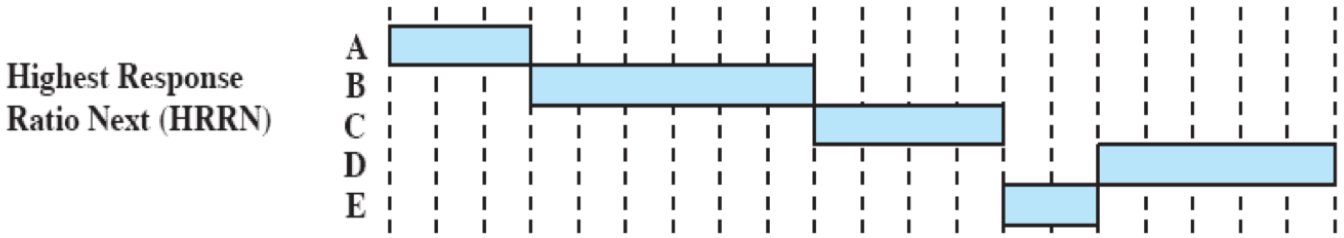
Shortest Remaining Time (SRT)



- Preemptive version of SJF

- Choose thread with shortest expected remaining time
- May starve longer-running threads
- Should give better performance to SJF as a short job is given immediate preference over a longer running job

Highest Response Ratio Next (HRRN)



$$R = \frac{t_w + t_s}{t_s}$$

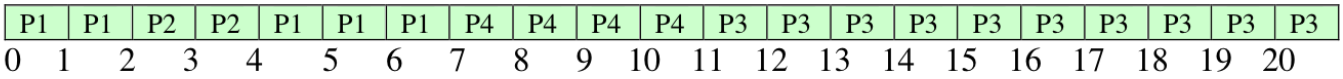
- Chooses next process with greatest ratio
- Attractive because it considers aging of threads
- Favors shorter jobs, aging without service increases the ration so that a longer-running thread will eventually get past shorter jobs

Priority Scheduling

- A priority number is associated with each thread
- CPU allocated to the thread with highest priority (preempt or non-premptive)
- SJF is priority scheduling where priority is the inverse of the predicted next CPU burst time
- Must age threads so that as time progresses the priority of the thread increases. Avoids low priority starvation

Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

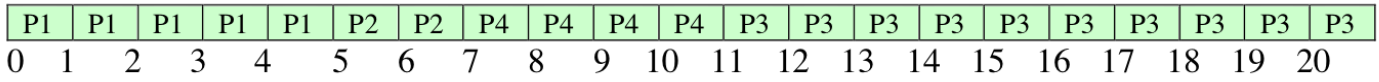
Shortest Time First



P1 starts but is preempted after 20ms when P2 arrives and has shorter burst time (20ms) than the remaining burst time of P1 (30 ms) . So, P1 is preempted. P2 runs to completion. At 40ms P3 arrives, but it has a longer burst time than P1, so P1 will run. At 60ms P4 arrives. At this point P1 has a remaining burst time of 10 ms, which is the shortest time, so it continues to run. Once P1 finishes, P4 starts to run since it has shorter burst time than P3.

Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

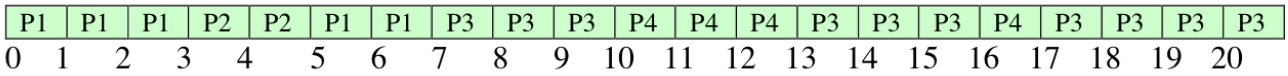
Non-preemptive Priority



P1 starts, but as the scheduler is non-preemptive, it continues executing even though it has lower priority than P2. When -69-P1 finishes, P2 and P3 have arrived. Among these two, P2 has higher priority, so P2 will be scheduled, and it keeps the processor until it finishes. Now we have P3 and P4 in the ready queue. Among these two, P4 has higher priority, so it will be scheduled. After P4 finishes, P3 is scheduled to run.

Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

Round Robin with Quantum of 30 ms



P1 arrives first, so it will get the 30ms quantum. After that, P2 is in the ready queue, so P1 will be preempted and P2 is scheduled for 20ms. While P2 is running, P3 arrives. Note that P3 will be queued after P1 in the FIFO ready queue. So when P2 is done, P1 will be scheduled for the next quantum. It runs for 20ms. In the mean time, P4 arrives and is queued after P3. So after P1 is done, P3 runs for one 30 ms quantum. Once it is done, P4 runs for a 30ms quantum. Then again P3 runs for 30 ms, and after that P4 runs for 10 ms, and after that P3 runs for 30+10ms since there is nobody left to compete with.

2 Multilevel Queues

Multilevel Queues

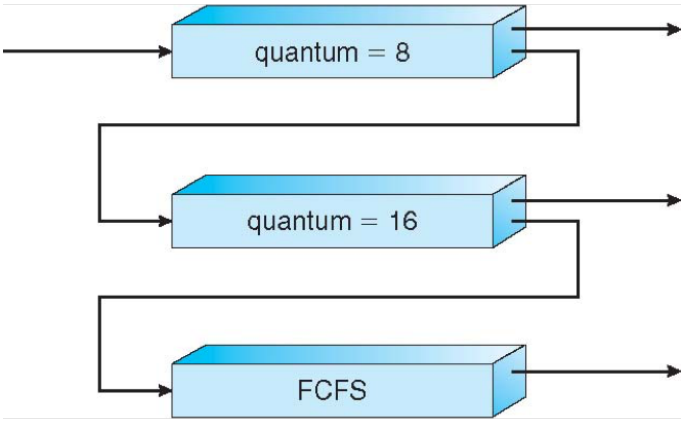
- Separate the ready queue into a foreground and background queue
- Each queue has it's own scheduling policy
  - For example, the foreground queue is round robin scheduled while the background is FCFS
- Scheduling is done between the queues
  - Fixed priority: foreground first, then background
  - Time slice: Give each queue a certain amount of CPU time (i.e., 80% to foreground queue in RR, 20% to background in FCFS)

Multilevel Feedback Queue

- A thread can move between the various queues
- Multilevel Feedback Queue schedulers are defined by
  - Number of queues
  - Scheduling algorithm for each queue
  - Method used to determine when to upgrade a thread
  - Method used to determine when to downgrade a thread
  - Method used to determine queue placement of a thread

Multilevel Feedback Queue - Example

- Three queues: q_1 is RR with 8ms slice, q_2 is RR with 16ms slice, and q_3 is FCFS
- Scheduling
  - New jobs enter q_1 and is served RR
    - Job gets 8ms on CPU; moved to q_2 if can't complete
  - Job is again served RR and gets 16ms. If still doesn't complete, move to q_3



3 Key Points

Key Points

- Scheduling policies
- Different types of scheduling policies
- Impact of each on thread execution