OSTEP Chapter 8

ECE 3600, Fall 2022

Table of Contents

1. Multi-Level Feedback Queue

Try to minimize response time and turnaround time

Priority based on observed behavior (history)

Round-robin for equal priorities

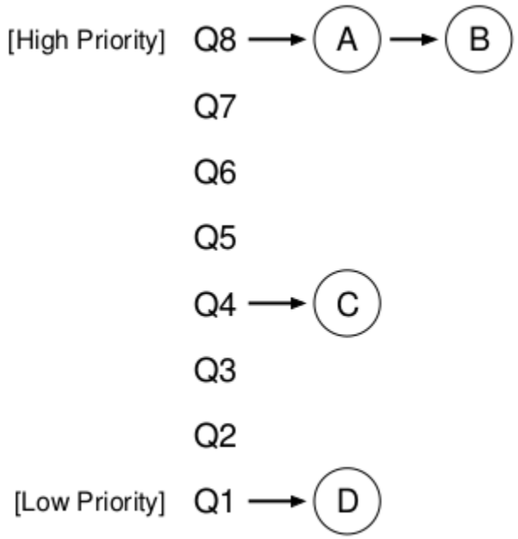

Figure 8.1: **MLFQ Example**

2. Long-Running Jobs

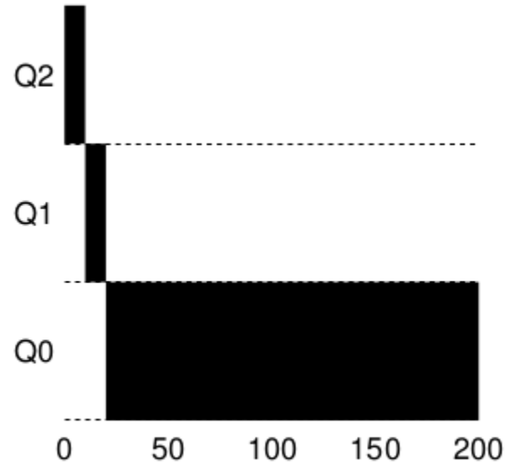

Figure 8.2: **Long-running Job Over Time**

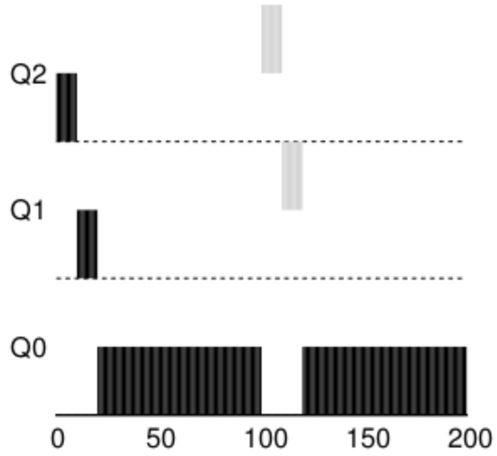

Figure 8.3: **Along Came An Interactive Job**

3. I/O vs. CPU-intensive Workloads

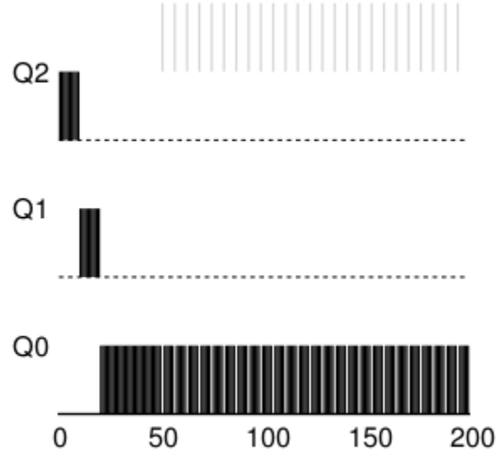

Figure 8.4: **A Mixed I/O-intensive and CPU-intensive Workload**

4. Priority Boost

Avoid starvation

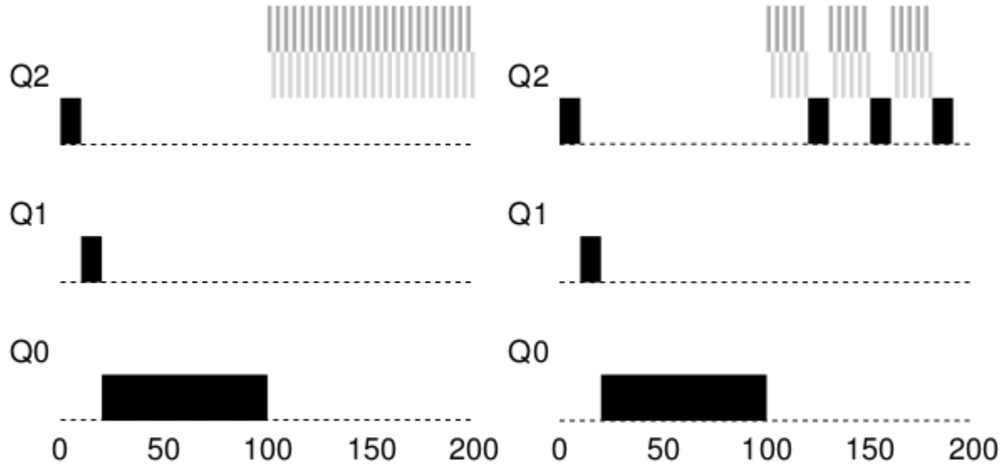

Figure 8.5: **Without (Left) and With (Right) Priority Boost**

5. Gaming Tolerance

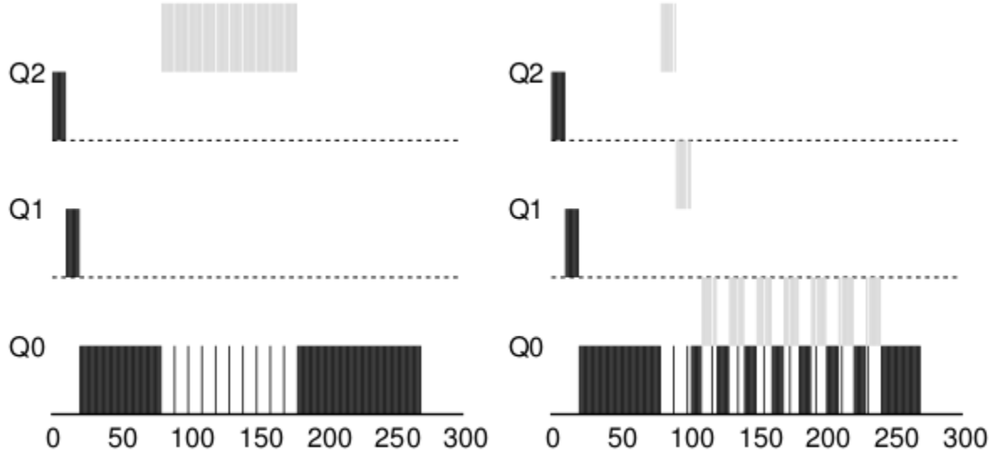

Figure 8.6: **Without (Left) and With (Right) Gaming Tolerance**

6. Non-Uniform Time Slices

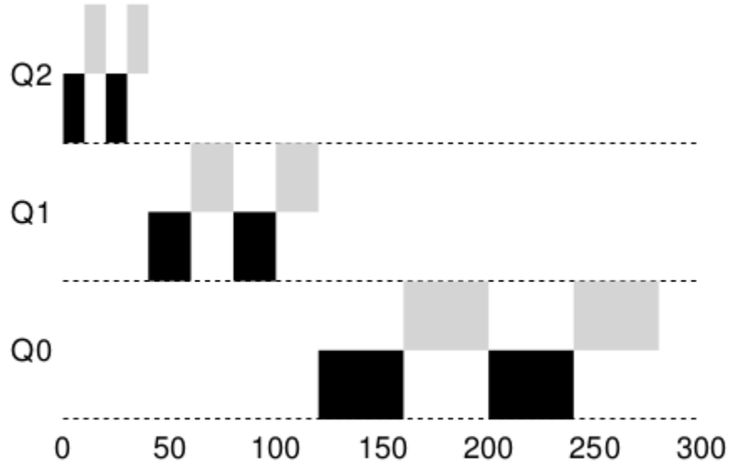

Figure 8.7: **Lower Priority, Longer Quanta**

7. Summary

MLFQ Rules:

1. If Priority(A) > Priority(B), A runs (B doesn't).
2. If Priority(A) = Priority(B), A & B run in round-robin fashion using the time slice (quantum length) of the given queue.
3. When a job enters the system, it is placed at the highest priority (the topmost queue).
4. Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced (i.e., it moves down one queue).
5. After some time period S, move all the jobs in the system to the topmost queue.

Other Scheduling Policies: (skip)

Chapter 9: fair-share, lottery scheduling (random), stride scheduling (deterministic)

Chapter 10: multiprocessor scheduling

8. Exercises

Exercises from the book using mlfq.py:

1. Run a few randomly-generated problems with just two jobs and two queues; compute the MLFQ execution trace for each. Make your life easier by limiting the length of each job and turning off I/Os.

2. How would you run the scheduler to reproduce each of the examples in the chapter?

3. How would you configure the scheduler parameters to behave just like a round-robin scheduler?