ITP 30002 Operating System

Multi-level Feedback Queue

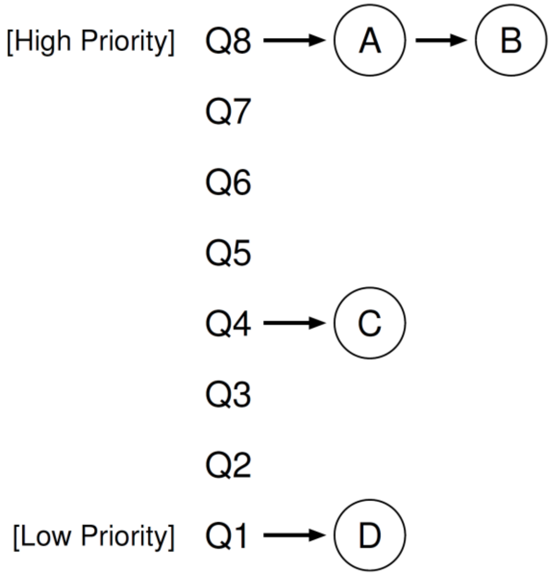
OSTEP Chapter 8

Shin Hong

Multi-level Feedback Queue (MLFQ)

- classify processes into multiple levels wrt their interactiveness
- run priority scheduling for processes across different levels, and run fair scheduling for processes of the same level
  - aim to optimize both performance (turnaround time) and responsiveness (response time) at the same time
- predict CPU-burst time that a process has based on history

MLFQ Mechanism



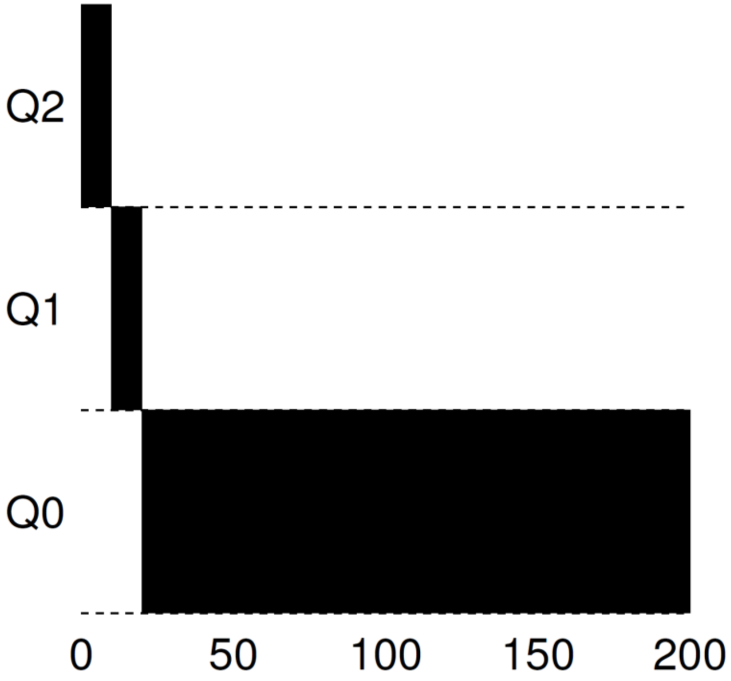
- Run multiple ready queues
  - each queue is assigned with a unique priority number
  - processes in the same queue have the same priority
- Scheduling algorithms: which process to dispatch next?
  - Rule 1. schedule from the non-empty queue of the highest priority
  - Rule 2. choose a process from the selected queue in a RR manner
- How to decide to which queue a process is given?
  - assigned by the user
  - determined by the observed behavior of the process

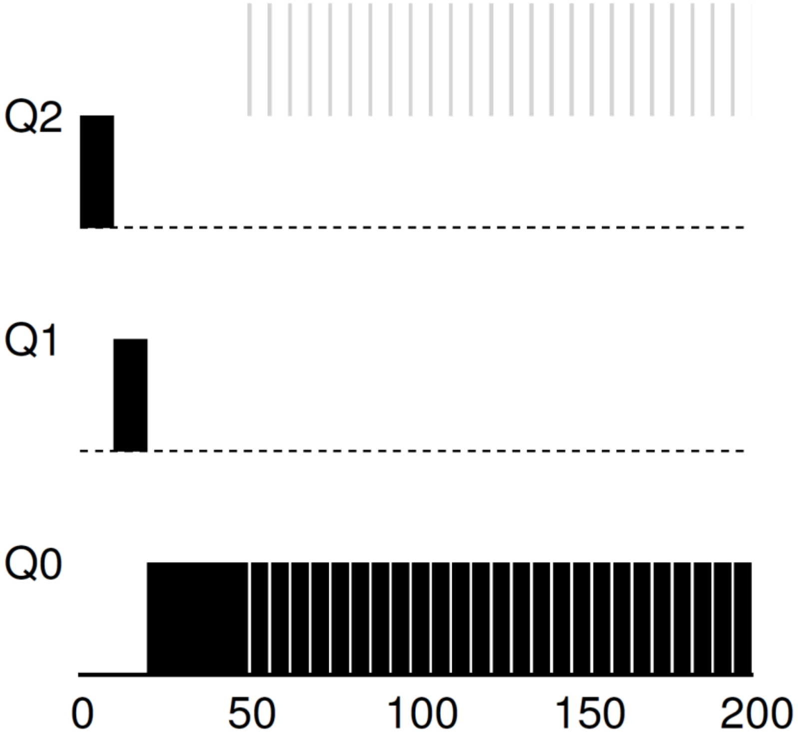
Change Priority Level of Process

- Observations
  - an interactive process runs for a short period of time and goes blocked (relinquishes the CPU) frequently
  - a CPU-intensive process uses up a given CPU time and gets preempted frequently

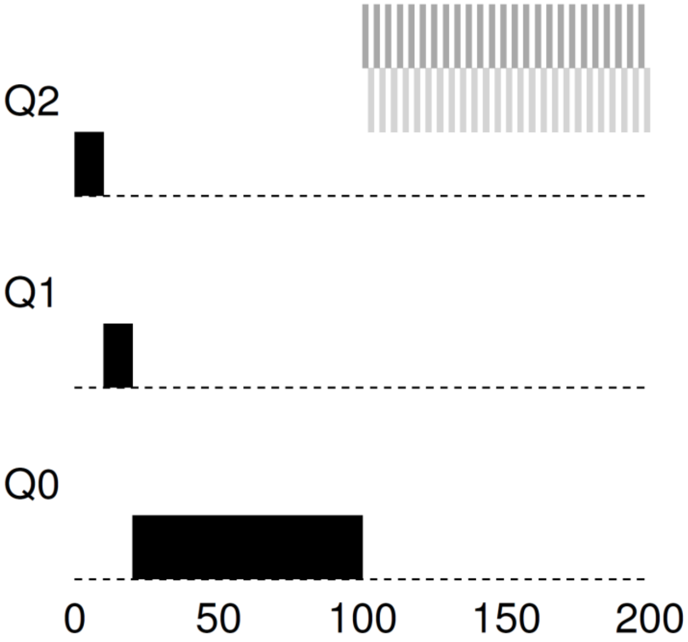
- Scheduling algorithm for controlling process priority
  - Rule 3. a new process is initially placed at the highest priority
  - Rule 4. a process is degraded to one level lower if it uses up a time slice
  - Rule 5. a process stays at the same priority if it releases a CPU without preemption

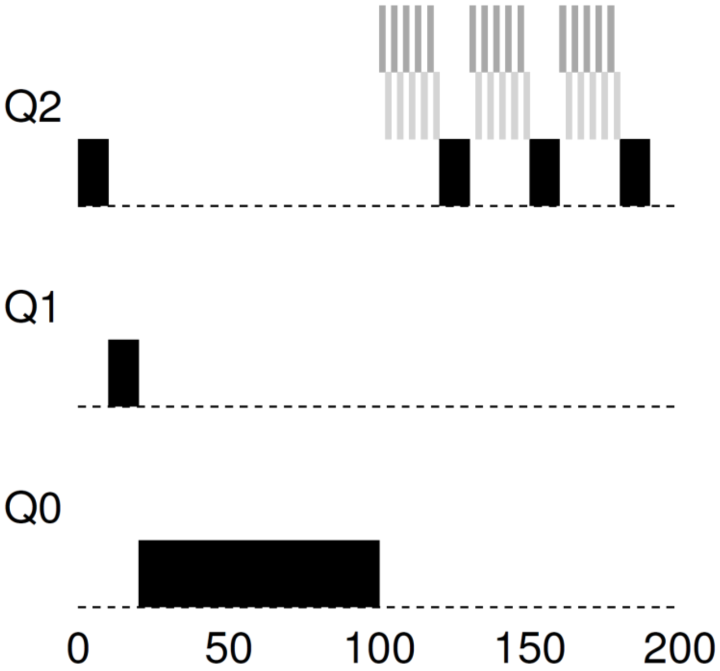
Examples





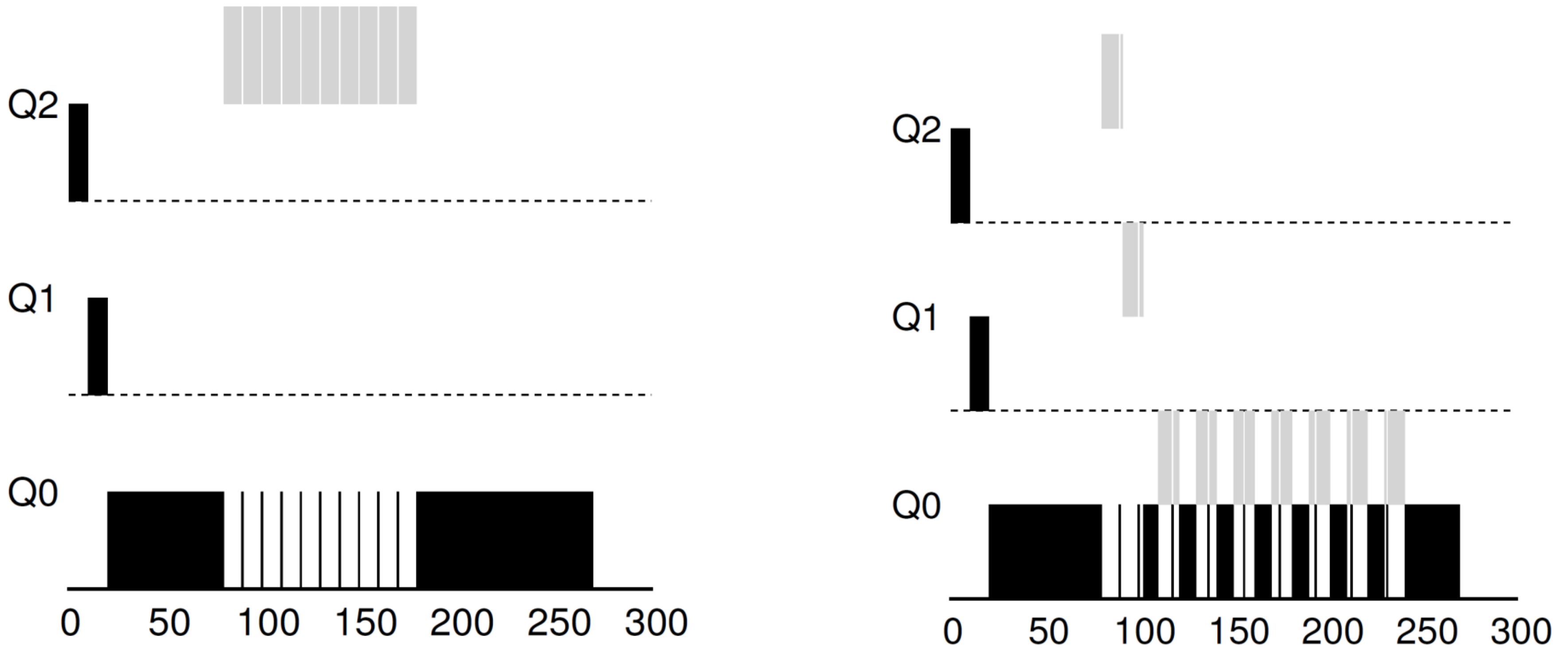
Problems of Priority Scheduling





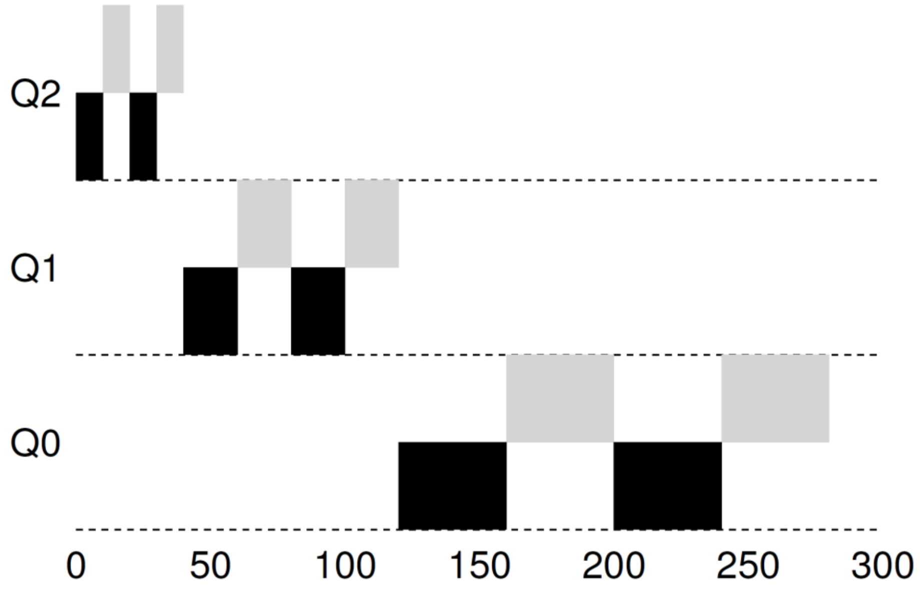
- **Problems**
  - non-interactive process can be left out from scheduling if there are too many interactive ones and suffers starvation
  - a process has no chance to upgrade its priority even if its behavior were changed
- **Solution**: move the priorities of all processes to the top every *S* time

Avoiding Gaming



- **Problem**: A user can program to trick the scheduling by putting meanignless blocking operations to keep the process in a high priority queue
- **Solution**: if a process uses up a time allotment, move it one level down no matter how quickly it was to release the CPU (i.e., aging)

Parameter Tuning



- Example: give a longer time slice for a lower priority queue
- Performance will largely depends on parameters of scheduling policies
  - length of time slice, priority boosting frequency, etc.
  - Problem of *Voo-doo* constants
- Some systems use hints or commands from the user at scheduling

c.f. Danger of Priority Scheduling

- Priority scheduler dispatches a ready process of the highest priority at a time where each process is associated with a certain priority number
  - often used for real-time system to provide strong completion time guarantee

- Under priority scheduling, multiple processes can be stuck (i.e., deadlock) if a process with a higher priority is waiting for a resource held by a process with a lower priority
  - E.g., *What really happend on Mars Rover PathFinder* by Mike Jones, Risks Digests, 1997