5118020-03 Operating Systems

Multi-level Feedback Queue

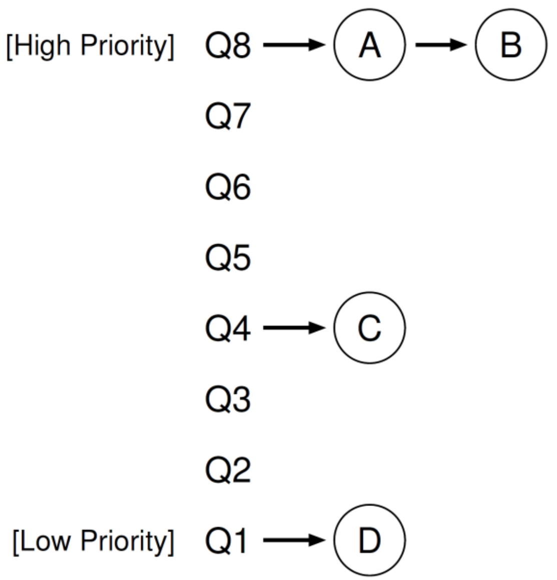
OSTEP Chapter 8

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
  - aims to achieve short response time of interactive processes, and also to achieve short turnaround time of computation-intensive processes
- predict next period of CPU-burst time of a process based on history

MLFQ Mechanism



- Use 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. find the highest non-empty priority
  - Rule 2. select a process in a RR manner from the selected queue

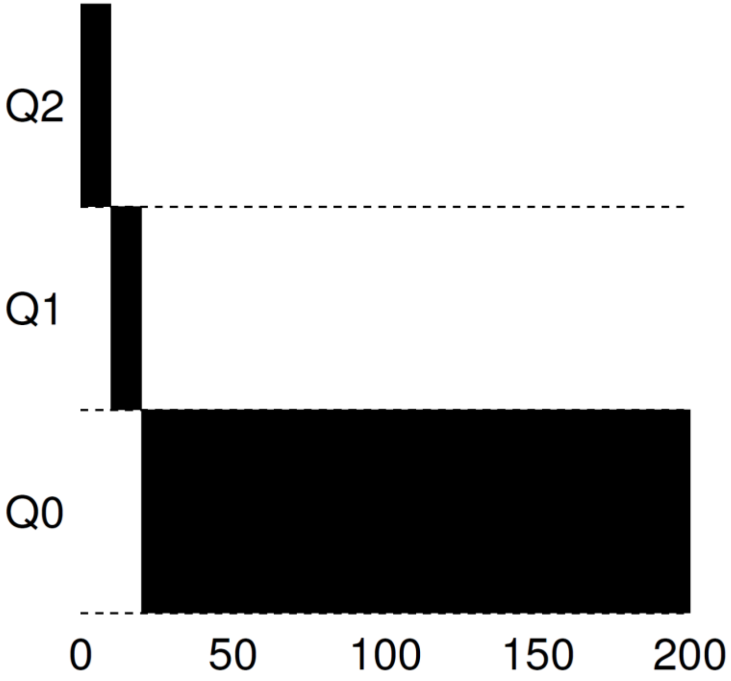
- How to decide to the priority (queue) a process belongs?
  - assigned by the user
  - determined by the observed behaviors of runnable processes

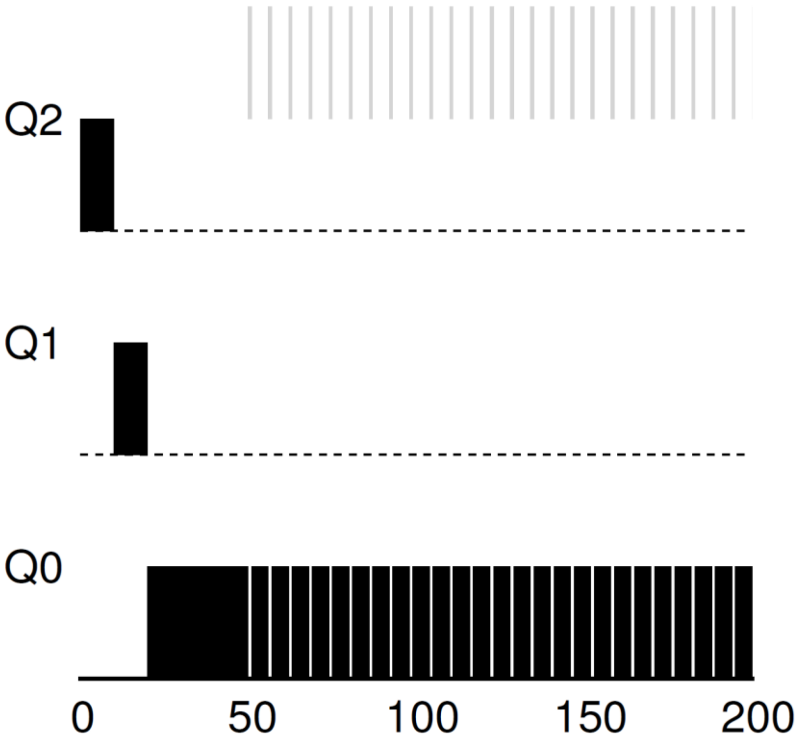
Change Priority Level of Process

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

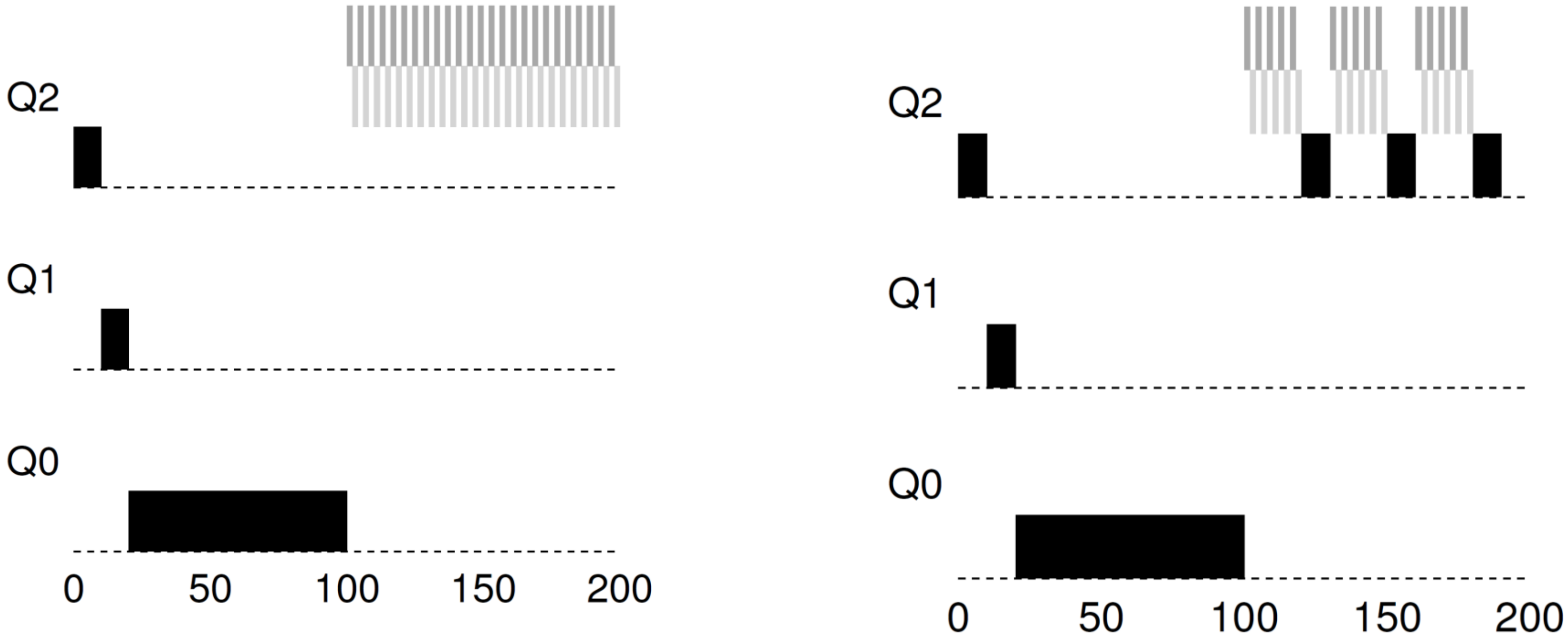
- Scheduling algorithm for controlling process priority
  - Rule 3. a new process is initially placed at the highest priority
  - Rule 4. a process is degraded to one level low priority if it uses up a time slice
  - Rule 5. a process stays at the same priority if it releases a CPU without preemption

Examples



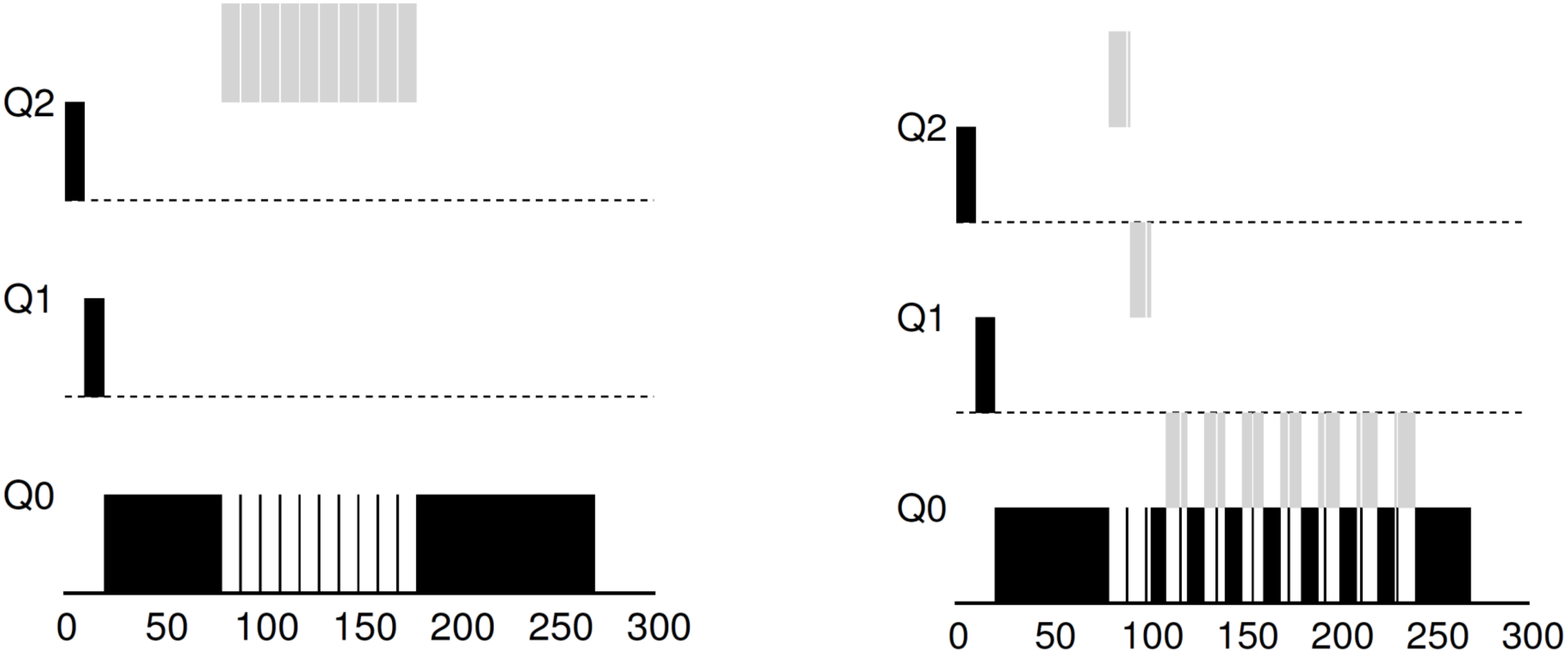


Problems of Priority Scheduling



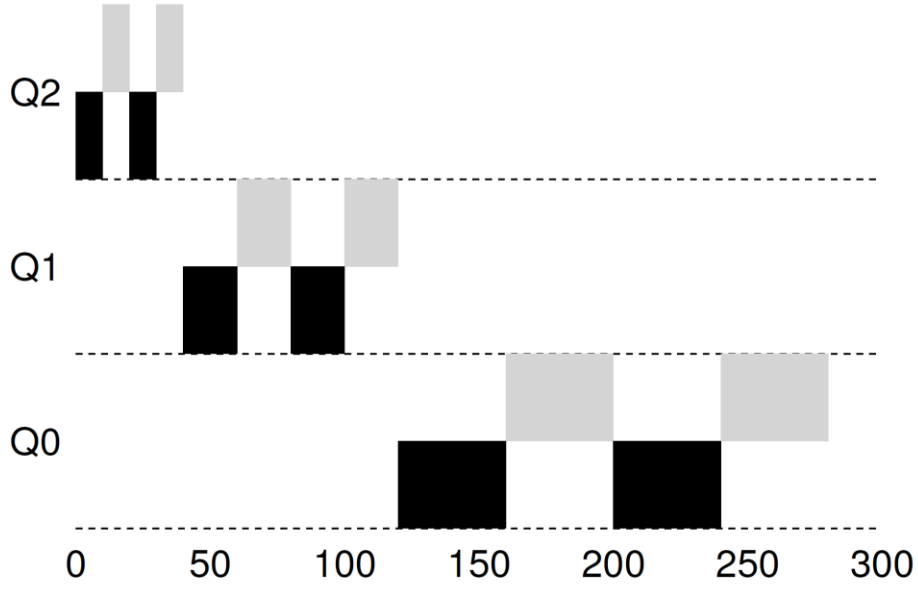
- **Problems**
  - non-interactive processes can be left out from scheduling if there are too many interactive ones; they suffers starvation
  - a process has no chance to upgrade its priority even if its behavior changes
- **Solution**: periodically move the priorities of all processes to the top, every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

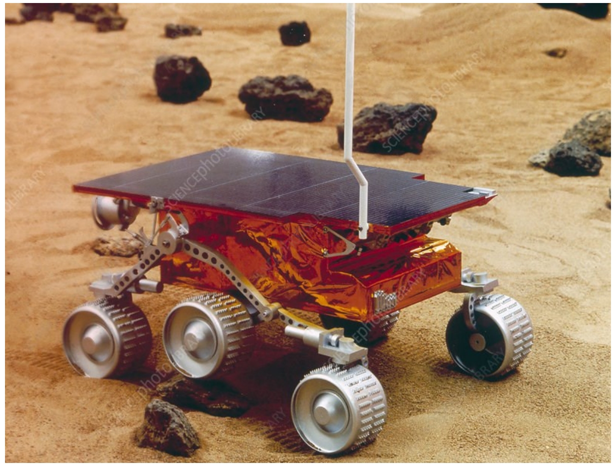


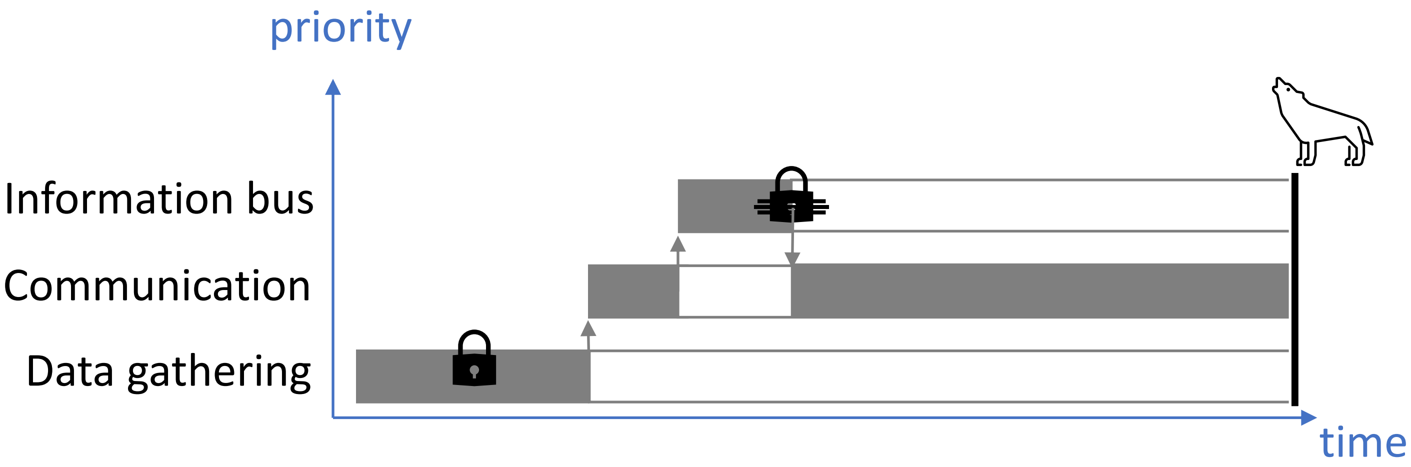
- Performance will largely depend on parameters of scheduling policies
  - length of time slice, priority boosting frequency, etc.
  - Ex.: give a longer time slice to a process in a lower priority queue
  - Problem of *Voo-doo* constants
- Some systems use hints or commands from the user at scheduling

c.f. Danger of Priority Scheduling

- Priority scheduler strictly dispatches a ready process of the highest priority at a time
  - often used in real-time systems for 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 happened on Mars Rover PathFinder* by Mike Jones, Risks Digests, 1997




priority
Information bus
Communication
Data gathering
time