Operating Systems
Lecture 6: CPU Scheduling Policies

Nipun Batra

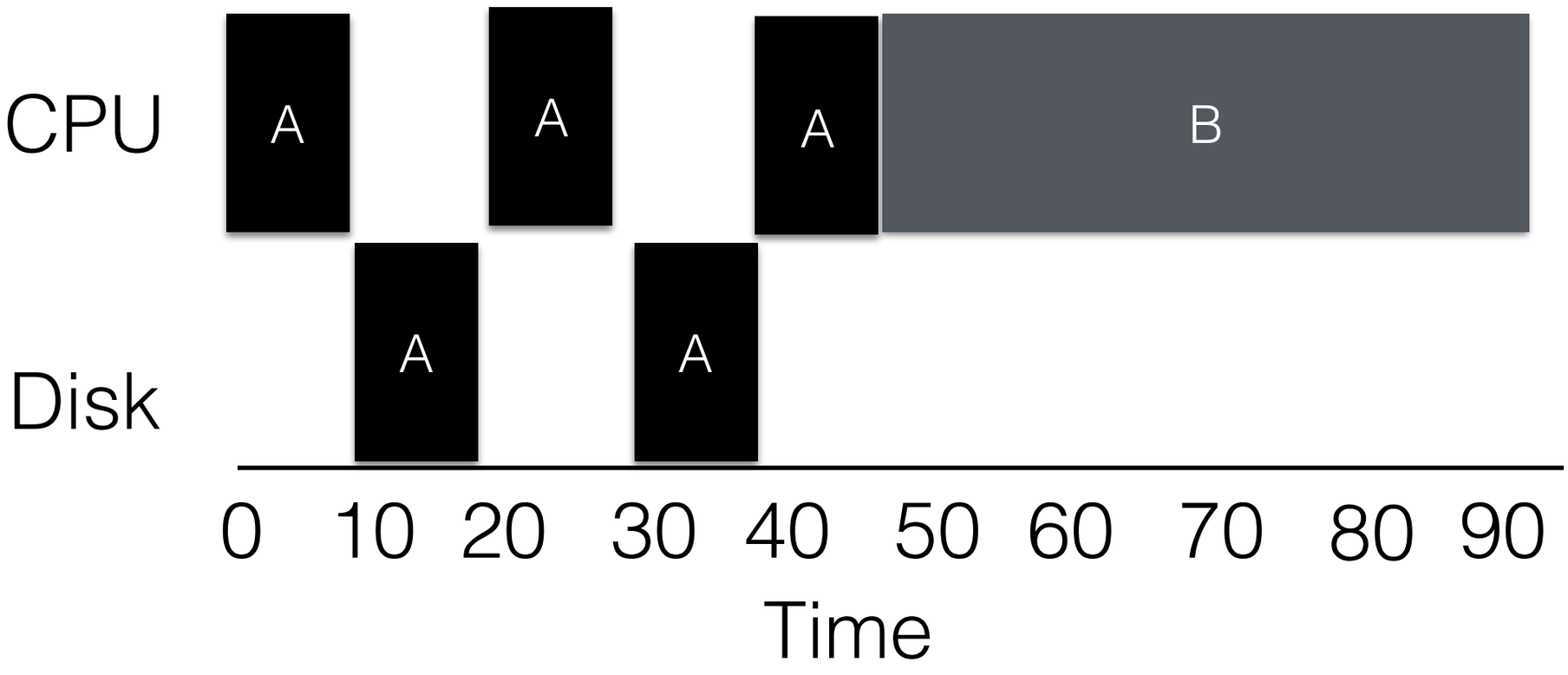
Workload Assumptions

1. ~~Each job runs for the same time~~
2. ~~All jobs arrive at the same time~~
3. ~~Once started, each job runs to completion~~ (Pre-emptible)
4. All jobs use only the CPU
5. Run time of each job is known

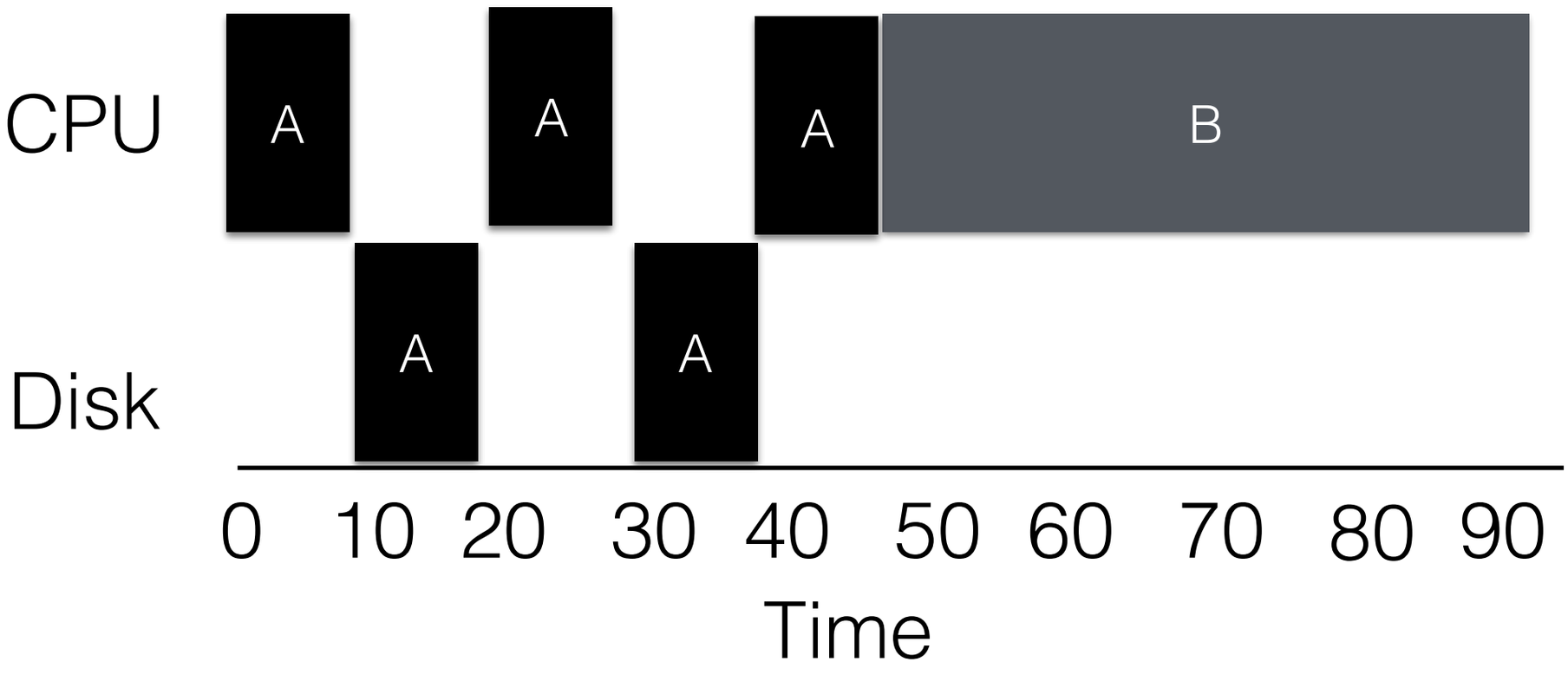
Workload Assumptions

1. ~~Each job runs for the same time~~
2. ~~All jobs arrive at the same time~~
3. ~~Once started, each job runs to completion~~ (Pre-emptible)
4. ~~All jobs use only the CPU~~
5. Run time of each job is known

Incorporating IO

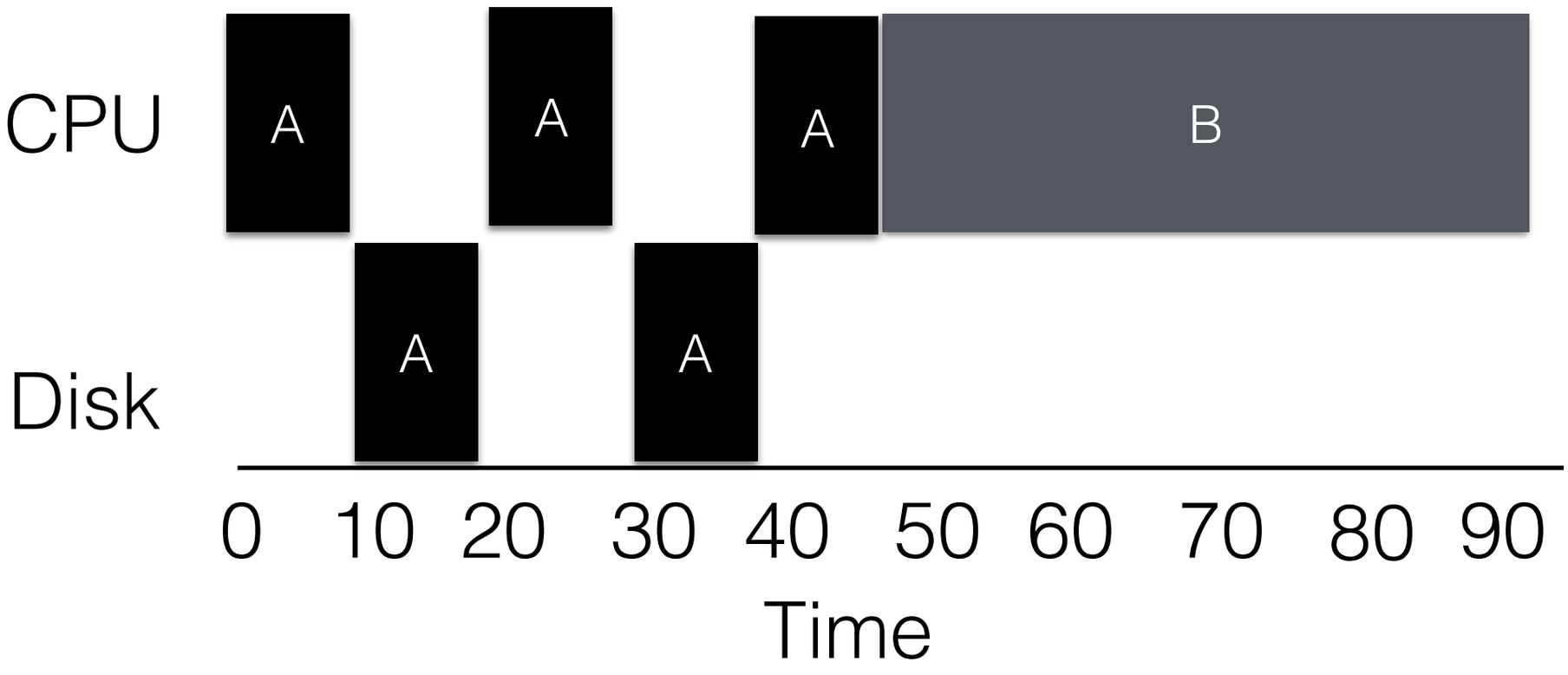


Incorporating IO



CPU utilisation (%) = (30+40)*100%/90 ~77%

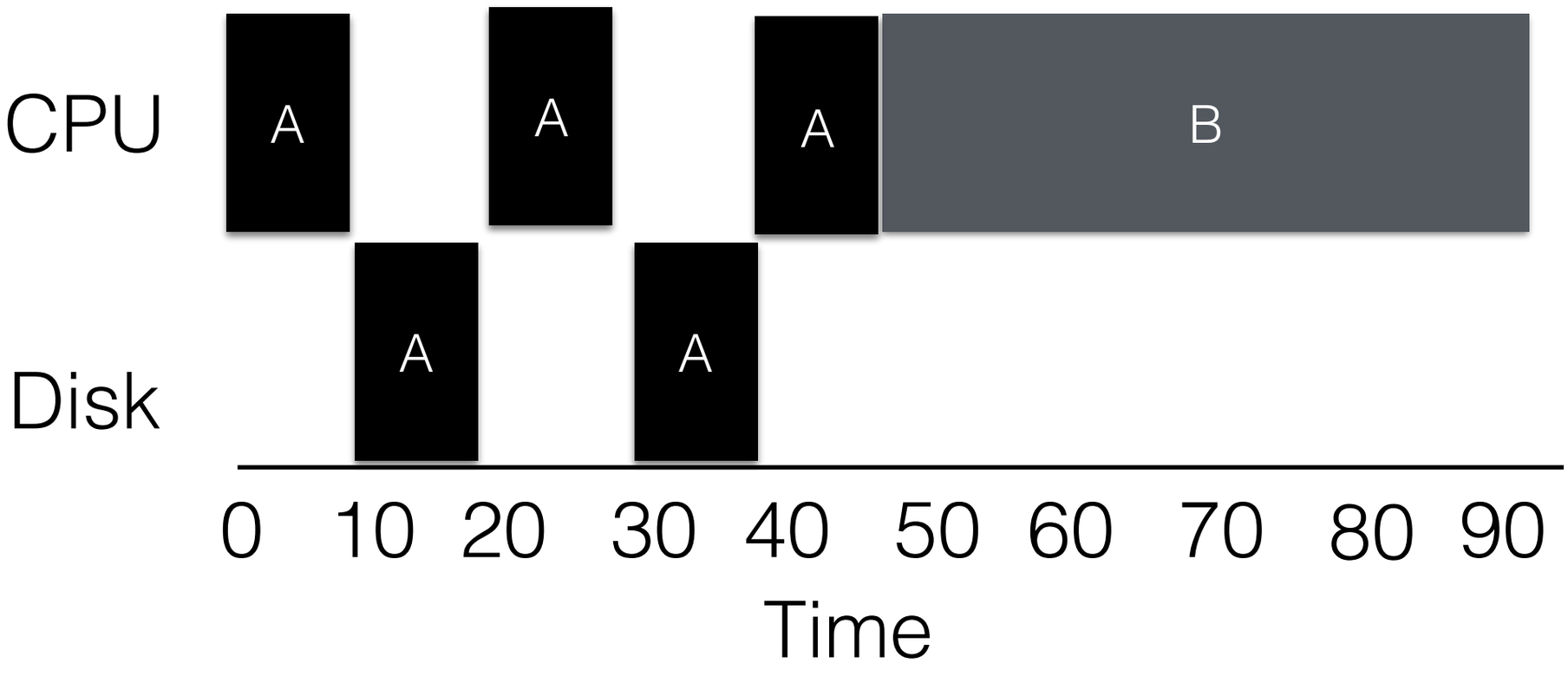
Incorporating IO



CPU utilisation (%) = (30+40)*100%/90 ~77%

Avg. Response Time = (0+50)/2 =25

Incorporating IO

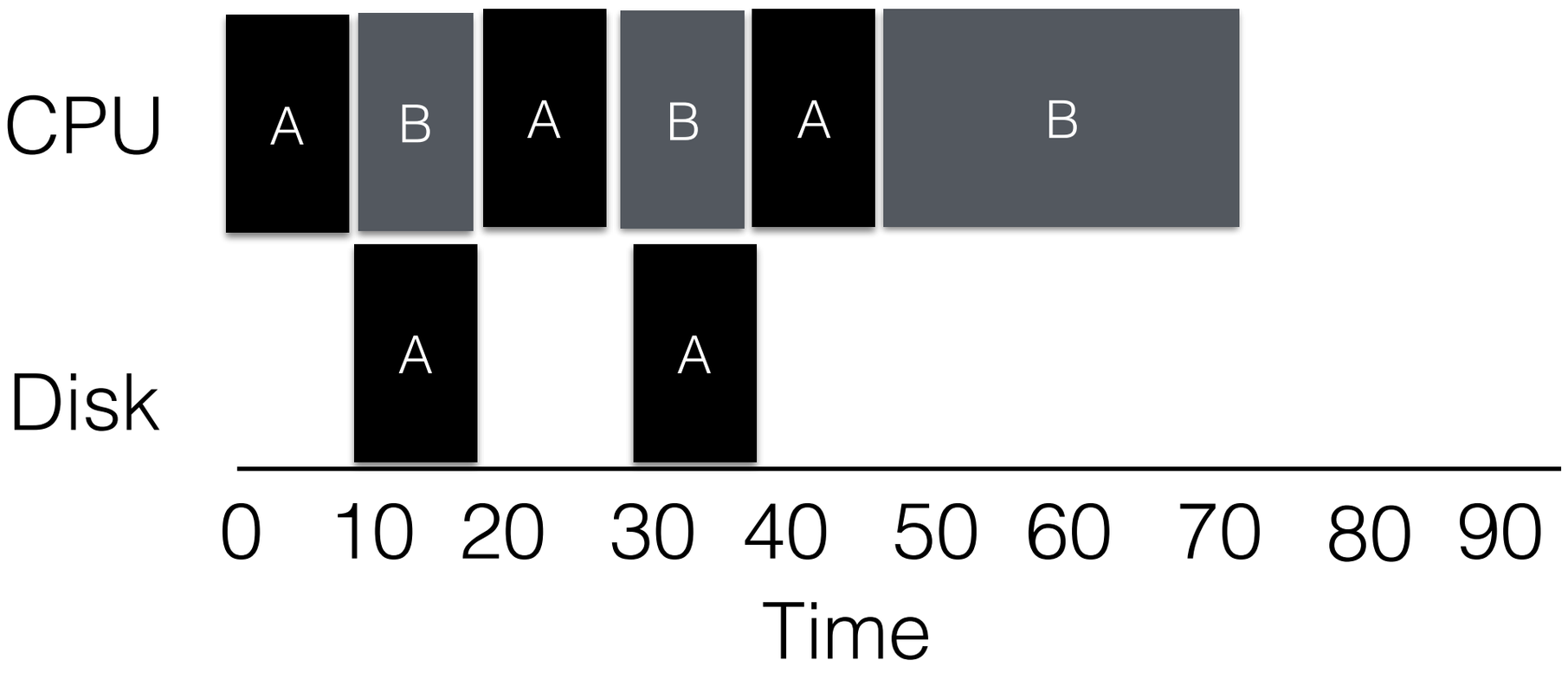


CPU utilisation (%) = (30+40)*100%/90 ~77%

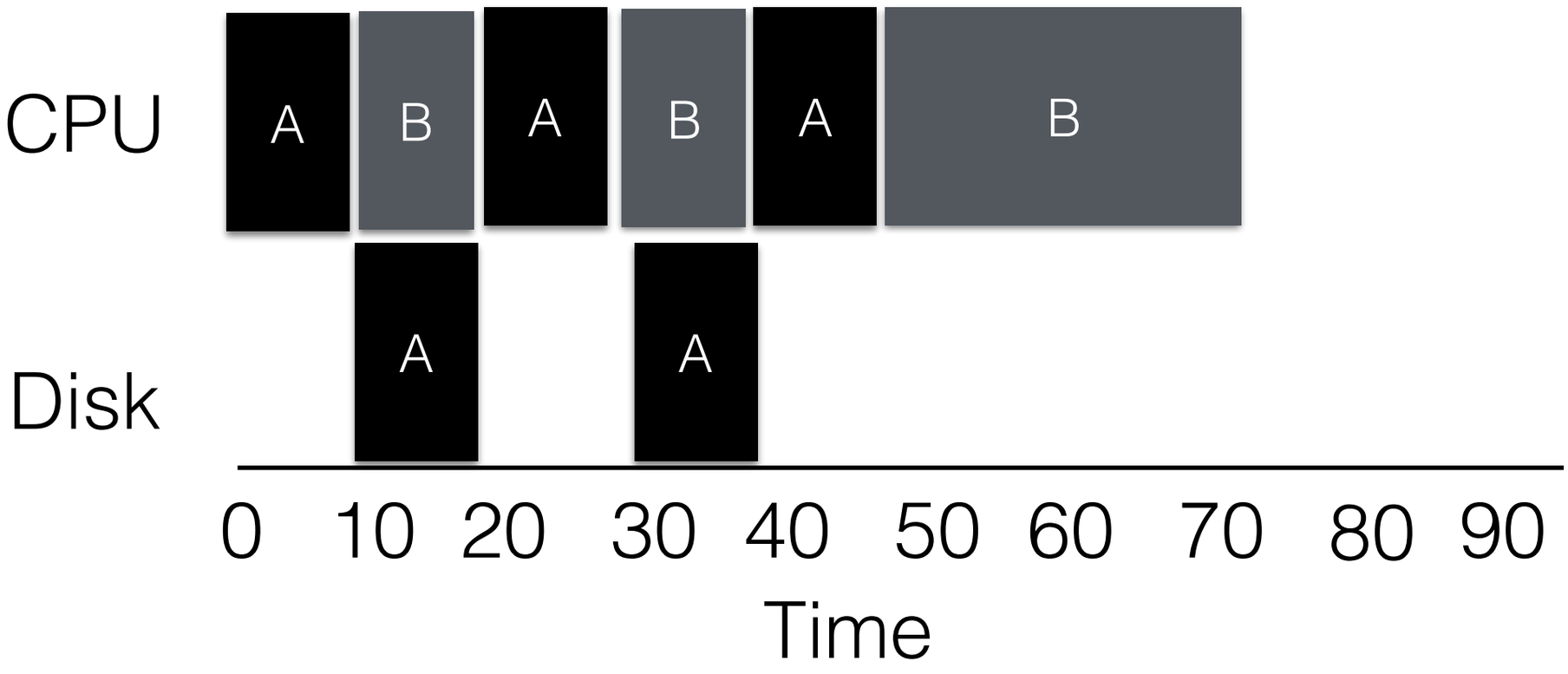
Avg. Response Time = (0+50)/2 =25

Avg. Turnaround Time = (50+90)/2 =70

Incorporating IO

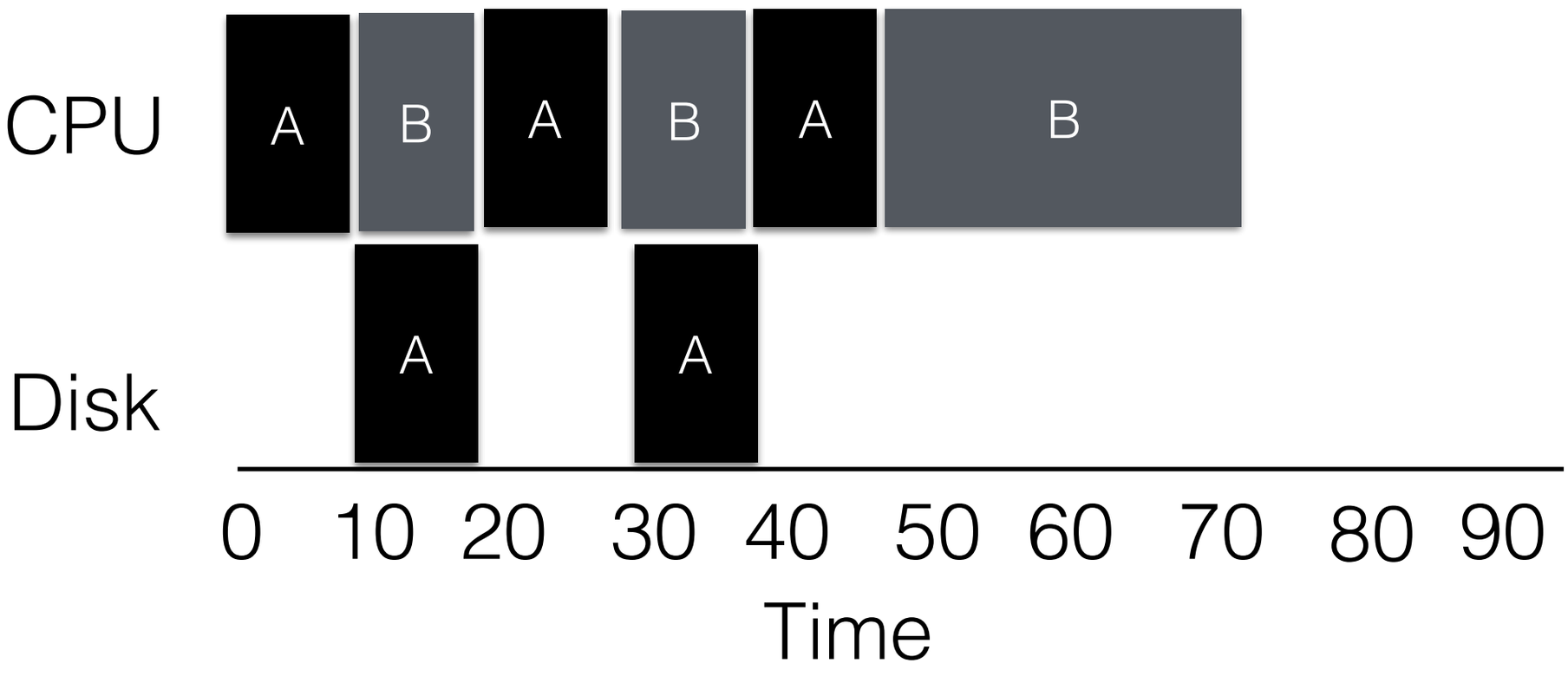


Incorporating IO



CPU utilisation (%) = (30+40)*100%/70 =100%

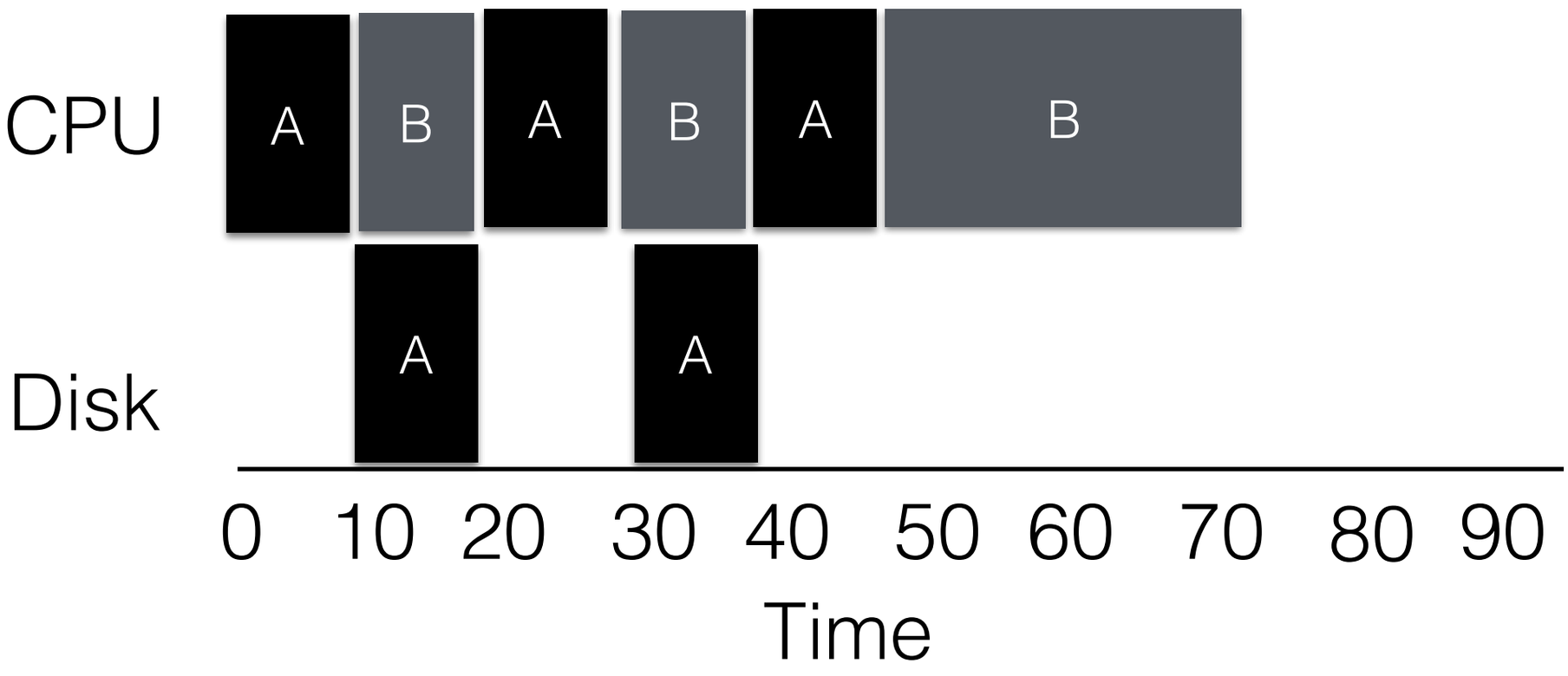
Incorporating IO



CPU utilisation (%) = (30+40)*100%/70 =100%

Avg. Response Time = (0+10)/2 =5

Incorporating IO



CPU utilisation (%) = (30+40)*100%/70 =100%

Avg. Response Time = (0+10)/2 =5

Avg. Turnaround Time = (50+70)/2 =60

Practice

1. Compute the response time and turnaround time when running three jobs of length 200 with the SJF and FIFO schedulers.
2. Now do the same but with jobs of different lengths: 100, 200, and 300.
3. Now do the same, but also with the RR scheduler and a time-slice of 1.
4. For what types of workloads does SJF deliver the same turnaround times as FIFO?
5. For what types of workloads and quantum lengths does SJF deliver the same response times as RR?
6. What happens to response time with SJF as job lengths increase? Can you use the simulator to demonstrate the trend?
7. What happens to response time with RR as quantum lengths increase? Can you write an equation that gives the worst-case response time, given N jobs?

Workload Assumptions

1. ~~Each job runs for the same time~~
2. ~~All jobs arrive at the same time~~
3. ~~Once started, each job runs to completion~~ (Pre-emptible)
4. ~~All jobs use only the CPU~~
5. Run time of each job is known

Workload Assumptions

1. ~~Each job runs for the same time~~
2. ~~All jobs arrive at the same time~~
3. ~~Once started, each job runs to completion~~ (Pre-emptible)
4. ~~All jobs use only the CPU~~
5. ~~Run time of each job is known~~

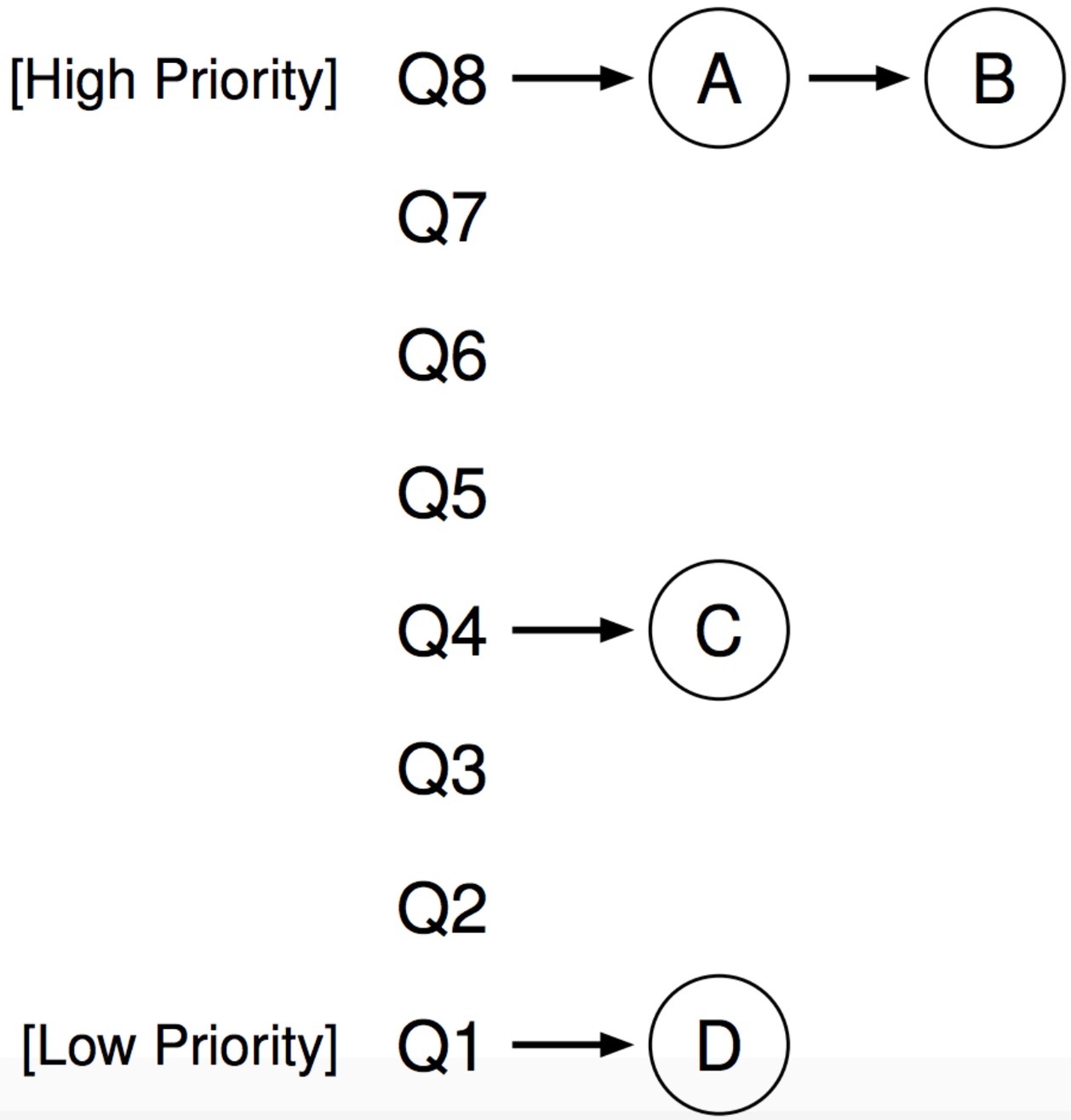
Multi-level Feedback Queue

Fernando José "Corby" Corbató

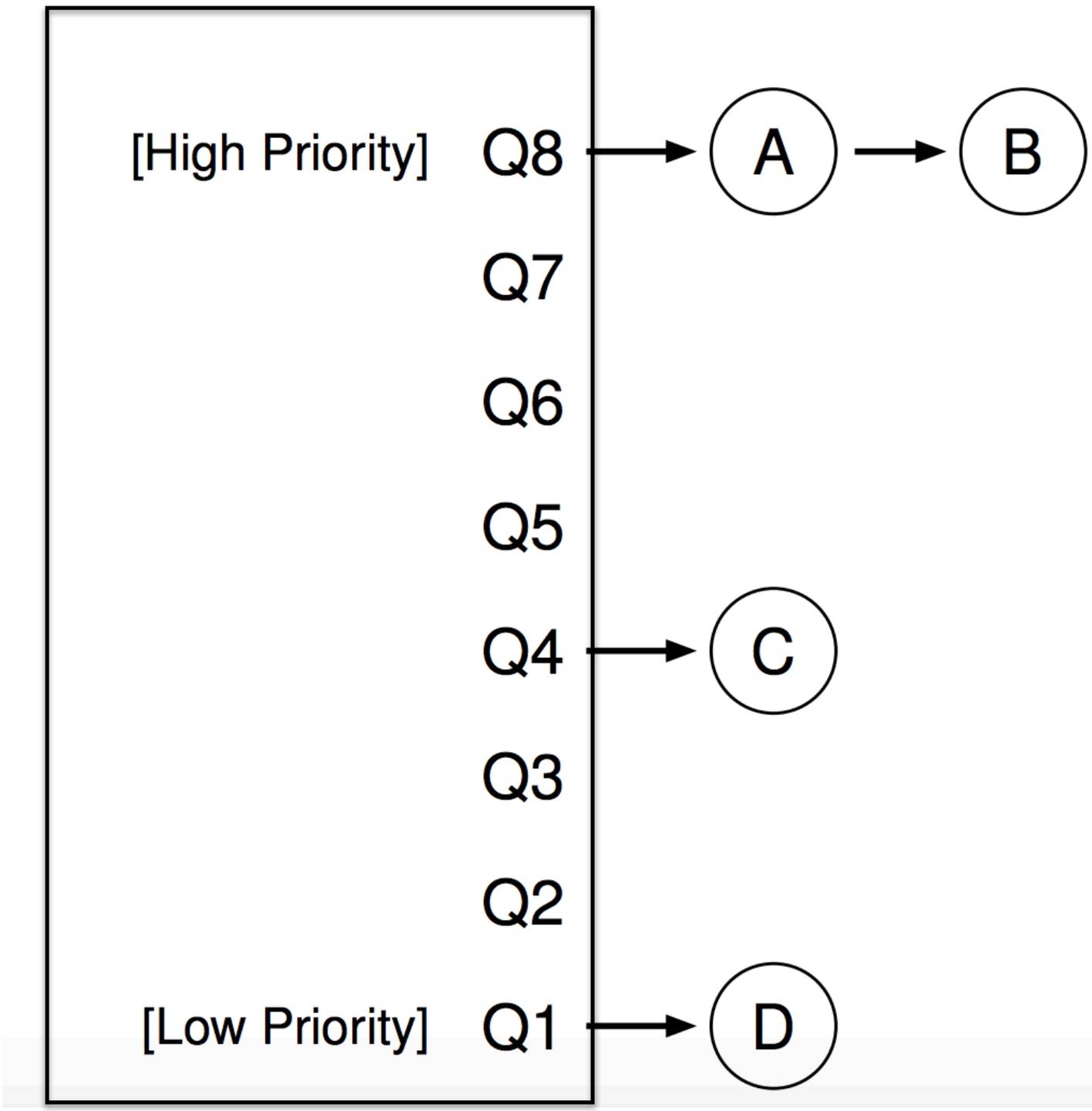
Multi-level Feedback Queue (MLFQ)

1. Optimize Turnaround time - run shorter jobs first
    1. But we don't know "length" of a job
2. Optimize Response time
    1. Round-robin optimises response time, but poor at turnaround time

MLFQ: Rules

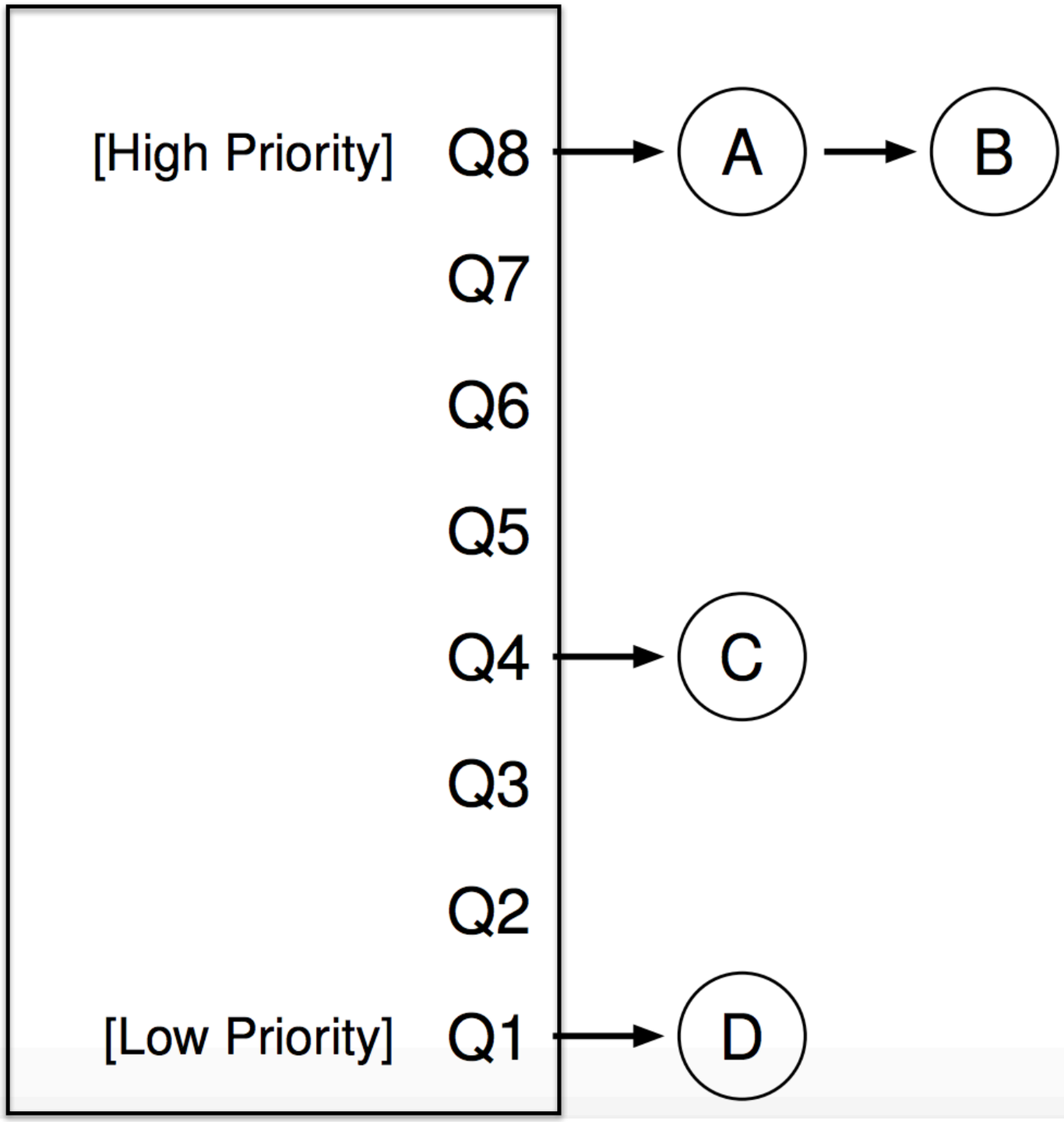


MLFQ: Rules



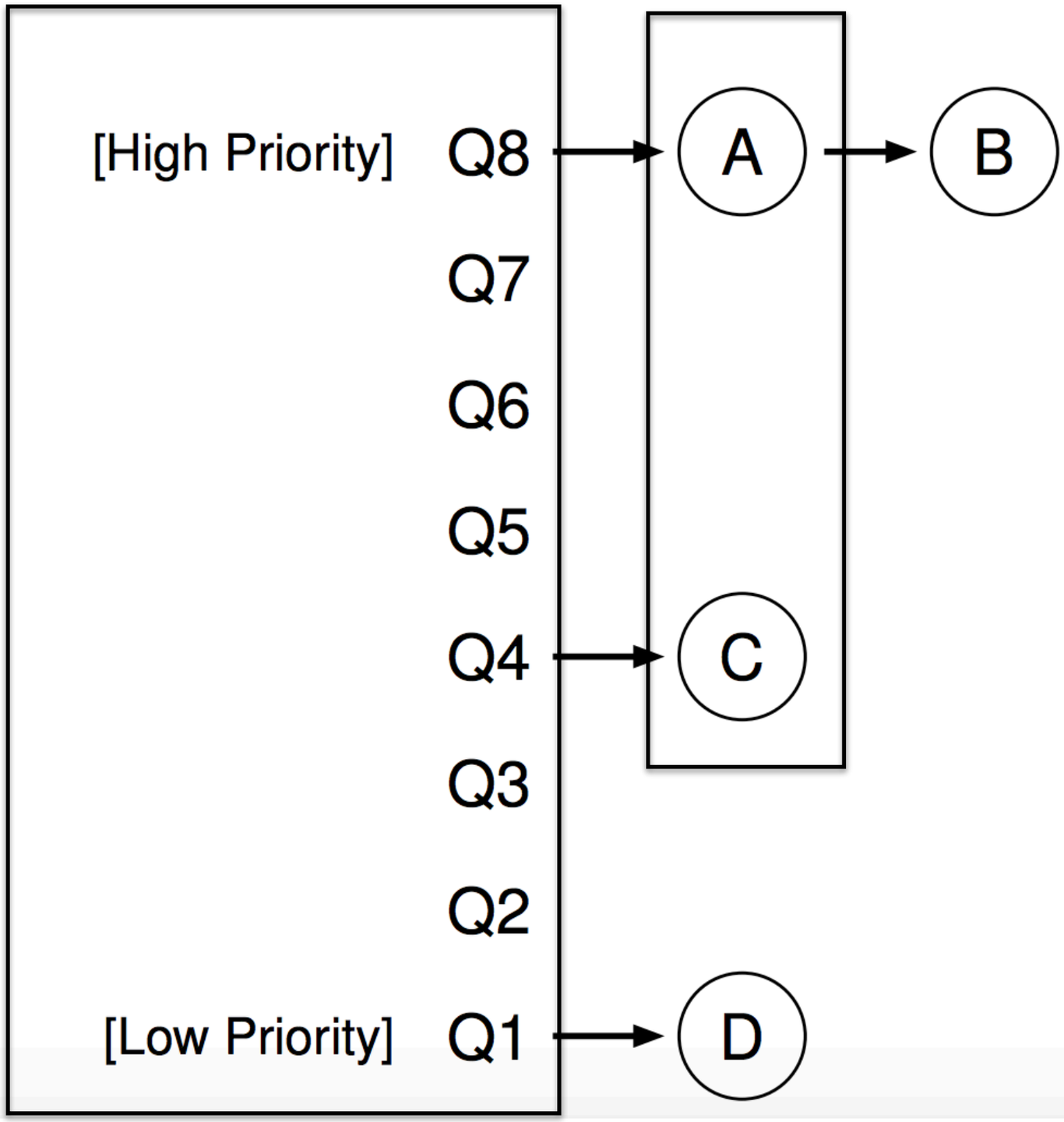
MLFQ: Rules

Distinct queues of different priority



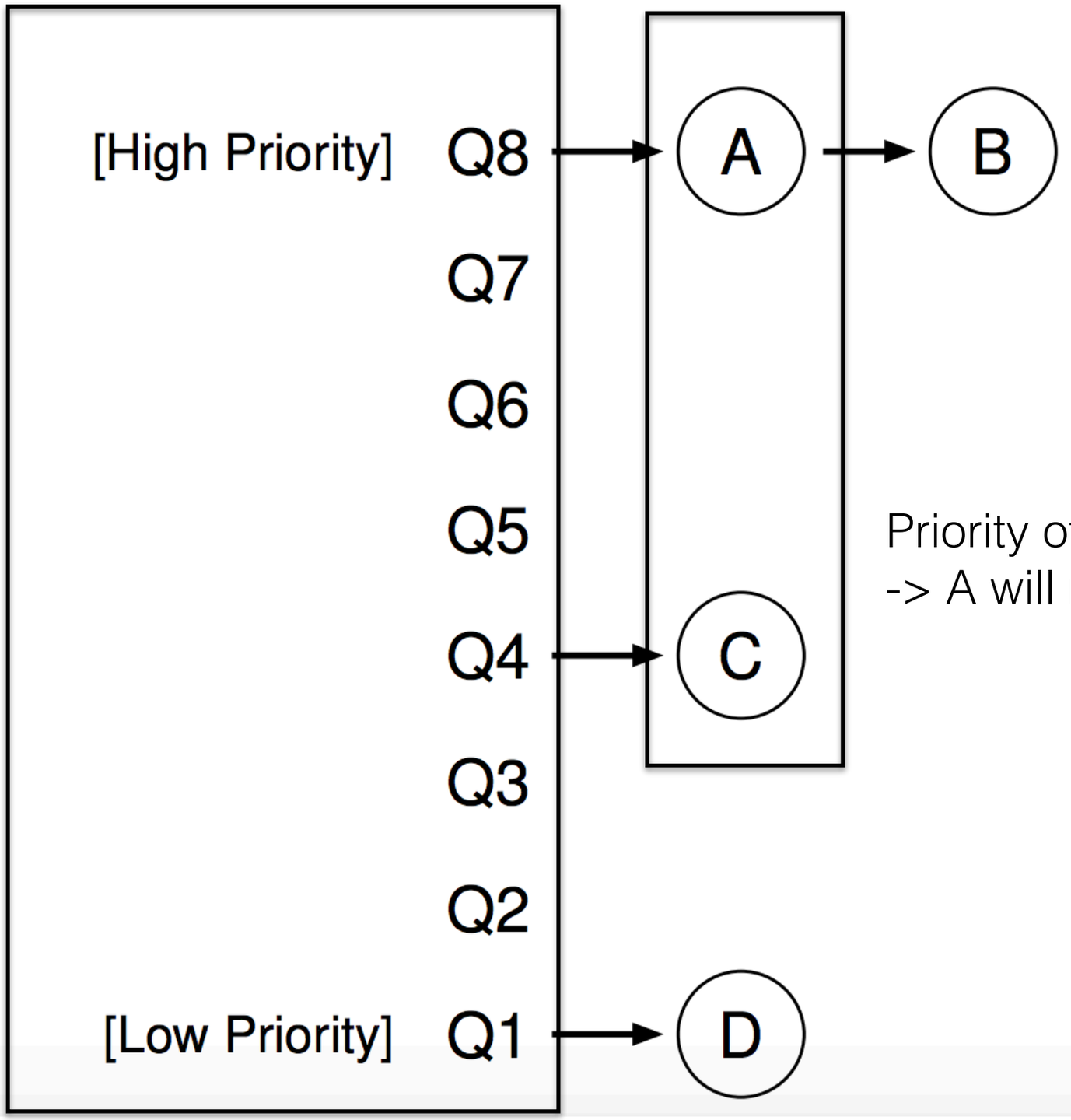
MLFQ: Rules

Distinct queues of different priority



MLFQ: Rules

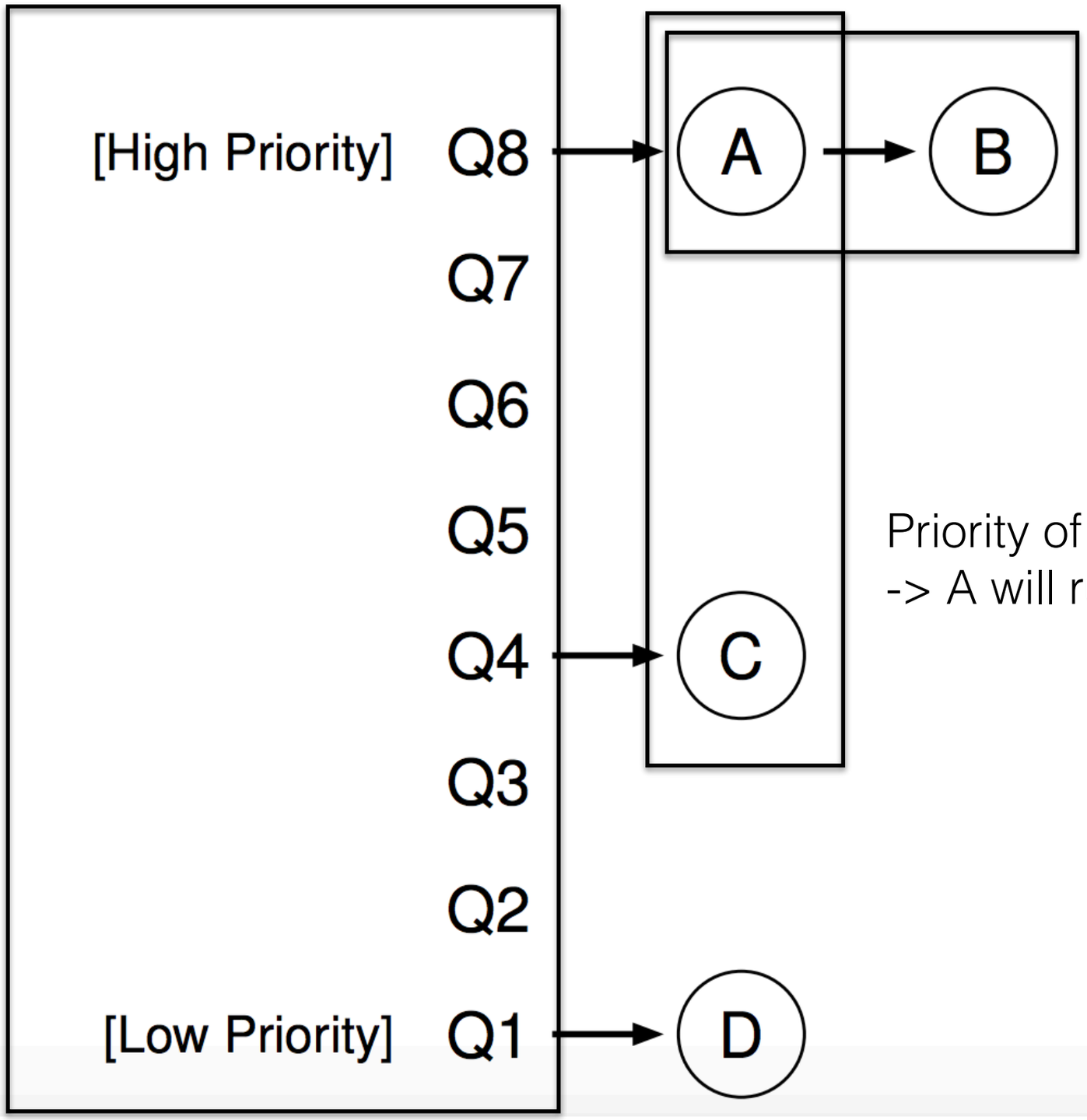
Distinct queues of different priority



Priority of A > Priority of C
-> A will run before C

MLFQ: Rules

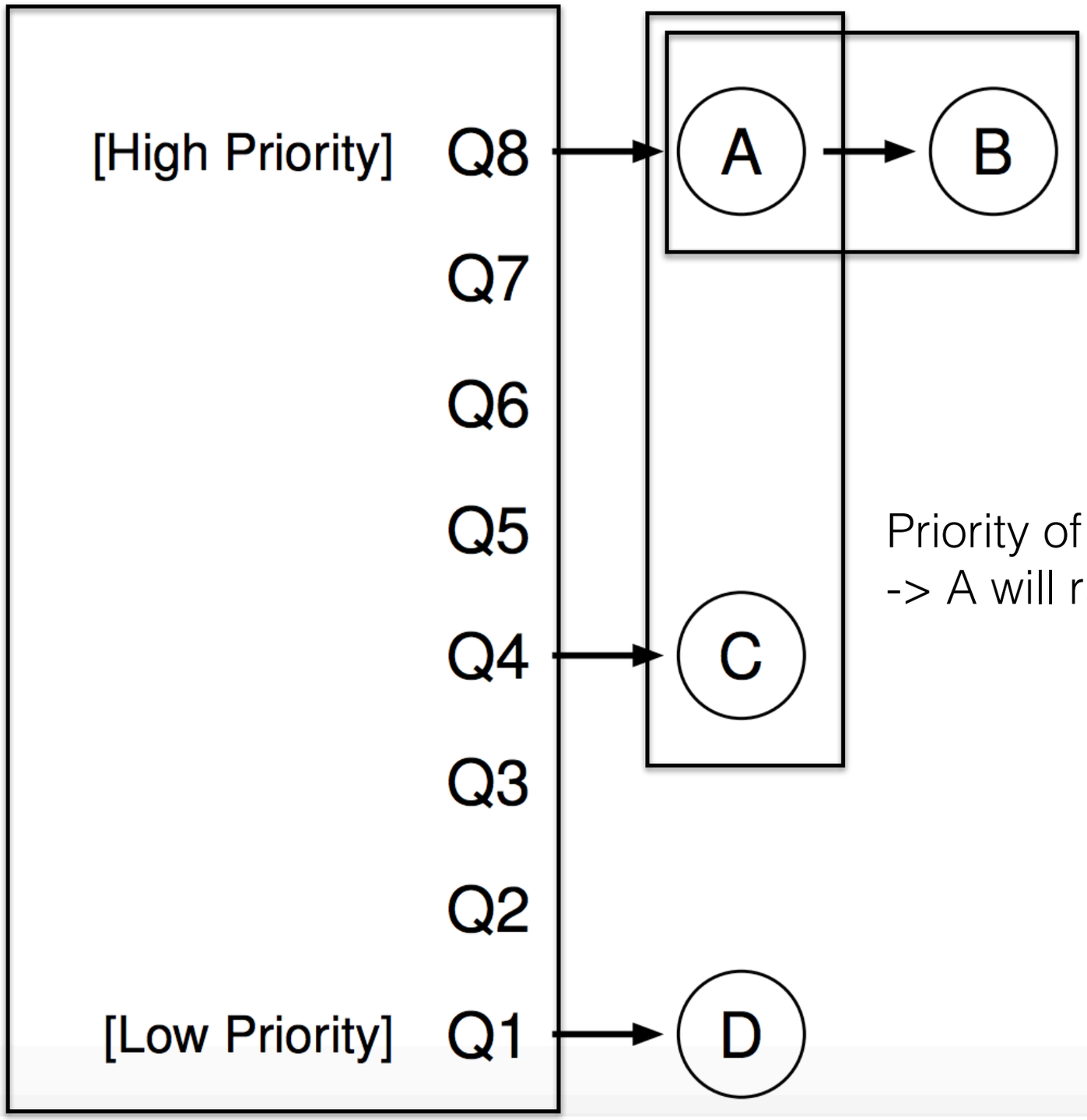
Distinct queues of different priority



Priority of A > Priority of C
-> A will run before C

MLFQ: Rules

Distinct queues of different priority

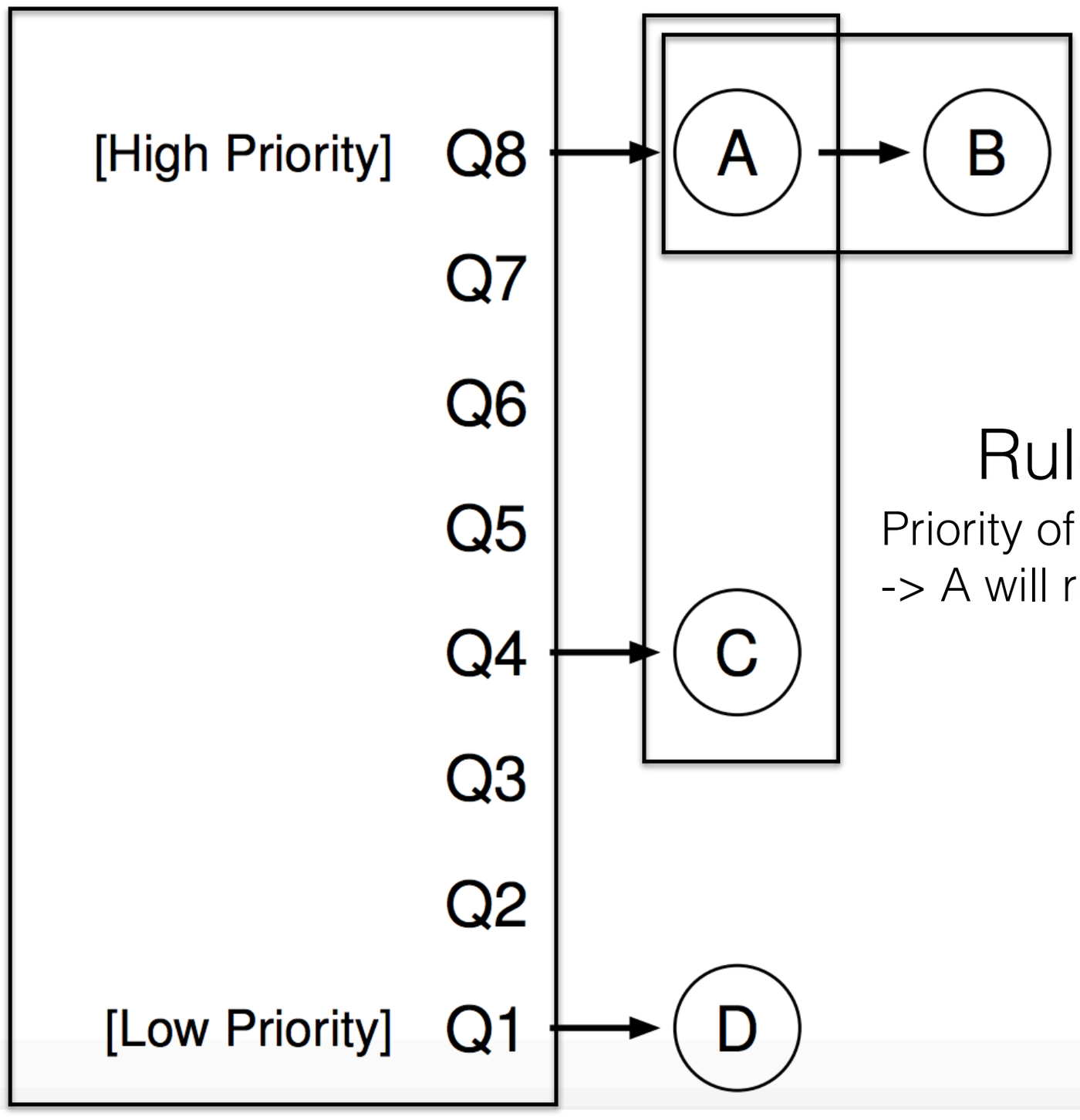


Priority of A = Priority of B -> They are run in round robin fashion

Priority of A > Priority of C
-> A will run before C

MLFQ: Rules

Distinct queues of different priority



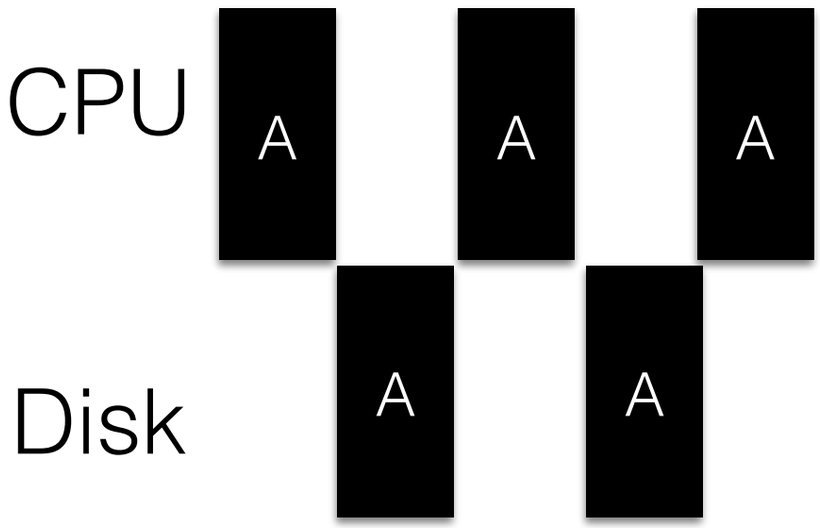
Rule #2

Priority of A = Priority of B -> They are run in round robin fashion

Rule #1

Priority of A > Priority of C
-> A will run before C

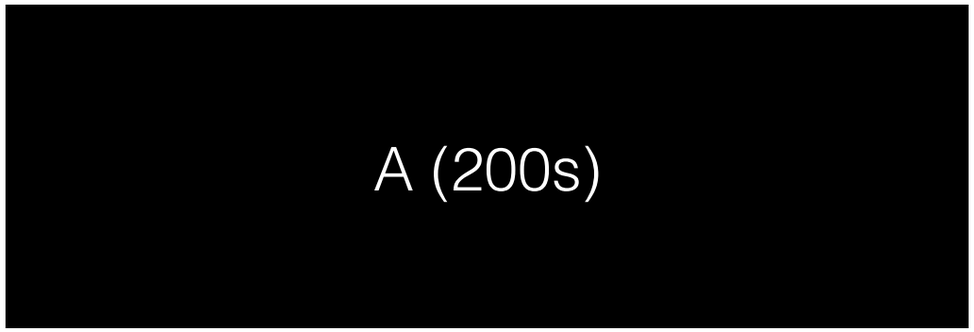
MLFQ Priority Intuition



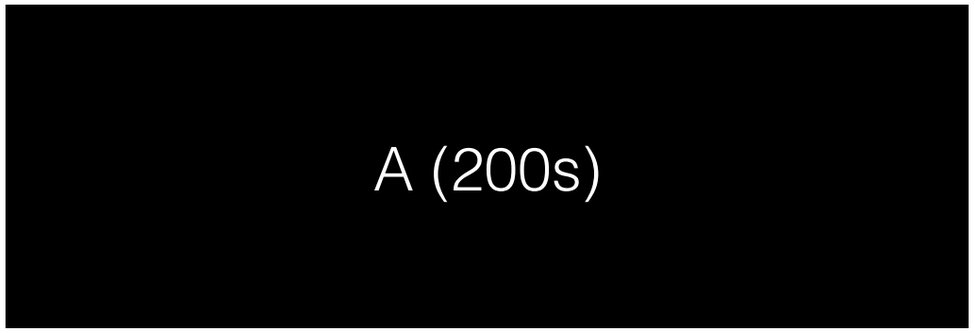
A is interactive ->
Keep it high priority

B is CPU-intensive ->
Lower its priority over time

MLFQ Attempt #1



MLFQ Attempt #1



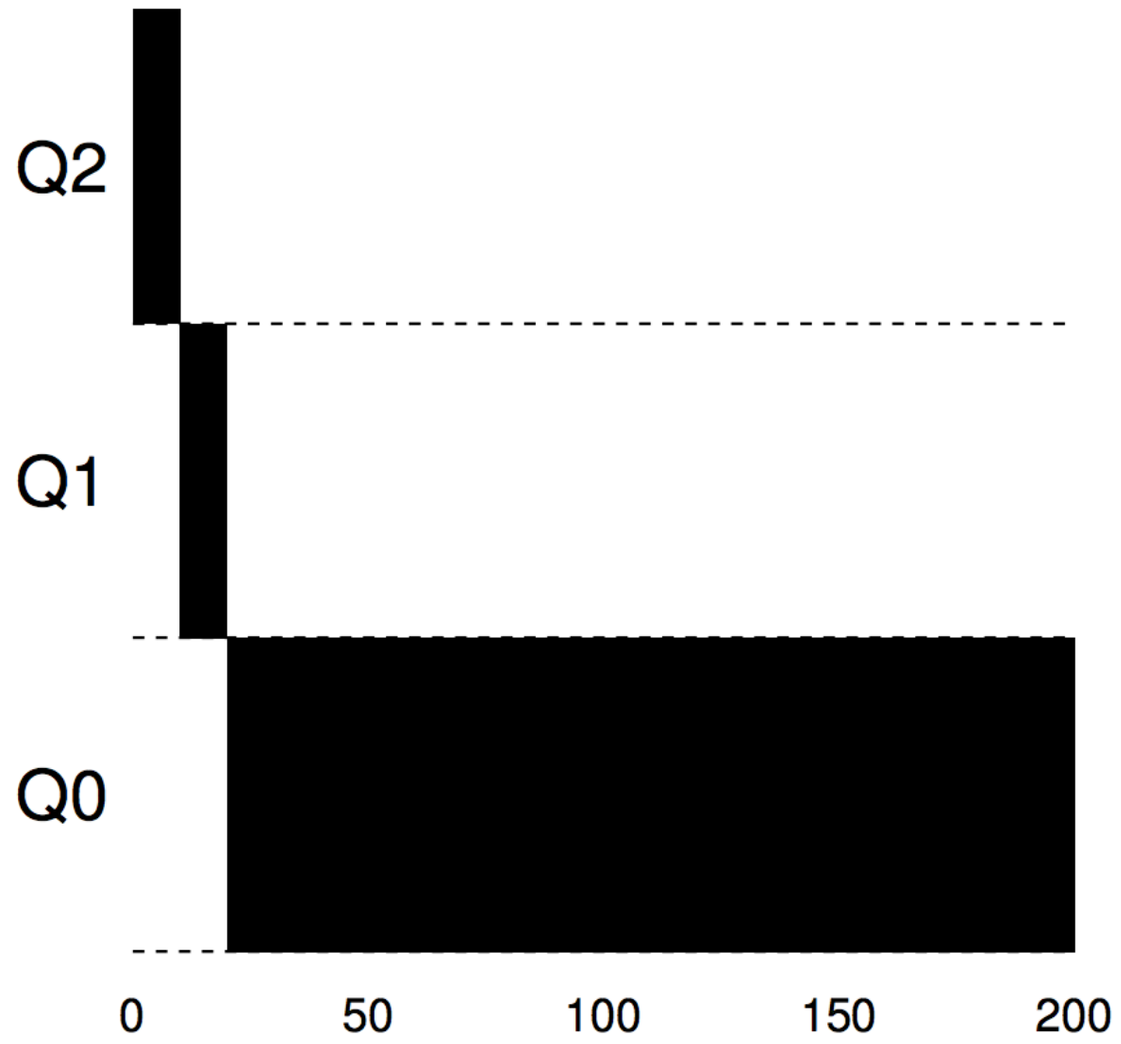
1. 3 queues
2. Time-quantum = 10s

MLFQ Attempt #1

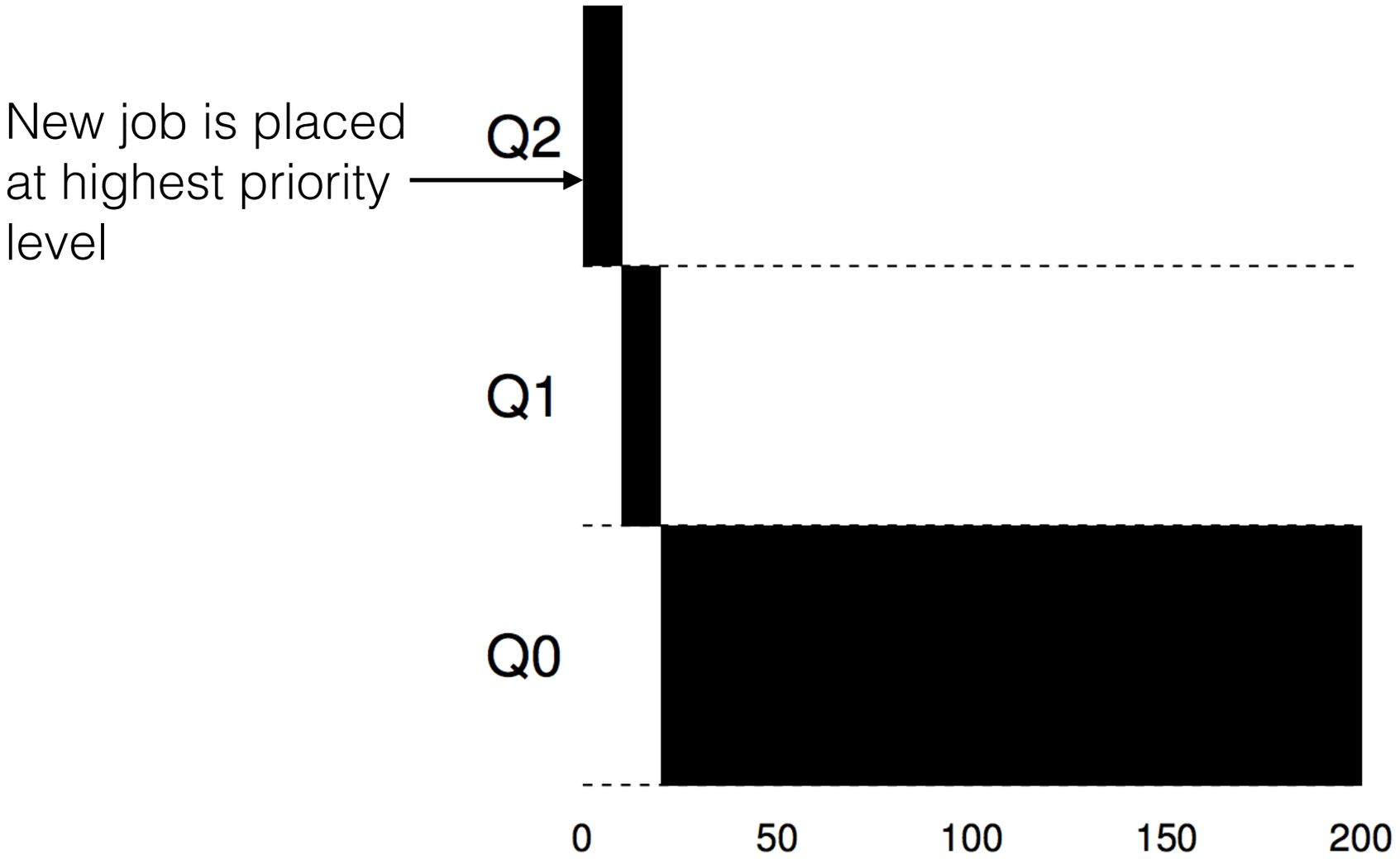
1. 3 queues
2. Time-quantum = 10s

MLFQ Attempt #1

MLFQ Attempt #1

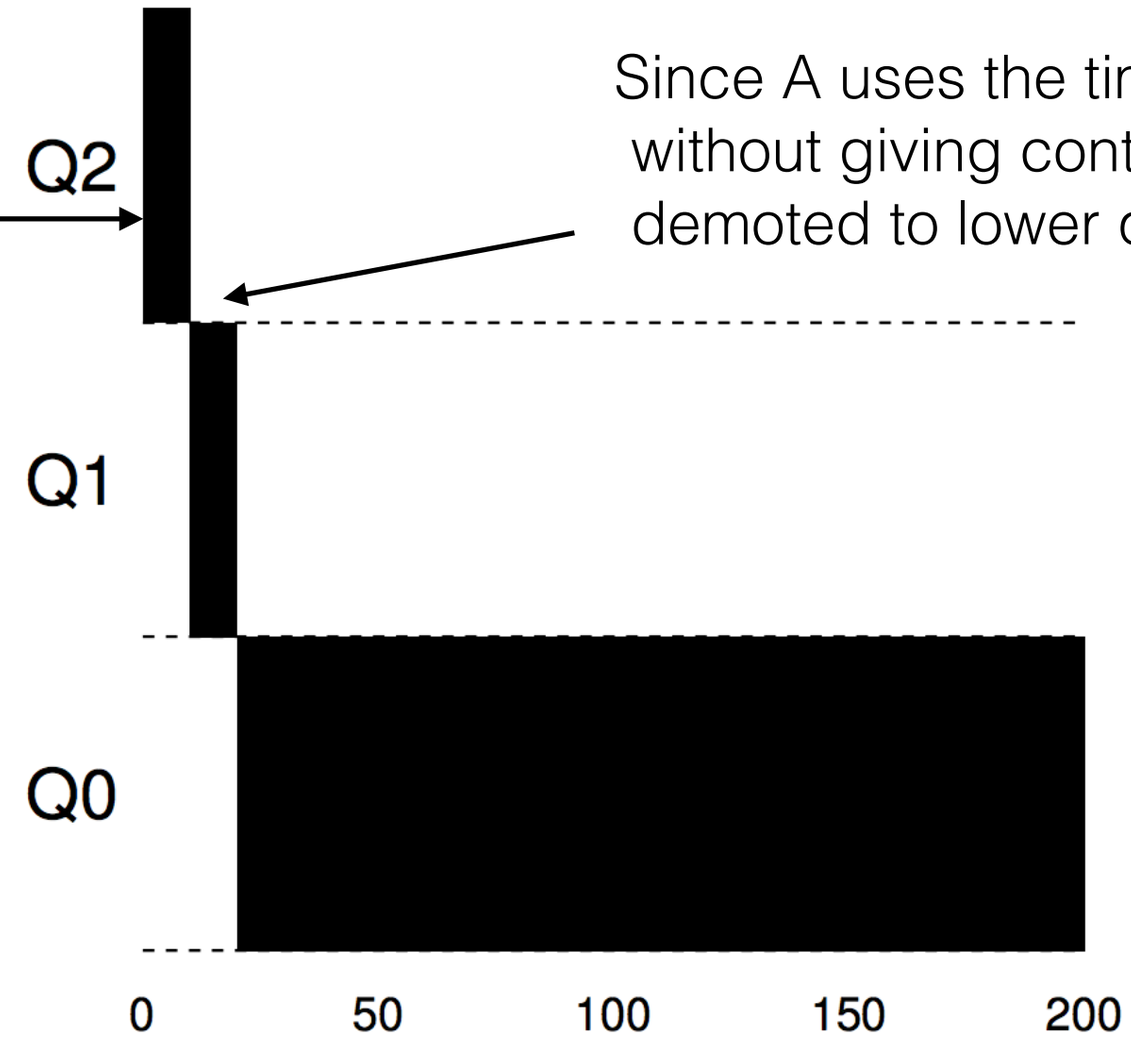


MLFQ Attempt #1



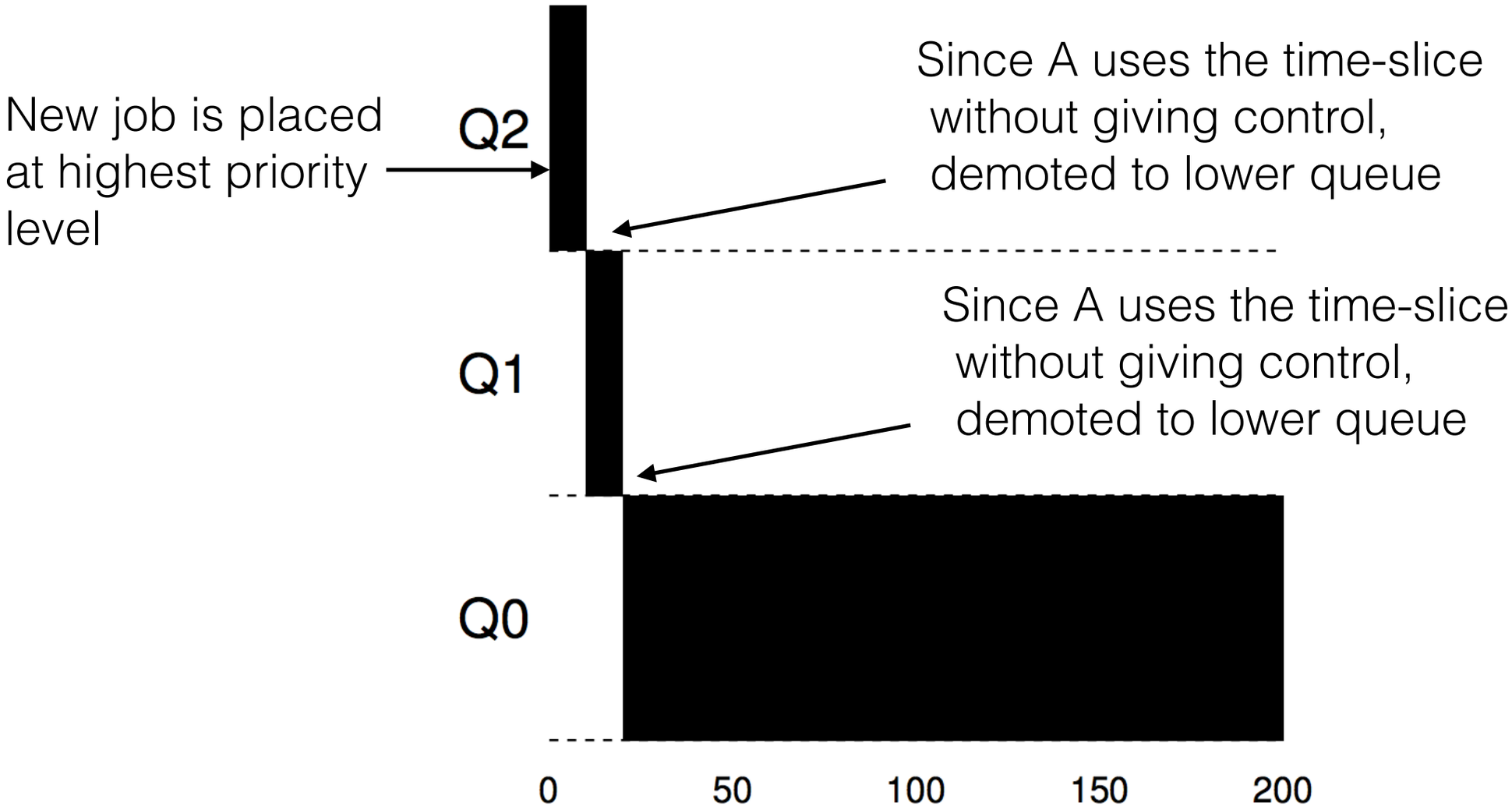
MLFQ Attempt #1

New job is placed at highest priority level



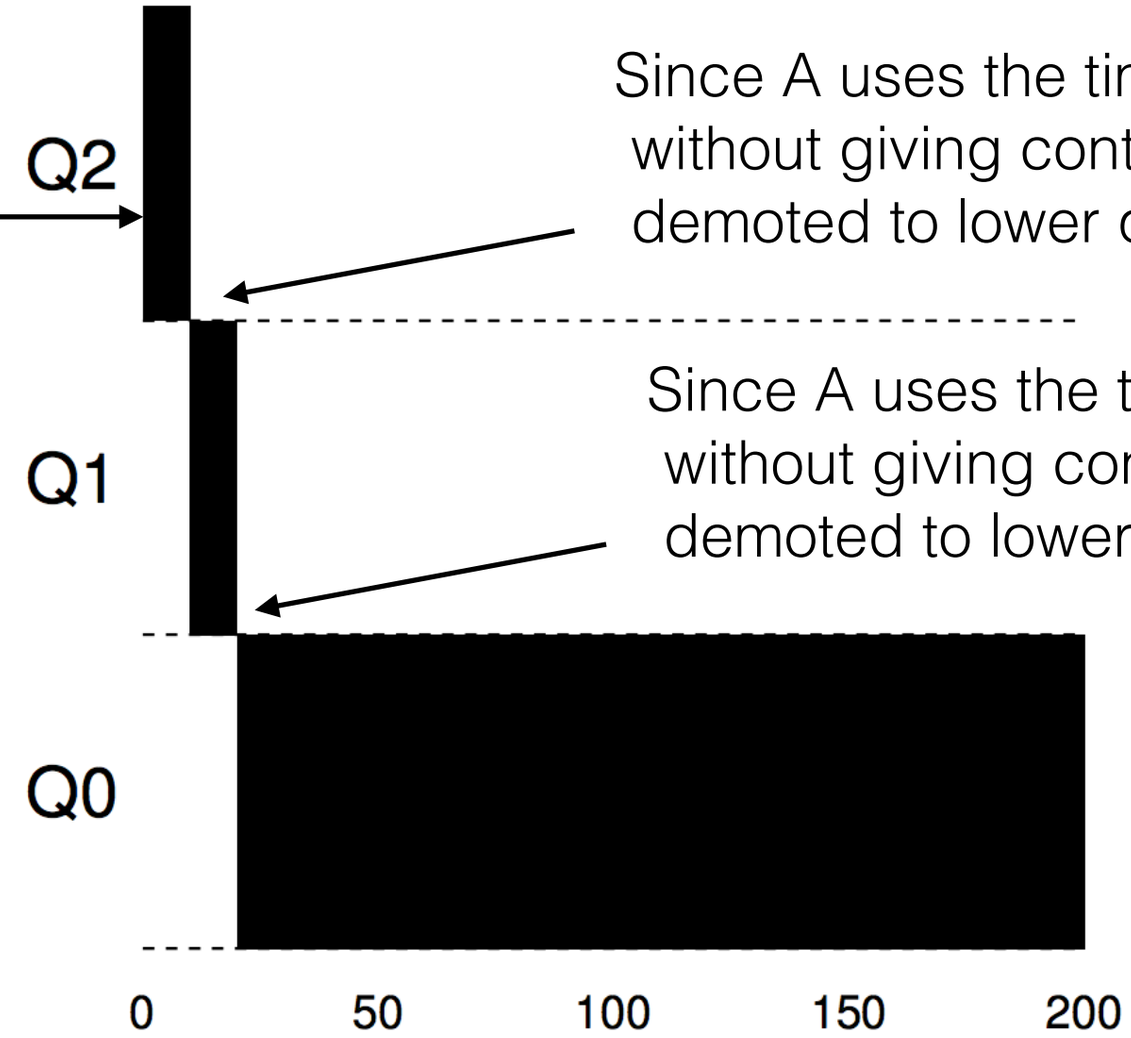
Since A uses the time-slice without giving control, demoted to lower queue

MLFQ Attempt #1



MLFQ Attempt #1

New job is placed at highest priority level



Since A uses the time-slice without giving control, demoted to lower queue

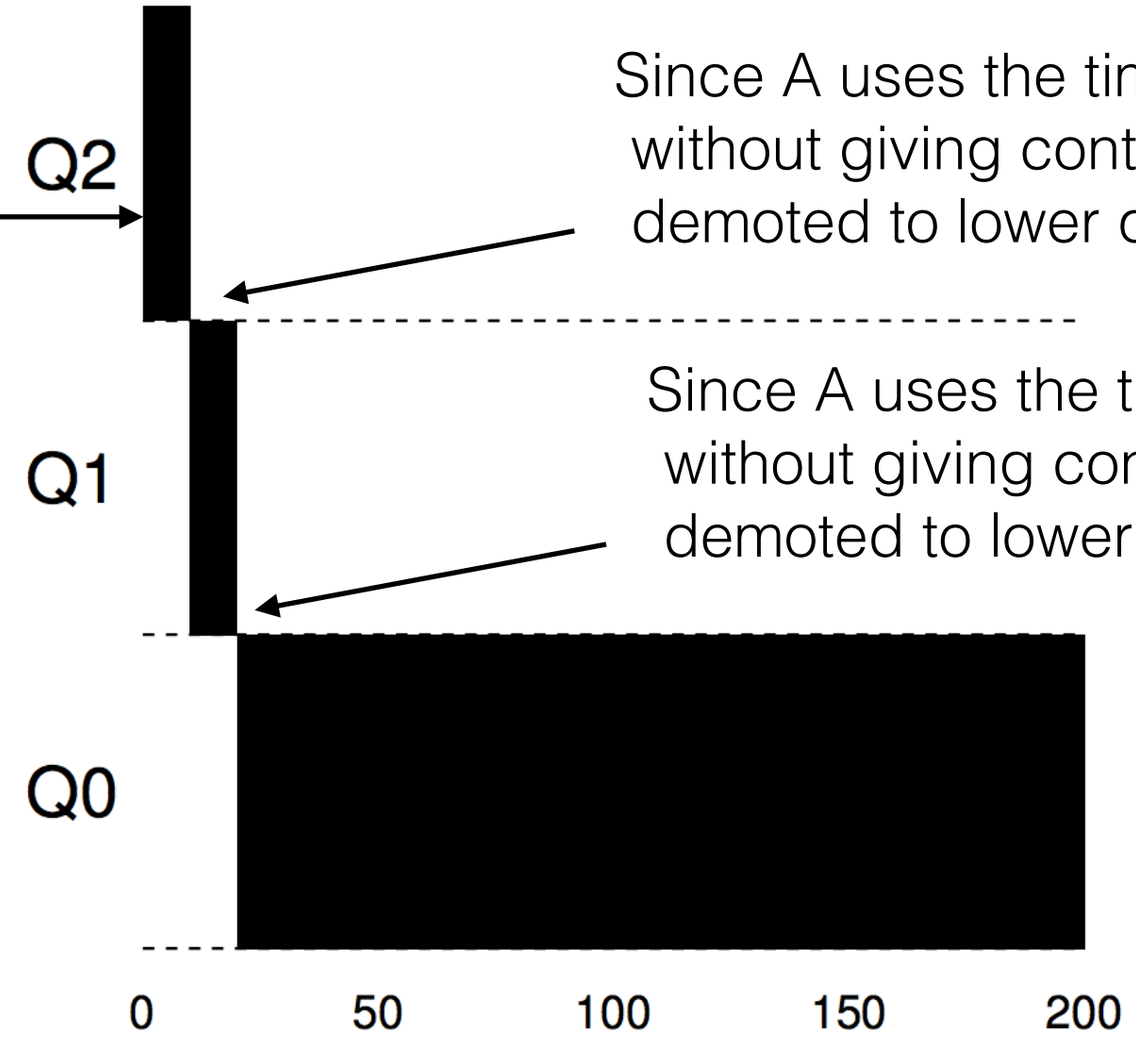
Since A uses the time-slice without giving control, demoted to lower queue

Rule 4a

MLFQ Attempt #1

Rule 3

New job is placed at highest priority level

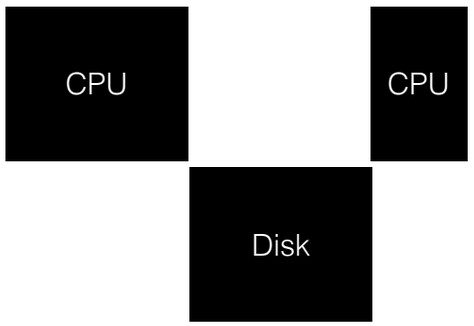


Since A uses the time-slice without giving control, demoted to lower queue

Since A uses the time-slice without giving control, demoted to lower queue

Rule 4a

MLFQ Attempt #1

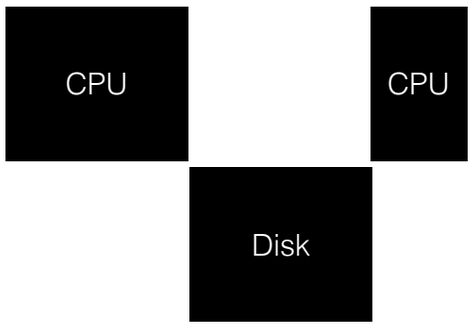


Q2

Q1

Q0

MLFQ Attempt #1

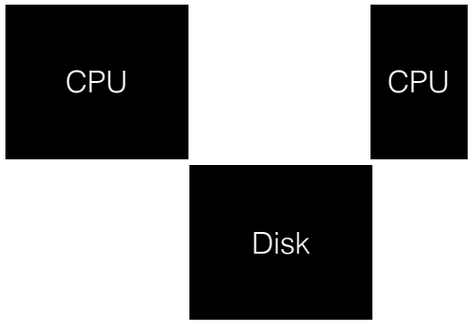


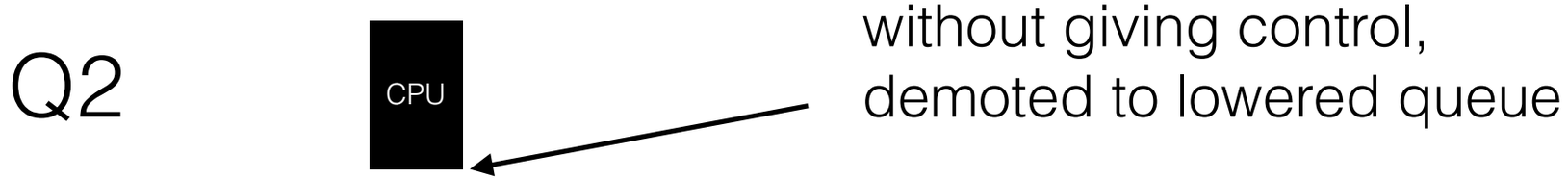
Q2

Q1

Q0

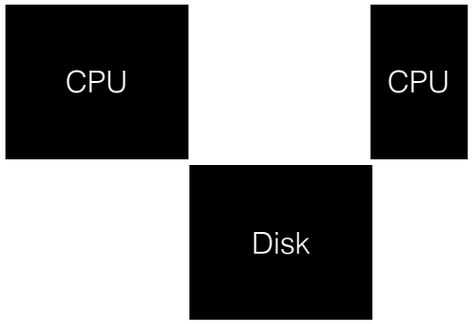
MLFQ Attempt #1

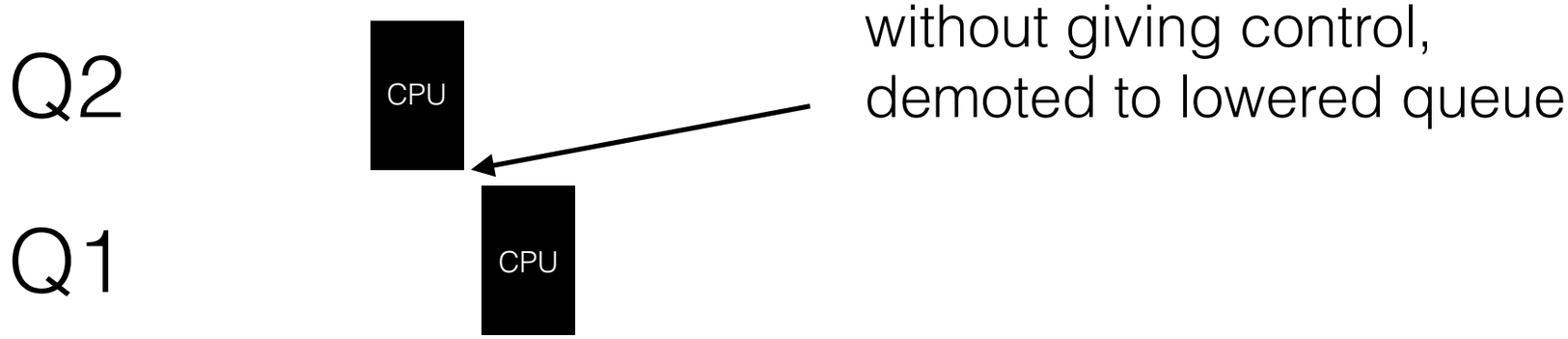




Since A uses the time-slice without giving control, demoted to lowered queue

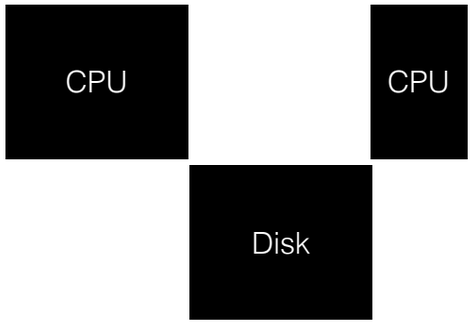
MLFQ Attempt #1

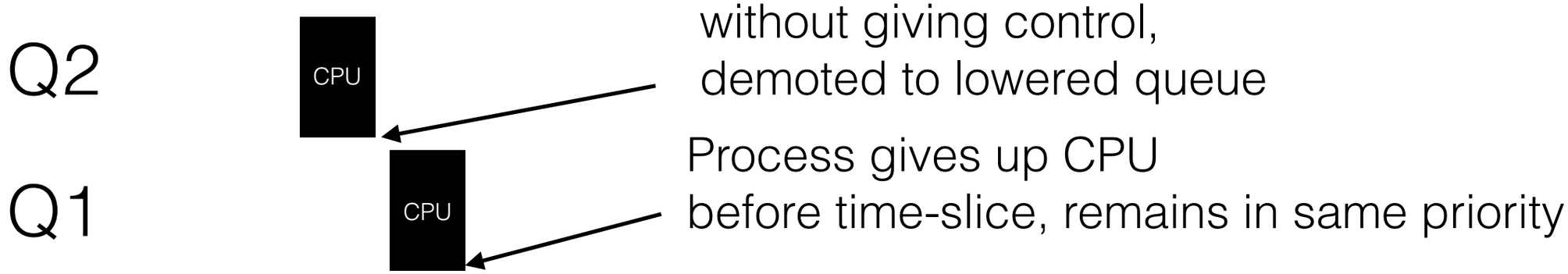




Since A uses the time-slice without giving control, demoted to lowered queue

MLFQ Attempt #1

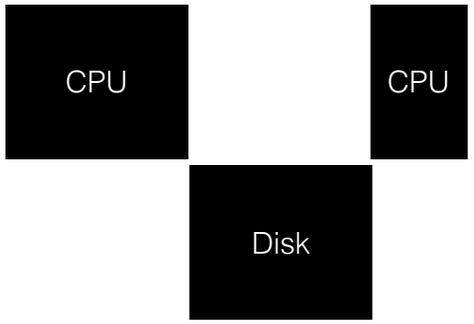


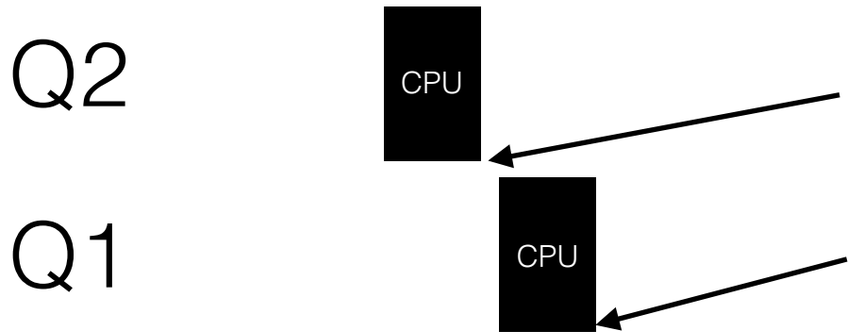


Since A uses the time-slice without giving control, demoted to lowered queue

Process gives up CPU before time-slice, remains in same priority

MLFQ Attempt #1



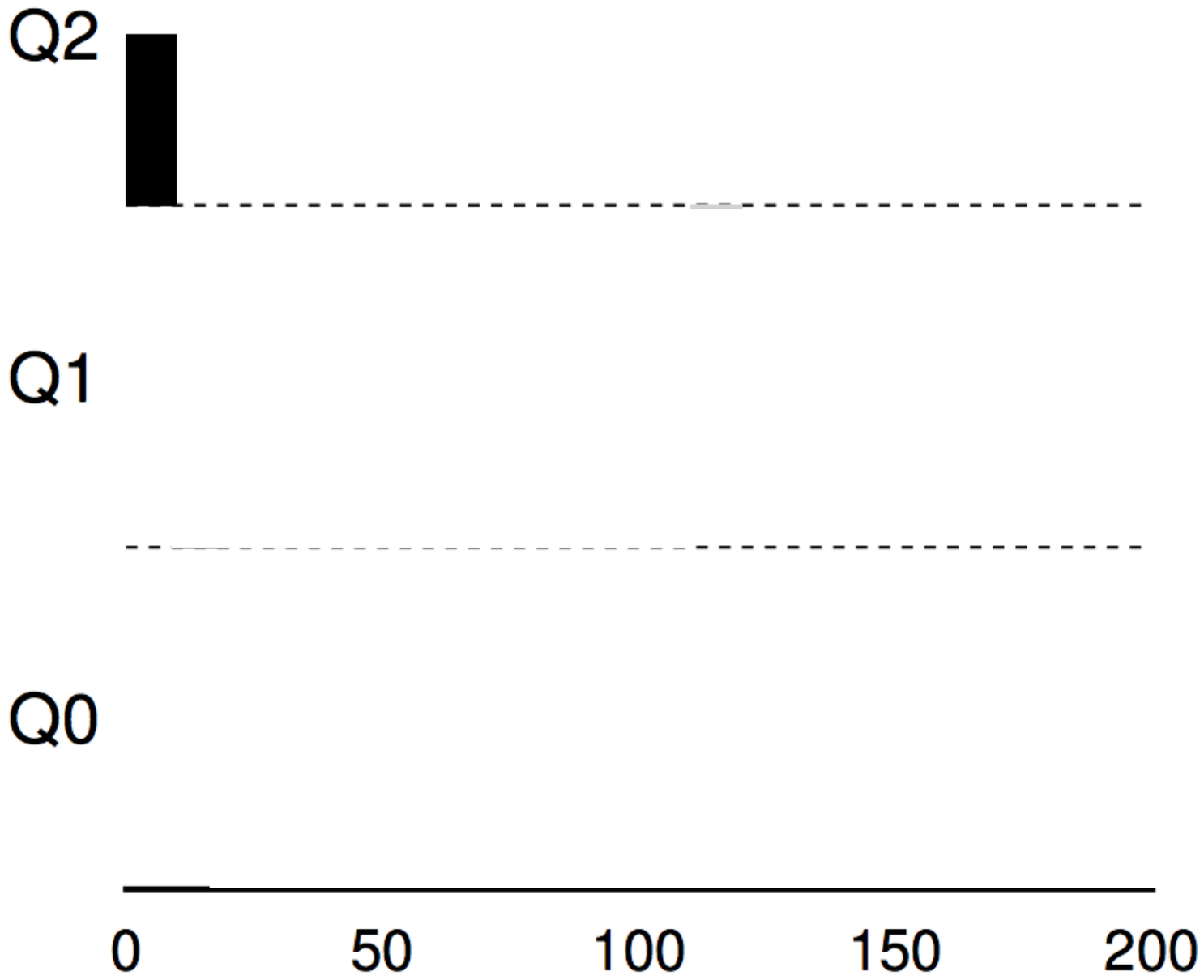


Since A uses the time-slice without giving control, demoted to lowered queue

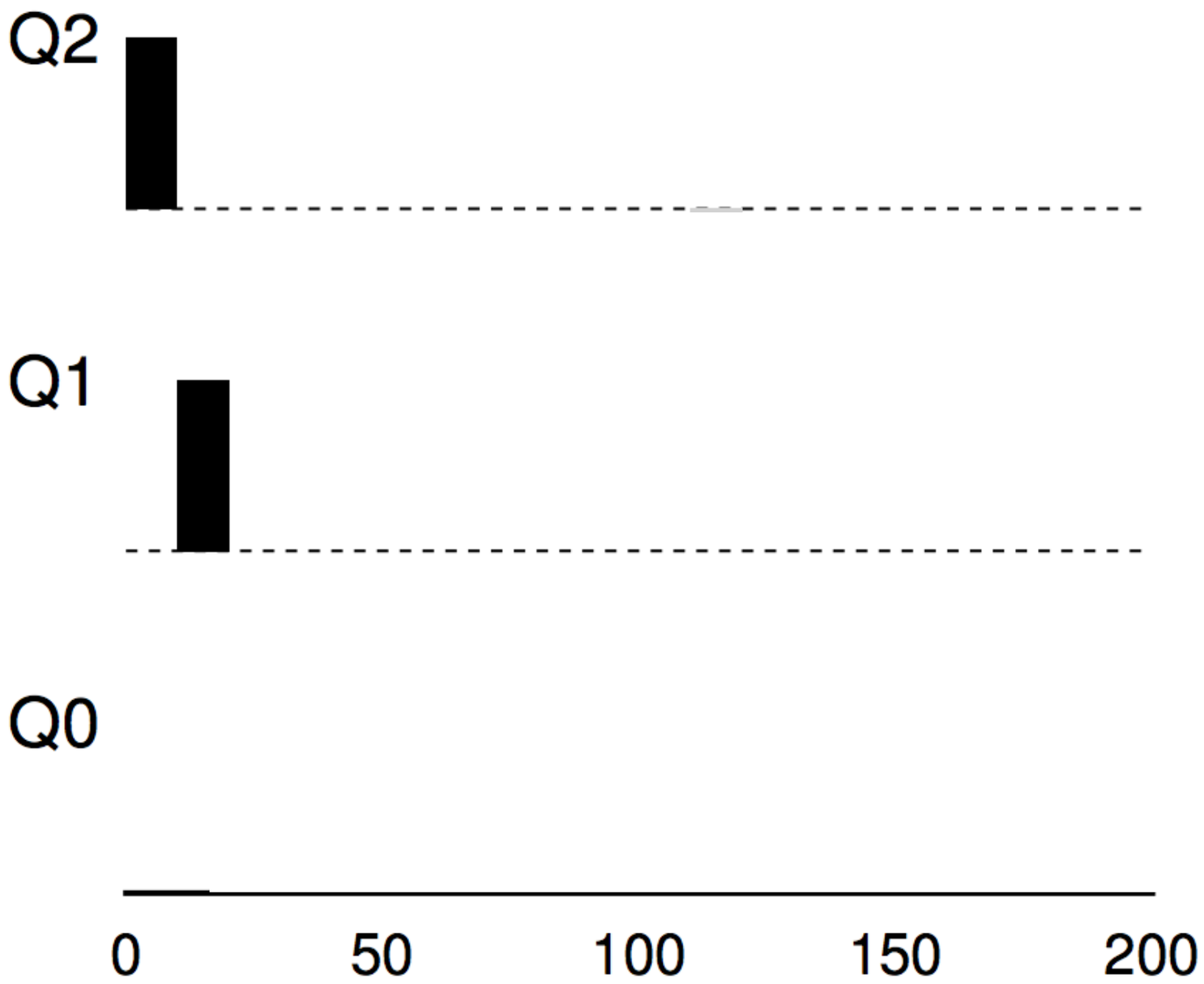
Process gives up CPU before time-slice, remains in same priority

Rule 4b

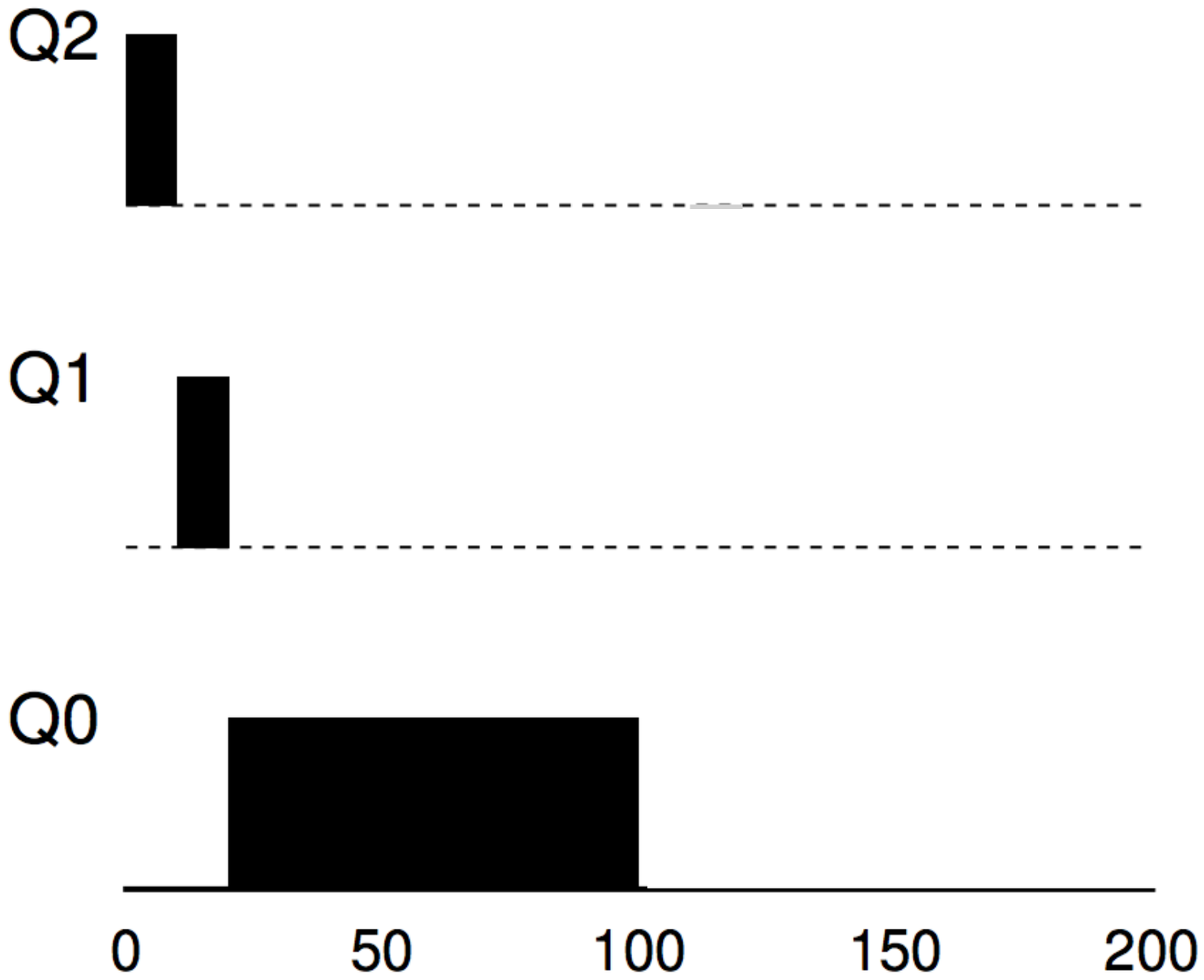
MLFQ Attempt #1 (MLFQ approximates SJF)



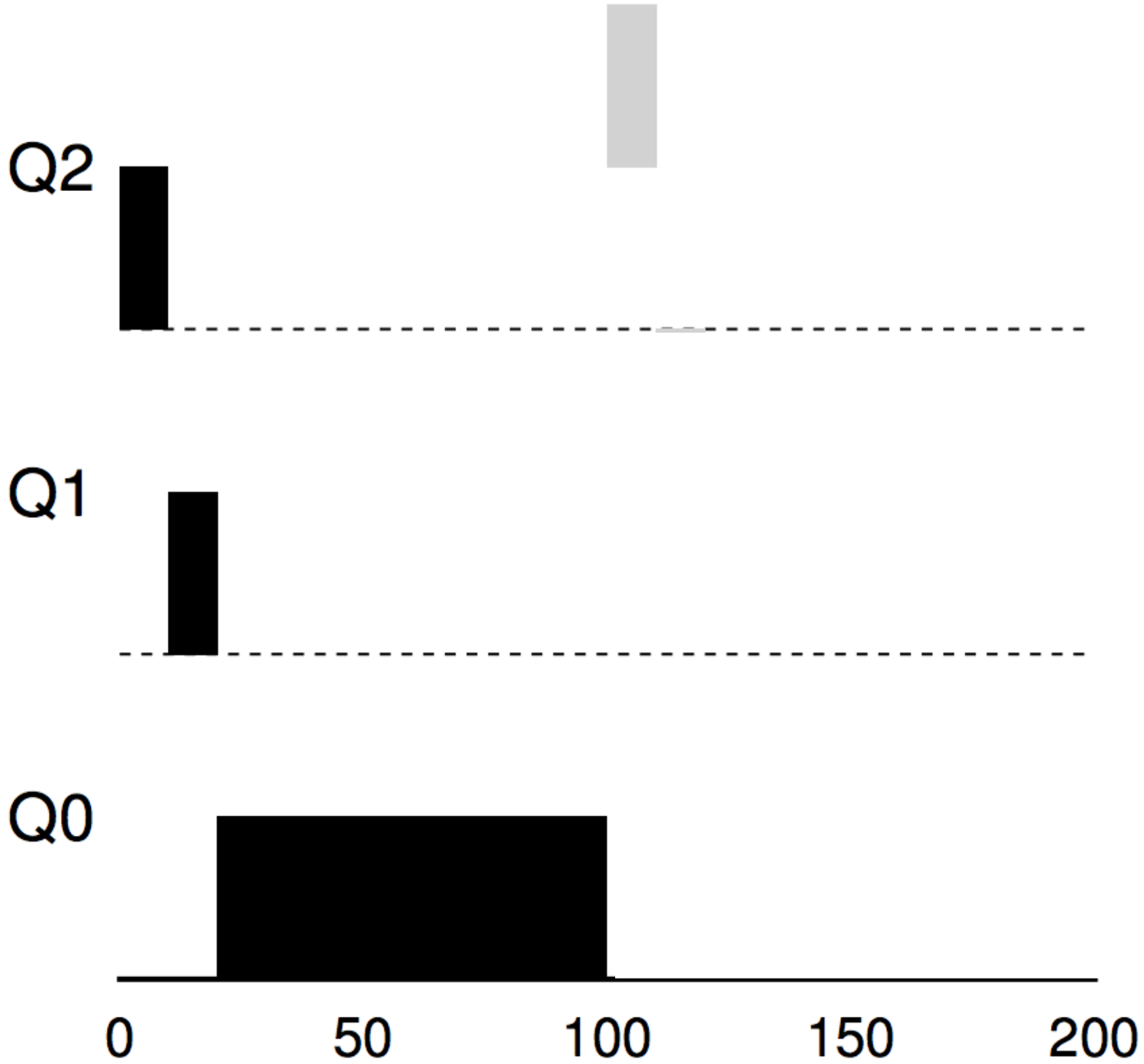
MLFQ Attempt #1 (MLFQ approximates SJF)



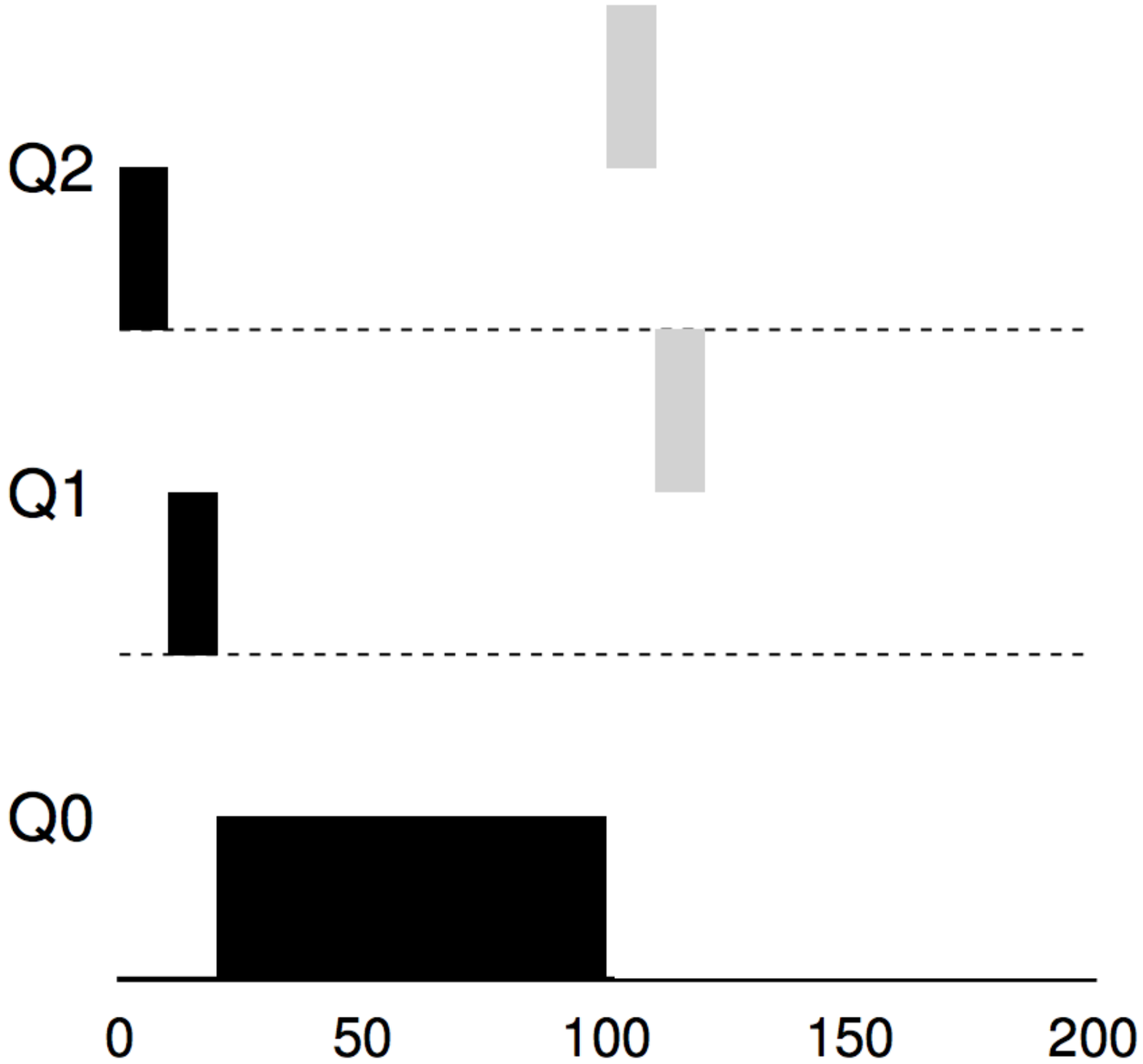
MLFQ Attempt #1 (MLFQ approximates SJF)



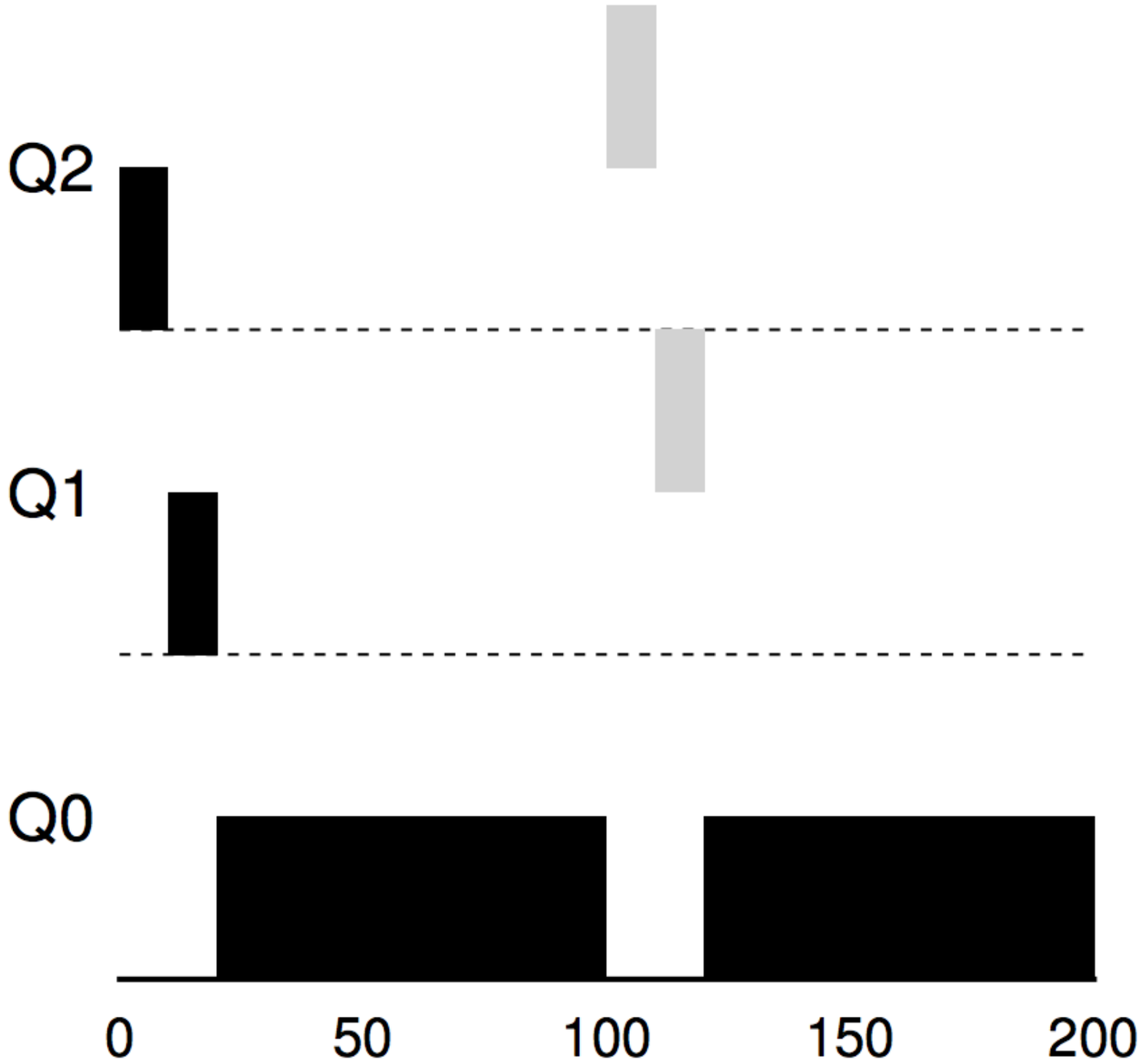
MLFQ Attempt #1 (MLFQ approximates SJF)



MLFQ Attempt #1 (MLFQ approximates SJF)

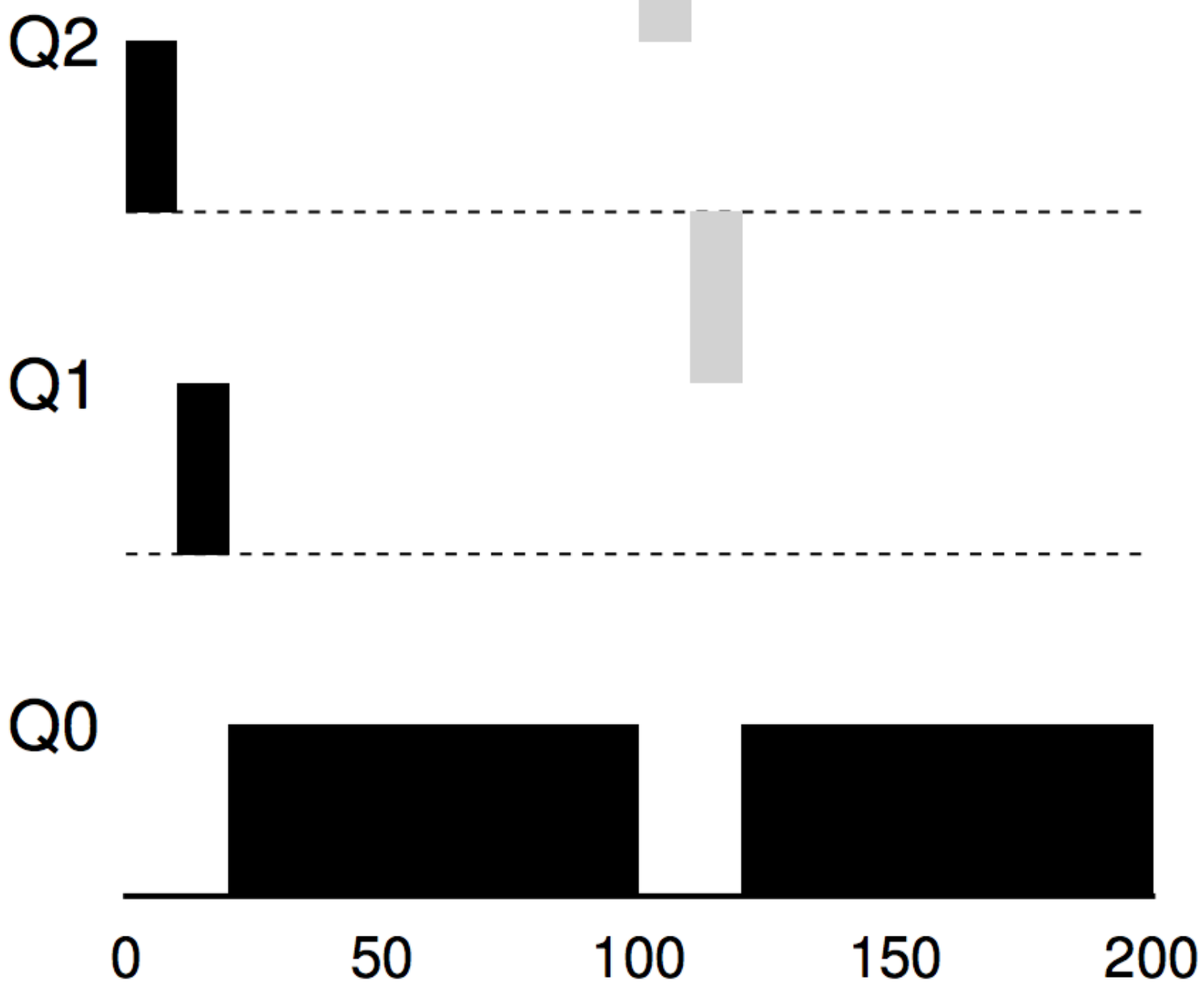


MLFQ Attempt #1 (MLFQ approximates SJF)



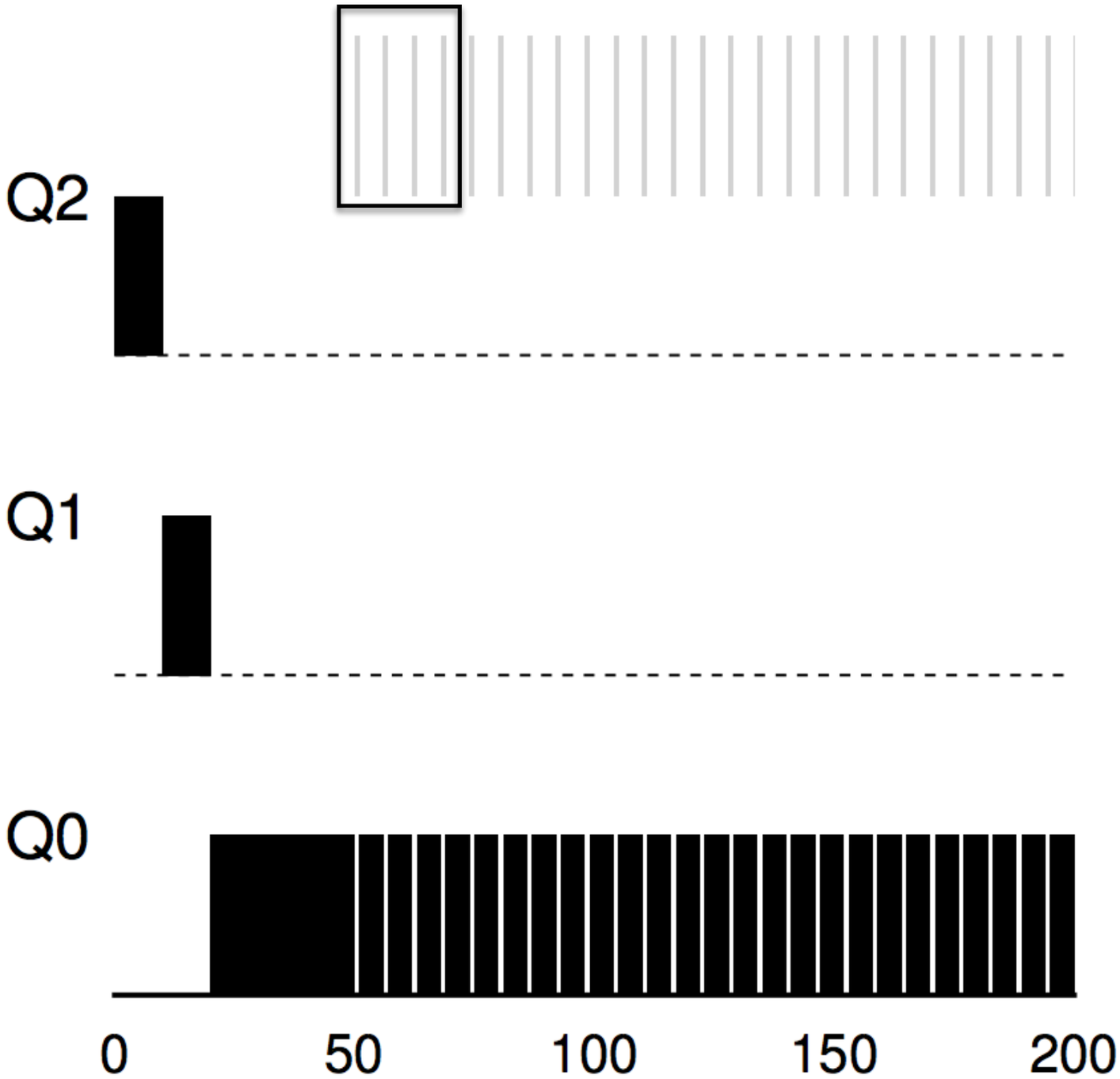
MLFQ Attempt #1 (MLFQ approximates SJF)

- Assume job is short and give highest priority
- If its short, completes soon, else, demoted



MLFQ Attempt #1 : IO + CPU intensive

IO heavy jobs, relinquish control soon, remain at same priority



MLFQ Attempt #1: Shortcomings

MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones

MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :

MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :
    1. Time slice = x

MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :
    1. Time slice = x
    2. Run CPU for 0.99*x

MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :
    1. Time slice = x
    2. Run CPU for 0.99*x
    3. Request I/O

MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :
    1. Time slice = x
    2. Run CPU for 0.99*x
    3. Request I/O
    4. Remain in same priority

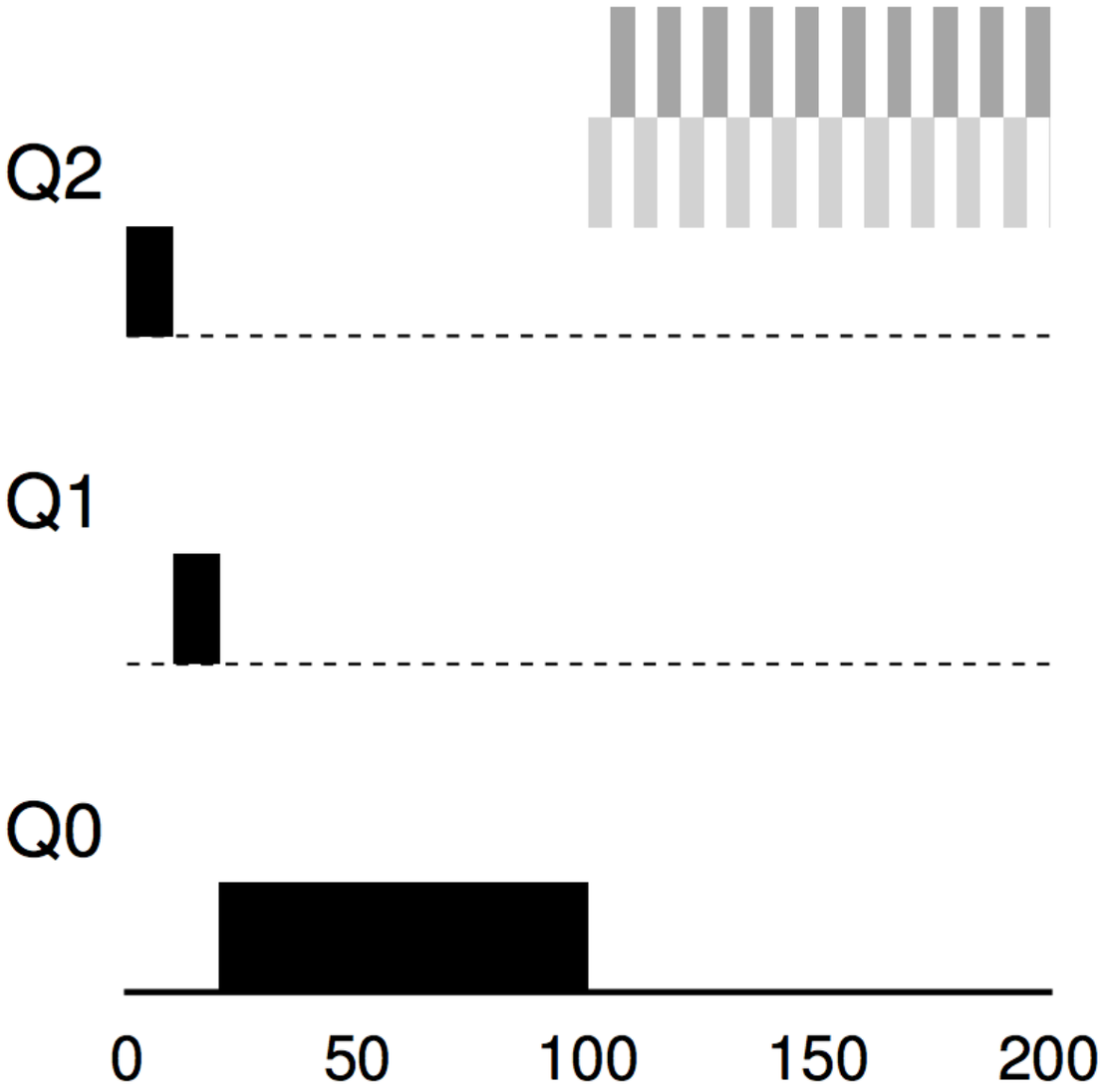
MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :
    1. Time slice = x
    2. Run CPU for 0.99*x
    3. Request I/O
    4. Remain in same priority
    5. Goto 2

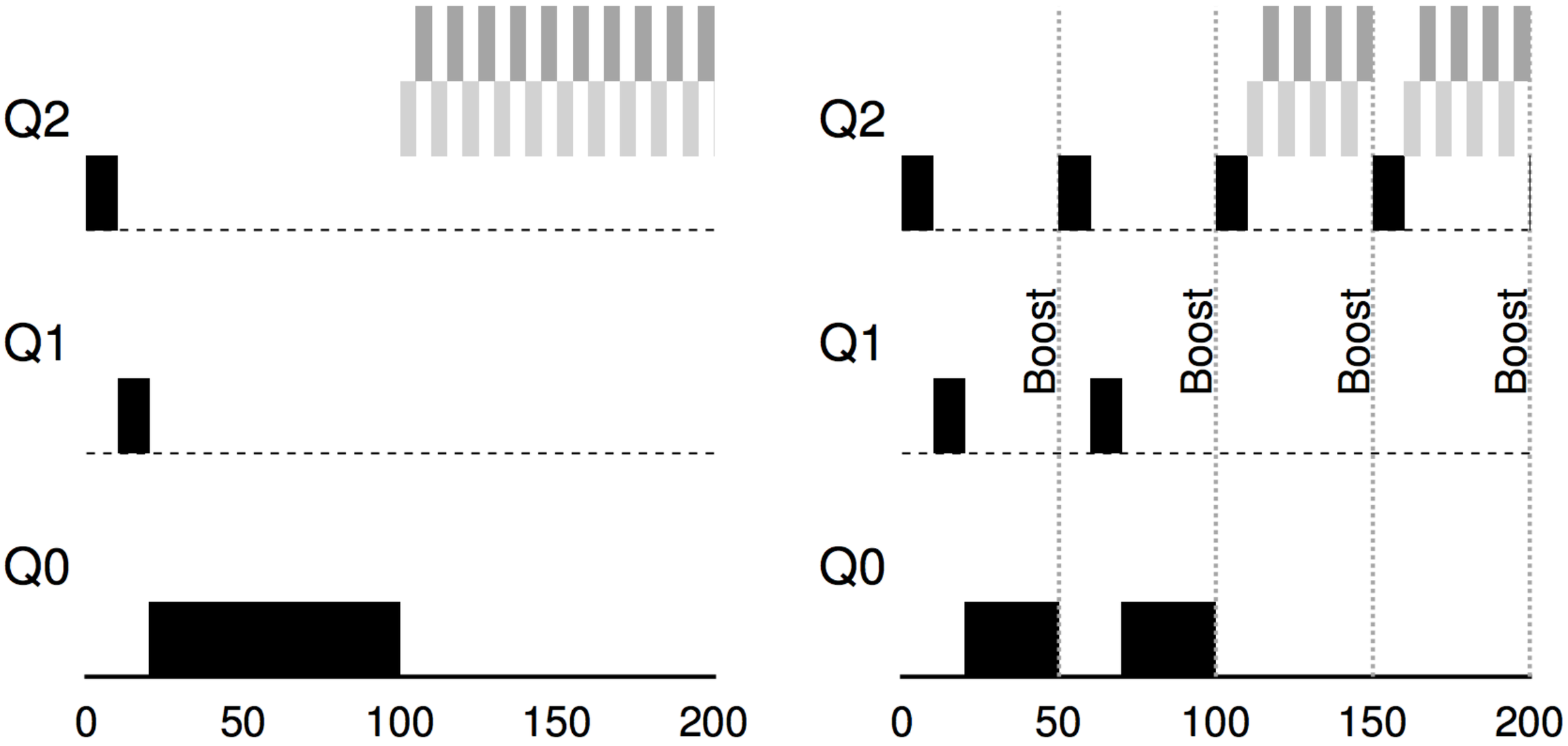
MLFQ Attempt #1: Shortcomings

1. Starvation : Too many I/O jobs will eat up the CPU; no execution for CPU intensive ones
2. Scheduler gaming :
    1. Time slice = x
    2. Run CPU for 0.99*x
    3. Request I/O
    4. Remain in same priority
    5. Goto 2
3. Behaviour change: CPU intensive went to lowest priority, but has loads of I/O after say y time units

MLFQ Attempt 2: Priority Boost

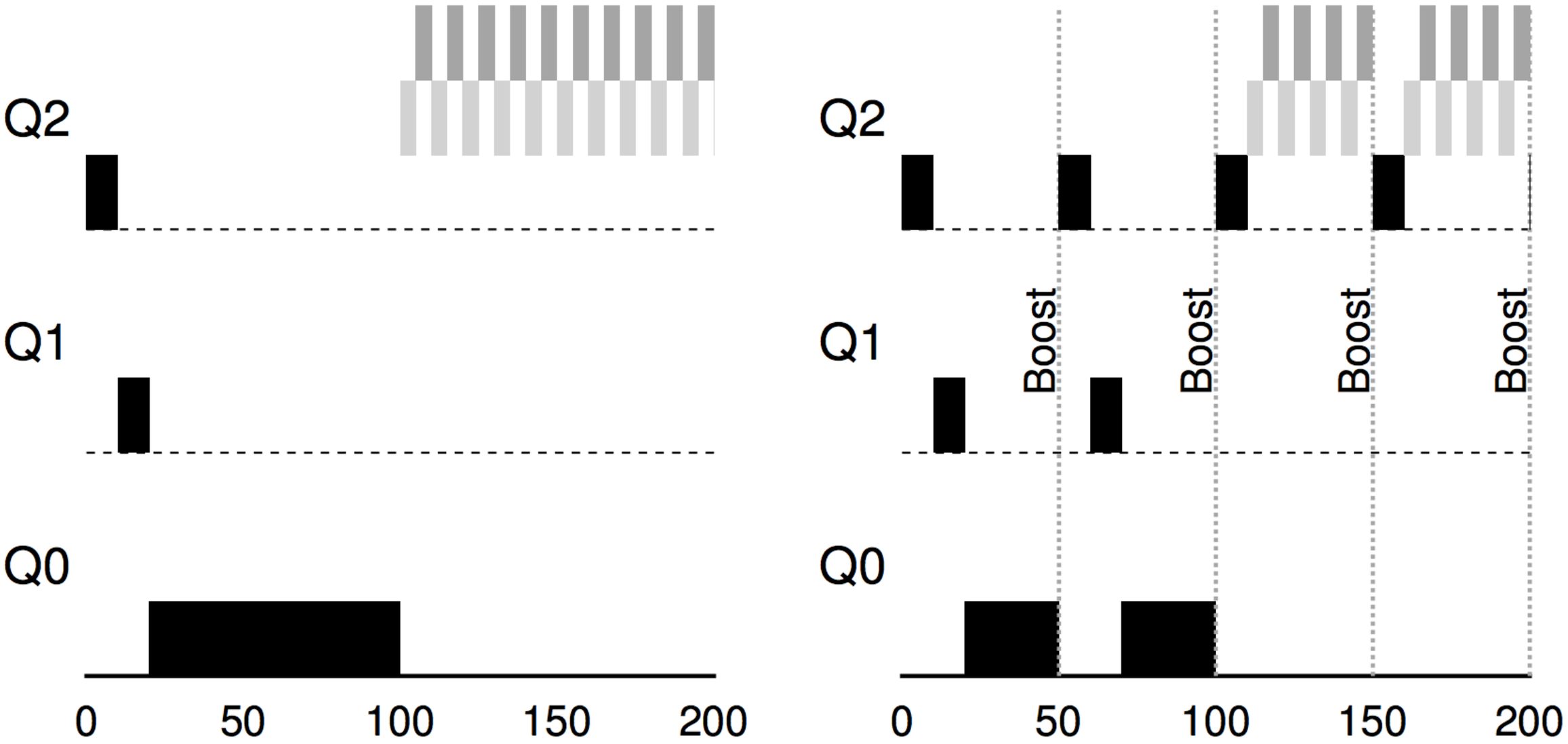


MLFQ Attempt 2: Priority Boost



MLFQ Attempt 2: Priority Boost

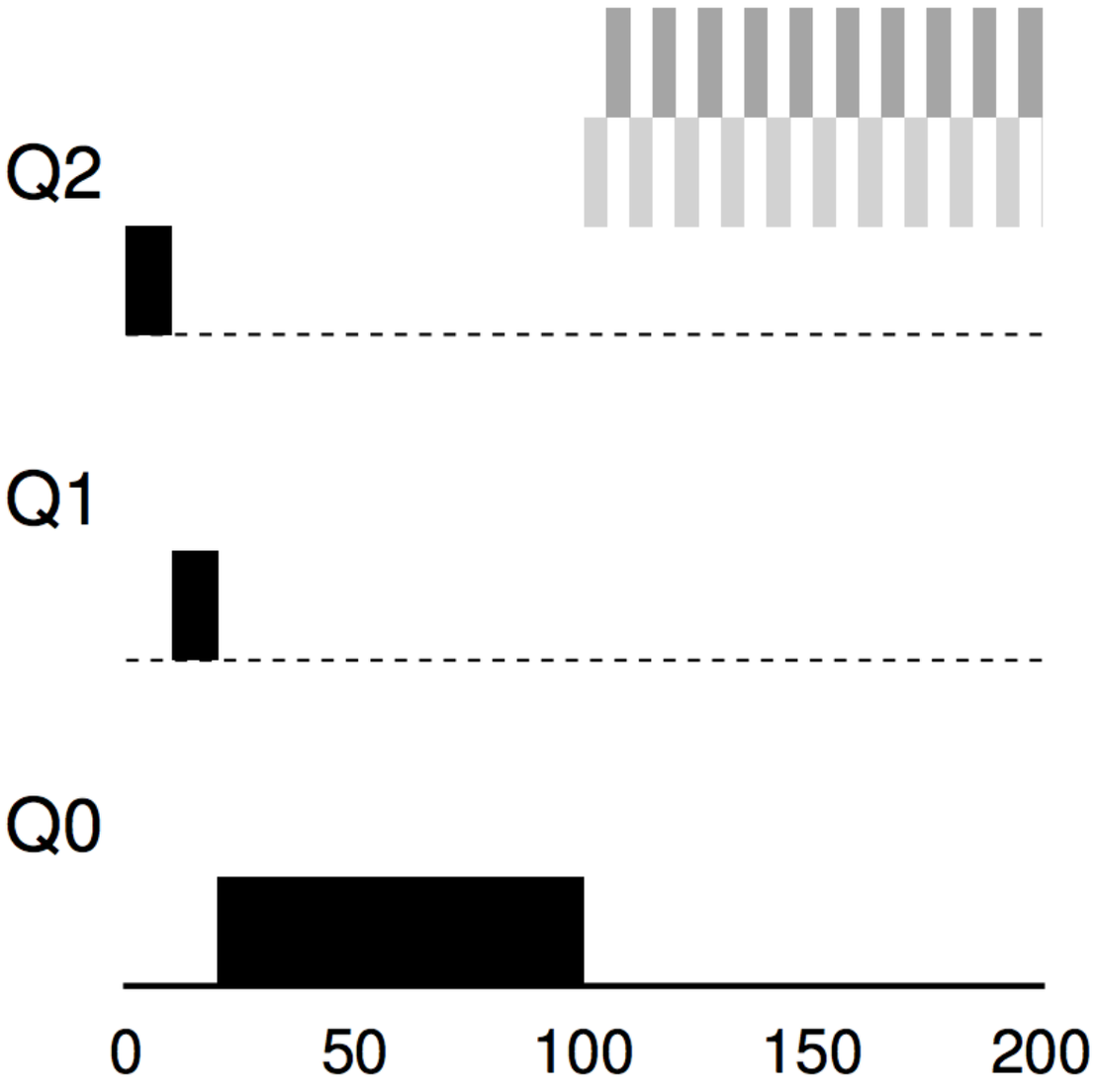
Rule 5: After some time period S, move all the jobs in the system to the topmost queue.

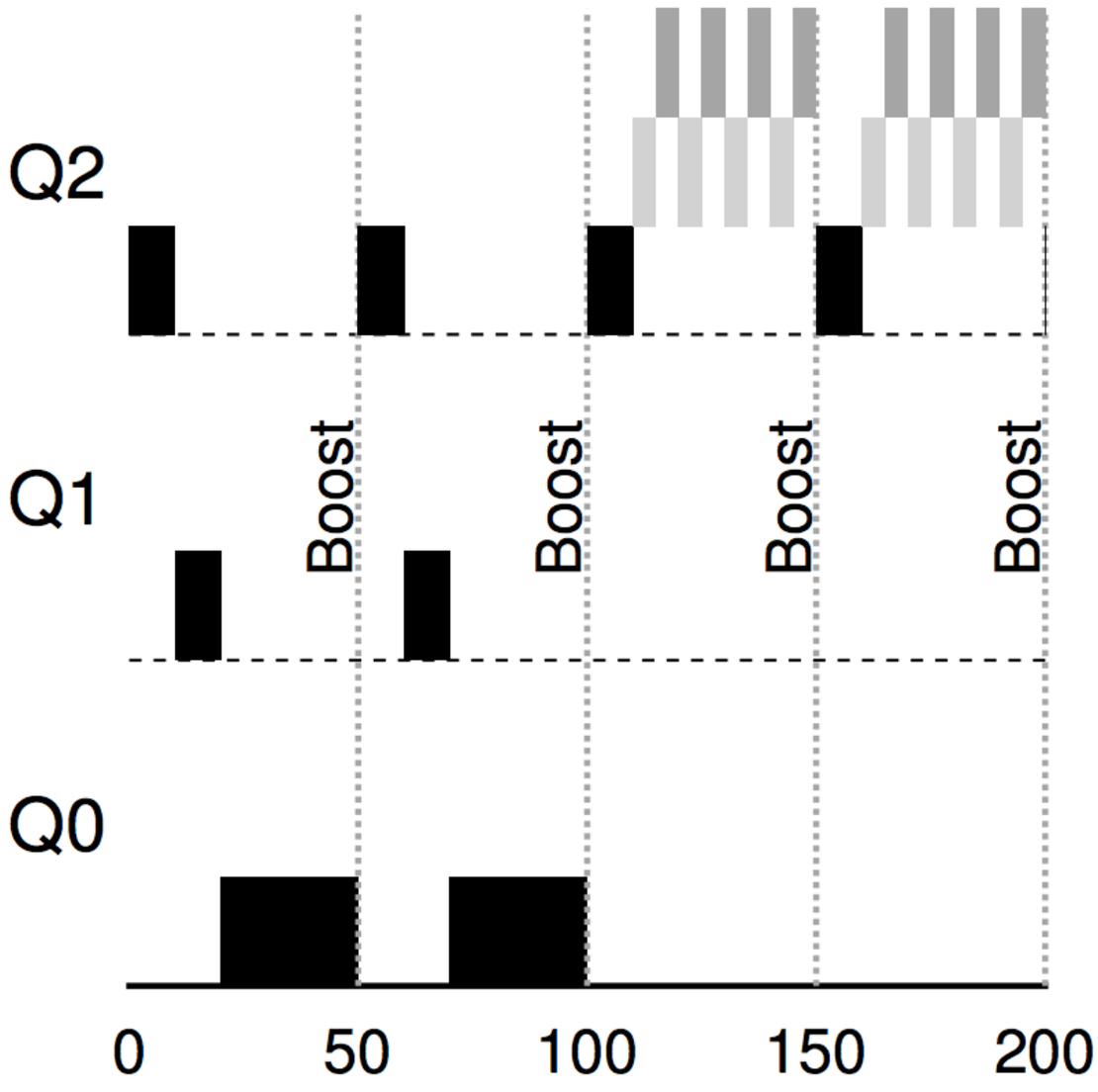


MLFQ Attempt 2: Priority Boost

Rule 5: After some time period S, move all the jobs in the system to the topmost queue.

- No process starvation

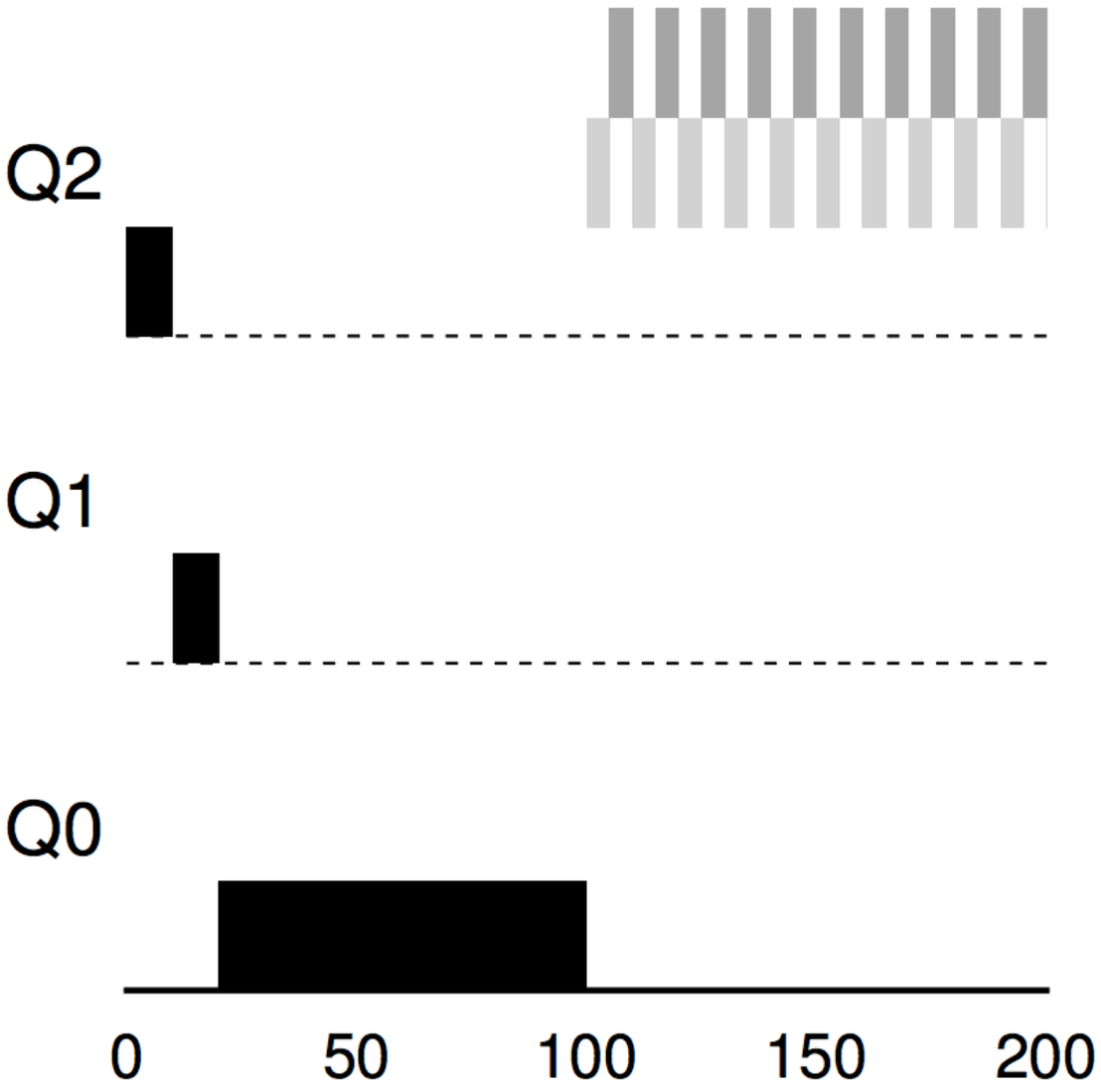


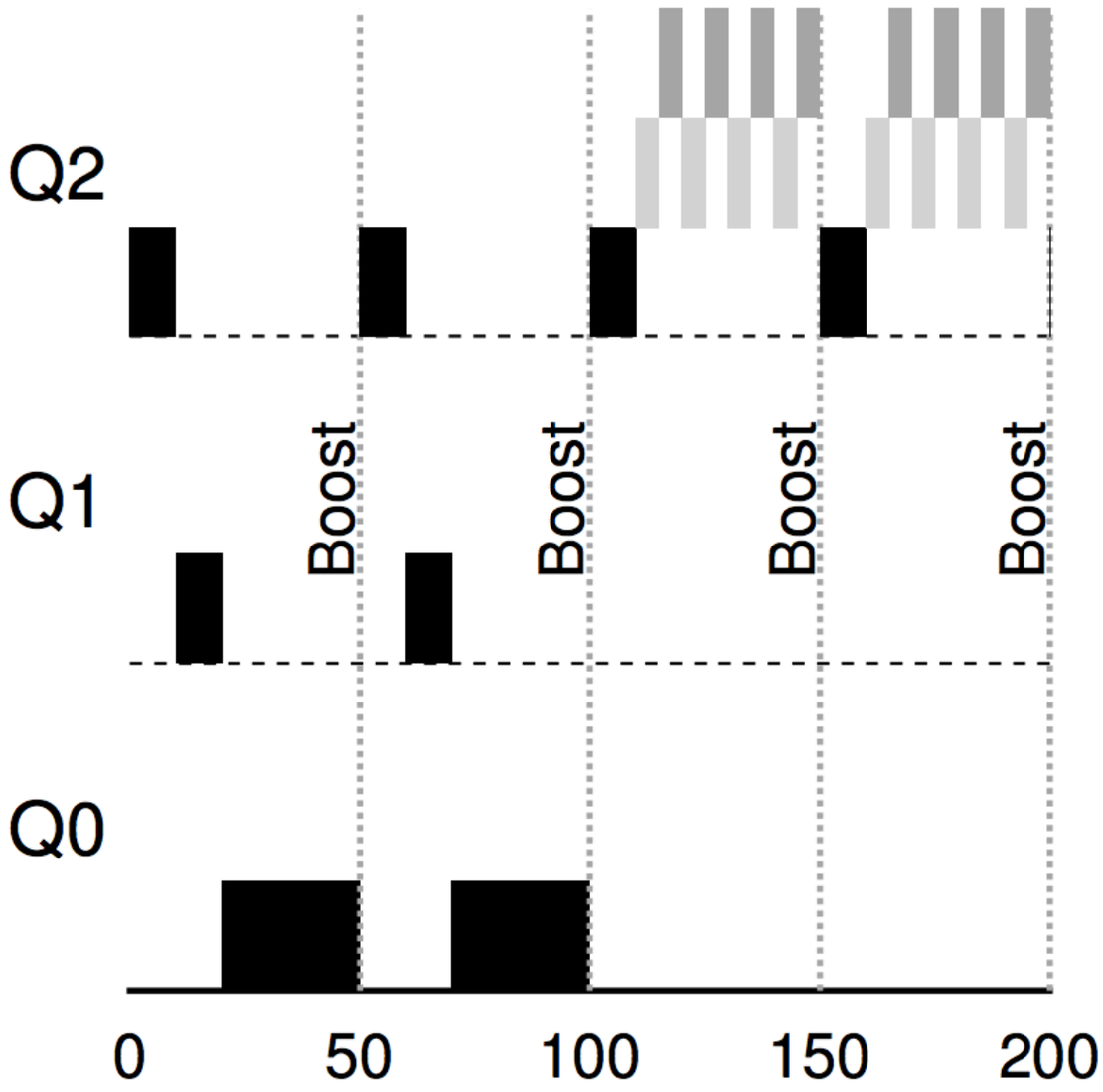


MLFQ Attempt 2: Priority Boost

Rule 5: After some time period S, move all the jobs in the system to the topmost queue.

- No process starvation
- Behaviour change handled





MLFQ Attempt 2: Priority Boost

How to choose S?

- Very high S -> Starvation
- Very low S -> Response time (in particular of interactive jobs) will get worse

MLFQ Attempt #1: Shortcomings

1. ~~Starvation : Too many I/O jobs will eat up the CPU~~; ~~no execution for CPU intensive ones~~
2. Scheduler gaming :
    1. Time slice = x
    2. Run CPU for 0.99*x
    3. Request I/O
    4. Remain in same priority
    5. Goto 2
3. ~~Behaviour change: CPU intensive went to lowest priority, but has loads of I/O after say y time units~~

MLFQ: Attempt 3: Better Accounting

Rule 4a Demote process if it uses up its quota

Rule 4b Process gives up CPU

before time-slice, remains in same priority

MLFQ: Attempt 3: Better Accounting

Rule 4a Demote process if it uses up its quota

Rule 4b Process gives up CPU

before time-slice, remains in same priority

Replace with

MLFQ: Attempt 3: Better Accounting

Rule 4a Demote process if it uses up its quota

Rule 4b Process gives up CPU

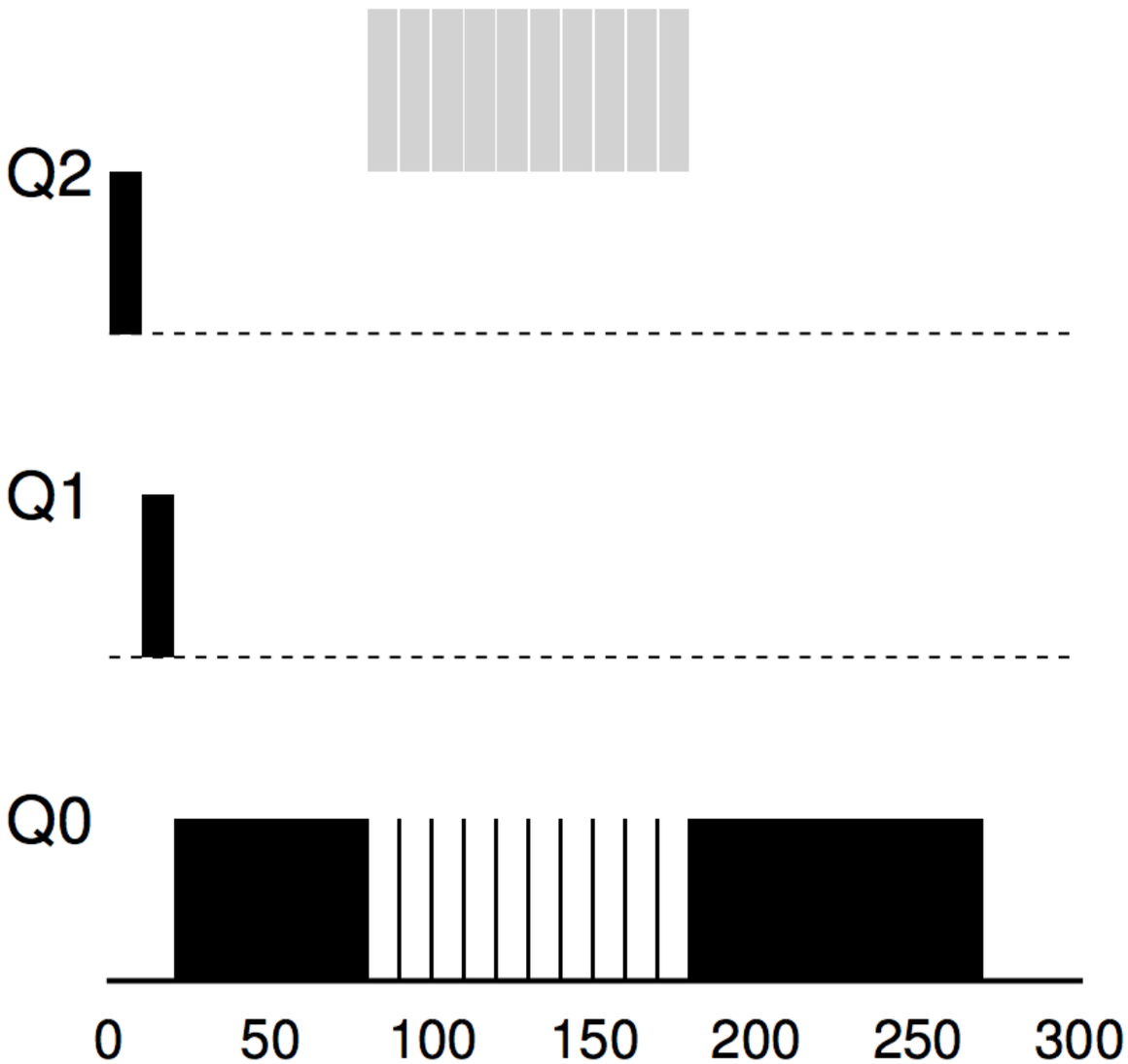
before time-slice, remains in same priority

Replace with

Rule 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).

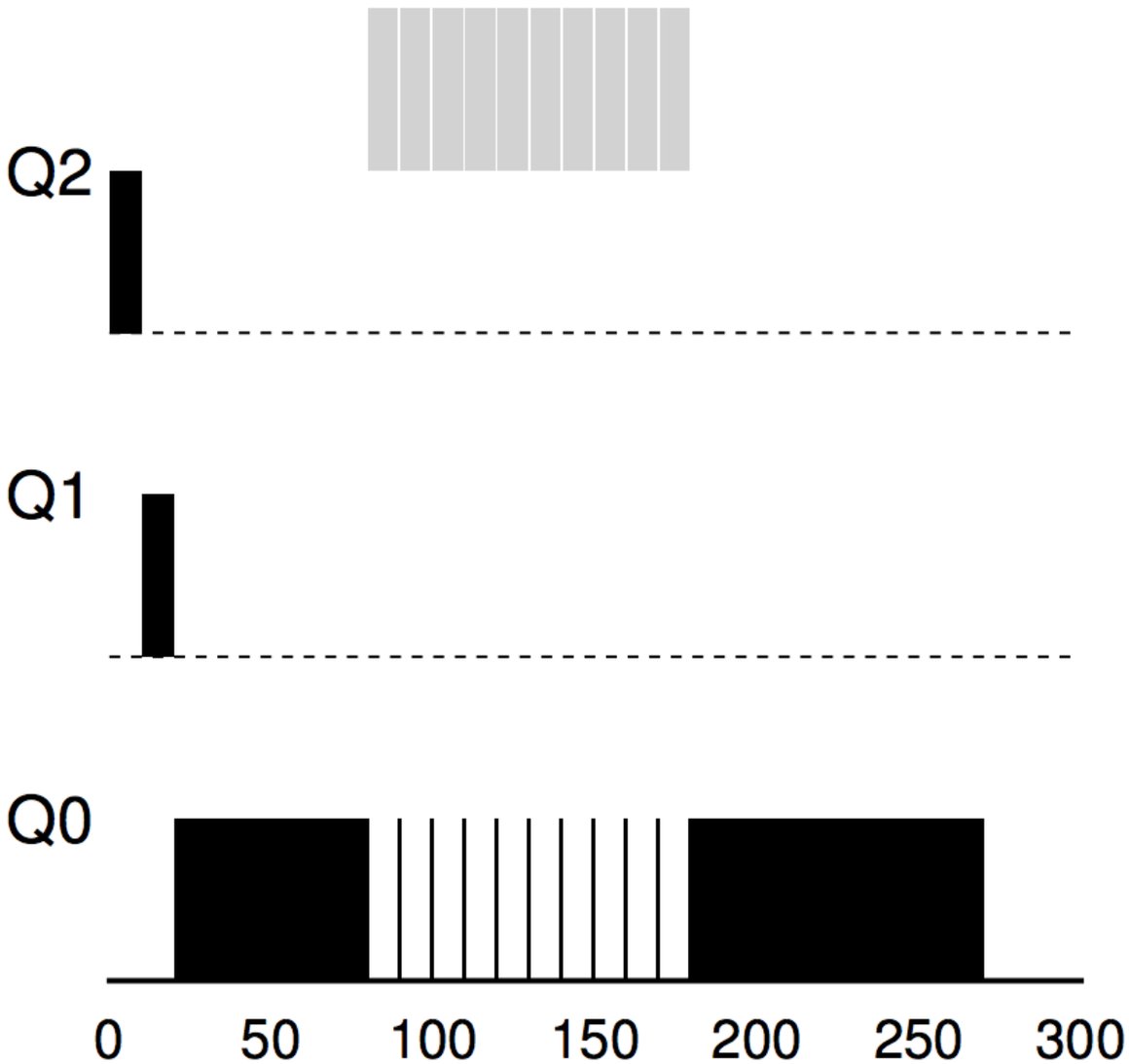
MLFQ: Attempt 3: Better Accounting

With Rule 4a and 4b

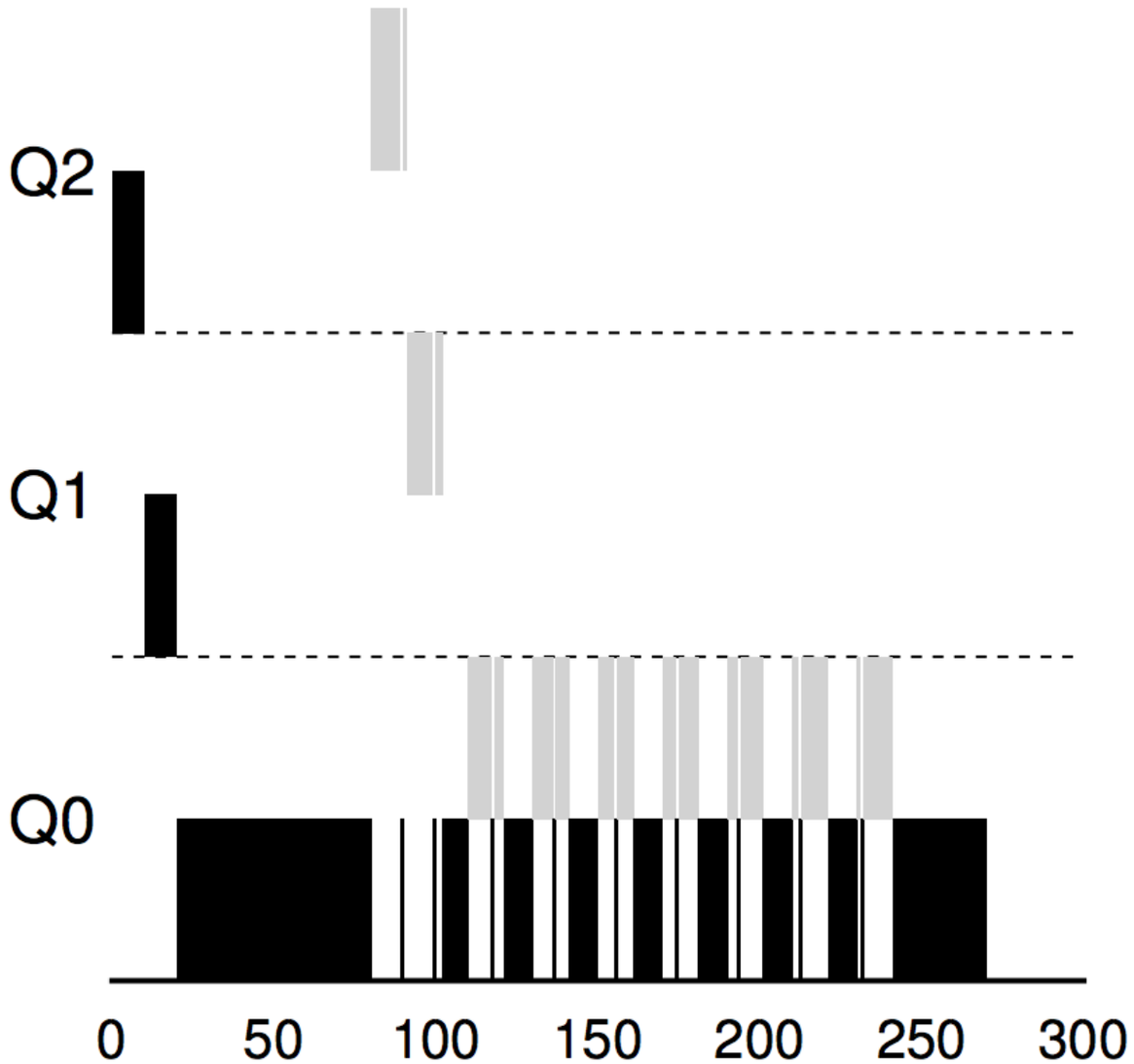


MLFQ: Attempt 3: Better Accounting

With Rule 4a and 4b



With new Rule 4



Summary

- Rule 1: If Priority(A) > Priority(B), A runs (B doesn't).
- Rule 2: If Priority(A) = Priority(B), A & B run in RR.
- Rule 3: When a job enters the system, it is placed at the highest priority (the topmost queue).
- Rule 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).
- Rule 5: After some time period S, move all the jobs in the system to the topmost queue.

Practice Session

```
./mlfq.py -s 5 -Q 10,10,10 -n 3 -j 3 -M 0 -m 30
```

```
OPTIONS jobs 3
OPTIONS queues 3
OPTIONS quantum length for queue  2 is  10
OPTIONS quantum length for queue  1 is  10
OPTIONS quantum length for queue  0 is  10
OPTIONS boost 0
OPTIONS ioTime 5
OPTIONS stayAfterIO False
OPTIONS iobump False


For each job, three defining characteristics are given:
  startTime : at what time does the job enter the system
  runTime   : the total CPU time needed by the job to finish
  ioFreq    : every ioFreq time units, the job issues an I/O
              (the I/O takes ioTime units to complete)

Job List:
  Job  0: startTime   0 - runTime  19 - ioFreq   0
  Job  1: startTime   0 - runTime  24 - ioFreq   0
  Job  2: startTime   0 - runTime  22 - ioFreq   0
```

Practice Session

```
./mlfq.py -s 5 -Q 2,10,15 -n 3 -j 3 -M 0 -m 30 -c
```

```
OPTIONS jobs 3
OPTIONS queues 3
OPTIONS quantum length for queue  2 is   2
OPTIONS quantum length for queue  1 is  10
OPTIONS quantum length for queue  0 is  15
OPTIONS boost 0
OPTIONS ioTime 5
OPTIONS stayAfterIO False
OPTIONS iobump False


For each job, three defining characteristics are given:
  startTime : at what time does the job enter the system
  runTime   : the total CPU time needed by the job to finish
  ioFreq    : every ioFreq time units, the job issues an I/O
              (the I/O takes ioTime units to complete)

Job List:
  Job  0: startTime   0 - runTime  19 - ioFreq   0
  Job  1: startTime   0 - runTime  24 - ioFreq   0
  Job  2: startTime   0 - runTime  22 - ioFreq   0
```