



Operating Systems

scheduling

Mohamed Zahran (aka Z)

mzahran@cs.nyu.edu

<http://www.mzahran.com>



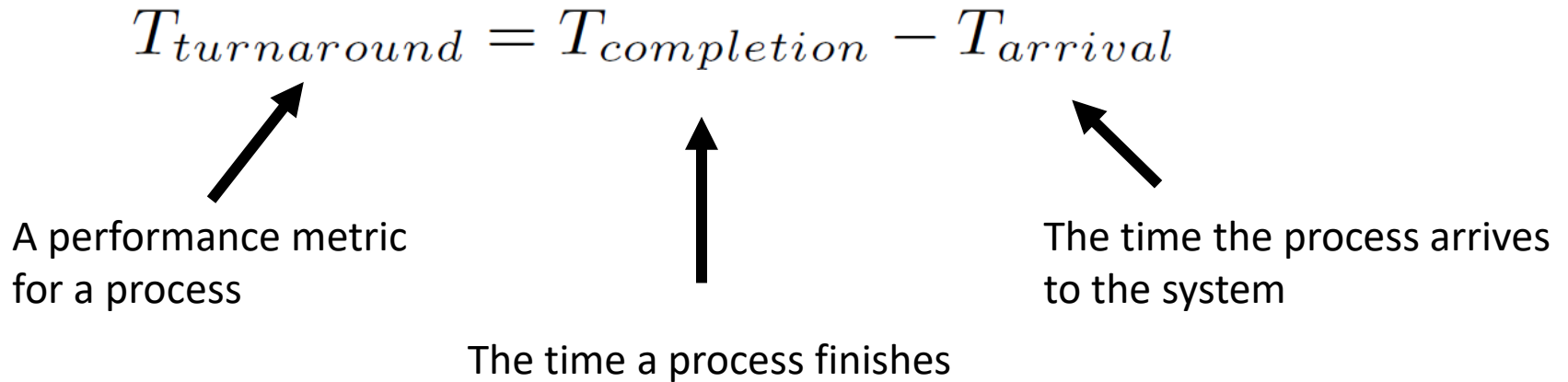
What are we discussing here?

- **Main Question:** How to decide which process to run next?
- This leads to many questions:
 - How should we develop a basic framework for thinking about scheduling **policies**?
 - What metrics are important?

Let's start with a simple metric:

$$T_{turnaround} = T_{completion} - T_{arrival}$$

A performance metric
for a process



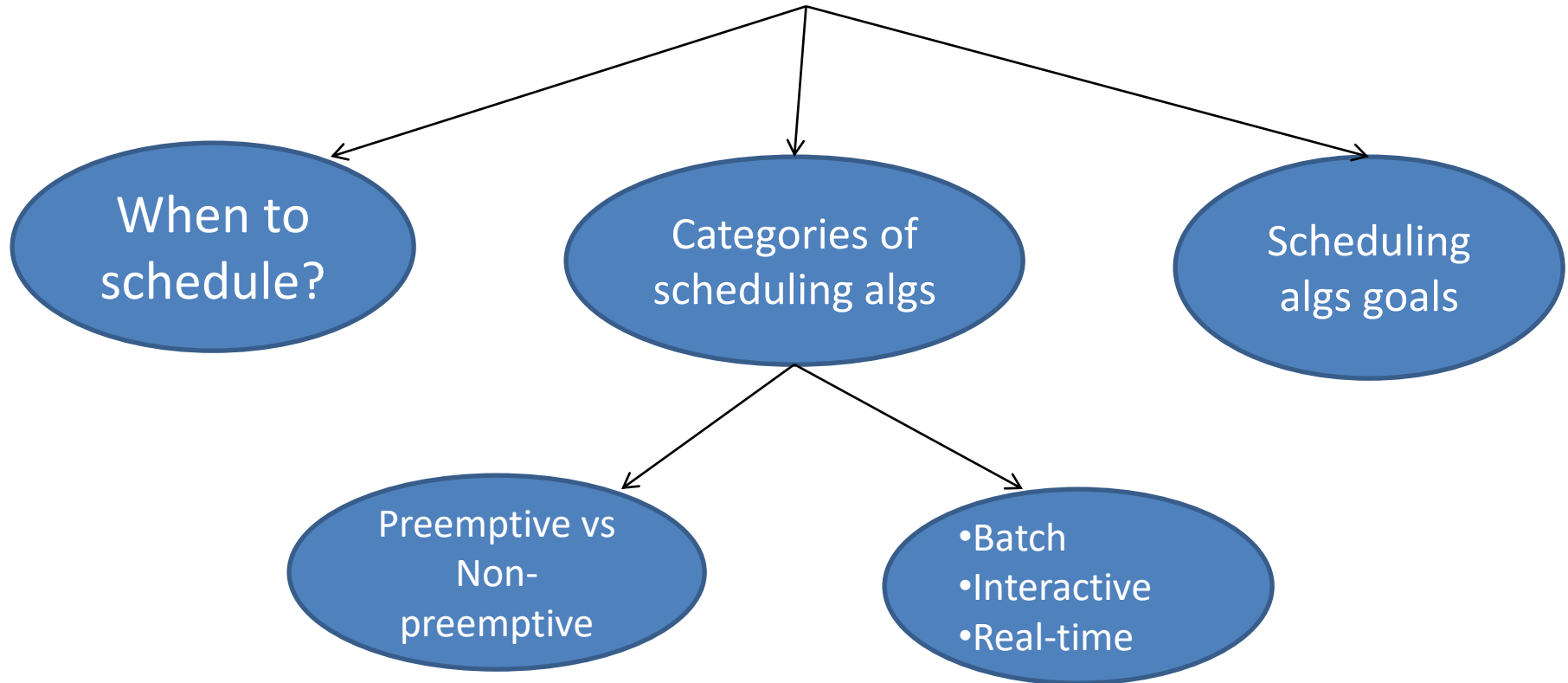
The time a process finishes

The time the process arrives
to the system

- In the scheduling lingo, a process is sometimes called a job.
- We will see more metrics as we proceed.

Scheduling

**Given a group of ready processes,
which process to run next?**



When to Schedule?

- When a process is created
- When a process exits
- When a process blocks
- When an I/O interrupt occurs

Definition

- **Preemptive** scheduling: A process can be interrupted, and another process scheduled to execute.
- **Non-preemptive** scheduling: The current running process must finish/exit first before another process is scheduled to execute.

Categories of Scheduling Algorithms

- Batch
 - No users impatiently waiting
 - mostly non-preemptive
- Interactive
 - preemption is essential
- Real-time
 - deadlines

Scheduling Algorithms Goals

All systems

Fairness - giving each process a fair share of the CPU

Policy enforcement - seeing that stated policy is carried out

Balance - keeping all parts of the system busy

Batch systems

Throughput - maximize jobs per hour

Turnaround time - minimize time between submission and termination

CPU utilization - keep the CPU busy all the time

Interactive systems

Response time - respond to requests quickly

Proportionality - meet users' expectations

Real-time systems

Meeting deadlines - avoid losing data

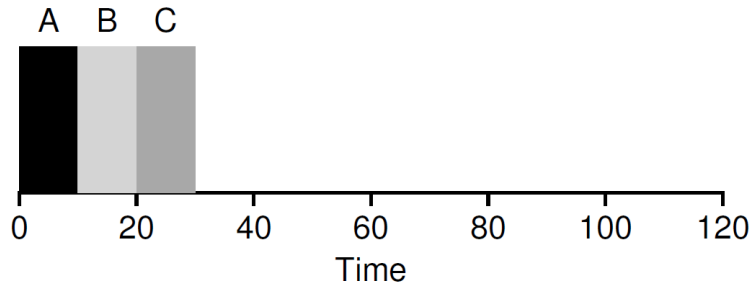
Predictability - avoid quality degradation in multimedia systems

Scheduling in Batch Systems: First-Come First-Served

- Non-preemptive
- Processes ordered as queue
- A new process added to the end of the queue
- A blocked process that becomes ready added to the end of the queue
- Main disadv: Can hurt I/O bound processes
 - Because they will be blocked for I/O then added to the end of the queue.

Examples

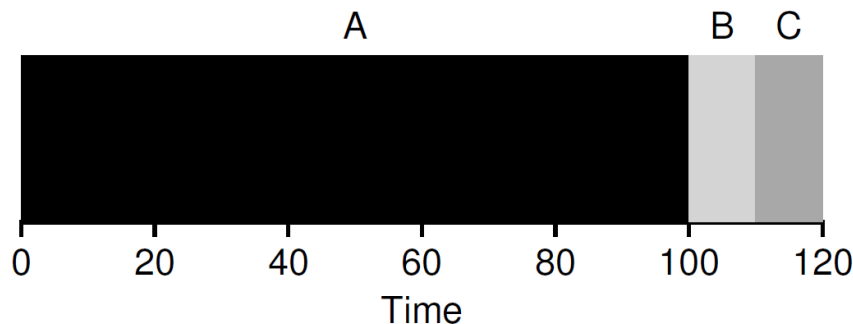
- Assume all processes arrive at the same time and each process needs 10 seconds.



Average turnaround time =
 $(10+20+30)/3 = 20$

Examples

- Assume all processes arrive at the same time and process A needs 100 seconds while processes B and C need 10 seconds each.

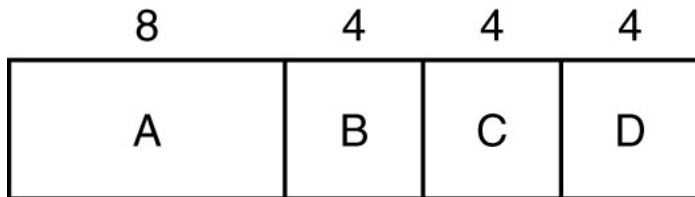


Average turnaround time =
 $(100 + 110 + 120) / 3 = 110$

This is a good example why FCFS may not be the best scheduling policy.

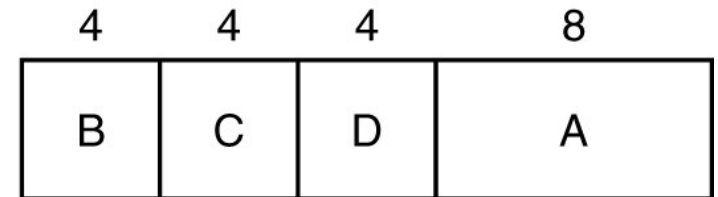
Scheduling in Batch Systems: Shortest Job First

- Non-preemptive
- Assumes runtime is known in advance
- Is only optimal when all the jobs are available simultaneously



(a)

Run in original order

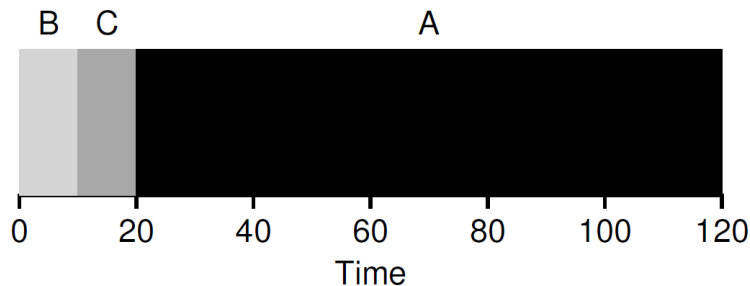


(b)

Run in shortest job first

Examples

- Assume all processes arrive at the same time and process A needs 100 seconds while processes B and C need 10 seconds each. [Same example as the one we saw in FCFS]

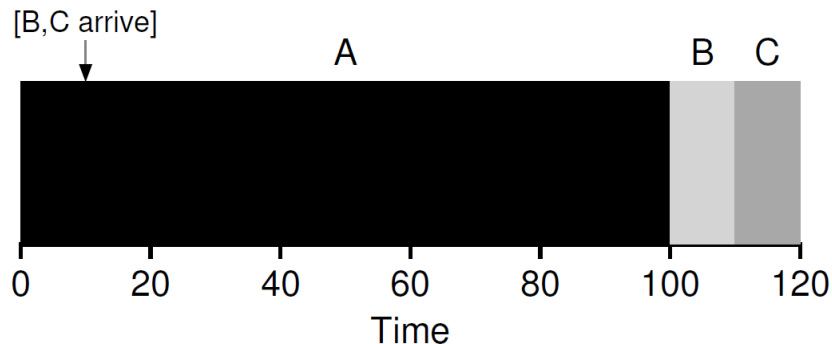


Average turnaround time =
 $(10+20+120) / 3 = 50$

More than twice better than FCFS
for the same example.

Examples

- Assume process A arrives at time 0. Processes B and C arrive at time 10. Process A needs 100 seconds while processes B and C need 10 seconds each.



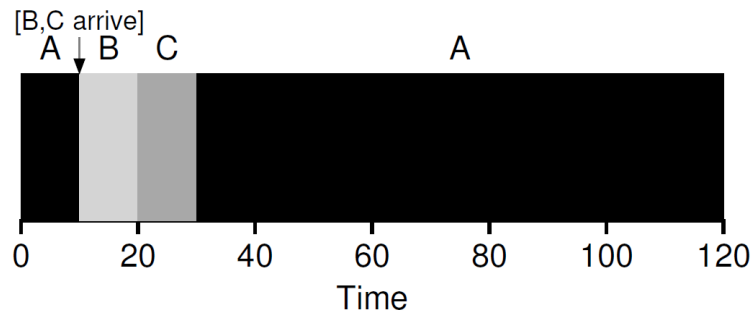
$$\begin{aligned}\text{Average turnaround time} &= \\ &= (100 + [110-10] + [120-10]) / 3 \\ &= 103.333\end{aligned}$$

Scheduling in Batch Systems:
Shortest Remaining Time Next
(aka: Shortest-Time-To-Completion First: STCF)

- Preemptive
- Scheduler always chooses the process whose remaining time is the shortest.
- Runtime must be known in advance.

Examples

- Assume process A arrives at time 0. Processes B and C arrive at time 10. Process A needs 100 seconds while processes B and C need 10 seconds each.



$$\begin{aligned}\text{Average turnaround time} &= \\ &= ([120-0] + [20-10] + [30-10]) / 3 \\ &= 50\end{aligned}$$

STCF seems the best

- When we have a batch system.

Most systems are
interactive now.



- When we know the total runtime of a process beforehand.

Not realistic



Let's introduce another metric

- Response time
 - In interactive systems
 - Depends on arrival time and the *first* time a process is scheduled to run

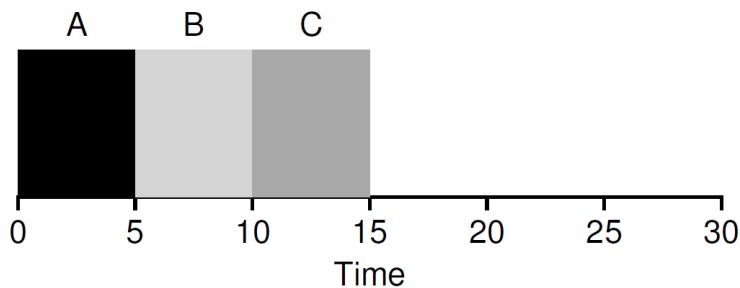
$$T_{response} = T_{firstrun} - T_{arrival}$$

Scheduling in Interactive Systems: Round-Robin

- Each process is assigned a time interval: **quantum (or time slice)**
- After this quantum, the CPU is given to another process
- What is the length of this quantum?
 - too short -> too many context switches -> lower CPU efficiency
 - too long -> poor response to short interactive

Example

Processes A, B, and C arrive at the same time 0. Each one needs to run for 5 seconds



SJF

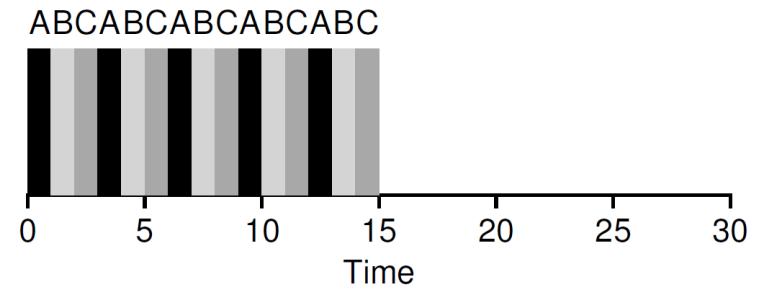
Response time for A: 0

Response time for B: 5

Response time for C: 10

Avg response time: 5

Avg turnaround: $(5+10+15)/3 = 10$



RR: Time slice = 1

Response time for A: 0

Response time for B: 1

Response time for C: 2

Avg response time: 1

Avg turnaround: $(13+14+15)/3 = 14$

Scheduling in Interactive Systems:

Priority Scheduling

- Each process is assigned a priority.
- Ready process with the highest priority is allowed to run.
- Priorities are assigned statically or dynamically.
- Must not allow a process to run forever
 - Can decrease the priority of the currently running process.
 - Use time quantum for each process.

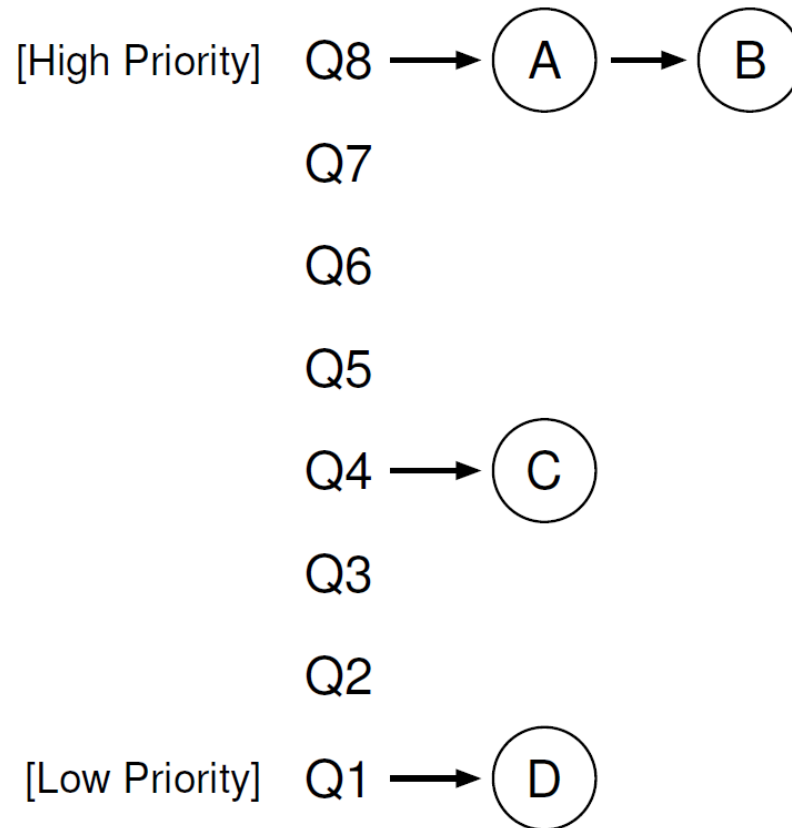
We need to

- Optimize turnaround time
- Reduce response time for interactive users
- Without knowing the total runtime of a process a priori.
- Multi-level Feedback Queue (MLFQ)

MLFQ: Basics

- We have several distinct queues.
- Each queue is assigned a different priority level.
- A process that is ready to run is on a single queue.
- A process priority may change overtime.
 - That is, assigned to a different queue.
- Rules:
 - **Rule 1:** If $\text{Priority}(A) > \text{Priority}(B)$, A runs (B doesn't).
 - **Rule 2:** If $\text{Priority}(A) = \text{Priority}(B)$, a case where A & B are in the same queue, A & B run in RR.

MLFQ: Basics



MLFQ: How does a process's priority change?

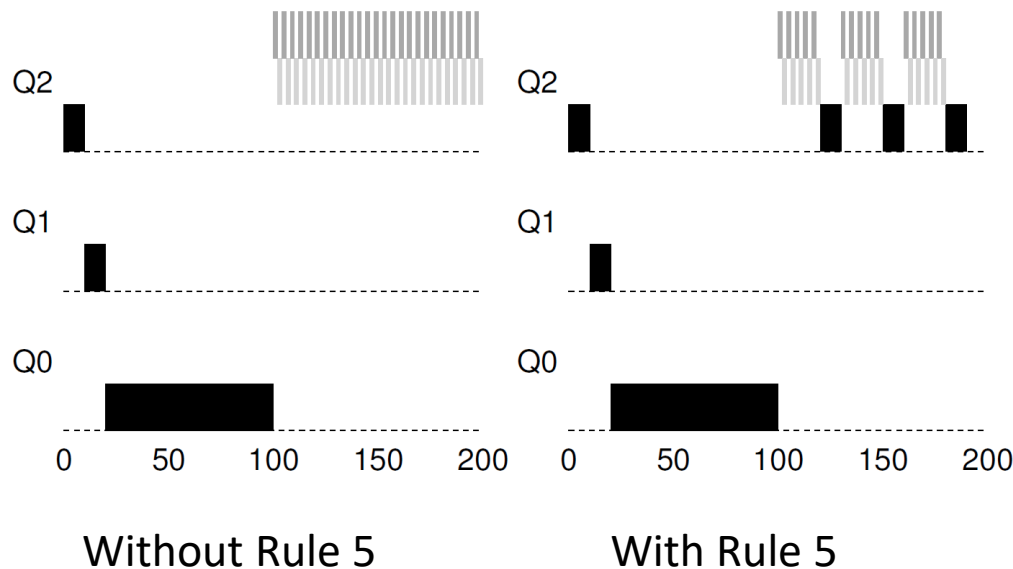
- **Rule 3:** When a job enters the system, it is placed at the highest priority (the topmost queue).
- **Rule 4a:** If a job uses up an entire time slice while running, its priority is reduced (i.e., it moves down one queue).
- **Rule 4b:** If a job gives up the CPU before the time slice is up, it stays at the same priority level.

MLFQ: Problematic Scenarios

- **Starvation:** A long CPU intensive process will go down to the lowest queue. If there are many interactive processes (i.e. will be in higher priority queue), that long process will starve.
- **Gaming the system:** A process can use 99% of its time slice, then do an unneeded I/O to relinquish the CPU and stay in the same queue.
- **Changing behavior:** What if a process changes its behavior from being a CPU-intensive one to an interactive one? We don't have rules to move a process up.

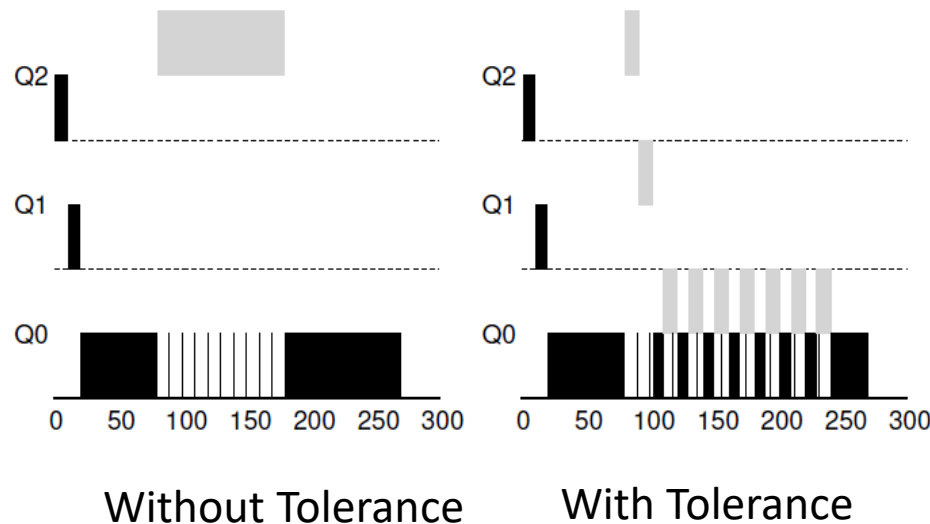
MLFQ: Priority Boost

- **Rule 5:** After some time period S , move all the jobs in the system to the topmost queue.
- This deals with the first and third problematic scenarios.



MLFQ: Gaming Tolerance

- **Revisiting 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).



Tuning MLFQ

- How many queues?
- What is time S used in rule 5 (to boost processes to top queue)?
- How big is a time slice?
- etc

Scheduling in Real-Time

- Process must respond to an event within a deadline.
- Hard real-time vs soft real-time
 - Hard: Result/Response becomes incorrect if you miss the deadline. Used in critical applications like medical, air-traffic control,
 - Soft: If deadline is missed, system is still correct but with degraded performance. Example: computer games.
- Periodic vs aperiodic events
- Processes must be schedulable
- Scheduling algorithms can be static or dynamic

Conclusion

- Scheduling is about policy: deciding which process to run on a CPU next.
- The main goals: reduce response time and optimize turnaround time.
- We have seen few policies but there are many more.