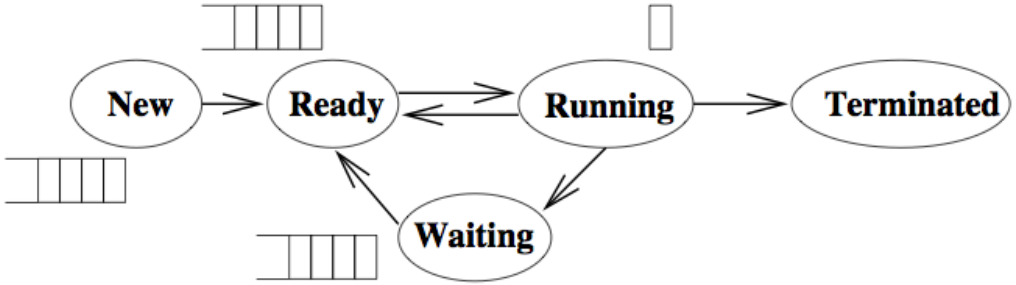
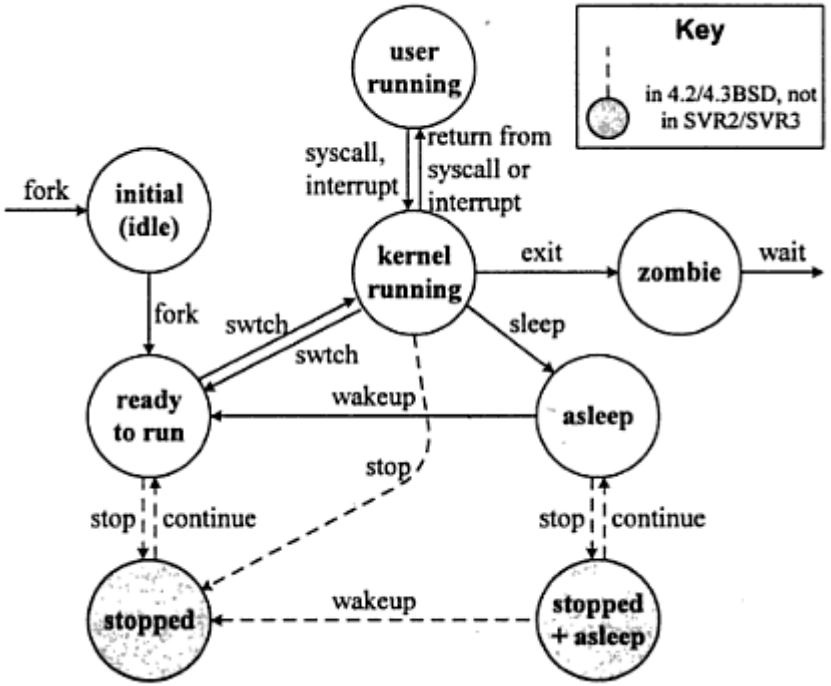
Operating Systems

Scheduling (Lecture-3)

Monsoon 2014, IIIT-H, Suresh Purini

Process States and State Transitions





Two Policy Questions

1. When do we call the scheduler?
2. How does the scheduler chooses the next process?

Mechanism Question: How does the actual context switch happens?

Scheduling the Scheduler

The kernel runs the scheduler whenever

- Some process voluntarily relinquishes the CPU
  - I/O block, termination, SIGSTOP, sleep, ….
- a processes time slice is over (timer ISR)
- ……..
- ……..
- Question: What if a process is in the kernel mode and its time slice is over?

Non-preemptive Kernels

- Non Pre-emptive Kernels: A process executing in kernel mode cannot be preempted by another process even though its time quantum expires.
- Pre-emptive Kernels and Synchronization Issues

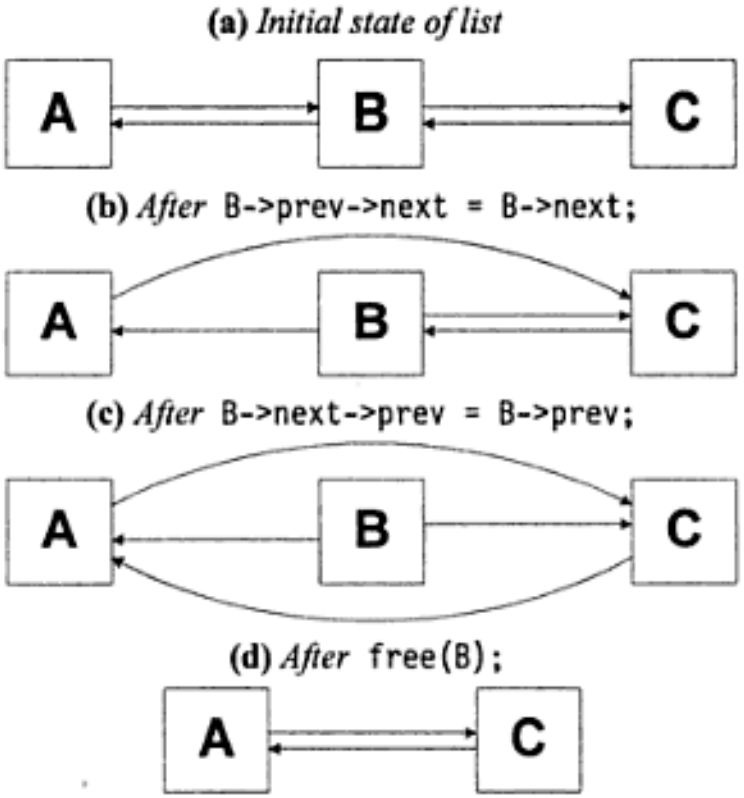


Figure 2-6. Removing an element from a linked list.

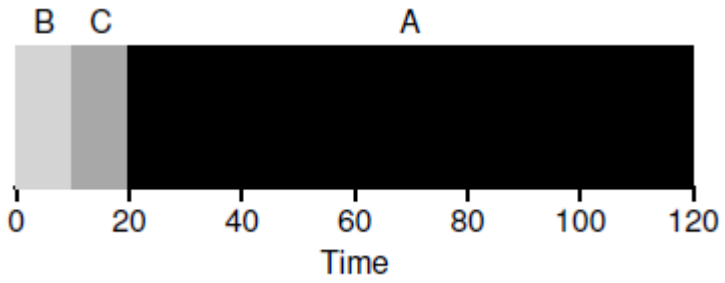
Workloads and Scheduling Metrics

- Interactive Applications – Shells, editors, etc.
  - Response time: (first response time – arrival time)
- Batch Applications – scientific computations, kernel compilation, etc.
  - Turnaround time: (completion time – arrival time)
- Real-time Applications – Audio/Video players, missile tracking processes, etc.
  - Strict deadlines

Batch Applications

Assumptions

- All jobs arrive at the same time
- We know the job times ahead
- No I/O blocking (not necessary)
- What should be the scheduling policy to minimize the turnaround time?

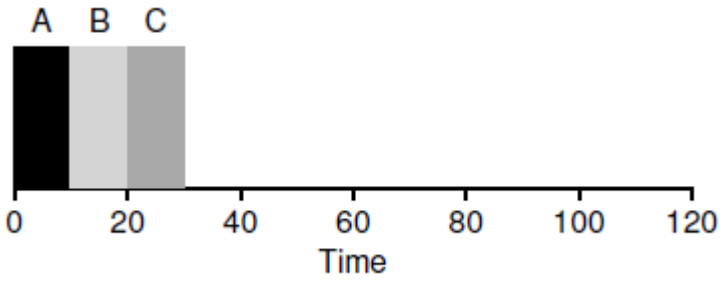


SJF

Batch Applications

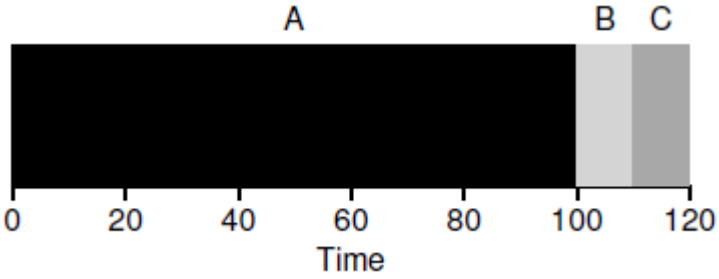
Assumptions

- We know the job times ahead but jobs can have different arrival times
- No I/O blocking (not necessary)
- What should be the scheduling policy to minimize the turnaround time?



FIFO

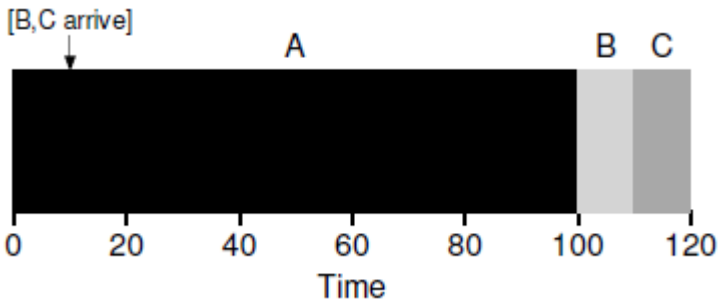
Convoy Effect

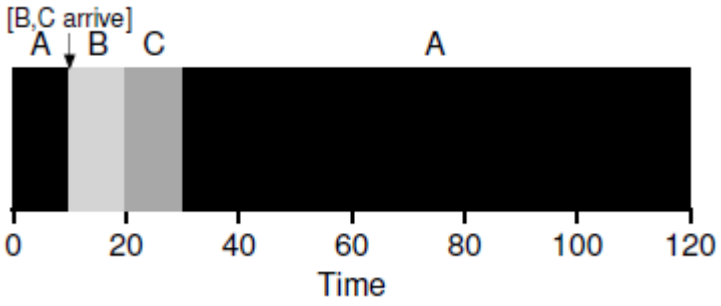


Batch Applications

Assumptions

- We know the job times ahead but jobs can have different arrival times
- No I/O blocking (not necessary)
- What should be the scheduling policy to minimize the turnaround time?



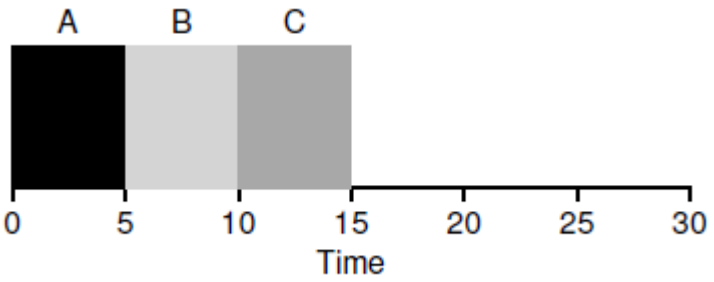


STCF

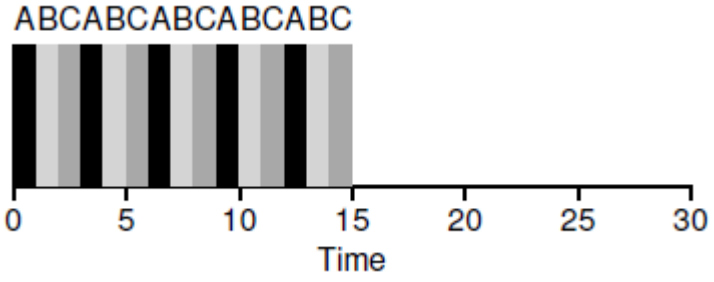
Optimizing for Response Time

Assumptions

- We know the job times ahead but jobs can have different arrival times
- No I/O blocking (not necessary)
- What should be the scheduling policy to minimize the response time?
  - How would it impact the turnaround time?



SJF

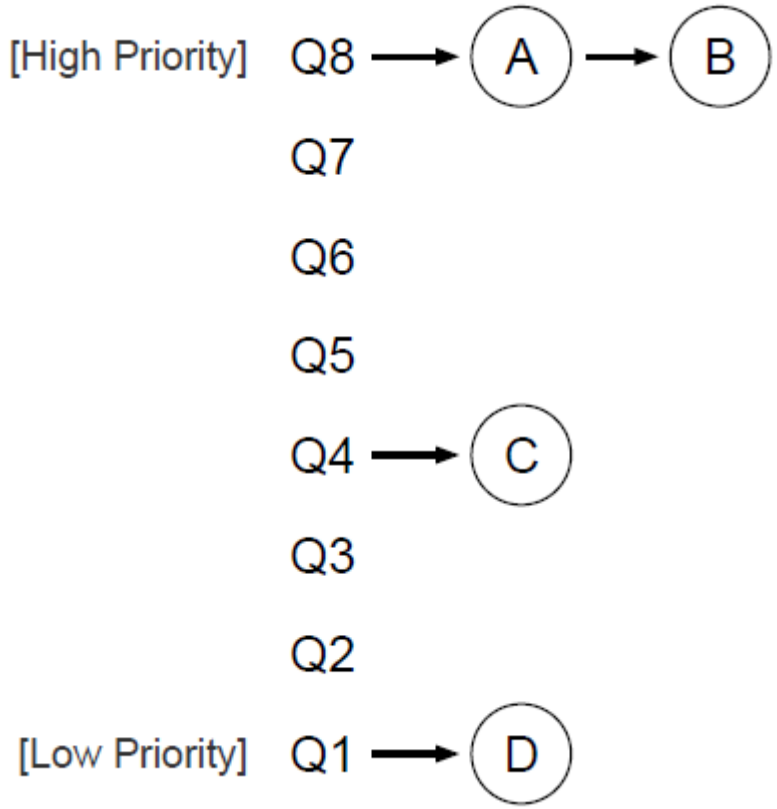


Round Robin

As time slice length increases the amortized cost decreases.

Multi-Level Feedback Queues

- Rule 1: If Priority(A) > Priority(B), A runs (B doesn't).
- Rule 2: If Priority(A) = Priority(B), A & B run in RR.



- Any problems here?

Multi-level Feedback Queues

- 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.*
- Any problems here?

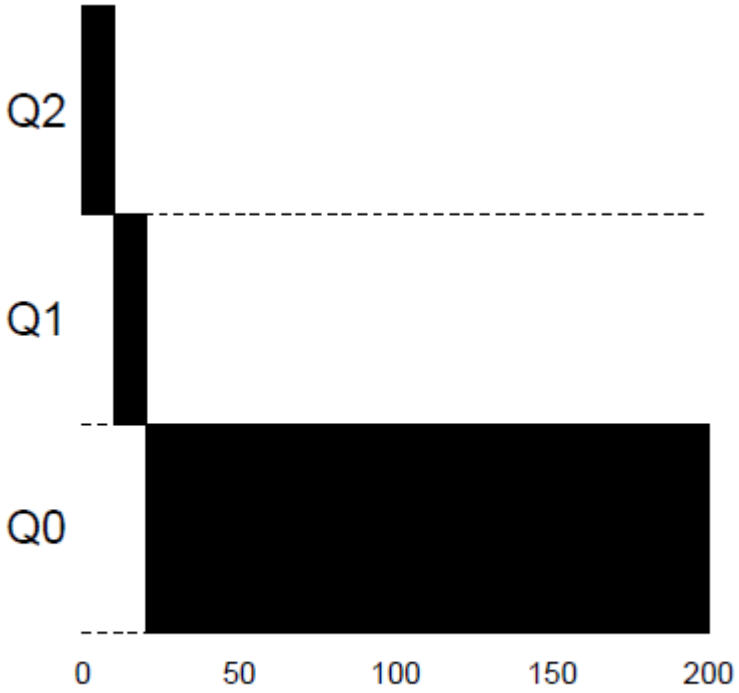


Figure 8.2: **Long-running Job Over Time**

Mixture of Interactive Jobs and Long Running Jobs

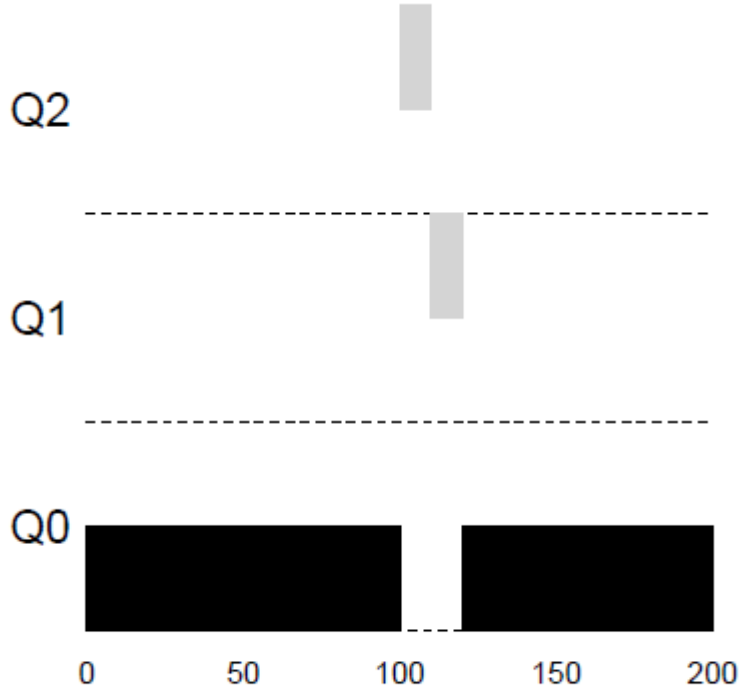
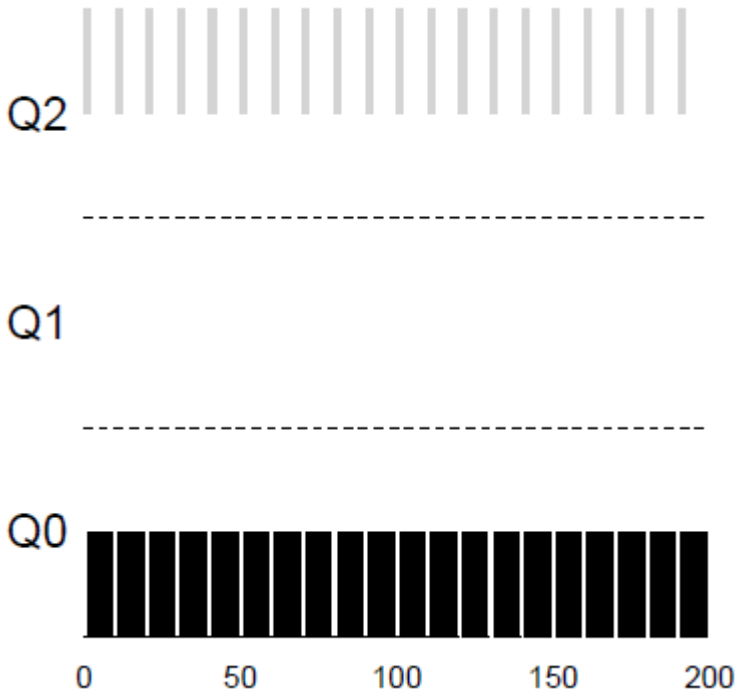


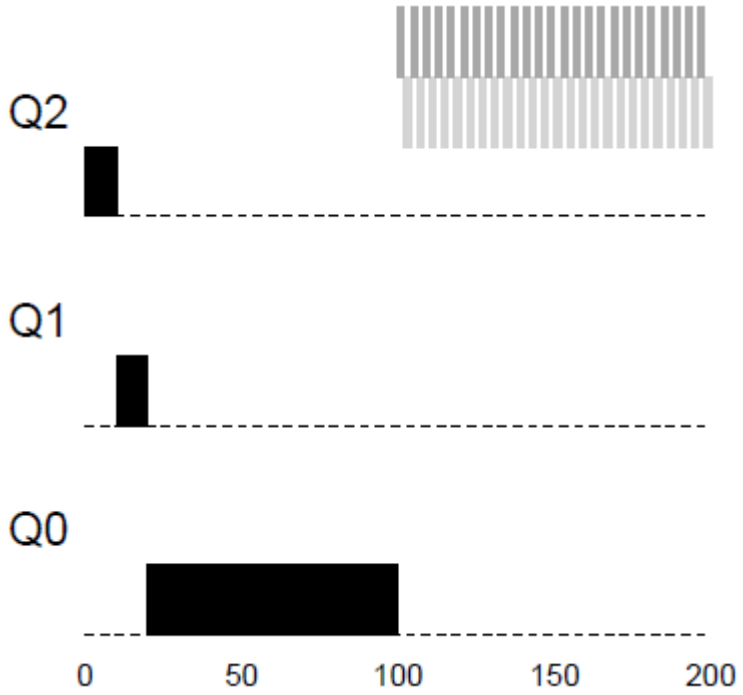
Figure 8.3: **Along Came An Interactive Job**

Mixture of I/O Intensive and CPU Intensive Workloads

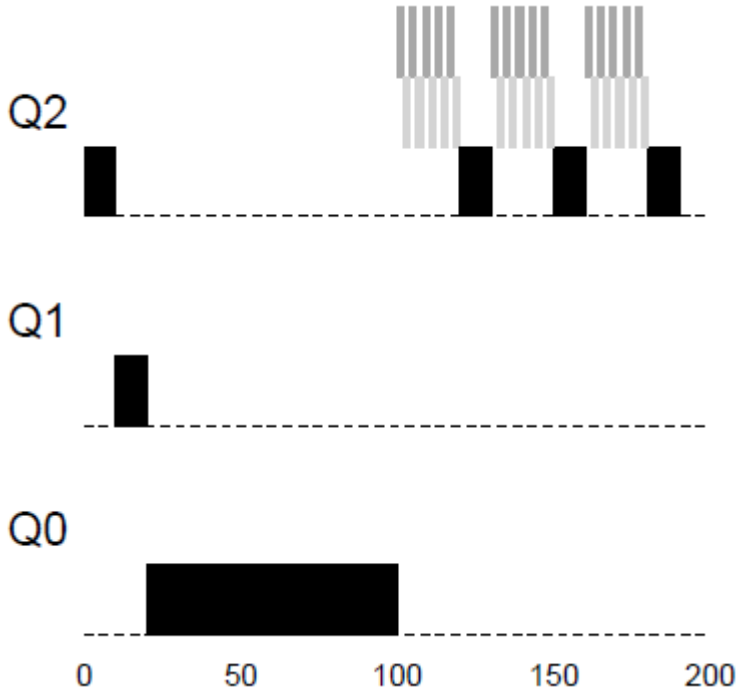


So Far So Good

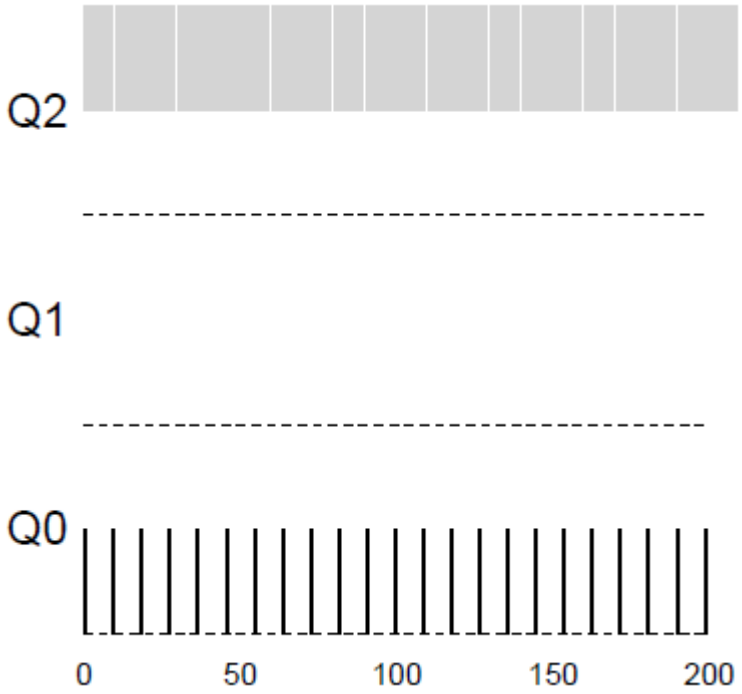
What is happening here?



Priority Boosting



Gaming the Scheduler



Multi-level Feedback Queues

- Priority Levels: A priority level is assigned to every process (let us say between 0 and 100.
  - Lower value means higher priority
  - Processes stuck in the kernel gets high priority (let us say between 0 and 25)
- Always the process at the highest priority gets the CPU.
  - If more than two processes at this level, do round robin.
- Dynamic priority assignment
  - Process priority = base priority + nice value + (“recent CPU usage”/constant)