Lecture 4: Scheduling Policies

Operating Systems

Content taken from: https://www.cse.iitb.ac.in/~mythili/os/

Last Class

- How does the OS run a process?
- How does it handle a system call?
- How does it context switch from one process to the other?

The OS scheduler

- OS scheduler has two parts
  - Mechanism to switch from process A to process B (last lecture)
  - Policy to pick which process to run next (this lecture)
- **What is a scheduling policy?**
  - On context switch, which process to run next, from set of ready processes?

Workload Assumptions

1. Each job runs for the same amount of time.
2. All jobs arrive at the same time.
3. Once started, each job runs to completion.
4. All jobs only use the CPU (i.e., they perform no I/O)
5. The run-time of each job is known.

Scheduling Metric: Turnaround Time

- Turnaround time = time from process arrival to completion

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

- **Goal:** Minimize the average turnaround time for all the processes

First-In-First-Out (FIFO)

- Example: three processes A, B, C arrive at t=0 (same time)
- Compute the average turnaround time
  - Arrival time is 0s
  - Each job runs for 10s
  - Avg TT = (10+20+30)/3 = 20

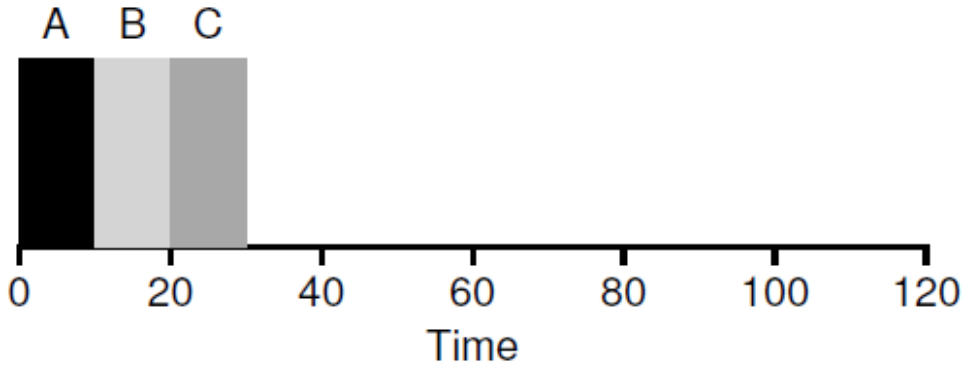

Figure 7.1: **FIFO Simple Example**

First-In-First-Out (FIFO)

- What happens if we relax assumption 1?
- Jobs don't have the same run time
- Recompute the average turnaround time
- Problem: convoy effect
- Turnaround times tend to be high
  - A runs for 100s, B and C for 10s
  - Avg TT = (100+110+120)/3 = 110

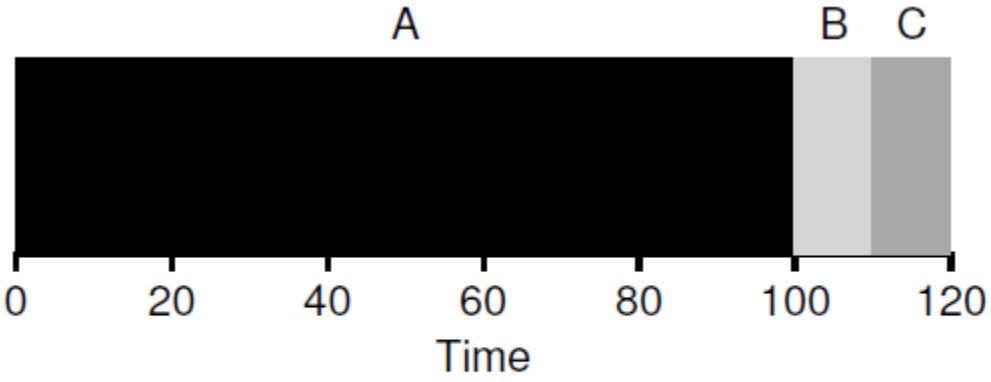

Figure 7.2: **Why FIFO Is Not That Great**

Shortest Job First (SJF)

- Run the shortest job first, then the next shortest, and so on…
- Example: three processes A, B, C arrive at t=0 (same time)
- Provably optimal when all processes arrive together.
- Compute the average turnaround time for this scenario
  - Avg TT = (10+20+120)/3 = 50

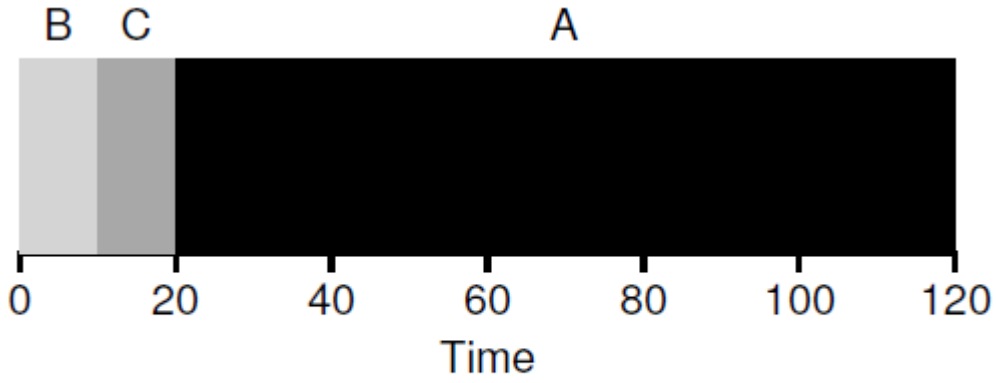

Figure 7.3: **SJF Simple Example**

Shortest Job First (SJF)

- What happens if we relax assumption 2?
- Jobs can arrive at any time instead of all at once.
- A arrives at t=0, B and C arrive at t=10.
- Recompute the average turnaround time for this scenario
  - Avg TT = (100+(110-10)+(120-10))/3 = 103.33 s
- SJF is non-preemptive, so short jobs can still get stuck behind long ones.

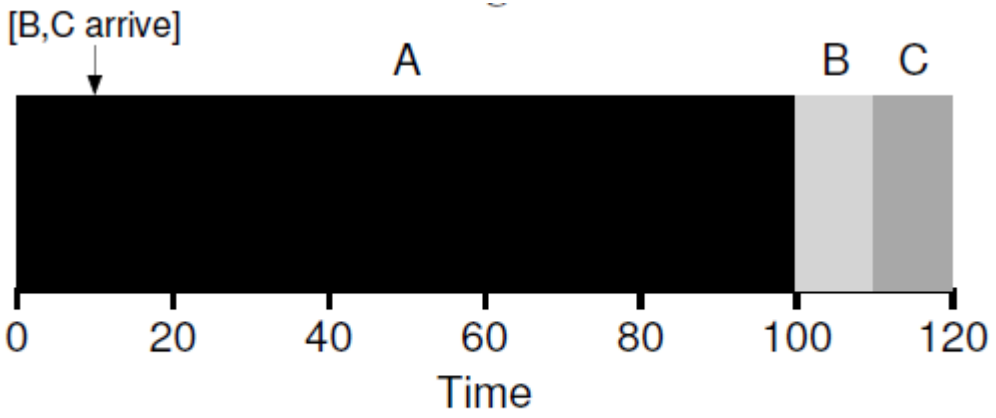

Figure 7.4: **SJF With Late Arrivals From B and C**

Shortest Time-to-Completion First (STCF)

- Also called Shortest Remaining Time First (SRTF)
- Preemptive scheduler
- Preempts running task if time left is more than that of new arrival
- Requires relaxing assumption 3: jobs can be preempted.
- What is the average turnaround time in this scenario?
  - Avg TT = ((120-0)+(20-10)+(30-10))/3 = 50

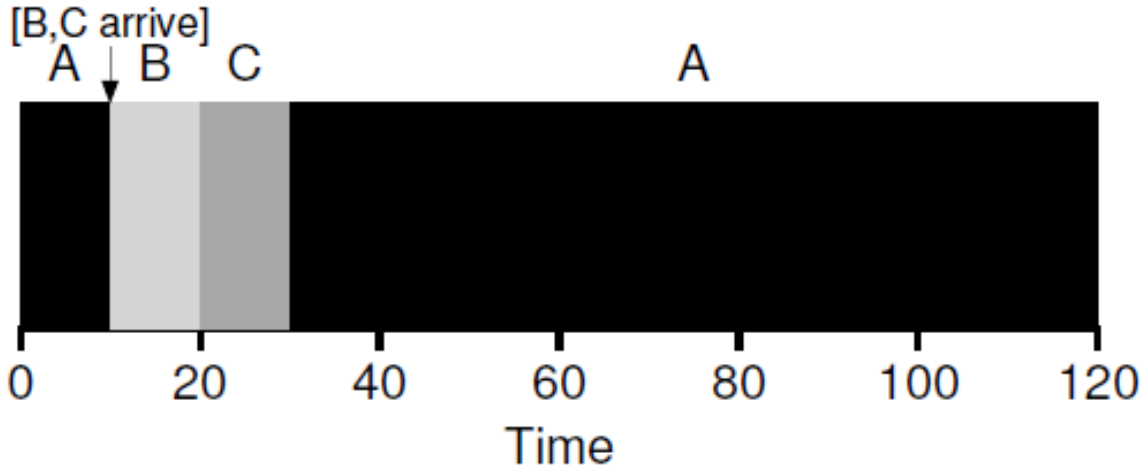

Figure 7.5: **STCF Simple Example**

Another Scheduling Metric: Response Time

- Response time is the time from when the job arrives in a system to the first time it is scheduled.

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

- What is the average response time in the below scenario?

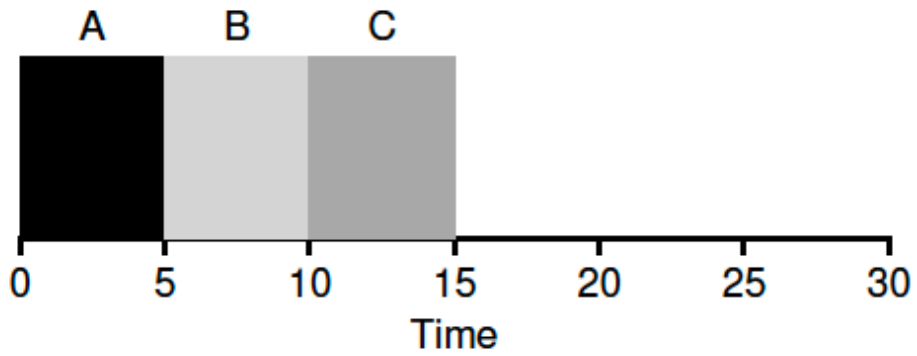

Figure 7.6: **SJF Again (Bad for Response Time)**

Round Robin (RR)

- Every process executes for a fixed quantum slice instead of running to completion.
- Slice big enough to amortize cost of context switch
- Preemptive
- Compute the average response time and turnaround time
- Good for response time and fairness
- Bad for turnaround time
  - A, B, C arrive at the same time and each runs for 5 seconds
  - The time slice is 1 second
  - Avg RT = (0+1+2)/3 = 1; RT for SJF is (0+5+10)/3 = 5
  - Avg TT = (13+14+15)/3 = 14

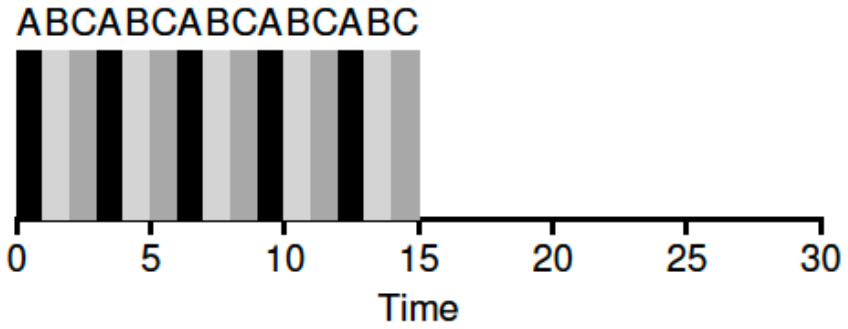

Figure 7.7: **Round Robin (Good For Response Time)**

Trade-off

- Round-Robin (RR) gives good response time but bad turnaround time
- Shortest-Job First (SJF) and Shortest Time-to-Completion First (STCF) gives bad response time but good turnaround time

Incorporating I/O

- Relax assumption 4
  - What if jobs performed I/O as well?
- Interactive jobs:
  - CPU burst -> I/O burst -> CPU burst -> I/O burst….
  - A (50 s of CPU and each I/O takes 10 s); B needs only CPU (50 s)

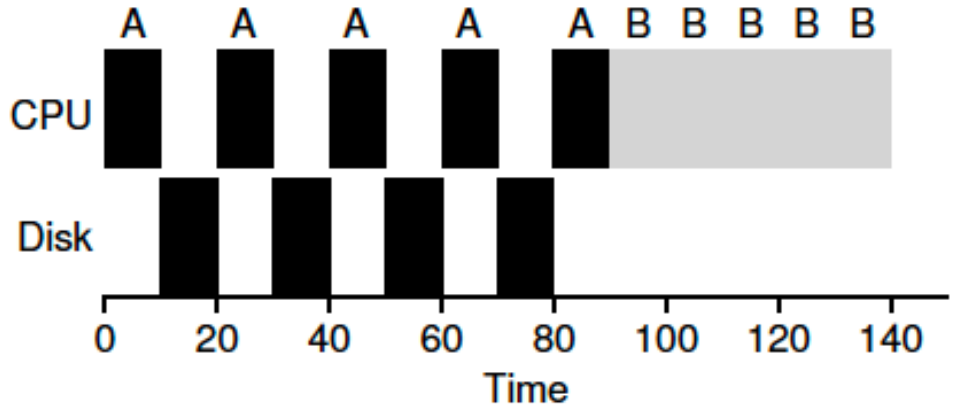

Figure 7.8: **Poor Use Of Resources**

Treat each CPU burst as a job/process

- Allow the CPU to be used by one process while waiting for the I/O of another process to complete

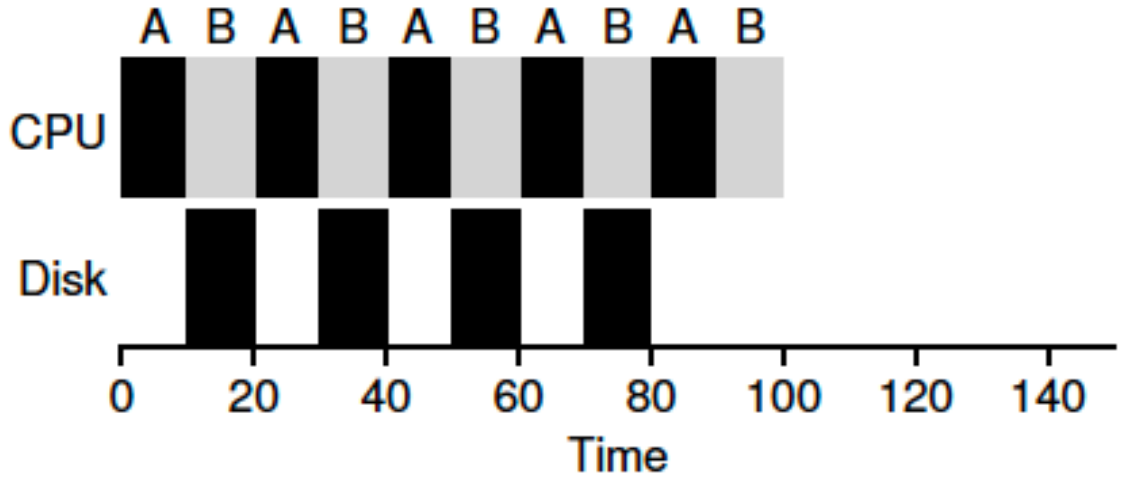

Figure 7.9: **Overlap Allows Better Use Of Resources**

No More Oracle

- Relax Assumption 5
  - OS does not know about the run-time of a job

Multi-Level Feedback Queue

- Optimize both turnaround time and response time
- How can we design a scheduler that both minimizes response time for interactive jobs while also minimizing turnaround time without a priori knowledge of job length?