7. Scheduling: Introduction

Operating System: Three Easy Pieces

Scheduling: Introduction

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

Scheduling Metrics

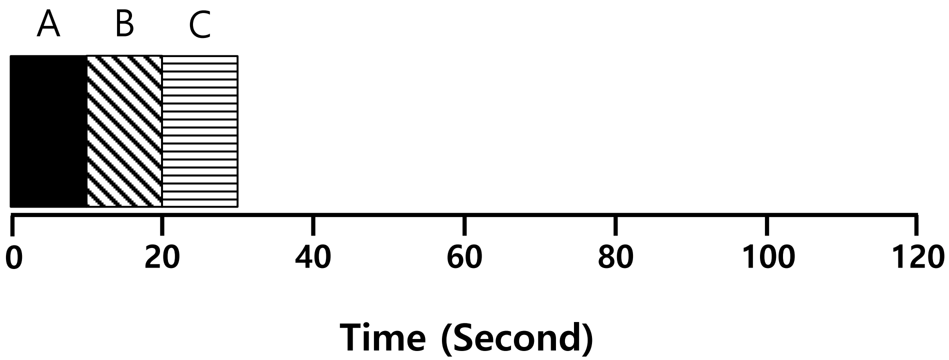
- Performance metric: Turnaround time
  - The time at which **the job completes** minus the time at which **the job arrived** in the system.

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

- Another metric is fairness.
  - Performance and fairness are often at odds in scheduling.

First In, First Out (FIFO)

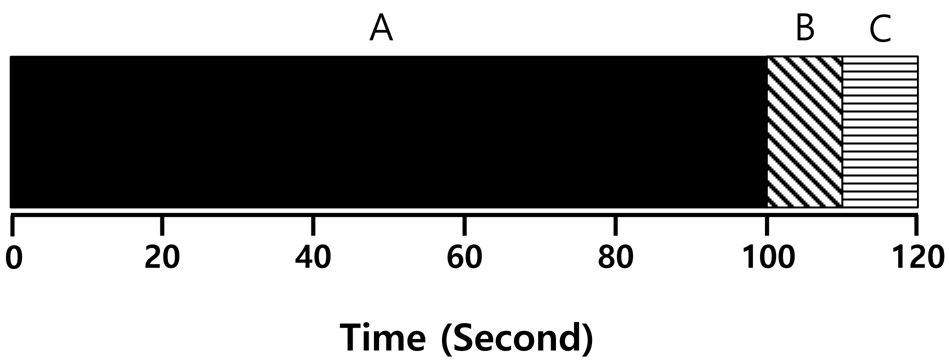
- First Come, First Served (FCFS)
  - Very simple and easy to implement
- Example:
  - A arrived just before B which arrived just before C.
  - Each job runs for 10 seconds.



$$\boldsymbol{Average\ turnaround\ time = \frac{10 + 20 + 30}{3} = 20\ sec}$$

Why FIFO is not that great? – Convoy effect

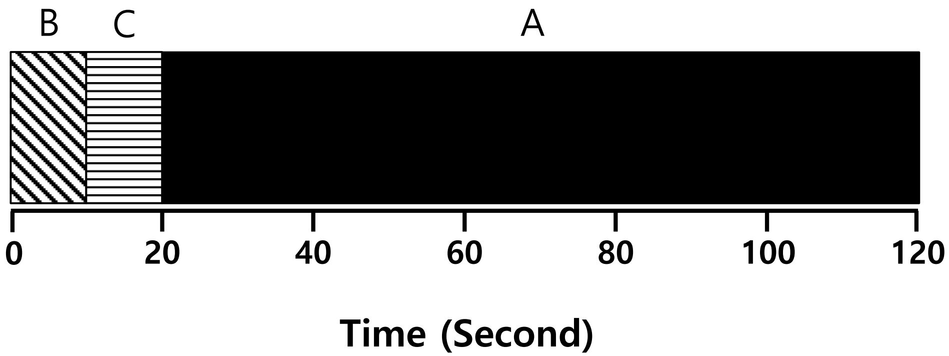
- Let's relax assumption 1: Each job **no longer** runs for the same amount of time.
- Example:
  - A arrived just before B which arrived just before C.
  - A runs for 100 seconds, B and C run for 10 each.



$$\textit{Average turnaround time} = \frac{100 + 110 + 120}{3} = 110\ \textit{sec}$$

Shortest Job First (SJF)

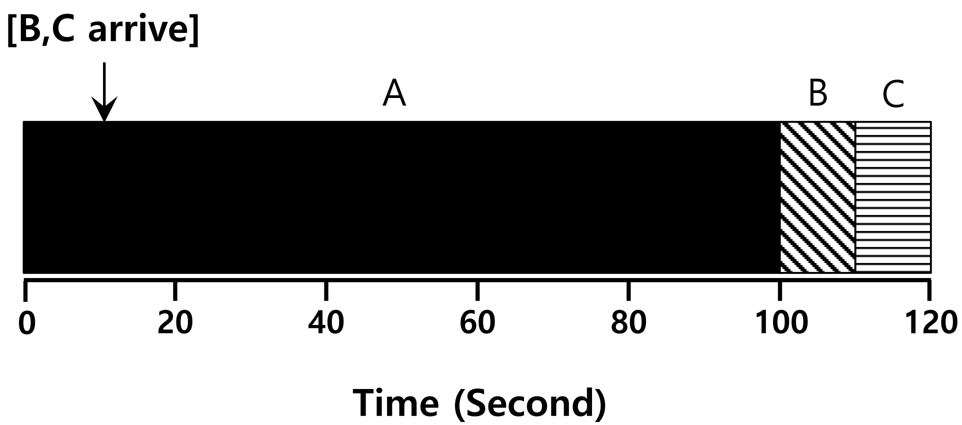
- Run the shortest job first, then the next shortest, and so on
  - Non-preemptive scheduler
- Example:
  - A arrived just before B which arrived just before C.
  - A runs for 100 seconds, B and C run for 10 each.



$$Average\ turnaround\ time = \frac{10 + 20 + 120}{3} = 50\ sec$$

SJF with Late Arrivals from B and C

- Let's relax assumption 2: Jobs can arrive at any time.
- Example:
  - A arrives at t=0 and needs to run for 100 seconds.
  - B and C arrive at t=10 and each need to run for 10 seconds



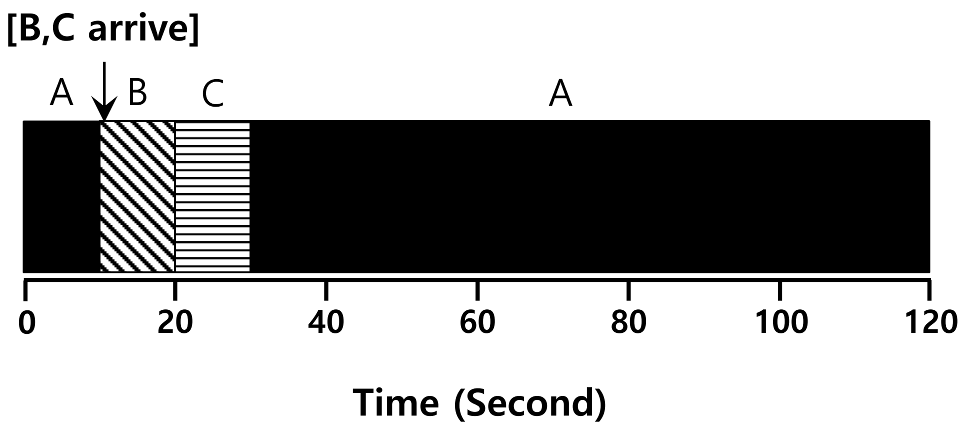
$$Average\ turnaround\ time = \frac{100 + (110 - 10) + (120 - 10)}{3} = 103.33\ sec$$

Shortest Time-to-Completion First (STCF)

- Add preemption to SJF
  - Also knows as Preemptive Shortest Job First (PSJF)
- A new job enters the system:
  - Determine of the remaining jobs and new job
  - Schedule the job which has the lest time left

Shortest Time-to-Completion First (STCF)

- Example:
  - A arrives at t=0 and needs to run for 100 seconds.
  - B and C arrive at t=10 and each need to run for 10 seconds



$$\boldsymbol{Average\ turnaround\ time = \frac{(120-0)+(20-10)+(30-10)}{3} = 50\ sec}$$

New scheduling metric: Response time

- The time from **when the job arrives** to the **first time it is scheduled**.

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

  - STCF and related disciplines are not particularly good for response time.

How can we build a scheduler that is
sensitive to response time?

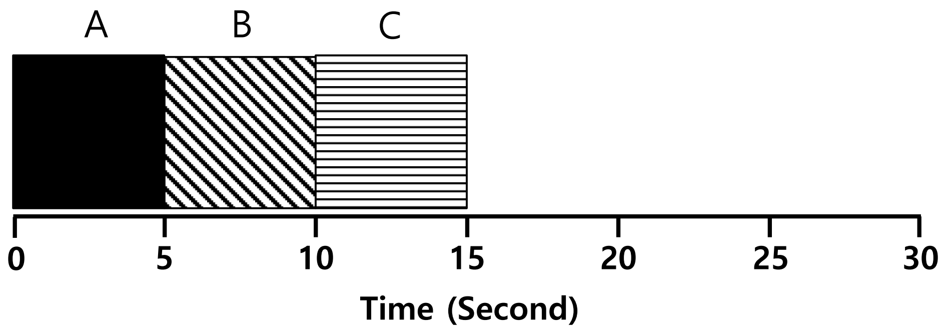
Round Robin (RR) Scheduling

- Time slicing Scheduling
  - Run a job for a time slice and then switch to the next job in the **run queue** until the jobs are finished.
    - Time slice is sometimes called a scheduling quantum.
  - It repeatedly does so until the jobs are finished.
  - The length of a time slice must be *a multiple of* the timer-interrupt period.

RR is fair, but performs poorly on metrics such as turnaround time

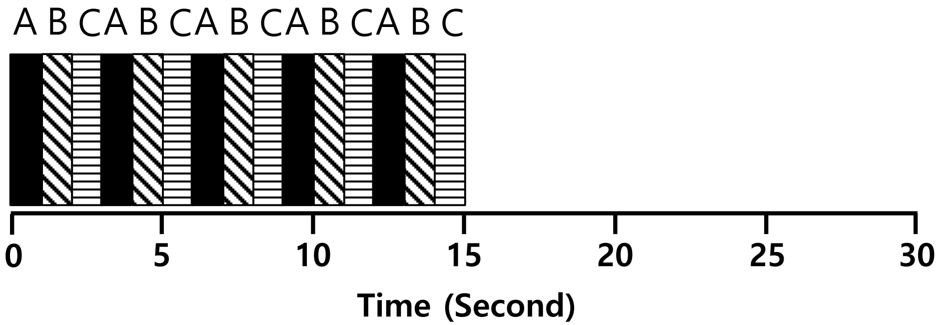
RR Scheduling Example

- A, B and C arrive at the same time.
- They each wish to run for 5 seconds.



SJF (Bad for Response Time)

$$T_{average\ response} = \frac{0 + 5 + 10}{3} = 5sec$$



RR with a time-slice of 1sec (Good for Response Time)

$$T_{average\ response} = \frac{0 + 1 + 2}{3} = 1sec$$

The length of the time slice is critical.

- The shorter time slice
  - Better response time
  - The cost of context switching will dominate overall performance.

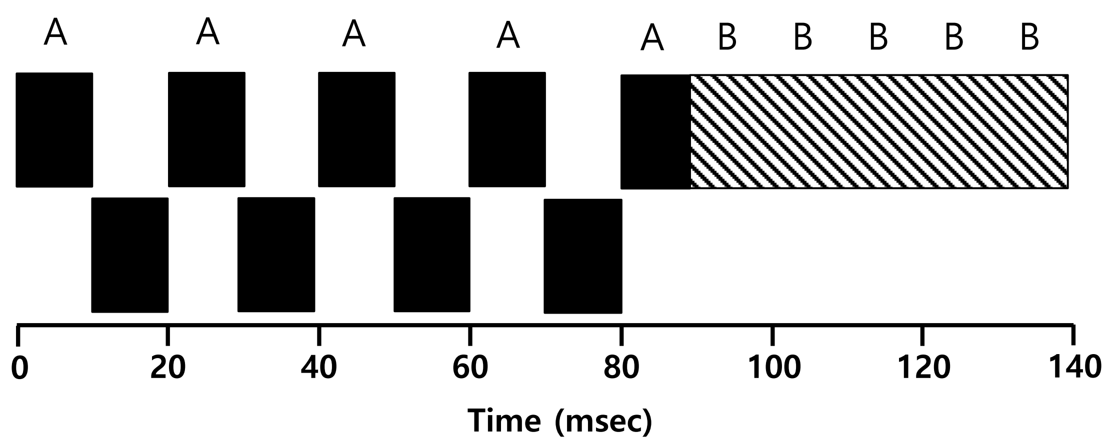
- The longer time slice
  - Amortize the cost of switching
  - Worse response time

Deciding on the length of the time slice presents a trade-off to a system designer

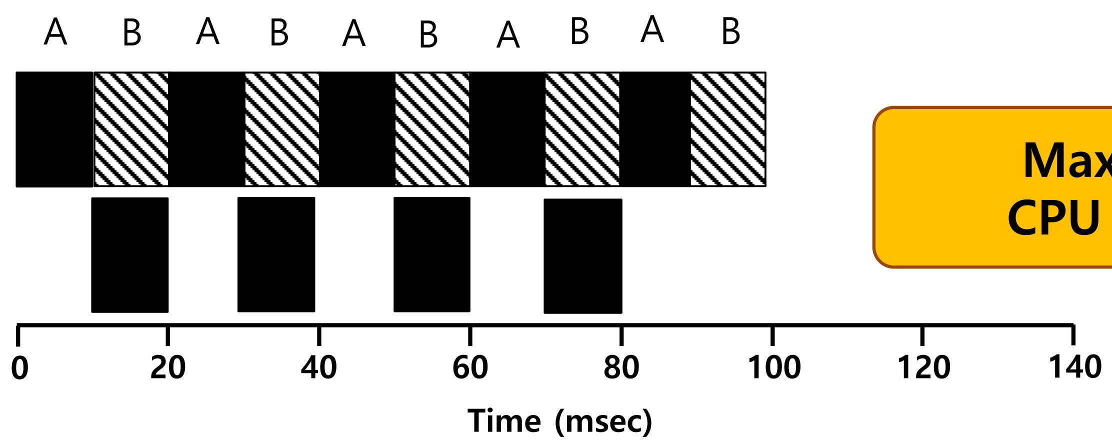
Incorporating I/O

- Let's relax assumption 3: All programs perform I/O
- Example:
  - A and B need 50ms of CPU time each.
  - A runs for 10ms and then issues an I/O request
    - I/Os each take 10ms
  - B simply uses the CPU for 50ms and performs no I/O
  - The scheduler runs A first, then B after

Incorporating I/O (Cont.)



Poor Use of Resources



Overlap Allows Better Use of Resources

Maximize the CPU utilization

Incorporating I/O (Cont.)

- When a job initiates an I/O request.
  - The job is blocked waiting for I/O completion.
  - The scheduler should schedule another job on the CPU.

- When the I/O completes
  - An interrupt is raised.
  - The OS moves the process from blocked back to the ready state.

- Disclaimer: This lecture slide set was initially developed for Operating System course in Computer Science Dept. at Hanyang University. This lecture slide set is for OSTEP book written by Remzi and Andrea at University of Wisconsin.