香港城市大學
City University of Hong Kong

Tutorial 4: CPU Scheduling

CS3103

Operating Systems

專業 創新 胸懷全球
Professional · Creative
For The World

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

Scheduling: Introduction

- Workload assumptions:
  1. Each job runs for the **same amount of time.**
  2. All jobs **arrive** at the same time (Roughly).
  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 Metrics

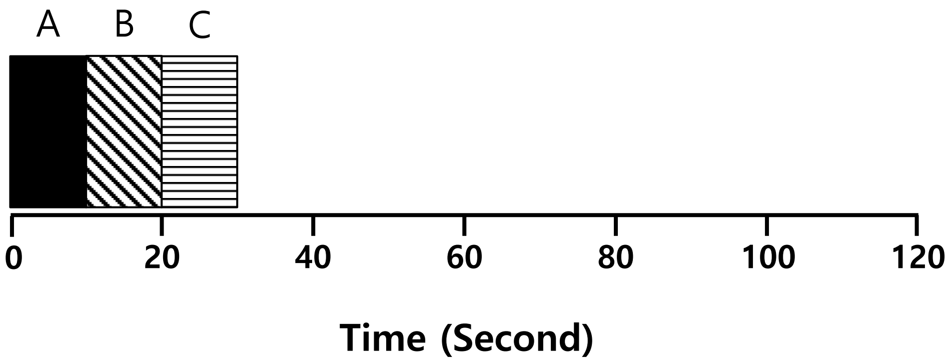
- Performance metric: turnaround time
  - The time at which **the job completes** minus the time at which **the job arrived** in the system.

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

- Another metric is fairness (response time).
  - Performance and fairness are often at odds in scheduling.

First In, First Out (FIFO)

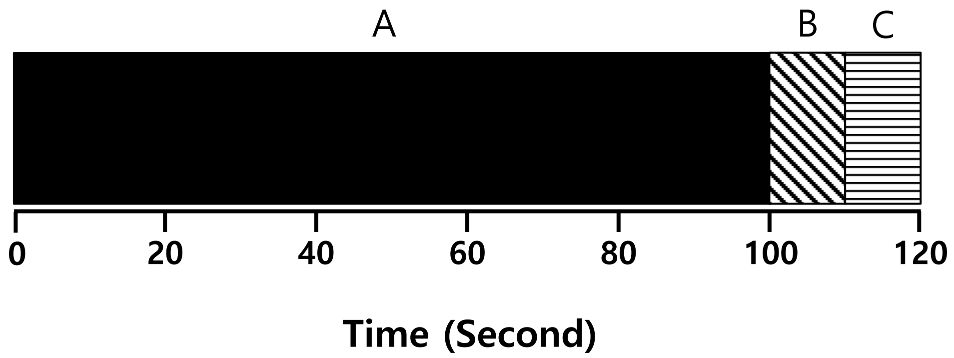
- First Come, First Served (FCFS)
  - Very simple and easy to implement
- Example:
  - A arrived just before B which arrived just before C.
  - Each job runs for 10 seconds.



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

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.

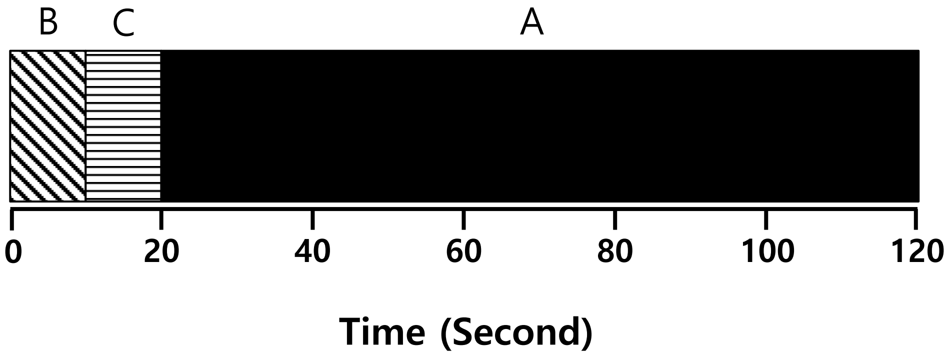


bloated !

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

Shortest Job First (SJF)

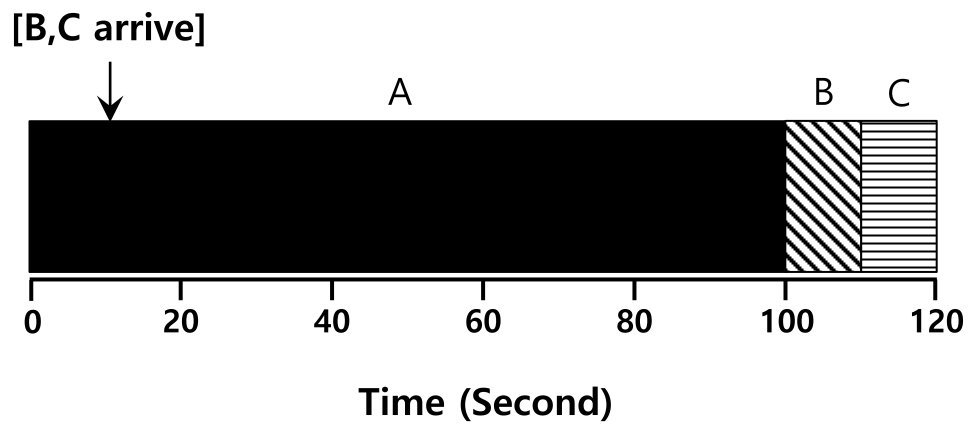
- Run the shortest job first, then the next shortest, and so on
  - Non-preemptive scheduler
- Example:
  - A arrived just before B which arrived just before C.
  - A runs for 100 seconds, B and C run for 10 each.



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

SJF with Late Arrivals from B and C (without preemptive scheduler)

- 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



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

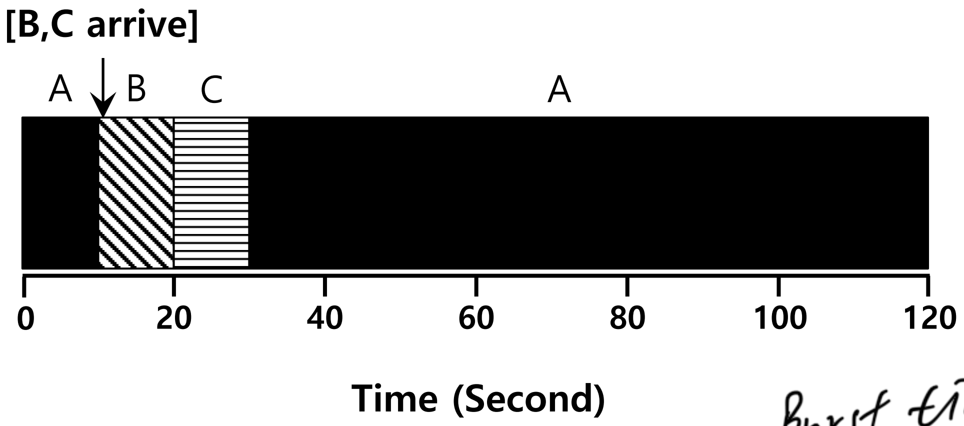
Shortest Time-to-Completion First (STCF)

- Let's relax assumption 3: A job can preempt other jobs.
- 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 least 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

A → 90s → B, C First → Remaining A



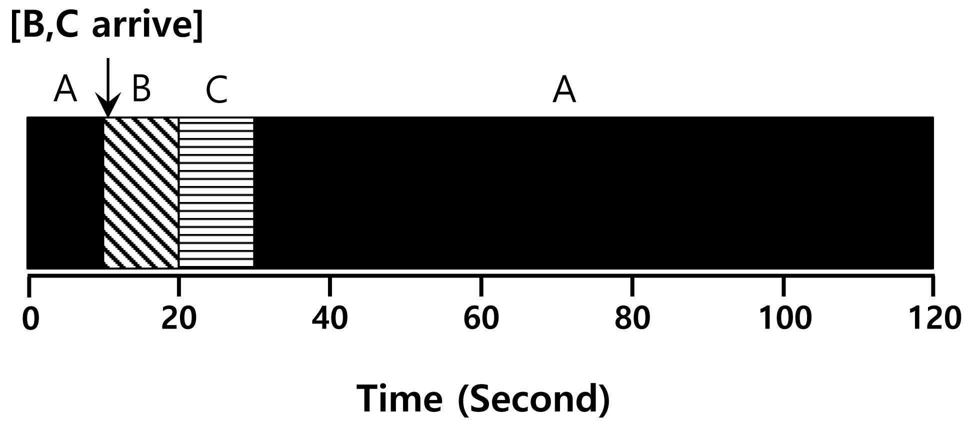
Burst time

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

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 scheduling policies are not particularly good for response time.

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

Round Robin (RR) Scheduling

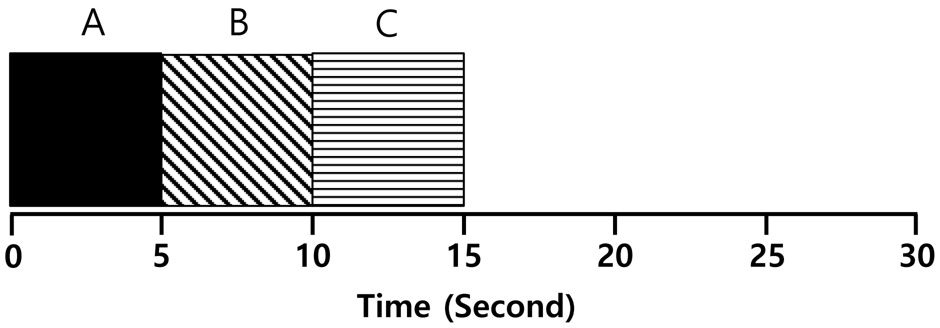
- Time slicing Scheduling
  - Run a job for a time slice and then switch to the next job in the **run queue** until the jobs are finished.
    - Time slice is sometimes called a scheduling quantum.
  - It repeatedly does so until the jobs are finished.
  - The length of a time slice must be *a multiple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

fairness ✓ performance ↓

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

RR Scheduling Example

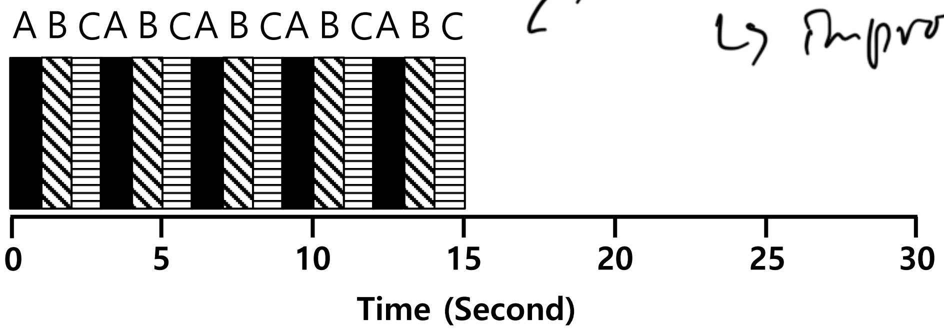
- A, B and C arrive at the same time.
- They each wish to run for 5 seconds.



SJF (Bad for Response Time)

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

→ time slice ≈> 0,5s
↳ Improved!



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

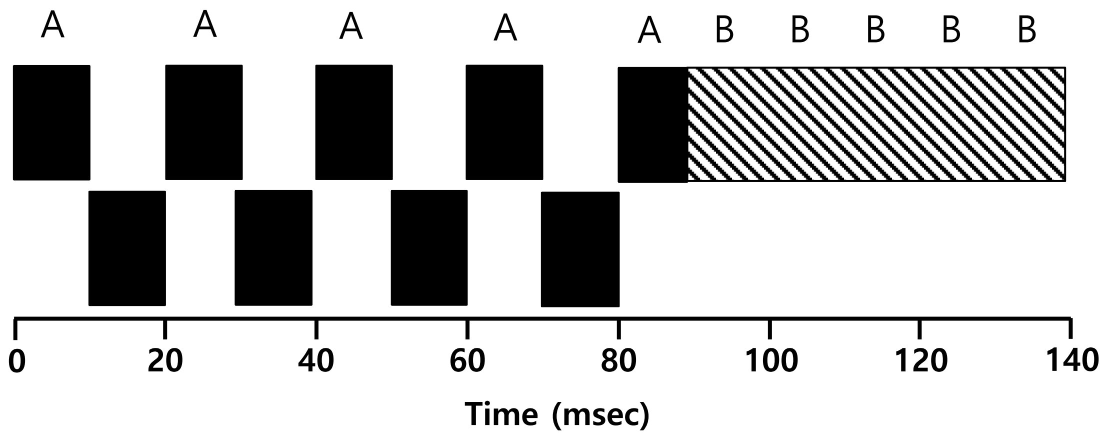
Incorporating I/O

- Let’s relax assumption 4: All programs perform I/O
- 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.

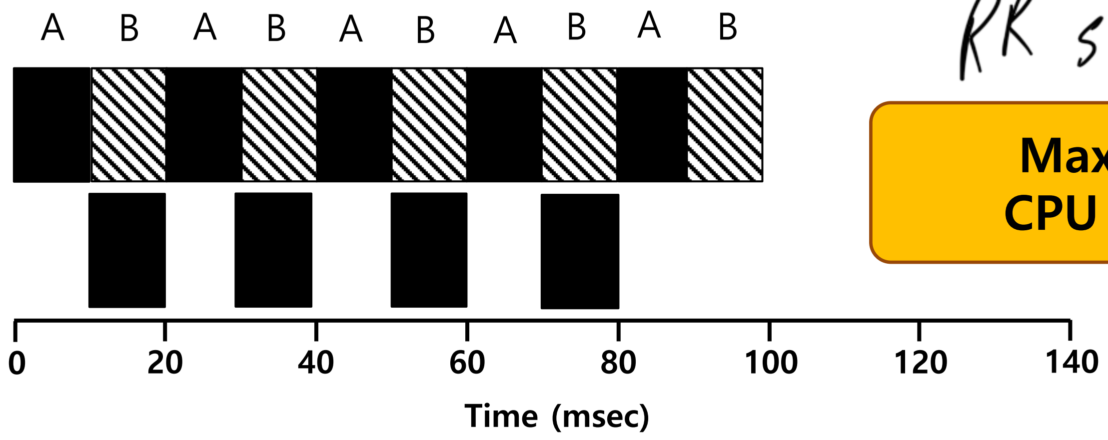
Incorporating I/O (Cont.)

- 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

RR scheduling

Maximize the CPU utilization

Scheduling without *a priori* knowledge

- Let’s relax assumption 5: the OS usually knows very little about the length of each job.
  - Use the recent past to predict the future
  - multi-level feedback queue (covered in the lecture).