

# CSL 301

## OPERATING SYSTEMS



### Lecture 4

#### Scheduling

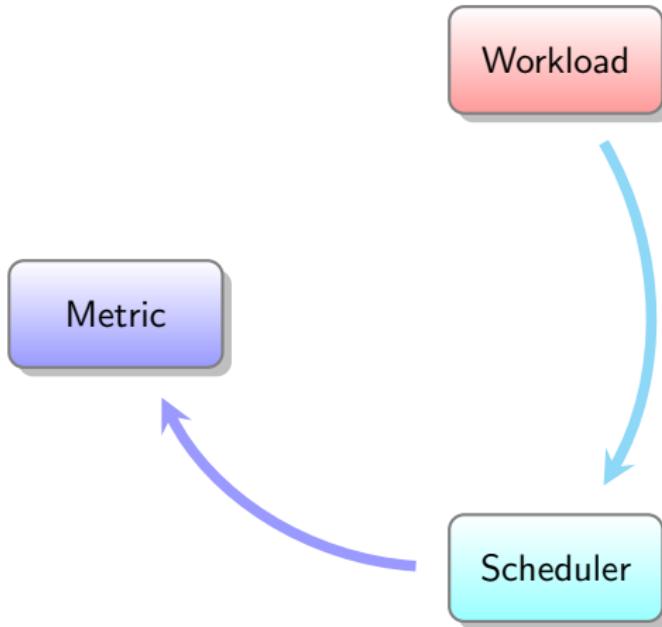
Instructor  
Dr. Dhiman Saha

# Processes running in the system

Workload

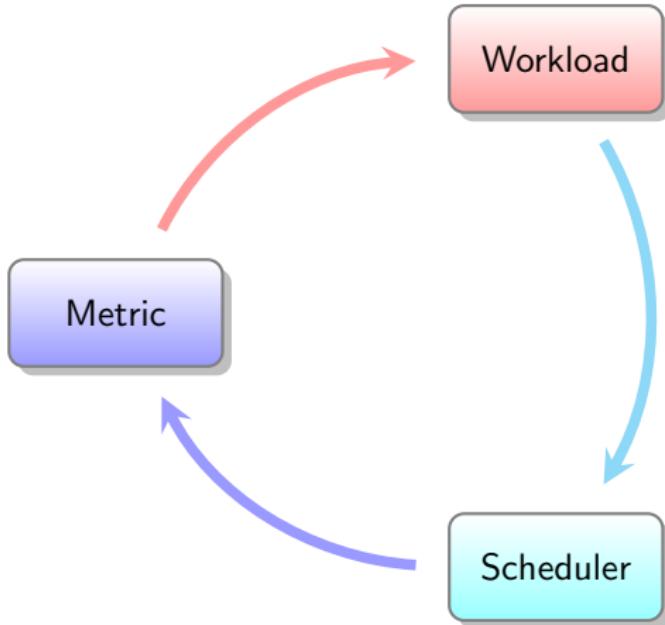


# Measuring the scheduling “quality”



# Intuition

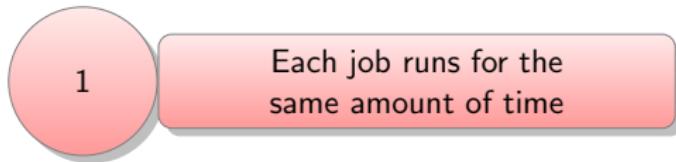
Given two of these find the third



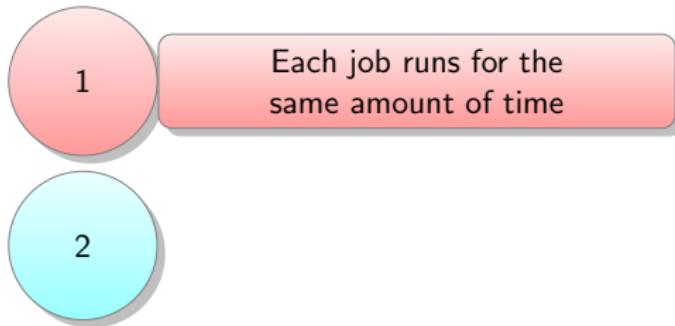
# Workload Assumptions



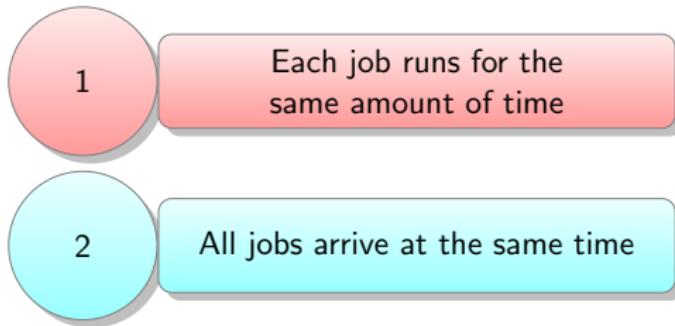
# Workload Assumptions



# Workload Assumptions



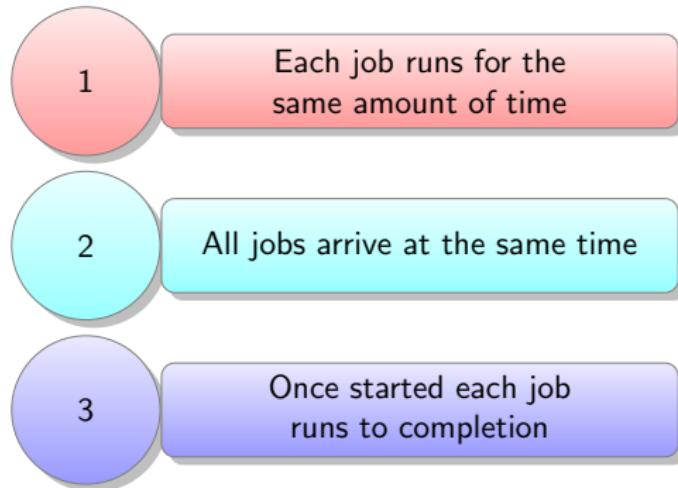
# Workload Assumptions

- 
- 1      Each job runs for the same amount of time
  - 2      All jobs arrive at the same time

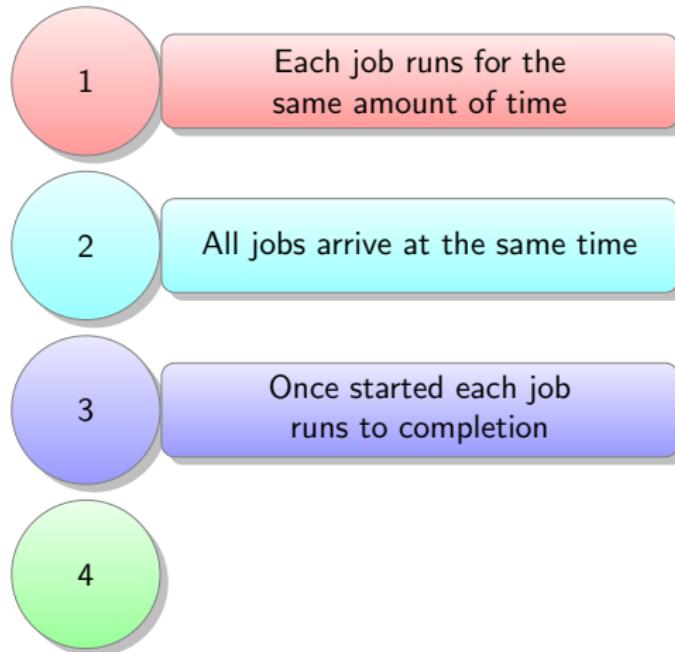
# Workload Assumptions

- 
- 1      Each job runs for the same amount of time
  - 2      All jobs arrive at the same time
  - 3

# 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

# 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

# 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 (no I/O)

# Workload Assumptions

- 
- The diagram consists of five horizontal rows. Each row contains a colored circle on the left and a text box on the right. The circles are arranged vertically from top to bottom, corresponding to the numbers 1 through 5. The colors of the circles are red, cyan, purple, green, and orange respectively. The text boxes contain the following 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 (no I/O)
  - 5: (No text provided)

# 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 (no I/O)
  - 5 The run-time of each job is known

# 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 (no I/O)
  - 5 The run-time of each job is known

## turnaround time

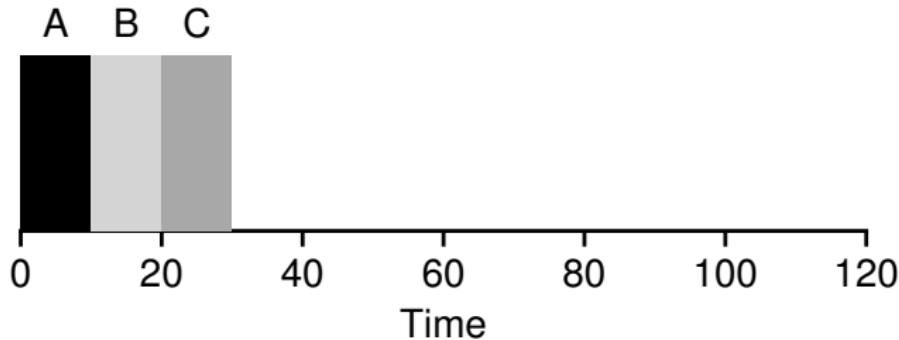
The turnaround time of a job is defined as the **time at which the job completes** minus **the time at which the job arrived** in the system.

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

First In, First Out  
First Come, First Served

FIFO / FCFS

- ▶ A, B, and C, arrive at roughly the same time
- ▶  $T_{arrival} = 0$



## Classwork

What will the **average turnaround time** be for these jobs?

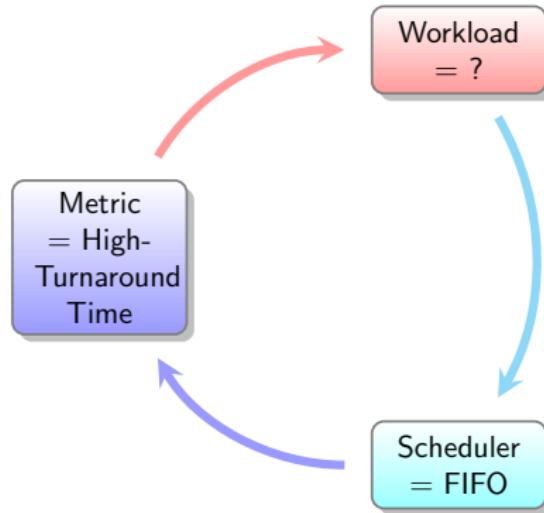
- 
- 1 Each job runs for the same amount of time
  - 2 All jobs arrive at the same time
  - 3 Once started each job runs to completion
  - 4 All jobs only use the CPU (no I/O)
  - 5 The run-time of each job is known

# How does FIFO perform now?

CW-1

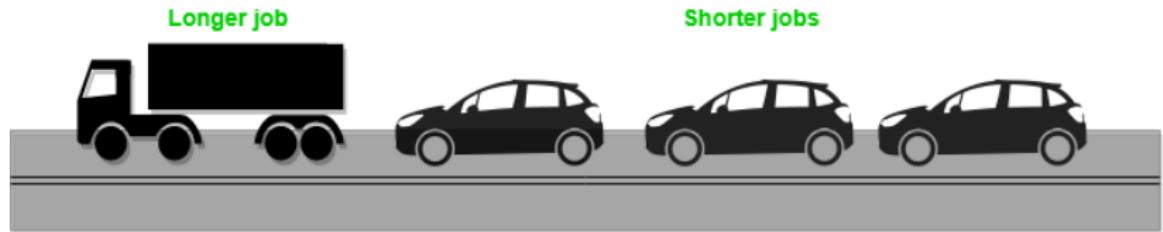
Classwork

Construct a **workload** to make FIFO perform poorly



average turnaround time?

# The Convoy Effect



## New Scheduler

How can we develop a better algorithm to deal with our new reality of jobs that run for different amounts of time?

## Shortest Job First

SJF

## Simple Approach

- ▶ Run the shortest job first, then the next shortest, and so on.
  - ▶ Idea borrowed from operations research
- 
- ▶ Redo the last FIFO workload with SJF
  - ▶ What is the turn around time now?
  - ▶ What is the improvement?

## Point-to-Ponder

Is SJF an **optimal** scheduling algorithm? How?

- 
- The diagram consists of five colored circles numbered 1 through 5, each paired with a text box containing a specific workload assumption. Circle 1 is pink, circle 2 is light blue, circle 3 is light purple, circle 4 is light green, and circle 5 is light orange.
- 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 (no I/O)
  - 5 The run-time of each job is known

# How does SJF perform now?

CW-2

Classwork

Construct a **workload** to make SJF perform poorly

Need for a new scheduler

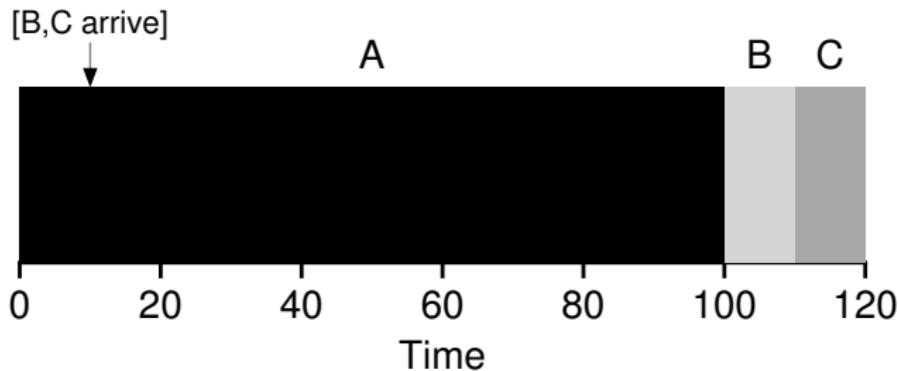
- ▶ Again what is the average turnaround time now?

# How does SJF perform now?

CW-2

Classwork

Construct a **workload** to make SJF perform poorly



The Convoy Effect

Need for a new scheduler

- ▶ Again what is the average turnaround time now?

- 
- 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 (no I/O)
- 5 The run-time of each job is known

# Preemptive Scheduling

The scheduler can perform a **context switch**, stopping one running process temporarily and resuming (or starting) another.

- ▶ Using the mechanisms we discussed earlier
- ▶ The timer interrupt
- ▶ And context switching

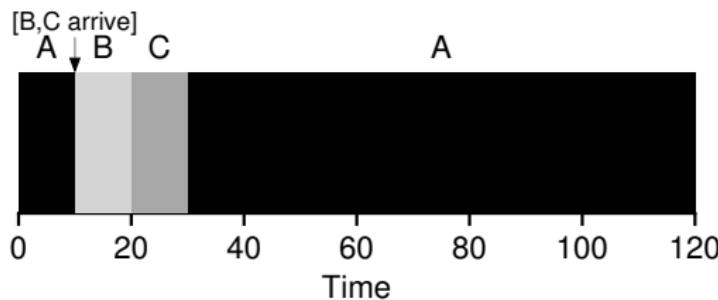
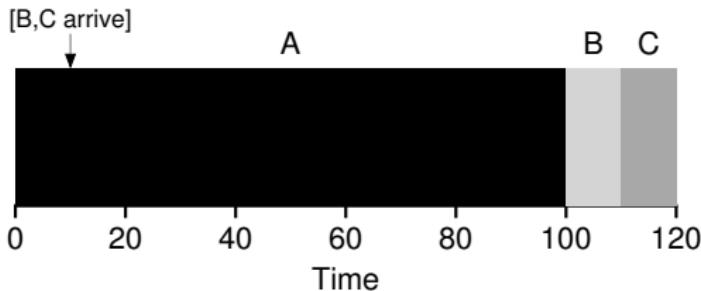
Shortest Time-to-Completion First  
Preemptive Shortest Job First  
Shortest Remaining Time First

STCF/PSJF/SRTF

# SJF

# Vs

# Preemptive SJF



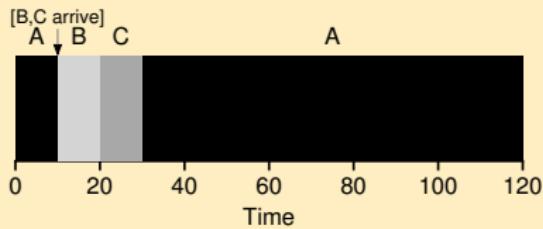
Compare avg. turnaround time

- ▶ Sometimes we care about when a job starts
- ▶ Instead of when it finishes

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

- ▶ What does  $T_{response}$  capture?

CW-3

Compute Avg.  $T_{response}$  for Preemptive SJF

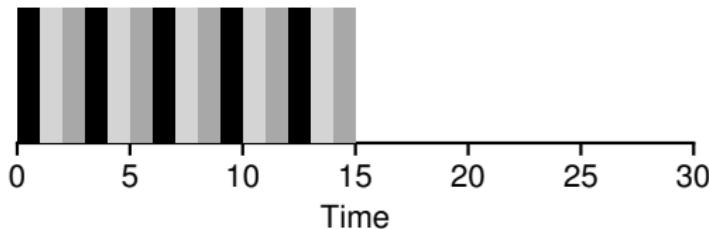
- ▶ How can we build a scheduler that is sensitive to response time?

# Round Robin

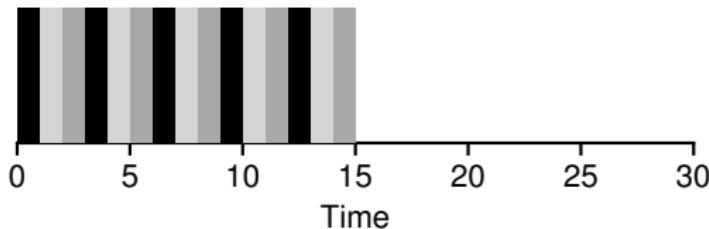
Time-slicing

RR

- ▶ Instead of running jobs to completion, RR runs a job for a time slice (sometimes called a **scheduling quantum**) and
- ▶ Then switches to the next job in the run queue.
- ▶ It repeatedly does so until the jobs are finished.



- ▶ Instead of running jobs to completion, RR runs a job for a time slice (sometimes called a **scheduling quantum**) and
- ▶ Then switches to the next job in the run queue.
- ▶ It repeatedly does so until the jobs are finished.



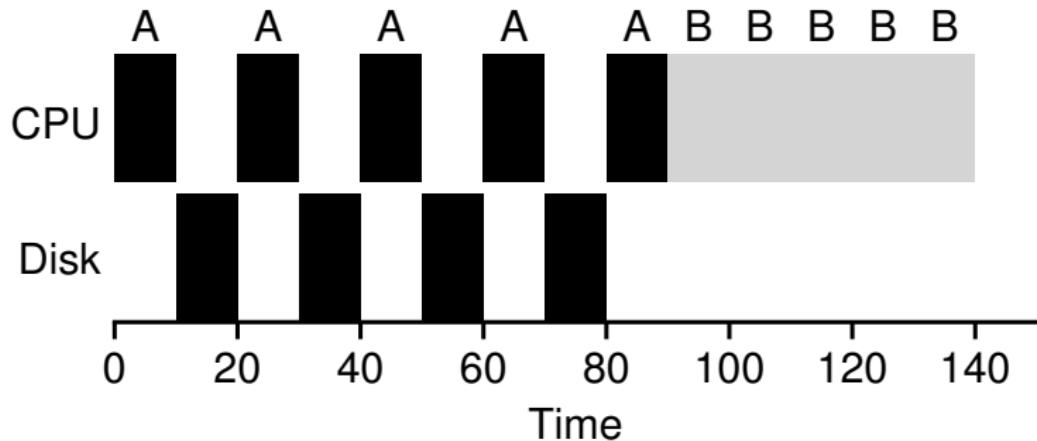
- ▶ Avg. Response Time?
- ▶ Compare with FIFO
- ▶ What about avg. turn-around time?
- ▶ How to choose the time-slice?

## Relaxing #4

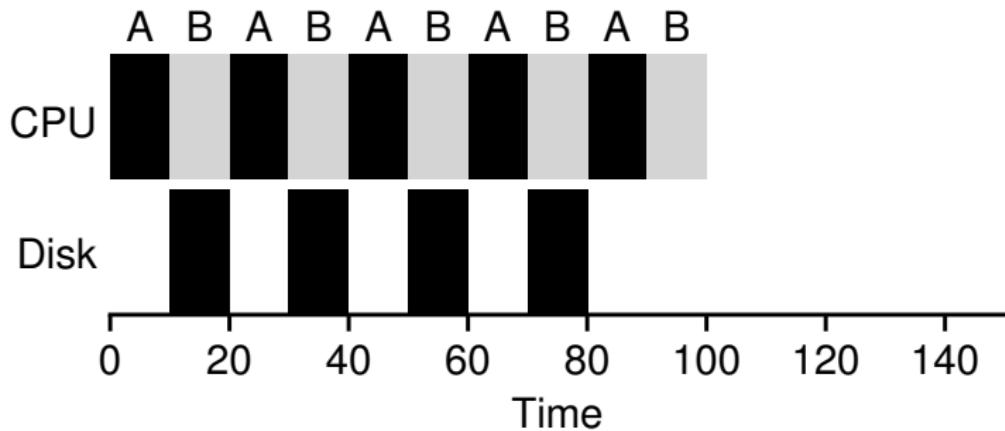
## Workload Assumptions

- 
- 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 (no I/O)
- 5 The run-time of each job is known

## Not I/O Aware



## I/O Aware (Overlap)



## Relaxing #5

## Workload Assumptions



- ▶ Smarter Scheduling
- ▶ Multi-level feedback queue