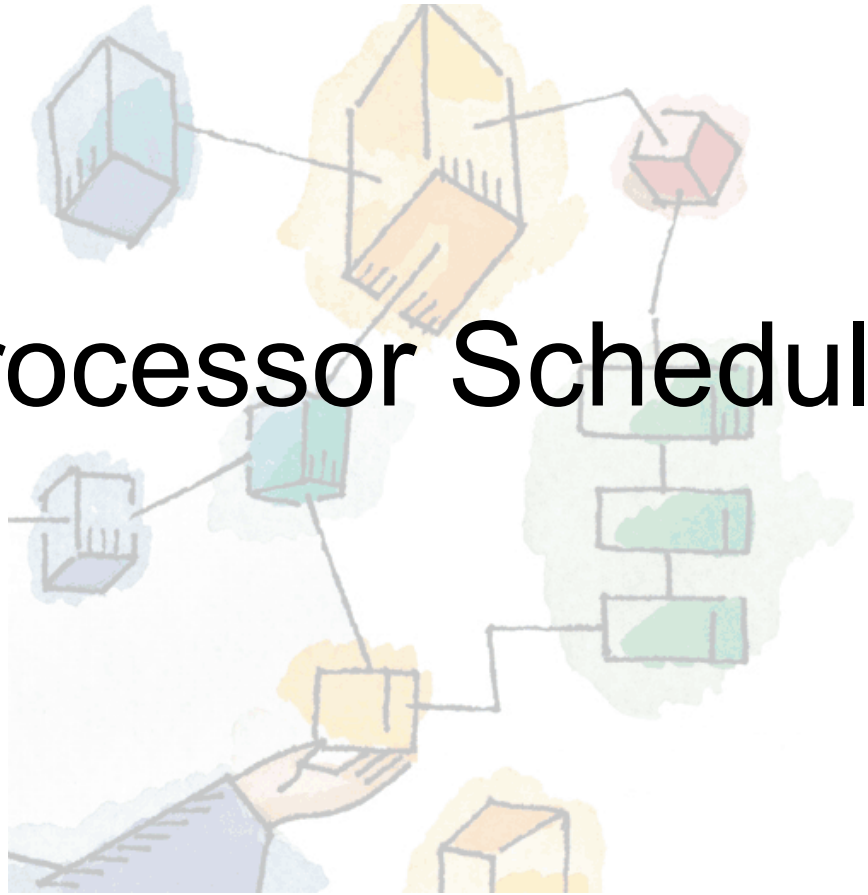
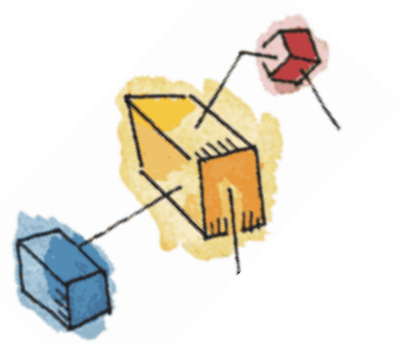
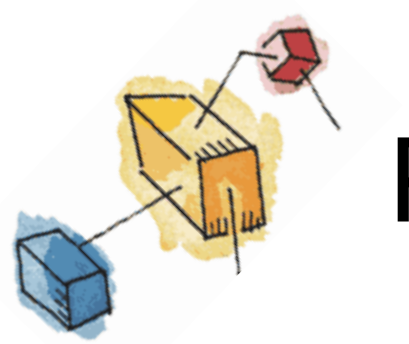


Processor Scheduling



Outline

- Types of scheduling
  - Long term, medium term and short term
- Scheduling criteria
  - Turnaround time
  - Response time
- Scheduling policies/algorithms
  - Preemptive and non-preemptive
  - FCFS
  - SJF
  - SRT
  - RR
  - MFBQ



Processor Scheduling

- The resource provided by a processor is execution time
  - The resource is allocated by means of a schedule
- Aim is to assign processes to be executed by the processor in a way that meets system objectives, such as turnround time, response time, and processor efficiency

Types of Scheduling

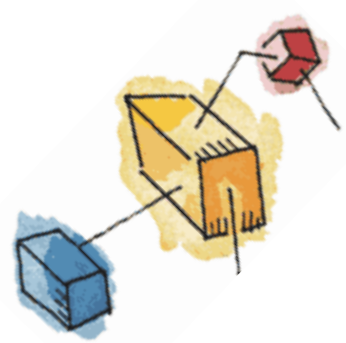
- Broken down into three separate functions:

Long-Term Scheduling

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
  - More processes, smaller percentage of time each process is executed
  - may limit to provide satisfactory service to the current set of processes

Medium-Term Scheduling

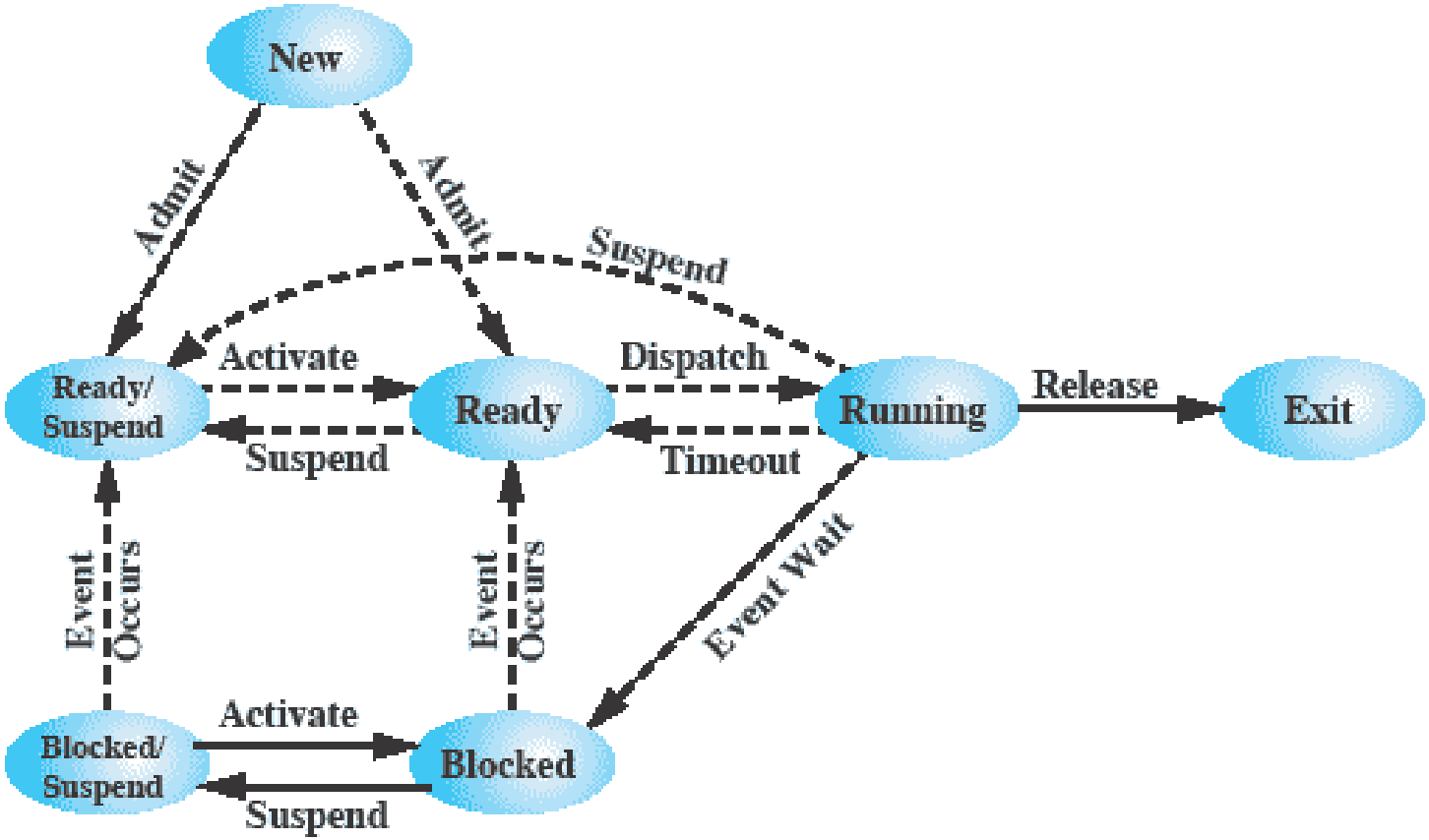
- Part of the swapping function
- Swapping-in decisions are based on the need to manage the degree of multiprogramming
- considers the memory requirements of the swapped-out processes



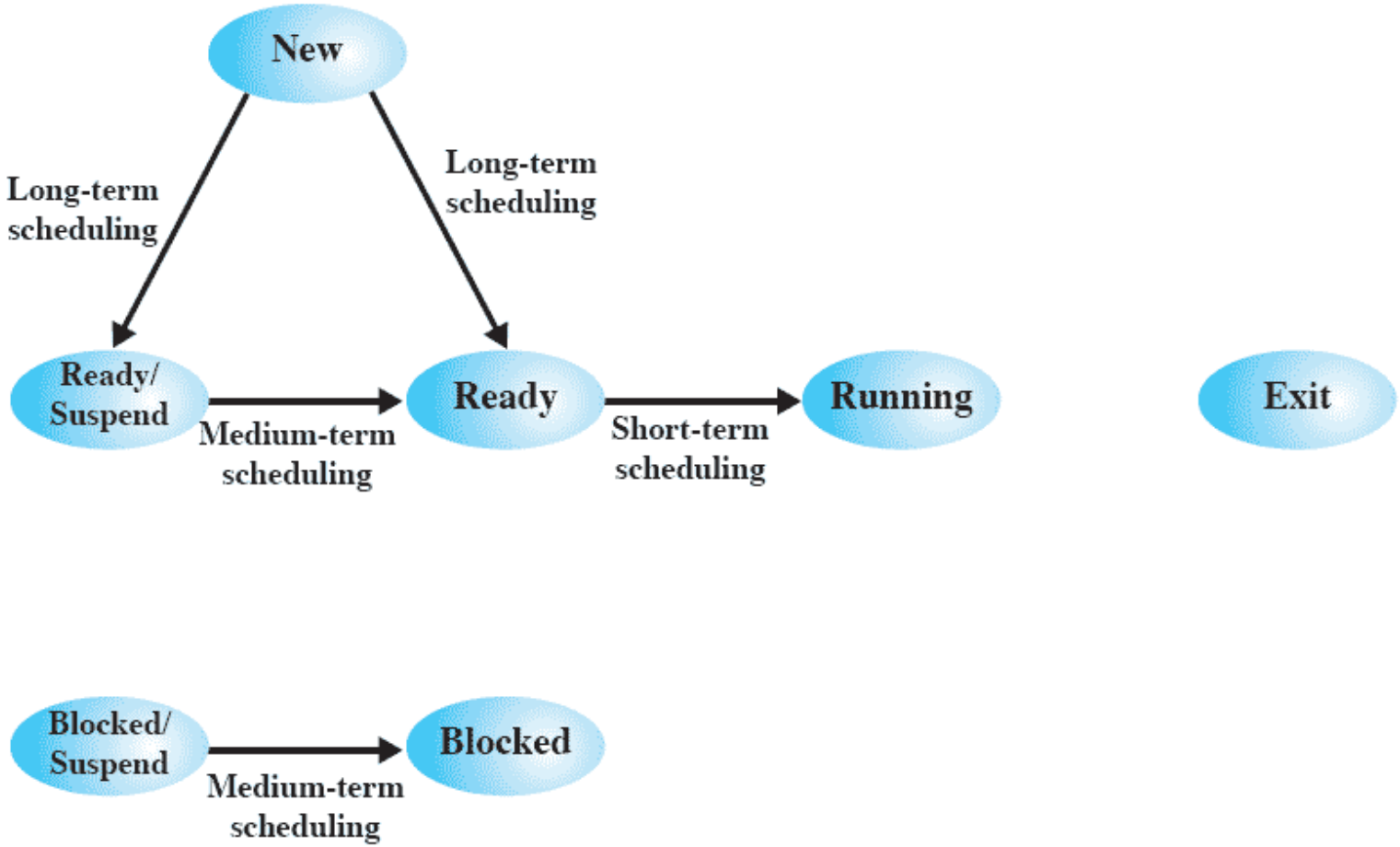
Short-Term Scheduling

- Known as the dispatcher
- Executes most frequently
- Invoked when an event occurs
  - Clock interrupts
  - I/O interrupts
  - Operating system calls
  - Signals

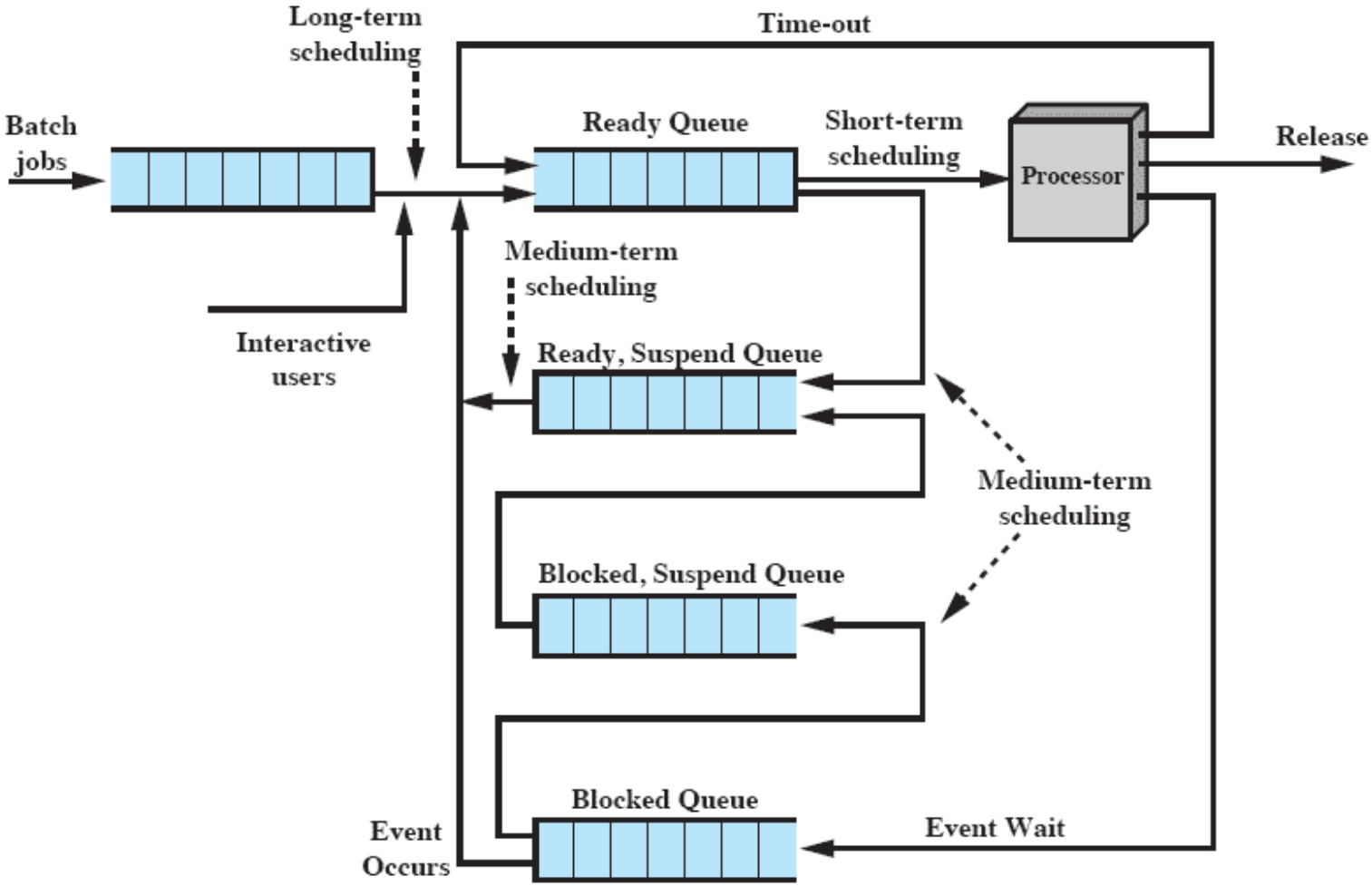
Two Suspend States

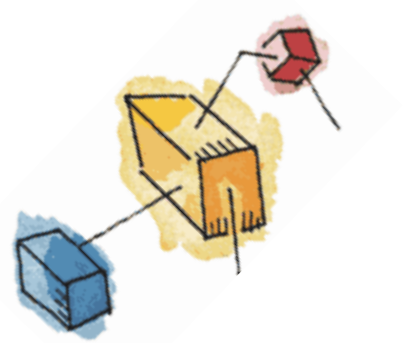


Scheduling and Process State Transitions



Queuing Diagram





Aim of Short Term Scheduling

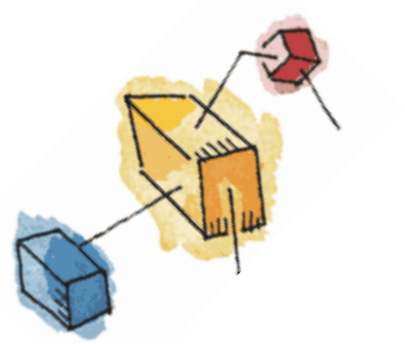
- Main objective is to allocate processor time to optimize certain aspects of system behaviour
- A set of criteria is needed to evaluate the scheduling policy
  - e.g., turnaround time, response time, deadlines, fairness

Process Scheduling Example

- Assume a set of processes

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

- Service time is the required total execution time
- We want to calculate turnaround time & response time



Turnaround Time

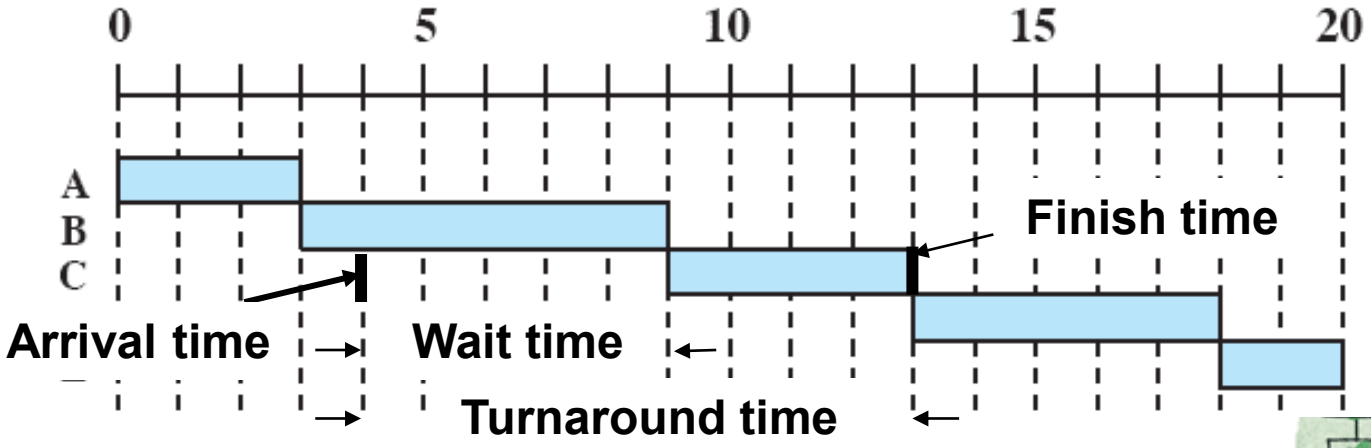
- Turnaround_time = completion_time – arrival_time
  - Including both execution_time and wait_time
  - Wait_time = turnaround_time – execution_time
- Example:
  - A process's execution time is 15
  - It arrives at time 10 and completed execution at time 30
  - Then the turnaround_time = 30 – 10 = 20
  - Wait_time = 20 – 15 = 5

First-Come-First-Served

- Each process joins the Ready queue in a first-in-first-out manner
- When the current process ceases to execute, the process waiting the longest time in the Ready queue is selected

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

First-Come-First Served (FCFS)



Turnaround time = Finish time – Arrival time
Wait time = Turnaround time - Service time

First-Come-First-Served

- Process A:
  - Turnaround_time = 3 – 0 = 3
  - Wait_time = 3 – 3 = 0
- Process B:
  - Turnaround_time = 9 – 2 = 7
  - Wait_time = 7 – 6 = 1
- Process C:
  - Turnaround_time = 13 – 4 = 9
  - Wait_time = 9 – 4 = 5
- Process D:
  - Turnaround_time = 18 – 6 = 12
  - Wait_time = 12 – 5 = 7
- Process E:
  - Turnaround_time = 20 – 8 = 12
  - Wait_time = 12 – 2 = 10

Average turnaround_time = (3+7+9+12+12)/5 = 8.6
Average wait_time = (0+1+5+7+10)/5 = 4.6

First-Come-First-Served

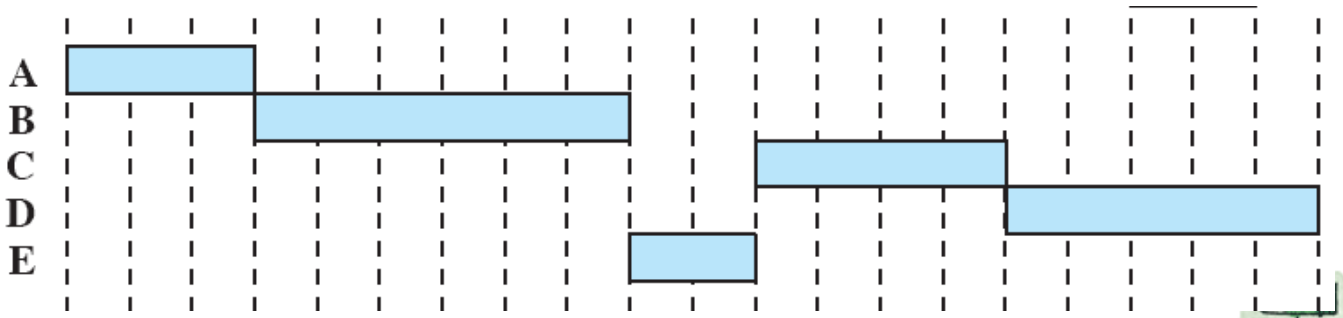
- The simplest scheduling policy
- Issues:
  - A short process may have to wait a very long time before it can execute
  - Favors CPU-bound processes
    - I/O-bound processes have to wait until CPU-bound process completes

Shortest Job First (SJF)

- or Shortest Process Next (SPN)
- Process with shortest expected processing time is selected next
- Short process jumps ahead of longer processes

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Shortest Job First (SJF)



Shortest Job First (SJF)

- Average turnaround_time = 7.6
- Average wait_time = 3.6

- Favors short jobs
  - shorter turnaround and wait time
- Issues:
  - Predictability of longer processes is reduced
  - Possibility of starvation for longer processes

Non-preemptive vs Preemptive

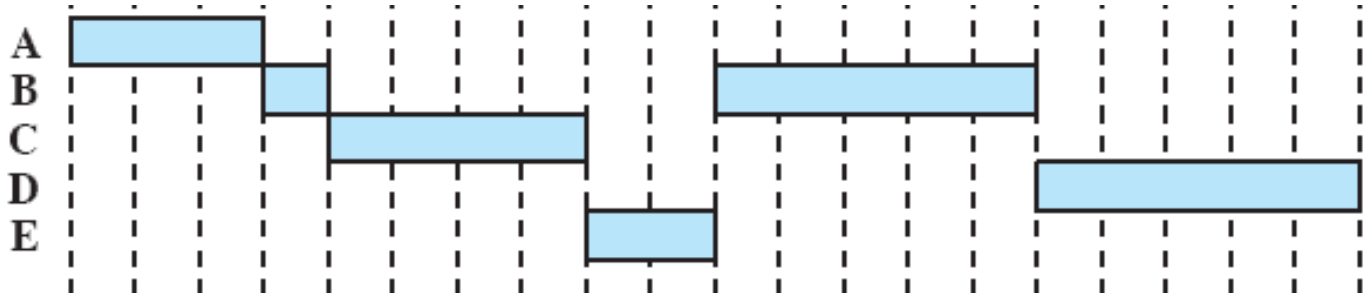
- Non-preemptive
  - Once a process is in the running state, it will continue until it terminates or blocks itself for I/O
  - FCFS and SJF are non-preemptive policies
  - Even using SJF short jobs still need to wait for a long time if a very long job has been scheduled and running
- Preemptive
  - Currently running process may be interrupted and moved to ready state by the OS

Shortest Remaining Time (SRT)

- Preemptive version of shortest job first policy
- Scheduler always chooses the process that has the shortest expected remaining processing time

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Shortest Remaining Time (SRT)



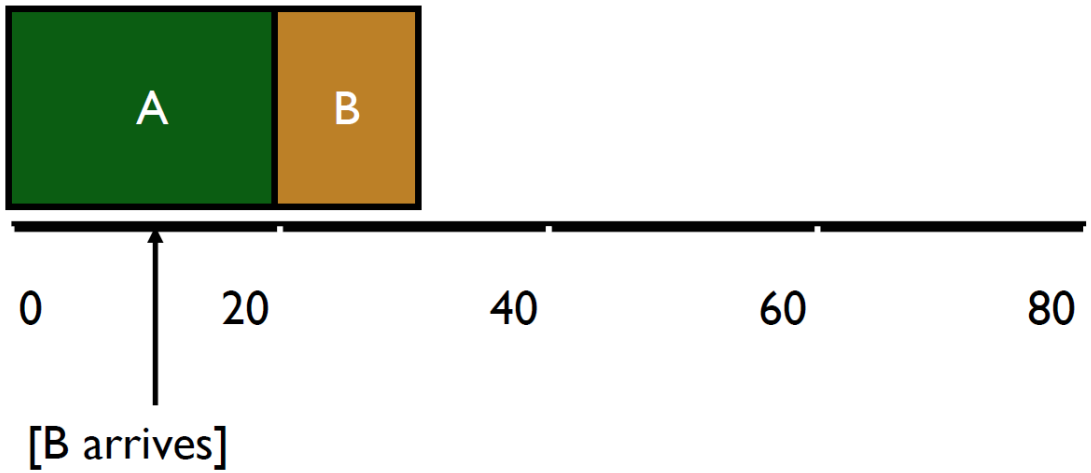
Shortest Remaining Time (SRT)

- Average turnaround_time = ?
- Average wait_time = ?

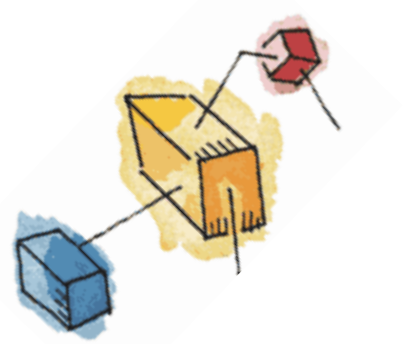
- Should give superior turnaround time performance to SJF because a short job is given immediate preference to a running longer job
- However, risk of starvation of longer processes
- Both SJF and SRT policies
  - must estimate processing time and choose the shortest
    - Question: how?
  - Possibility of starvation for longer processes

Response Time

- Response_time = first_run_time – arrival_time

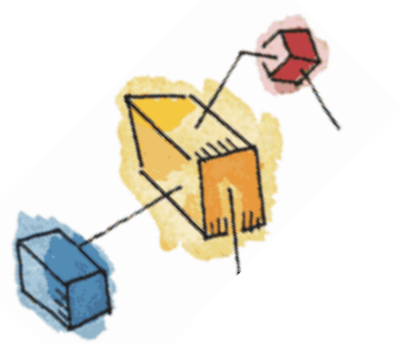


- B's turnaround_time = 20
- B's response_time = 10



Round Robin (RR)

- Clock interrupt is generated at periodic intervals
- When an interrupt occurs, the currently running process is placed in the ready queue
  - Next ready job is selected

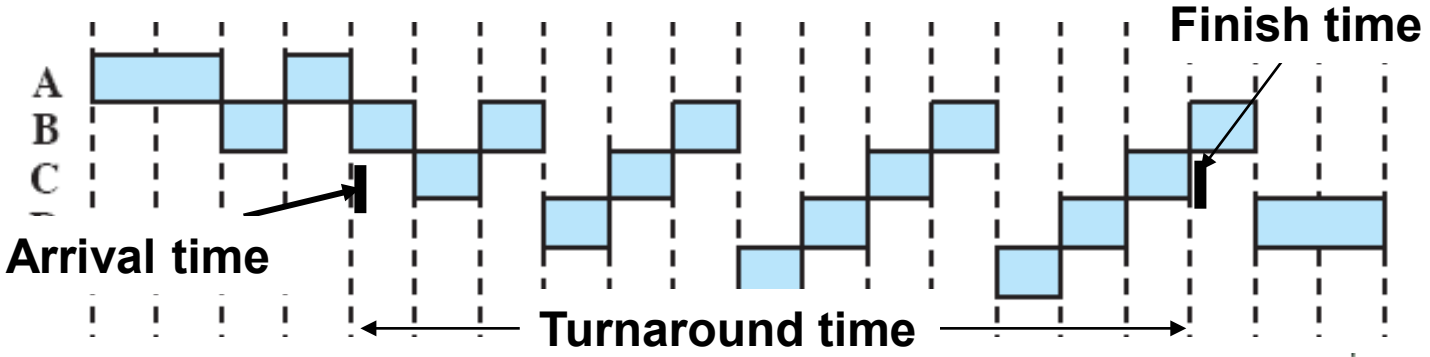


Round Robin (RR)

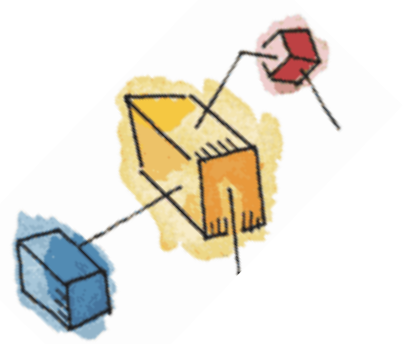
- Uses preemption based on a clock
  - also known as time slicing, because each process is given a slice of time before being preempted.

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Round-Robin (RR), $q = 1$



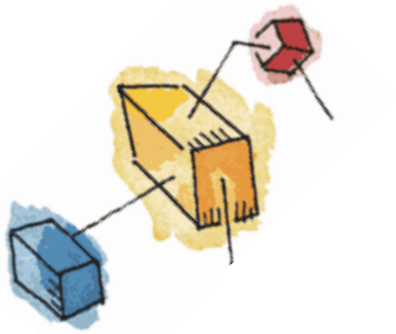
Wait time = ?



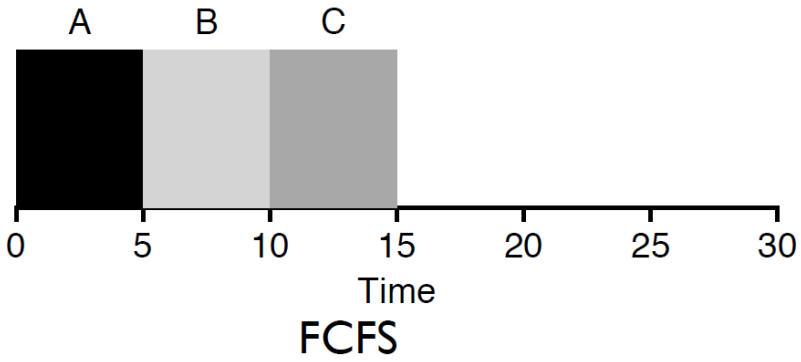
Round Robin (RR)

- Round robin scheduling policy
  - Decreases response time
  - But will significantly increase turnaround time when multiple long jobs are running simultaneously

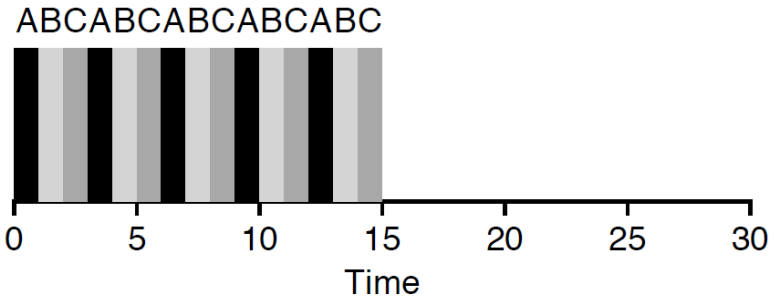
- Tuning challenges:
  - What is a good time slice for round robin?
  - What is the overhead of context switching?



Round Robin (RR)



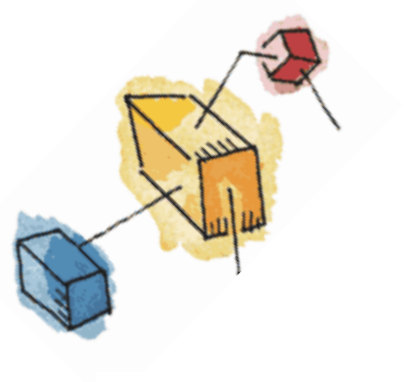
FCFS



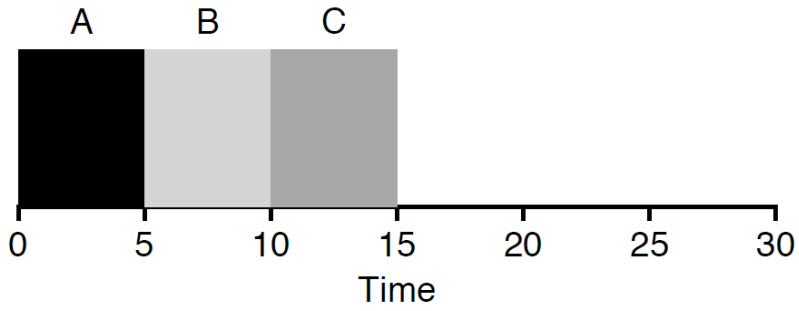
Round Robin every 1s

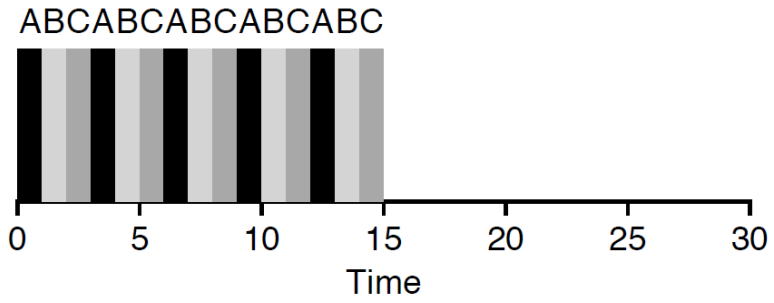
Average Response Time?

Average Turnaround Time?



Round Robin (RR)





Average Response Time

(0 + 5 + 10)/3 = 5s

(0 + 1 + 2)/3 = 1s

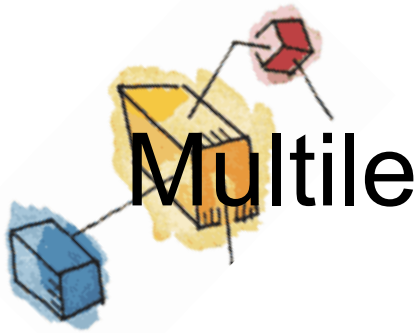
Average Turnaround Time

(5 + 10 + 15)/3 = 10s

(13 + 14 + 15)/3 = 14s

Next Week's Quiz

- **Next week's quiz:**
- Assume three processes A, B and C
  - Arrived at the same time (say at time 0)
  - A's execution time is 20
  - B's execution time is 1.2
  - C's execution time is 1
- A starts first, then B and C, calculating each process's turnaround time and wait time, and then average turnaround time and average wait time When FCFS is used
- When RR is used (with time slice = 1), calculate turnaround times, wait times and averages
- Compare and discuss the results
- What conclusions can you make from the results?

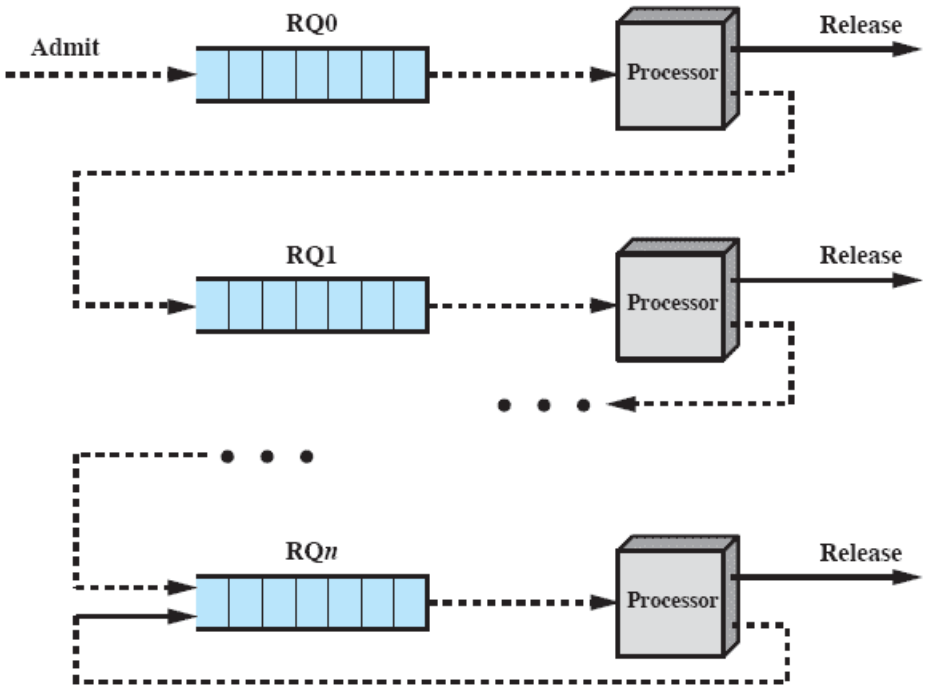


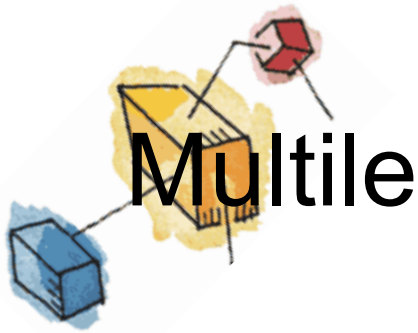
Multilevel Feed Back Queue (MFBQ)

- To optimize turnaround time
  - Running shorter jobs first
    - Need a priori knowledge of each job's length – How?
- To minimize response time
  - Round robin
    - Sometimes terrible for turnaround time, especially when multiple long jobs run simultaneously
- How can we design a scheduler that minimizes response time for interactive jobs while also minimizing turnaround time without a priori knowledge of job length?

Multilevel Feed Back Queue (MFBQ)

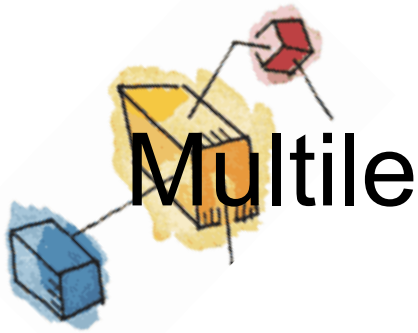
- A number of distinct queues
- Each assigned a different priority level
- At any given time a job that is ready to run is on a single queue





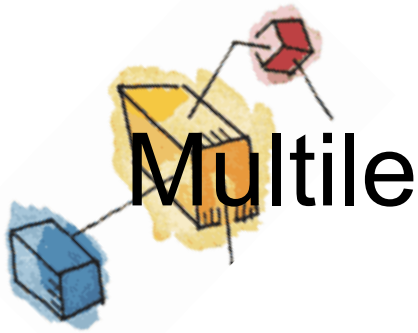
Multilevel Feed Back Queue (MFBQ)

- How to select a job to run?
  - Always choose a job with higher priority to run
  - **If priority(A) > Priority(B), A runs**
  - **If priority(A) == Priority(B), A & B run in RR**
- How to set priority to each job?
  - Dynamically vary the priority of a job based on its behaviour
  - Use the history of the job to predict its future behaviour
  - **Jobs start at top priority**
  - Each job is given a time slice to run
  - **If job uses the whole slice, demote process to the next lower priority queue**



Multilevel Feed Back Queue (MFBQ)

- Issue:
  - Since the CPU is always allocated to the process with the highest priority, low priority processes at the bottom queue may never execute – **starvation**!
- Solution:
  - After some time period S, move all the jobs in the system to the topmost queue
  - Promoted job will share the CPU with other high-priority jobs in a round-robin fashion
    - Guaranteed not to starve



Multilevel Feed Back Queue (MFBQ)

- How to parameterize MFBQ?
  - How many queues should there be?
  - How big should the time slice be per queue?
  - How often should priority be boosted in order to avoid starvation?
- No easy answers
  - Workloads dependent

Contemporary Scheduling

- CPU sharing -- timer interrupts
  - Time quantum (or time slice) determined by interval timer
- With preemption
- Priority-based process (job) selection
  - Select the highest priority process
  - Priority reflects policy
- Usually a variant of Multilevel Feedback Queues