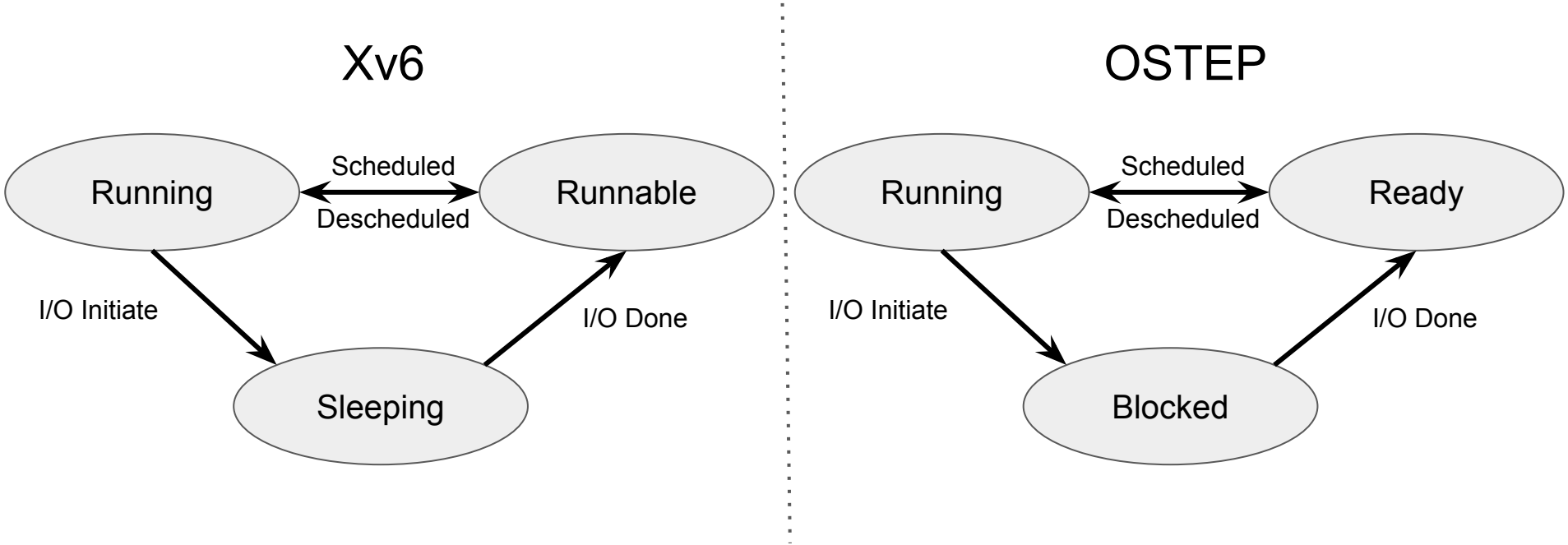
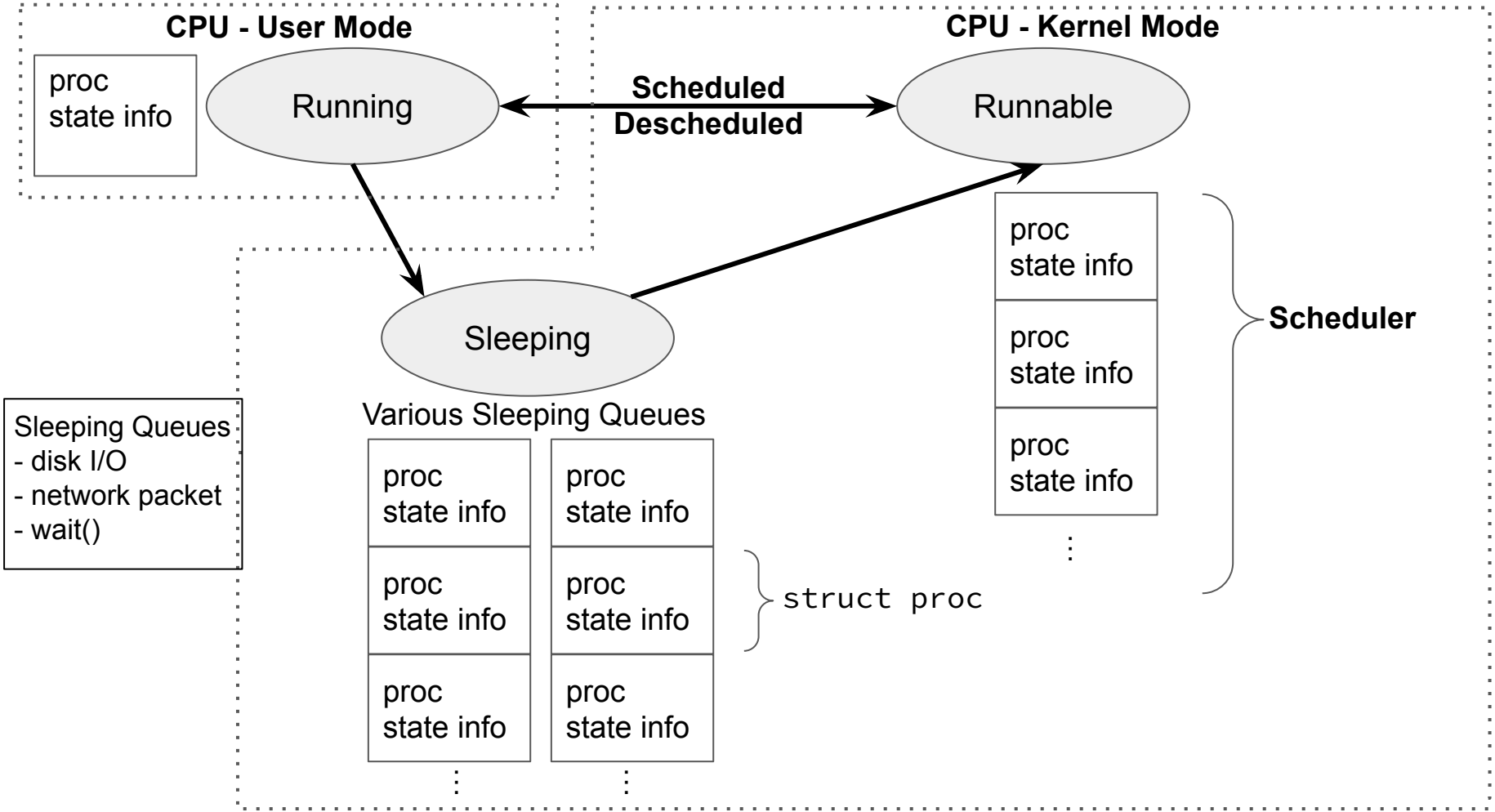
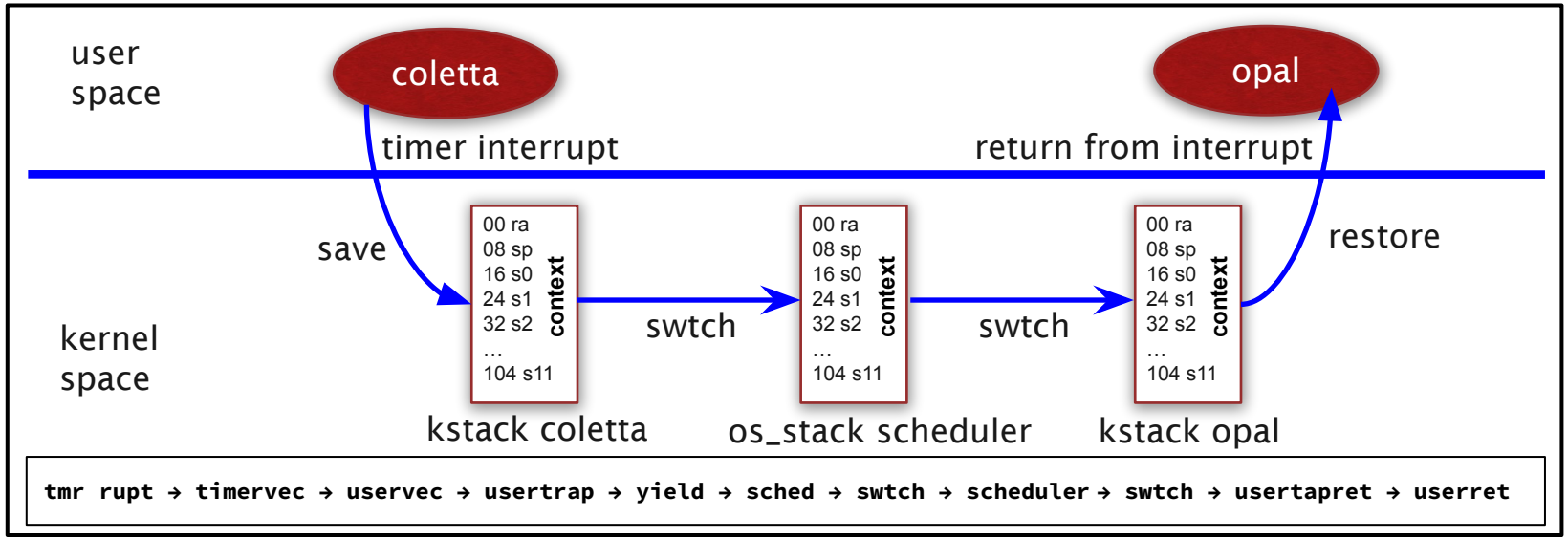
Scheduling Basics

Process State Transitions




CPU - User Mode
CPU - Kernel Mode
proc
state info
Running
Scheduled
Descheduled
Runnable
Sleeping
proc
state info
proc
state info
proc
state info
Scheduler
Sleeping Queues
- disk I/O
- network packet
- wait()
Various Sleeping Queues
proc
state info
proc
state info
proc
state info
proc
state info
proc
state info
proc
state info
struct proc

Context Switch



Vocabulary

- **Workload**: set of **job** descriptions (arrival time, run_time)
  - Job: Process - has to complete some computation
  - Workload is multiple jobs
  - Process alternates between CPU and I/O
  - Process moves between runnable and sleeping queues
- **Scheduler**: logic that decides which runnable job to run
- **Metric**: measurement of scheduling quality

What We Do

Examine various schedulers on various workloads and use metrics to compare the schedulers

Scheduling Goals (Metrics)

- Minimize turnaround time - time to complete a job
  - Do not want to wait long for job to complete
  - Completion_time - arrival_time
- Minimize response time - time to start a job
  - Schedule interactive jobs promptly so users see output quickly
  - Start_time - arrival_time
- Minimize waiting time
  - Do not want to spend much time in Ready queue
- Maximize throughput
  - Want many jobs to complete per unit of time
- Maximize resource utilization
  - Keep expensive devices busy
- Minimize overhead
  - Reduce number of context switches
- Maximize fairness
  - All jobs get same amount of CPU over some time interval

Can Be Measured

Schedulers

- Basic Schedulers
  - FIFO - first in first out
  - SJF - shortest job first
  - STCF - shortest time to completion first
  - RR - round robin
- MLFQ - Multilevel Feedback Queue
- Fair Schedulers
  - Lottery
  - Stride
  - Linux Completely Fair Scheduler

Examine each scheduler
Compare them using some metrics from previous slide

Initial Workload Assumptions

Unrealistic, but we can get started

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 (no I/O)
4. Run-time of each job is known

Scheduling Basics

Workloads:

- arrival_time
- run_time

Schedulers:

- FIFO
- SJF
- STCF
- RR

Metrics:

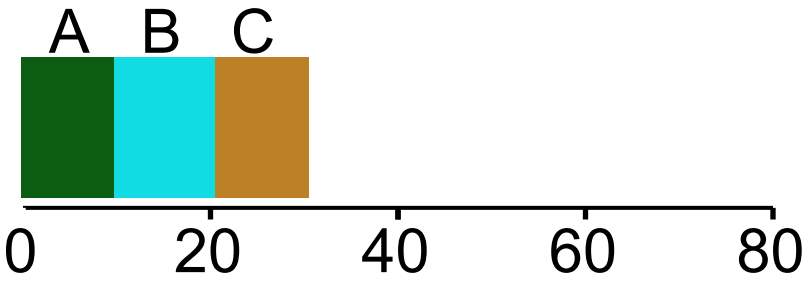
- turnaround_time
- response_time

FIFO Scheduler

1. Jobs run the same time
2. Jobs arrive at same time
3. We know the run time
4. Jobs have no I/O

JOB	arrival_time	run_time
A	~0	10
B	~0	10
C	~0	10

Time	**Event**
0	A arrives
0	B arrives
0	C arrives
0	run A
10	complete A
10	run B
20	complete B
20	run C
30	complete C



turnaround_time = *completion_time* - *arrival_time*

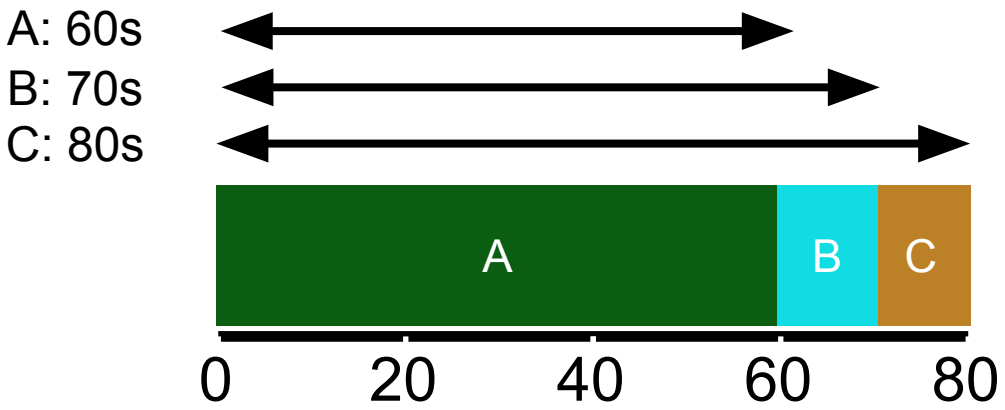
Average: (10 + 20 + 30) / 3 = **20s**

FIFO Big Job First

1. ~~Jobs run the same time~~
2. Jobs arrive at same time
3. We know the run time
4. Jobs have no I/O

JOB	arrival_time	run_time
A	~0	60
B	~0	10
C	~0	10

Time	Event
0	A arrives
0	B arrives
0	C arrives
0	run A
60	complete A
60	run B
70	complete B
70	run C
80	complete C



turnaround_time = completion_time - arrival_time

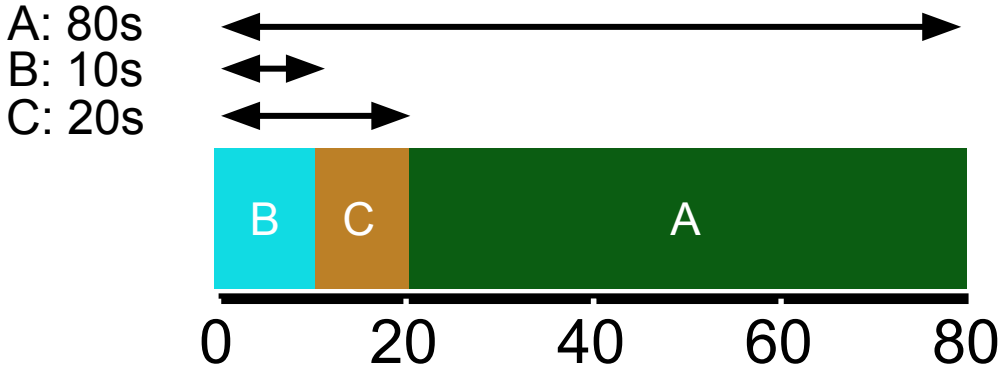
Average turnaround time: **70s**

Shortest Job First (SJF)

1. ~~Jobs run the same time~~
2. Jobs arrive at same time
3. We know the run time
4. Jobs have no I/O

JOB	arrival_time	run_time
A	~0	60
B	~0	10
C	~0	10

Time	Event
0	A arrives
0	B arrives
0	C arrives
0	run B
10	complete B
10	run C
20	complete C
20	run A
80	complete A



turnaround_time = completion_time - arrival_time

Average turnaround time with SJF

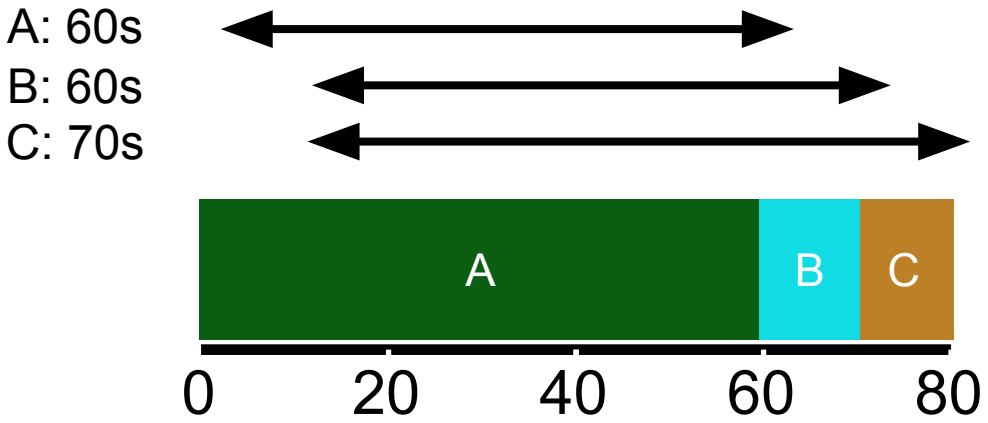
(80 + 10 + 20) / 3 = **~36.7s**

Shortest Job First (SJF)

1. ~~Jobs run the same time~~
2. ~~Jobs arrive at same time~~
3. We know the run time
4. Jobs have no I/O

JOB	arrival_time	run_time
A	~0	60
B	~10	10
C	~10	10

Time	Event
0	A arrives
10	B arrives
10	C arrives
0	run A
60	complete A
60	run B
70	complete B
70	run C
80	complete C



turnaround_time = completion_time - arrival_time

Average turnaround time

(60 + (70 – 10) + (80 – 10)) / 3 = **63.3s**

Preemptive Scheduling

- Previous schedulers:
  - FIFO and SJF defined as non-preemptive
  - Only schedule new job when previous job **voluntarily** relinquishes CPU (performs I/O or exits)

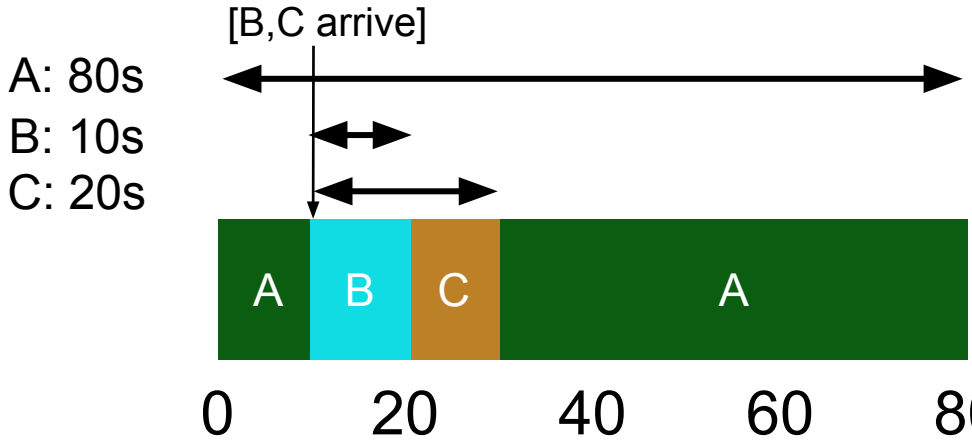
- New scheduler:
  - Preemptive: Potentially schedule different job at any point by taking CPU away from running job
  - STCF (Shortest Time-to-Completion First)
  - Always run job that will complete the quickest

Shortest Time to Completion

1. ~~Jobs run the same time~~
2. ~~Jobs arrive at same time~~
3. We know the run time
4. Jobs have no I/O

JOB	arrival_time	run_time
A	~0	60
B	~10	10
C	~10	10

Time	Event
0	A arrives
10	B arrives
10	C arrives
0	run A
10	run B
20	complete B
20	run C
30	complete C
80	complete A



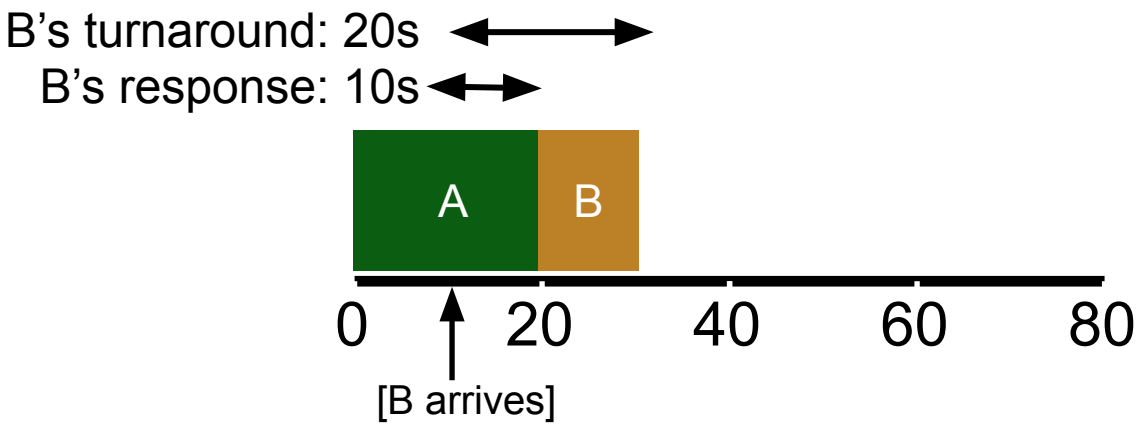
turnaround_time = completion_time - arrival_time

Average turnaround time

(80 + 10 + 20) / 3 = **36.6s**

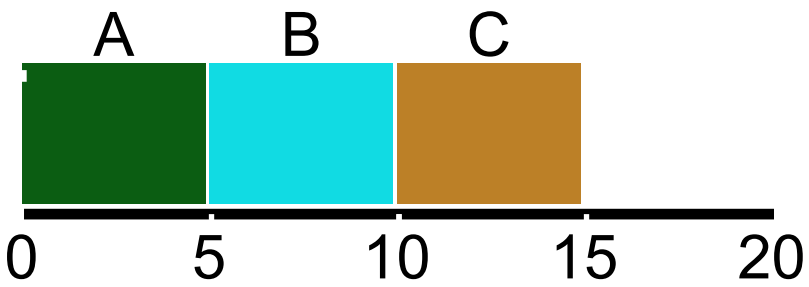
Response Time Metric

- We care when job starts instead of when it finishes
- New metric:
  - *response_time* = *first_run_time* - *arrival_time*

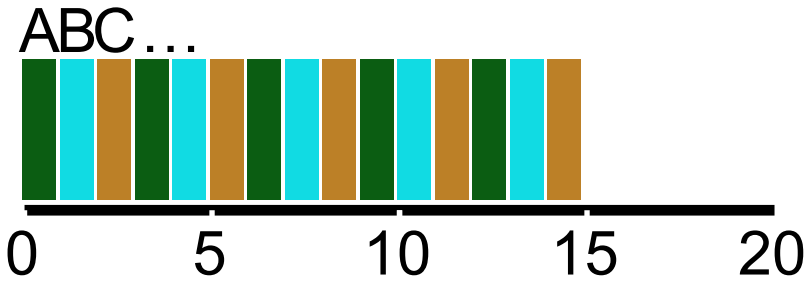


FIFO vs Round Robin

- Round Robin - alternate jobs at fixed length time intervals



Avg Response Time?
(0+5+10)/3 = **5**

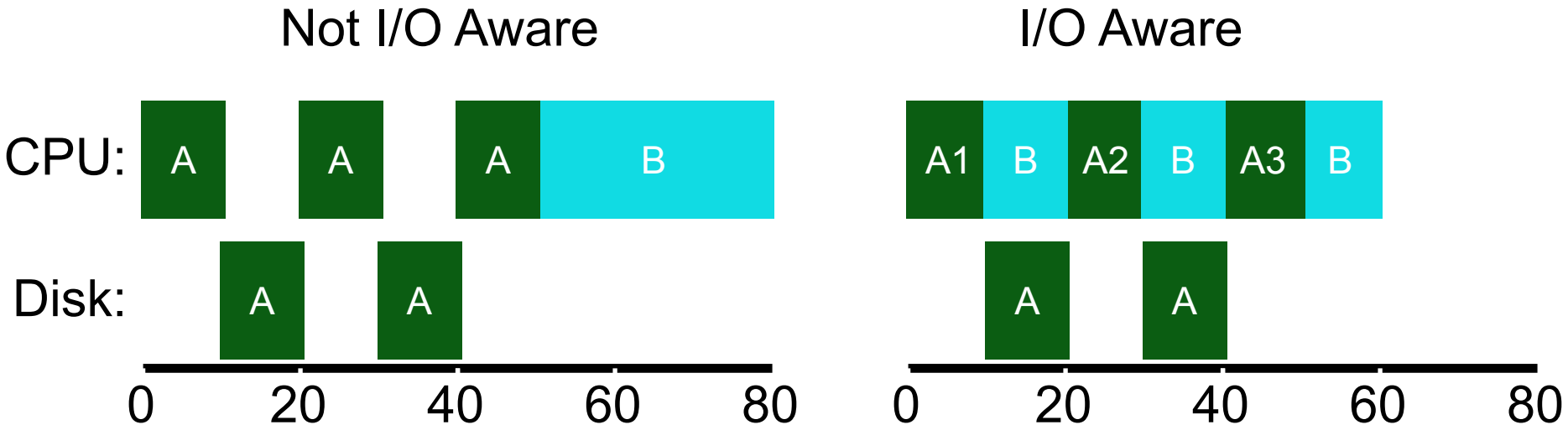


Avg Response Time?
(0+1+2)/3 = **1**

- RR : Average turnaround time with equal job lengths is horrible
- RR: gives short jobs a chance to run and finish fast

I/O in Scheduling

1. ~~Jobs run the same time~~
2. ~~Jobs arrive at same time~~
3. We know the run time
4. ~~Jobs have no I/O~~



Summary

JOB	arrival_time	run_time
A	~0	40
B	~0	20
C	~5	10

Scheduler Analysis:
What is turnaround time?
What is response time?
What is preemption?
What is FIFO scheduler?
What is SJF scheduler?
What is STCF scheduler?
What is RR?
What scheduler optimizes
average turnaround time?
What scheduler optimizes
average response time?

Timelines