CSE 421/521 - Operating Systems
Fall 2018

Lecture - VI
CPU Scheduling - I

Tevfik Koşar

University at Buffalo

September 13th, 2018

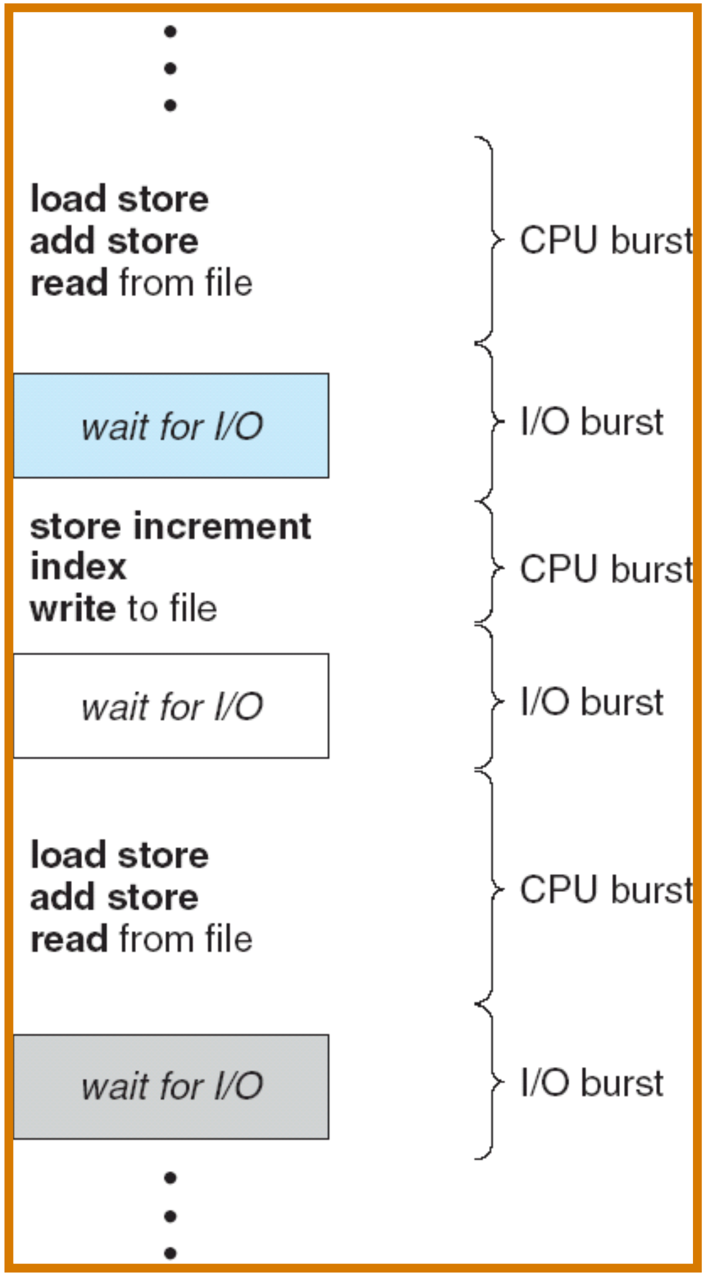
Roadmap

- CPU Scheduling
  - Basic Concepts
  - Scheduling Criteria & Metrics
  - Different Scheduling Algorithms
    - FCFS
    - SJF
    - Priority
    - RR
  - Preemptive vs Non-preemptive Scheduling
  - Gantt Charts & Performance Comparison

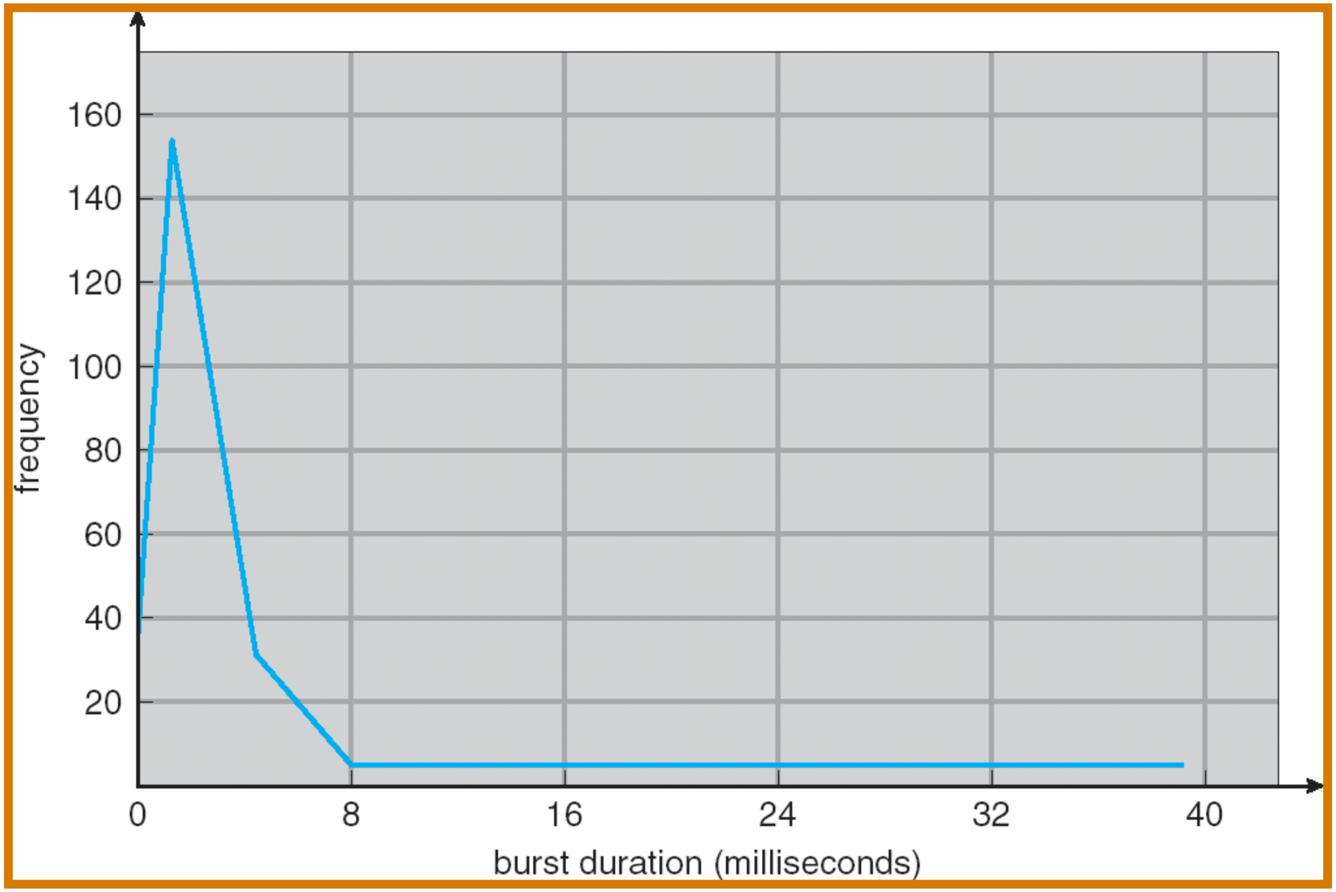
Basic Concepts

- Multiprogramming is needed for efficient CPU utilization
- CPU Scheduling: deciding which processes to execute when
- Process execution begins with a CPU burst, followed by an I/O burst
- CPU–I/O Burst Cycle – Process execution consists of a *cycle* of CPU execution and I/O wait

Alternating Sequence of CPU And I/O Bursts

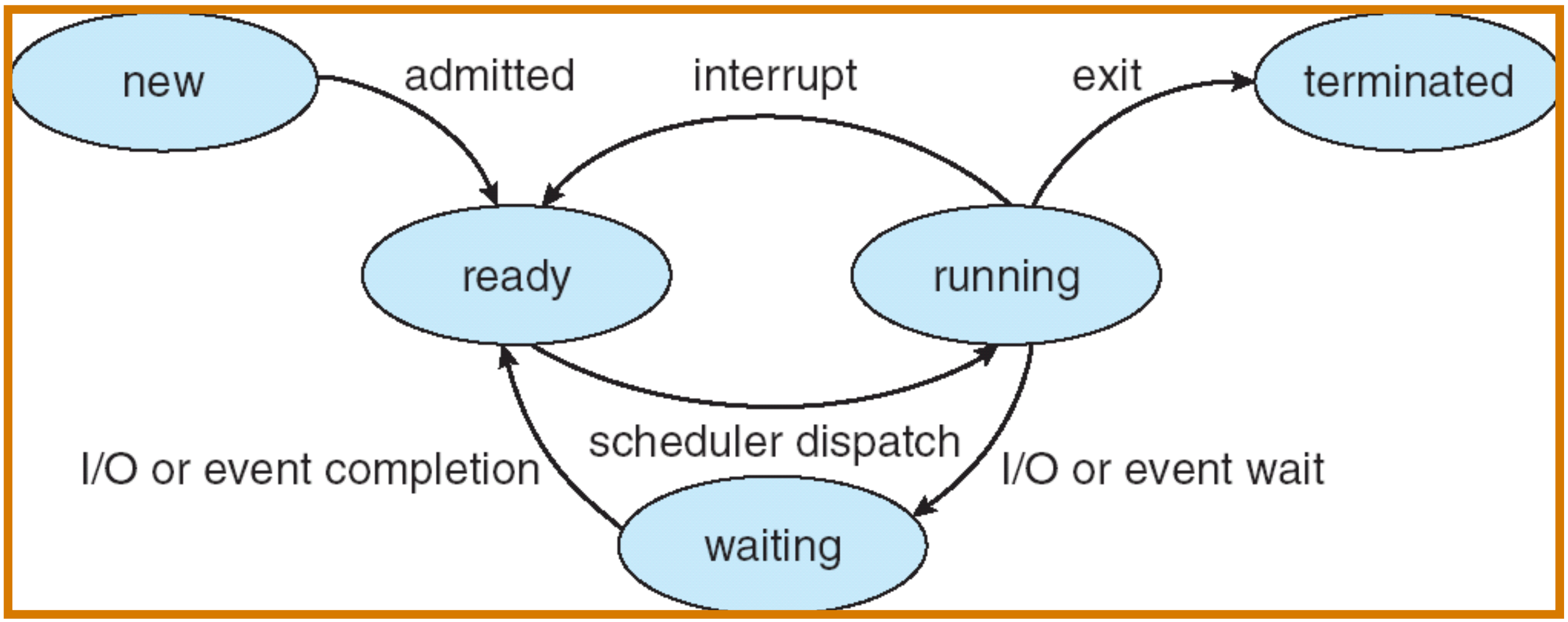


Histogram of CPU-burst Durations



Process State

- As a process executes, it changes *state*
  - **new**: The process is being created
  - **ready**: The process is waiting to be assigned to a process
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **terminated**: The process has finished execution



CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
  - ➔ short-term scheduler
- CPU scheduling decisions may take place when a process:
  1. A new process arrives
  2. Terminates
  3. Switches from running to waiting state
  4. Switches from running to ready state
  5. Switches from waiting to ready
- Scheduling under 2 and 3 is *nonpreemptive/cooperative*
  - Once a process gets the CPU, keeps it until termination/switching to waiting state/release of the CPU
- All other scheduling is *preemptive*
  - Most OS use this
  - Cost associated with access to shared data
  - i.e. time quota expires

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler;

  Its function involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

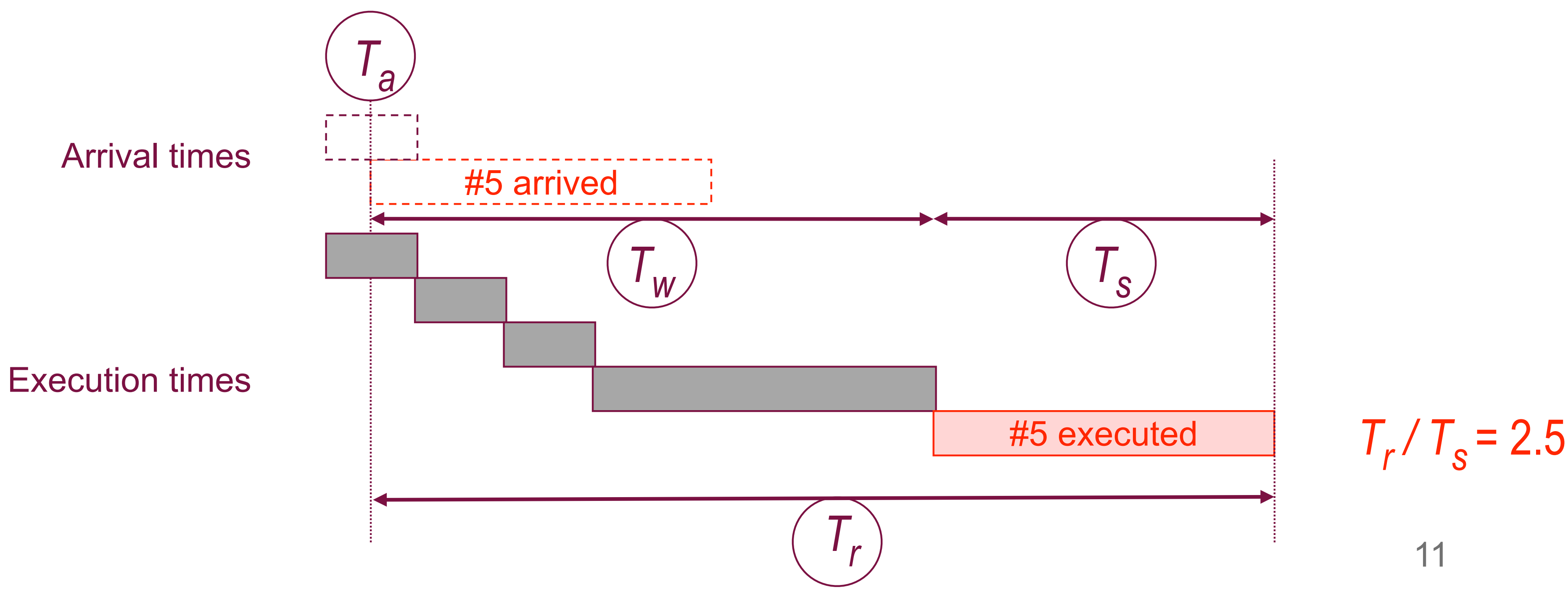
- CPU utilization – keep the CPU as busy as possible --> maximize
- Throughput – # of processes that complete their execution per time unit -->maximize
- Turnaround time – amount of time passed to finish execution of a particular process --> minimize
  - i.e. execution time + waiting time
- Waiting time – total amount of time a process has been waiting in the ready queue -->minimize
- Response time – amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment) -->minimize

Optimization Criteria

- Maximize CPU utilization
- Maximize throughput
- Minimize turnaround time
- Minimize waiting time
- Minimize response time

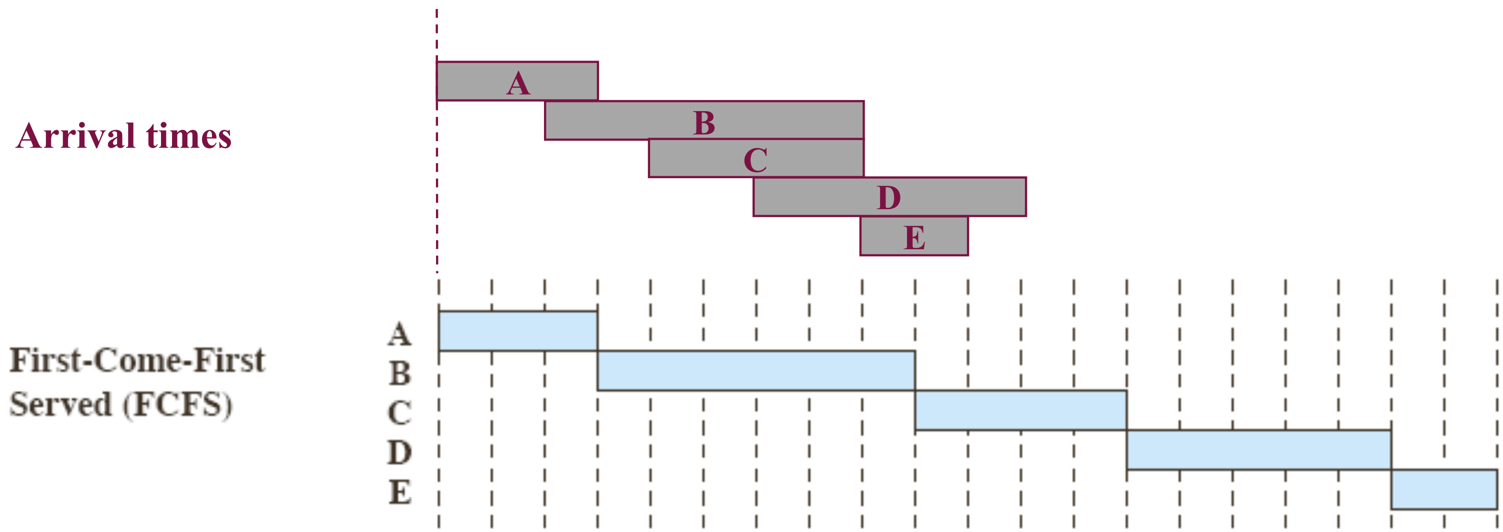
Scheduling Metrics

- <u>Scheduling metrics</u>
  - ✓ arrival time T_a = time the process became "Ready" (again)
  - ✓ wait time T_w = time spent waiting for the CPU
  - ✓ service time T_s = time spent executing in the CPU
  - ✓ <u>turnaround time</u> T_r = total time spent waiting and executing $= T_w + T_s$



First-Come, First-Served (FCFS) Scheduling

- ✓ processes are assigned the CPU in the order they request it
- ✓ when the running process blocks, the first “Ready” is run next
- ✓ when a process gets “Ready”, it is put at the end of the queue



FCFS	Finish Time	A 3	B 9	C 13	D 18	E 20	Mean
	Turnaround Time (T_r)	3	7	9	12	12	8.60
	T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56

FCFS scheduling policy

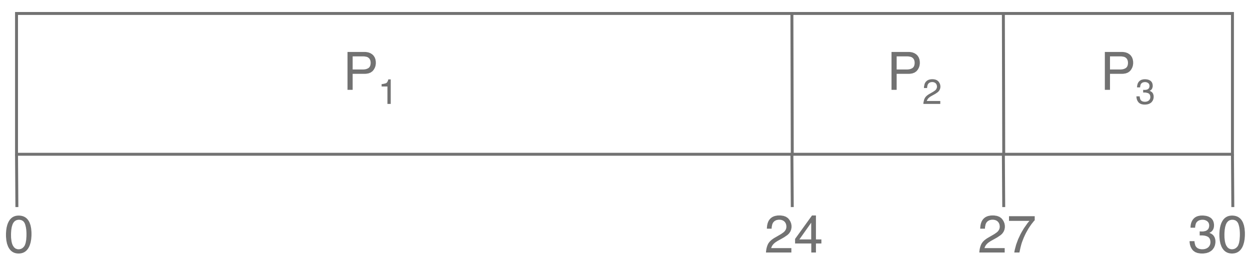
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

FCFS Scheduling - Example

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3

  The Gantt Chart for the schedule is:



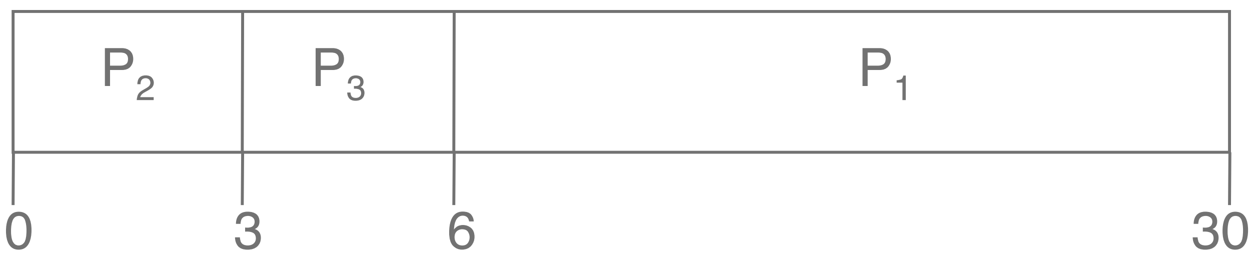
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

FCFS Scheduling - Example

Suppose that the processes arrive in the order

$$P_2 , P_3 , P_1$$

- The Gantt chart for the schedule is:



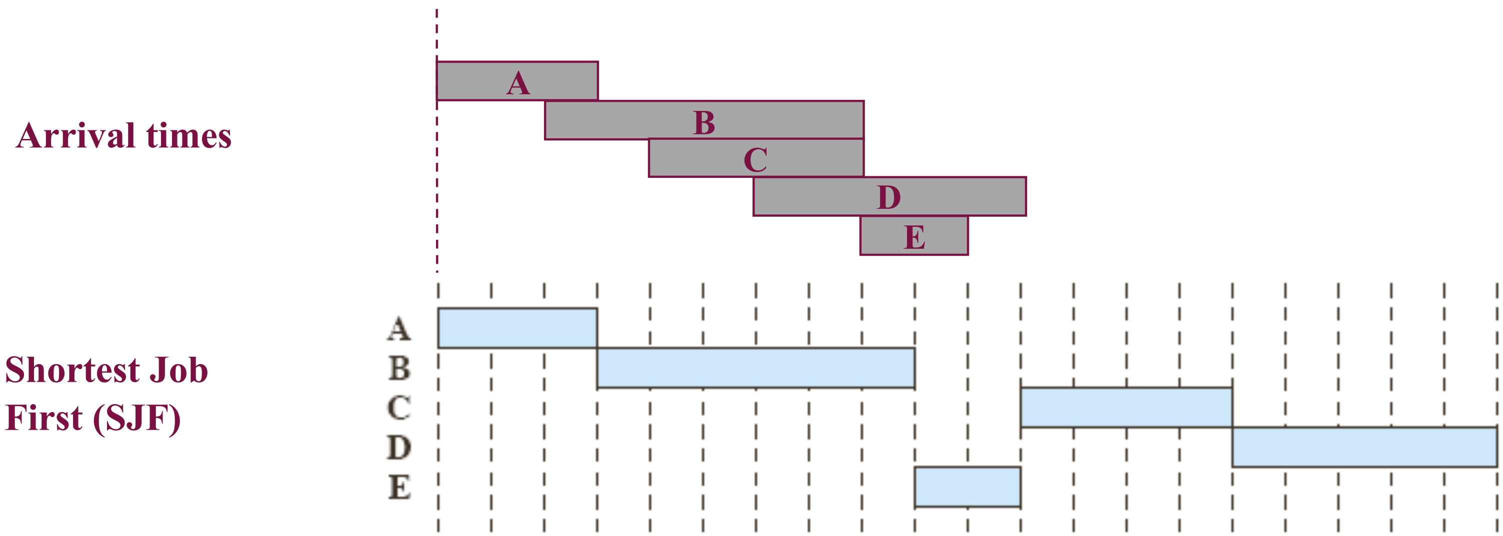
- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case
- *Convoy effect* short process behind long process

Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst
  - preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. -->This scheme is know as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal – gives minimum average waiting time for a given set of processes

Non-Preemptive SJF

- nonpreemptive, assumes the run times are known in advance
- among several equally important “Ready” jobs (or CPU bursts), the scheduler picks the one that will finish the earliest



		A	B	C	D	E	Mean
SJF	Finish Time	3	9	15	20	11	
	Turnaround Time (T_r)	3	7	11	14	3	7.60
	T_r/T_s	1.00	1.17	2.75	2.80	1.50	1.84

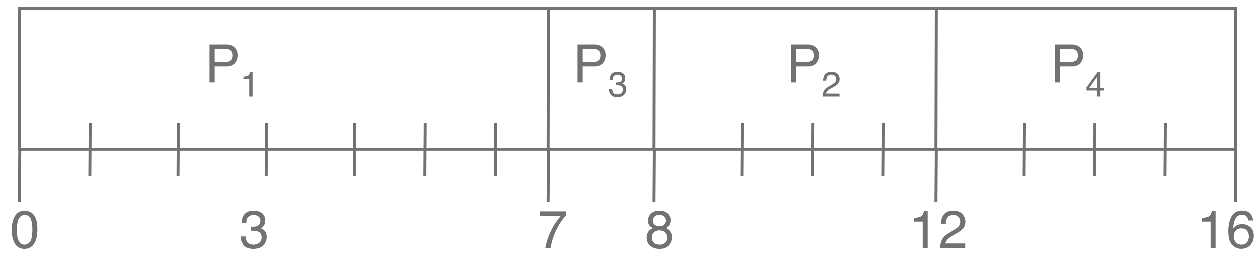
SJF scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Non-Preemptive SJF - Example

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

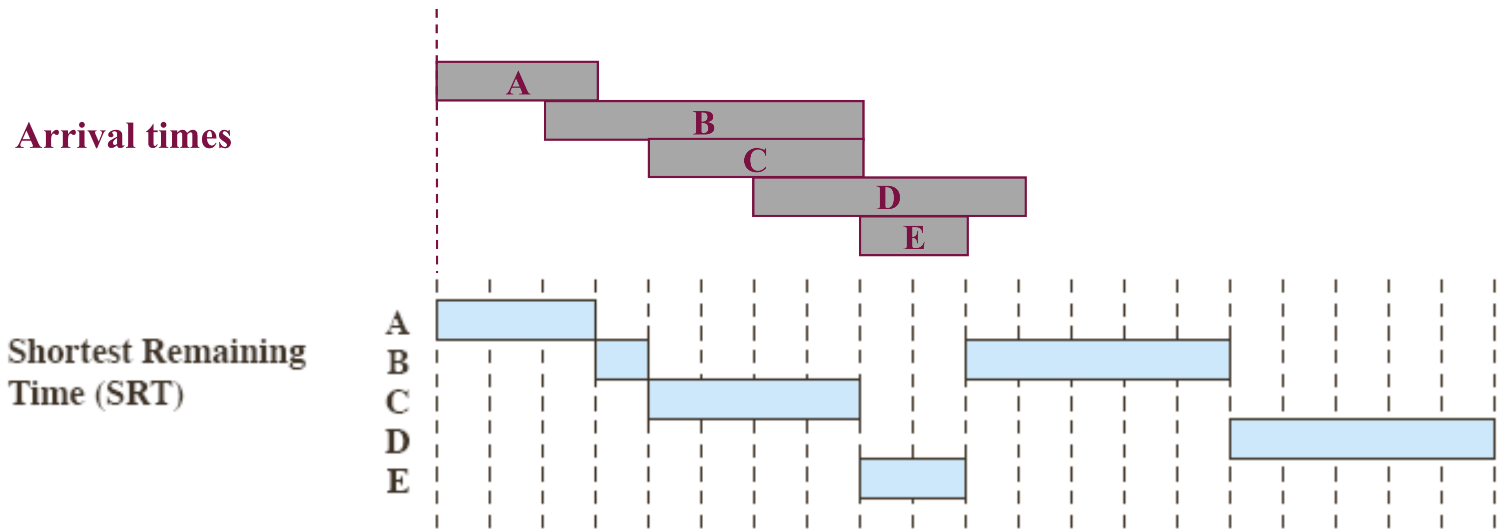
- SJF (non-preemptive) Gantt Chart



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Preemptive SJF (SRT)

- **Shortest Remaining Time (SRT)**
  - ✓ preemptive version of SJF, also assumes known run time
  - ✓ choose the process whose remaining run time is shortest
  - ✓ allows new short jobs to get good service



SRT		A	B	C	D	E	Mean
	Finish Time	3	15	8	20	10	
	Turnaround Time (T_r)	3	13	4	14	2	7.20
	T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59

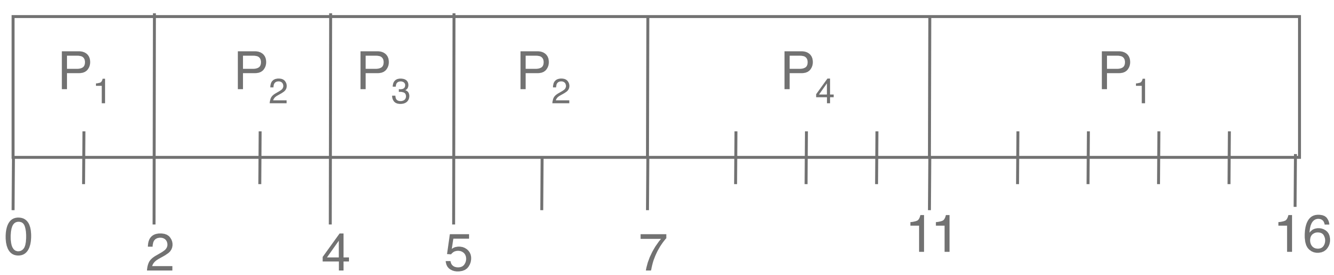
SRT scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive) Gantt Chart



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem ≡ Starvation – low priority processes may never execute
- Solution ≡ Aging – as time progresses increase the priority of the process

Example of Priority

Process	Arrival Time	Burst Time	Priority
P_1	0.0	7	2
P_2	2.0	4	1
P_3	4.0	1	4
P_4	5.0	4	3

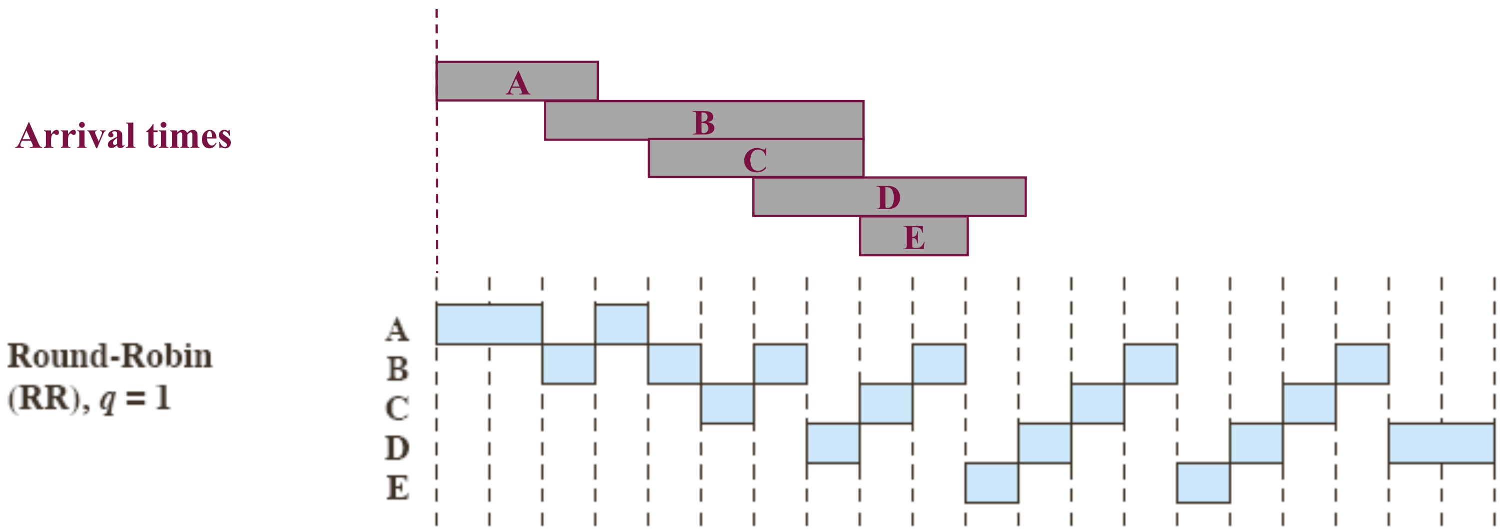
- Priority (non-preemptive)
  - P1 --> P2 --> P4 --> P3
- Priority (preemptive)
  - ??

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then each process gets $1/n$ of the CPU time in chunks of at most q time units at once. No process waits more than $(n-1)q$ time units.
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

Round Robin (RR)

- ✓ preemptive FCFS, based on a timeout interval, the **quantum** *q*
- ✓ the running process is interrupted by the clock and put last in a FIFO "Ready" queue; then, the first "Ready" process is run instead



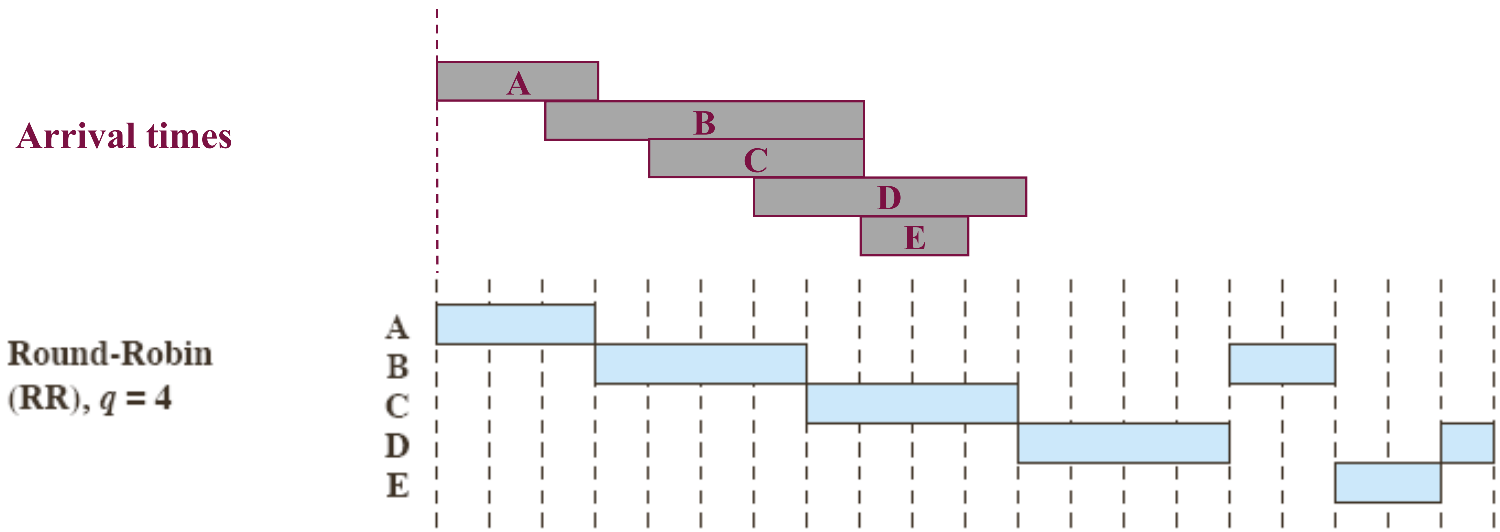
RR $q = 1$	Finish Time Turnaround Time (T_r) T_r/T_s	**A** 4 4 1.33	**B** 18 16 2.67	**C** 17 13 3.25	**D** 20 14 2.80	**E** 15 7 3.50	**Mean** 10.80 2.71

RR (q = 1) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Round Robin (RR)

- a crucial parameter is the quantum *q* (generally ~10–100ms)
  - *q* should be big compared to context switch latency (~10μs)
  - *q* should be less than the longest CPU bursts, otherwise RR degenerates to FCFS



RR $q = 4$		A	B	C	D	E	Mean
	Finish Time	3	17	11	20	19	
	Turnaround Time (T_r)	3	15	7	14	11	10.00
	T_r/T_s	1.00	2.5	1.75	2.80	5.50	2.71

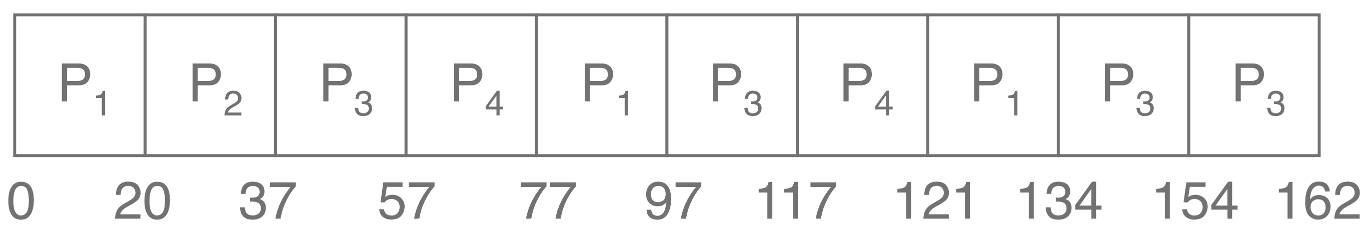
RR (*q* = 4) scheduling policy

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Example of RR with Time Quantum = 20

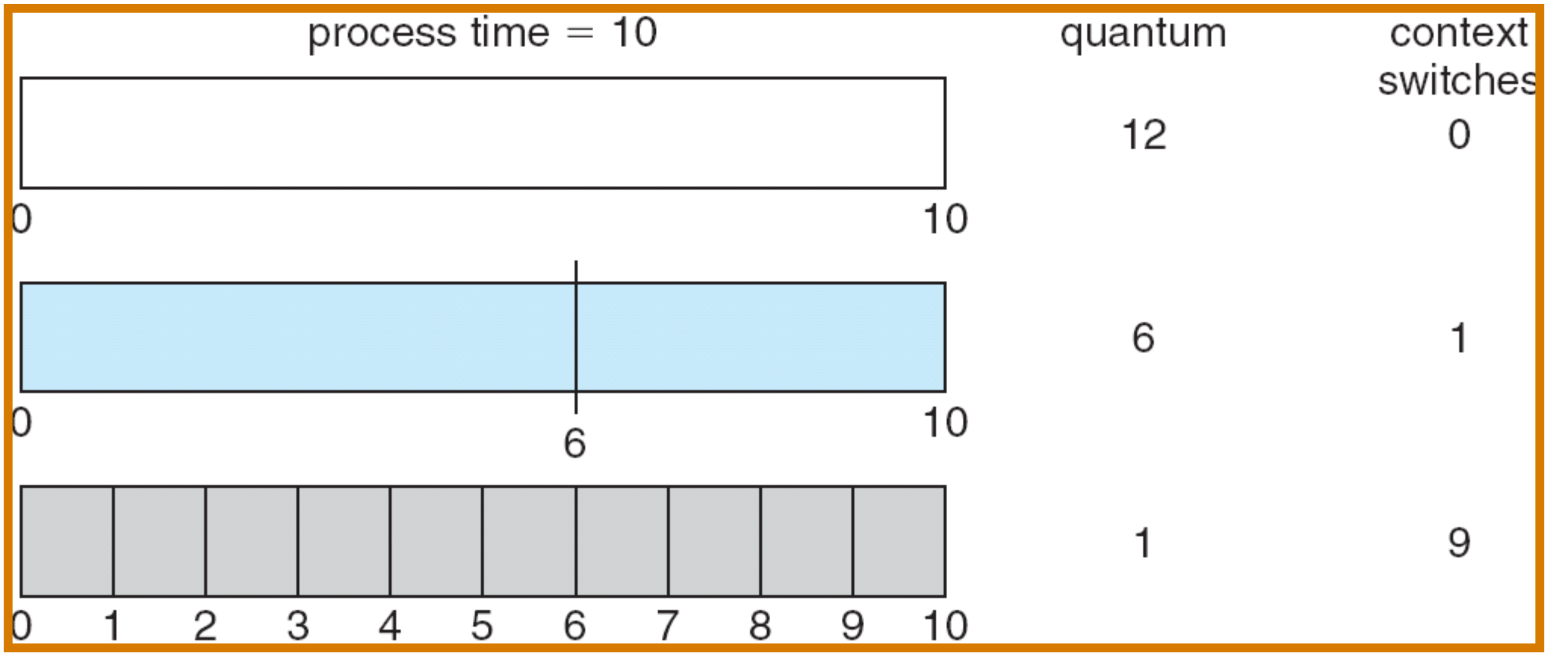
Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- For q=20, the Gantt chart is:

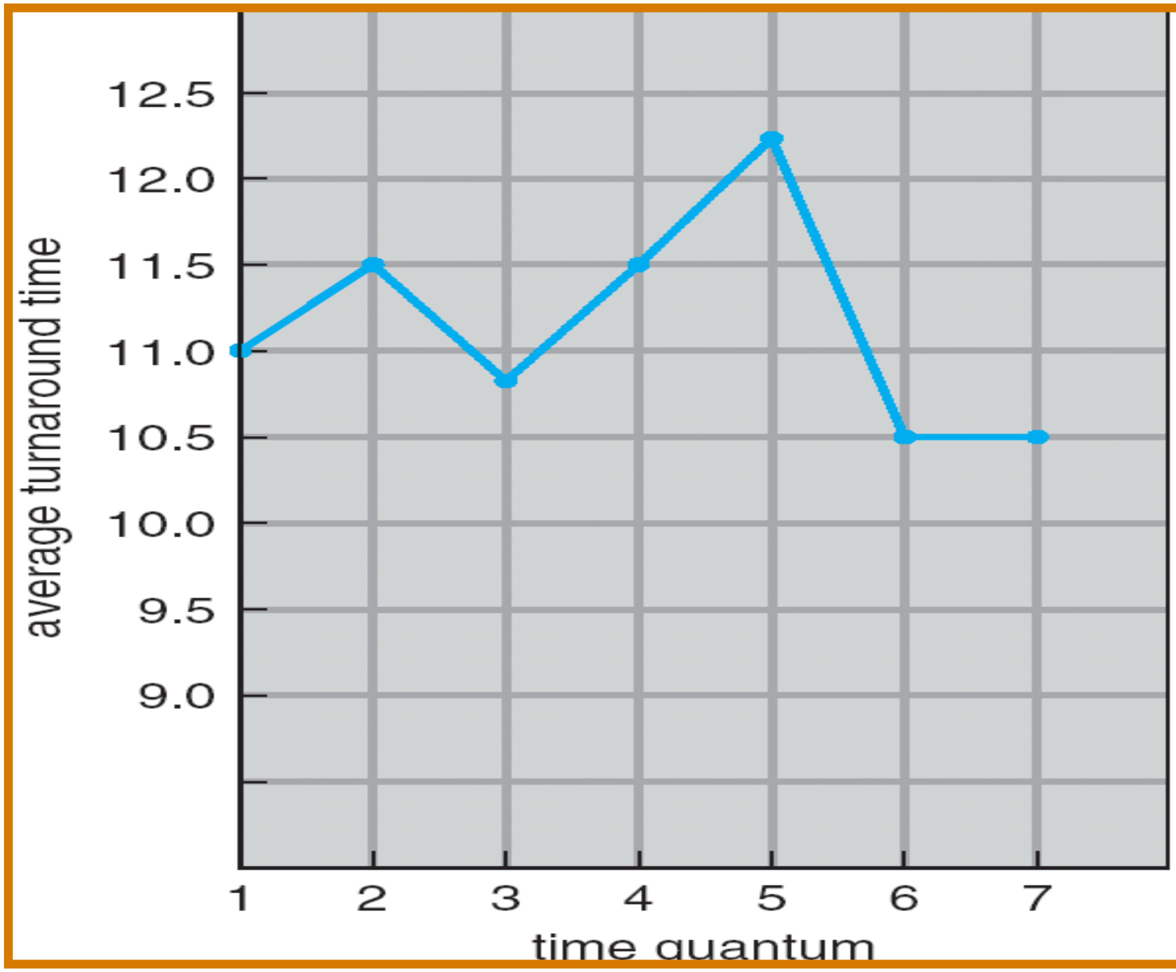


Typically, higher average turnaround than SJF, but better *response*

Time Quantum and Context Switch Time



Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Exercise

Consider a system using round-robin scheduling with a fixed quantum q. Every context switch takes s milliseconds. Any given process runs for an average of t milliseconds before it blocks or terminates (burst time).

> Determine the fraction of CPU time that will be wasted because of context switches for each of the following cases (your answer should be in terms of q, s, and t).

a) $t <= q$

b) $t >> q$ (t is much greater than q)

c) q approaches 0

Exercise (Soln)

Consider a system using round-robin scheduling with a fixed quantum *q*. Every context switch takes *s* milliseconds. Any given process runs for an average of *t* milliseconds before it blocks or terminates (burst time).

> Determine the fraction of CPU time that will be wasted because of context switches for each of the following cases (your answer should be in terms of *q*, *s*, and *t*).

a) t <= q —> **s / (t+s)**

b) t >> q (t is much greater than q) —> **s / (q + s)**

c) q approaches 0 —> **100%**

Comparison of Scheduling Algorithms

FCFS

PROS:

- It is a fair algorithm
  - schedule in the order that they arrive

CONS:

- Average response time can be lousy
  - small requests wait behind big ones (convoy effect)
- May lead to poor utilization of other resources
  - FCFS may result in poor overlap of CPU and I/O activity
    - E.g., a CPU-intensive job prevents an I/O-intensive job from doing a small bit of computation, thus preventing it from going back and keeping the I/O subsystem busy

SJF

PROS:

- Provably optimal with respect to average waiting time
  - prevents convoy effect (long delay of short jobs)

CONS:

- Can cause starvation of long jobs
- Requires advanced knowledge of CPU burst times
  - this can be very hard to predict accurately!

Priority Scheduling

PROS:

- Guarantees early completion of high priority jobs

CONS:

- Can cause starvation of low priority jobs
- How to decide/assign priority numbers?

Round Robin

PROS:

- Great for timesharing
  - no starvation
- Does not require prior knowledge of CPU burst times
- Generally reduces average response time

CONS:

- What if all jobs are almost time same length?
  - Increases the turnaround time
- How to set the "best" time quantum?
  - if small, then context switch often, incurring high overhead
  - if large, then response time degrades

Exercise

Consider the following set of processes with the arrival time, priorities, and the length of the CPU burst time given in milliseconds:

Process	P1	P2	P3	P4	P5
Arrival Time	9	6	8	7	5
Priority	5	4	3	2	1
Burst Time	3	4	5	9	7

- Consider Shortest Remaining Time First (SRTF) scheduling policy is used; a) draw the GANNT chart; b) find the response time, waiting times and turnaround time for each process.

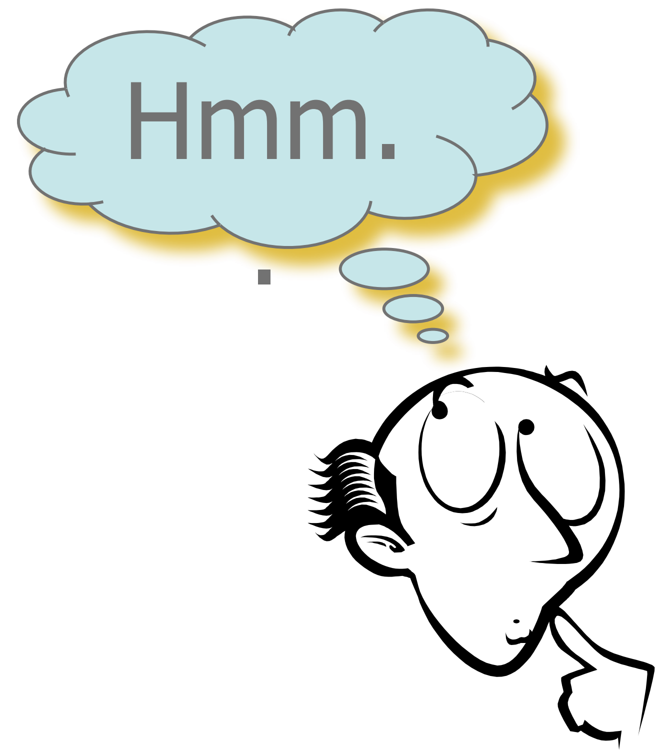
Exercise

Process ID	Arrival Time	Priority	Burst Time
A	0	3	20
B	5	1	15
C	10	2	10
D	15	4	5

- Draw gantt charts, find average turnaround, waiting, and response times for above processes, considering:
- 1) First Come First Served Scheduling
- 2) Shortest Job First Scheduling (non-preemptive)
- 3) Shortest Job First Scheduling (preemptive)
- 4) Round-Robin Scheduling (q=4)
- 5) Priority Scheduling (non-preemptive)
- 6) Priority Scheduling (preemptive)

Summary

- CPU Scheduling
  - Basic Concepts
  - Scheduling Criteria & Metrics
  - Different Scheduling Algorithms
    - FCFS
    - SJF
    - Priority
    - RR
  - Comparison of Algorithms



- Reading Assignment: Chapter 6 from Silberschatz.

Acknowledgements

- "Operating Systems Concepts" book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- "Operating Systems: Internals and Design Principles" book and supplementary material by W. Stallings
- "Modern Operating Systems" book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR