Operating Systems
CS2006

Lecture 8

CPU Scheduling-II

22nd February 2023

Dr. Rana Asif Rehman

What's in today's lecture

- Basic Concepts
- Scheduling Criteria
- **Scheduling Algorithms**

Scheduling Algorithms

- First come, First serve (FCFS)
- Shortest Job First (SJF)
- Priority Scheduling
- Round-Robin Scheduling
- Multi-level Queue Scheduling
- Multi-level Feed back queue Scheduling

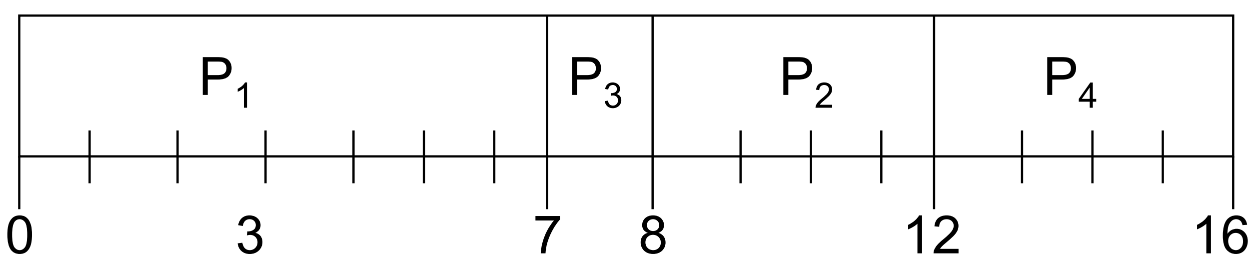
2. Shortest-Job-First (SJF) Scheduling

- 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 is given to the process it cannot be preempted until the process completes its CPU burst
  - Preemptive – if a new process arrives with CPU burst length less than the 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

Example of Non-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 (non-preemptive)



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

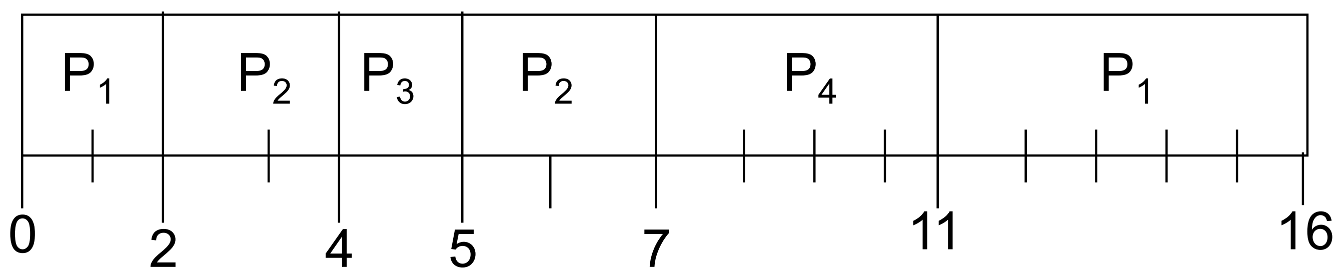
SRTF - Shortest Remaining Time First

- Preemptive version of SJF
- Ready queue ordered on length of time till completion (shortest first)
- Arriving jobs inserted at proper position
- Shortest job
  - Runs to completion (i.e. CPU burst finishes) or
  - Runs until a job with a shorter remaining time arrives (i.e. placed in the ready queue)

Example of Preemptive SJF (i.e., SRTF)

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

- Preemptive SJF (i.e., SRTF)



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

Shortest-Job-First (SJF) Scheduling

- Ready queue treated as a priority queue based on smallest CPU-time requirement
  - Arriving jobs inserted at proper position in queue
  - Shortest job (1st in queue) runs to completion
- In general, SJF is often used in long-term scheduling
- Advantages: provably optimal w.r.t. average waiting time
- Disadvantages: Unimplementable at the level of short-term CPU scheduling. Also, starvation is possible!
- Can do it approximately: use exponential averaging to predict length of next CPU burst ==> pick shortest predicted burst next!

Determining Length of Next CPU Burst

- Can only estimate the length
- Can be done by using the length of previous CPU bursts, using exponential averaging

1. $t_n =$ actual lenght of n^{th} CPU burst
2. $\tau_{n+1} =$ predicted value for the next (i.e., $n^{th}+1$) CPU burst
3. $\tau_{\mathrm{n}} =$ predicted value for the n^{th} CPU burst
4. $\alpha,\ 0 \leq \alpha \leq 1$
5. Define: $\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n.$

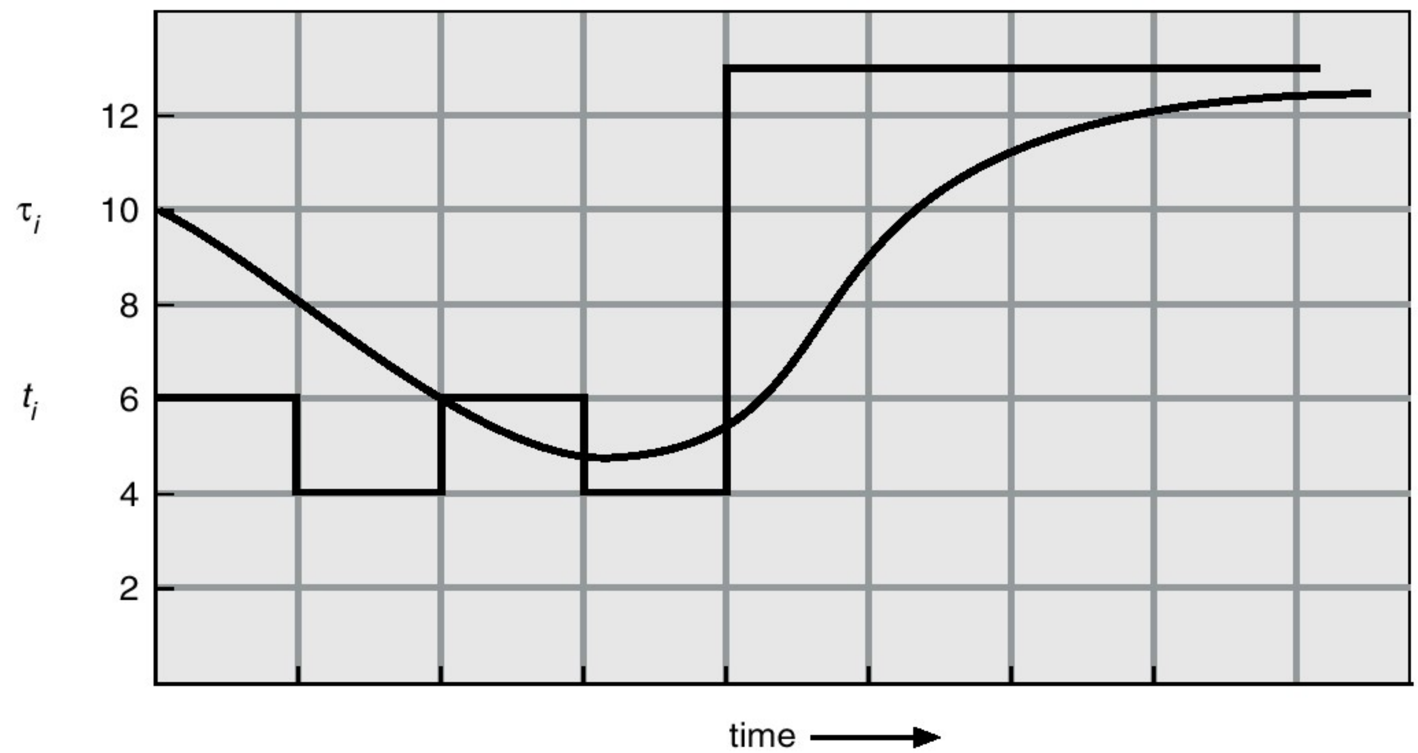
α = 0 implies making no use of recent history ($\tau_{n+1} = \tau_n$)

α = 1 implies τ_{n+1} = t_n (past prediction not used)

α = 1/2 implies weighted (older bursts get less and less weight)

Prediction of the Length of the Next CPU Burst

This figure is for $\alpha = 0.5$ and $\tau_0 = 10$



CPU burst (t_i)		6	4	6	4	13	13	13	. . .
"guess" (τ_i)	10	8	6	6	5	9	11	12	. . .

3. Priority Scheduling

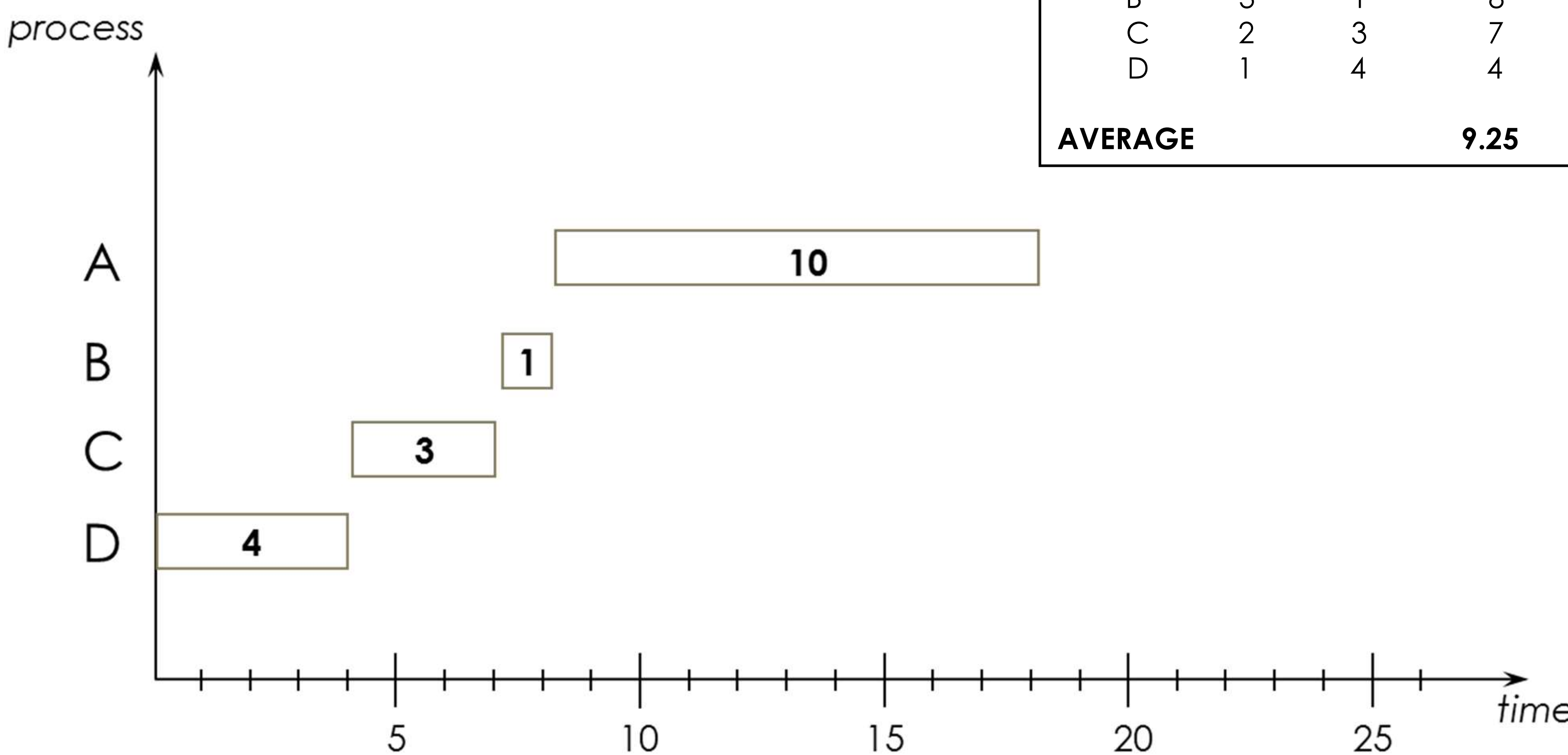
- A priority number (integer) is associated with each process
  - Priority can be internally computed (e.g., may involve time limits, memory usage) or externally (e.g., user type, funds being paid)
  - In SJF, priority is simply the predicted next CPU burst time
- The CPU is allocated to the process with the highest priority (smallest integer might mean highest priority)
- **Starvation** is a problem, where low priority processes may never execute
- **Solution:** as time progresses, the priority of the long waiting (starved) processes is increased. This is called **aging**

Priority scheduling

- Priority scheduling can be Preemptive or Non-Preemptive
- When a process arrives and enters the Ready Queue
- Its priority is compared with the currently Running Process
- If Higher
  - Preemptive Scheduling
    - Run the New process
  - Non-Preemptive Scheduling
    - Continue running the process

Priority Scheduling

process	priority	service time t_s	turnaround time t_t	waiting time t_w
A	4	10	18	8
B	3	1	8	7
C	2	3	7	4
D	1	4	4	0
AVERAGE			**9.25**	**4.75**



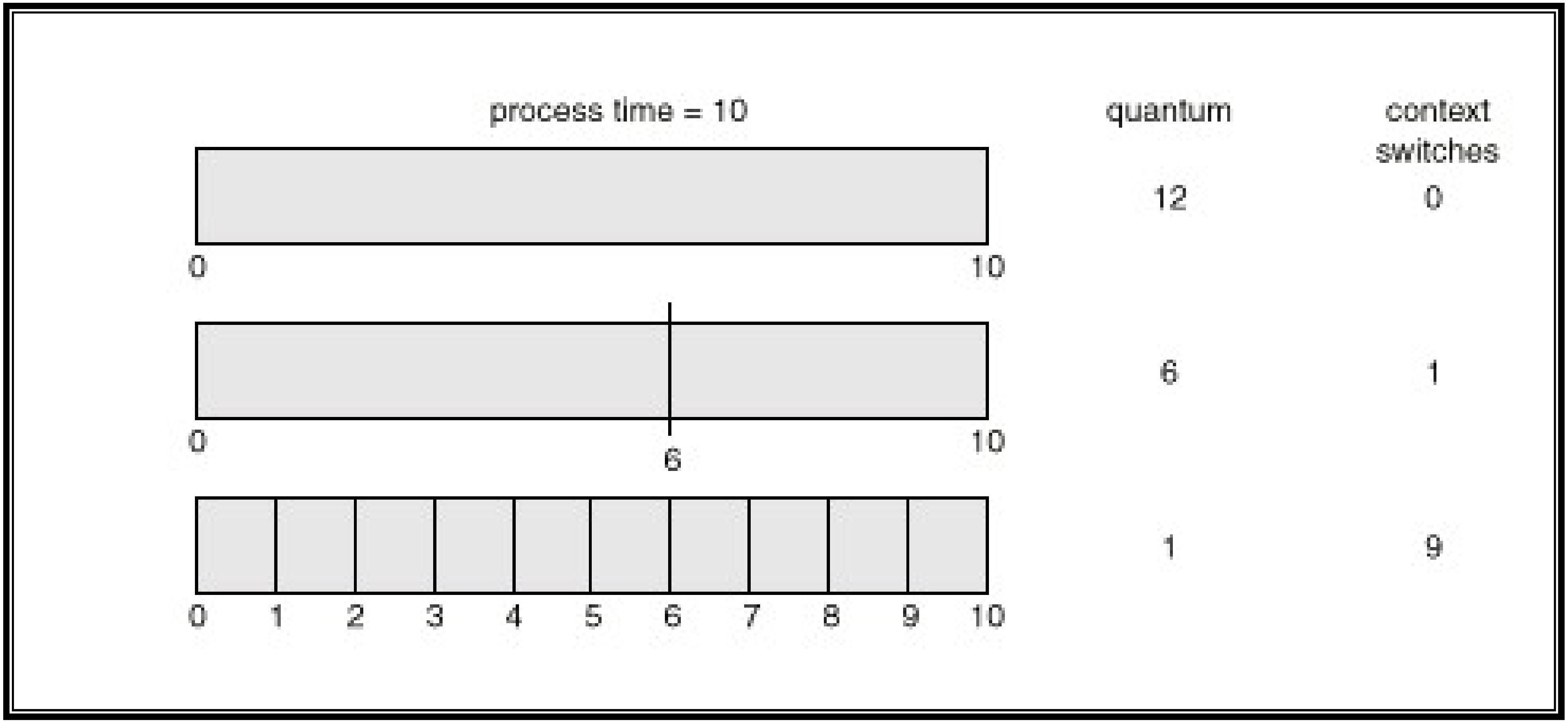
4. Round Robin (RR)

- RR reduces the penalty that short jobs suffer with FCFS by preempting running jobs periodically
- Each process gets a small unit of CPU time (***time quantum***), usually 10-100 milliseconds
- The CPU blocks the current job when its reserved ***time quantum (time-slice)*** is exhausted
  - The current job is then put at the end of the ready queue if it has not yet completed
  - If the current job is completed, it will exit the system (terminate)

Round Robin (RR)

- 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: the critical issue with the RR policy is the length of the quantum q
  - q is large: RR will behave like FIFO and hence interactive processes will suffer
  - q is small: the CPU will be spending more time on context switching
    - q must be large with respect to context switch, otherwise overhead is too high

Time Quantum and Context Switch Time



Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

0 20 37 57 77 97 117 121 134 154 162

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

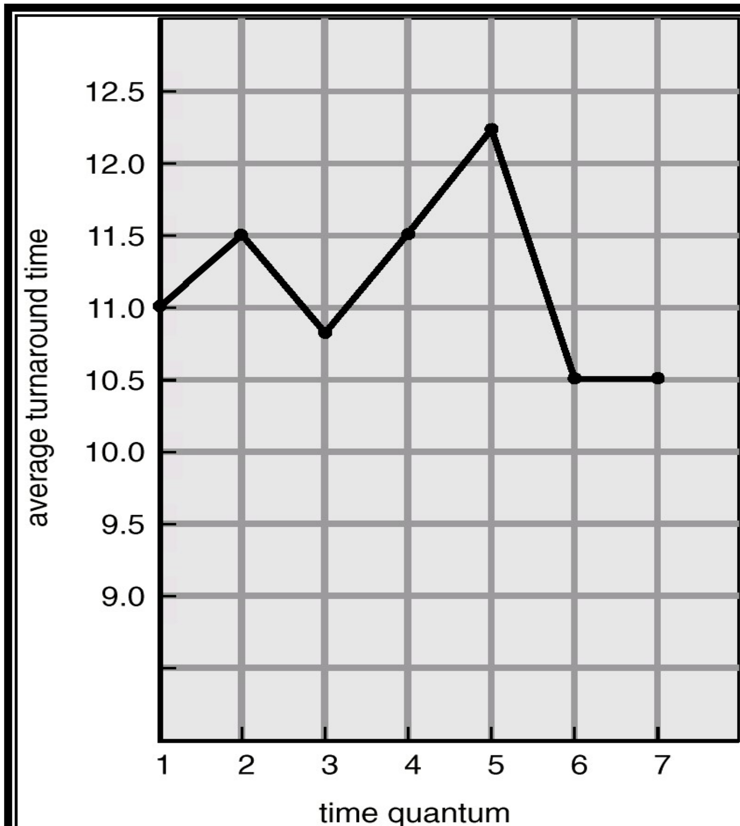
Round Robin's Disadvantage

- Good for Varying sized jobs
- But what about same-sized jobs?
- Assume 2 jobs of time =100 each:

- Avg completion time?
- (200 + 200) / 2 = 200
- How does this compare with FCFS for same two jobs?
- (100 + 200) / 2 = 150

Turnaround Time Varies With The Time Quantum

- Increasing the time quantum does not necessarily improve the average turnaround time!

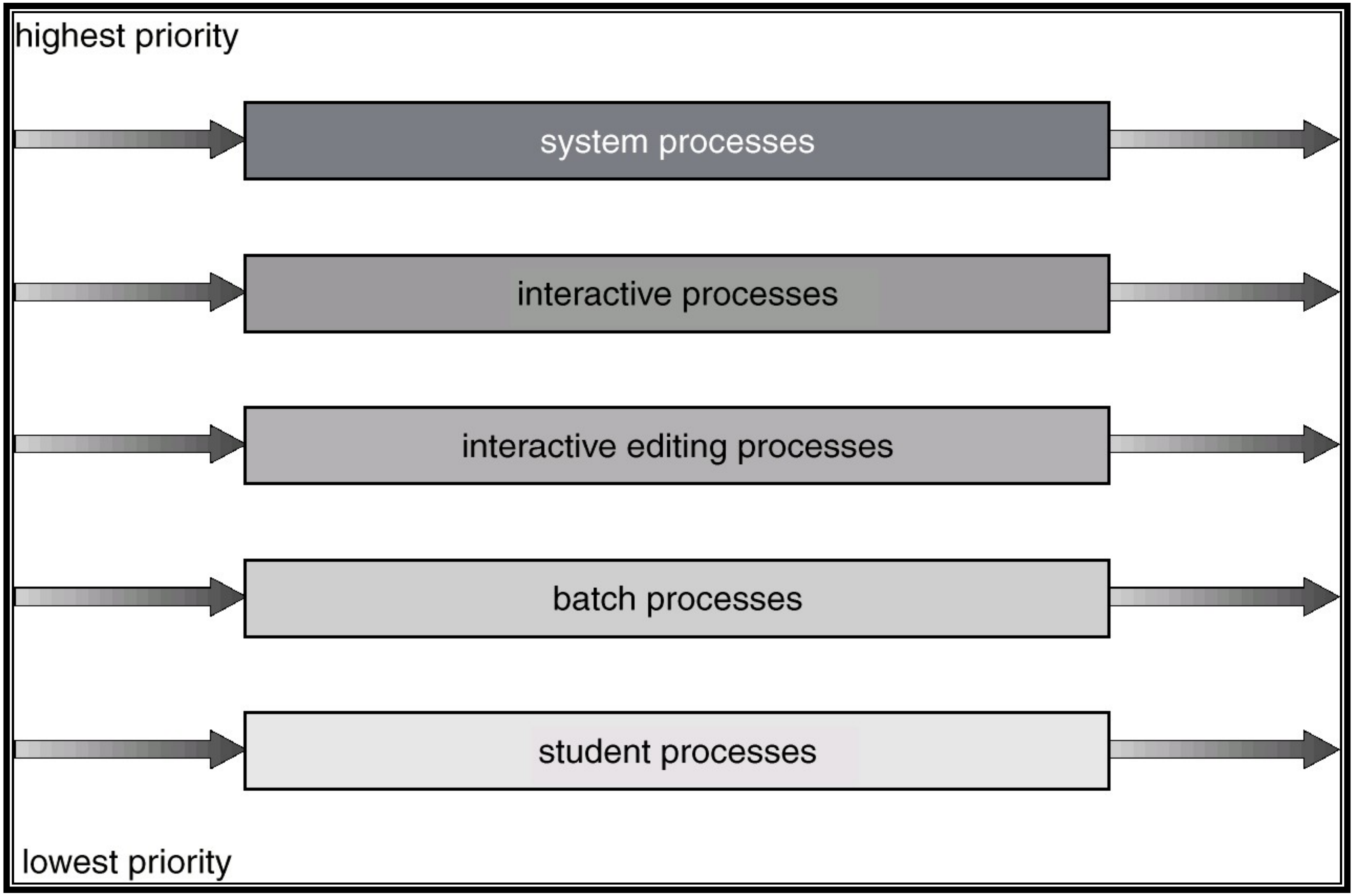


process	time
P_1	6
P_2	3
P_3	1
P_4	7

5. Multilevel Queue Scheduling

- Ready queue is partitioned into separate queues:
  - For example, foreground (interactive) and background (batch)
- Each queue has its own scheduling algorithm:
  - Foreground – RR
  - Background – FCFS
- Scheduling must be done between the queues
  - Fixed priority scheduling; (i.e., serve all from foreground then from background)
    - Possibility of starvation
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes
    - i.e., 80% to foreground in RR and 20% to background in FCFS

Multilevel Queue Scheduling



6. Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter
  - when that process needs service

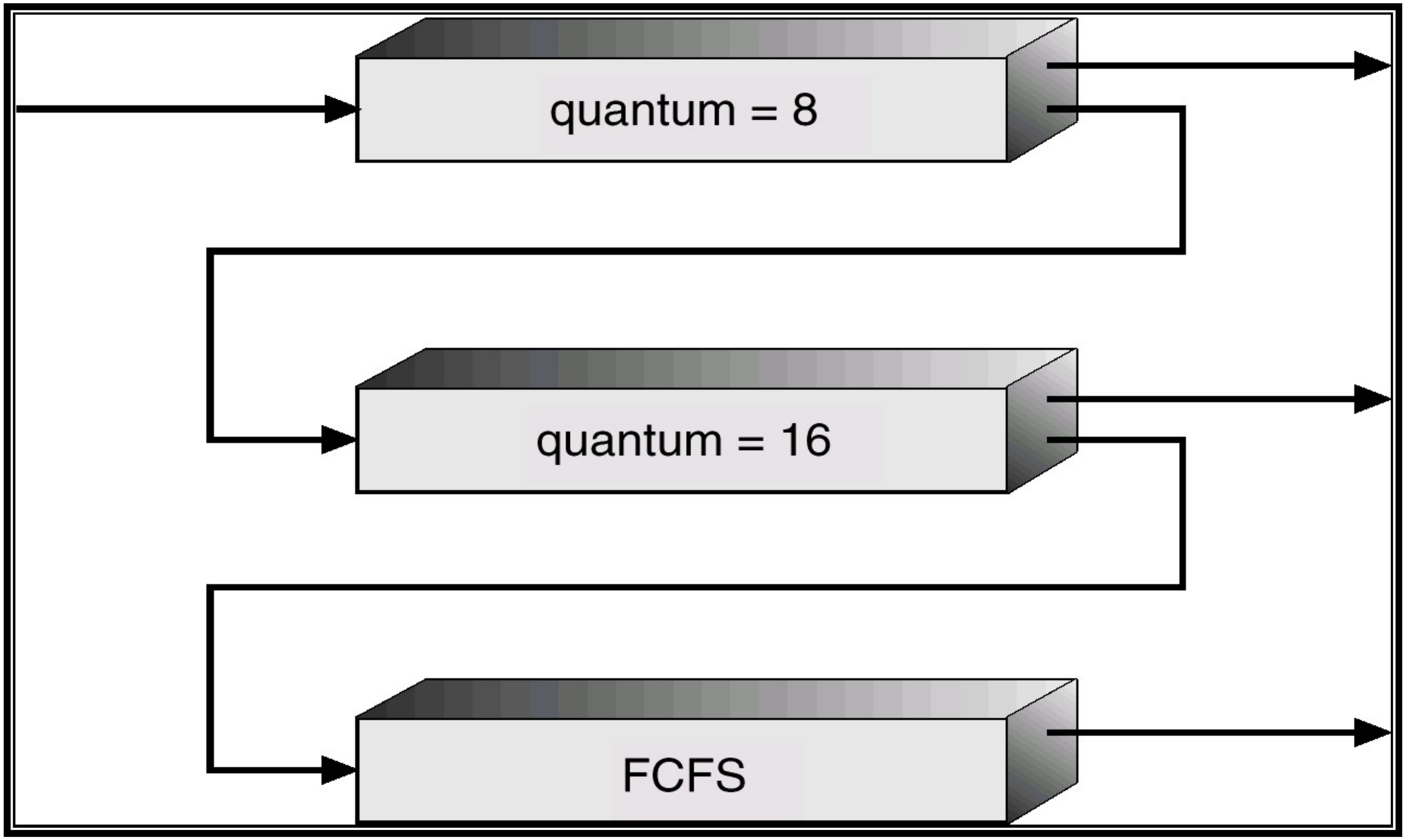
Multilevel Feedback Queue Scheduling

- Example
  - If a process used too much CPU time, then move it to a lower-priority queue
  - If a process waits too long in a lower priority queue, then move it to a higher priority queue

Multilevel Feedback Queue

- Three queues:
  - Q_0 – time quantum 8 milliseconds
  - Q_1 – time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 which is served FCFS. When it gains CPU, the job receives 8 milliseconds. If it does not finish in 8 milliseconds, the job is moved to queue Q_1
  - At Q_1 , the job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2

Multilevel Feedback Queue



MLFQ Example

- At t=0 there are three CPU bound processes (no I/O) in the system. The CPU burst lengths are: 30 units for P1, 20 units for P2, and 10 units for P3. The system has three RR queues with the following time slices: 1 for queue1, 2 for queue 2, and 4 for queue3. RR- 1 Unit of time slice

- Gantt Chart
- Average Completion Time
- Average Waiting Time

Solution:

The following GANTT chart.

```
  p  Process p, p=1,2,3 is running
  Qk Queue on level k

  Q1:      123
  Q2:         112233                      T                    T      T
  Q3:               111122223333111122223331111222211112222111121111111
 Time:              1         2         3         4         5         6
           123456789012345678901234567890123456789012345678901234567890
```

Completion Times

MLFQ P1:60; P2:53; P3:32 Mean Completion: (60+53+32)/3 = 48 1/3

Waiting Times

MLFQ P1:30; P2:33; P3:22 Mean Waiting: (30+33+22)/3 = 28 1/3

Summary of CPU Scheduling Algorithms

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
  - Nonpreemptive
  - Preemptive or Shortest Remaining Time First (SRTF)
- Priority Scheduling
  - Preemptive
  - Nonpreemptive
- Round Robin (RR)
- Multilevel Queue Scheduling
- Multilevel Feedback Queue

References

- Operating System Concepts (Silberschatz, 9th edition) Chapter 5