Operating Systems

Lecture 4a

Dr Ronald Grau

School of Engineering and Informatics

Spring term 2018

Previously

Scheduling

- Time scales: Long-, medium-, and short-term scheduling
- Scheduling criteria
- Scheduling algorithms: FCFS, SJF, SRT, RR

Today

Scheduling

- Performance overview of last week's scheduling policies: FCFS, SJF, SRT, RR
- Another look at Round Robin
- Multi-level queue scheduling
- Feedback scheduling
- Parametrised Scheduling Algorithms
- Real-time scheduling
- Java thread scheduling

Recap: Scheduling basics

Questions:

- What are CPU-bound and I/O-bound processes?
- What is starvation?
- What is throughput?
- What is turnaround time?
- What is waiting time?
- Why is it important for a scheduling algorithm to be fair?

Recap: Scheduling basics

Throughput and Turnaround Time

Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time

Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time
Decision mode	non-preemptive	non-preemptive	preemptive	Preemptive (at time quantum)

Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time
Decision mode	non-preemptive	non-preemptive	preemptive	Preemptive (at time quantum)
Overhead	very small	can be high	can be high	very small

Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time
Decision mode	non-preemptive	non-preemptive	preemptive	Preemptive (at time quantum)
Overhead	very small	can be high	can be high	very small
Fairness	penalises short processes	penalises long processes	penalises long processes	fair

Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time
Decision mode	non-preemptive	non-preemptive	preemptive	Preemptive (at time quantum)
Overhead	very small	can be high	can be high	very small
Fairness	penalises short processes	penalises long processes	penalises long processes	fair
Starvation	no	possible	possible	no

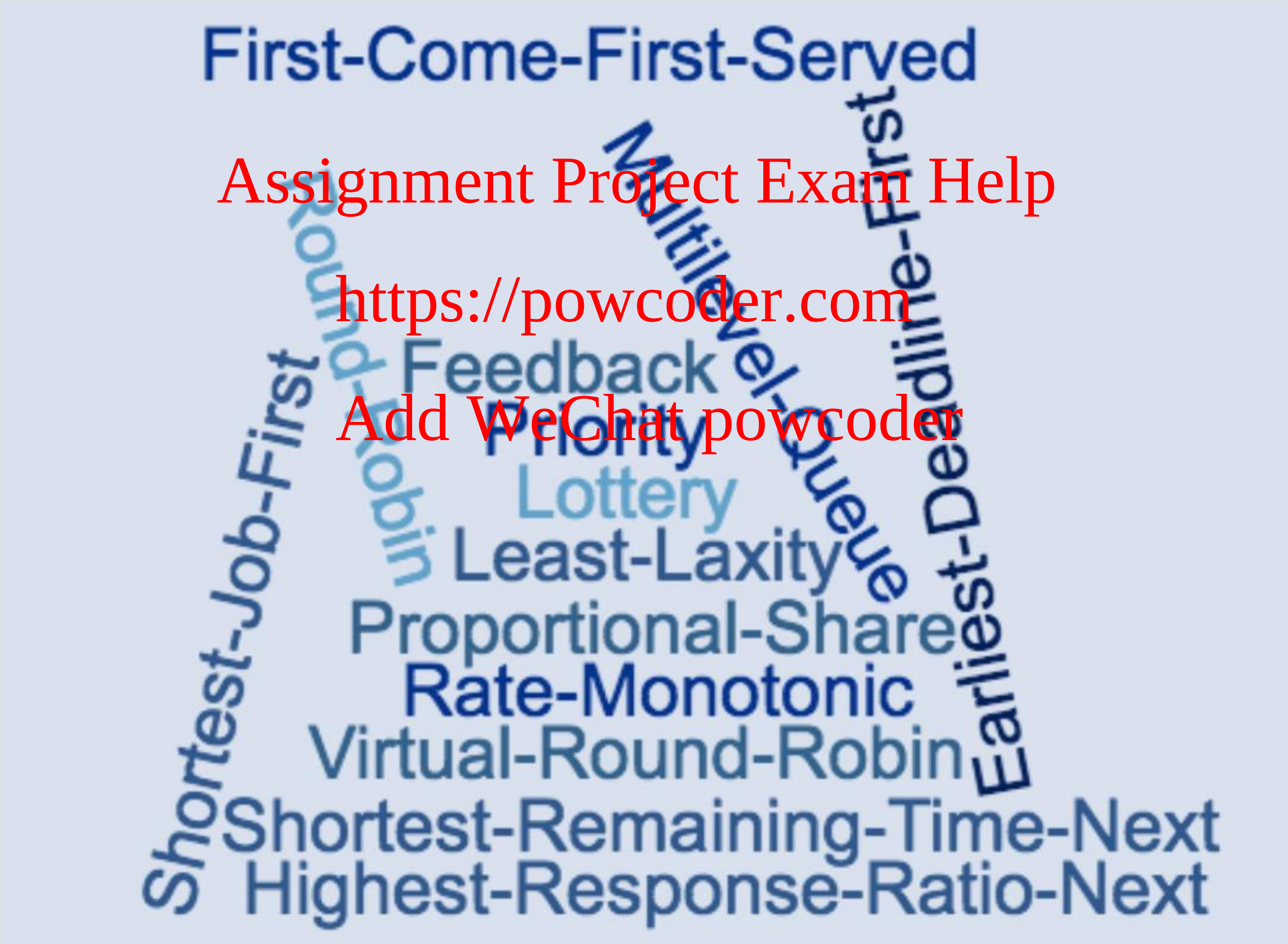
Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time
Decision mode	non-preemptive	non-preemptive	preemptive	Preemptive (at time quantum)
Overhead	very small	can be high	can be high	very small
Fairness	penalises short processes	penalises long processes	penalises long processes	fair
Starvation	no	possible	possible	no
Throughput	variable	high	high	depends on time quantum

Performance of scheduling policies

	First-Come First-Served	Shortest Job First	Shortest Remaining Time	Round Robin
Selection function	max waiting time	min execution time	min remaining execution time	max waiting time
Decision mode	non-preemptive	non-preemptive	preemptive	Preemptive (at time quantum)
Overhead	very small	can be high	can be high	very small
Fairness	penalises short processes	penalises long processes	penalises long processes	fair
Starvation	no	possible	possible	no
Throughput	variable	high	high	depends on time quantum
Waiting time	can be high	good for short proc.	better than SJF	depends on time quantum

Scheduling Algorithms

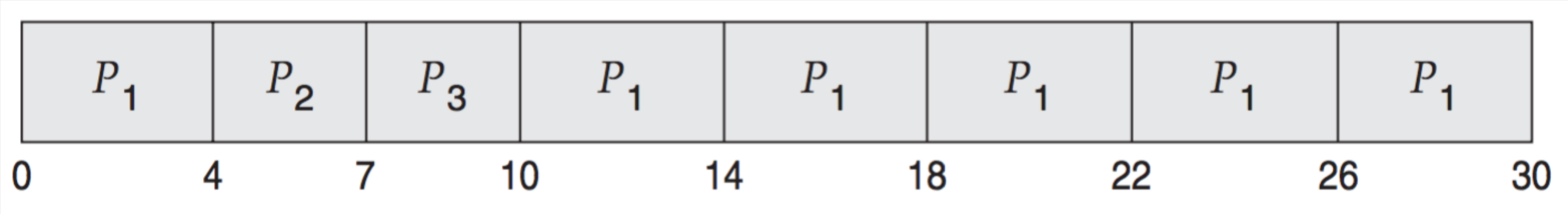


Another look at Round Robin

Recap:

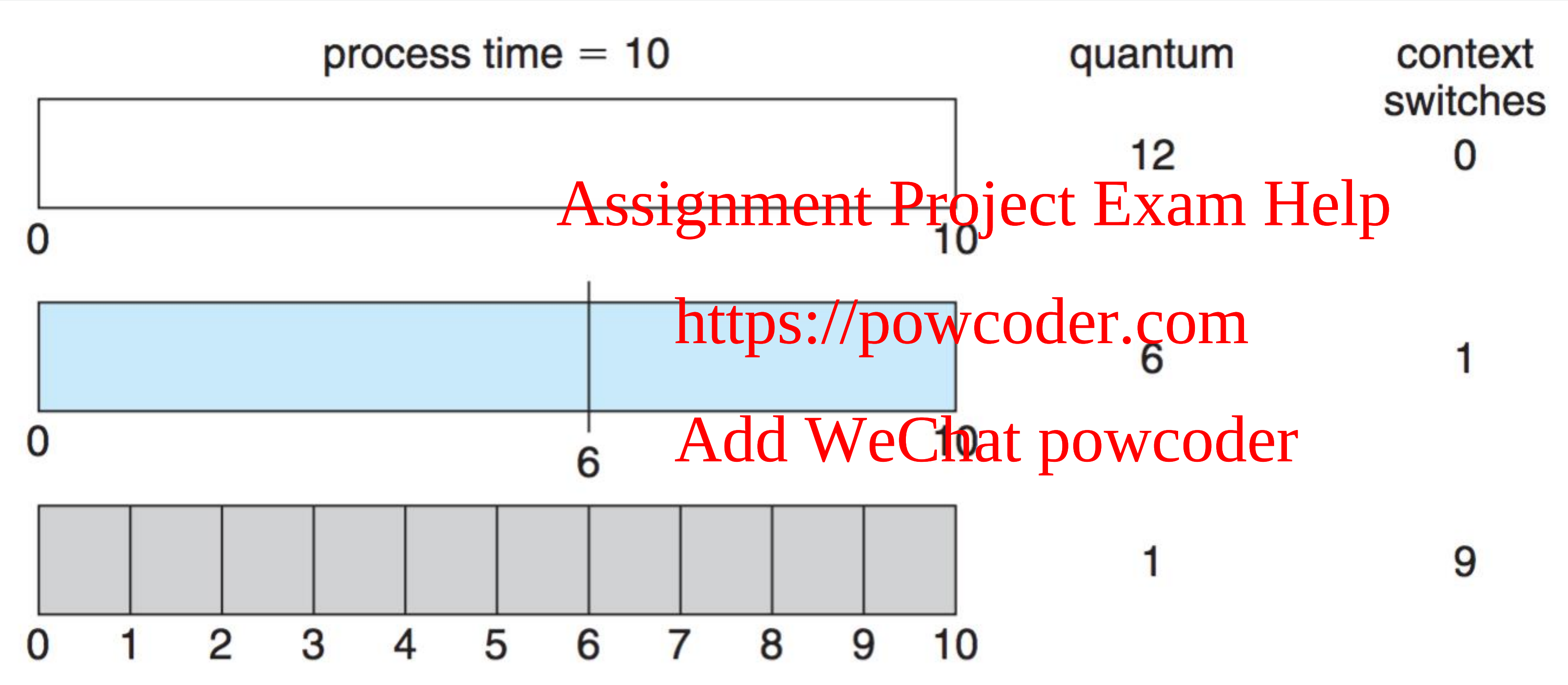
- FCFS + preemption at the end of a time slice (time quantum)
- Implementation: FIFO queue + Timer interrupt

Process	Burst time
P1	24
P2	3
P3	3



Time quantum: 4ms

How to choose the time quantum?



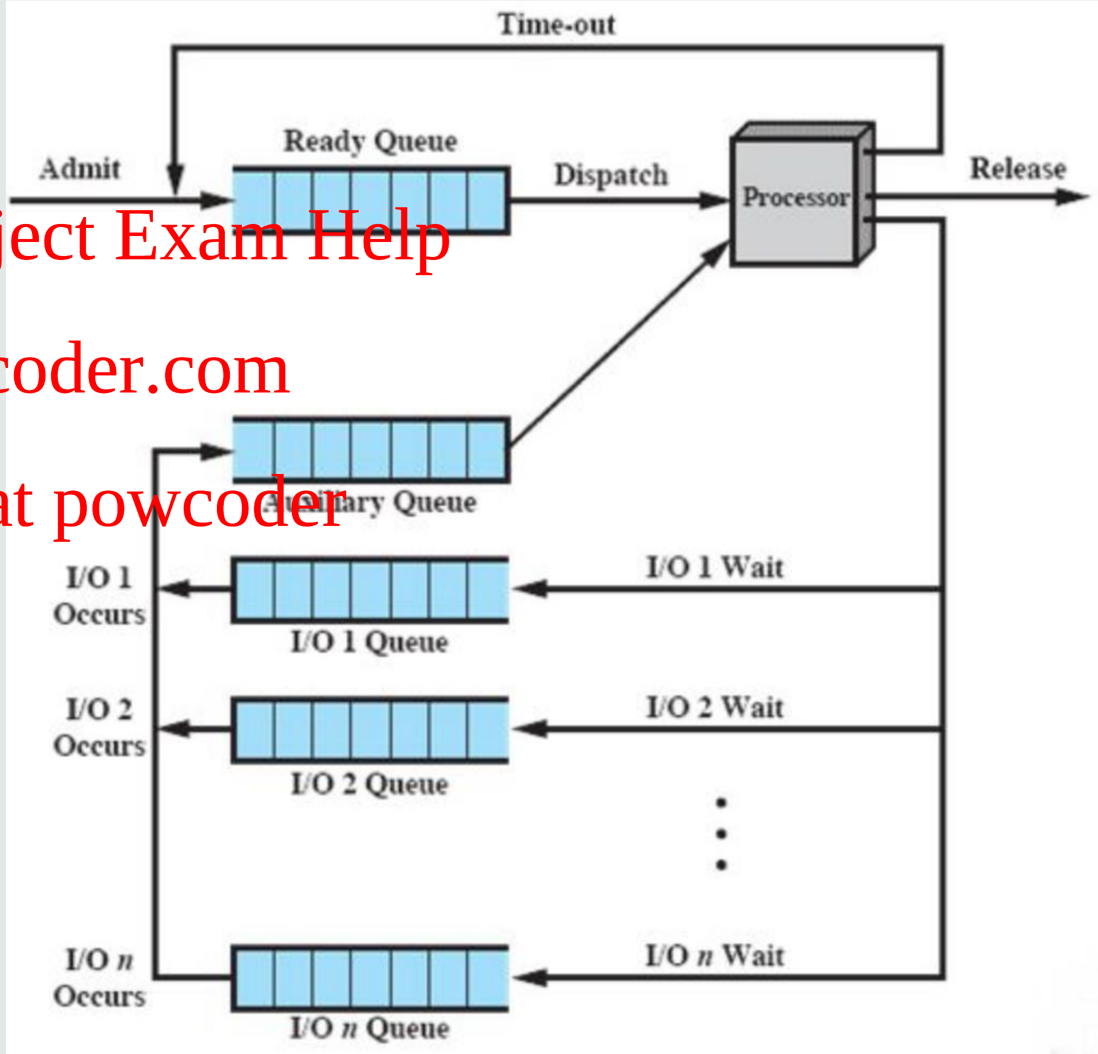
Quantum too large: Scheduling degenerates and works like FCFS

Quantum too short: Too many context switches

Rule of thumb: 80% of CPU bursts <= time quantum

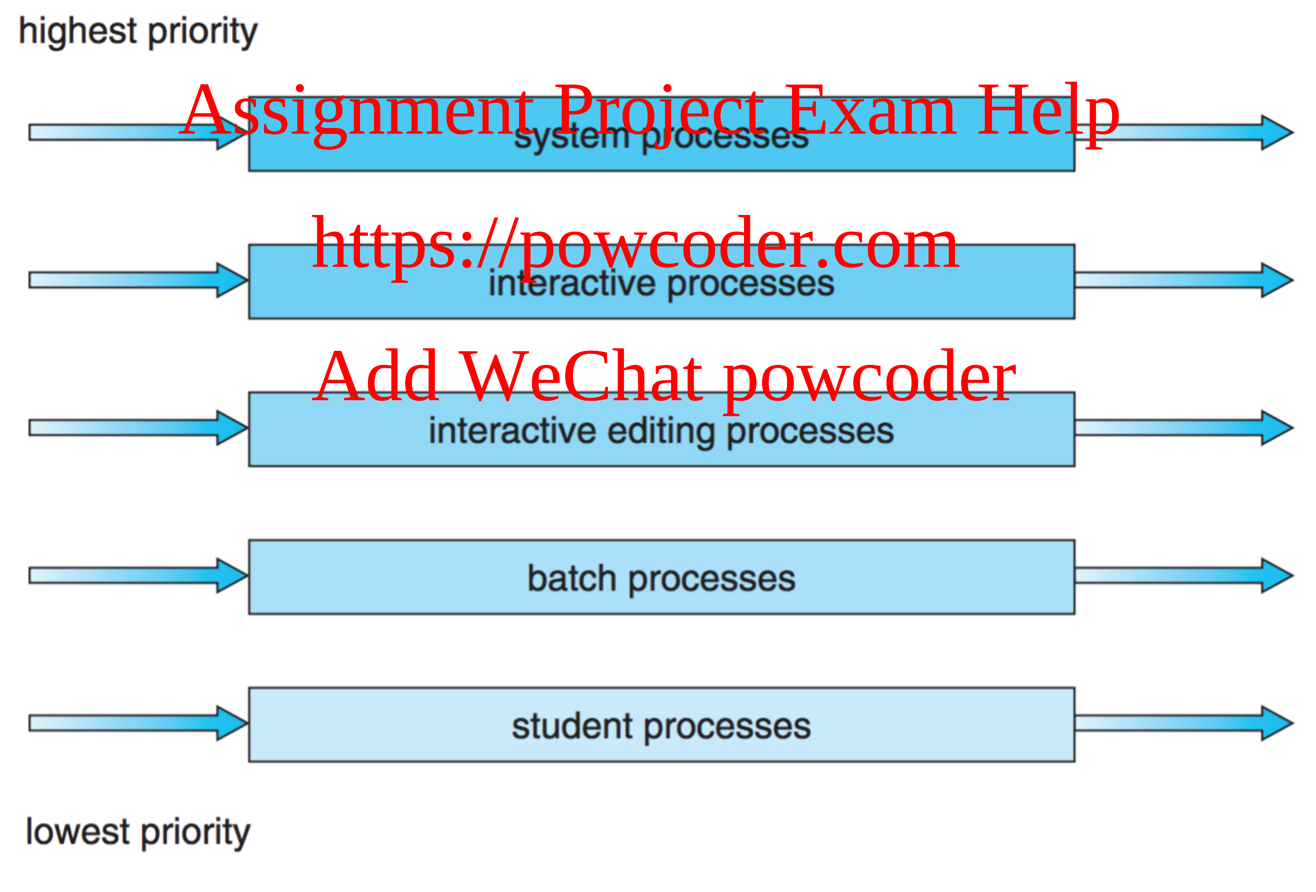
Virtual Round Robin

- Problem of RR
  - I/O-bound processes use only a small fraction of their time slice → spend a long time in the Ready queue
- VRR
  - Auxiliary queue → higher priority



Multi-level Queue Scheduling

- Separate processes according to their expected behaviour

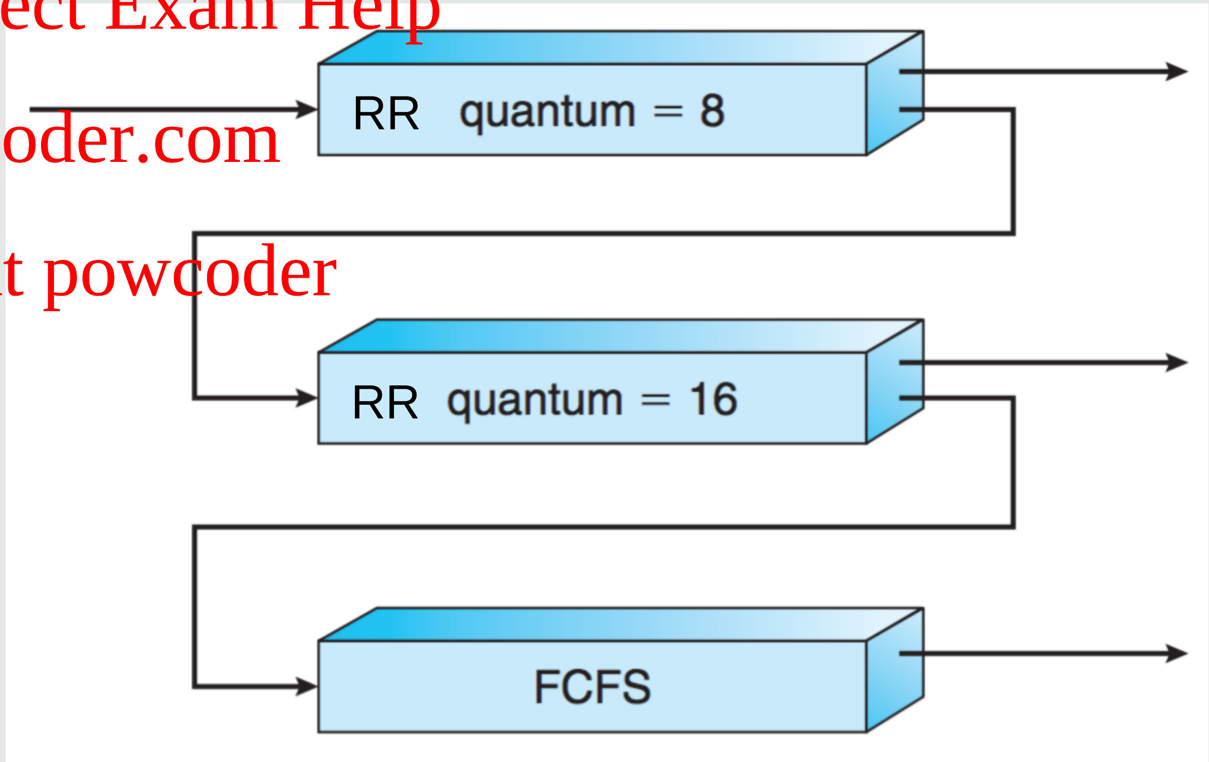


Multi-level Feedback Queue Scheduling

- Dynamically adapt to process behaviour

  → move processes between queues as they change

- Example:



I/O-bound processes get served quickly

CPU-bound processes get demoted

Parameterised Scheduling Algorithms

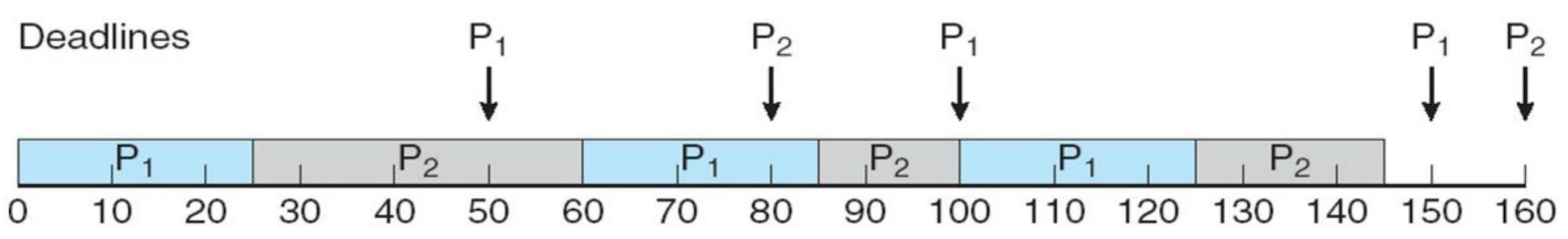
Scheduling policies can be flexibly designed and highly configurable

E.g. Multi-level Feedback Queues

- Number of levels in the scheduling system
- Methods for determining at which level a process is admitted
- Methods for upgrading and demoting processes between queues
- Choice of scheduling algorithm at every level
  - e.g. Round Robin
    - time quantum parameter

Real-time Scheduling

- Processes with periodic arrival times
- Deadlines
- Hard real-time: Guarantee that there are no deadline misses
- Soft real-time: Minimise number of deadline misses
- Preemptive scheduling with static or dynamic priorities



Example: Earliest Deadline First

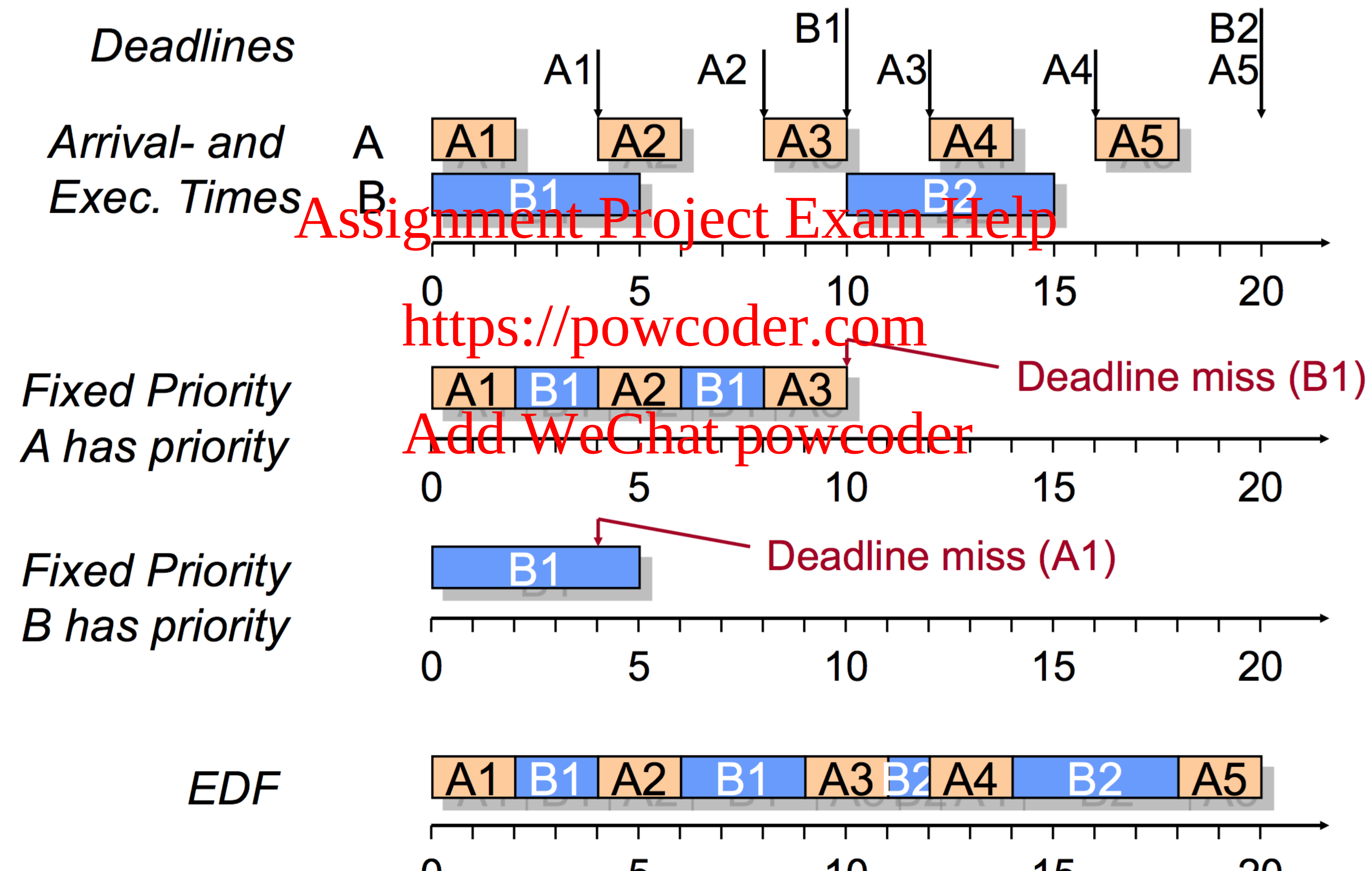


Image: Earliest Deadline First. P. Puscher, TU Wien.

Real-time Scheduling

- A set of tasks is schedulable if and only if $\sum_i \frac{C_i}{T_i} \leq 1$

where

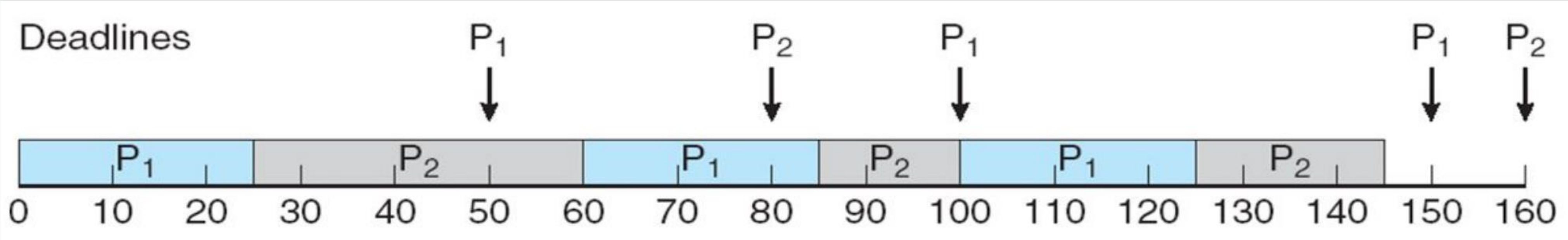
Worst-Case Execution time: C_i

Task deadline = period T_i

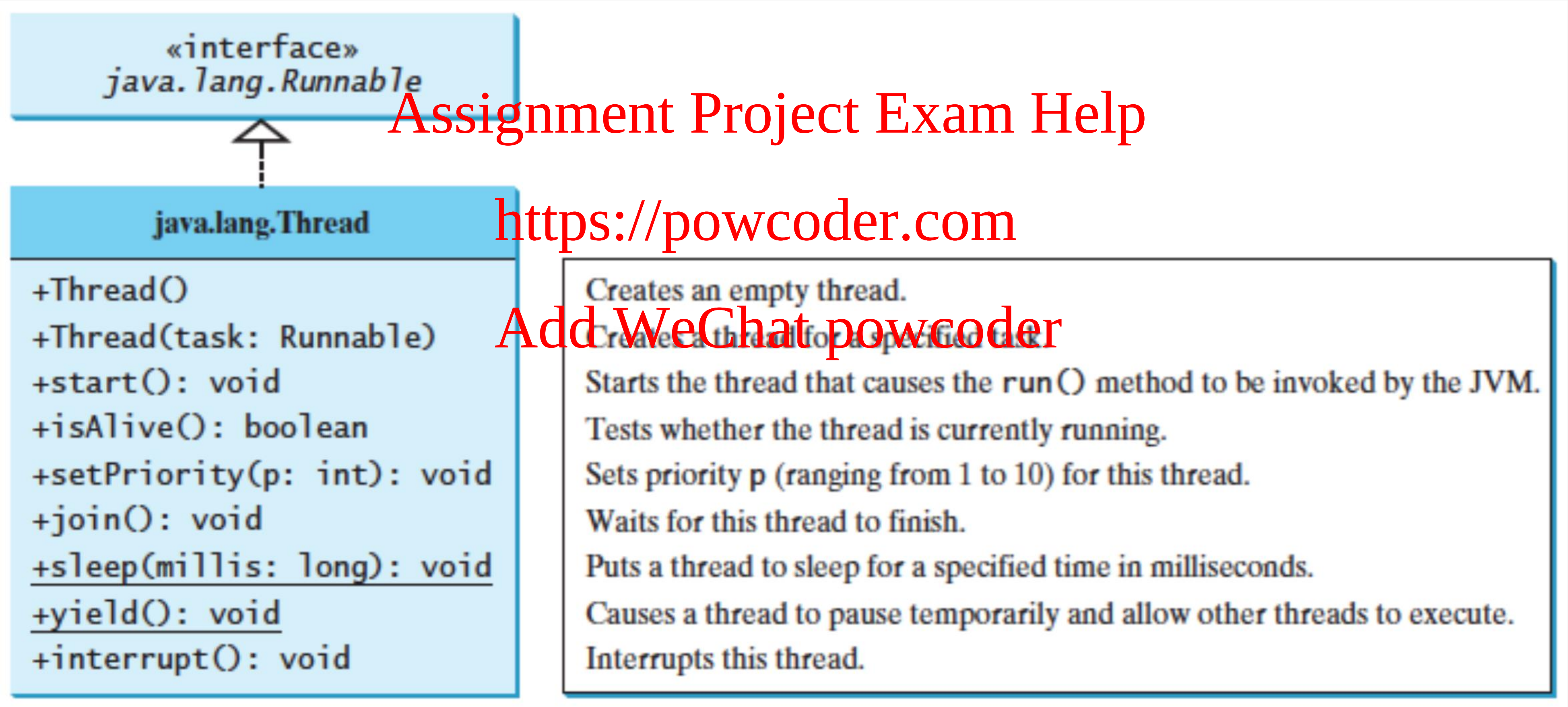
$$\frac{C_1}{T_1} = \frac{25}{50} = 0.5$$

$$\frac{C_2}{T_2} = \frac{35}{80} = 0.4375$$

$0.5 + 0.4375 < 1$ → all good!



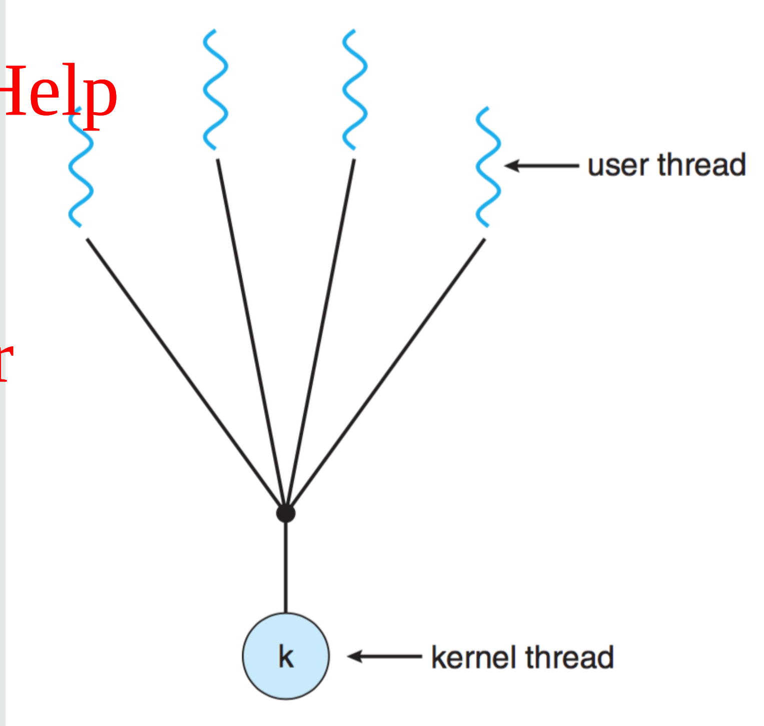
Java Thread Scheduling



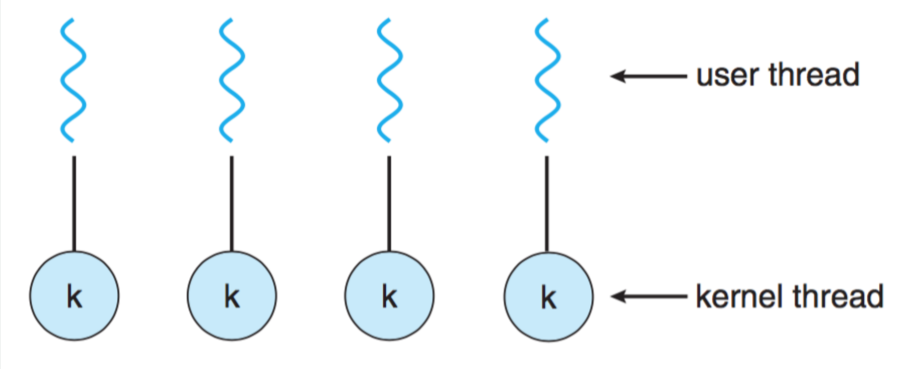
Java Thread Scheduling

Scheduling policy

- Early Java versions: user threads map to one kernel thread FCFS + priorities, preemptive



- Today:
  Depends on operating system,
  user threads map 1:1 to kernel-threads



Java Thread Scheduling

Scheduler is run when

- A thread terminates
- A higher priority thread becomes runnable (ready)
- A thread calls yield()

Summary

Scheduling algorithms

- Performance measures
- Round-Robin scheduling
- Multi-level queue scheduling
- Feedback scheduling
- Real-time scheduling

- Java thread scheduling

Read

- Tanenbaum & Bos., Modern Operating Systems
  - Chapter 2.4

- Silberschatz et al., Operating System Concepts
  - Chapter 5

- Further reading: Love. Linux Kernel Development:
  - Chapter 4

Next Lecture

- Introduction
- Operating System Architectures
- Processes
- Threads - Programming
- **Process Scheduling - Evaluation**
- Process Synchronisation
- Deadlocks
- Memory Management
- File Systems
- Input / Output
- Security and Virtualisation