UNIT IV · LESSON 03

# CPU Scheduling & Deadlocks

criteria · FCFS · SJF · SRTF · priority · round robin · multilevel queues · deadlock conditions · prevention · avoidance (Banker's) · detection

**30**
questions in Unit IV
OS · Networks · Cloud

## FORMULAS SCHEDULING CRITERIA

| Term | Formula / goal |
|---|---|
| **CPU utilisation** | maximise (40–90 %) |
| **Throughput** | processes completed per unit time (max) |
| **Turnaround time (TAT)** | completion − arrival (min) |
| **Waiting time (WT)** | TAT − burst time (min) |
| **Response time** | first run − arrival (min) |

- **Pre-emptive**: running process can be taken off (RR, SRTF, pre-emptive priority)
- **Non-pre-emptive**: runs until it blocks or ends (FCFS, SJF, non-pre-emptive priority)

## SUMMARY ALGORITHMS (CONTD.)

- RR: q very large → **FCFS**; q very small → too many context switches (processor sharing)
- RR response time ≤ (n − 1)·q

## SUMMARY ALGORITHMS

| Algorithm | Type | Key point |
|---|---|---|
| **FCFS** | non-pre-emptive | simple; **convoy effect** |
| **SJF** | non-pre-emptive | **minimum average WT** (optimal); needs burst estimate |
| **SRTF** | pre-emptive SJF | optimal among pre-emptive; starvation |
| **Priority** | either | **starvation** → fix with **ageing** |
| **Round robin** | pre-emptive | time quantum q; good response; time sharing |
| **Multilevel queue** | fixed queues | foreground (RR) / background (FCFS) |
| **Multilevel feedback queue** | processes move between queues | most general, complex |
| **HRRN** | non-pre-emptive | response ratio = (W + S)/S, avoids starvation |

- Burst prediction: $\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\, \tau_n$ (exponential averaging)

## SAME 4 PROCESSES WORKED EXAMPLE

| Process | Arrival | Burst | FCFS CT | SJF CT | SRTF CT | RR (q=2) CT |
|---|---|---|---|---|---|---|
| **P1** | 0 | 8 | 8 | 8 | 17 | 22 |
| **P2** | 1 | 4 | 12 | 12 | 5 | 12 |
| **P3** | 2 | 9 | 21 | 26 | 26 | 26 |
| **P4** | 3 | 5 | 26 | 17 | 10 | 23 |
| **Avg TAT** | | | **15.25** | **14.25** | **13.0** | **19.25** |
| **Avg WT** | | | **8.75** | **7.75** | **6.5** | **12.75** |

SJF: P1 0–8, P2 8–12, P4 12–17, P3 17–26. SRTF: P1 0–1, P2 1–5, P4 5–10, P1 10–17, P3 17–26 (best average). RR (q = 2, new arrivals queued before the pre-empted process): P1 P2 P3 P1 P4 P2 P3 P1 P4 P3 P1 P4 P3.

## STEPS GANTT-CHART METHOD

- Draw the timeline, note **completion time (CT)** of each process
- TAT = CT − AT; WT = TAT − BT; average over n
- Context switches = number of switches between different processes
- For priority: smaller number = higher priority (unless stated)

## DEFINITION DEADLOCK

- Set of processes each waiting for a resource held by another in the set
- **Four necessary conditions (Coffman)** — all must hold:

| Condition | Meaning |
|---|---|
| **Mutual exclusion** | resource used by one at a time |
| **Hold and wait** | holding some, waiting for more |
| **No pre-emption** | resource released only voluntarily |

## MORE MULTIPROCESSOR & REAL-TIME

- **Processor affinity** (soft / hard), **load balancing** (push / pull migration)
- Real-time: **Rate-monotonic** (static, shorter period = higher priority; utilisation bound $n(2^{1/n} - 1) \to 69\ \%$)
- **Earliest-deadline-first** (dynamic, up to 100 % utilisation)
- **Priority inversion** → priority-inheritance protocol
- Linux **CFS** (completely fair scheduler, red-black tree of virtual runtimes)

## DEFINITION DEADLOCK (CONTD.)

| Condition | Meaning |
|---|---|
| **Circular wait** | P0 → P1 → … → Pn → P0 |

- **Resource-allocation graph**: request edge P → R, assignment edge R → P
- Cycle with **single-instance** resources ⇒ deadlock; with multi-instance ⇒ **possible**

## 4 STRATEGIES HANDLING DEADLOCKS

| Strategy | How |
|---|---|
| **Prevention** | deny one Coffman condition |
| **Avoidance** | grant only if state stays **safe** (Banker's) |
| **Detection & recovery** | allow, detect cycles, then recover |
| **Ignore (ostrich)** | UNIX, Windows — rare, costly to prevent |

- Prevent hold-and-wait: request all at start (low utilisation, starvation)
- Prevent no-pre-emption: take resources from waiting processes
- Prevent circular wait: **total ordering** of resource types, request in increasing order
- Recovery: abort all / abort one by one; pre-empt resources (rollback, avoid starvation)

## FORMULA MINIMUM RESOURCES

- n processes each needing at most k units: deadlock-free if total R ≥ **n(k − 1) + 1**
- 3 processes need 2 tape drives each → minimum **4** drives
- R drives, each process needs k: max processes without deadlock = ⌊(R − 1)/(k − 1)⌋

## STARVATION · LIVELOCK RELATED PROBLEMS

- **Starvation**: indefinite waiting (priority scheduling, writers in readers–writers)
- **Livelock**: processes keep changing state but make no progress
- Deadlock ⇒ starvation for those processes; starvation does not imply deadlock

## AVOIDANCE BANKER'S ALGORITHM

- **Need = Max − Allocation**
- Safety: Work = Available; find Pi with $Need_i \le Work$; Work += $Allocation_i$; repeat
- **Safe state** has a safe sequence; unsafe ≠ deadlock, but may lead to one
- Request algorithm: Request ≤ Need, Request ≤ Available, pretend allocate, test safety

| Proc | Alloc A B C | Max A B C | Need A B C |
|---|---|---|---|
| **P0** | 0 1 0 | 7 5 3 | 7 4 3 |
| **P1** | 2 0 0 | 3 2 2 | 1 2 2 |
| **P2** | 3 0 2 | 9 0 2 | 6 0 0 |
| **P3** | 2 1 1 | 2 2 2 | 0 1 1 |
| **P4** | 0 0 2 | 4 3 3 | 4 3 1 |

- Available = (3 3 2) → safe sequence **P1, P3, P4, P0, P2** (one of several)
- Work: 332 → 532 → 743 → 745 → 755 → 10 5 7

## Q → A RAPID RECALL

| | |
|---|---|
| **Optimal for minimum average waiting time** | SJF |
| **Convoy effect occurs in** | FCFS |
| **RR with very large quantum becomes** | FCFS |
| **Fix for starvation in priority scheduling** | ageing |
| **Pre-emptive version of SJF** | SRTF |
| **Need matrix** | Max − Allocation |
| **Deadlock avoidance algorithm** | Banker's |
| **Ordering resources prevents** | circular wait |
| **Cycle in RAG with single instances means** | deadlock |
| **3 processes × 3 units each, minimum resources** | 7 |
| **Utilisation bound of RM for many tasks** | ln 2 ≈ 69 % |

## TRAP EXAM TRAPS

- Unsafe state is not necessarily deadlock
- With multiple instances, a cycle in the RAG does not guarantee deadlock
- Response time ≠ turnaround time
- SJF is optimal for average WT only when all jobs are available together (non-pre-emptive)
- Waiting time in RR counts all time spent in the ready queue, not just the first wait