OPERATING SYSTEM

Unit II — CPU & Multiprocessor Scheduling

Topics Covered in this Unit

Workload Assumptions & Formal Scheduling Metrics (Turnaround Time, Response Time, Waiting Time, Fairness)
Classic Uniprocessor Schedulers: FIFO / FCFS (Convoy Effect), SJF (Shortest Job First), STCF (Shortest Time-to-Completion First)
Round Robin (RR) Scheduling: Time Quantum Selection, Overhead Tradeoffs & Interactive Responsiveness
Incorporating I/O: CPU-I/O Burst Overlap & Maximizing System Utilization
Multi-Level Feedback Queue (MLFQ): Rules 1 to 5, Preventing Gaming, Priority Boost & Accurate Time Allotment
Multiprocessor Scheduling Hardware: Multi-Core Architecture, Cache Coherence (MESI Protocol) & Cache Affinity
Single-Queue Multiprocessor Scheduling (SQMS) vs. Multi-Queue Multiprocessor Scheduling (MQMS)
Load Balancing & Work Stealing in Multiprocessor Environments
Linux Kernel Multiprocessor Schedulers: The O(1) Scheduler vs. The Completely Fair Scheduler (CFS with vruntime & Red-Black Trees)
High-Yield University Exam Question Bank with Detailed Model Answers & Unit Summary

Compiled in a structured, easy-to-understand, textbook style format
Specially curated for B.Tech / B.E. / BCA / MCA / B.Sc Computer Science & Information Technology Students

COURSE SYLLABUS & MAPPING

[UNIT II SYLLABUS — PRESCRIBED UNIVERSITY CURRICULUM]

This section contains the official syllabus for Unit II. Verify with your university curriculum mapping.

Unit II Syllabus Breakdown:

Unit II Scheduling: Workload assumptions, scheduling metrics, response time, first in, first out (FIFO) shortest job first (SJF), shortest time-to-completion first (STCF), round robin, incorporating I/O, the multi-level feedback queue, the priority boost, attempt, better accounting, multiprocessor scheduling, synchronization, cache affinity, single-queue scheduling multi-queue scheduling, Linux multiprocessor schedulers.

• Course Code: _____ • Semester / Year: _____

TABLE OF CONTENTS

2.1 Workload Assumptions & Formal Scheduling Metrics

Fundamental Scheduling Metrics

1. Turnaround Time: Total time taken from job arrival to its final completion:

T_turnaround = T_completion - T_arrival

2. Response Time: Time from job arrival to the FIRST time it is scheduled on CPU (vital for interactive systems):

T_response = T_first_run - T_arrival

3. Waiting Time: Total accumulated time spent sitting in the ready queue:

T_waiting = T_turnaround - T_burst_time

Classic Uniprocessor Scheduling Algorithms

- **1. First-In, First-Out (FIFO / FCFS):** Non-preemptive. Schedules jobs in the exact order of arrival. Suffers from the Convoy Effect where short CPU-bound jobs queue behind a single massive long-running job, causing average turnaround time to spike.
- **2. Shortest Job First (SJF):** Non-preemptive. Runs the job with the shortest execution time first. Optimal for average turnaround time ONLY if all jobs arrive at the exact same instant (t=0). If a long job arrives first, newly arriving short jobs are blocked until it finishes.
- **3. Shortest Time-to-Completion First (STCF / Preemptive SJF / SRTF):** Preemptive. Whenever a new job arrives, if its remaining run time is less than the currently running job's remaining time, the current job is preempted. Mathematically optimal for turnaround time under arbitrary arrival times.
- **4. Round Robin (RR) Scheduling:** Runs a job for a fixed Time Slice (Quantum Q) and switches to the next job in the run queue. Optimizes Response Time drastically.
  - **• Time Quantum Tradeoff:** If Q is too small, context switch overhead dominates CPU time. If Q is too large, RR degrades into FIFO. Typical standard: Q = 10ms to 100ms (context switch overhead < 1% of Q).

Algorithm	Preemptive?	Turnaround Time Metric	Response Time Metric	Primary Weakness / Gotcha
FIFO / FCFS	NO	Poor (Convoy Effect)	Poor	Short jobs trapped behind long jobs.
SJF	NO	Optimal (if all arrive at t=0)	Poor	Suffers from arrival order; starvation of long jobs.

Algorithm	Preemptive?	Turnaround Time Metric	Response Time Metric	Primary Weakness / Gotcha
STCF / SRTF	YES	Provably Optimal	Moderate	High context switching overhead; starvation.
Round Robin (RR)	YES	Poor for identical jobs	Excellent (Interactive)	Poor turnaround time for equal-length jobs.

2.2 Incorporating I/O & CPU Utilization

Real-world processes alternate between CPU bursts and I/O bursts. When a process issues an I/O request (e.g., reading a disk block), it enters the Blocked state. The scheduler treats each sub-burst independently:

- • **Overlapping CPU and I/O:** While Process A waits for disk I/O, the scheduler assigns the CPU to Process B, ensuring 100% hardware resource utilization and preventing CPU starvation.

2.3 The Multi-Level Feedback Queue (MLFQ)

Definition: Multi-Level Feedback Queue (MLFQ - Corbato, 1962)
MLFQ is the gold-standard general-purpose scheduler that optimizes both Turnaround Time (for long batch jobs) and Response Time (for interactive I/O jobs) WITHOUT requiring advance knowledge of job burst lengths. It learns job behavior dynamically from past execution history.

The 5 Core Rules of MLFQ

- • **Rule 1:** If Priority(A) > Priority(B), A runs (B does not).
- • **Rule 2:** If Priority(A) = Priority(B), A and B run in Round Robin (RR) using the time slice of that priority queue.
- • **Rule 3:** When a job enters the system, it is placed at the topmost highest-priority queue (assumed to be a short interactive job).
- • **Rule 4 (Better Accounting / Anti-Gaming):** Once a job uses up its allotted time budget at a given level (regardless of how many times it relinquished the CPU for I/O), its priority is reduced by one level. (Prevents malicious jobs from gaming the scheduler by yielding at 99% of time slice).
- • **Rule 5 (The Priority Boost / Anti-Starvation):** After some periodic time period S, move all jobs in the system to the topmost queue. (Prevents long-running CPU jobs from starving, and allows batch jobs that become interactive to regain priority).

2.4 Multiprocessor Scheduling & Cache Architecture

1. Hardware Fundamentals: Cache Coherence & Affinity

- • **Cache Hierarchy:** Each CPU core possesses dedicated ultra-fast L1 and L2 hardware caches. When a process runs on Core 1, its working set is loaded into Core 1's cache.
- • **Cache Affinity:** A process should ideally continue running on the same CPU core across time slices to take advantage of warm cache lines and avoid costly DRAM reloads.
- • **Cache Coherence & MESI Protocol:** Hardware ensures shared memory consistency across multiple CPU caches using bus snooping and MESI state machines (Modified, Exclusive, Shared, Invalid).

2. SQMS vs. MQMS Architectures

Architecture	Single-Queue Multiprocessor Scheduling (SQMS)	Multi-Queue Multiprocessor Scheduling (MQMS)
Structure	All CPUs share a single global ready queue protected by a centralized lock.	Each CPU core maintains its own independent private ready queue.
Lock Contention	High lock contention and severe scalability bottlenecks as core count increases.	Zero lock contention during regular scheduling; extremely high scalability.
Cache Affinity	Poor cache affinity; processes constantly bounce randomly between different CPU cores.	Natural, strong cache affinity; processes stay pinned to their local core queue.
Load Balance	Perfect inherent load balance (all CPUs pull from same pool).	Susceptible to Load Imbalance (one CPU queue empty while another has 10 jobs). Solved via Work Stealing.

- • **Work Stealing in MQMS:** An underworked or idle CPU core periodically inspects another core's queue and steals ready processes to equalize load across all processors.

2.5 Linux Multiprocessor Schedulers: O(1) vs. CFS

1. The Linux O(1) Scheduler (Ingo Molnar, Linux 2.6)

- • **Mechanism:** Maintains two array-based priority queues per CPU: an Active Array and an Expired Array across 140 priority levels. When a task consumes its time slice, it moves to the Expired array.

When the Active array is empty, the kernel swaps pointers between Active and Expired in O(1) constant time.

2. The Completely Fair Scheduler (CFS - Default Linux Scheduler)

Definition: Completely Fair Scheduler (CFS)
CFS implements Weighted Proportional-Share Scheduling by modeling an 'Ideal Multi-tasking CPU'. It tracks each task's virtual runtime (vruntime), running the process that has received the least CPU time first using a self-balancing Red-Black Tree.

- **• Virtual Runtime (vruntime) Update Formula:**

vruntime += (Weight_default / Weight_i) × Physical_Runtime_Delta

where Weight is derived from the process's `nice` value (nice -20 [highest weight] to +19 [lowest weight]). High-priority jobs accumulate vruntime very slowly, receiving proportionally more physical CPU time.

- **• Red-Black Tree Data Structure:** CFS stores runnable tasks ordered by vruntime in an augmented Red-Black Tree. Selecting the next task to run takes O(1) time (leftmost node), while inserting/updating takes O(log N) time.

2.6 High-Yield University Exam Question Bank & Model Answers

Part A: Short Answer Questions (2–3 Marks)

- **Q1. What is the Convoy Effect in CPU scheduling? Which algorithm suffers from it?**

Answer: The Convoy Effect occurs in FIFO (FCFS) scheduling when a long CPU-intensive job holds the CPU, forcing many short I/O-bound jobs to wait in the ready queue, severely degrading average turnaround time and device utilization.

- **Q2. Define Turnaround Time and Response Time formally.**

Answer: Turnaround Time is the total time from job arrival to completion (T_turnaround = T_completion - T_arrival). Response Time is the time from arrival to first execution on the CPU (T_response = T_first_run - T_arrival).

- **Q3. State Rule 5 (Priority Boost) of MLFQ and explain why it is essential.**

Answer: Rule 5 resets the priority of all jobs in the system by periodically boosting them to the highest queue. It solves two critical flaws: (1) It prevents CPU-bound jobs at the bottom queues from starving, and (2) It allows processes that change behavior from batch to interactive to regain high priority.

- **Q4. Explain the role of vruntime in the Linux Completely Fair Scheduler (CFS).**

Answer: `vruntime` (virtual runtime) tracks the normalized CPU time consumed by a process scaled inversely by its `nice` priority weight. CFS always schedules the process with the smallest `vruntime` (stored at the leftmost node of the Red-Black tree) to guarantee fair CPU sharing.

Part B: Long Answer Questions (8–10 Marks)

- **Q5. Explain the Multi-Level Feedback Queue (MLFQ) scheduling algorithm in detail. State all 5 rules and explain how MLFQ prevents gaming and starvation.**

Model Answer Outline:

- 1. Motivation: Designing a scheduler that optimizes Turnaround Time without advance knowledge of job lengths.
- 2. Structural architecture: Multiple priority queues with different time slice lengths (shorter slices at top, longer slices at bottom).
- 3. Detailed explanation of all 5 formal rules:
- - Rule 1: Highest priority job runs.
- - Rule 2: Equal priority jobs run in Round Robin.
- - Rule 3: New jobs enter at top queue.
- - Rule 4: Total time allotment accounting preventing gaming of the scheduler.
- - Rule 5: Periodic priority boost preventing starvation.
- 4. Attack scenario: Malicious process relinquishing CPU at 99% of time slice and how accounting solves it.
- 5. Complete MLFQ queuing diagram showing job transitions.

- **Q6. Discuss Multiprocessor Scheduling challenges: Cache Coherence, Cache Affinity, Single-Queue (SQMS), and Multi-Queue (MQMS) architectures with Work Stealing.**

Model Answer Outline:

- 1. Multi-core hardware realities: L1/L2 caches, DRAM bus contention, and the MESI cache coherence protocol.
- 2. Concept of Cache Affinity: Performance costs of migrating processes between cores.
- 3. Single-Queue Multiprocessor Scheduling (SQMS): Architecture, simplicity, lock contention bottlenecks, and poor affinity.
- 4. Multi-Queue Multiprocessor Scheduling (MQMS): Per-CPU private queues, lock-free scalability, and affinity preservation.
- 5. Load Imbalance problem in MQMS and solving it via Work Stealing algorithms.

- 6. Comparative evaluation matrix between SQMS and MQMS.

2.7 Unit II Summary & Quick Revision Sheet

Scheduling Domain	Key Theoretical & Mathematical Takeaways for Exams
Metrics	Turnaround = T_comp - T_arr (SJF/STCF optimal). Response = T_first - T_arr (RR optimal).
Classic Schedulers	FIFO (Convoy effect), SJF (Non-preemptive), STCF (Preemptive optimal turnaround), RR (Time quantum Q).
MLFQ (5 Rules)	Learns from past. Short interactive jobs stay top; long batch jobs sink. Rules: R1/R2 (Priority/RR), R3 (Start top), R4 (Time allotment anti-gaming), R5 (Priority boost anti-starvation).
Multiprocessor	Cache Affinity + Coherence. SQMS (Global lock bottleneck) vs MQMS (Scalable private queues + Work Stealing).
Linux Schedulers	O(1) (Active/Expired pointer swap) → CFS (Completely Fair Scheduler: Red-Black tree tracking `vruntime` scaled by nice weight).