# Real-time Systems

CSE0420 – Embedded Systems

By

Z. Cihan TAYŞİ

## Outline

- Scheduling approaches
  - clock driven
  - priority driven
- Preemptive vs. Non-preemptive scheduling
- Offline scheduling
- Online scheduling

## Clock Driven Approach

- Decisions about which job executes and at what time are made at ***predefined times.***
- Typically, decision points are scheduled at regular intervals.
- When system starts, scheduler chooses and schedules the jobs that will execute until the next decision point.
- It then sets a hardware timer, and sleeps until timer expires.
- Scheduler awakes and then repeats the process.

## Round Robin

- Round-robin approach is typically used to schedule time-shared applications.
- When a job becomes ready to execute, it is added to a first-in-first-out (FIFO) queue.
- The job at head of queue then executes for at most a single ***time slice.***
- A **time-slice** is basic unit of time allocated to a job, typically on order of tens of milliseconds
- if job doesn't finish by end of time-slice, ***it is preempted*** and put at end of queue.
- If n jobs waiting, each job gets $1/n^{th}$ of the processor.

# Round Robin - Example

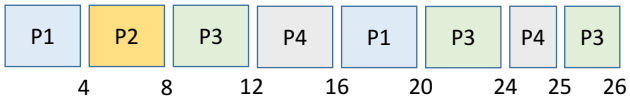
**Parameters**

- quatum
  - 4
- no priority based preemption
- what is the **average wait time** ?

**Workload Model**

| Process # | Release Time | Execution Time |
|---|---|---|
| 1 | 0 | 8 |
| 2 | 1 | 4 |
| 3 | 2 | 9 |
| 4 | 3 | 5 |

# Round Robin - Example (Solution)




| Process # | Release Time | Completion Time | Response Time |
|---|---|---|---|
| 1 | 0 | 20 | 20 |
| 2 | 1 | 8 | 7 |
| 3 | 2 | 26 | 24 |
| 4 | 3 | 25 | 22 |
| Avg. | - | - | 18.5 |

## Weighted Round Robin

- Weighted round-robin approach is similar but each job gets a ***weighted share*** of the processor.
- If a job has weight *wt*, then it is allocated *wt* time slots each round.
- Length of a round is equal to sum of weights of all waiting jobs.
- By ***changing weights***, we can speed up or slow down a job.
- Round-robin methods delays completion of all job, as each is only given a fraction of the processor.
- Method thus **not suitable** for scheduling precedence constrained jobs as response time of job chain could ***be quite large***.

## Priority Driven Approach – I

- Priority-driven approach maintains one or more queues of jobs ready to execute, sorted by priority.
- Method never leaves a resource idle on purpose.
- If resource becomes available, then highest priority waiting job is given the resource.
- **Scheduling decisions are made when a job becomes ready or a job completes.**
- How algorithm decides priority defines algorithm.
  - Could assign priority based on release time such as FIFO, and LIFO (last-in-first-out) algorithms.
  - Could also assign priority based on job execution time such as SETF (shortest-execution-time-first), and LETF (longest-execution-time-first) algorithms.

## Priority Driven Approach – II

- Most scheduling algorithms used in **non real-time systems** are priority-driven
  - Release time
    - First-In-First-Out
    - Last-In-First-Out
  - Execution time
    - Shortest-Execution-Time-First
    - Longest-Execution-Time-First
- Real-time priority scheduling assigns priorities based on deadline or some other timing constraint
  - Earliest deadline first
  - Least slack time first

## Preemptive vs. Non-preemptive

- Not known in general when preemptive is better than non-preemptive scheduling.
- Usually though, preemptive gives better results.
- In special case when all jobs have zero release time, preemptive is better if we ignore cost of preemption.

## Fixed and Dynamic Priority Algorithms

- Two classifications for algorithms that schedule periodic tasks:
  - Fixed-priority algorithms give same priority to each job in a given task.
    - Means priority of a given task is fixed relative to other tasks.
  - Dynamic-priority algorithms give different priorities to individual jobs in same task.
- In both cases, most algorithms give a fixed priority to a given job. i.e. once a job is given a priority, it is not changed.

## Earliest-Deadline-First Algorithm

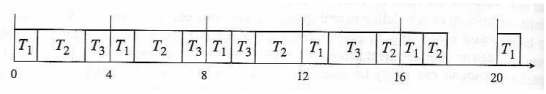
- Can choose to assign priorities based on deadlines.
- A common approach is earliest-deadline-first (EDF).
- For EDF, the earlier the deadline the higher the priority.
- Theorem
  - When preemption allowed and jobs do not contend for resources, then EDF algorithm is optimal.
  - Look for the prof at...

# Deadline-Monotonic Algorithm

- For deadline-monotonic (DM) algorithm, tasks with shortest relative deadline are given highest priority.
- Calculate DM schedule for system with tasks
  - $T_1 = (\Phi_1, p_1, e_1, D_1) = (50, 50, 25, 100)$,
  - $T_2 = (0, 62.5, 10, 20)$,
  - $T_3 = (0, 125, 25, 50)$.
- What is utilization of the system !?

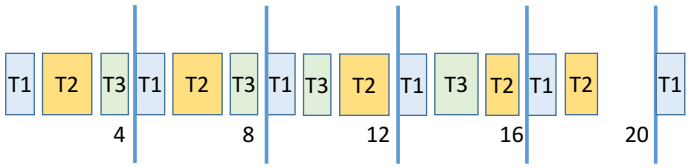
# Rate-Monotonic Algorithm

- For rate-monotonic (RM) algorithm, tasks with shortest period are given highest priority.
- Rate of job releases for a task is inverse of its period.
- Refer to a schedule produced by the algorithm as an **RM schedule**.
- Example below shows RM schedule for system with tasks
  - T1 = ($p_1$, $e_1$) = (4, 1), T2 = (5, 2), and T3 = (20, 5).
- W



# Rate-Monotonic Algorithm

- Example below shows RM schedule for system with tasks
  - T1 = ($p_1$, $e_1$) = (4, 1), T2 = (5, 2), and T3 = (20, 5).

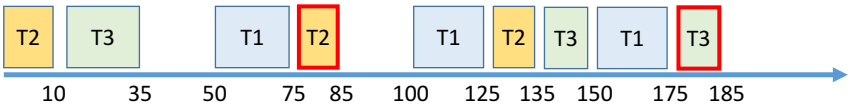



# RM versus DM Algorithm

- When relative deadline of each task is proportional to its period, then RM and DM are equivalent.
- When relative deadlines arbitrary, DM can sometimes find feasible schedule when RM fails.
- RM algorithm always fails when DM fails.
- Consider RM algorithm for tasks
  - T1 = ($\Phi_i$, $p_1$, $e_1$,$D_1$) = (50, 50, 25, 100),
  - T2 = (0, 62.5, 10, 20), and
  - T3 = (0, 125, 25, 50).

# RM vs. DM Algorithm

- Consider RM algorithm for tasks
  - T1 = ($\Phi_i$, $p_1$, $e_1$,$D_1$) = (50, 50, 25, 100),
  - T2 = (0, 62.5, 10, 20), and
  - T3 = (0, 125, 25, 50).




# Offline Scheduling

- Offline schedules are computed in advance before system begins to execute.
- Requires knowledge of release times and resource requirements for all jobs.
- Disadvantage is that it is inflexible.
- Advantages for deterministic systems is that the schedule is deterministic.
- Also, as scheduling is not done in real-time, complexity of scheduling algorithm not important.

# Online Scheduling

- Online schedules are computed on the fly.
- Distinguished by fact scheduler makes all decisions without knowledge of jobs that will be released in the future.
- Details of job become known only when job released.
- Only option when future workload is unpredictable.
- Advantages are more flexible and adaptable.
- Disadvantage is it is less likely to make best use of resources.