RTOS scheduling

- Admission control system: It verifies whether newly created task can be completed within deadline. If possible, then only task is executed further; otherwise task is rejected.
- All admitted task are guaranteed to be completed within deadline.
- RTOS use specialized scheduling algorithms
  - RMA (or RMS) -- Rate Monotonic Scheduling Algorithm (depends on "p")
  - DMA (or DMS) -- Deadline Monotonic Scheduling Algorithm (depends on "d")
  - EDF -- Earliest Deadline First
  - LSTF -- Least Slack Time First

Rate Monotonic Algorithm

- This algo is used for executing tasks with static priorities. The priorities are decided before execution of task and are not modified at runtime
- In this algorithm, lower is the period, higher is the priority.
- Example:
  - P1: t=20, d=50, p=50 (lower period => higher priority)
  - P2: t=35, d=100, p=100 (higher period => lower priority)
- CPU utilization = (20 / 50) + (35 / 100) = 0.75
- In general max CPU utilization can be 1 (100%). If utilization is more than 100%, the tasks are not schedulable. Such task must be rejected (by ACS).
- In RMA, max CPU utilization = n * (2 ^ 1/n - 1) -- Max 69.3% for many tasks. (Liu and Layland theorm)
- If CPU utilization is less than calculated max limit, then tasks can be definitely scheduled within deadline.
- If CPU utilization is beyond the calculated max limit, then deadline may miss for some task.
- Max CPU utilization for 2 task = 0.82; so P1 & P2 will work properly (within deadline).
- Example:
  - P1: t=20, d=50, p=50
  - P2: t=35, d=100, p=100
  - P3: t=20, d=100, p=100
- CPU utilization = (20 / 50) + (35 / 100) + (20 / 100) = 0.95
- Max CPU utilization for 3 task = 0.779; so P1, P2, & P3 task may not complete within deadline. They will be rejected.

Deadline Monotonic Scheduling

- Refer diagram

Earliest Deadline First

- The priority of the task is calcuated at run time (whenever a task is scheduled / rescheduled).
- Early the deadline, higher will be the priority.
- Priority of the task will keep changing as per deadlines.
- This algorithm can give 100% CPU utilization.
- Example:
  - P1: t=25, d=50, p=50

  - P2: t=35, d=80, p=80

Least Slack Time First

- Refer diagram

RTOS Performance Metrices

- An RTOS should quickly and predictably respond to the event.
- It should have minimum interrupt latency and fast context switching latency.

Three Models for Performance Measures

(i) Ratio of the sum of latencies of the tasks and Interrupt with respect to the sum of the execution times. (ii) CPU load for how much time CPU not idle (iii) Worst-Case Execution time with respect to mean execution time.

Interrupt latencies

- Interrupt and task execution latencies with respect to the sum of the execution times must be very small
  - sum of task & intr latencies : sum of task execution times
- There must be fast context switching .

CPU Load

- CPU Load = sum(T1/P1 + T2/P2 + ...) + sum(ISR Duration) + sum(RTOS Execution)
  - cpu load < 100% (to sched all tasks in deadlines)
  - For RMA sched : CPU load < 69.3 % (for m task)
- Each task gives a load to the CPU that equals the task execution time divided by the task period
- CPU load or system load estimation in the case of multitasking is as follows.
  - Suppose there are m tasks. For the multiple tasks, the sum of the CPU loads for all the tasks and ISRs should be less than 1
- CPU load equal to 0.1 (10%)— means the CPU is underutilized and spends its 90% time in a waiting mode.
- Since the executions times can vary or and the task periods vary, the CPU loads can also vary

Worst Case Execution Time

- worst case execution time : average execution time
  - ratio should be close to 1.
  - ie. worst case execution time should not differ much from avg time

Scheduling latency

- House keeping + scheduler execution time + dispacher(context restore) time
- if system is loaded with multiple tasks, sched latency may differ/ vary

Number of priorities

- more num of unique priorities will have more overhead on system (ie.more memory required and may need more time to select the task)