Real-Time Scheduling

Priority-driven scheduling of
periodic tasks

Outline

- Assumptions
- Scheduling Algorithms
- Schedulable Utilization and Optimality
- Schedulability Tests

Assumptions

- Independent periodic tasks
- No aperiodic or sporadic tasks
- Priority-driven scheduling
- Preemptibility of CPU
- Scheduling decision made when job is released or completed
- Static / hard real-time / uniprocessor
  - results can be used in more general systems

Scheduling Algorithms

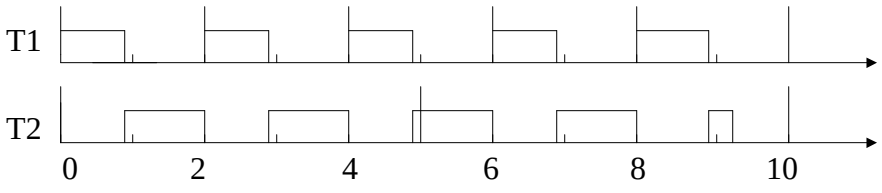
- Priority-driven
  - released and ready jobs in queue in nondecreasing priority order in job queue
  - scheduling decision made whenever job is released
- Algorithms differ in how prios are assigned

Fixed vs. Dynamic Priority

- Two categories of priority assignment
- Fixed:
  - assigns same prio to all jobs in each task
  - prio does not change (fixed)
  - Ex: rate-monotonic
- Dynamic:
  - may assign different prios to different jobs in a task
  - prios can change with time
  - Ex: earliest-deadline first

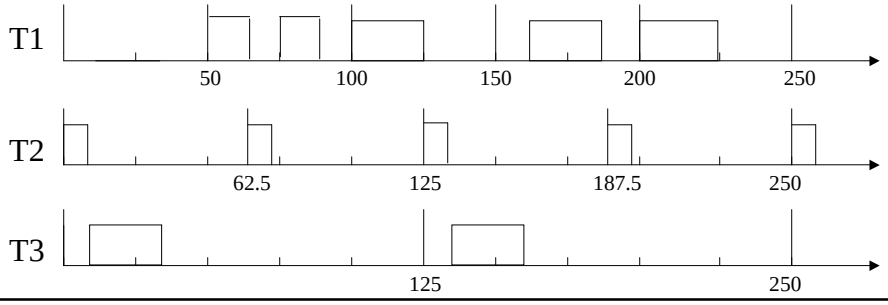
Fixed Priority Scheduling

- Rate-monotonic (RM)
  - shorter period = higher priority
  - Ex: T_1 = (2, 0.9); T_2 = (5, 2.3)



Fixed Priority Scheduling

- Deadline monotonic (DM)
  - assigns prios according to relative deadlines
  - shorter relative deadline = higher prio
  - **Ex:** T1=(50, 50, 25, 100); T2=(0, 62.5, 10, 20); T3=(0, 125, 25, 50)

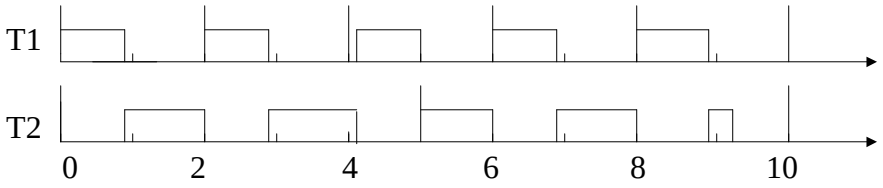


Dynamic Priority Assignment

- Earliest deadline first (EDF)
  - assigns prio to jobs in tasks by relative deadline
  - relative prios of tasks change as new jobs are released and completed
- Least slack time (LST)
  - slack = d - t - x
    - d = deadline
    - t = time at which slack is computed
    - x = execution of remaining portion
  - scheduler checks slack whenever new job is released

Dynamic Priority Assignment

- Earliest Deadline First (EDF)
  - closest deadline = highest priority
  - Ex: $T_1 = (2, 0.9)$; $T_2 = (5, 2.3)$



Schedulable Utilization

- *Def:* A scheduling algorithm can feasibly schedule any set of periodic tasks on a processor if the total utilization of the tasks is equal to or less than the *schedulable utilization (SU) of the algorithm.*
- An algorithm with SU = 1 is optimal.
- If task system has total utilization > 1, not feasibly schedulable

Optimality

- *Thm*: Both EDF and LST algorithms are optimal in the uni-processor case.
- *Proof*: follows from the fact that a schedule produced by any third algorithm can always be transformed to either EDF or LST by appropriate pairwise swappings of tasks.

Optimality

- Rate-Monotonic
  - *Thm*: The RM algorithm is optimal among all fixed priority algorithms whenever the relative deadlines of the tasks are proportional to their periods.
  - *Proof*: based on the idea that given a feasible test of tasks, NOT scheduled using RM, it can be transformed into a RM schedule by switching order, without missing any deadlines

Optimality

- Deadline-monotonic
  - Thm. 6-4: A system of independent, preemptable periodic tasks that are in phase and have relative deadlines equal to or less than their respective periods can be feasibly scheduled on one processor according to the DM algorithm whenever it can be feasibly scheduled according to any fixed priority algorithm.
  - *Proof*: Same idea as RM proof.

Schedulable Utilization of EDF

- The schedulable utilization of EDF $U_{EDF}(n)$ for n independent, preemptible periodic tasks with relative deadlines equal to or larger than their periods is equal to one. (see proof in Liu text)
- Same for LST

Schedulability Test for EDF

$$\sum_{k=1}^{n} \frac{e_k}{\min(D_k, p_k)} \leq 1$$

e = execution time; D = relative deadline p = period

- If condition is satisfied - set of tasks is schedulable by EDF
- If condition is not satisfied:
  - If $D_k >= p_k$ for all $k=1$ to n
    - reduces to U<=1 - necessary and sufficient condition for schedulability
  - If $D_k < p_k$ for some k
    - only sufficient condtion
    - system may be schedulable

Schedulable Utilization of RM and DM

- Sufficient schedulability test for RM and DM
- Thm: A system of n independent, preemptible periodic tasks with relative deadlines equal to their periods, can be feasibly scheduled on a single processor according to the RM algorithm if:

  $U(n) <= U_{RM}(n)=n(2^{1/n}-1)$
- No information if $U(n) > U_{RM}(n)$
  - must use time-demand analysis in this case

Schedulability Tests for RM and DM

- Check schedulability of one task T_i at a time
  - test if response time of all of its jobs are <= D_i
- A *critical instant* of a task T_i is a time instant which is such that:
  - the job in T_i released at the instant has the max. response time of all jobs in T_i, if the response time of every job in T_i is <= D_i
  - response time of the job released at the instant is >D_i if the response times of some jobs in T_i exceed D_i
- W_i = max response time of jobs in T_i

Schedulability Tests for RM and DM

- *Thm 6-5*: In a fixed priority system where every job completes before the next job in the same task is released, a critical instant of an task T_i occurs when on of its jobs $J_{i,c}$ is released at the same time with a job of every higher priority task.
- *Proof*: see Liu textbook
  - W_i = smallest value of t that satisfies the following:

$$t = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

Schedulability Tests for RM and DM

- Time-Demand Analysis
  - Compute total demand for processor time by a job released at a critical instant and by all the higher priority tasks as a function of time from the critical instant
  - Then check if this demand can be met before the deadline of the job

Schedulability Tests for RM and DM

- Time -Demand Analysis
  - Consider one task at a time starting with T_1 (highest priority) in decreasing prio order
  - For job T_i, let time t_0 be a critical instant
    - at time t_0+t, $t>=0$, total time demand ($w_i(t)$) of T_i and all higher prio jobs released in $[t_0,t]$ is:

$$w_i(t) = e_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k \qquad for\ 0 < t \le p_i$$

Schedulability Tests for RM and DM

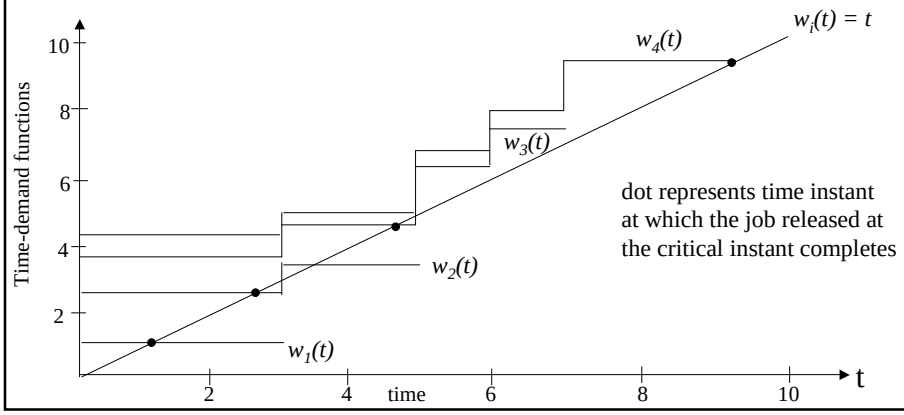
- Time-Demand Analysis
  - T_i can meet its deadline t_0+D_i if at some time $t_0+t<=t_0+D_i$, the supply of processor time (t) becomes equal to or greater than the demand for processor time ($w_i(t)$).
  - In other words: $w_i(t)<=t$ for some $t<=D_i$ where $D_i<=p_i$
  - If $w_i(t)>t$ for all $0<t<= D_i$, then T_i cannot complete by its deadline
  - Hence, the system of tasks is infeasible

Schedulability Tests for RM and DM

- Time-Demand Analysis - of task T_i
  - 1) compute the time-demand function $w_i(t)$
  - 2) check whether the inequality
    - $w_i(t) <= t$
    - is satisfied for values of t that are equal to:
      - $t = jp_k;$ $k=1,2,\ldots,i;$ $j=1,2,\ldots,floor(min(p_i,D_i)/p_k)$
    - If the inequality is satisfied at any of these instants, then task T_i is schedulable
    - If the inequality is not satisfied at any of these instants, then task T_i is not schedulable

Schedulability Tests - Example

- T_1=(3,1), T_2=(5,1.5), T_3=(7,1.25), T_4=(9,0.5)
- total utilization =1/3 + 1.5/5 + 1.25/7 + 0.5/9 = 0.87
- Using RM scheduling



Schedulability Tests - Example

- add a task T_5 = (10, 1)
- T_1=(3,1), T_2=(5,1.5), T_3=(7,1.25), T_4=(9,0.5)
- total utilization =1/3 + 1.5/5 + 1.25/7 + 0.5/9 + 1/10 = 0.97