Two Types of Priority-Driven Algorithms (1/2)

- A fixed-priority algorithm assigns the same priority to all the jobs in each task.
  - The priority of each periodic task is fixed relative to other tasks.
  1. Rate-Monotonic Algorithm
  2. Deadline-Monotonic Algorithm

Two Types of Priority-Driven Algorithms (2/2)

- A dynamic-priority algorithm assigns different priorities to the individual jobs in each task.
  - The priority of the task with respect to that of other tasks changes as jobs are released and completed.
  1. Task-level dynamic-priority (and job-level fixed-priority) algorithms, e.g., EDF algorithm.
  2. Job-level (and task-level) dynamic-priority algorithms, e.g., LST algorithm.

Outline

- Fixed-Priority vs. Dynamic-Priority Algorithms:
  1. ***Rate-Monotonic and Deadline-Monotonic Algorithms***
  2. Well-Known Dynamic Algorithms
  3. Relative Merits

Rate-Monotonic Algorithm

- Assign priorities to tasks based on their period: the shorter the period, the higher the priority.
- The rate (of job releases) of a task is the inverse of its period.
- We will refer to this algorithm as the RM algorithm for short and a schedule produced by the algorithm as an RM schedule.
- Example of an RM Schedule:
  $T_1 = (4, 1)$, $T_2 = (5, 2)$, $T_3 = (20, 5)$

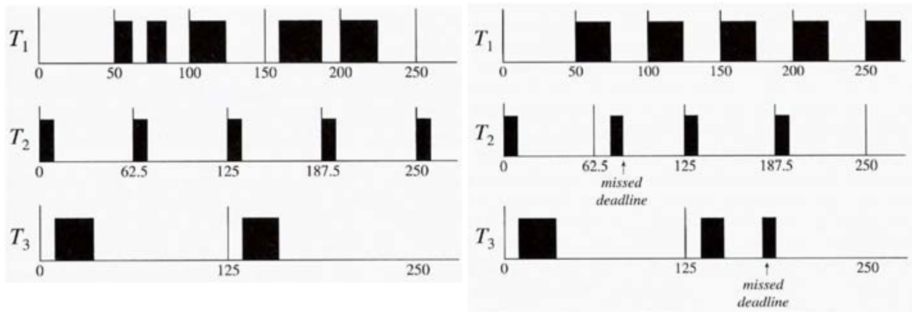
Deadline-Monotonic Algorithm

- Assign priorities to tasks according their relative deadlines: the shorter the relative deadline, the higher the priority.
- Notation: (*phase*, *period*, *execution time*, *deadline*)

Fixed-Priority Schedules

- $T_1 = (50, 50, 25, 100)$, $T_2 = (0, 62.5, 10, 20)$, $T_3 = (0, 125, 25, 50)$

(phase, period, execution time, deadline)



(a) DM $\tau_2 > \tau_3 > \tau_1$

(b) RM $\tau_1 > \tau_2 > \tau_3$

RM vs. DM

- When the relative deadline of every task is proportional to its period, the RM and DM algorithms are identical.
- When the relative deadlines are arbitrary, the DM algorithm performs better in the sense that it can sometimes produce a feasible schedule when the RM algorithm fails, while the RM algorithm always fails when the DM algorithm fails.

Outline

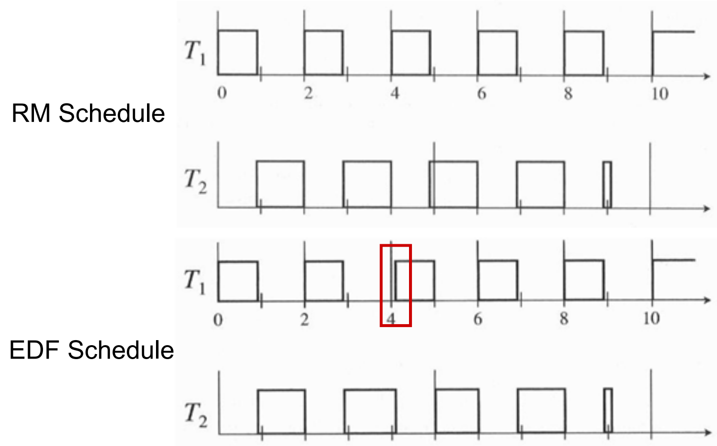
- Fixed-Priority vs. Dynamic-Priority Algorithms:
  1. Rate-Monotonic and Deadline-Monotonic Algorithms
  2. ***Well-Known Dynamic Algorithms***
  3. Relative Merits

Earliest Deadline First Algorithm

- The Earliest-Deadline-First (EDF) algorithm assigns priorities to individual jobs in the tasks according to their absolute deadlines.

EDF Schedule vs. RM Schedule

- $T_1 = (2, 0.9)$, $T_2 = (5, 2.3)$
- EDF algorithm is a task-level dynamic-priority algorithm, but a job-level fixed-priority algorithm.



Least-Slack-Time First Algorithm

- Least-Slack-Time First (LST) algorithm
  - At time t, the slack of a job whose remaining execution time is x and whose deadline is d is equal to $d - t - x$.
  - The scheduler checks the slacks of all the ready jobs each time a new job is released and orders the new job and the existing jobs on the basis of their slacks: the smaller the slack, the higher the priority.
  - LST algorithm is a job-level dynamic-priority algorithm.

Nonstrict LST vs. Strict LST

- Nonstrick LST: scheduling decisions are made only when jobs are released or completed.
- Strict LST: reassigns priorities to jobs whenever their slacks change relative to each other.
- The run-time overhead of the strict LST algorithm includes the time required to monitor and compare the slacks of all ready jobs as time progresses.
- By letting jobs with equal slacks execute in a round-robin manner, these jobs suffer extra context switches.

Outline

- Fixed-Priority vs. Dynamic-Priority Algorithms:
  1. Rate-Monotonic and Deadline-Monotonic Algorithms
  2. Well-Known Dynamic Algorithms
  3. ***<u>Relative Merits</u>***

Performance Criterion

- A criterion we use to measure the performance of algorithms used to schedule periodic tasks is the schedulable utilization.
- Schedulable Utilization of the Algorithm: 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 of the algorithm.
- Since no algorithm can feasibly schedule a set of tasks with a total utilization greater than 1, an algorithm whose schedule utilization is equal to 1 is an optimal algorithm.

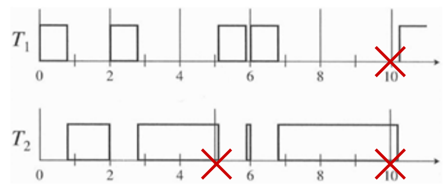
Advantage of Fixed-Priority Algorithms

- Although optimal dynamic-priority algorithms outperform fixed-priority algorithms, an advantage of fixed-priority algorithms is predictability.

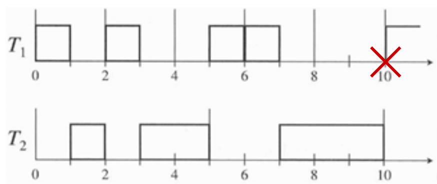
➡ When tasks have fixed priorities, overruns of jobs in a task can never affect higher-priority tasks!

Unpredictability and Instability of the EDF Algorithm

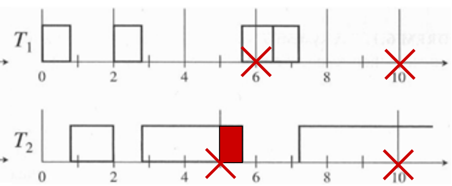
- There is no easy test, short of an exhaustive one, that allows us to determine which tasks will miss their deadlines and which tasks will not.



(b) $T_1 = (2, 0.8)$, $T_2 = (5, 3.5)$; $U = 1.1$



(a) $T_1 = (2, 1)$, $T_2 = (5, 3)$; $U = 1.1$



(c) $T_1 = (2, 0.8)$, $T_2 = (5, 4.0)$; $U = 1.2$

Disadvantages of the EDF Algorithm

- Unpredictable during an overload.
- If the execution of a late job is allowed to continue, it may cause some other jobs to be late.

➡ The scheduler should either lower the priorities of some or all the late jobs, or discards some jobs if they cannot complete by their deadlines and logs this action.

Outline

- Assumptions
- Fixed-Priority vs. Dynamic-Priority Algorithms
- *Maximum Schedulable Utilization*
- Optimality of the RM and DM Algorithms
- A Schedulability Test for Fixed-Priority Tasks with Short Response Times
- Schedulability Test for Fixed-Priority Tasks with Arbitrary Response Times
- Sufficient Schedulability Conditions for the RM and DM Algorithms

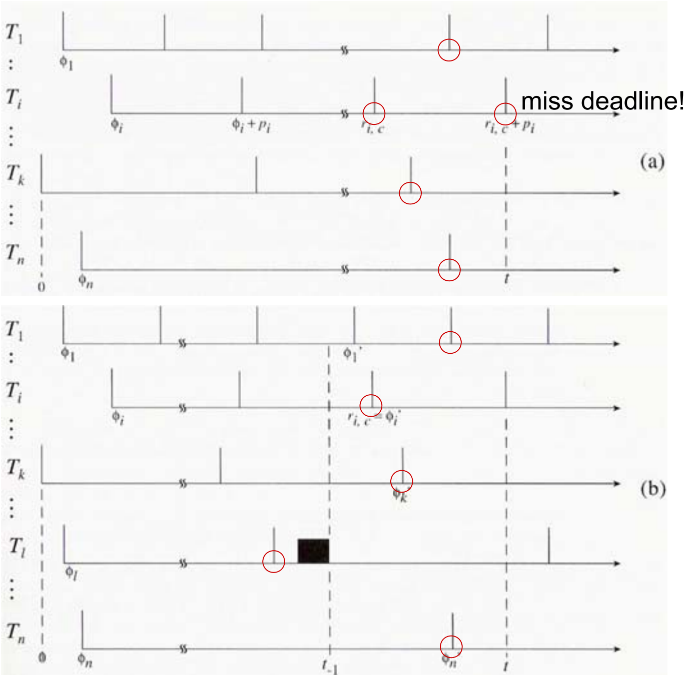
Definition

- Suppose a task set is scheduled by *S* scheduling algorithm, the schedule for the task set is feasible if all the jobs in each task can meet their corresponding deadlines under *S* scheduling. A task set is schedulable if there exists a feasible schedule.
- At any time t, the current period of a task is the period that begins before t and ends at or after t.
- We call the job that is released in the beginning of the current period the current job.

Schedulable Utilizations of the EDF

- **Theorem 1.** A system T of independent, preemptable tasks *with relative deadlines equal to their respective periods* can be feasibly scheduled on one processor if and only if its total utilization is equal to or less than 1.

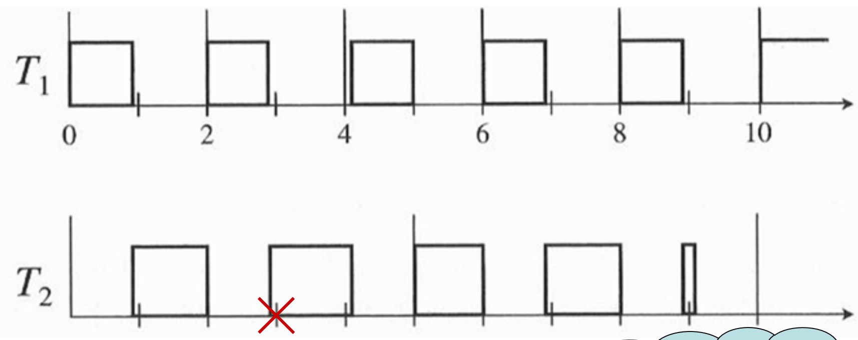
  Proof. Please see the handout.



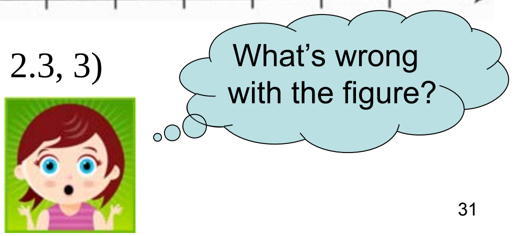
From Theorem 1, We Know That…

- A system of independent, preemptable periodic tasks with *relative deadlines longer than their periods* can be feasibly scheduled on a processor as long as the total utilization is equal to or less than 1.
- The schedulable utilization $U_{EDF}(n)$ of the EDF algorithm for n independent, preemptable periodic tasks with relative deadlines equal to or larger than their periods is equal to 1.

How about deadline less than period?



- $T_1 = (2, 0.9)$, $T_2 = (5, 2.3, 3)$



Density of the System

- We call the ratio of the execution time e_k of a task T_k to the minimum of its relative deadline D_k and period p_k the density of the task. In other word, the density of T_k is $e_k / \min(D_k, p_k)$.
- The sum of the densities of all tasks in a system is the density of the system and is denoted by Δ.
- When $D_i < p_i$ for some task T_i, $\Delta > U$.