Real Time systems 2

Initializing valid schedule

Initializing Valid Schedule

- Feasible schedule: is a valid schedule that is all tasks meet their respective time constraints in the schedule (its deadline).

- Schedulable Job: we say that a set of jobs is schedulable if the scheduling algorithm used, always produces a feasible schedule.

- Timing Constraints can be divided into two types:
- Hard timing constraint or hard deadline if the failure considered to be fatal fault.
- Soft timing constraint or soft deadline its not serious harm, only the system overall performance becomes poor.

Initializing Valid Schedule

Our task model is as follows:

We have a set of tasks T={T1, T2, ….., Tn}. For each task Ti we are given the worst-case execution time ei,

the deadline di, and

the release time ri.

Initializing Valid Schedule

- In addition, we are given the following relations between every pair of tasks:

1. Ti PRECEDES Tj is TRUE if Ti is in the precedence set of Tj, that is, if Tj needs the output of Ti and we cannot start executing Tj until Ti has finished executing.

2. Ti EXCLUDES Tj is TRUE if Ti is not allowed to preempt Tj.

3. Ti PREEMPTS Tj is TRUE if, when Ti is ready to run and Tj is currently running, Tj is always preempted by Ti.

Initializing Valid Schedule

- A task is said to be eligible to run at time t if the following properties are satisfied:
- All tasks Tj such that Tj PRECEDES Ti have completed by time t and ,there is no as-yet-unfinished task Tk that was started before t, and such that Tk EXCLUDES Ti

Initializing Valid Schedule

- A Schedule is said to be valid if it satisfies the following properties:
- V1. The processor is not idle if there are one or more tasks that are ready to run.
- V2. Exclusion, Precedence and preemption constraints are all satisfied throughout the schedule.

Initializing Valid Schedule

- Now according to EDF algorithm:
- If two tasks are both eligible to run (under the constraints) and have identical deadlines, the tie is broken on the basis of which one has the greater execution time.

Initializing Valid Schedule

An eligible task Tj will not run if there is a task Ti that is as yet unfinished but eligible to run, such that:

1- Ti PREEMPTS Tj,

2- [di $<$ dj] and NOT [Tj PREEMPTS Ti], and

3- [di=dj] and NOT [Tj PREEMPTS Ti] and ei>ej.

The algorithm for generating a valid Initial Schedule

t=0;

While (there are still unfinished tasks) do

if (∃i : t = r' V t=f(i) then

select for execution the highest priority eligible task with the minimum deadline.

if more than one eligible task has the same minimum deadline, break the ties according to their execution times. Giving priority to the one with the greatest execution time.

Endif

t=t+1

endwhile

Example:

- Consider a four-task system with the following parameters:

Task	T1	T2	T3	T4
ri	1	0	14	13
Ei	1	10	2	3
Di	5	30	18	25

- Suppose no precedence conditions exist, and that the only EXCLUDE relation is T1 EXCLUDES T2.

Example

- The valid initial schedule will be generated as follows:
- Task T2 release at time 0.
- It is scheduled to start running at that point.
- At time 1, T1 arrives and is prevented from preempting T2 due to the EXCLUDES relation.
- At time 10 T2 finishes. T1 is started and runs until 11.
- The CPU then is idle until 13.
- At 13 T4 starts .
- T3 arrives at 14 and because it has earlier deadline, it preempts T4 and execute to completion at time 16.
- T4 then resumes and complete at 18.

Initializing Valid Schedule

- Example:

- T1 misses its deadline because it must wait for T2 to finish.

- A Processor ***Busy Period*** is a time interval during which the processor is continuously busy.

- In the previous example, the processor busy periods for the valid initial schedule are [0,11] and [13, 18]

Initializing Valid Schedule

- Our next step is to check if any tasks have missed their deadlines.
- If none of them has do so, we are done.
- If some deadlines have been missed, then we try to rectify this situation by reordering the tasks in the schedule.
- We can make the CPU idle until time 1 and then start T1 then T2 after T1 has finished.

Initializing Valid Schedule

- Note that it is only useful to reorder the tasks in the same busy period as the one that missed its deadline.
- T3 and T4 arrive after finishing T1 and T2 , so they don’t affect the missing of T1.

HW:

- Consider a seven-task system with the following parameters:

Task	T1	T2	T3	T4	T5	T6	T7
ri	0	2	1	1	14	15	15
Ei	3	4	3	2	2	3	2
Di	13	14	14	14	18	18	22

- Suppose no precedence conditions exist, and that the only EXCLUDE relation is T2 EXCLUDES T3.