Report Title Here
Mälardalen University

Emil Broberg
School of Innovation, Design and Engineering
Mälardalen University, Västerås, Sweden
Email: ebg20007@student.mdu.se

I. Question 1

A. Part 1

1) Sufficient and necessary schedulability test: A suffiient schedulability test provide a guarantee of schedulability when passed. If the test is fulfilled, it ensures that the task set is schedulable. However if it is not passed, it does not provide any conclusive answer about schedulability. On the other hand, a necessary schedulability test provides a guarantee of unschedulability when failed. Conclusively, an exact schedulability test, which is both sufficient and necessary, can always guarantee whether the task set is schedulable or not.

2) Feasible schedule: A feasible schedule is simply a schchedule for a task set that is schedulable in a way so that all tasks completed within their deadlines. In other words, it is a schedule where all tasks meet their timing requirements without violating any system constraints.

3) Task and task instance: A task is a set of instructions that are executed within a larger program by a processor. A task instance refers to a single execution or occurrence of a task.

4) Task and processor utilization factor: The processor utilization factor is denoted by U, and is used to describe how much time of the processor is used. U is ranging from 0 to 1 or 0-100% describing the time utilization of the processor. The processor utilization factor is calculated by dividing the total execution time by the total period time of the task set.If U is larger than 1 (or 100%), the task set is not schedulable because the task set needs more processor time is required than is available.

5) Static and dynamic priority scheduling: Static priority scheduling and dynamic priority scheduling are two different approaches to online scheduling. Static priority scheduling is a scheduling method where the priority of a task is fixed and determined pre-run-time, it does not change during run-time. Dynamic priority scheduling is a scheduling method where the priority of a task is not fixed and can change during run-time. The change of the priorities in dynamic priority scheduling is based on the current state of the system.

6) Critical instant: A critical instant represents the worst-case scenario for a task in terms of meeting its deadline and timing constraints. The critical instant occurs when higher-priority tasks preempt the task in question, causing it to experience the longest possible delay before it can resume execution. If a task can meet its deadline at its critical instant, it ensures that the task will meet its timing requirements under all other conditions as well.

B. Part 2 - Offline scheduling

1) Schedulability: Offline scheduling offers more room for complexity and predictability. This is because the execution pattern is known and it can be changed in a predictable way to make processor utilization more efficient, fit more tasks in the task set etc. The schedulability is provable by construction since it will always execute the task set in the same way and can therefore be analyzed and verified before run-time.

2) Predictability: Offline scheduling is more predictable than online scheduling since the schedule is constructed before run-time. This means that the system will behave the exact way it was designed, unless some external factors affect the system e.g. hardware failure etc.

3) Flexibility: In offline scheduling, the schedule is predetermined and does not change once the system begins operating. This means that the system is not flexible and cannot adapt to changes in the environment or system.

4) Communication and synchronization: Communication refers to the exchange of data between tasks or components.

5) Jitter: Jitter is the variation in the execution time of a task. It is the difference between the best-case execution time and the worst-case execution time. A system with low jitter is more predictable than a system with high jitter.

6) Heuristic: If we are traversing a search tree, we can use heuristic strategy to choose which branch to traverse in the tree. We can choose for example to go with the task that has the earliest deadline, or the task with the shortest execution time etc. Heuristic is a way to make a decision based on some criteria, without knowing the exact outcome of the decision.

II. Question 2

A. a

Sufficient schedulability test for RM scheduling: $U <= n(2^{1/n} - 1)$
where n is the number of tasks in the task set and U is the processor utilization factor, $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$.

Task	T=D	C
A	3	1
B	5	2
C	2	0.5

Figure 1. Task set

1) Task set schedulable?: $U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{1}{3} + \frac{2}{5} + \frac{0.5}{2} = 0.98$
$U <= n(2^{1/n} - 1) = 3(2^{1/3} - 1) = 0.78$
Since the statement $U <= n(2^{1/n} - 1)$ is false in this case, the task set cannot be proven to be schedulable with this sufficient test method.

B. b

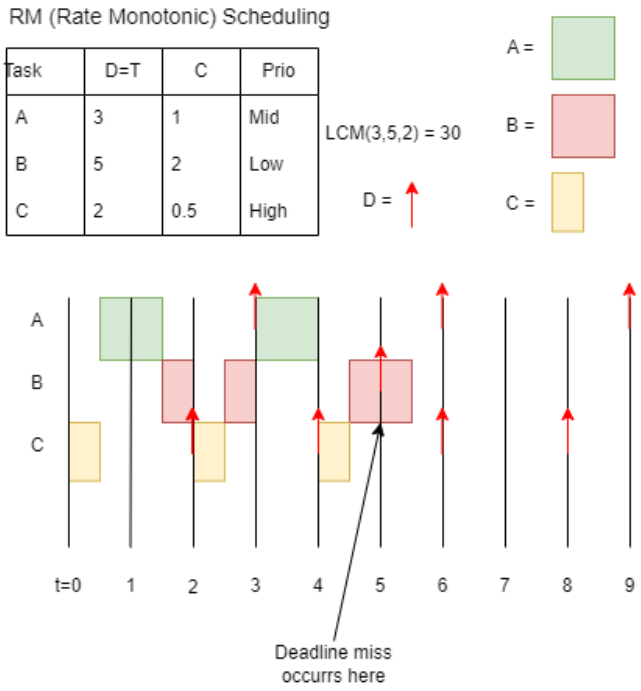

Figure 2. Tracing of the task set proves that it is not schedulable with RM.

1) Exact schedulability test, Tracing: As demonstrated in figure above, the task set is not schedulable with RM scheduling. The task set is not schedulable because task B misses its deadline at $t = 5$.

III. Question 3

To calculate the maximum execution time for Task A to make the task set schedulable, the first thing to try is to make the statement $U <= n(2^{\frac{1}{n}} - 1)$ true. In this case $n = 3$ which gives us $U <= 0.78$. So continuing from there we can find the maximum Ca using the equation $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$. The following table contains the given values for period and execution time for the tasks.

Task	T=D	C
A	4	Ca
B	12	4
C	20	9

Figure 3. Task set

Using the values in the table and the formula $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$ we can extract Ca and get the following equation: $Ca = 4 * (0.78 - \frac{4}{12} - \frac{9}{20}) = -0.014ms$. This is not a valid value for Ca, so we need to try another approach.

We will now try to find maximum Ca by calculating the total amount of execution time needed for all tasks in one hyperperiod. The hyperperiod, or the lowest common multiple (LCM) of the periods of the tasks, is $60ms$. We need to find how many instances occur of each task and then multiply it by the execution time of each task.
Instances of each task in one hyperperiod:

- Instances of task A: $\frac{LCM}{Ta} = \frac{60}{4} = 15$
- Instances of task B: $\frac{LCM}{Tb} = \frac{60}{12} = 5$
- Instances of task C: $\frac{LCM}{Tc} = \frac{60}{20} = 3$

The total amount of execution time needed for all tasks in one hyperperiod is then:

- Task A: $15 * Ca = ?$
- Task B: $4 * 5 = 20ms$
- Task C: $9 * 3 = 30ms$

We now have the following equation: $15 * Ca + 20 + 30 <= 60$. Solving for Ca gives us $Ca <= \frac{60-30-20}{15} = 0.67ms$. This gives us a maximum execution time for task A of $0.67ms$ to make the task set schedulable.

IV. Question 4

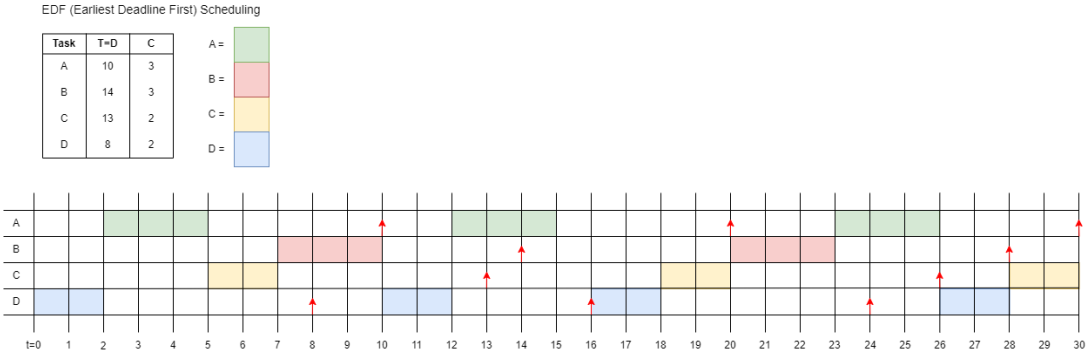

Figure 4. Tracing of the task set with EDF scheduling in the time period $0 - 30ms$.

V. Question 5

VI. Question 6

VII. Question 7

VIII. Question 8