Embedded Systems II, Assignment 1
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. Synchronization refers to the coordination of tasks or components. In offline scheduling, communication and synchronization can be done in a predictable way since the schedule is known before run-time.

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*

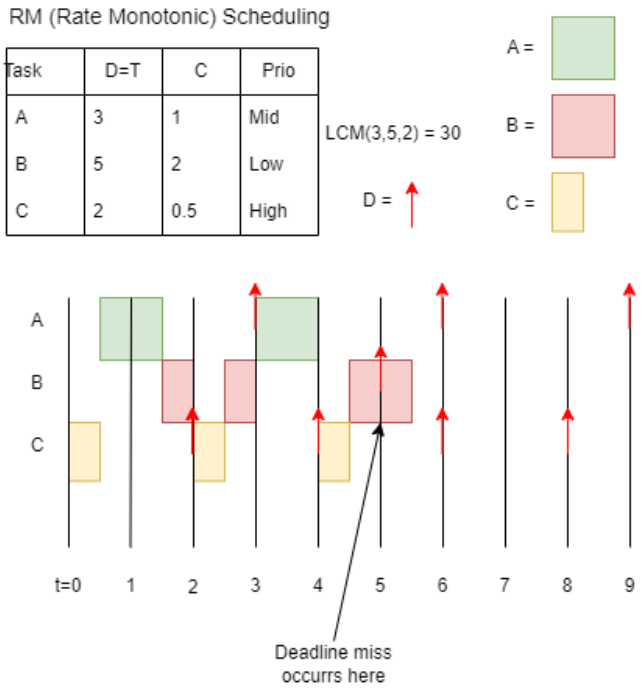

Figure 2. Tracing of the task set proves that it is not schedulable with RM.

*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 C_A 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	C_A
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 C_A and get the following equation: $C_A = 4 * (0.78 - \frac{4}{12} - \frac{9}{20}) = -0.014ms$. This is not a valid value for C_A, so we need to try another approach.

We will now try a different approach by finding the worst case execution time (WCET) for task C which is the lowest priority task according to RM scheduling protocols. WCET for task C cannot be slower than $20ms$ because then it will miss its deadline. WCET is found by releasing all higher priority tasks at the same time as task C, but since we do not know the execution time of task A we will only release task B and C and then find the remaining execution time within the $20ms$ period. Within the $20ms$ time period task B will be released twice and execute a total of $2 * 4ms = 8ms$, task C will be released once and execute a total of $9ms$. If we add up the execution time of task B and C we get $8ms + 9ms = 17ms$. This means that the remaining execution time for task A is a total of $20ms - 17ms = 3ms$. To find the maximum execution time for task A we will have to divide the remaining free time in the $20ms$ period with the number of times task A is released. Task A is released every $4ms$ which means that it will be released 5 times within the $20ms$ period. The maximum execution time for task A is then $\frac{3ms}{5} = 0.6ms$. So the maximum execution time for task A is $0.6ms$ to make the task set schedulable.

IV. Question 4

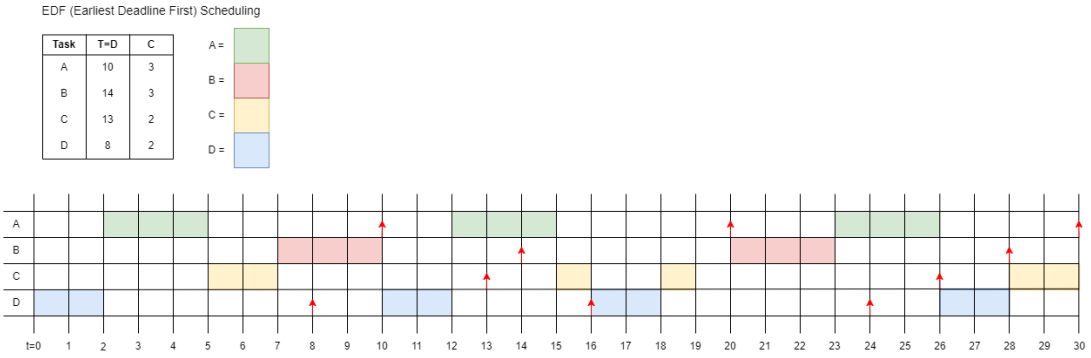

Figure 4. Tracing of the task set with EDF scheduling in the time period $0 - 30ms$.

V. Question 5

To make a task set that is schedulable by EDF but not RM i will find a task set that gives a utilization factor $U = 1$. This will be schedulable with EDF but it is not guaranteed that it is schedulable with RM. To make a task set with $U = 1$ i will use the following equation: $U = \sum_{i=1}^{n} \frac{C_i}{T_i} = 1$. I will use the following task set:

Task	T=D	C
A	3	1
B	6	2
C	9	3

Figure 5. Task set

The task set in figure 5 has a utilization factor of $U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{1}{3} + \frac{2}{6} + \frac{3}{9} = 1$. This task set is schedulable with EDF but most likely not with RM. To prove this I will draw the trace for each scheduling algorithm.

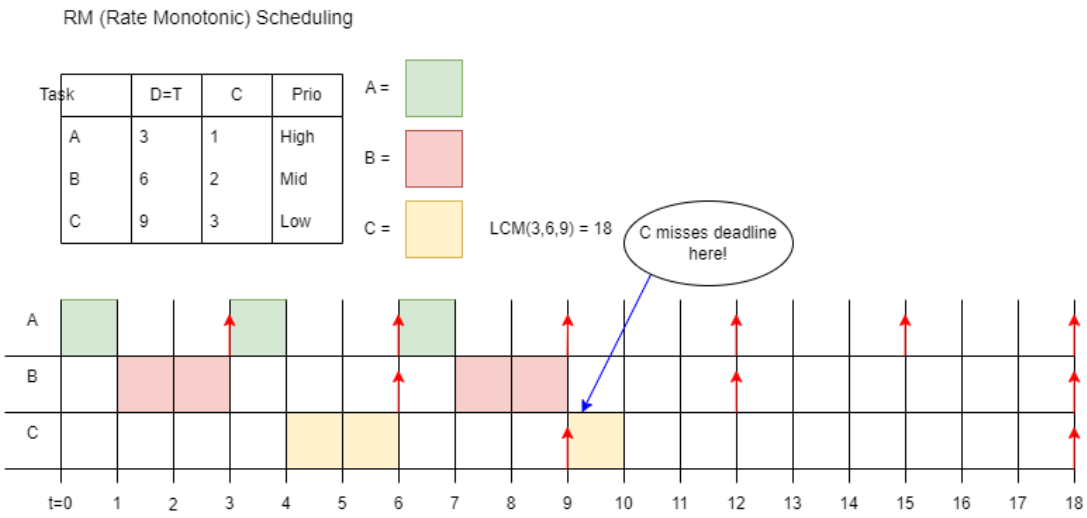

Figure 6. Tracing of the task set with RM scheduling does not work as seen in the figure. The red arrows in the figure indicate the deadlines/period for the tasks.

As seen in figure 6 the task set is not schedulable with RM scheduling. Task C misses its deadline at $t = 9$.

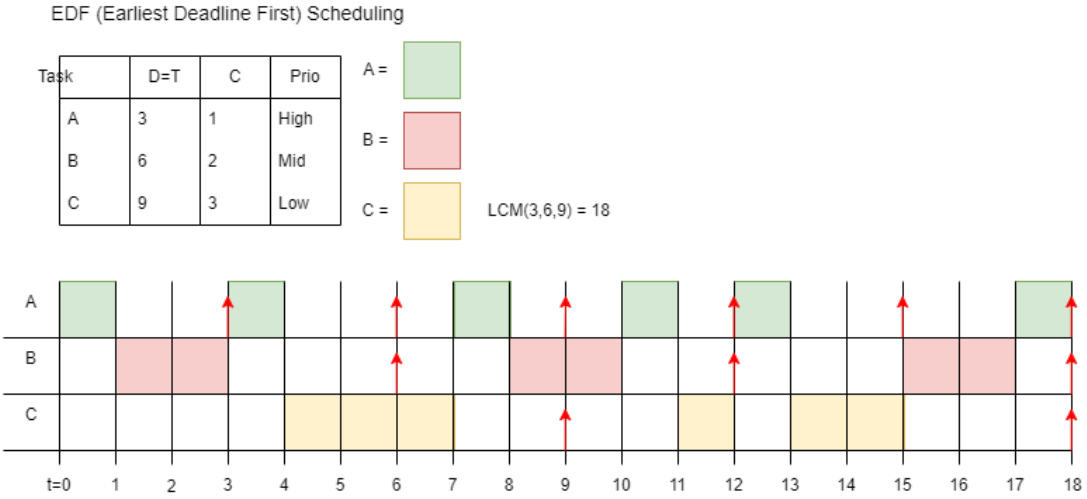

Figure 7. Tracing of the task set with EDF scheduling is feasable as seen in the figure. The red arrows in the figure indicate the deadlines/period for the tasks.

As seen in figure 7 the task set is schedulable with EDF scheduling. The task set is schedulable with EDF because the task with the earliest deadline is always executed first which guarantees schedulability as long as $U <= 1$.

VI. Question 6

To find the minimum inter-arrival time of the high priority task H, we need to consider the worst-case scenario. This means that the high priority task will occur with the highest frequency possible while the task set is still schedulable. To find the minimum inter-arrival time we need to find the remaining time of the period for the lower task after it has executed and fill that time with the execution of the high priority task and find the amount of instances that can fit in that time period. The tasks have the following properties:

- $C_L = 10ms$
- $T_L = 50ms$
- $C_H = 3ms$

The remaining time in the time period after low priority task L has executed is $50ms - 10ms = 40ms$. The high priority task H can execute $\frac{40}{3} = 13.33$ times in that time period. This means that the minimum inter-arrival time of the high priority task H is $\frac{50}{13.33} = 3.75ms$.

VII. Question 7

A. *Difference between polling server and deferrable server*

A polling server polls aperiodic tasks at a fixed period, at each poll a specific amount of execution capacity is provided for use if there is any aperiodic task request but it will lose its execution capacity if there are no aperiodic task requests. On the other hand a deferrable server will not lose its capacity if there are no aperiodic task requests. In the case of the polling server an aperiodic task that arrives after a poll has been made will have to wait until the next poll to be executed. In the case of the deferrable server an aperiodic task that arrives after a poll has been made will be executed immediately, given that it is the highest priority task ready at that instance, since the execution capacity has been retained.

B. *Does Deferrable Server used together with Rate monotonic increase or decrease the Rate Monotonic schedulability bound?*

If the U_S (server utility factor) is less than 0.4 the RM bound is decreased. If the U_S is greater than 0.4 the RM bound is increased.

C. *Main difference between Total Bandwidth Server and Constant Bandwidth Server?*

A Total Bandwidth Server (TBS) is a dynamic server which is used to schedule aperiodic task requests. It assigns each aperiodic request a specific deadline such that $U_P + U_S <= 1$ where U_S is the processor utilization factor of the aperiodic task set. Since the TBS is used with EDF scheduling the each deadline will be met as long as the statement is true and each task is executed as expected. The deadline is assigned using the formula: $d_k = a_k + \frac{C_k}{U_s}$ or $d_k = max(a_k, d_{k-1}) + \frac{C_k}{U_s}$ if there was a previous request. A Constant Bandwidth Server (CBS) is also a dynamic server which uses a budget based system. It assigns each incoming task request a deadline, if the task uses the entire budget capacity it will be replenished again to be able to continue executing or be ready for the next request. When the budget is replenished the deadline is also postponed. The server has a server period of T_s, a server bandwidth of $U_s = \frac{Q_s}{T_s}$ where Q_s is the server budget. This makes it impossible for the server bandwidth to become greater than U_s. This results in the task set always being schedulable with EDF. So the main difference between TBS and CBS is that CBS uses a budget mechanism to make sure the server bandwidth does not exceed a certain limit while the TBS does not.

VIII. Question 8

A. *Specify the Polling Server for this task set. Maximize server utilization, i.e., do the best server you can. Motivate your answer e.g. by a schedulability analysis*

The task set will be scheduled with RM. To make sure the task set is schedulable with the extension of a PS, we have to make sure the following statement is true: $U_P + U_S <= (n+1)(2^{\frac{1}{n+1}} - 1)$ where n is the number of tasks in the task set. The following is the given task set:

Task	T=D	C
A	6	1
B	8	2
C	12	3

Figure 8. Task set

- $U_P = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{1}{6} + \frac{2}{8} + \frac{3}{12} = \frac{2}{3}$
- $\frac{2}{3} + U_S <= (3+1)(2^{\frac{1}{3+1}} - 1) = 0.757$
- $U_S <= 0.757 - \frac{2}{3} = 0.09$

When using the sufficient testing method above, the maximum U_S is 0.09 to make the task set schedulable with the extension of a PS. A PS which supports this U_S is:

- $C_s = 9ms$
- $T_s = 100ms$

This is not a good solution since the server utilization is very low. We can try to find a better solution by analyzing the trace of the task set. The following figure shows the tracing of the task set with the PS specified above.

To find a better we have to analyse the trace of the task set. The sak set without the PS is shown in the following figure.

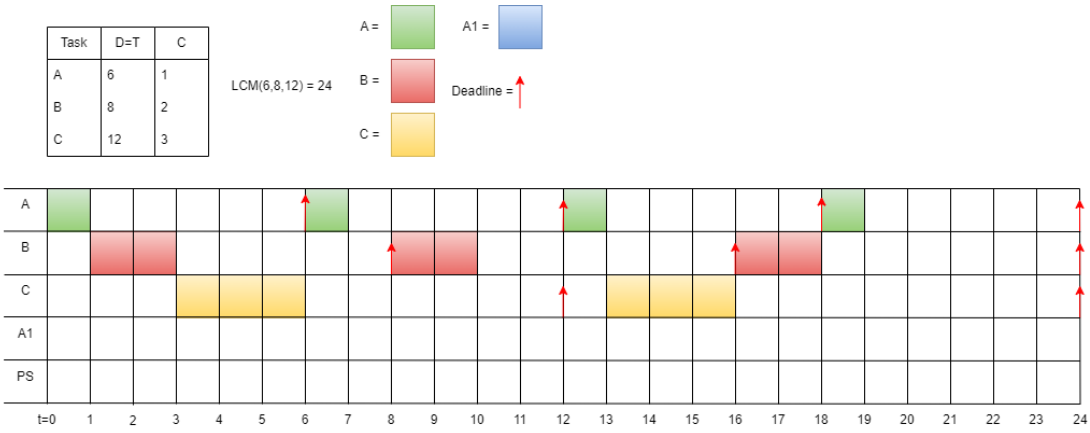

Figure 9. Tracing of the task set without PS.

In figure 9 we can see that we have a total of 8 time units of no execution. What we can do first is to try to find a PS which can fill all of the empty time units. In that case we get a PS with $T_S = \frac{24}{8} = 3ms$ and $C_S = 1ms$. This gives a server utilization of $U_S = \frac{1}{3}$. Using this server we would fill all empty time units. However, we can see that there is only thee empty time units in the first half of the LCM trace when we would need 4. This means we cannot use a server with $C_S = 1ms$ and $T_S = 3ms$ since other tasks would miss their deadlines. We can the try to reduce the server utilization to $U_S = \frac{1}{4}$. This gives a server with $C_S = 1ms$ and $T_S = 4ms$. This server would only fill 6 of the empty time units, but it might make the task set schedulable. The following figure shows the tracing of the task set with the PS specified above.

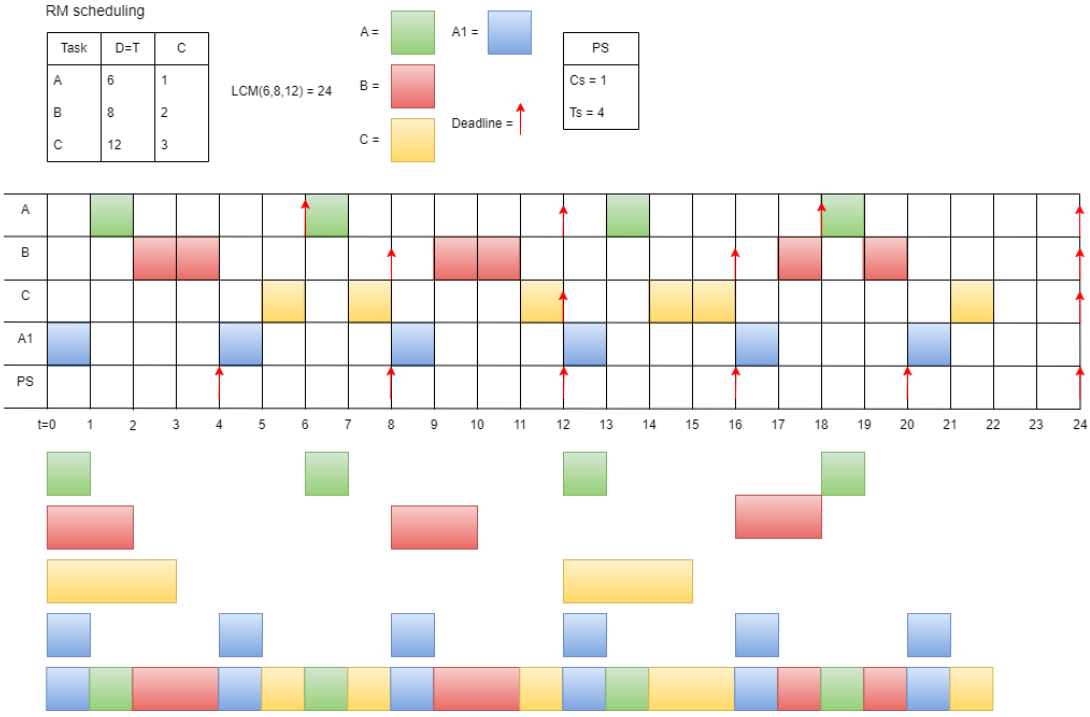

Figure 10. Tracing of the task set along PS with $C_S = 1ms$ and $T_S = 4ms$.

In figure 10 we can see that the task set is schedulable with the PS specified above. This means that the maximum server utilization is $U_S = \frac{1}{4}$ to make the task set schedulable with the extension of a PS. This is a better solution than the one found using the sufficient testing method since the test is exact and we get a better result.

B. *The same question as above, but for Total Bandwidth Server*

This time the task set will be scheduled using EDF. To make sure the task set is schedulable with the extension of a TBS, we have to make sure the following statement is true: $U_P + U_S <= 1$ where U_S is the processor utilization factor of the TBS. $U_P = \frac{2}{3}$ as before and to make the statement true we have to make sure that $U_S <= 1 - \frac{2}{3} = \frac{1}{3}$. So the maximum U_S is $\frac{1}{3}$ to make the task set schedulable with the extension of a TBS.

C. *Soft aperiodic tasks*

Assuming the aperiodic tasks enters the TBS system from question b. Aperiodic task A1 enters at $t = 4$, $a_{A1} = 4$. This will give A1 a deadline of $4 + \frac{C_{A1}}{U_S} = 4 + \frac{2}{\frac{1}{3}} = 4 + 6 = 10ms$. Task A2 enters at $t = 6$, a_{A2}. This will give A2 a deadline of $d_{A2} = max(a_{A2}, d_{A2}) + \frac{C_{A2}}{U_S} = 10 + \frac{C_{A2}}{U_S} = 10 + \frac{2}{\frac{1}{3}} = 10 + 6 = 16ms$. The task set and the aperiodic tasks will be scheduled using EDF. If two tasks happen to have the same deadline, the one that arrived earlier will be executed first. The following figure shows the tracing of the task set with the soft aperiodic tasks A1 and A2.

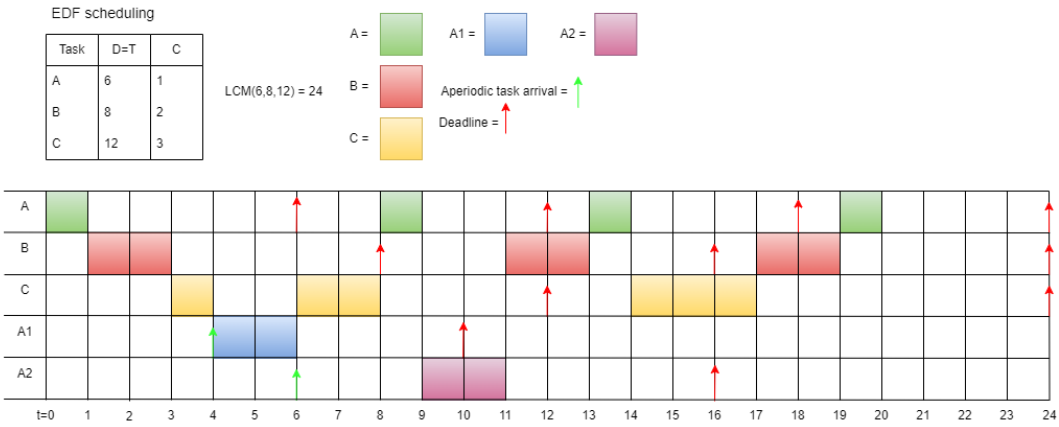

Figure 11. Tracing of the task set with soft aperiodic tasks A1 and A2.