

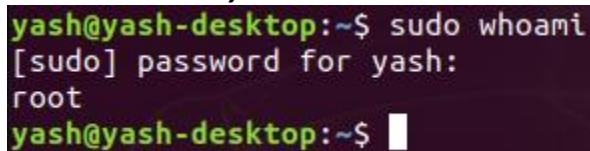
ECEN-5623
ASSIGNMENT-2

YASH GUPTE

Date: 21st June 2019

Q1)

[5 points] If you're using embedded Linux, make yourself an account on your R-Pi3b, Jetson Nano, or Jetson TK1 system. To do this on Jetson TK1, use the reset button if the system is locked, use the well-known "ubuntu" password to login, and then use "sudo adduser", enter the well-known password, and enter user information as you see fit. Add your new user account as a "sudoer" using "visudo" right below root with the same privileges (if you need help with "vi", here's a quick reference or reference card– use arrows to position cursor, below root hit Esc, "i" for insert, type username and privileges as above, and when done, Esc, ":", "wq"). The old unix vi editor was one of the first full-screen visual editors – it still has the advantage of being found on virtually any Unix system in existence, but is otherwise cryptic – along with Emacs it is still widely used in IT, by developers and systems engineers, so it's good to know the basics. If you really don't like vi or Emacs, your next best bet is "nano" for Unix systems. Do a quick "sudo whoami" to demonstrate success. Logout of ubuntu and test your login, then logout. Use Alt+Print-Screen to capture your desktop and save as proof you set up your account. Note that you can always get a terminal with Ctrl+Alt+t key combination. If you don't like the desktop, you can try "GNOME Flashback" and please play around with customizing your account as you wish. Make sure you can access our class web page on Firefox (or default browser) and set your home page to http://ecee.colorado.edu/~ecen5623/index_summer.html . Overall, make sure you are comfortable with development, debug, compiler general native or cross-development tools and document and demonstrate that you know them.



```
yash@yash-desktop:~$ sudo whoami
[sudo] password for yash:
root
yash@yash-desktop:~$
```

Q2) [10 points] Read the paper "Architecture of the Space Shuttle Primary Avionics Software System" [available on Canvas], by Gene Carlow and provide an explanation and critique of the frequency executive architecture. What advantages and disadvantages does the frequency executive have compared to the real-time threading and tasking implementation methods for real-time software systems? Please be specific about the advantages and disadvantages and provide at least 3 advantages as well as 3 disadvantages.

Summary:

The Primary Avionics Sub-system(PASS) is a frequency/cyclic executive architecture implemented as a **hard-real time system** which is **asynchronous** in nature.

The cyclic executive is designed such that all the tasks/processes in the system are scheduled based on priority to occur at fixed intervals during an entire cycle. It is ensured that **each of these tasks is serviced during 1 complete cycle**. The **GN&C system** can have up-to **200 of such tasks**.

Each of these tasks are categorized into high frequency – All tasks related to the Vehicle Flight Control which include space shuttle navigation, landing etc and mid-low frequency – CRT displays and initiation of principle processes.

The high frequency tasks occur at 25Hz i.e. 40ms intervals whereas the mid-low frequency tasks occur at 6.25Hz – 0.25Hz.

In order to achieve a successful implementation of the cyclic executive multiple sub-systems play a major role in the PASS architecture. These include:

1. **The Flight Control Operating System (FCOS)** is responsible for **allocation of all internal resources** for tasks such as I/O resources and **balancing the system load**.
2. **System Control Unit** of the PASS is responsible for **establishing the time intervals** as well as to **service these tasks**.

3. **User Interface** has functions which include **input command processing** which includes commands from keyboard, **output message processing** and displaying it on CRT.
4. **Guidance and Navigation Control (GN&C)** performs about 200 tasks with extreme efficiency and hard deadline bounds.
5. **Vehicle Checkout(VCO)** has functions like **initializing the avionics system**.

These sub-systems together constitute the PASS architecture.

Advantages:

1. The cyclic executive has a main loop which has the scheduler. This scheduler is responsible for task assignment at fixed intervals. However, there is a **provision for interrupts as well**. These interrupts which occur asynchronously have a **small execution time** and pass the **major processing to the main loop**. This could include a simple flag being set or a bit manipulation. The major processing is done in the main loop. This helps **reduce latency between the occurrence and actual execution of the request**.
2. The cyclic executive **does not require mechanisms which prevent deadlocks** or resource contention. These includes semaphores, mutexes etc. This is because the system is designed such that no 2 tasks occur at the same time. As a result, the system remains predictable hence alleviating the need for such mechanisms.
3. The processing management function ensures that the **highest priority system** or application process, **ready with work to perform, is given the CPU resource** required to accomplish it. In non-cyclic executive systems events where a task is scheduled to run after a fixed interval, there could be times when due to lack of resources the event would run but not complete its processing. Whereas, in a system where a task with work to perform is given preference, there would not be an unnecessary cycle in the cyclic execution method.

Disadvantages:

1. Once established it is **difficult to modify the cyclic schedule**. While this may seem insignificant initially, there could be a future need to add or remove certain aspects from the system. For example, an added functionality like pressure measurement inside the vehicle running at mid frequency may require re-assignment of the entire cyclic schedule.
2. The PASS architecture has **provisions for interrupts in case of malfunctions or other critical events not accounted for by the cyclic schedule**. While these malfunctions are not to occur frequently there could be instances when they could occur more frequently than expected and affect the actual cyclic schedule. For example, like in the Apollo-11 mission the navigation switches were faulty causing a repeated alarm, in PASS a break in the navigation system or change in internal pressure could induce repeated interrupts affecting the system functionality severely.
3. **Release time of the tasks must be fixed**. In order to ensure a successful implementation of frequency executive design, each task must be ensured to have a fixed amount of time which includes resource acquiring and releasing. This could be an issue if a task depends on data provided by another task. In this case, even if a task begins at the specified time it could happen that it is delayed because it does not the data it needs to begin processing.

Q3) [50 points] Download feasibility example code and build it on a Jetson or alternate system of your choice (or ECES Linux if you have not mastered the Jetson yet) and execute the code. Compare the tests provided to analysis using Cheddar for the first 4 examples tested by the code. Now, implement remaining examples [5 more] of your interest from those that we reviewed in class (found here). Complete analysis using Cheddar RM. In cases where RM fails, test EDF or LLF to see if it succeeds and if it does, explain why. Cheddar uses both service simulations over the LCM of the periods as well as feasibility analysis based on the RM LUB and scheduling-point/completion-test algorithms, referred to as “Worst Case Analysis”. Does your modified Feasibility code agree with Cheddar analysis in all 5 additional cases? Why or why not?

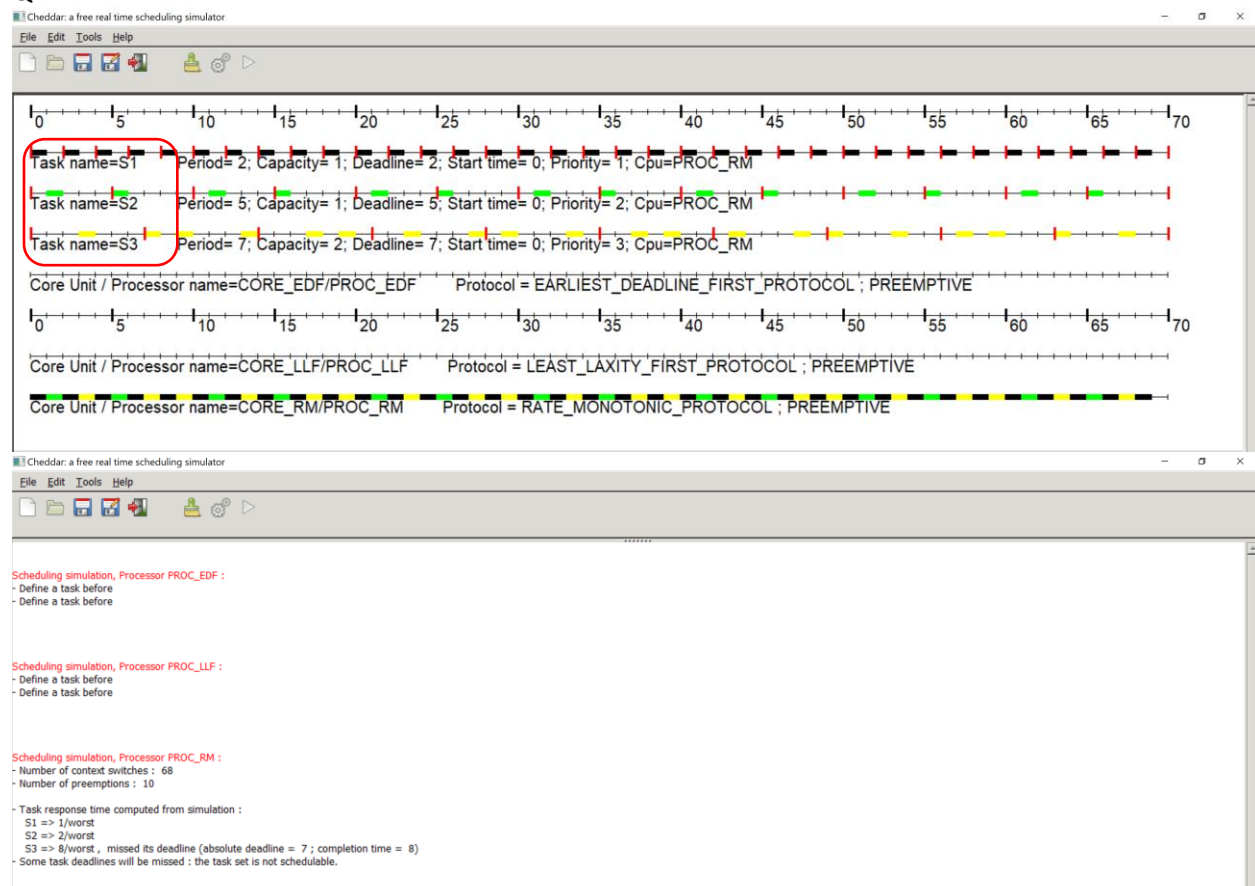
Example code 0 : $C_1 = 1, C_2=1, C_3=2; T_1=2, T_2=10, T_3=15$

Feasibility code analysis: The feasibility code output matches the Cheddar analysis for the Rate Monotonic scheduling used in feasibility code.

Example code 1 : $C_1 = 1, C_2=1, C_3=2; T_1=2, T_2=5, T_3=7$

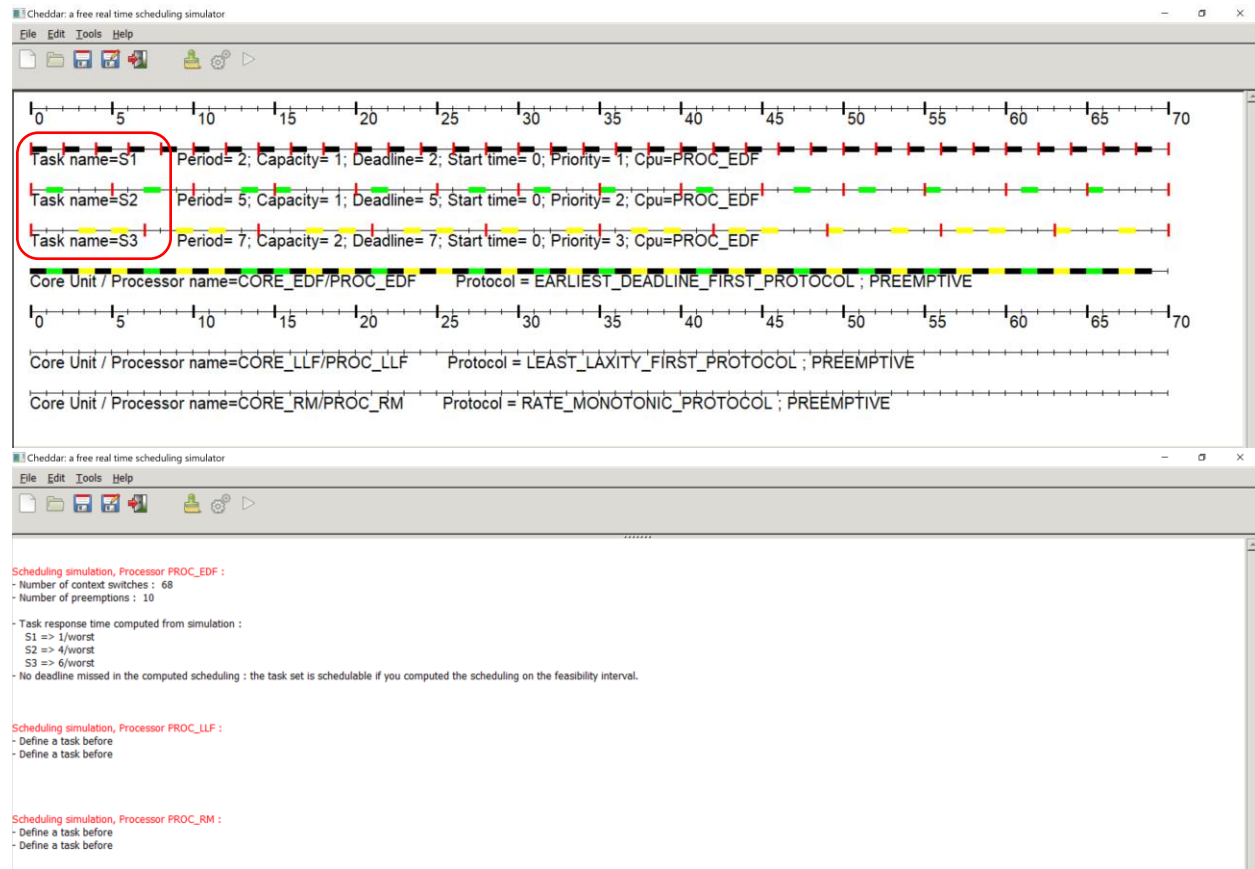
Feasibility code analysis: The feasibility code output matches the Cheddar analysis for the Rate Monotonic scheduling used in feasibility code.

Q2-RM



Ans: The third task misses its deadline. In the time interval marked above, due to higher priority of S1 and S2, S3 is serviced only once before its deadline at instant 3-4. Since, S1 is processed every 2s and S2 at every 5s, processing of S3 is starved till 3s and by 7s it misses the deadline.

Q2-EDF

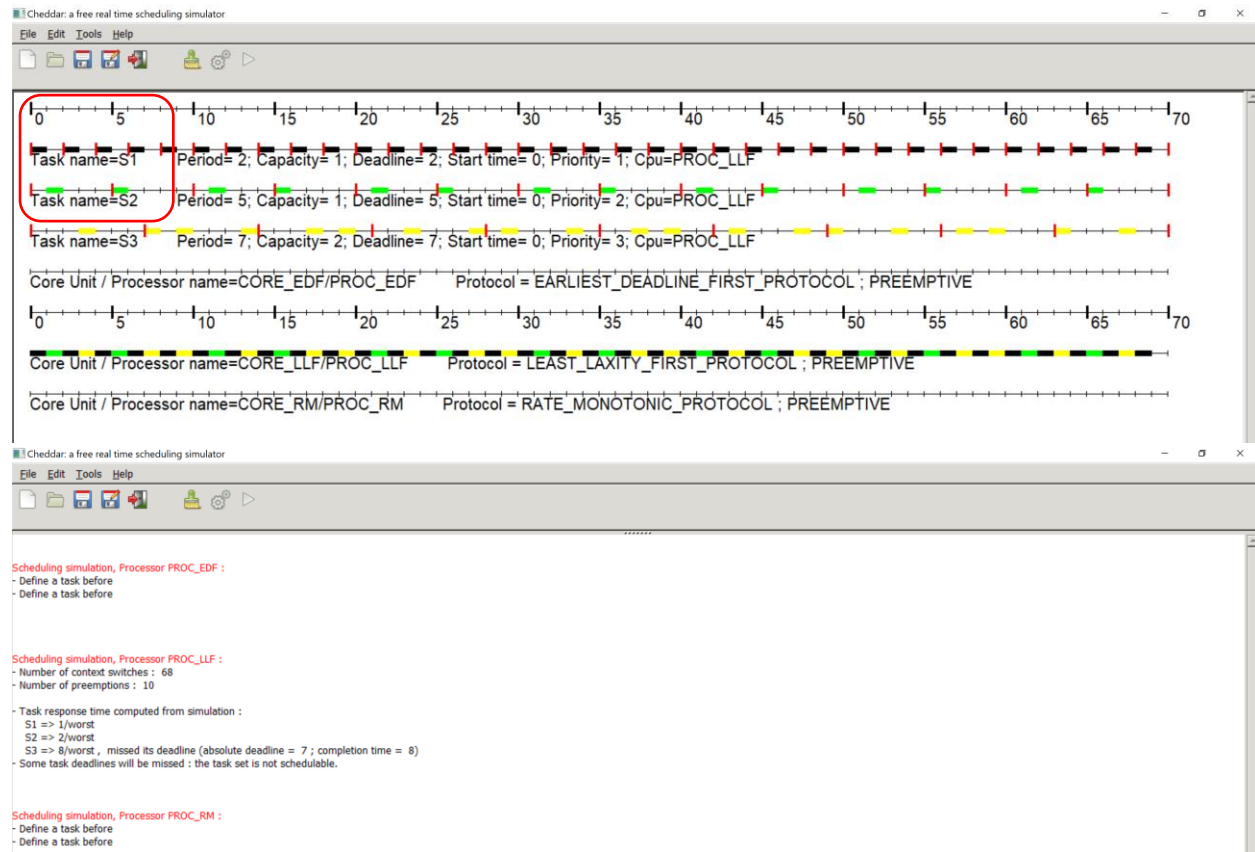


Ans: EDF was successful. Considering instant from 0 – 7s.

1. S1 has priority because it has the smallest deadline of 2s. Hence it is serviced first.
2. (1-2)s Now S2 has the second highest priority with deadline of 5s. Hence is serviced.
3. (2-3) Now, the deadline for S2 approaches again hence, it gets the highest priority and is serviced.
4. (3-4) S1 and S2 have been serviced so S3 is scheduled in this time instant.
5. (4-5) S1 is serviced again due to higher priority.
6. (5-6) S2 has already been serviced and S1 as well. Now S3 is serviced.

Similar behavior is carried out in the remaining time. Hence, all tasks are serviced before their deadlines.

Q2-LLF



Ans:

According to LLF, the task with the smallest difference between Deadline and Computation time is executed first. For given problem, For S1 => $D_1 - C_1 = 1$; S2 => $D_2 - C_2 = 4$; S3 => $D_3 - C_3 = 5$.

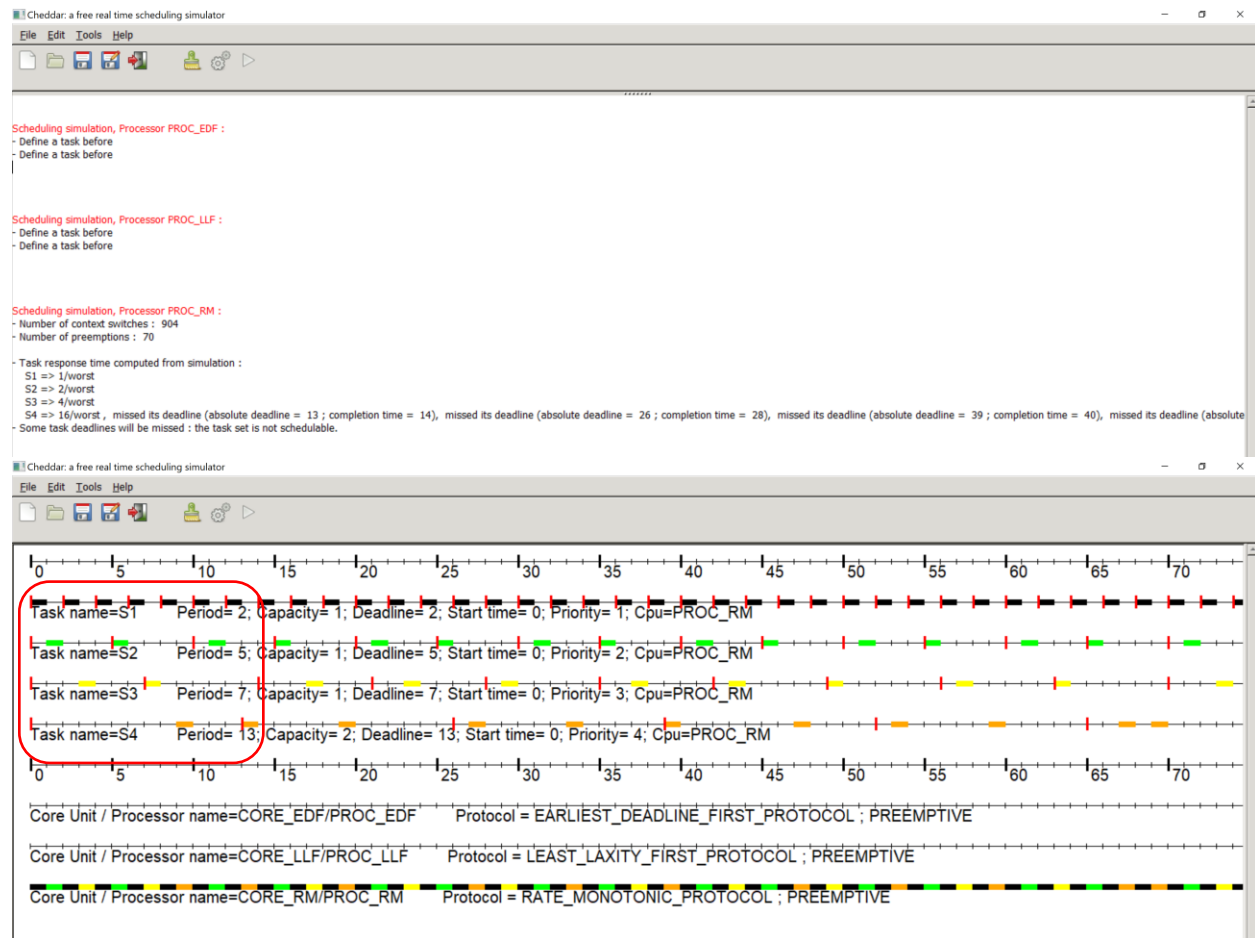
Which gives S1 highest priority at start. Considering instant (0 -7)s

- (1-2) Now that S1 is already serviced, S2 has second highest priority and is executed.
- (2-3) Now S1 has the highest priority again it has completed before its deadline.
- (3-4) Now S3 has highest priority since other tasks have finished completion before deadline.
- (4-5) S1 is processed again.
- (5-6) Now considering S2; $D_2 - C_2 \Rightarrow (5-5) - 1 = -1$. For S3; $D_3 - C_3 \Rightarrow (7-5) - 2 = 0$. Hence, S2 gets higher priority.
- (6-7) S1 is processed and S3 misses the deadline.
- This pattern repeats till 7s. That's when S3 misses the deadline. This is because, in LLF consideration is given to difference between deadlines and computation time. Since, S1 has the smallest value, it was serviced at highest priority followed by S2 then S3. The issue arises in LLF when the computation.

Example code 2 : $C1 = 1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13$

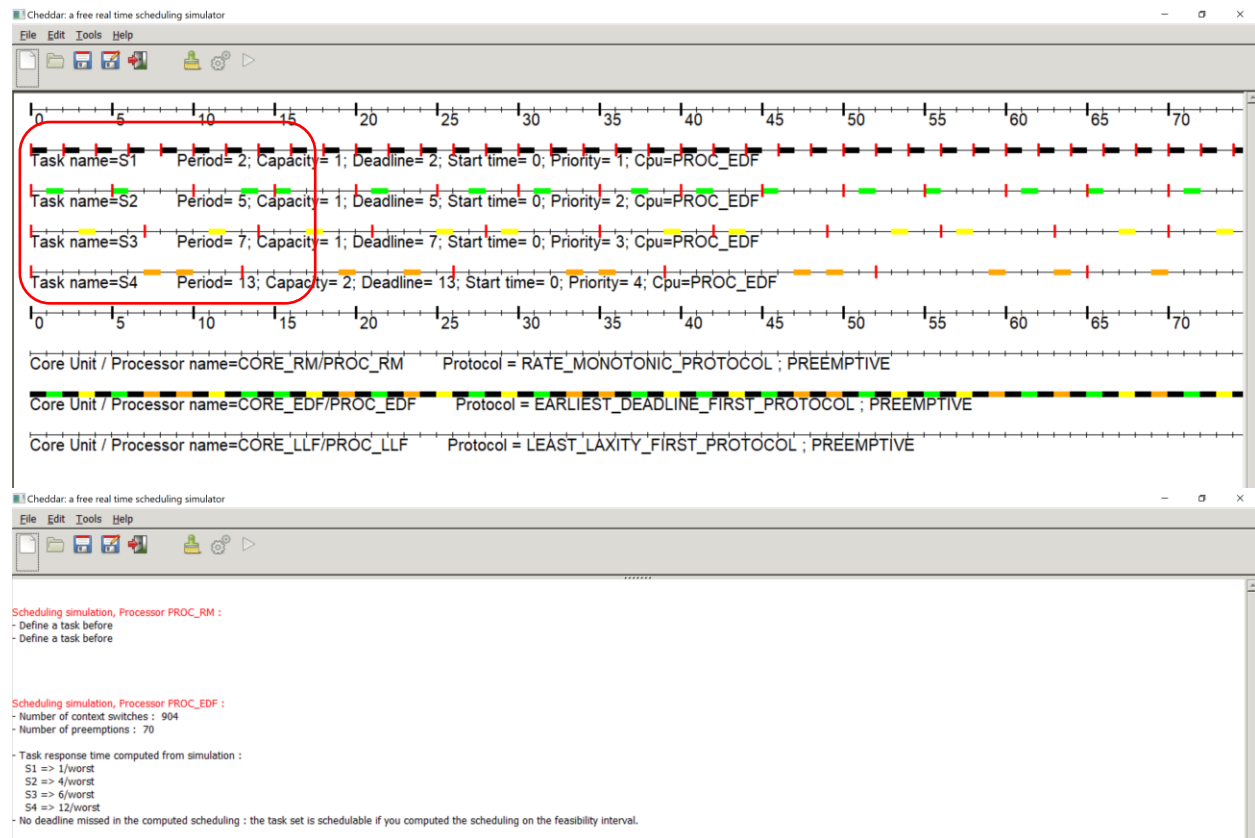
Feasibility code analysis: The feasibility code output matches the Cheddar analysis for the Rate Monotonic scheduling used in feasibility code.

Q3-RM



Ans: Till tasks S3 execution is normal. However due to priority given to fastest execution time first, S4 is serviced only at instant (9-10) before deadline. The next time it executes, it has already missed a deadline this trend carries over till 910s.

Q3-EDF

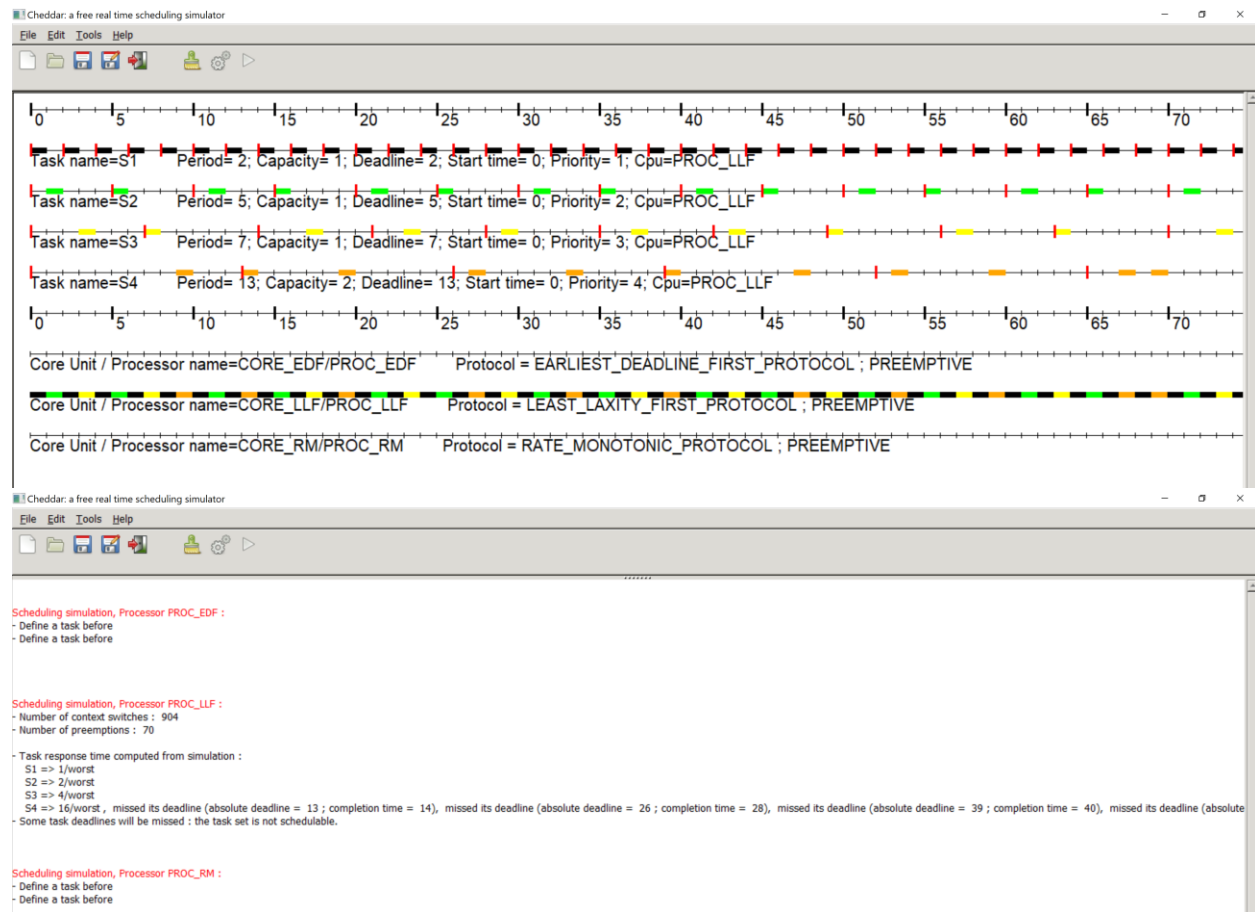


Ans: EDF was successful. Considering instant from 0 – 13s.

1. S1 has priority because it has the smallest deadline of 2s. Hence it is serviced first.
2. (1-2)s Now S2 has the second highest priority with deadline of 5s. Hence is serviced.
3. (2-3)Now, the deadline for S2 approaches again hence, it gets the highest priority and is serviced.
4. (3-4) S1 and S2 have been serviced so S3 is scheduled in this time instant.
5. (4-5) S1 is serviced again due to higher priority.
6. (5-6) S2 is now the highest priority since S1,S2 have been serviced and S3 and S4 remain.
7. (6-7) S1 is serviced due to earliest deadline.
8. (7-8) S4 is serviced because all other tasks have completed before their deadline at this instant.
9. (8-9) S1 executes.
10. (9-10) Earliest deadline is for S4 at this point and hence is executed.

Similar behavior is carried out in the remaining time. Hence, all tasks are serviced before their deadlines.

Q3-LLF



Ans: LLF is applied successfully despite the erroneous report generated by Cheddar. This is because at 11s, by LLF criteria which suggests Di-Ci, we can see the following:

For S2; $(15-10)-1 = 4$.

For S4; $(13-10)-2 = 1$. Which suggests S4 should be executed first.

Note: Additional Screenshots in attached PDF in this zip-file.

Example code 3 : $C1 = 1, C2=2, C3=3; T1=5, T2=5, T3=15$

Feasibility code analysis: The feasibility code output matches the Cheddar analysis for the Rate Monotonic scheduling used in feasibility code.

Example code 4 : $C1 = 1, C2=1, C3=4; T1=2, T2=4, T3=16$

Feasibility code analysis: The feasibility code output matches the Cheddar analysis for the Rate Monotonic scheduling used in feasibility code.

All the additional 5 codes also match the Cheddar analysis. This is because, the algorithm used in the code for Rate Monotonic implementation is the same algorithm used in the feasibility example code.

Feasibility Code Screenshots:

```
yash@yash-desktop:~/RTES$ make
gcc -O0 -g -c feasibility_tests.c
gcc -O0 -g -o feasibility_tests feasibility_tests.o -lm
yash@yash-desktop:~/RTES$ ./feasibility_tests
***** Completion Test Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE

***** Scheduling Point Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
yash@yash-desktop:~/RTES$
```

Fig. Original Code output

```
yash@yash-desktop:~/RTES$ make
gcc -O0 -g -c feasibility_tests_modified.c
gcc -O0 -g -o feasibility_tests_modified feasibility_tests_modified.o -lm
yash@yash-desktop:~/RTES$ ./feasibility_tests_modified
***** Completion Test Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.20 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
Ex-8 U=0.99 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=14; T=D): FEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE

***** Scheduling Point Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.20 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
Ex-8 U=0.99 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=14; T=D): FEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE
yash@yash-desktop:~/RTES$
```

Fig. Modified Code output

Feasibility code explanation:

1. The code takes 1 pair of time-periods for 3 tasks and corresponding execution times for the same tasks.
2. There are 2 main algorithms implemented for testing the feasibility which are “completion time feasibility” and “scheduling point feasibility”. They take the previously mentioned time-periods and execution times as the arguments.
3. The exact algorithm has not been disclosed. The output of the functions provide “FEASIBLE” or “INFEASIBLE” as outputs depending on whether the function returned a 1 or 0 respectively.

Q4) [15 points] Provide 3 constraints that are made on the RM LUB derivation and 3 assumptions as documented in the Liu and Layland paper and in Chapter 3 of RTECS with Linux and RTOS. Finally, list 3 key derivation steps in the RM LUB derivation that you either do not understand or that you would consider “tricky” math. Attempt to describe the rationale for those steps as best you can do based upon reading in Chapter 3 of RTECS with Linux and RTOS.

Assumption: Tasks fully utilize the processor and minimize processor utilization factor.

Constraint : Ration between the tasks is < 2 .

The assumption stems from the claim of RM LUB. The claim is that the utilization factor of any task less than processor utilization factor calculated by RM LUB is guaranteed to be schedulable.

This constraint is only used in theorem 4 which proves the RM LUB theorem for 2 tasks. The constraint implies that for any given task, its period (T_i) must be such that it occurs only once before the deadline / period of the next task. For example, $T_1 = 2$, $T_2 = 3$. $T_2/T_1 = 1.5$ Which is acceptable. However, if $T_1=2$, $T_2=5$; $T_2/T_1 = 2.5$ which is not acceptable for the sake of this derivation.

Assumption: Time period of $T_m > T_{m-1} > T_{m-2} > \dots$

Constraint : $\ln C_1 = T_2 - T_1 + \delta$; $\delta > 0$.

The assumption is based on the rules of Rate Monotonic policy which states that the task with the fastest execution time must have the highest priority. The remaining tasks will have priorities based off the same assumption. This means for 2 tasks S_1 and S_2 , if $T_1 < T_2$; S_1 has higher priority than S_2 . In order to schedule them via RM, the constraint would be $T_2 > T_1$. Which is the same assumption in the RM LUB derivation only extended to multiple tasks.

Applying this rule on RM LUB, we get that $T_m > T_{m-1} > T_{m-2} > \dots T_1$. In the Theorem-4 derivation, δ is assumed to be greater than 0. If the equation is broken down into words it suggests that, time of execution of task-1 (C_1) is greater than the difference between time-periods of execution between S_2 and S_1 by a positive factor. This implies that execution of task C_1 ends at a time much closer to its deadline than if δ were 0. For example, let $T_1 = 3$, $T_2=5$; $C_1 = 5-3+\delta \Rightarrow 2+\delta$. If $\delta>0$, task s_1 ends much closer to its deadline as compared to when $\delta=0$.

Assumption: The CI (Critical Instant) is considered.

Constraint : The priorities of the task are static.

Critical Instant is the instant of time at which all the tasks request to be processed. RM LUB is a sufficient feasibility test. This means that the assumptions and constraints on the task scheduling are such that they provide an outcome considering all the worst-case scenarios. One such scenario is Critical Instant (CI).

This means that the outcome of a RM LUB does not imply that the task cannot be scheduled however, a value which is lower than RM LUB guarantees that it can be scheduled.

This constraint while not directly mentioned in the derivation forms a very important basis of it. Dynamically changing priorities at run-time involves re-scheduling of all tasks. This adds an overhead which could affect the deadlines. In case of RM LUB where only *static priorities* are considered, this not a concern.

Constraint: None of the tasks have same deadline.

This is an important constraint. Theorem-4 and Theorem-5 have the same constraint that no 2 or more tasks have the same time of execution. This is because, when such a scenario occurs, the compiler decides which task to execute first. This could lead to only one service being starved while the other being processed adequately.

Tricky Math:

1. Theorem 4, $U-U'=(\delta/T1)-(\delta/T2)>0$

The derivation directly jumps to this equation after assuming $C'm = C_m$. The steps for calculation of U and U' have been skipped making it an arduous task to understand this step.

2. Theorem 4, $U-U''=-(\delta/T1)+(2*\delta/T1)>0$

This step is based off the previous step mentioned above. The ambiguity is further instantiated when a double derivative has been taken without details of U' making it difficult to understand the math.

3.
$$\partial U / \partial g_j = (g_j^2 + 2g_j - g_{j-1}) / (g_j + 1)^2 - (g_{j+1}) / (g_{j+1} + 1) = 0, \quad j = 1, 2, \dots, m - 1. \quad (5)$$

It feels like the partial derivative steps have been skipped and the equation has been presented directly. Moreover, what g_j is has not been specified clearly.