

Real Time Embedded Systems – ECEN 5623

Summer 2019

Professor Sam Siewert

Report By: Poorn Mehta

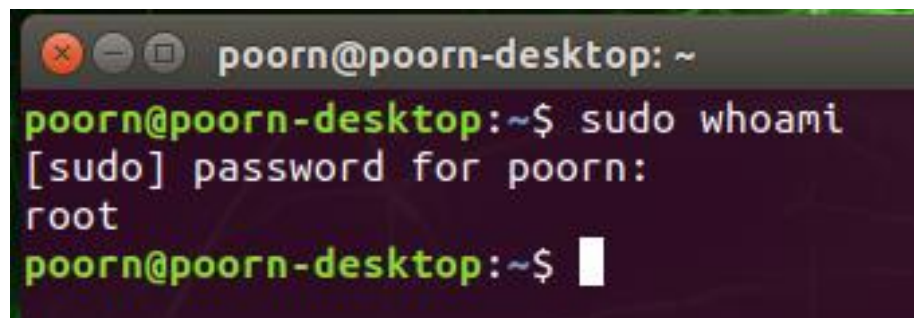
Platform: Jetson Nano

Homework 2 (git: <https://github.com/Poorn-Mehta/RTES/tree/master/HW2>)

Q1: 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.

Solution:

I already set up my account on Jetson Nano as part of the first boot procedure. Below is the screenshot to reflect the same.



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

Q2: 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.

Solution:

1. Summary

- **PASS** is developed to provide a front end, well-structured system architecture, for NASA's Space Shuttle Orbiter – in which, it's a **vital component** since almost all operations are being handled by PASS.
- Owing to the **multitude of constraints and requirements**, the PASS is **segmented into 8 different functional combinations** – each of which is identified as OPS (Operational Sequence).
- Every OPS has further **classifications in terms of various resources** – for example, main memory load consists of 3 parts – each carrying distinct useful information/data.
- This implementation of OPS is **backed by the control segment** – essentially a series of standardized logic blocks, responsible for establishing the structure of an OPS, major mode, and some other key functionalities.
- Control segment defines the **sequencing of all processing required** within an OPS, major mode, etc. by its design.
- **Conventional approach** to flight software development was **synchronous in nature** – which enjoyed the **advantages such as repeatability**, but it **wasn't very flexible**. However, since the **PASS** required to be very flexible by design, a **nonsynchronous approach was adapted** – along with some other aspects such as: **isolation of the application processes** from the external I/O devices, as well as introducing **computer redundancies**.
- It should be noted that the **extremely complex data network** of the PASS is **result of the redundancies built in avionics and data processing**.
- The system control contains a **cyclic executing** – servicing interfaces, while having **frequency of 25Hz**.
- Design structure of **GN&C** (which is one of the three main application software) – is **largely affected by the phasing of the function executions & associated I/O**, as well as **CPU loading constraints**.
- The GN&C is also, a **cyclic closed loop application** – **supporting execution rates from 0.25Hz to 25Hz**.

- Overall, **quoting the paper**: “The structural aspects of the software architecture and HAL/S language together with the standardization of interfaces through the FCOS SVCs and control segment grammar have been major factors in the successful implementation of PASS software. They have made it possible to pursue the implementation of software requirements in a top-down sequence and follow a release plan that used a building-block approach.”

2. Advantages of Presented Method (Cyclic Executive) [\[1\]](#)

- Very simple to design and implement
- No race condition
- No deadlock
- No need of mutex/semaphore locks
- No multithreading
- Task dispatch is efficient
- No scheduling anomalies
- Highly reliable

3. Disadvantages of Presented Method (Cyclic Executive) [\[1\]](#)

- Usually non-preemptive
- Mode switching only possible at the boundaries – might have very high latency
- Highly inflexible
- Task release times have to be fixed
- Task slicing is difficult and prone to errors
- Lots of offline computation might be required to design scheduler

Q3: 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?

Solution:

Note: Please find all screenshots – that are not included here, in the separate zip file (uploaded on canvas, also they’re available on the GitHub). LLF simulation is somewhat broken in the Cheddar 3.1 – so some of them are giving incorrect output.

Why do all solutions match? All of the code output and Cheddar analysis are matching since (I believe) the theorems being utilized are same, and therefore, algorithms are also more or less similar to each other.

Original Feasibility Code Output

```
root@poorn-desktop:~/Workspace/RTES/HW2/Q3# make
gcc -O0 -g -c feasibility_tests.c
gcc -O0 -g -o feasibility_tests feasibility_tests.o -lm
root@poorn-desktop:~/Workspace/RTES/HW2/Q3# ./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
```

Modified Feasibility Code Output

```
root@poorn-desktop:~/Workspace/RTES/HW2/Q3# make
gcc -O0 -g -c feasibility_tests.c
gcc -O0 -g -o feasibility_tests feasibility_tests.o -lm
root@poorn-desktop:~/Workspace/RTES/HW2/Q3# ./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
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
Ex-10 U=0.99 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=14; T=D): FEASIBLE
Ex-11 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-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
Ex-10 U=0.99 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=14; T=D): FEASIBLE
Ex-11 U=1.00 (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE
root@poorn-desktop:~/Workspace/RTES/HW2/Q3#
```

Example 0

S₁: T₁ = 2, C₁ = 1; S₂: T₂ = 10, C₂ = 1; S₃: T₃ = 15, C₃ = 2

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Cheddar RM Analysis:

Scheduling feasibility, Processor cpu_rm :

1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 30 (see [18], page 5).
- 8 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.73333 (see [1], page 6).
- Processor utilization factor with period is 0.73333 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.73333 is equal or less than 0.77976 (see [1], page 16, theorem 8).

2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time : (see [2], page 3, equation 4).

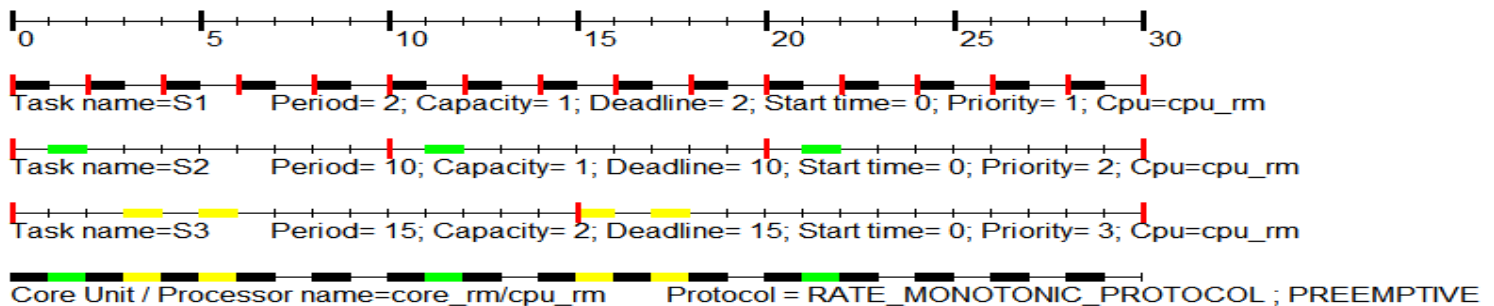
S₃ => 6

S₂ => 2

S₁ => 1

- All task deadlines will be met : the task set is schedulable.

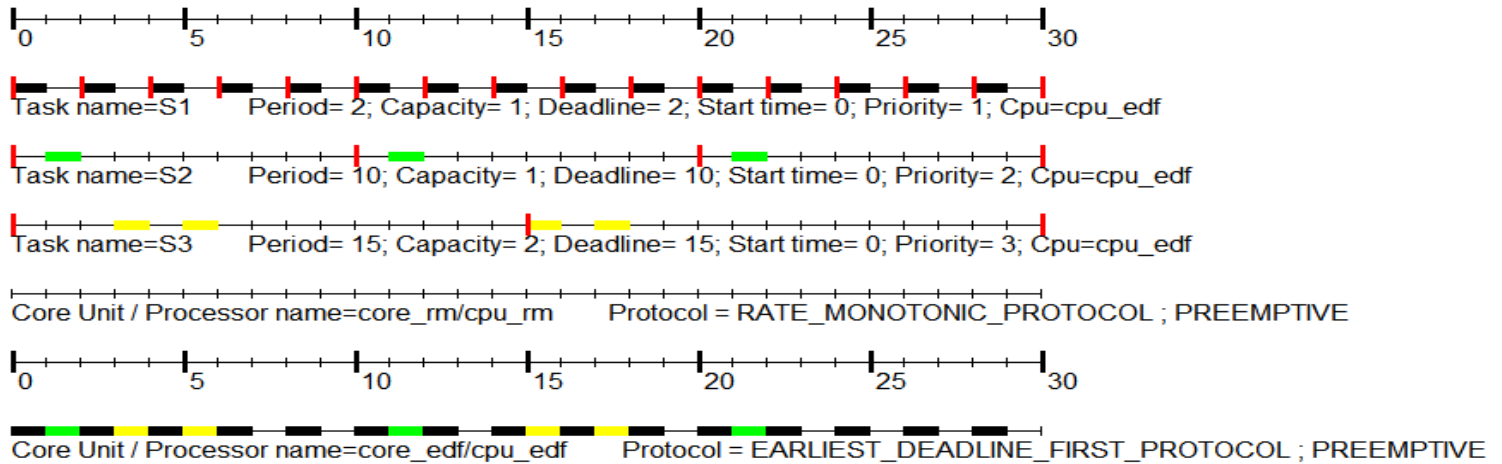
Cheddar RM Simulation (Success):



Scheduling simulation, Processor cpu_rm :

- Number of context switches : 14
- Number of preemptions : 2
- Task response time computed from simulation :
 - S₁ => 1/worst
 - S₂ => 2/worst
 - S₃ => 6/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

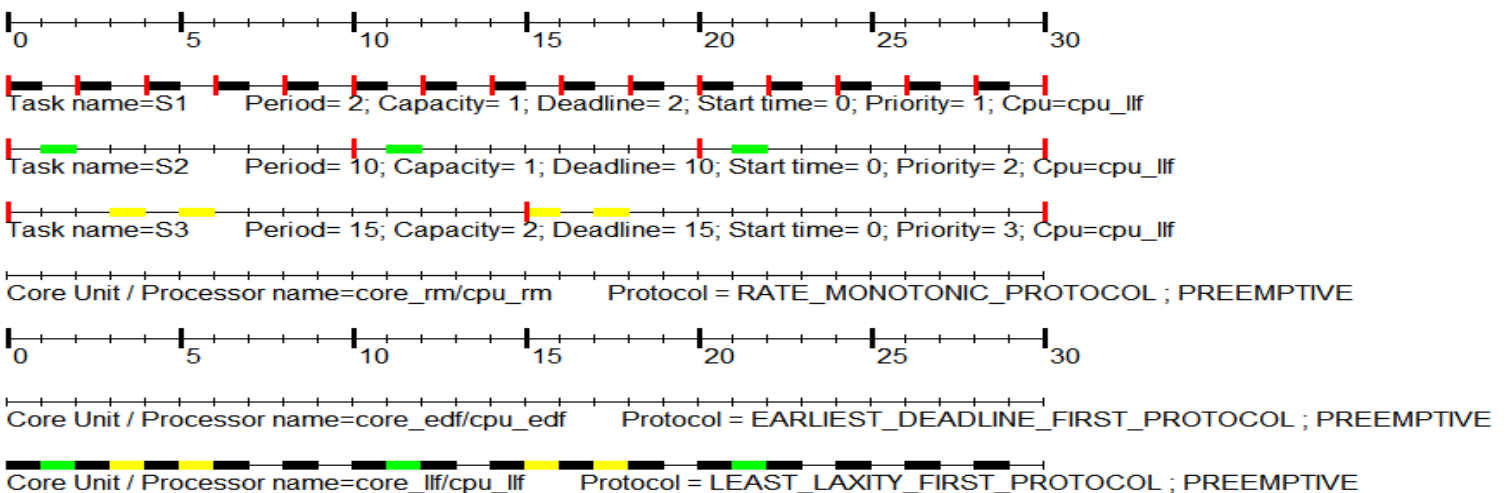
Cheddar EDF Simulation (Success):



Scheduling simulation, Processor cpu_edf :

- Number of context switches : 14
- Number of preemptions : 2
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 6/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Cheddar LLF Simulation (Success):



Scheduling simulation, Processor cpu_llf :

- Number of context switches : 14
- Number of preemptions : 2
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 6/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Example 1

S₁: T₁ = 2, C₁ = 1; S₂: T₂ = 5, C₂ = 1; S₃: T₃ = 7, C₃ = 2

Code Output: **Infeasible**

Cheddar Feasibility Analysis: **Infeasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **infeasible** since all tests have been failed by given set of services. **First missed deadline is at 7th time slot – for S₃**. This has taken place **because of static priority assignment** – the higher priority services are running even when they are not critical with respect to their own deadlines. **This is effectively a priority inversion.**

EDF and **LLF** has to be **feasible** because the scheduling is **fully preemptive** with **dynamic priority** and the **total utilization is not exceeding 100%**. However, it might be impractical since EDF and LLF both requires quite a bit of information about all services, their deadlines, their required processing time, etc.

Cheddar RM Analysis:

Scheduling feasibility, Processor `cpu_rm` :

1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 70 (see [18], page 5).
- 1 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.98571 (see [1], page 6).
- Processor utilization factor with period is 0.98571 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor utilization factor 0.98571 is more than 0.77976 (see [1], page 16, theorem 8).

2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time : (see [2], page 3, equation 4).
 - S₃ => 8, missed its deadline (deadline = 7)
 - S₂ => 2
 - S₁ => 1
- Some task deadlines will be missed : the task set is not schedulable.

Cheddar EDF Analysis:

Scheduling feasibility, Processor `cpu_edf` :

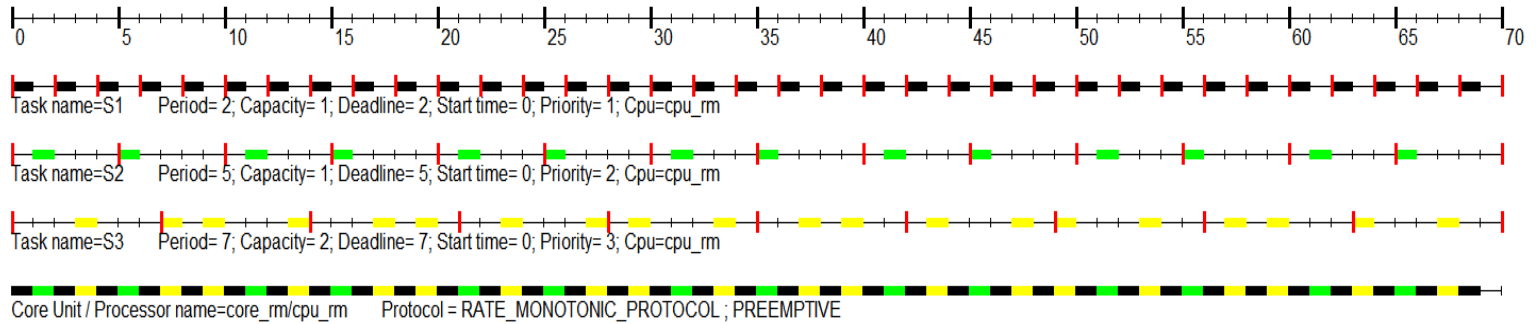
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 70 (see [18], page 5).
- 1 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.98571 (see [1], page 6).
- Processor utilization factor with period is 0.98571 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 0.98571 is equal or less than 1.00000 (see [1], page 8, theorem 2).

2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :
 - S₁ => 1
 - S₂ => 4
 - S₃ => 6
- All task deadlines will be met : the task set is schedulable.

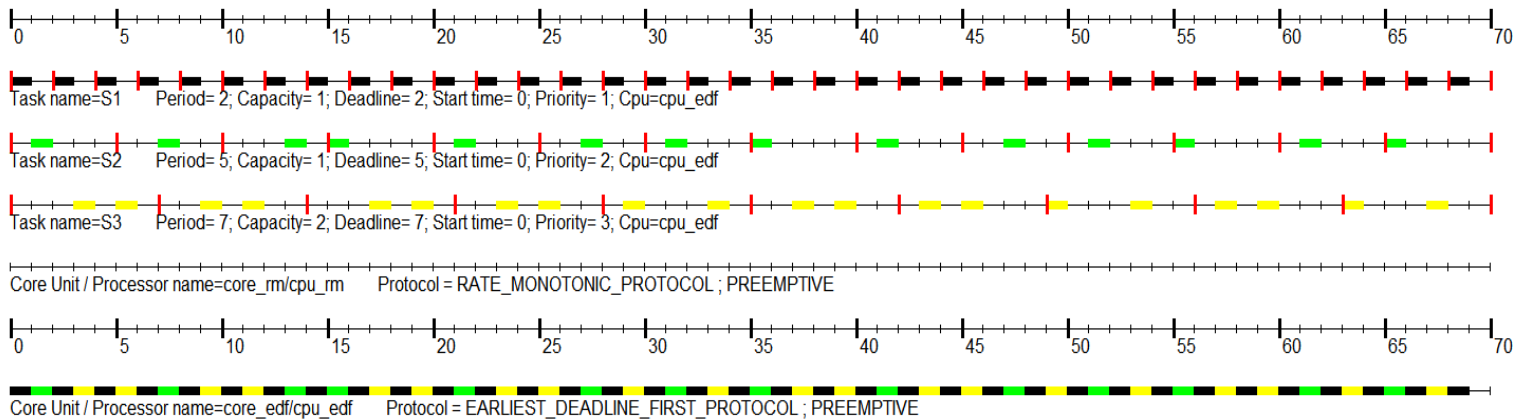
Cheddar RM Simulation (Failure):



Scheduling simulation, Processor cpu_rm :

- Number of context switches : 68
- Number of preemptions : 10
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 8/worst , missed its deadline (absolute deadline = 7 ; completion time = 8)
- Some task deadlines will be missed : the task set is not schedulable.

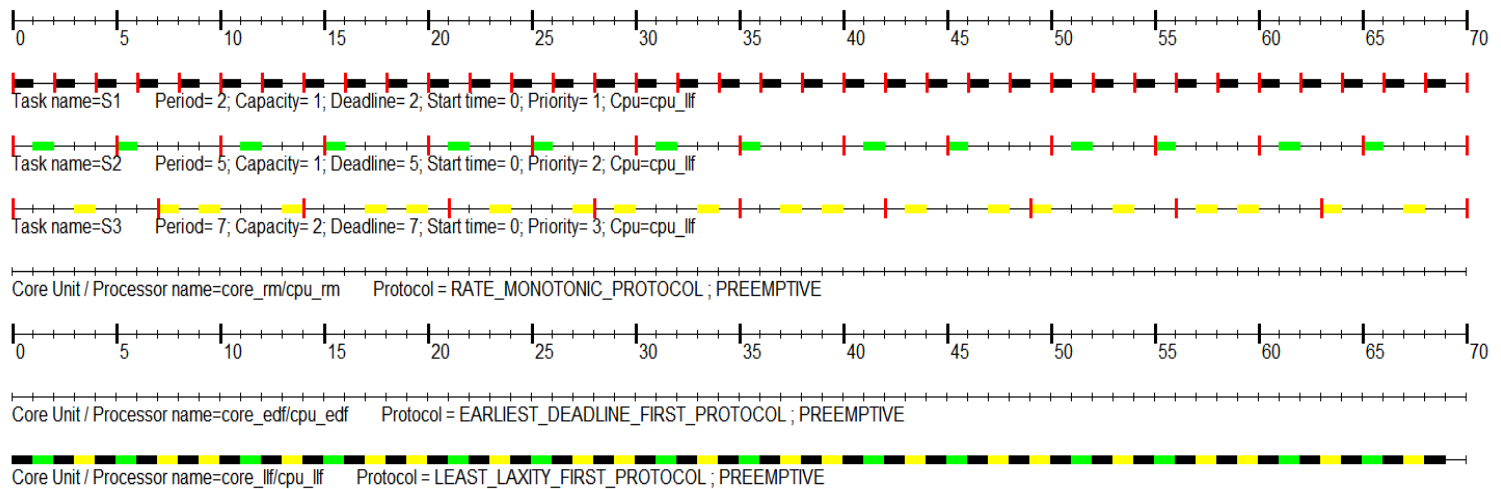
Cheddar EDF Simulation (Success):



Scheduling simulation, Processor cpu_edf :

- Number of context switches : 68
- Number of preemptions : 10
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 6/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Cheddar LLF Simulation (Failure – Faulty Simulation by Cheddar):



Scheduling simulation, Processor cpu_llf :

- Number of context switches : 68
- Number of preemptions : 10
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 8/worst , missed its deadline (absolute deadline = 7 ; completion time = 8)
- Some task deadlines will be missed : the task set is not schedulable.

Example 2

S₁: T₁ = 2, C₁ = 1; S₂: T₂ = 5, C₂ = 1; S₃: T₃ = 7, C₃ = 1; S₄: T₄ = 13, C₄ = 2

Code Output: **Infeasible**

Cheddar Feasibility Analysis: **Infeasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **infeasible** since all tests have been failed by given set of services. **First missed deadline is at 13th time slot – for S₄.** This has taken place **because of static priority assignment** – the higher priority services are running even when they are not critical with respect to their own deadlines. **This is effectively a priority inversion.**

EDF and **LLF** has to be **feasible** because the scheduling is **fully preemptive** with **dynamic priority** and the **total utilization is not exceeding 100%**. However, it might be impractical since EDF and LLF both requires quite a bit of information about all services, their deadlines, their required processing time, etc.

Example 3

S₁: T₁ = 3, C₁ = 1; S₂: T₂ = 5, C₂ = 2; S₃: T₃ = 15, C₃ = 3

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Example 4

S₁: T₁ = 2, C₁ = 1; S₂: T₂ = 4, C₂ = 1; S₃: T₃ = 16, C₃ = 4

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Example 5

S₁: T₁ = 2, C₁ = 1; S₂: T₂ = 5, C₂ = 2; S₃: T₃ = 10, C₃ = 1

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Example 7

S₁: T₁ = 3, C₁ = 1; S₂: T₂ = 5, C₂ = 2; S₃: T₃ = 15, C₃ = 4

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Example 9

S₁: T₁ = 6, C₁ = 1; S₂: T₂ = 8, C₂ = 2; S₃: T₃ = 12, C₃ = 4; S₄: T₄ = 24, C₄ = 6

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Example 10

S₁: T₁ = 2, C₁ = 1; S₂: T₂ = 5, C₂ = 1; S₃: T₃ = 7, C₃ = 1; S₄: T₄ = 14, C₄ = 2

Code Output: **Feasible**

Cheddar Feasibility Analysis: **Feasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **feasible** since all tests have been passed by given set of services.

Example 11

S₁: T₁ = 3, C₁ = 1; S₂: T₂ = 6, C₂ = 2; S₃: T₃ = 9, C₃ = 3

Code Output: **Infeasible**

Cheddar Feasibility Analysis: **Infeasible**

Result: **Everything Matches** (Including the analysis done in spreadsheet)

RM scheduling has to be **infeasible** since all tests have been failed by given set of services. **First missed deadline is at 9th time slot – for S₃. This has taken place because of static priority assignment** – the higher priority services are running even when they are not critical with respect to their own deadlines. **This is effectively a priority inversion.**

EDF and **LLF** has to be **feasible** because the scheduling is **fully preemptive** with **dynamic priority** and the **total utilization is not exceeding 100%**. However, it might be impractical since EDF and LLF both requires quite a bit of information about all services, their deadlines, their required processing time, etc.

Code Summary:

- The code utilizes various algorithms – which are based upon some specific theorems, to determine whether a given set of services would be feasible with RM scheduling or not.
- For input, separated arrays are used – in which, indexes represent service number and priority. The lower the index, the higher the priority.
- Each of these arrays – containing time periods, and worst case execution timings – are fed to the functions – which test the given example using scheduling point test or completion time test.
- Both of these functions work in very specific manner, utilizing cascading loops – to calculate terms presented in the equation governed by the theorem – and finally the decision is made based on the passing/failure criteria of the equation. This is returned as a binary value – where ‘1’ represents success, while ‘0’ represents failure.

Q4: 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.

Solution:

Key Assumptions from Liu and Layland Paper

1. The requests for all tasks for which hard deadlines exist are periodic, with constant interval between requests – This assumption potentially discards all the situations wherein, a task with hard deadline can exist – without being periodic in nature. This will affect the system which can have a critical error responding behavior.
2. The tasks are independent in that requests for a certain task do not depend on the initiation or the completion of requests for other tasks – This assumption prevents the use of this measure in systems wherein, multiple periodic tasks with hard deadlines, could actually be dependent on some other task.
3. Run-time for each task is constant for that task and does not vary with time. Run-time here refers to the time which is taken by a processor to execute the task without interruption – This assumption could be difficult to achieve in some systems, given that many processes have multiple paths to take while running, each path having significantly different length than the other. Therefore, ensuring that task execution time is always constant – is not easy.

Assumptions and Constraints for Least Upper Bound

1. The run-times of the tasks are such – that these will fully utilize the processor, while also minimizing the processor utilization factor: This is an assumption stated in the very initial stage of LUB derivation. However, by nature, it also puts constraint on the LUB itself. Least Upper Bound therefore, can't be used directly in a case where the given capacities of the tasks, are not fulfilling this condition.
2. The periods for all given tasks are in decreasing order: This is an assumption made for the mathematical derivation of the LUB. It clearly indicates the need to arrange given services, and their priorities in proper order.
3. The ratio between any two request periods must be less than 2: As per the Theorem 4 in Liu and Layland paper, it is a constraint on LUB derivation that the given set of services can't have request periods that are too far apart from each other.
4. If $C_1 = T_2 - T_1 + \Delta$ ($\Delta > 0$), then take $C_1' = T_2 - T_1$, $C_2' = C_2 + \Delta$, $C_m' = C_m$ such that these services also fully utilize the processor: This is clearly an assumption, which is taken in order to prove that $C_1 = T_2 - T_1$.
5. G_0 is 1: The term G_0 is taken 1 for convenience. This puts a constraint on the overall LUB derivation, in my opinion.

LUB Derivation steps with Tricky Math/Difficult to Understand (since I already read chapter 3 before assignment 1, and I haven't gained any new insight on this topic yet, my answer to this portion will not change from the assignment 1)

1. **Understanding:** Theorem 4 states a condition '... and the restriction that the ratio between any two request periods is less than two ...' – in this, 'any' is the keyword – which implies that I cannot have request periods such as 1,2,3,4 – since pairs 1 & 3 as well as 1 & 4 violates the given restriction. Now, I do not understand that how this condition – transformed into showing that $C_{m-1} = T_m - T_{m-1}$.
2. **Tricky Math:** How is $C_{m-1} = T_m - T_{m-1}$ translated into $C_m = T_m - 2*(C_1 + C_2 + \dots + C_{m-1})$
3. **Tricky Math:** How step 2 is derived from step 1 in following screen capture

$$= \sum_{i=1}^{m-1} [g_i - g_{i+1}(g_i + 1)/(g_{i+1} + 1)] + 1 - 2[g_1/(g_1 + 1)] \dots 1$$

$$= 1 + g_1[(g_1 - 1)/(g_1 + 1)] + \sum_{i=2}^{m-1} g_i[(g_i - g_{i-1})/(g_i + 1)]. \dots 2$$

4. **Tricky Math:** The first derivative of step 2 in above picture
5. **Tricky Math:** How that derivative results in $U = m*(2^{1/m} - 1)$
6. **Understanding:** This entire portion

Let

$$\begin{aligned} C_1' &= T_2 - T_1 \\ C_2' &= C_2 + \Delta \\ C_3' &= C_3 \\ &\vdots \\ C_{m-1}' &= C_{m-1} \\ C_m' &= C_m \end{aligned}$$

Clearly, $C_1', C_2', \dots, C_{m-1}', C_m'$ also fully utilize the processor. Let U' denote the corresponding utilization factor. We have

$$U - U' = (\Delta/T_1) - (\Delta/T_2) > 0.$$

