Prerit Oberai
Nikhil Divekar

ECEN 5623

Exercise 2

February 21, 2019

Introduction

The purpose of exercise is to dive deeper into Rate Monotonic (RM), Earliest Deadline First (EDF) and Least Laxity First (LLF) scheduling policies and investigate how they can be simulated on an RTOS. Specifically, we used a Jetson TK1 board for simulation purposes and so all results obtained should be reproducible with the appropriate reference material and included scripts.

Problem 1

[5 points] Make yourself an account on your Dev Kit.Add your new user account as a "sudoer" using"visudo" right below root with the same privileges. Do a quick "sudo whoami" to demonstrate success. Logout of Linux and test your login, then logout. Use Alt+Print-Screen to capture your desktop and save as proof you set up your account.

Solution:

Therefore, in order to setup a new account and give ourselves root privileges, we followed the following steps:

- 'Sudo adduser prerito'
- Adding a new password, and filling out the required information for a new user
- Updating the privileges using sudo visudo

Upon doing this, we got the following results:

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

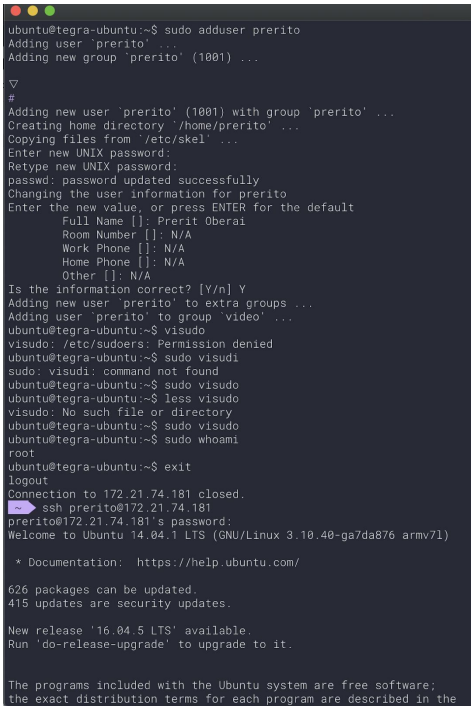


Figure 1

Notice from the screen shot above, we were able to successfully make a new user with username prerito and then login the Jetson TK1 board with the new user. We were also able to give this user root priveleges as the following screenshot shows:

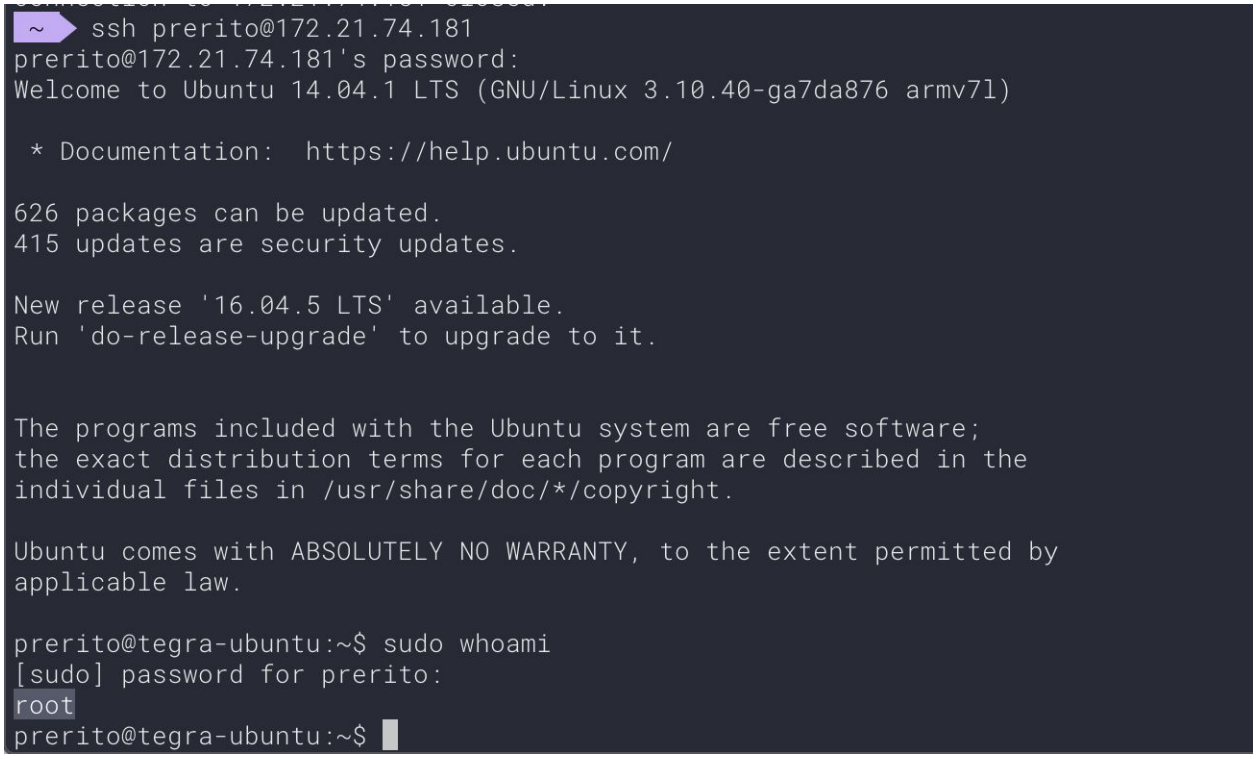


Figure 2

Therefore, as it can be seen from the highlighted cursor, the user prerito is treated as root and therefore a “sudoer”.

Problem 2

[10 points] Read the paper Architecture of the Space Shuttle Primary Avionics Software System 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:

Explanation of the PASS and Key Points

The frequency executive architecture that the Space Shuttle Primary Avionics Software System (PASS) implemented is regarded as one of the most complex computer programs ever developed. This is because it was very largely constrained by a multitude of drivers such as the Data Processing System (DPS), General Purpose Computer (GPC), memory/CPU, the multi-computer redundancy management and synchronization, operational sequencing and mode control, and the man to machine interfaces which all factored into its system design. Additionally, there were physical constraints along with memory capacity constraints (i.e. the AP-101 could not handle all of the software) which meant that the implementation of the design had to take into account how to support mission critical phases without compromising the mission.

The result of such tight constraints and requirements resulted in the segmentation of the PASS into 8 different “function combinations” where each function would is defined by its operational sequence. This operational sequence is loaded into the general purpose computer of the main memory from the hierarchical mass memory upon initiation of the operational sequence. In addition, the OPS is fragmented into the mass memory in a manner which optimizes how it is fetched at the time of initiation meaning that only the OPS overlay has to be loaded from mass memory when a specific combination is triggered.

Man-Machine Interface
Now, one of the major components of PASS was how it would be operated with the end user to monitor and control the Orbiter avionics system. As such, it was implemented in a manner which assumes the user is knowledgeable about avionics and also would only interact with the system minimally - both in time and efforts. Consequently, this structure was broken into three substructures which were comprised of the major modes, specialist functions and display functions.

- Major Mode: This mode allows the ability of the OPS to be split into major steps or sequences. As such, these modes are further substrucuted into clocks that are linked to display pages resulting in an orderly sequence for the user to be able to control the software.
- Specialist Functions: This mode is only triggered upon a keyboard entry by the user and once triggered, it is executed independent of any other process running in the OPS. This is mostly used for secondary and background processes but also has links to the display meaning that the user can monitor the results of the active process.
- Display Function: Finally, this function is unique in that it does not do any processing and is only used to monitor the results from a Major mode and/or the specialist functions.

Prerit Oberai ECEN 5623
Nikhil Divekar

Exercise 2

February 21, 2019

Systems Software

In addition to the handling the inputs from the user and displaying the outputs, the PASS must also be able to accommodate the management and control of the general purpose computer's internal resources and external interfaces, loading and initialization of processes and inter-computer communication. However, it needed to be able to perform these tasks without being overrun at the process and system level. Additionally, because it was anticipated that the requirements of the Shuttle Program, at large, would continue to change, a non-synchronous approach was adapted for the PASS architecture.

- FCOS: The Flight Computer Operating System (FCOS) had 3 primary functions - process management, I/O management and DPS Configuration Management.
    - Process Management: It primarily uses a multitasking priority queue in order to schedule and allocate CPU resources when requests are spawned and these resource requesters are ofer made through service interfaces which have reference to the event, frequency and priority of the service.
    - I/O Management: Controls the allocation of the input and output processor resources. In this case, however, each general purpose computer command data buses which are assigned to it for specifically I/O control. This lets the master sequence controller for each GPS to be able to control the 24 bus control elements and this process is completed asynchronously.
    - DSP Configuration Management: Finally, the the data processing systems allows for the transfer of loading the code and data between mass memory and the of GPC main memory. This is done synchronously at about 350 times per second (i.e. 350 Hz) via special process that works in parallel to the other FCOS processes.
- System Control: The system control is responsible for the initialization and configuration control of of the DPS configuration management system as well as the associated avionic data network. The primary functions include synchronizing the internal timers with the avionic system master timing unit (done by the FCOS), establish interface with primary and secondary relationships with other GPC's and establish the basis for further system configuration.
- User Interface: Finally, as discussed previously, the user (i.e. crew) need to be able to interface with PASS and this is often done via keyboard entries and the process information is displayed on a display for on-board software as well as offline processes.

Applications Software

There is another component to PASS in which it has support ground and in-flight Shuttle operations and this is primarily handled by the applications software. The primary applications are Guidance, Navigation and Control (GN&C), vehicle systems management (SM), and vehicle checkout (VCO).

- GN&C: The Guidance, Navigation and Control completes the following tasks: determines the vehicle position, velocity and attitude, sensor redundancy management, displays the data entry abilities to monitor and control the avionics subsystem and issues the engine and affector

commands for a mission from lift-off. In terms of embedded software, the GN&C is a cyclic closed loop application which performs about 200 principal functions. The cyclic process (i.e. executive) controls the initiation and phasing of the principal functions through the use of an FCOS process scheduling SVC. There is a 40 ms minor cycle in which all critical flight processing must be completed and in order to regulate this, there are 3 executive structures included in the design which operate at different rates - a high priority cycle at 25 Hz rate to process directly with vehicle flight control, and mid-low priority functions operating at 0.25-6.25 Hz.

- SM: The vehicle systems management application provides the crew with status monitoring for orbiters which are not involved directly with the vehicle flight control. The design of this system includes sets of parametric data tables which store information on how the processing should be managed for each processes where the inputs supplied are data items specific to the specified orbiter. The data acquisition is executed at 5 Hz and the parameter preconditioning and fault detection is executed at 1Hz. There is an additional special process which run on the SM controlled by the executive dispatcher responsible for reconfiguring telemetry downlink formal, controlling payload operations and operating the remote manipulation system for deployment and retrieval at a rate of 5 Hz.
- VCO: The Vehicle Checkout application is intended to support the avionics system initialization and checkout under control of the ground and flight crew. Specifically, its principal function is to work with associated I/O interfaces which are configured for ground and in-flight checkouts. The paper also discussed about the Test Control Supervisor (TCS) which allows the end user to be able to command and verify specific processes allowing for test sequences to be developed for execution in a vehicle checkout operating state.

Evidently, it can be seen that the PASS architecture is very complex in its nature due to the numerous requirements it had to have met but it provided enough of a foundation to accommodate a very dynamic environment for the orbiters.

Advantages over Real-Time Threading

- One of the advantages discussed in the paper for cyclical executive approach over threading was that it was repeatable meaning that there was a predictable schedule and not erroneous execution time (i.e. jitter) for each task to be completed.
- Another advantage for not using real-time threading is that context switches are very minimized because there are no deadlines which have to be met meaning that threads don't have to be preempted or dispatched multiple times before the task itself is completed. This results in more predictability in the scheduling of the tasks.
- Finally, building on the fact that there is less context switching, there is also less overhead and wasted memory space allowing for a more reliable serviceability of tasks.

Disadvantages over Real-Time Threading

- Unfortunately, when we don't implement a scheduling policy via an RTOS, this means that we have no way to ensure that all the tasks will run to completion or that it's even feasible for the tasks to run all in one cycle. As such, the cycle might have to run multiple times in order for all the tasks to complete.
- Additionally, building on the notion that cyclic execution really doesn't really have priority enforcement other than the ordering of the ISR table, there can be events in which certain critical deadlines are not met because they are not given priority over other tasks which are less critical. For instance, this works just fine when there a few number of services requesting hardware and access to the ISR table but for many many services (such as this application), it can lead to blocking in extraneous situations and this can prove to be fatal when meeting critical deadlines.
- [NEED ONE MORE DISADVANTAGE HERE]

Problem 3

[5 points] Read the paper Building Safety-Critical Real-Time Systems with Reuseable

Prerit Oberai ECEN 5623
Nikhil Divekar

Exercise 2

February 21, 2019

Cyclic Executives. In other embedded systems classes you built ISR (Interrupt Service Routine) processing software and polling/control loops to control for example stepper motors – describe the concept of the Cyclic Executive and how this compares to the Linux POSIX RT threading and RTOS approaches we have discussed.

Solution:

Introduction to cyclic executives:

The paper "Building Safety-Critical Real-Time Systems with reusable cyclic executable" talks about how safety requirements of safety-critical real time systems can be achieved by the means of cyclic executable. This paper considers particularly the use case of Ada 95 to demonstrate reusable cyclic executable implementation.
Some application areas like railway or air-traffic control have stringent safety requirements apart hard real-time requirements. Most of the safety-critical system are designed using synchronous architecture based on cyclic executives. Benefits of using synchronous architecture in case of cyclic executable is no interleaving of operation, predictable system and can be easily tested, small code size. However, there are some problems like more complexity of building cyclic schedule and low abstraction. Ada 95 and C++ provide strong usage of abstract data type and generic packages or templates which can be used to achieve synchronous architecture for cyclic executive.

Requirements for generic cyclic executive:

The main purpose of cyclic executive is to make executive into generic, reusable component for a wide variety of real time systems. A cyclic schedule is a table which contains full execution pattern for a major cycle. This cycle consists of number of minor periodic cycles. Every minor cycle consists of variable number of frames of various durations. Each process is one or more frames of minor cycles.
List of operation modes:

1. Duration of minor and major cycle periods.
2. Number of minor cycles per major cycle.
3. Maximum number of frames per minor cycle.
4. Process that has to be executed

Cyclic scheduler structure:

Cyclic scheduler is the main component of the synchronous architecture. It is an iterative procedure that allows the explicit multiplexing of a periodic process set in one processor. A cyclic schedule is a table which contains full execution pattern for a major cycle. This cycle consists of number of minor periodic cycles. Every minor cycle consists of variable number of frames of various durations. Each process is one or more frames of minor cycles.
So, the order of structure of cyclic scheduler structure from higher level to lower level is as given below:

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

---

1. Cyclic Scheduler.
2. Major Schedule.
3. Minor Schedule.
4. Frames.

Each tick of minor cycle indicates the end of minor cycle. At that time, the process sequence corresponding to this minor cycle must have finished the execution. If not, minor_cycle_overrun occurs. Each minor schedule is divided into frames. Within each frame, only one process of sequence corresponding to this minor cycle must have finished the execution. When the frame ends, the corresponding process must have finished, or it results in frame_overrun.

Scheduler exception:

An exception is an event that leads to temporary suspension of the normal program execution. Ada allows code exception handlers to capture predefined exceptions or user defined exceptions. When an exception is activated in a unit, its execution is terminated, and an exception handler takes over the control of the execution. Exceptions are the most natural way of dealing with abnormal and erroneous execution. There are predefined exception and user can have their own new exception as well. Use of access types and task operation such as delays and asynchronous transfer of control cannot be used in the cyclic executives.

Generic architecture for synchronous real-time systems:

A generic architecture is a template for building a system from component blocks, interconnected by set of standard elements. The generic architecture is made of the following components:

1. Structures:

These allow processes, modes, recovery groups and schedules to be defined. They are independent of development system and execution architecture.

2. Executive:

This is responsible for isolation of hardware, compiler and run-time system dependencies.

3. Shells:

These are generic templates for building application process and defining system structure.

Development view of cyclic executive software architecture is organized in hierarchy layers. Different layers are application, scheduler, interface, run-time system, hardware. Application layer. Different layers and their corresponding components of development view is as given below:

1. Application:

System_Structure_Shell, Main_Structure_Shell, Periodic_Process_Shell.

---

2. Scheduler:
Processes, Modes, Recovery_Groups, Schedules.

3. Interface:
Run_Time_System_interface, Cyclic_Scheduler_Timers,
4. Run-Time Systems:
Ada-Run-Time System, Hardware_Interface.

5. Hardware:
Hardware.

ADA Implementation:

In ADA 95, features like delays, dynamic storage allocation, recursive procedure call and loops are not used to ensure safety-critical system. Exceptions are used for error detection and recovery mechanisms. Some other important characteristics include Enhanced generic parameter including package and access type, enhanced access types like static objects and procedures. Parameterless procedure, hierarchical public and private package, standard interfaces to procedure written in other language. These features are very important for the construction of generic synchronous architecture which in turn enhance the flexibility, reliability and maintainability of the application code.

Hence, this paper demonstrates the definition of cyclic executable, its component, the synchronous architecture and implementation of all above in ADA.

Comparison to Linux and RTOS:

A cyclic executive architecture consists of main loop and ISR. Generally, the code runs in infinite loop.

Following are the points of cyclic executive as compared to Linux/RTOS:

1. Since cyclic executive just contains main loop (generally infinite loop) and ISR, there is very less branching in the program as compared to Linux/RTOS which may have many context switches and thus very less overhead.
2. RTOS provides more level of abstraction and portability as compared to the cyclic executive.
3. Implementation of cyclic executive is simpler than complete Linux/RTOS system.
4. Cyclic executives are non interrupt-driven systems that can provide the illusion of simultaneity by taking advantage of relatively short processes on a fast processor in a continuous loop. While on the other hand, Linux and RTOS has to do with task, thread for synchronization and communication. The cyclic executive merely contains ISR routine which might be triggered on the basis of external events.

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623

February 21, 2019

---

5. Cyclic executive works best with short process generally. RTOS gives almost equal performance regardless of code size.
6. Kernel is the core of RTOS which helps with all scheduling mechanism. There is a separate scheduler layer as described in development view of cyclic executive.
7. RTOS provides more flexibility for code enhancement as it has threads involved. It's difficult to add code to the cyclic executive as you have to keep the existing code intact and add onto that.
8. Cyclic Executive generally runs on the single processor while RTOS might have Asynchronous multi-processing and synchronous multiprocessing architecture.

Problem 4

[50 points] Download Feasibility example code and build it on a Jetson, DE1-SoC or TIVA or Virtual Box and execute the code. Compare the tests provided to analysis using Cheddar for the first 4 examples. Now, implement the remaining examples [5 more] that we reviewed in class. Complete analysis for all three policies using Cheddar (RM, EDF, LLF). In cases where RM fails, but EDF or LLF succeeds, 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: Looking through the feasibility code (as included in references without any modification), it appears that there are two main tests the script has implemented - the Completion Feasibility Test and the Scheduling Point Feasibility Test. Therefore, building the unmodified code on the Jetson TK1, we get the following output:

```
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ make
gcc -O0 -g   -c feasibility_tests.c
gcc  -O0 -g   -o feasibility_tests feasibility_tests.o -lm
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ls -lrt
total 36
-rw-r--r-- 1 ubuntu ubuntu  6083 Feb 15 18:54 feasibility_tests.c
-rw-r--r-- 1 ubuntu ubuntu   373 Feb 15 18:54 Makefile
-rw-rw-r-- 1 ubuntu ubuntu  8200 Feb 15 18:54 feasibility_tests.o
-rwxrwxr-x 1 ubuntu ubuntu 11210 Feb 15 18:54 feasibility_tests
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ./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
```

Figure 3

What the screenshot above indicates is that when the first 4 examples are implemented, EX-0 passes the feasibility test, Ex-1 does not pass the feasibility test, Ex-2 does not pass the feasibility test and both Ex-3 and Ex-4 pass the feasibility tests. Note that the parameters C, T

---

and U are consistent with what's been presented in class so far where C is how low it takes for a task to be completed, T is the period of the task and its deadline and U is the CPU utilization using this policy. For instance, Ex-0 has three services, where the first service has a period of 2 but only takes 1 time instance to complete, the second service has a period of 10 instances of time but only takes 1 time instance to complete and so forth.

Now, comparing these results with what Cheddar reports using the RM, EDF, LLF policies, we get the following results:

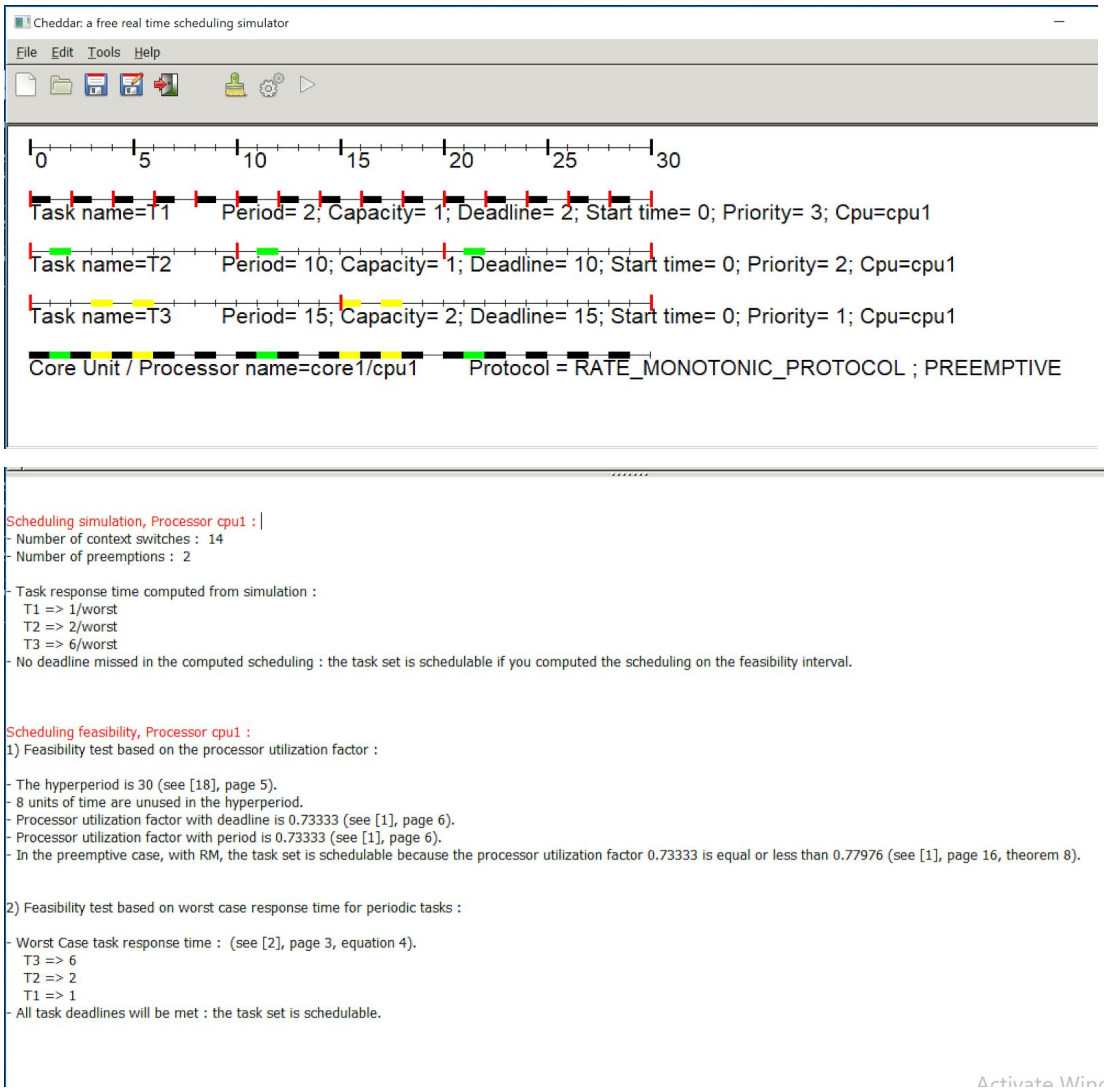
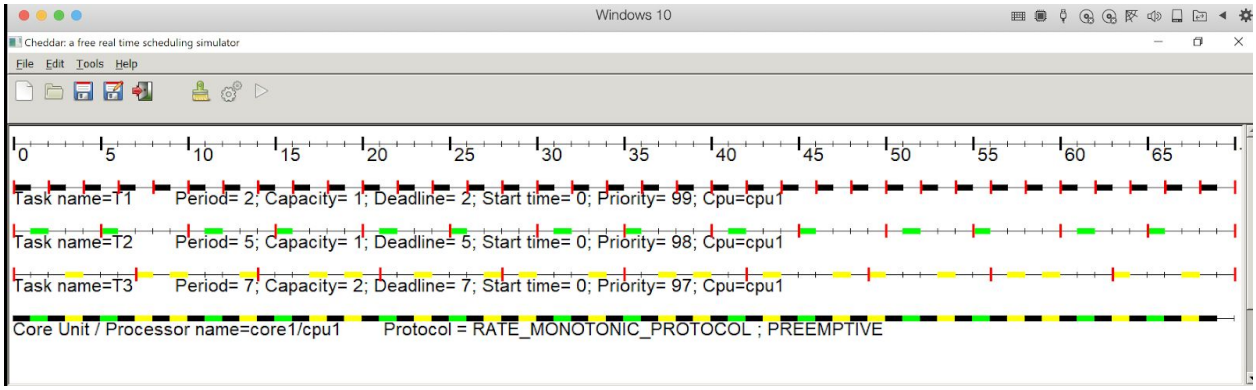


Figure 4: Cheddar Results for Ex-0 Scheduling with Rate Monotonic Policy

---

Therefore, as it can be seen, from our code output in Figure 3, Ex-0 was regarded as feasible and this aligns with Cheddar's output (in Figure 4) as it clearly states that "All the task deadlines will be met: the task set is schedulable". We also have the same result for the CPU utilization confirming that our schedule was scheduled correctly and that the results match one another.

Doing a similar analysis for Ex-1, we get the following results:




```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  68
- Number of preemptions :  10

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 8/worst ,  missed its deadline (absolute deadline =  7 ; completion time =  8)
- Some task deadlines will be missed : the task set is not schedulable.


Scheduling feasibility, Processor cpu1 :
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).
   T3 => 8,  missed its deadline (deadline =  7)
   T2 => 2
   T1 => 1
- Some task deadlines will be missed : the task set is not schedulable.
```

Figure 5: Cheddar Results for Ex-1 Scheduling with Rate Monotonic Policy

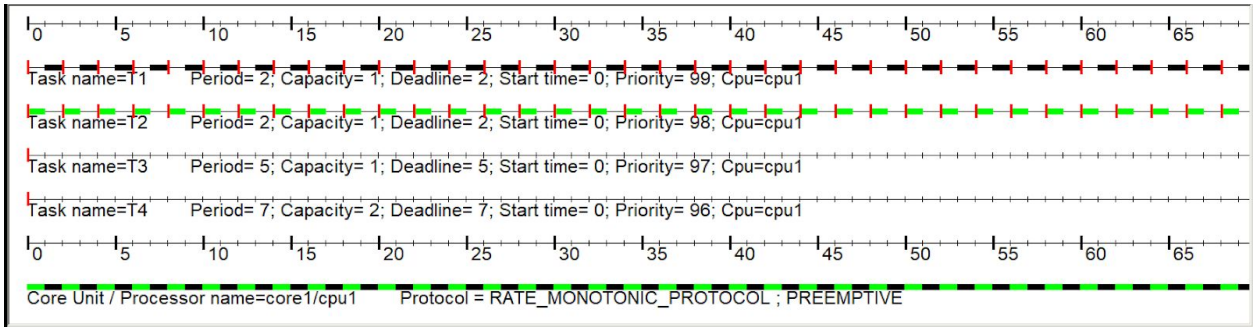
From comparing the Cheddar results in Figure 5 and the results from our Jetson TK1 board in Figure 3, we can see that both of those results conclude that this schedule is NOT feasible using via RM policy. Specifically, Figure 5 states that "Some tasks deadlines will be missed" - i.e. Task 3 and this overlaps with what is shown in the scheduler output in Figure 4.

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

For Ex-2, we get the following results:

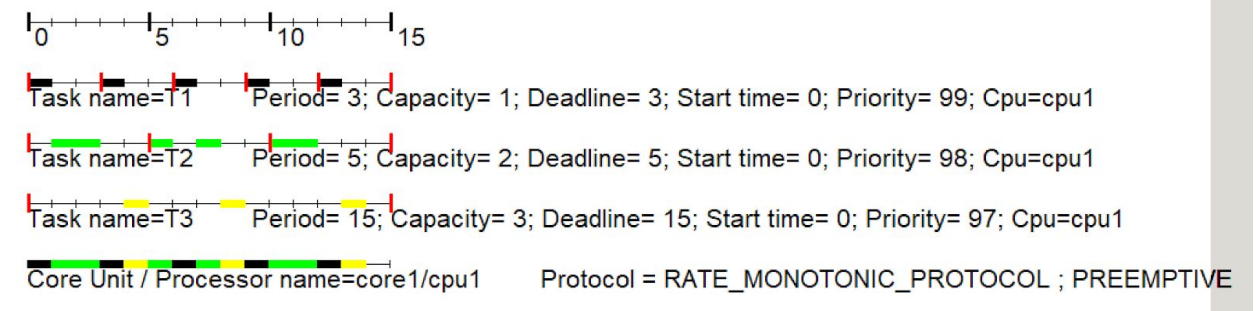



```
Scheduling simulation, Processor cpu1 :
- Number of context switches : 69
- Number of preemptions : 0

- Task response time computed from simulation :
  T1 => 2/worst
  T2 => 1/worst
  T3 => 0/worst , response time not computed since the task did not run all its capacity
  T4 => 0/worst , response time not computed since the task did not run all its capacity
- One or several tasks did not complete their execution.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 70 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.48571 (see [1], page 6).
- Processor utilization factor with period is 1.48571 (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 1.48571 is more than 0.75683 (see [1], page 16, theorem 8).


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

- Processor utilization exceeded : can not compute worst case response time with this task set.
```

Figure 6: Cheddar Results for Ex-2 Scheduling with Rate Monotonic Policy

From comparing the Cheddar results in Figure 6 and the results from our Jetson TK1 board in Figure 3, we can see that both of those results conclude that this schedule is NOT feasible using via RM policy. In this case, Tasks 3 and 4 were not run to its capacity and note that the processor utilization actually exceeds 1.0 which is not even possible.

For Ex-3, we get the following results:

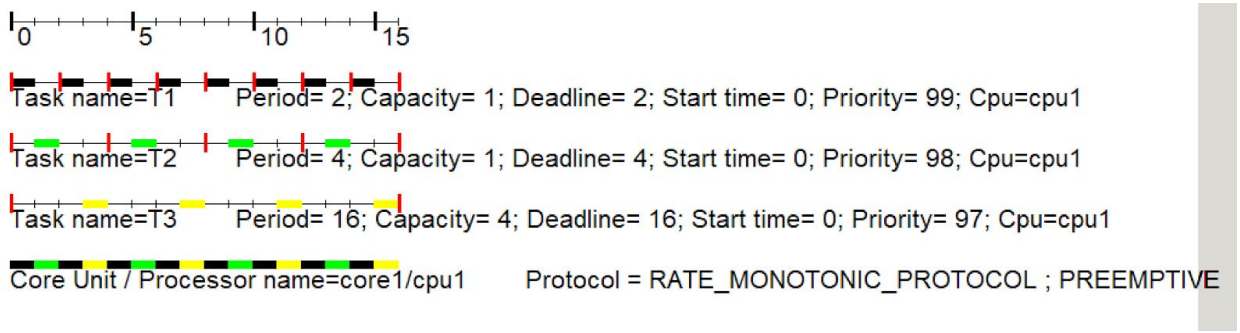


```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  11
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 14/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 15 (see [18], page 5).
- 1 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.93333 (see [1], page 6).
- Processor utilization factor with period is 0.93333 (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 0.93333 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).
   T3 => 14
   T2 => 3
   T1 => 1
- All task deadlines will be met : the task set is schedulable.
```

Figure 7: Cheddar Results for Ex-3 Scheduling with Rate Monotonic Policy

The Cheddar results in Figure 7 and the results from our Jetson TK1 board in Figure 3 show that Ex-3 is feasible via RM Policy. The CPU utilization calculated is 93% which is > RM LUB meaning that no definitive conclusion can be made but when the WCET case is taken into consideration, all the tasks still meet their appropriate deadlines.

Finally, for Ex-4, we get the following results:



```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  15
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 16/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 16 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 1.00000 is equal or less than 1.00000 (see [19], page 13).


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

- Worst Case task response time :  (see [2], page 3, equation 4).
   T3 => 16
   T2 => 2
   T1 => 1
- All task deadlines will be met : the task set is schedulable.
```

Figure 8: Cheddar Results for Ex-4 Scheduling with Rate Monotonic Policy

Consequently, the Cheddar results in Figure 8 and the results from our Jetson TK1 board in Figure 3 show that Ex-4 is also is feasible via RM Policy. The CPU utilization calculated here is 100% and with the worst case execution time (WCET), all the tasks still meet their appropriate deadlines.

Now, the next part of this question requires that we update the feasibility test code to implement the tasks as outlined in Examples 5-8 on Canvas. The modified code is zipped up in the same package as this file and also included in references. However, methodology behind updating the code was to just add more arrays in the format of:
U32_T ex#_period[] = {T1, T2, T3};
U32_T ex#_wcet[] = {C1, C2, C3};
And let the code compute whether it was feasible or not as per Rate Monotonic Policy. Doing so yielded the following results:

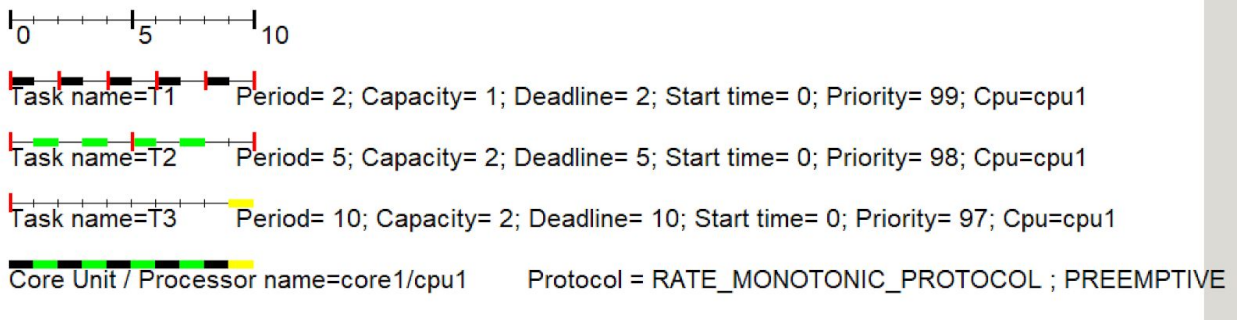
```
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ./feasibility_tests
******** Completion Test Feasibility Example
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): INFEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): INFEASIBLE


******** Scheduling Point Feasibility Example
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): INFEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): INFEASIBLE
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$
```

Figure 9: Functionality Script run on Jeston TK1 with Rate Monotonic Policy

As it can be seen from Figure 9, Ex-6 and Ex-8 are infeasible whereas Ex 5 and Ex 7 are feasible and able to be scheduled via rate monotonic policy. Note that all the schedules are also using 100% CPU utilization rate and so if we were to plug this into Cheddar, it would complete the WCET analysis in order to see if can be scheduled adequately.

Sure enough, when seeing this in Cheddar, we get the following results:

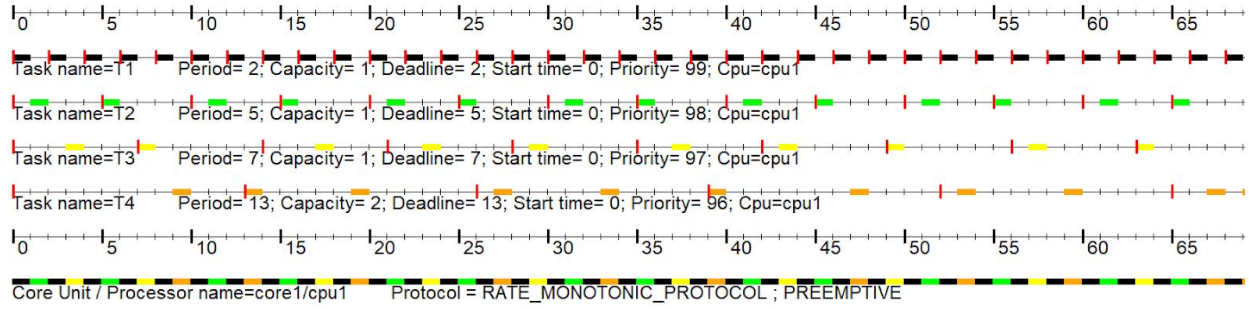
For Ex-5



```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  9
- Number of preemptions :  2

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 4/worst
   T3 => 0/worst ,  response time not computed since the task did not run all its capacity
- One or several tasks did not complete their execution.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 10 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.10000 (see [1], page 6).
- Processor utilization factor with period is 1.10000 (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 1.10000 is more than 0.77976 (see [1], page 16, theorem 8).


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

- Processor utilization exceeded : can not compute worst case response time with this task set.
```

Figure 10: Cheddar Results for Ex-5 Scheduling with Rate Monotonic Policy

For Ex-6

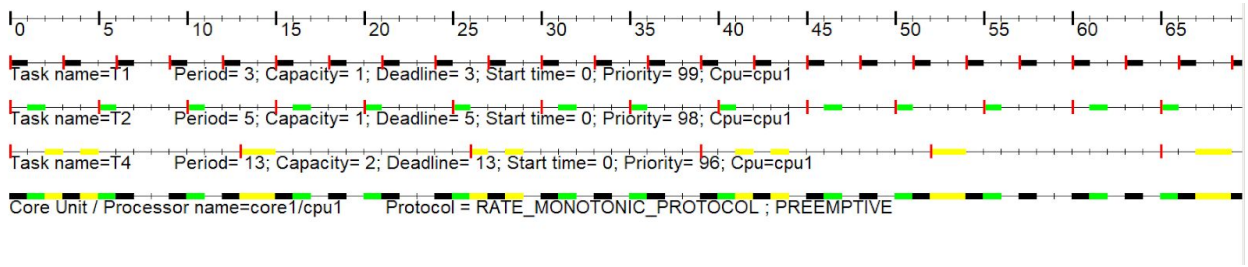



```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 4/worst
   T4 => 16/worst ,  missed its deadline (absolute deadline =  13 ; completion time =  14),  missed its deadline (absolute deadline =  26 ; completion time =  28),  missed its deadline (absolute deadline =  39 ; completion time =  40),  missed its
- Some task deadlines will be missed : the task set is not schedulable.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (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 0.99670 is more than 0.75683 (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).
   T4 => 16,  missed its deadline (deadline =  13)
   T3 => 4
   T2 => 2
   T1 => 1
- Some task deadlines will be missed : the task set is not schedulable.
```


Figure 11: Cheddar Results for Ex-6 Scheduling with Rate Monotonic Policy

---

For Ex-7




```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  115
- Number of preemptions :  10

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T4 => 5/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.



Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 195 (see [18], page 5).
- 61 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.68718 (see [1], page 6).
- Processor utilization factor with period is 0.68718 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.68718 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).
   T4 => 5
   T2 => 2
   T1 => 1
- All task deadlines will be met : the task set is schedulable.
```

Figure 12: Cheddar Results for Ex-7 Scheduling with Rate Monotonic Policy

---

Finally, for Ex-8, we have:


```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 4/worst
   T4 => 16/worst ,  missed its deadline (absolute deadline =  13 ; completion time =  14),  missed its deadline (absolute deadline =  26 ; completion time =  28),  missed its deadline (absolute deadline =  39 ; completion time =  40),  missed its c
- Some task deadlines will be missed : the task set is not schedulable.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (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 0.99670 is more than 0.75683 (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).
   T4 => 16,  missed its deadline (deadline =  13)
   T3 => 4
   T2 => 2
   T1 => 1
- Some task deadlines will be missed : the task set is not schedulable.
```

Figure 13: Cheddar Results for Ex-8 Scheduling with Rate Monotonic Policy

As it can be seen from the results of Cheddar, examples 5 and 7 are feasible to be scheduled via RM policy but examples 6 and 8 are not and this aligns with the conclusions from the modified script run on Jeston TK1 as well. Specifically, for examples 6 and 8, the RM policy misses task's 4 deadlines over and over again.

Now, because examples 6 and 8 failed the RM Policy scheduler, we are told to implement it via the EDF policy. Consequently, when implemented this on the Jetson TK1 as a script, we get the following result. Note that the EDF script as also included in the final submission folder.

```
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ./feasibility_test_EDF
******** EDF Test Feasibility Example
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): LCM of services is 3
Utilization is 1.000000 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): LCM of services is 4
Utilization is 0.996703 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
Running service 3 in time period 4
FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): LCM of services is 3
Utilization is 1.000000 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 2 in time period 3
FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): LCM of services is 4
Utilization is 0.996703 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
Running service 3 in time period 4
FEASIBLE
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$
```

Figure 14:Functionality Script run on Jeston TK1 with EDF Policy

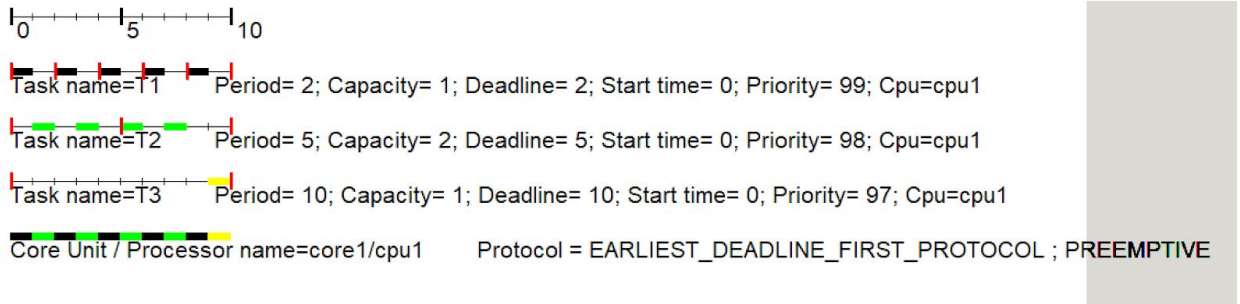
Now, working through the least laxity first (also included in references package), we get the following output:

```
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ./feasibility_test_LLF
******** LLF Test Feasibility Example
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): LCM of services is 3
Utilization is 1.000000 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): LCM of services is 4
Utilization is 0.996703 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
Running service 3 in time period 4
FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): LCM of services is 3
Utilization is 1.000000 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 2 in time period 3
FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7; T4=13; T=D): LCM of services is 4
Utilization is 0.996703 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
Running service 3 in time period 4
FEASIBLE
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$
```

Figure 15: Functionality Script run on Jeston TK1 with LLF Policy

Therefore, all of the examples are feasible when evaluated on either EDF or LLF scheduling policies and this makes sense because note that the way rate monotonic policy is laid out, the task with the shortest period gets the highest priority but in cases of Examples 6 and 8, the tasks with the smallest periods often cause an interruption in the ongoing longer processes. However, using Earliest deadline first (a dynamic priority policy) and Least laxity first (also a dynamic priority policy which considers the remaining time), the smaller tasks are put on a lower priority if an upcoming deadline is arriving for a task with a longer period.

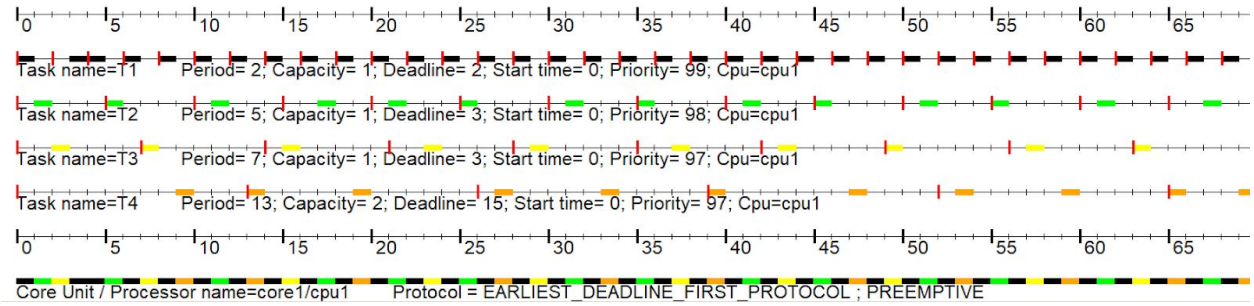
When this is conformed with Cheddar, we get the following results:



```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  9
- Number of preemptions :  2

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 4/worst
   T3 => 10/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 10 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 1.00000 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 :
   T1 => 2
   T2 => 5
   T3 => 10
- All task deadlines will be met : the task set is schedulable.
```

Figure 16: Cheddar Output for Ex-5 using EDF Policy

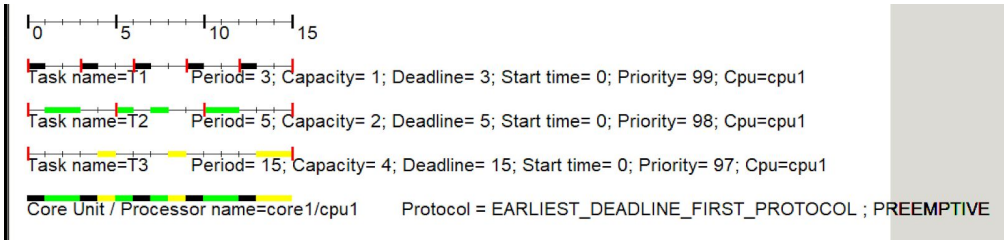
For Ex-6



```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  891
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 2/worst
   T2 => 3/worst
   T3 => 3/worst
   T4 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.30000 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with EDF, the task set is not schedulable because the processor utilization factor 1.30000 is more 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 :
   T1 => 2
   T2 => 3
   T3 => 3
   T4 => 15
- All task deadlines will be met : the task set is schedulable.
```

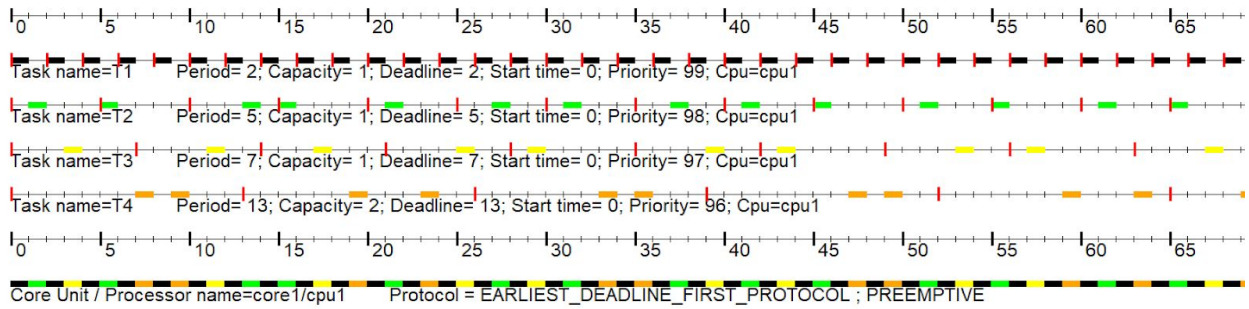
Figure 17: Cheddar Output for Ex-6 using EDF Policy

For Ex-7




```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  11
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 15 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 1.00000 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 :
   T1 => 3
   T2 => 5
   T3 => 15
- All task deadlines will be met : the task set is schedulable.
```

Figure 18: Cheddar Output for Ex-7 using EDF Policy



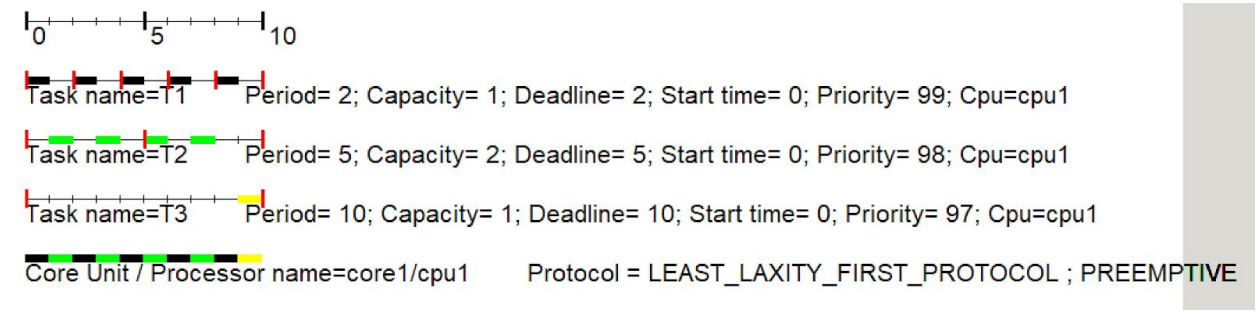

```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 4/worst
   T3 => 6/worst
   T4 => 12/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 0.99670 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 :
   T1 => 1
   T2 => 4
   T3 => 6
   T4 => 12
- All task deadlines will be met : the task set is schedulable.
```

Figure 19: Cheddar Output for Ex-8 using EDF Policy

From the cheddar results above, note that they actually do agree with the conclusions from our script in that is shows that all of the different examples are feasible to be scheduled using Earliest Deadline First (EDF) policy.

Finally, showing LLF policy using Cheddar, we get the following output:

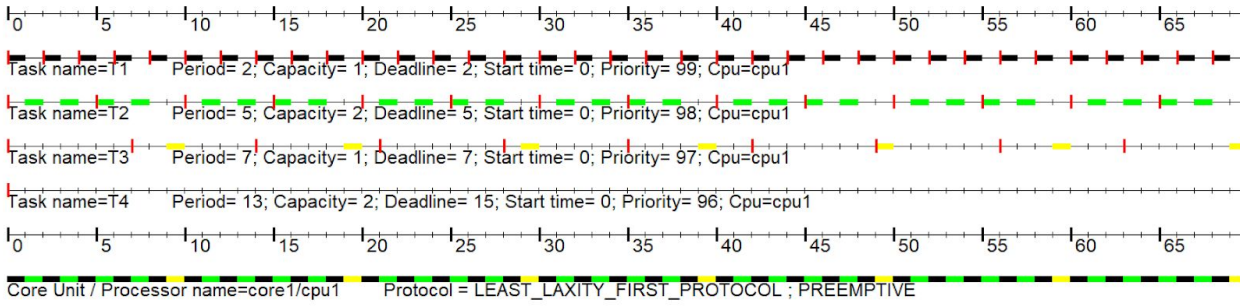
For Ex-5




```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  9
- Number of preemptions :  2

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 4/worst
   T3 => 10/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.



Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 10 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with LLF, the task set is schedulable because the processor utilization factor 1.00000 is equal or less than 1.00000 (see [7]).

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

- Worst Case task response time :
   T1 => 2
   T2 => 5
   T3 => 10
- All task deadlines will be met : the task set is schedulable.
```

Figure 20: Cheddar Output for Ex-5 using LLF Policy

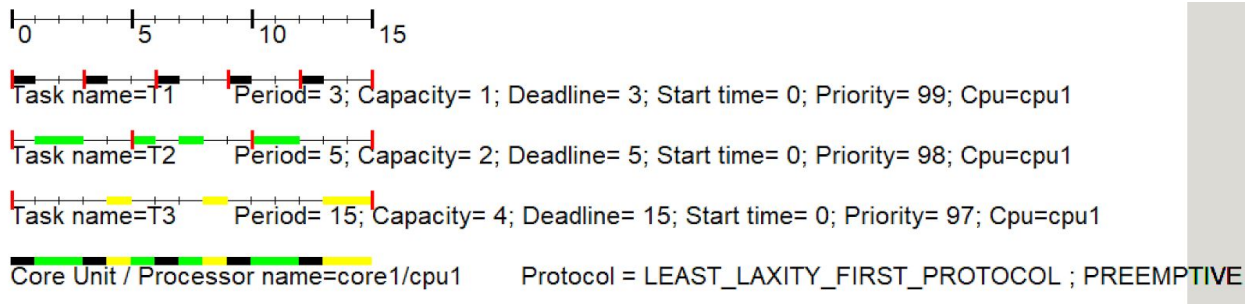
For Ex-6:



```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  909
- Number of preemptions :  182

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 4/worst
   T3 => 280/worst ,  missed its deadline (absolute deadline =  7 ; completion time =  10),  missed its deadline (absolute deadline =  14 ; completion time =  20),  missed its deadline (absolute deadline =  21 ; completion time =  30),  missed its
   T4 => 0/worst ,  response time not computed since the task did not run all its capacity
- One or several tasks did not complete their execution.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.17619 (see [1], page 6).
- Processor utilization factor with period is 1.19670 (see [1], page 6).
- In the preemptive case, with LLF, the task set is not schedulable because the processor utilization factor 1.17619 is more than 1.00000 (see [7]).

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

- Processor utilization exceeded : can not compute worst case response time with this task set.
```

Figure 21: Cheddar Output for Ex-6 using LLF Policy

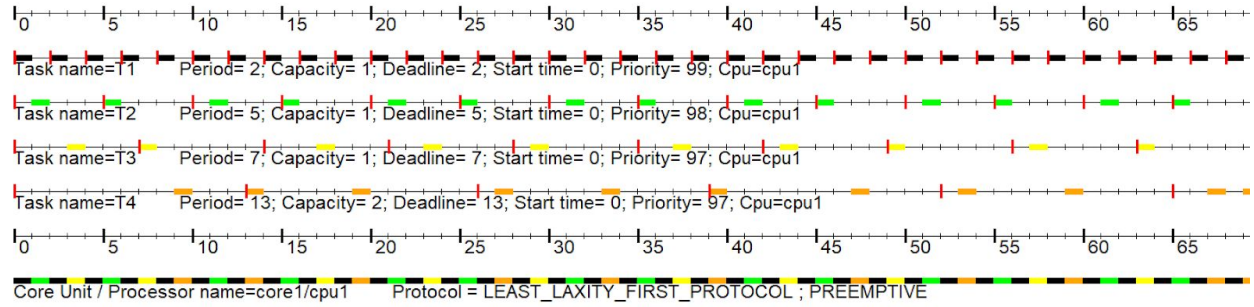
For Ex-7




```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  11
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 15 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with LLF, the task set is schedulable because the processor utilization factor 1.00000 is equal or less than 1.00000 (see [7]).

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

- Worst Case task response time :
   T1 => 3
   T2 => 5
   T3 => 15
- All task deadlines will be met : the task set is schedulable.
```

Figure 22: Cheddar Output for Ex-7 using LLF Policy

Finally for Example 8, we have:




```
Scheduling simulation, Processor cpu1 :
- Number of context switches : 904
- Number of preemptions : 70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 4/worst
   T4 => 16/worst , missed its deadline (absolute deadline = 13 ; completion time = 14), missed its deadline (absolute deadline = 26 ; completion time = 28), missed its deadline (absolute deadline = 39 ; completion time = 40), missed its
- Some task deadlines will be missed : the task set is not schedulable.


Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with LLF, the task set is schedulable because the processor utilization factor 0.99670 is equal or less than 1.00000 (see [7]).

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

- Worst Case task response time :
   T1 => 1
   T2 => 4
   T3 => 6
   T4 => 12
- All task deadlines will be met : the task set is schedulable.
```

Figure 23: Cheddar Output for Ex-8 using LLF Policy

Consequently, note that when comparing out the results in Figures 20-23, we see that our results actually don't overlap with the results obtained from the Jetson Tk1 when running the modified script. Specifically, just like Rate Monotonic Policy, Examples 6 and 8 were shown to not be feasible from Cheddar but that's because it uses LUB CPU utilization thresholds where those thresholds are not factored into the decision making process for the Jetson TK1 script. As such, the EDF policy has the best results because all of the examples were able to be scheduled whereas RM and LLF policy had similar results.

Finally, comparing the EDF implementation for Examples 0-4, we got:

```
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ./feasibility_test_EDF
******** EDF Test Feasibility Example
Ex-0LCM of services is 3
Utilization is 0.733333 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
Ex-1LCM of services is 3
Utilization is 0.985714 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
Ex-2LCM of services is 4
Utilization is 0.996703 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
Running service 3 in time period 4
FEASIBLE
Ex-3LCM of services is 3
Utilization is 0.933333 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 2 in time period 3
FEASIBLE
Ex-4LCM of services is 3
Utilization is 1.000000 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for EDF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$
```

And using the LLF policy, we get:

```
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$ ./feasibility_test_LLF
******** LLF Test Feasibility Example
Ex-0LCM of services is 3
Utilization is 0.733333 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
Ex-1LCM of services is 3
Utilization is 0.985714 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
Ex-2LCM of services is 4
Utilization is 0.996703 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
Running service 3 in time period 4
FEASIBLE
Ex-3LCM of services is 3
Utilization is 0.933333 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 2 in time period 3
FEASIBLE
Ex-4LCM of services is 3
Utilization is 1.000000 schedule feasible accordinng to Utilization based test (Only Sufficient)
Now Checking Completetion Test over LCM for LLF
Running service 1 in time period 1
Running service 2 in time period 2
Running service 1 in time period 3
FEASIBLE
ubuntu@tegra-ubuntu:~/Desktop/ECEN_5623/Lab_2$
```

Therefore, from both of the screenshots above, it’s decipherable that the services are able to be schedulued using EDF and LLF policies and this aligns up with the results that were discussed in class. However, note that this doesn’t not align up with the results from RM policy for the same reasons as discussed above in which RM is a fixed priority scheduling scheme and both EDF and LLF are dynamic priorities.

Prerit Oberai
Nikhil Divekar

ECEN 5623

Exercise 2

February 21, 2019

Problem 5

[30 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 the text. 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 the text.

Solution:

Three constraints that are made in Liu and Layland paper are as given below:

1. Deadline is considered equal to the period for the derivation. In general, deadline and period are two different things. Deadline is defined as time by which the real-time system has to produce output else, the utility of system either decrease or becomes zero. While period is simply the execution time of the task and the time after which process repeats itself in case of periodic services. This constraint has been made for simpler derivation.

2. The RM Policy derivation does not take into consideration the time for context switching, overheads and latency. Context switching is defined as the time to store the register values and return address on the stack when higher priority task preempts lower priority task. This causes overhead. While latency is similar to context switching, but more often used in terms of interrupt and is defined as the time between interrupt occurs and source of the interrupt is cleared. Now these are variables and their values cannot be determined exactly. This is taken care of by Least Upper Bound of RM Policy which is large time buffer which allows all these variables to be taken into consideration.

3. The worst-case execution time is considered for checking the feasibility of the system. If all the services. Worst-case execution time is defined as time taken if all the services make request at a time and may also be defined as the maximum length of time the task could take for its execution. So, the idea here is, if system is rendered feasible for the WCET, then it would be feasible for all other execution time.

4. Another constraint, I think is the usage is fixed priority and run-to-completion algorithm.

Assumptions made in Liu and Layland paper:

1. The request for all tasks for which hard deadlines exit are periodic, with constant interval between requests.

2. Deadlines consist of run-ability constraints only - i.e each task must be completed before the next request for it occurs.

---

3. The tasks are independent in that request for a certain task do not depend on the initiation or the completion of requests for other tasks.

4. 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.

5. Any non-periodic tasks in the system are special; they are initialization or failure-recovery routines; they displace periodic tasks while they themselves are being run and do not themselves have hard, critical deadlines.

Key concepts:

1. Point 1:

In theorem 3, it is told what would be the least upper bound for fixed priority assignment.

Two task r1 and r2 are considered with period T1 and T2 and execution time being C1 and C2.

In this theorem, C2 is adjusted in such a way to fully utilize the processor time.

Case 1 suggests, The run time for C1 is short enough that requests for r1 within critical time cone of T2 are completed before r2 request.

$$C_1 \leq T_2 - T_1 \lfloor T_2/T_1 \rfloor .$$

So, this case was little tricky to understand at first, However on reading book chapter 3, it made more sense. It has been explained efficiently with the diagram below.

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

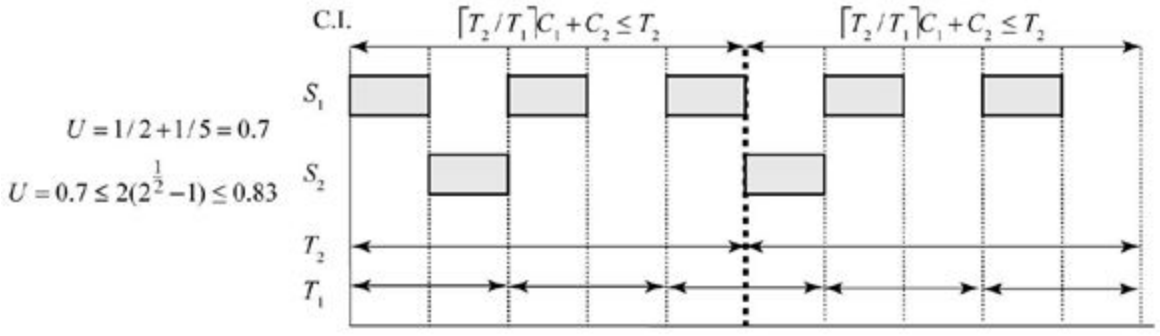


Figure 3.1 Example of Two-Service Feasibility Testing by Examination

The critical instant assumes that in the worst case, all services might be requested at the same time. To understand, we must draw the diagram over the time which is LCM of all periods.

Two services are considered S1 and S2. Their periods and run time being T1 and T2 and C1 and C2 respectively. T2 is greater than T1. In this case, T1 = 2, T2 = 5, C1 = 1, C2 =1. S1 has higher priority than S2. The above example has same parameters as in theorem 3.

From the above diagram, it was clear that we can find the utility of system by (C1/T1 + C2/T2). So, we get C2/T2 = -C1/T1
Hence for C2 to be very small enough, we have

$$C_2 = T_2 - C_1 \lceil T_2 / T_1 \rceil$$

In this case, T2 = 5 and T1 = 2, Hence floor of (T1/T2) = 2. Thus, we understand C2 = T2 - 2*T1

Thus, book provides more information with the diagram and example and helped me understand the assumption behind the above-mentioned equation from theorem 3 case1.

2. Point 2

In the same theorem, the paper gives us two cases.

Case 1 is given below:

Case 1. The run-time C_1 is short enough that all requests for τ_1 within the critical time zone of T_2 are completed before the second τ_2 request. That is,

$$C_1 \leq T_2 - T_1 \lfloor T_2/T_1 \rfloor .$$

Thus, the largest possible value of C_2 is

$$C_2 = T_2 - C_1 \lceil T_2/T_1 \rceil .$$

The corresponding processor utilization factor is

$$U = 1 + C_1[(1/T_1) - (1/T_2) \lceil T_2/T_1 \rceil].$$

In this case, the processor utilization factor U is monotonically decreasing in C_1 .

Case 2 is as given below:

Case 2. The execution of the $\lceil T_2/T_1 \rceil$th request for t_1 overlaps the second request for τ_2 . In this case

$$C_1 \geq T_2 - T_1 \lfloor T_2/T_1 \rfloor .$$

It follows that the largest possible value of C_2 is

$$C_2 = -C_1 \lfloor T_2/T_1 \rfloor + T_1 \lfloor T_2/T_1 \rfloor$$

and the corresponding utilization factor is

$$U = (T_1/T_2) \lfloor T_2/T_1 \rfloor + C_1[(1/T_1) - (1/T_2) \lfloor T_2/T_1 \rfloor].$$

In this case, U is monotonically increasing in C_1 .

In first case, the processor utilization factor is decreasing in C1 while in case 2 it is increasing.

Chapter 3 of book gives more insight about this with detailed example and especially the graphs below.

Case 1 graph:

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

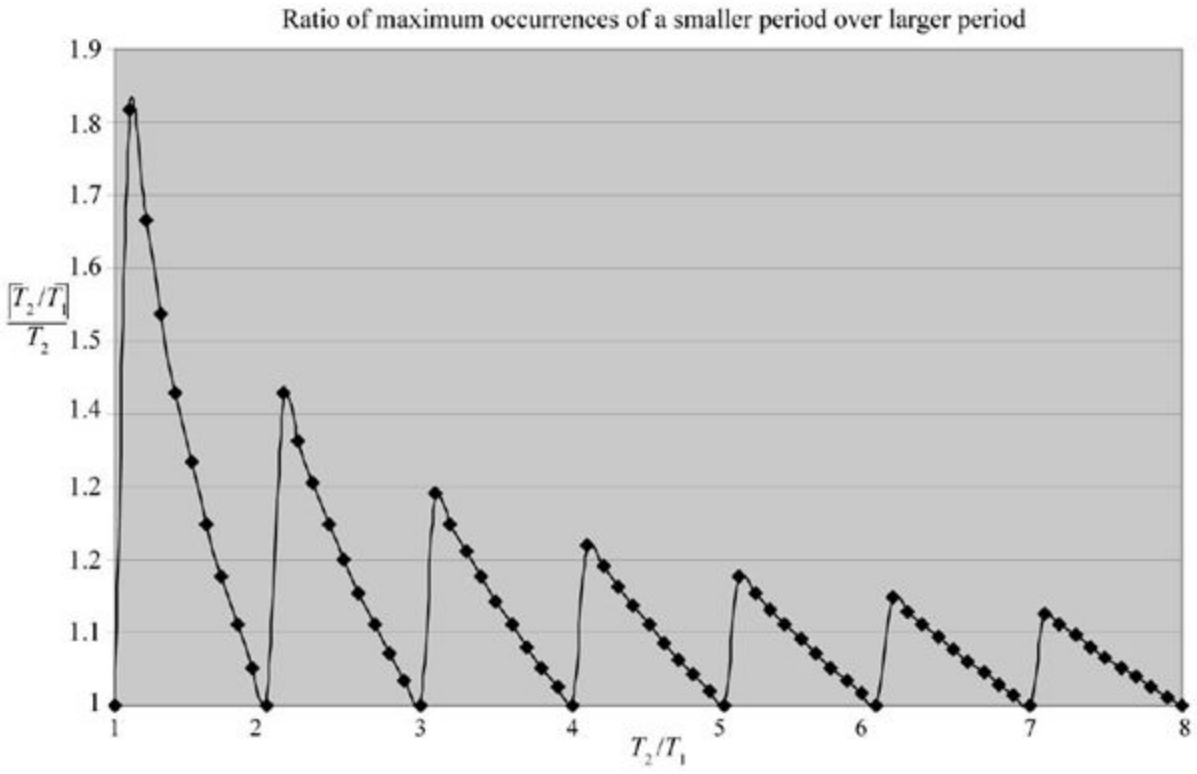


Figure 3.6 Case 1 Relationship of T_2 and T_1

Case 2 graph:

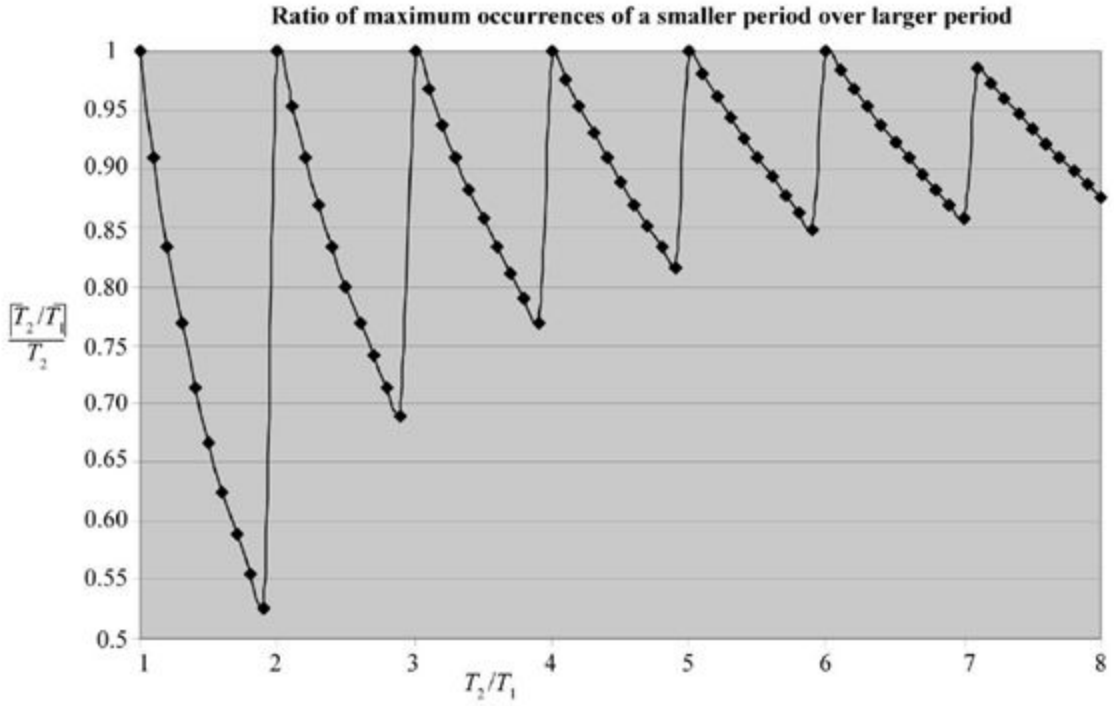


Figure 3.8 Case 2 Relationship of T_2 and T_1

3. Point 3:

The minimum of U clearly occurs at the boundary between these two cases. That is, for

$$C_1 = T_2 - T_1 \lfloor T_2/T_1 \rfloor$$

we have

$$U = 1 - (T_1/T_2)[\lceil T_2/T_1 \rceil - (T_2/T_1)][(T_2/T_1) - \lfloor T_2/T_1 \rfloor]. \qquad (3)$$

For notational convenience,[3] let $I = \lfloor T_2/T_1 \rfloor$ and $f = \{T_2/T_1\}$. Equation (3) can be written as

$$U = 1 - f(1 - f)/(I + f).$$

Since U is monotonic increasing with I, minimum U occurs at the smallest possible value of I, namely, $I = 1$. Minimizing U over f, we determine that at $f = 2^{\frac{1}{2}} - 1$, U attains its minimum value which is

$$U = 2(2^{\frac{1}{2}} - 1) \simeq 0.83.$$

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

In case 2, the paper talks about finding approximate utility but I did not understand the theorem much when read just from paper. There are bunch of assumptions made here with no display of derivation steps.

Below is the derivation I found in the book which helped me understand how author reached 0.83 value.

Now, let whole integer number of interferences of S_1 to S_2 over T_2 be $I = \lfloor T_2 / T_1 \rfloor$ and the fractional interference be $f = (T_2 / T_1) - \lfloor T_2 / T_1 \rfloor$. From this, we can derive a simple expression for utility:

$$U = 1 - \left(\frac{f(1-f)}{(T_2 / T_1)} \right) \qquad (3.13)$$

The derivation for Equation 3.11 is based upon substitution of I and f into Equation 3.12 as follows:

$$U = 1 - (T_1 / T_2)\left[\lceil T_2 / T_1 \rceil - (T_2 / T_1)\right]\left[(T_2 / T_1) - \lfloor T_2 / T_1 \rfloor\right]$$

$$U = 1 - (T_1 / T_2)\left[1 + \lfloor T_2 / T_1 \rfloor - (T_2 / T_1)\right]\left[(T_2 / T_1) - \lfloor T_2 / T_1 \rfloor\right] \text{ based on ceiling(N.d) = 1 + floor(N.d)}$$

$$U = 1 - (T_1 / T_2)\left[1 - \left((T_2 / T_1) - \lfloor T_2 / T_1 \rfloor\right)\right]\left[(T_2 / T_1) - \lfloor T_2 / T_1 \rfloor\right]$$

$$U = 1 - (T_1 / T_2)(1-f)(f)$$

$$U = 1 - \left(\frac{f(1-f)}{(T_2 / T_1)} \right)$$

By adding and subtracting the same denominator term to Equation 3.13, we can get:

$$U = 1 - \left(\frac{f(1-f)}{\lceil T_2/T_1 \rceil + (T_2/T_1) - \lceil T_2/T_1 \rceil} \right)$$

$$U = 1 - \left(\frac{f(1-f)}{(I+f)} \right)$$

The smallest I possible is 1, and the LUB for U occurs when I is minimized, so we substitute 1 for I to get:

$$U = 1 - \left(\frac{(f - f^2)}{(1+f)} \right)$$

Now taking the derivative of U w.r.t. f, and solving for the extreme, we get:

$$\partial U / \partial f = \frac{(1+f)(1-2f) - (f - f^2)(1)}{(1+f)^2} = 0$$

Solving for f, we get:

$$f = \left(2^{1/2} - 1\right)$$

And, plugging f back into U, we get:

$$U = 2\left(2^{1/2} - 1\right)$$

The RM LUB of $m(2^{1/m} - 1)$ is $2(2^{1/2} - 1)$ for m=2, which is true for the two-service case—QED

There are bunch of steps like adding and subtracting the same denominator term or taking derivative which is not present in paper, and hence was tricky to understand while book provides more detailed derivation.

Prerit Oberai
Nikhil Divekar

Exercise 2

ECEN 5623
February 21, 2019

References

1. Professor Scherr RTES Powerpoint Notes on Rate Monotonic Policy, EDF and LLC policy
2. Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment - C. L. LIU & JAMES W. LAYLAND
3. Provided examples from D2L for Scheduling examples 5-8 as well as reference material for examples 1-4
4. Lehoczky-Sha-Ding Paper on Canvas
5. Shuttle Paper from Canvas
6. https://en.wikipedia.org/wiki/Rate-monotonic_scheduling
7. Real time embedded systems and components by Prof. Sam Siewert and John Pratt
8. https://en.wikipedia.org/wiki/Cyclic_executive
9. https://stackoverflow.com/questions/115306/does-it-make-sense-to-mix-an-rtos-and-cyclic-executive
10. https://pubweb.eng.utah.edu/~cs5785/slides-f10/22-1up.pdf