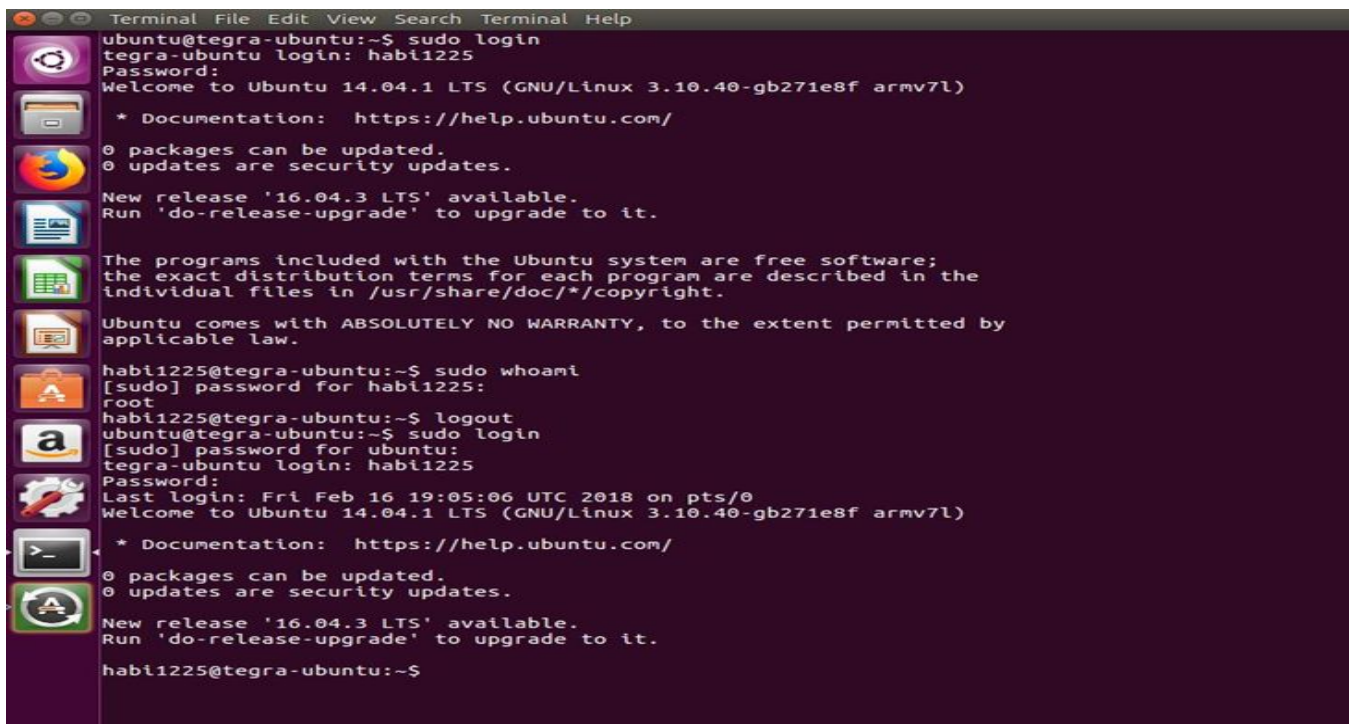


ECEN 5623, Real-Time Embedded Systems:Date 02/23/2018**Exercise #2 – Service Scheduling Feasibility****Question 1**

Make yourself an account on your Dev Kit. To do this, use the reset button if the system is locked, use your password to login, and then use “sudo adduser”, enter a 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 Linux 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.

Solution:**WHOAMI Command Result**

```
Terminal File Edit View Search Terminal Help
ubuntu@tegra-ubuntu:~$ sudo login
tegra-ubuntu login: habi1225
Password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.10.40-gb271e8f armv7l)

 * Documentation:  https://help.ubuntu.com/

0 packages can be updated.
0 updates are security updates.

New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

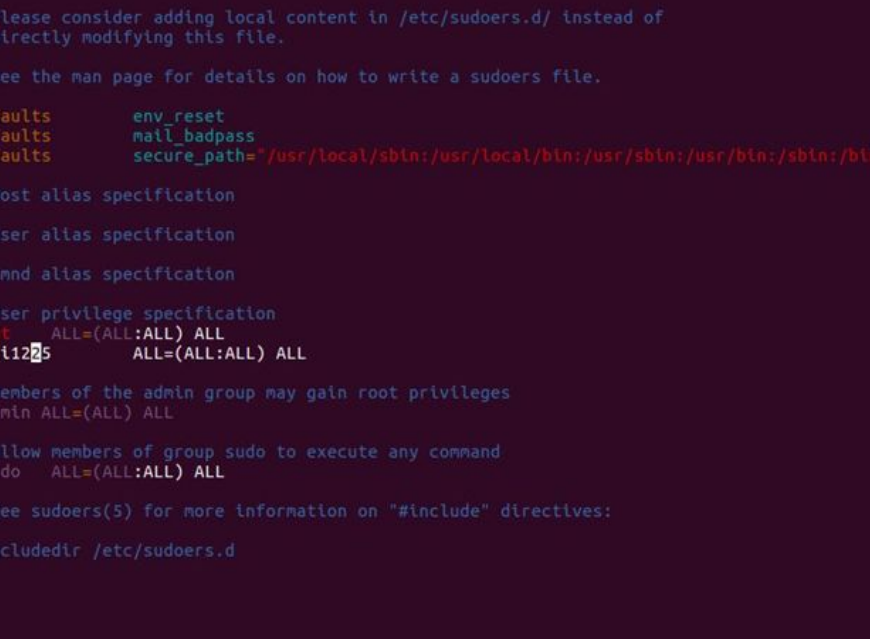
habi1225@tegra-ubuntu:~$ sudo whoami
[sudo] password for habi1225:
root
habi1225@tegra-ubuntu:~$ logout
ubuntu@tegra-ubuntu:~$ sudo login
[sudo] password for ubuntu:
tegra-ubuntu login: habi1225
Password:
Last login: Fri Feb 16 19:05:06 UTC 2018 on pts/0
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.10.40-gb271e8f armv7l)

 * Documentation:  https://help.ubuntu.com/

0 packages can be updated.
0 updates are security updates.

New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

habi1225@tegra-ubuntu:~$
```



```

ubuntu@tegra-ubuntu: ~
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults                env_reset
Defaults                mail_badpass
Defaults                secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
# Host alias specification
#
# User alias specification
#
# Cmnd alias specification
#
# User privilege specification
root    ALL=(ALL:ALL) ALL
habi1225    ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "#include" directives:
#includedir /etc/sudoers.d

```

Question 2

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

Overall understanding of paper and key point articulation[1]

The paper discusses a system called PASS (Primary Avionics Software System) which incorporates a well-structured architecture that has been influenced by redundancy and timing management, data processing Data Processing System, general purpose computer design GPC, memory and CPU constraints, scheduling and performance management. The architecture for PASS system involves cyclic executive approach and executed tasks based on priorities.

There are two types of memories associated with the architecture – main memory and mass memory. The vital applications are kept within the mass memory and as and when needed, the relevant applications are loaded onto the main memory from the mass memory, or when the crew enters so from the keyboard.

The PASS can be divided functionally into 8 main phases called as operational sequence (OPS). Each OPS consists of a major mode, which has a primary function, specialist function, which in turn has the background function and display function, all used for monitoring.

The PASS software architecture is designed to synchronously dispatch each functionality at a specific time in a cycle. This approach has a benefit of being repeatable but with limited flexibility for changes. But with the increase in the number of features, non-synchronous approach was finally used.

The applications were sequenced using the following –

1. FCOS: Used to manage the external interfaces and internal resources, along with the code transfer between mass and main memory.
2. Application Software: Used for communication with other computers and synchronization between them.
3. System Control: Used for loading and initialization of data.
4. User Interface: Used for communication between user and the application software implementing interfaces such as keyboard and CRT.

The paper also describes Application Software which comprises of three applications: SM, VCO and GN&C.

1. System Management (SM): This application monitors the performance and alerts the crew if something goes wrong.
2. Vehicle Checkout (VCO): This application ground testing, integration and certification.
3. Ground Navigation and Control (GN&C): This application determines attributes such as velocity, altitude and position of a vehicle. It also manages sensor redundancy and provide commands to any mission. The application runs in a cyclic manner called executive, with tight timings and phasing requirements. A table is selected based upon the current flight stage of the shuttle, which is then used by the executive design for dispatching the services.

Frequency executive architecture-

A frequency executive is an alternative to a real-time operating system. A single task is typically realized as an infinite loop, cycling through a repeating sequence of activities, at a set frequency. Cyclic scheduling is static and is computed offline and stored in a table. Time slots that are not used by periodic tasks can run non-periodic work.

Advantages of frequency executive architecture over real-time threading-[2]

1. No Threads: No involvement of threads leads to no complication of creation, locks, deadlocks, race conditions and semaphores etc.
2. Predictability: The entire system becomes predictable owing to cyclic executive as it provides information regarding when the execution of any service will end.
3. Less Context Switches: If the requirement of context switches is less, it saves time and resources. This applies to a system that does not involve any preemption.
4. Repeatability: Cyclic executive is used if it is predetermined that a set of activities are to be repeated at a fixed frequency. This works well for real-time operating systems as they perform repeated computations that have characteristic rates and response-time requirements.

Disadvantages of frequency executive architecture over real-time threading-

1. Asynchronous services: Asynchronous services are not entertained in cyclic executive, which might be very important for many real-time systems.
2. A lengthy task must be divided into various subtasks to avoid overrun that leads to overhead.
3. CPU cycle wastage: As an executive system makes a function call only at certain time events, there might arise a situation where CPU cycles are wasted while waiting for a timeout to occur.
4. Less flexibility: Simple changes like adding a new task or extending the existing task's period also require a lot of changes to be made in the existing algorithm to ensure that it does not lead to any overrun which ultimately leads to system failure.

Question 3

Read the paper “Building Safety-Critical Real-Time Systems with Reuseable Cyclic Executives”, available from [http://dx.doi.org/10.1016/S0967-0661\(97\)00088-9](http://dx.doi.org/10.1016/S0967-0661(97)00088-9). 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

Overall understanding of paper and key point articulation[3]:

“Building Safety- Critical real-time systems with reusable cyclic executives” paper basically explains that Synchronous architectures based on cyclic executive with static schedule is preferred for building safety critical applications. Asynchronous architecture is difficult to implement because of arbitrary interleaving which is not the case for synchronous architectures as they are fully predictable and can be completely tested hence operations is easier for synchronous architecture and are mainly preferred for safety critical applications. But there are some noticeable issues as they deal with a low level abstraction and it causes difficulty to maintain the system and make it costly to develop. Then there are techniques discussed to achieve the goal of enhancing flexibility making it configurable and reusable, one of the technique is discussed in detail namely ADA 95 Programming language. The cyclic executive with static schedule is stated as the core of the proposed synchronous architecture in the paper. Making the cyclic executive configurable and reusable is one of the main goals. The parameters which need to be configurable are namely

- Duration of minor and major cycle periods
- Number of minor cycles per major cycle
- Maximum number of frames per minor cycle
- Process (reference to procedure) that has to be executed in each frame.

Since the main focus of the paper is to build a Safety Critical real-time system it is important to include some fault tolerance in the system. Further it is mentioned that a dynamic recovery policy will be adopted, based on the concept of recovery groups. Another important factor to consider is failure in real-time system due to deadline overrun.

Basic Requirements for Generic Cyclic Executives

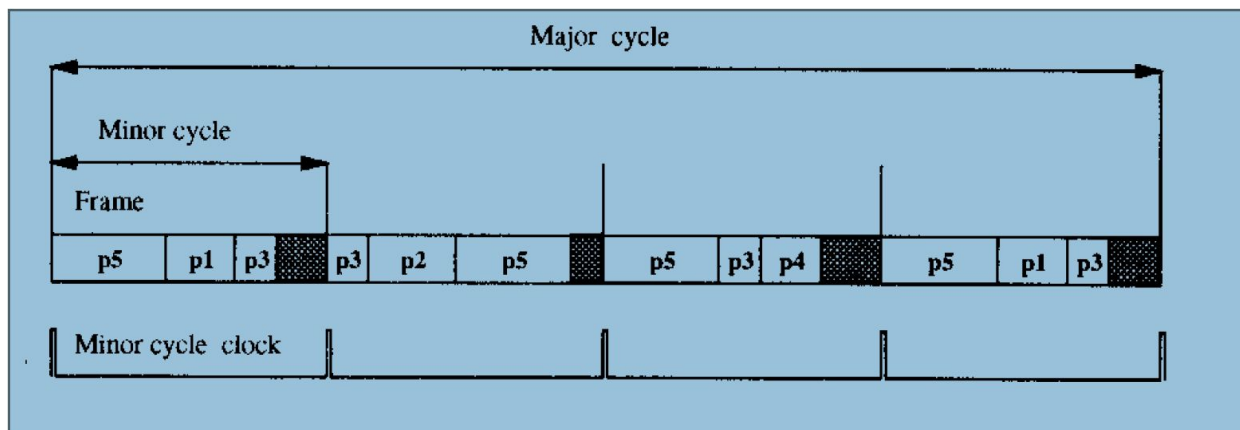
The Main purpose is to make the executive into a generic, reusable component which can be tailored to a wide variety of real-time applications. The Cyclic Schedule is basically a table which comprises of the full execution pattern for a major cycle. A major cycle is made up of a definite number of minor cycles which are executed periodically. The points which need to be configurable are discussed in the brief explanation above. To make the executive platform

independent(Portable) the real-time clock and other timer softwares used incorporated as different modules or components altogether.

Cyclic Scheduler Structure

Cyclic scheduler is the main component for this kind of synchronous architecture, it is simply an iterative process that explicitly multiplexes the periodic processes into one processor. First there is a major schedule which executes repeatedly until the mode of operation is changed. The major schedule consists of varying number of minor schedules. Minor schedules describes the process sequence executed during each of the minor cycle. The end of minor cycle is indicated by tick of the minor cycle clock. Before this time the process sequence corresponding to this minor cycle should finish execution or the minor_cycle_overrun occurs.

Minor schedule is further divided into one or more frames. Only one process per frame is executed in the minor schedule and if the process doesn't finish it's execution before the frame has end then frame_overrun occurs



Cyclic Scheduler Structure

Scheduler Exceptions

The most logical way to deal with errors or unexpected or anomalous conditions is by using exceptions. An exception can be simply defined as an event which causes temporary suspension of the normal execution of the program. Exceptions can be predefined or can be user defined. Predefined exceptions are those that are implicitly activated by the run-time system when the related event occurs. The user defined exceptions are those which can be explicitly activated by using the statement raise. Exceptions are one of the most convenient way to handle the overrun errors. The use of exceptions makes aids to write the cyclic scheduler's main loop structure in a simple and clean way.

General Architecture For Synchronous real-time systems

A generic architecture as defined in the paper is a “template for building a system or subsystem from component blocks, interconnected by a set of standard elements”.

There are Three Major components for generic architecture for synchronous real-time systems

- Structures: it is basically a module where all the basic data structures which are Architecture independent that is portable are stored. They are basically independent of the development system and the execution architecture.
- Executive: It is defined as a subsystem which consists of the a few simple components which isolate the hardware or provide compiler and run-time dependencies.
- Shells: These are templates used for building the application process and defining the system structure

The basic idea is all the components should be reusable, hence they are generic which makes them configurable. Use of global components makes it easier to build a system from its basic blocks.

Implementation in ADA 95

ADA 95 can be used to implement the generic architecture in a simple way. Features such as dynamic storage allocation, recursive procedure calls, and unlimited loops or any other tasking features are not used since they are strictly banned while building a safety critical application.

A preferable and powerful way of handling error-detection and recovery mechanism is by using exceptions.

Characteristics of ADA 95 used for building reusable code are:

- Enhanced generic parameters which also including the packages and access types
- Enhanced access types which includes access to static objects and procedures
- Hierarchical private and public packages
- Lastly, standard interfaces to procedures which are written in other languages

Conclusion

Finally it is concluded that new features in ADA (different packages mentioned above) makes it easy to develop reusable code. Moreover, the concept of recovery groups is reused in an ADA cyclic executive for space real-time applications. The architecture is also can also be implemented in C++ programming language but with meticulous care as the wrong use of pointers can create run-time errors.

Comparison to Linux and RTOS approaches

Firstly, cyclic executive can be considered as an alternative to a real-time operating system. There are many pros and cons in a cyclic executives when compared to the Linux and RTOS.

There is generally only a single task present in cyclic executive. The sole task is typically an infinite loop in main().

Distinguishing points

Points	Cyclic Executive	Linux or RTOS
Approach	It has a main loop approach	Create task and threads to execute their services
Architecture	It has run-time interface, scheduler and application on the top layer.	The scheduler is present in the Kernel which controls the scheduling process and priorities of threads and services running on the kernel
Processors	Normally runs on a single processor	Generally has AMP or SMP Architecture
Development, expandability and maintainability	It is difficult in cyclic executive as it has low abstraction level as mentioned in the "Building Safety- Critical real-time systems with reusable cyclic executives" paper [4]	It has high abstraction level that is it contains modules which can be easily expanded and can be ported to other system easily.
Overhead and context switches	Not present in cyclic executive hence there is extremely low jitter	Present in real-time operating systems and Linux which produces jitter.

Question 4

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:

We checked the feasibility of all examples given, for RM, EDF and LLF scheduling, by comparing the results we got by running the code, by cheddar analysis and as documented on the excel.

Earliest deadline first: A dynamic-priority scheme for scheduling where services are assigned priority dynamically every time the ready queue is updated, with highest priority given to the service with the earliest impending deadline; the scheme requires not only dynamic-priority but also preemption to work.[5]

Least laxity first (LLF): A dynamic-priority policy where services on the ready queue are assigned higher priority if their laxity is the least (where laxity is the time difference between their deadline and remaining computation time); this requires the scheduler to know all outstanding service request times, their deadlines, the current time, and remaining computation time for all services, and to re-assign priorities to all services on every preemption. Estimating remaining computation time for each service can be difficult and typically requires a worst-case approximation.[5]

We observed that EDF and LLF, which are dynamic policies, were able to schedule the example sets(Example 1,2,6,8) which the RM policy couldn't schedule. EDF/LLF policies could schedule those example sets because they assign priorities at run-time and the priorities are recalculated every unit time. In case of RM policy the priorities are static that is they do not change during run time.

We added required examples to the given RM code and observed the output. We also checked the results on cheddar software and compared the two. The codes for EDF and LLF were written by us and submission folder will contain the code files. Below we give results after running cheddar and the code.

Results:

Examples	Feasibility Code Analysis				Cheddar Analysis			
	RM	EDF	LLF	DM	RM	EDF	LLF	DM
Example 0	F	F	F		F	F	F	
Example 1	NF	F	F		NF	F	F	
Example 2	NF	F	F		NF	F	F	
Example 3	F	F	F		F	F	F	
Example 4	F	F	F		F	F	F	
Example 5	F	F	F		F	F	F	
Example 6	NF	F	F	Feasible by Completion time test. Infeasible by Scheduling Point test.	NF	F	F	NF
Example 7	F	F	F		F	F	F	
Example 8	NF	F	F		NF	F	F	
Example 9	F		F		F	F	F	

RM Policy Code Results^[1]

All 10 examples -

```
harsimransingh@harsimransingh-VirtualBox:~/Real_Time_Embedded_Systems/Exercises/Exercise 2/Feasibility$ ./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-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=0.9967 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; 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
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
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
```

DM Policy Code Results

All 10 examples -

```
harsimransingh@harsimransingh-VirtualBox:~/Real_Time_Embedded_Systems/Exercises/Exercise 2/Feasibility$ ./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-6(Deadline Monotonic) U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; D1=2, D2=3, D3=7, D4=15): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=0.9967 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; 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
Ex-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6(Deadline Monotonic) U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; D1=2, D2=3, D3=7, D4=15): 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
Ex-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
```


EDF Policy Code Results[\[1\]](#)

All 10 examples -

```
gcc -O3 -g -O Feasibility_tests_EDF Feasibility_tests.0 EDF_Functions.0 -lpthread -lGL
harsimransingh@harsimransingh-VirtualBox:~/rtes/Exercise2$ ./feasibility_tests_EDF
EDF Policy Feasibility Tests
done
***** Test for Feasibility *****

Schedulability Tests for all Examples
Example-0 F U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D):The Set is FEASIBLE

Example-1 F U=0.99 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D):The Set is FEASIBLE

Example-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D):The set is FEASIBLE

Example-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): The Set is FEASIBLE

Example-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): The Set is FEASIBLE

Example-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): The Set is FEASIBLE

Example-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): The Set is FEASIBLE

Example-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D):The Set is FEASIBLE

Example-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): The Set is FEASIBLE

Example-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D):The Set is FEASIBLE
```

LLF Policy Code Results[\[1\]](#)

All 10 examples -

```
harsimransingh@harsimransingh-VirtualBox:~/rtes/Exercise2/LLF$ ./feasibility_tests_LLFF
LLF Policy Feasibility Tests
done
***** Test for Feasibility *****

Schedulability Tests for all Examples
Example-0 F U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D):The Set is FEASIBLE

Example-1 F U=0.99 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D):The Set is FEASIBLE

Example-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D):The set is FEASIBLE

Example-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): The Set is FEASIBLE

Example-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): The Set is FEASIBLE

Example-5 U=1.00 (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): The Set is FEASIBLE

Example-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): The Set is FEASIBLE

Example-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D):The Set is FEASIBLE

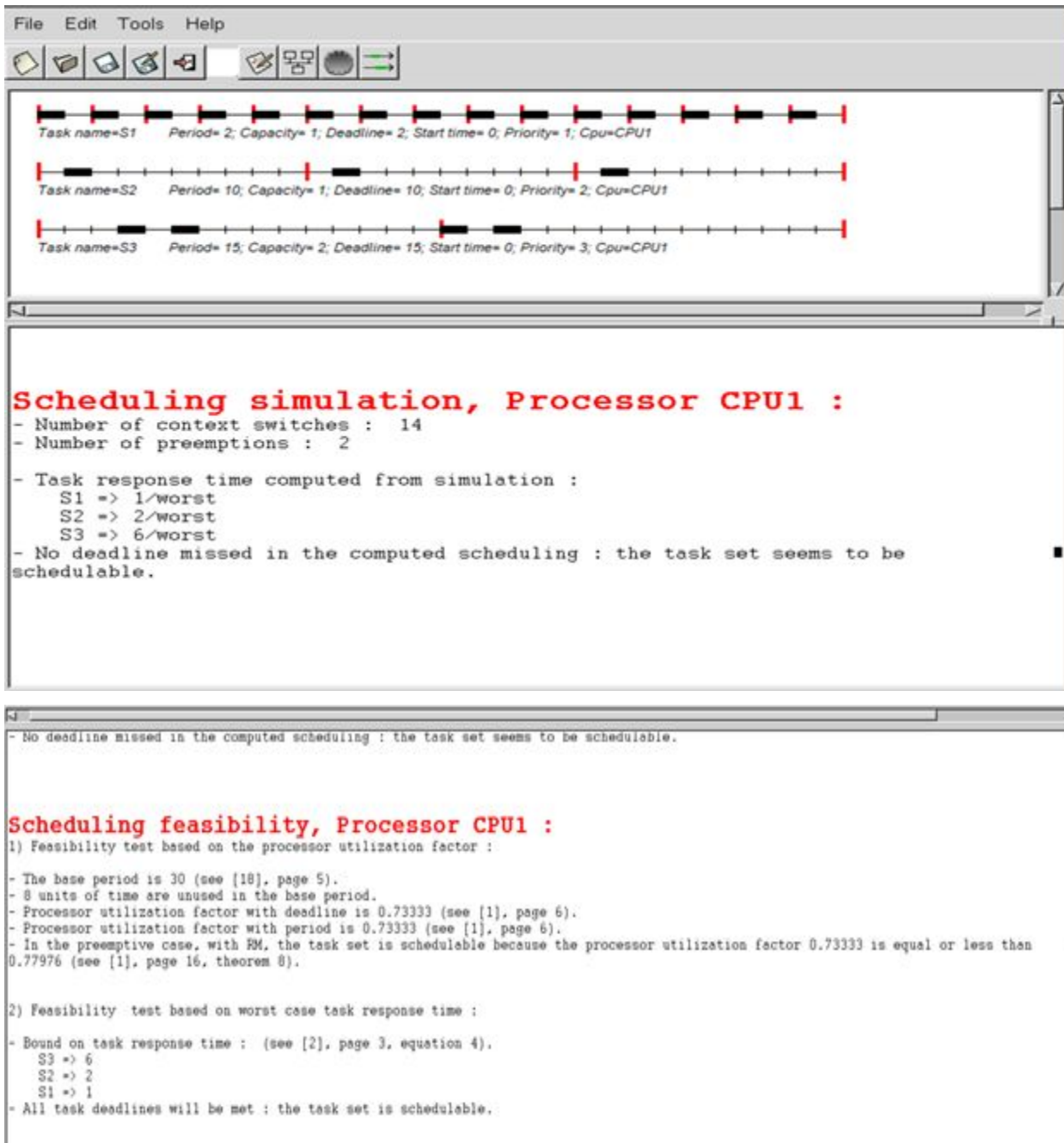
Example-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): The Set is FEASIBLE

Example-9 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D):The Set is FEASIBLE
harsimransingh@harsimransingh-VirtualBox:~/rtes/Exercise2/LLF$
```

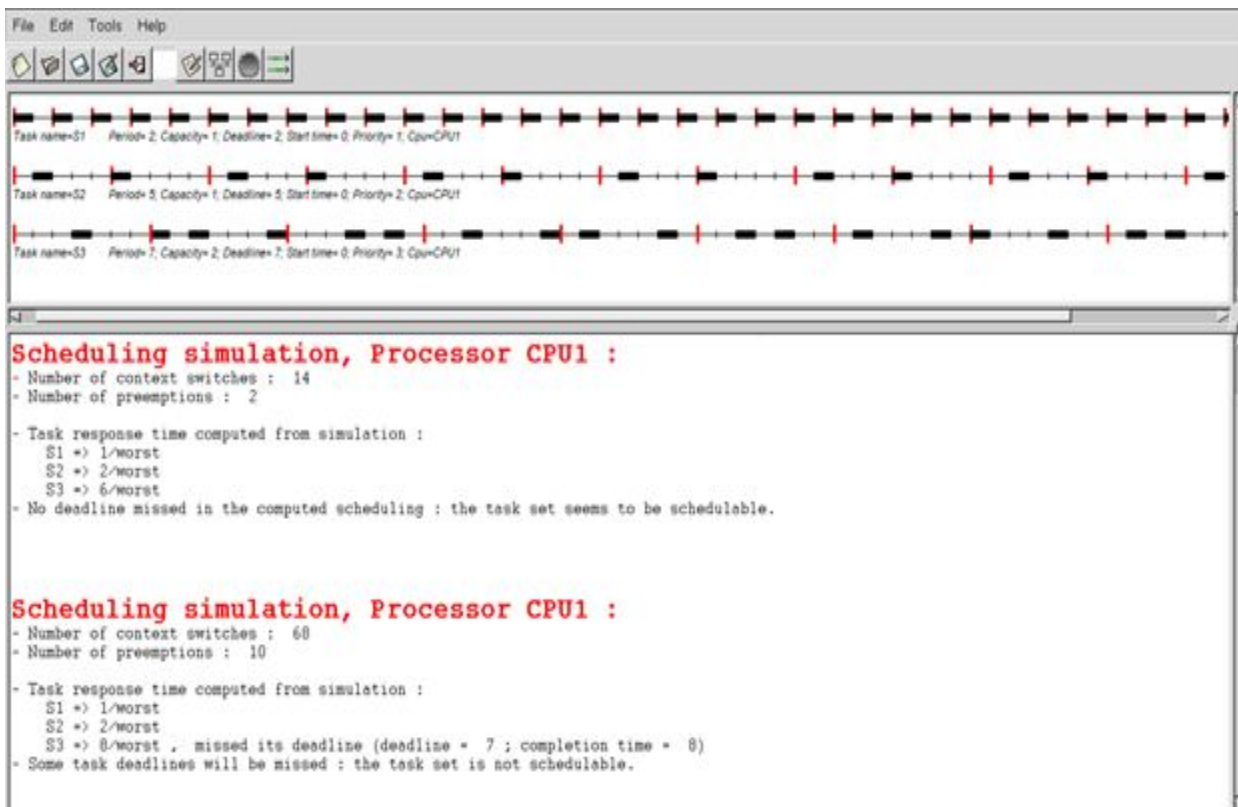
Cheddar Analysis

Rate Monotonic

Example 0



Example 1



Scheduling feasibility, Processor CPU1 :

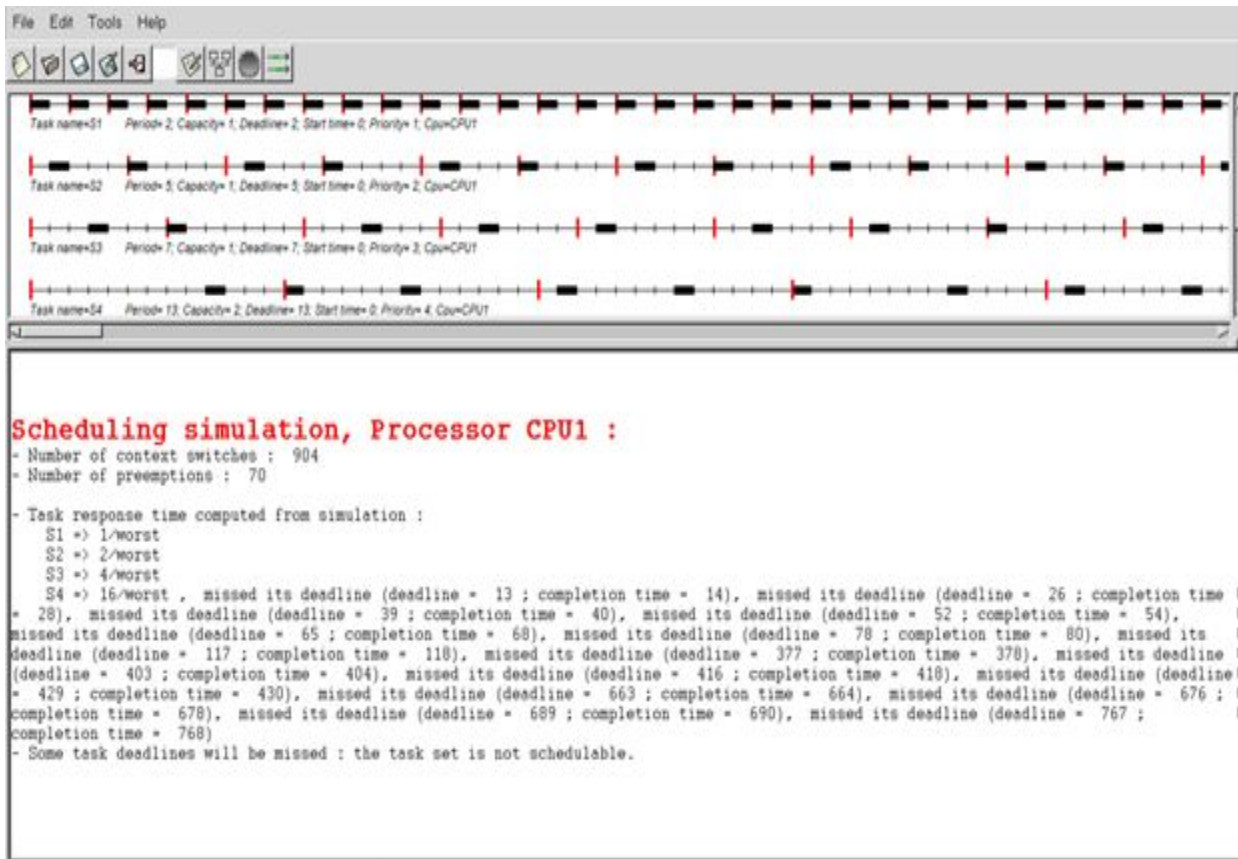
1) Feasibility test based on the processor utilization factor :

- The base period is 70 (see [18], page 5).
- 1 units of time are unused in the base period.
- 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 task response time :

- Bound on 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.

Example 2



Scheduling feasibility, Processor CPU1 :

1) Feasibility test based on the processor utilization factor :

- The base period is 910 (see [18], page 5).
- 3 units of time are unused in the base period.
- 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 task response time :

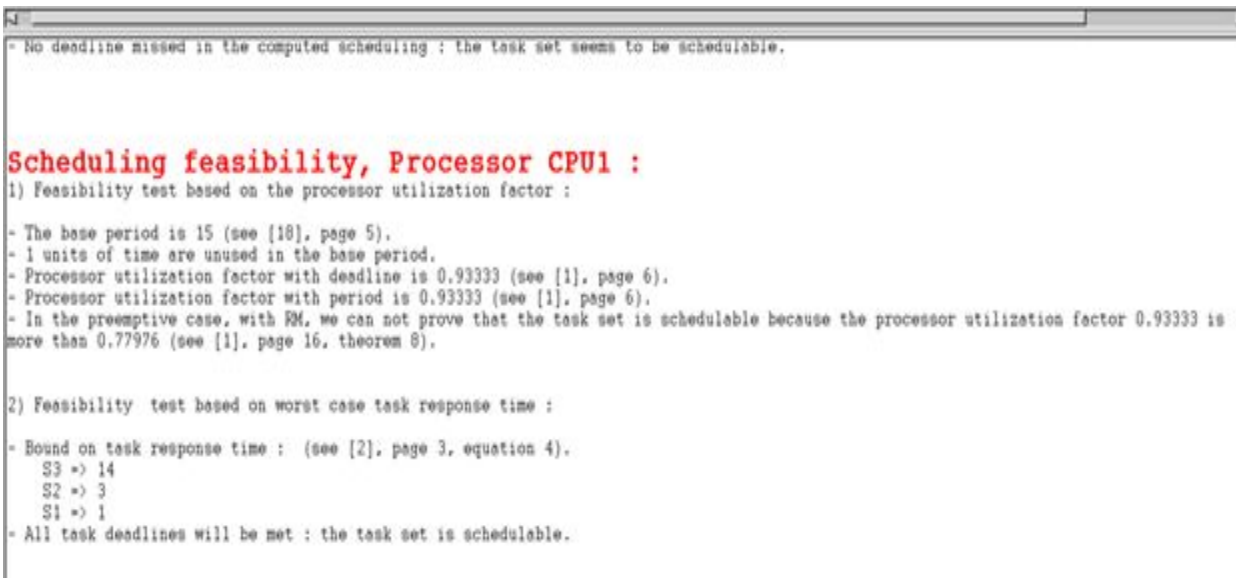
- Bound on task response time : (see [2], page 3, equation 4).
 - S4 => 16, missed its deadline (deadline = 13)
 - S3 => 4
 - S2 => 2
 - S1 => 1
- Some task deadlines will be missed : the task set is not schedulable.

Example 3

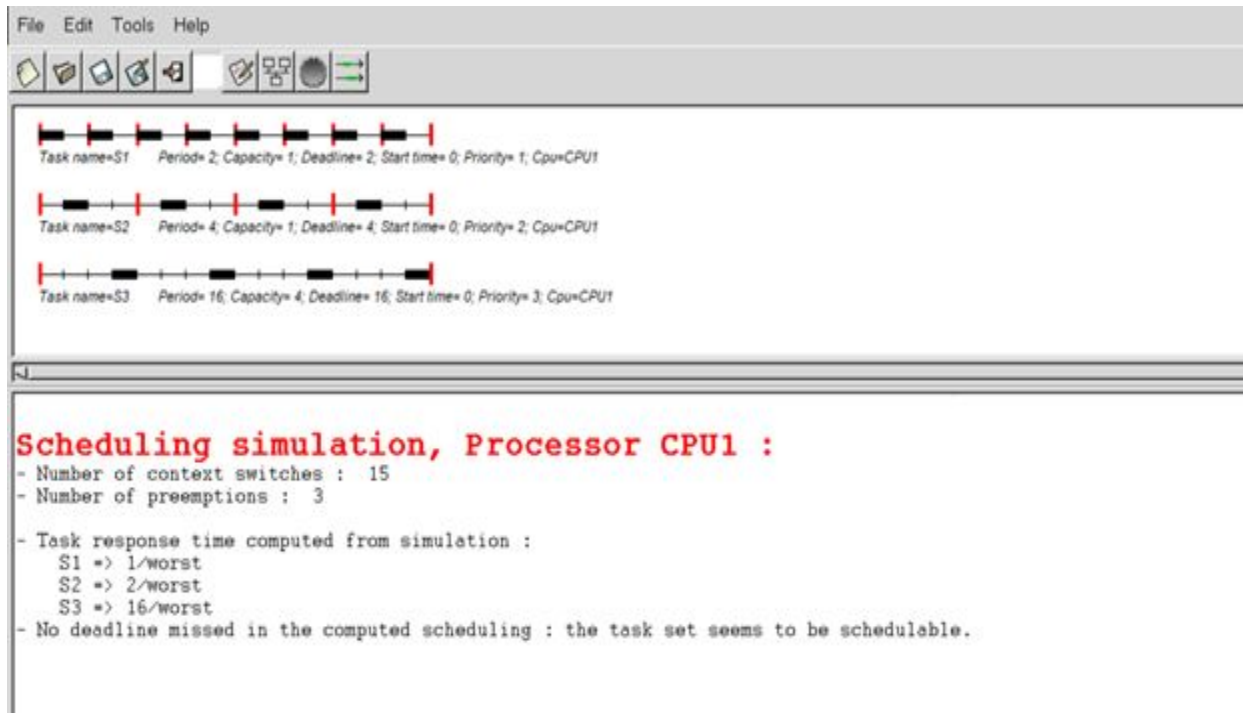


Scheduling simulation, Processor CPU1 :

- Number of context switches : 11
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 3/worst
 - S3 => 14/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.



Example 4



Scheduling feasibility, Processor CPU1 :

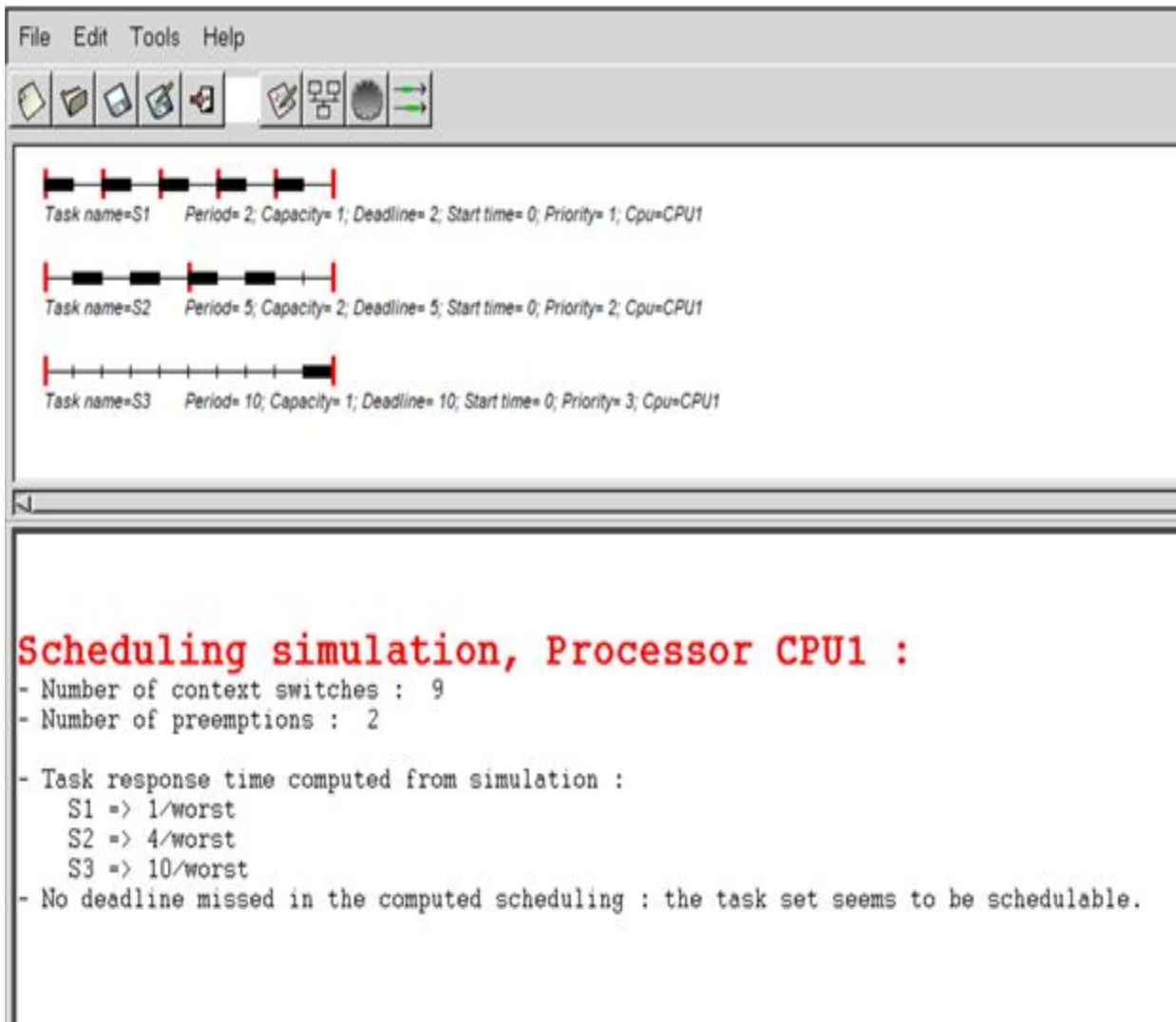
1) Feasibility test based on the processor utilization factor :

- The base period is 16 (see [10], page 5).
- 0 units of time are unused in the base period.
- 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 task response time :

- Bound on task response time : (see [2], page 3, equation 4).
 - S3 => 16
 - S2 => 2
 - S1 => 1
- All task deadlines will be met : the task set is schedulable.

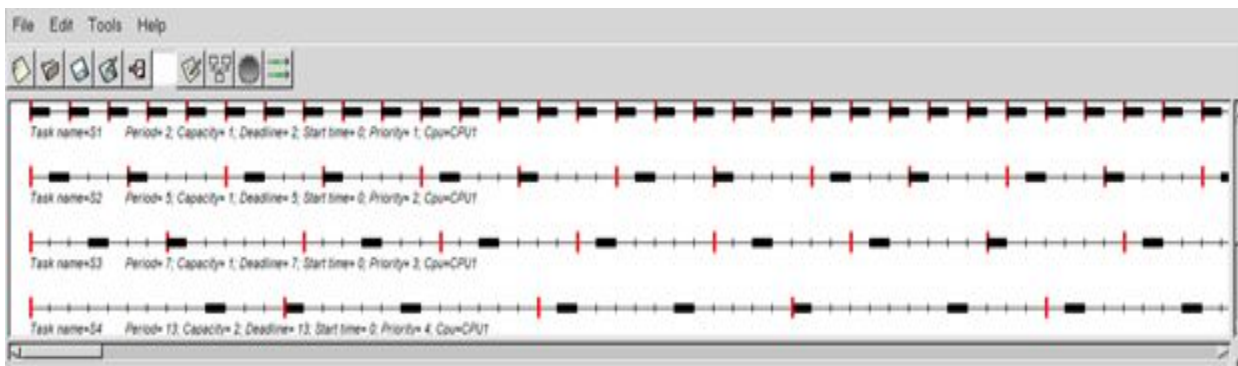
Example 5



Scheduling simulation, Processor CPU1 :

- Number of context switches : 9
- Number of preemptions : 2
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 10/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Example 6



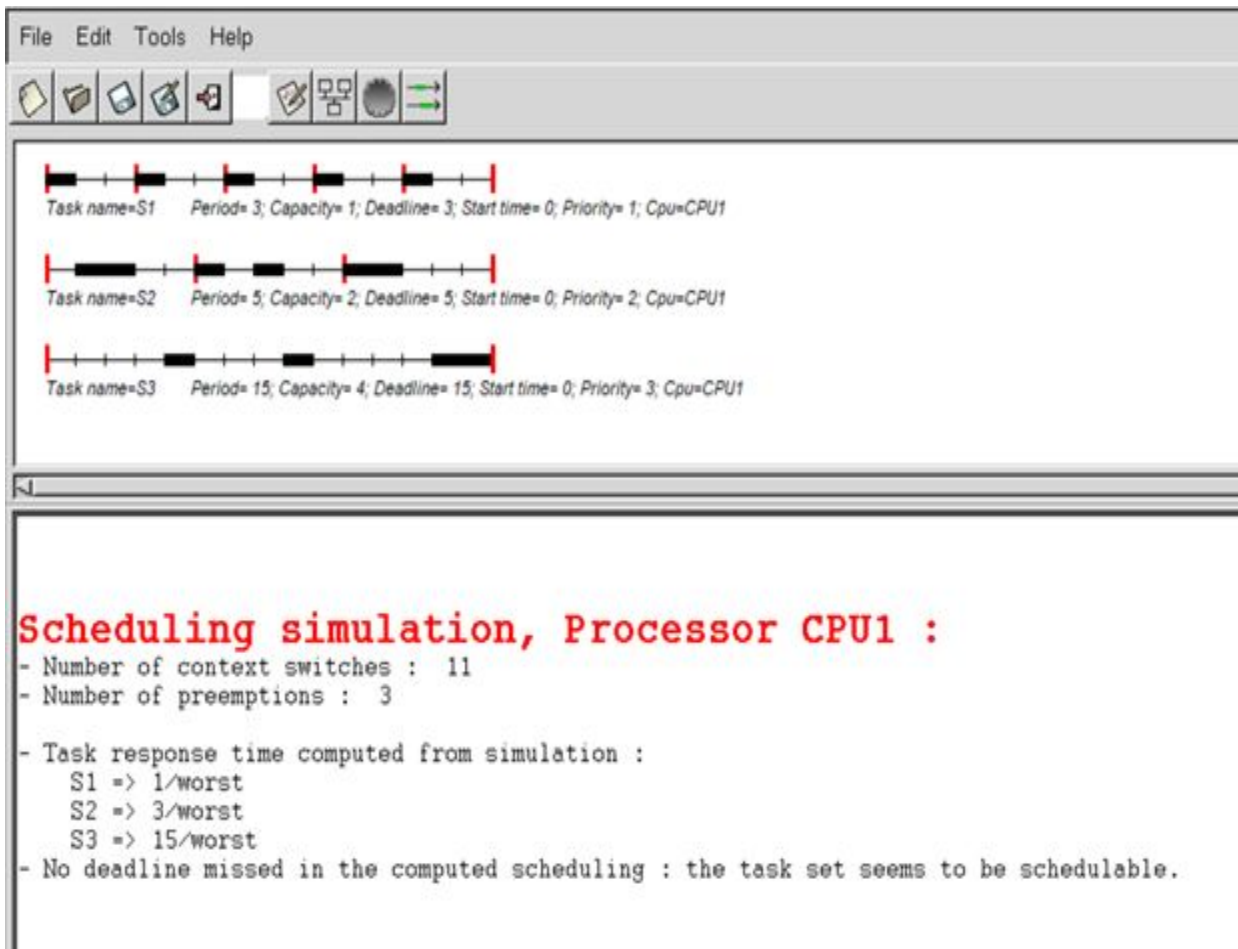
Scheduling simulation, Processor CPU1 :

- Number of context switches : 904
- Number of preemptions : 70
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 4/worst
 - S4 => 16/worst, missed its deadline (deadline = 13 ; completion time = 14), missed its deadline (deadline = 26 ; completion time = 28), missed its deadline (deadline = 39 ; completion time = 40), missed its deadline (deadline = 52 ; completion time = 54), missed its deadline (deadline = 65 ; completion time = 68), missed its deadline (deadline = 78 ; completion time = 80), missed its deadline (deadline = 117 ; completion time = 118), missed its deadline (deadline = 377 ; completion time = 378), missed its deadline (deadline = 403 ; completion time = 404), missed its deadline (deadline = 416 ; completion time = 418), missed its deadline (deadline = 429 ; completion time = 430), missed its deadline (deadline = 663 ; completion time = 664), missed its deadline (deadline = 676 ; completion time = 678), missed its deadline (deadline = 689 ; completion time = 690), missed its deadline (deadline = 767 ; completion time = 768)
- 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 base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time : (see [2], page 3, equation 4).
 - S4 => 16, missed its deadline (deadline = 13)
 - S3 => 4
 - S2 => 2
 - S1 => 1
 - Some task deadlines will be missed : the task set is not schedulable.

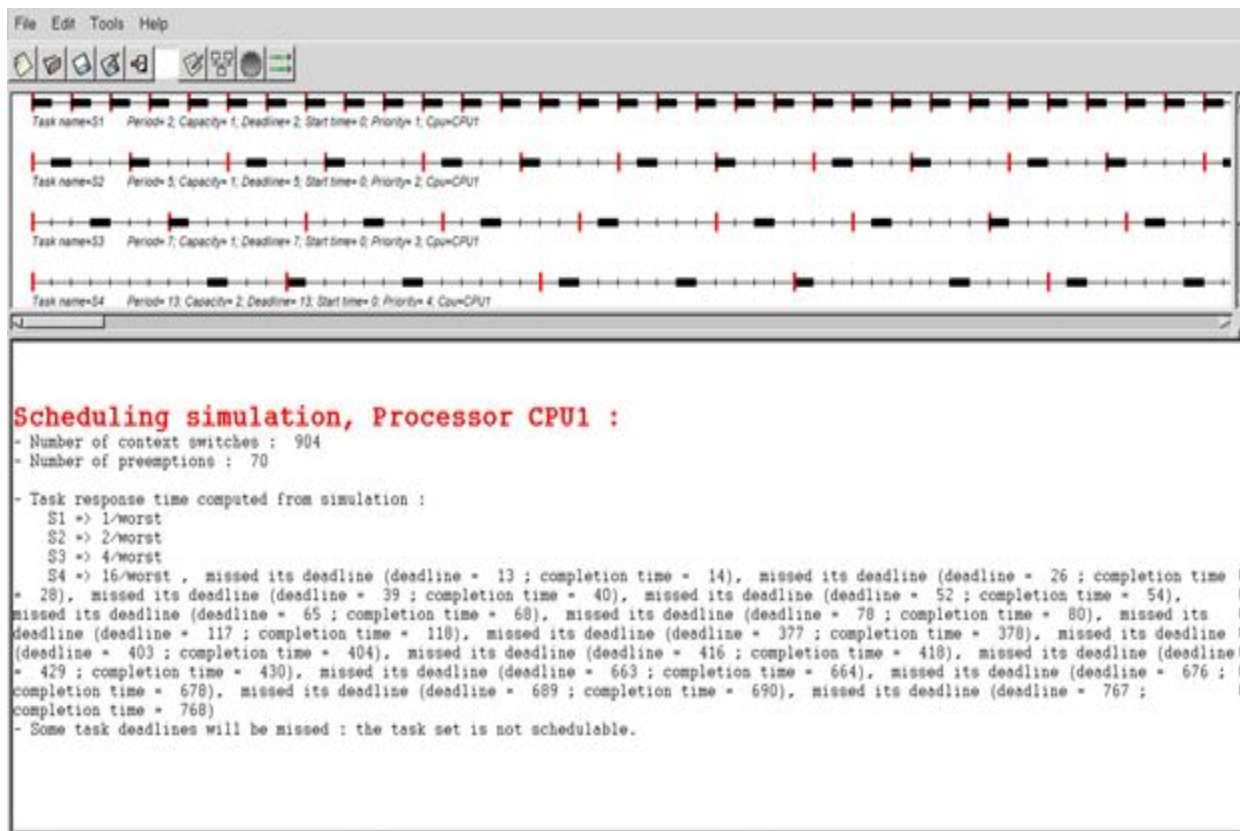
Example 7



Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 15 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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, we can not prove that the task set is schedulable because the processor utilization factor 1.00000 is more than 0.77976 (see [1], page 16, theorem 8).
- 2) Feasibility test based on worst case task response time :
 - Bound on task response time : (see [2], page 3, equation 4).
 - S3 => 15
 - S2 => 3
 - S1 => 1
 - All task deadlines will be met : the task set is schedulable.

Example 8



Scheduling feasibility, Processor CPU1 :

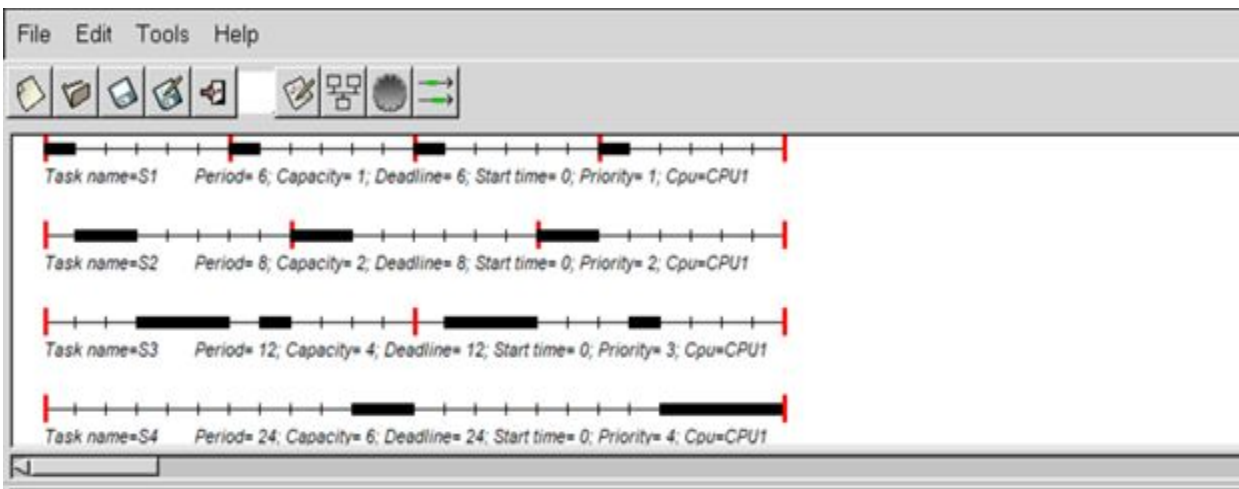
1) Feasibility test based on the processor utilization factor :

- The base period is 910 (see [18], page 5).
- 3 units of time are unused in the base period.
- 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 task response time :

- Bound on task response time : (see [2], page 3, equation 4).
 - S4 => 16, missed its deadline (deadline = 13)
 - S3 => 4
 - S2 => 2
 - S1 => 1
- Some task deadlines will be missed : the task set is not schedulable.

Example 9



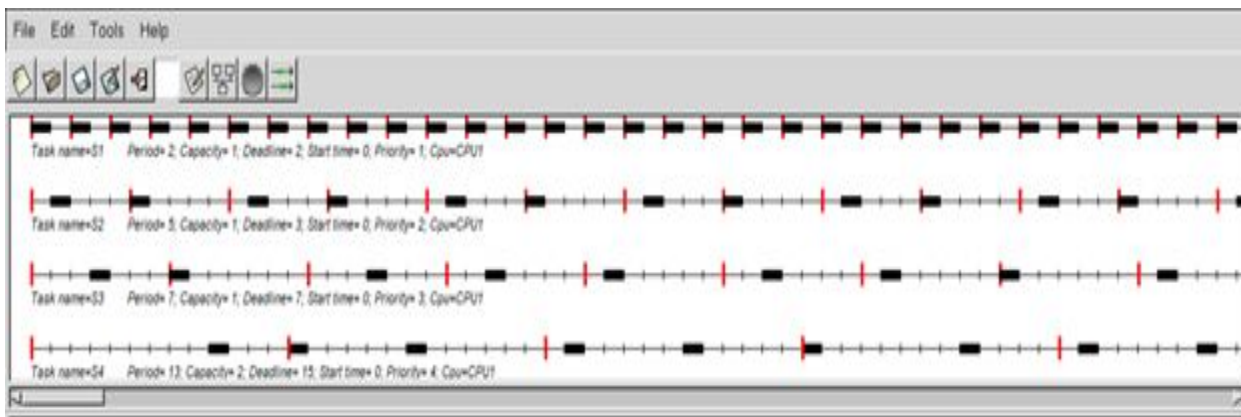
Scheduling simulation, Processor CPU1 :

- Number of context switches : 12
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 3/worst
 - S3 => 8/worst
 - S4 => 24/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 24 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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, we can not prove that the task set is schedulable because the processor utilization factor 1.00000 is more than 0.75683 (see [1], page 16, theorem 8).
- 2) Feasibility test based on worst case task response time :
 - Bound on task response time : (see [2], page 3, equation 4).
 - S4 => 24
 - S3 => 8
 - S2 => 3
 - S1 => 1
 - All task deadlines will be met : the task set is schedulable.

Cheddar Analysis Deadline Monotonic(Example 6)



Scheduling simulation, Processor CPU1 :

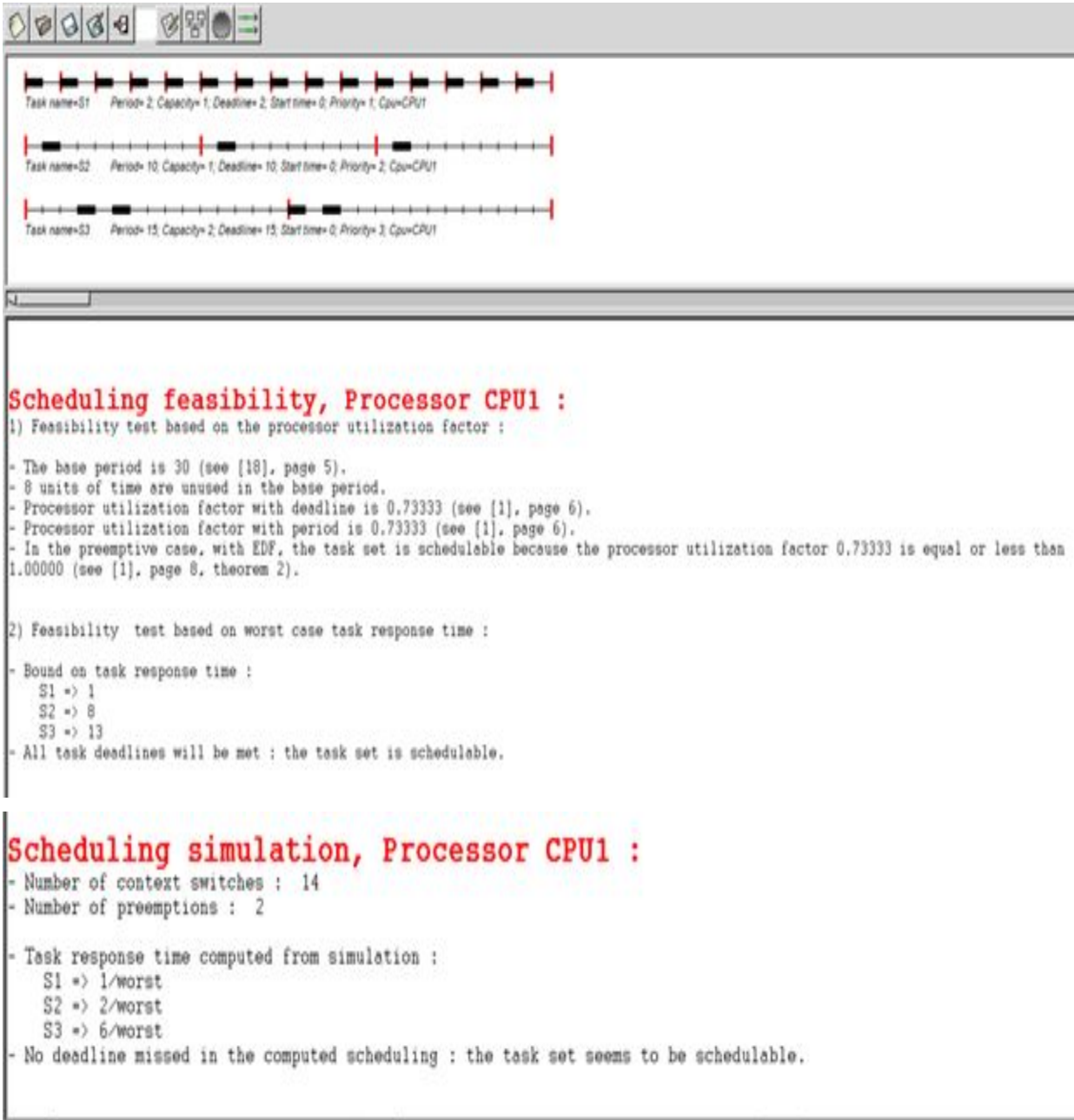
- Number of context switches : 904
- Number of preemptions : 70
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 4/worst
 - S4 => 16/worst , missed its deadline (deadline = 67 ; completion time = 68)
- Some task deadlines will be missed : the task set is not schedulable.

Scheduling feasibility, Processor CPU1 :

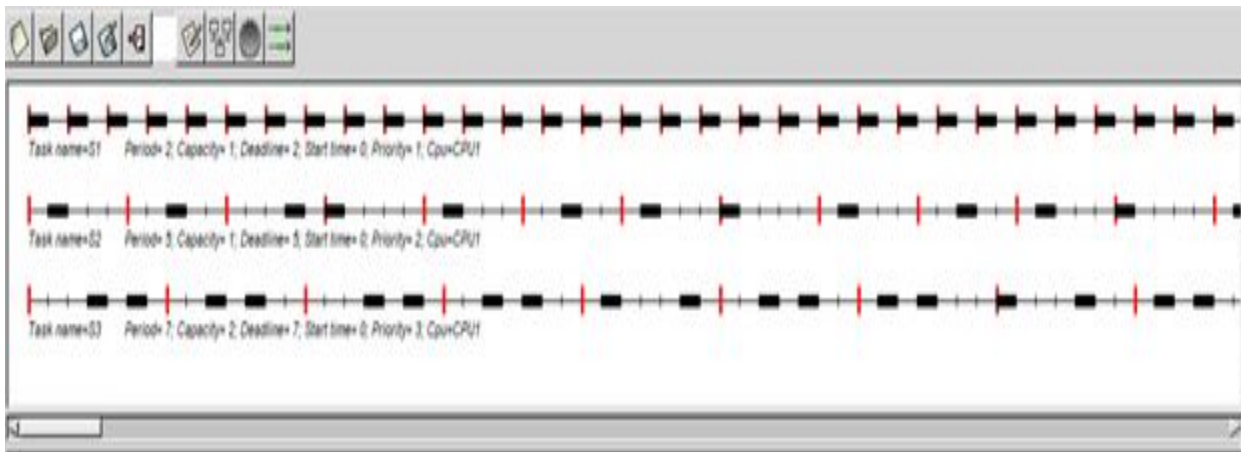
- 1) Feasibility test based on the processor utilization factor :
 - The base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - Processor utilization factor with deadline is 1.10952 (see [1], page 6).
 - Processor utilization factor with period is 0.99670 (see [1], page 6).
 - In the preemptive case, with DM, the task set is not schedulable because the processor utilization factor 1.10952 is more than 0.75603 (see [7]).
- 2) Feasibility test based on worst case task response time :
 - Bound on task response time : (see [2], page 3, equation 4).
 - S4 => 16, missed its deadline (deadline = 15)
 - S3 => 4
 - S2 => 2
 - S1 => 1
 - Some task deadlines will be missed : the task set is not schedulable.

Earliest Deadline First

Example 0



Example 1



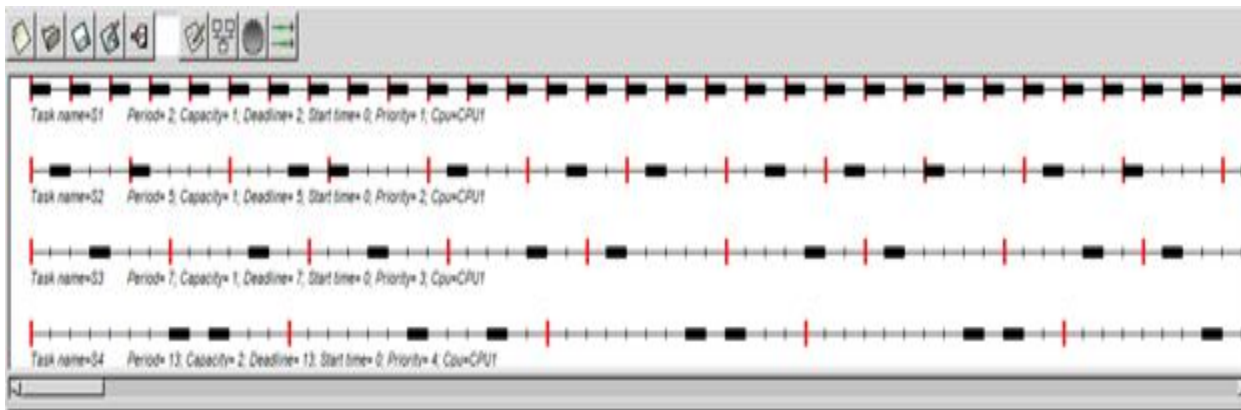
Scheduling simulation, Processor CPU1 :

- 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 seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 70 (see [18], page 5).
 - 1 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 1
 - S2 => 4
 - S3 => 6
 - All task deadlines will be met : the task set is schedulable.

Example 2



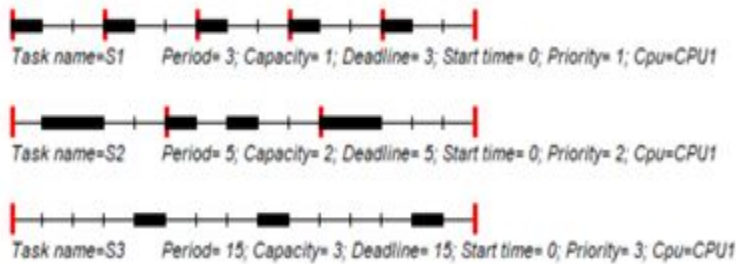
Scheduling simulation, Processor CPU1 :

- Number of context switches : 904
- Number of preemptions : 70
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 6/worst
 - S4 => 12/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 1
 - S2 => 4
 - S3 => 6
 - S4 => 12
 - All task deadlines will be met : the task set is schedulable.

Example 3



Scheduling simulation, Processor CPU1 :

- Number of context switches : 11
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 3/worst
 - S3 => 14/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

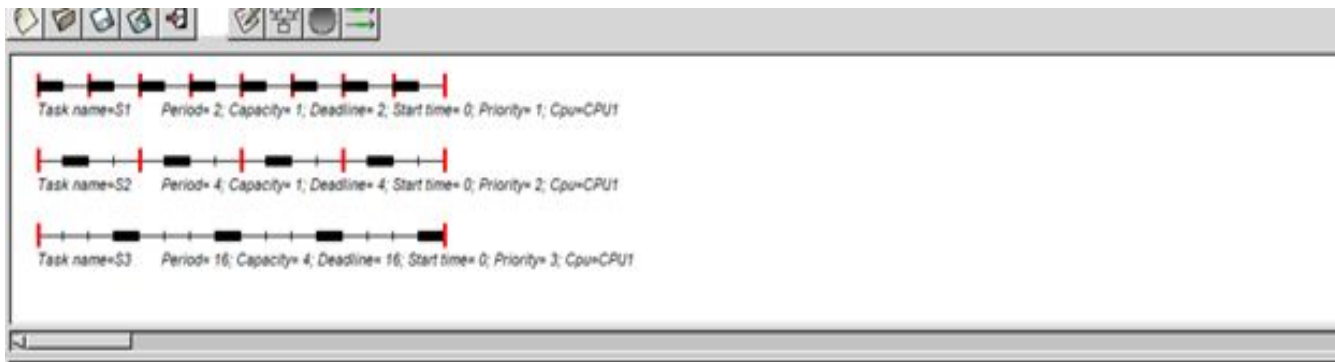
1) Feasibility test based on the processor utilization factor :

- The base period is 15 (see [18], page 5).
- 1 units of time are unused in the base period.
- 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 EDF, the task set is schedulable because the processor utilization factor 0.93333 is equal or less than 1.00000 (see [1], page 8, theorem 2).

2) Feasibility test based on worst case task response time :

- Bound on task response time :
 - S1 => 2
 - S2 => 4
 - S3 => 14
- All task deadlines will be met : the task set is schedulable.

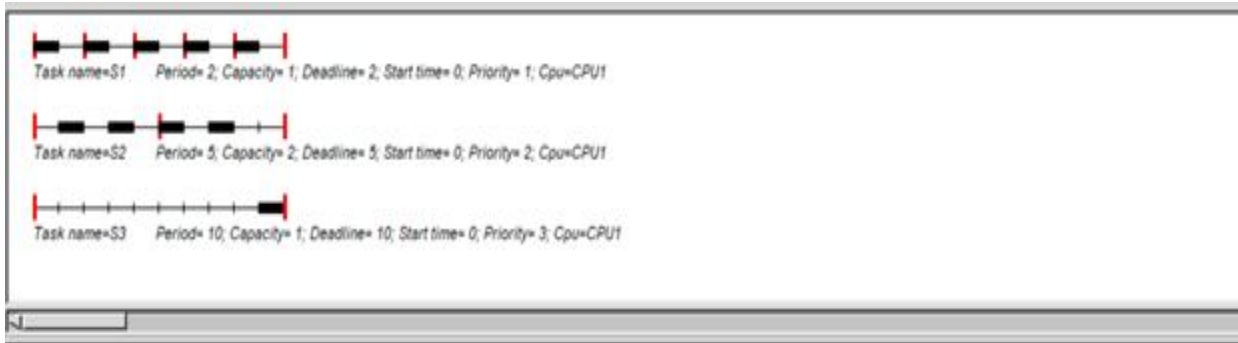
Example 4



Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 16 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 2
 - S2 => 4
 - S3 => 16
 - All task deadlines will be met : the task set is schedulable.

Example 5



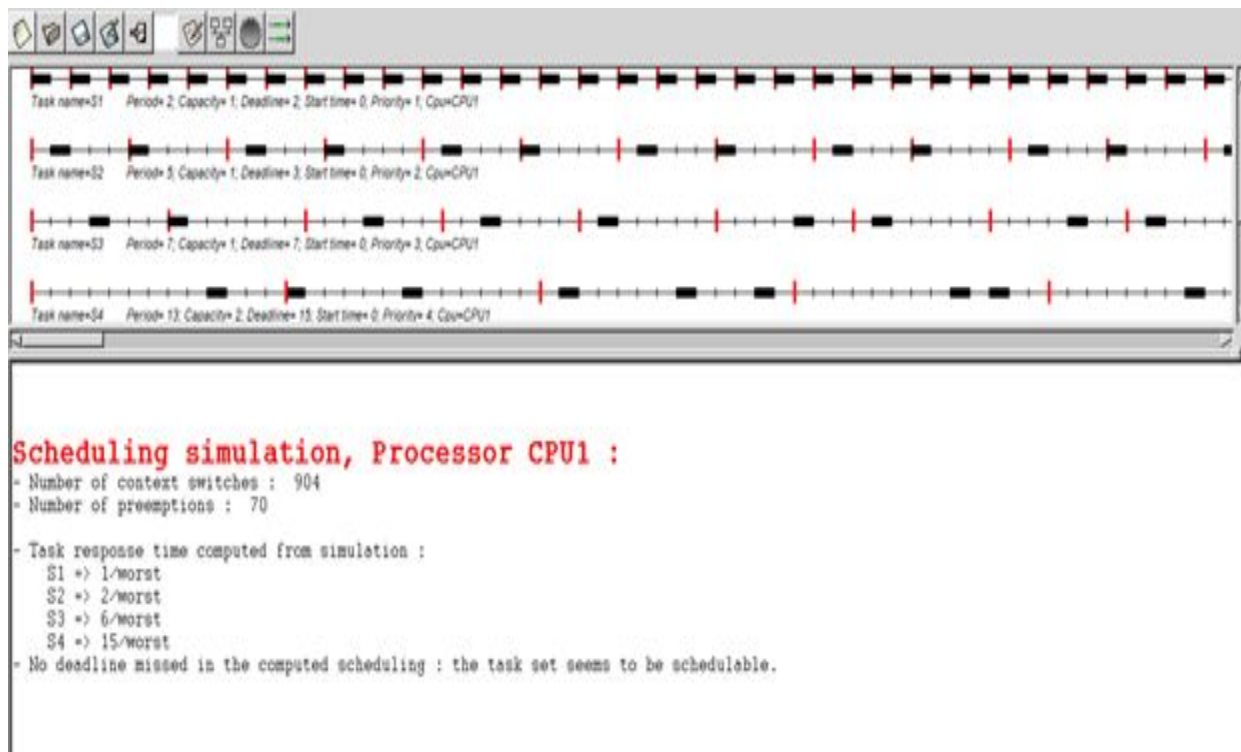
Scheduling simulation, Processor CPU1 :

- Number of context switches : 9
- Number of preemptions : 2
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 10/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 10 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 2
 - S2 => 5
 - S3 => 10
 - All task deadlines will be met : the task set is schedulable.

Example 6



Scheduling feasibility, Processor CPU1 :

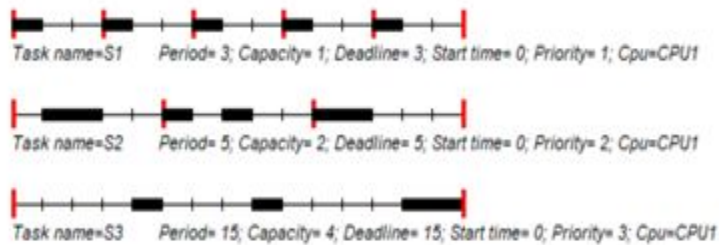
1) Feasibility test based on the processor utilization factor :

- The base period is 910 (see [10], page 5).
- 3 units of time are unused in the base period.
- Processor utilization factor with deadline is 1.10952 (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 1.10952 is more than 1.00000 (see [1], page 8, theorem 2).

2) Feasibility test based on worst case task response time :

- Bound on task response time :
 - S1 => 2
 - S2 => 3
 - S3 => 7
 - S4 => 15
- All task deadlines will be met : the task set is schedulable.

Example 7



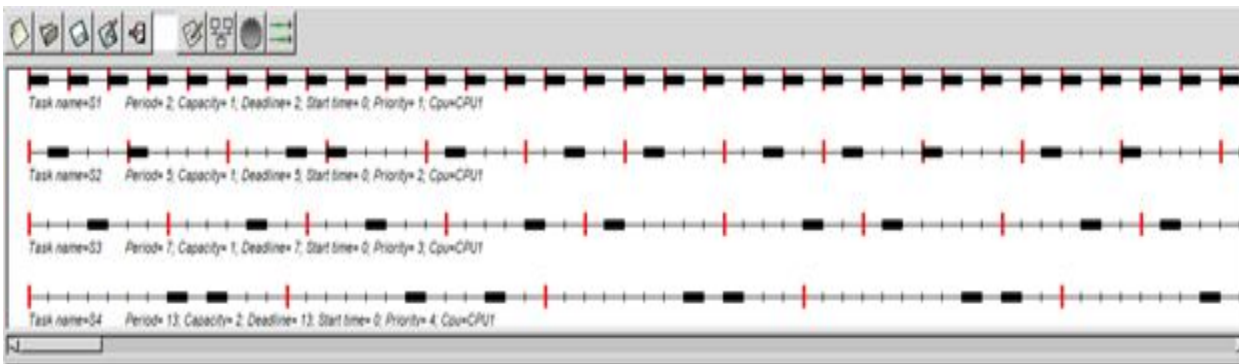
Scheduling simulation, Processor CPU1 :

- Number of context switches : 11
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 3/worst
 - S3 => 15/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 15 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 3
 - S2 => 5
 - S3 => 15
 - All task deadlines will be met : the task set is schedulable.

Example 8



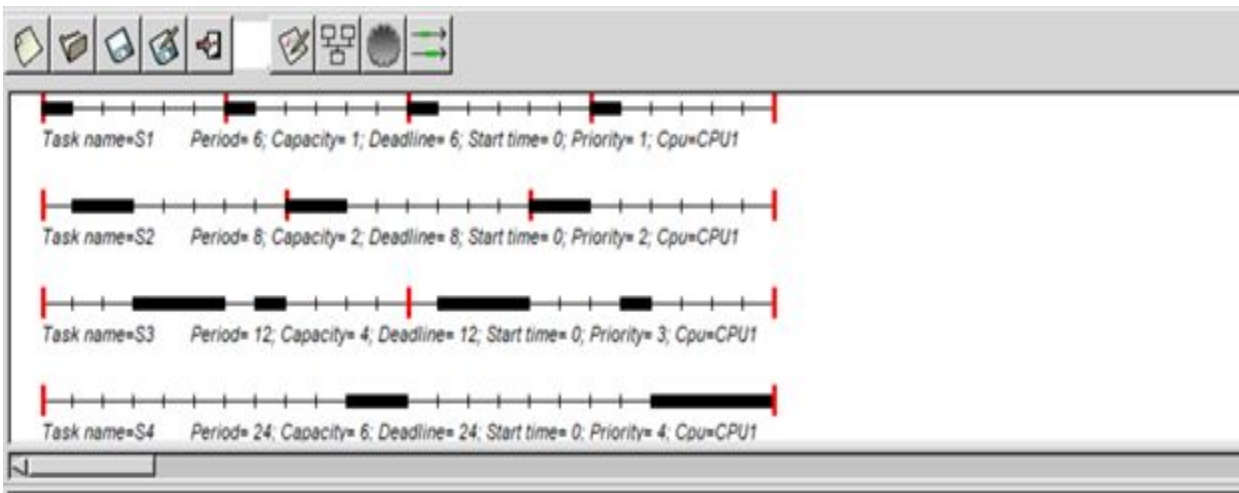
Scheduling simulation, Processor CPU1 :

- Number of context switches : 904
- Number of preemptions : 70
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 6/worst
 - S4 => 12/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 1
 - S2 => 4
 - S3 => 6
 - S4 => 12
 - All task deadlines will be met : the task set is schedulable.

Example 9



Scheduling simulation, Processor CPU1 :

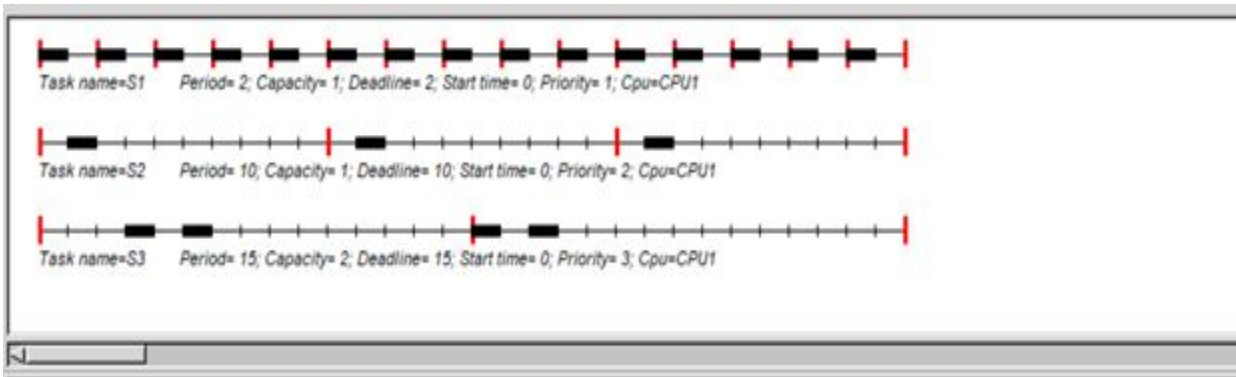
- Number of context switches : 12
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 3/worst
 - S3 => 8/worst
 - S4 => 24/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 24 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 6
 - S2 => 8
 - S3 => 12
 - S4 => 24
 - All task deadlines will be met : the task set is schedulable.

Least Laxity First policy

Example 0



Scheduling simulation, Processor CPU1 :

- 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 seems to be schedulable.

- S3 => 6/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

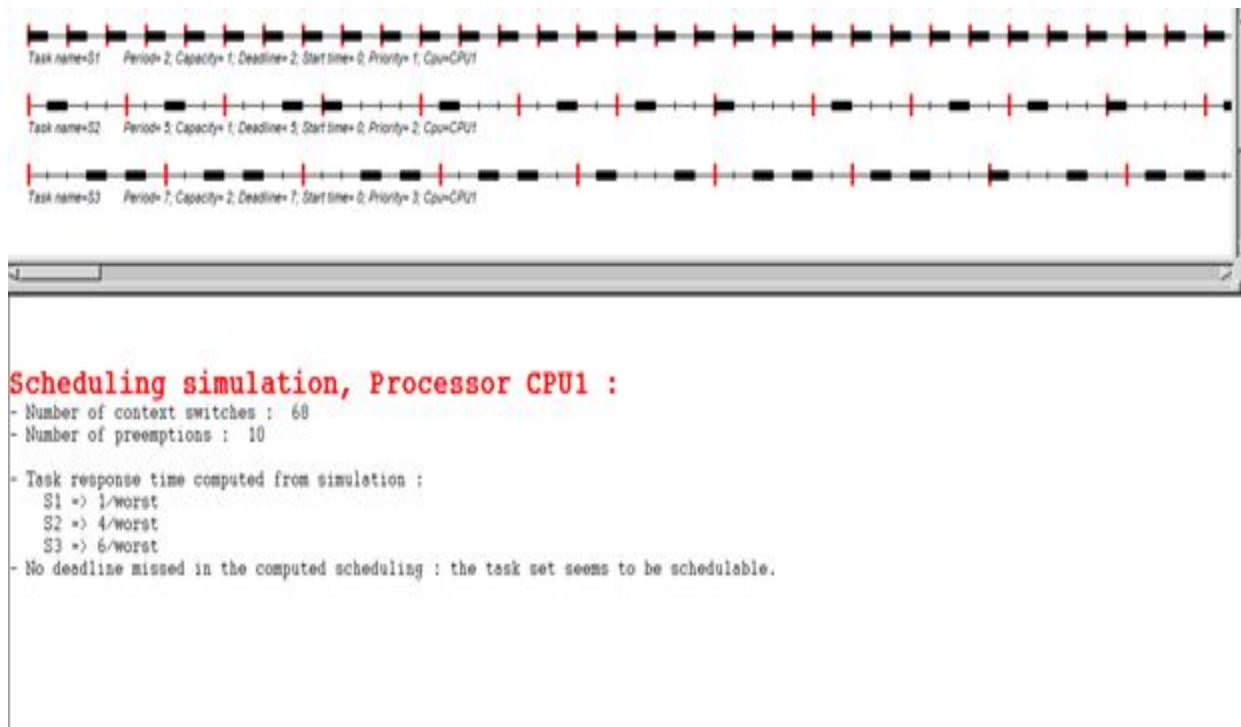
1) Feasibility test based on the processor utilization factor :

- The base period is 30 (see [18], page 5).
- 8 units of time are unused in the base period.
- 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 LLF, the task set is schedulable because the processor utilization factor 0.73333 is equal or less than 1.00000 (see [7]).

2) Feasibility test based on worst case task response time :

- Bound on task response time :
 - S1 => 1
 - S2 => 8
 - S3 => 13
- All task deadlines will be met : the task set is schedulable.

Example 1



Scheduling feasibility, Processor CPU1 :

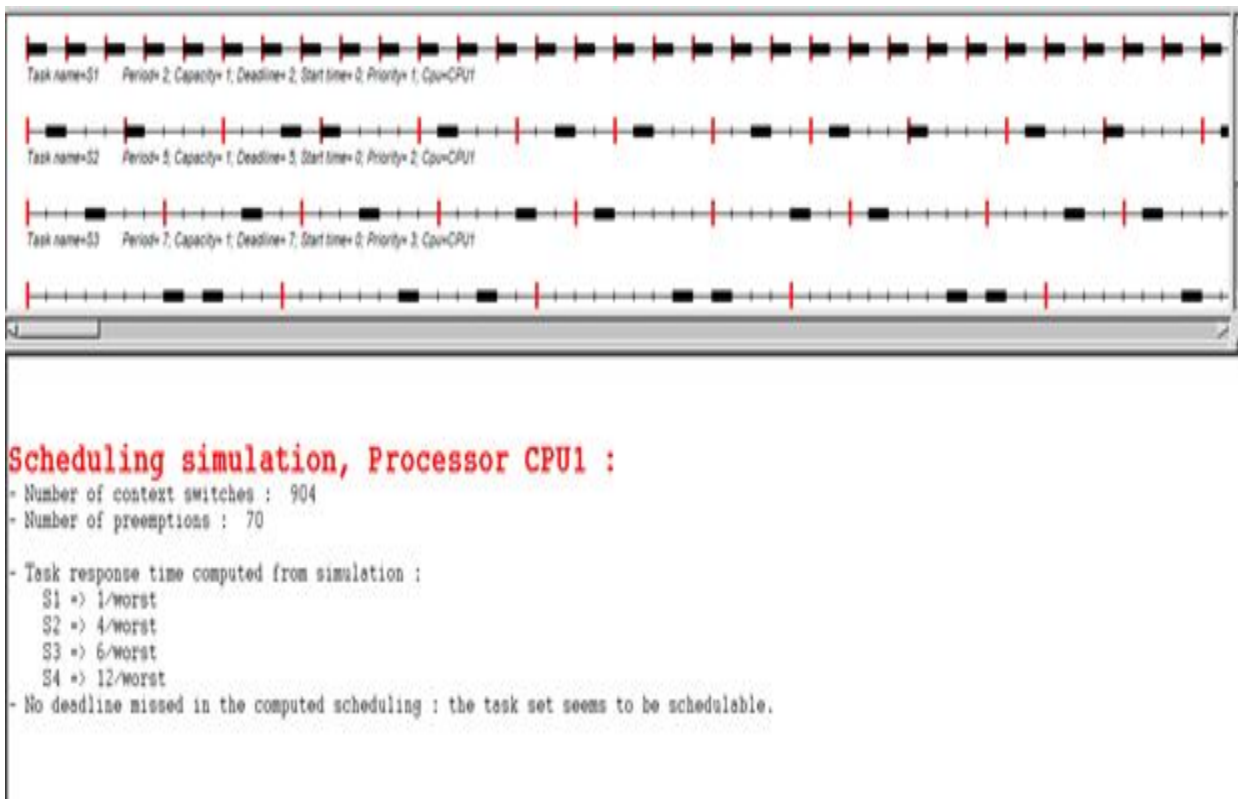
1) Feasibility test based on the processor utilization factor :

- The base period is 70 (see [10], page 5).
- 1 units of time are unused in the base period.
- 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 LLF, the task set is schedulable because the processor utilization factor 0.98571 is equal or less than 1.00000 (see [7]).

2) Feasibility test based on worst case task response time :

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

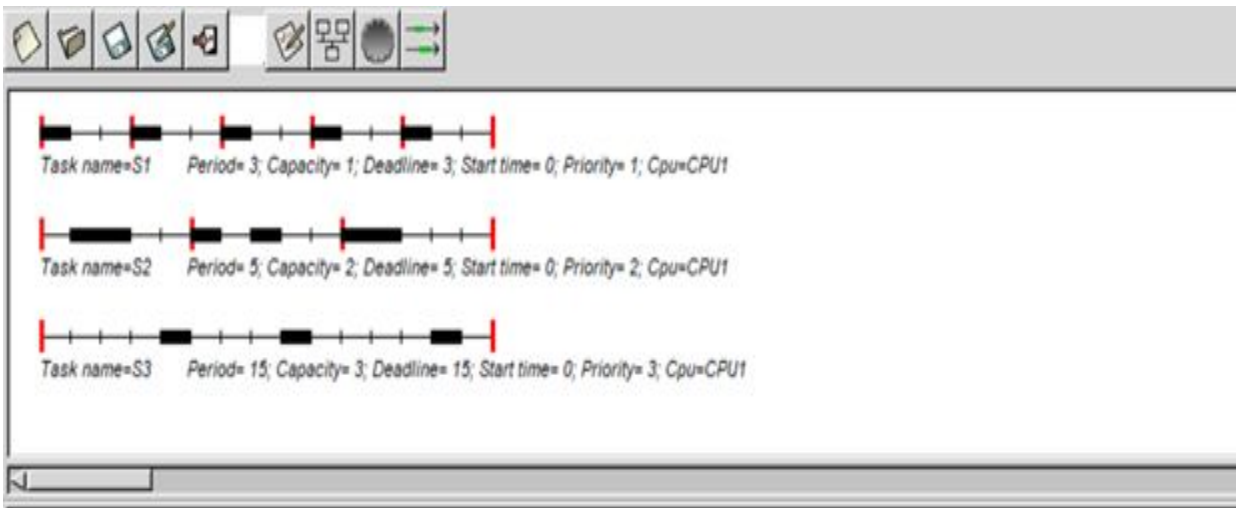
Example 2



Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 1
 - S2 => 4
 - S3 => 6
 - S4 => 12
 - All task deadlines will be met : the task set is schedulable.

Example 3



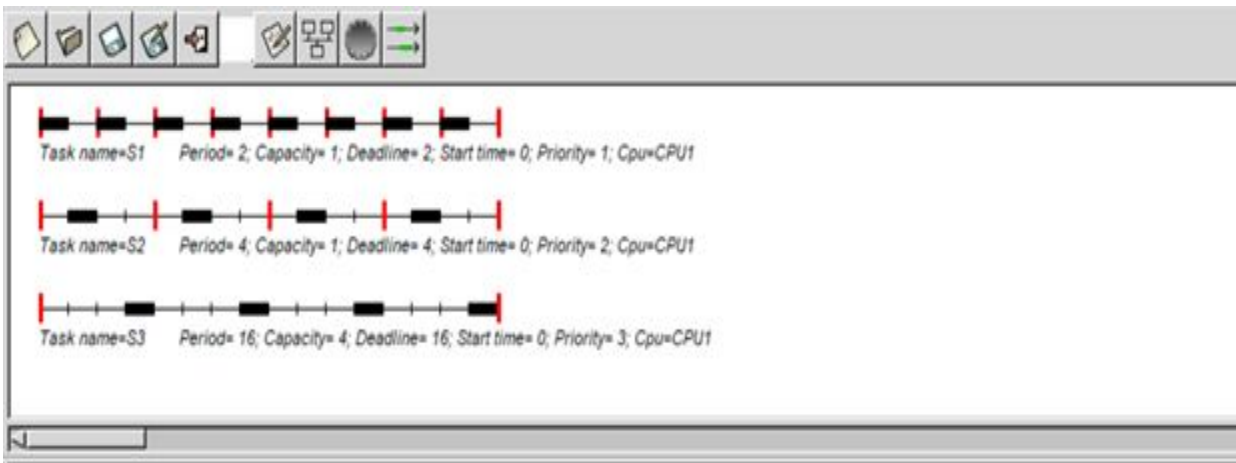
Scheduling simulation, Processor CPU1 :

- Number of context switches : 11
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 3/worst
 - S3 => 14/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 15 (see [18], page 5).
 - 1 units of time are unused in the base period.
 - 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 LLF, the task set is schedulable because the processor utilization factor 0.93333 is equal or less than 1.00000 (see [7]).
- 2) Feasibility test based on worst case task response time :
 - Bound on task response time :
 - S1 => 2
 - S2 => 4
 - S3 => 14
 - All task deadlines will be met : the task set is schedulable.

Example 4



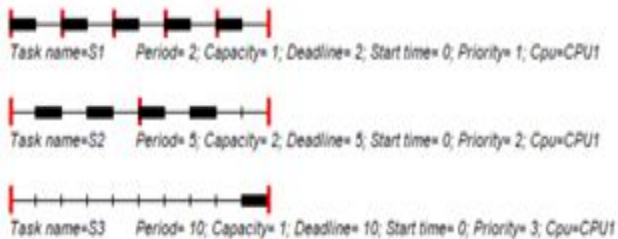
Scheduling simulation, Processor CPU1 :

- Number of context switches : 15
- Number of preemptions : 3
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 16/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 16 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 2
 - S2 => 4
 - S3 => 16
 - All task deadlines will be met : the task set is schedulable.

Example 5



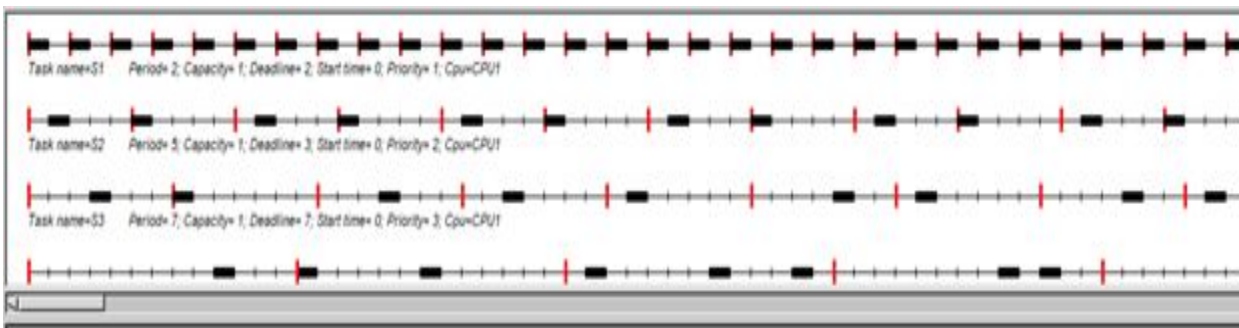
Scheduling simulation, Processor CPU1 :

- Number of context switches : 9
- Number of preemptions : 2
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 10/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 10 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 2
 - S2 => 5
 - S3 => 10
 - All task deadlines will be met : the task set is schedulable.

Example 6



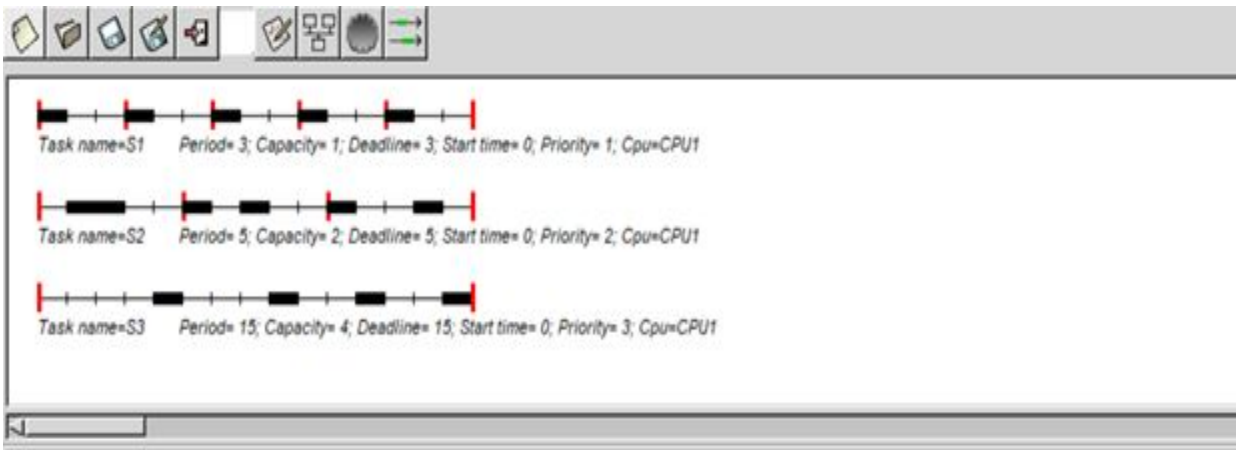
Scheduling simulation, Processor CPU1 :

- Number of context switches : 904
- Number of preemptions : 70
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 2/worst
 - S3 => 6/worst
 - S4 => 15/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - Processor utilization factor with deadline is 1.10952 (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 not schedulable because the processor utilization factor 1.10952 is more than 1.00000 (see [7]).
- 2) Feasibility test based on worst case task response time :
 - Bound on task response time :
 - S1 => 2
 - S2 => 3
 - S3 => 7
 - S4 => 15
 - All task deadlines will be met : the task set is schedulable.

Example 7



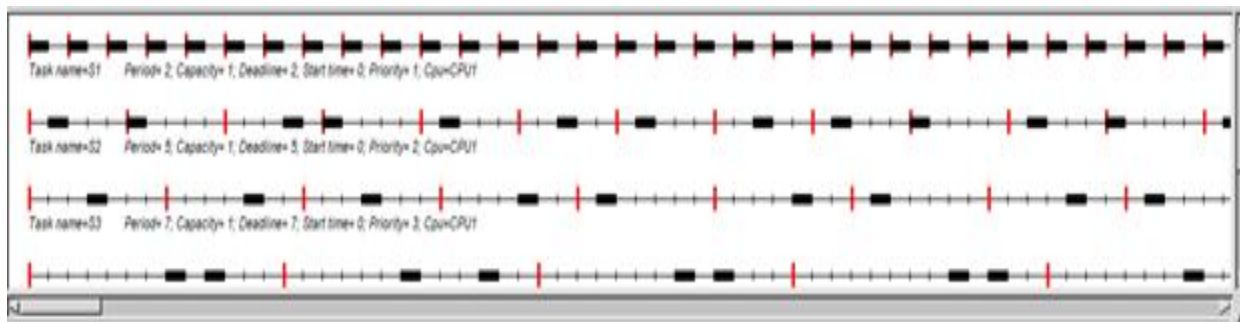
Scheduling simulation, Processor CPU1 :

- Number of context switches : 13
- Number of preemptions : 5
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 15/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 15 (see [10], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 3
 - S2 => 5
 - S3 => 15
 - All task deadlines will be met : the task set is schedulable.

Example 8



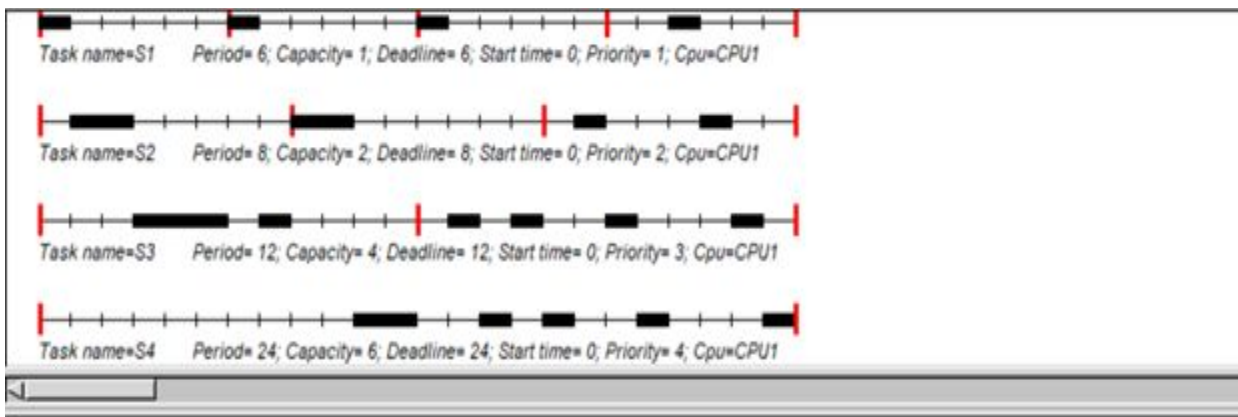
Scheduling simulation, Processor CPU1 :

- Number of context switches : 904
- Number of preemptions : 70
- Task response time computed from simulation :
 - S1 => 1/worst
 - S2 => 4/worst
 - S3 => 6/worst
 - S4 => 12/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 910 (see [18], page 5).
 - 3 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 1
 - S2 => 4
 - S3 => 6
 - S4 => 12
 - All task deadlines will be met : the task set is schedulable.

Example 9



Scheduling simulation, Processor CPU1 :

- Number of context switches : 18
- Number of preemptions : 9
- Task response time computed from simulation :
 - S1 => 3/worst
 - S2 => 6/worst
 - S3 => 11/worst
 - S4 => 24/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.

Scheduling feasibility, Processor CPU1 :

- 1) Feasibility test based on the processor utilization factor :
 - The base period is 24 (see [18], page 5).
 - 0 units of time are unused in the base period.
 - 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 task response time :
 - Bound on task response time :
 - S1 => 6
 - S2 => 8
 - S3 => 12
 - S4 => 24
 - All task deadlines will be met : the task set is schedulable.

Question 5

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 made on the RM LUB Derivation**

1. The Deadline of a particular service should be equal to the period of Service.
Deadline=Period by definition for simplicity of the derivation.
2. Fixed-priority, preemptive, run-to-completion scheduling method is used.
3. The worst-case scenario is considered that all the service in the system will simultaneously request for service. This is called the critical instant. This constraint eliminates the complexity of assuming that there is some sort of known relationship or phasing between service requests and this makes the RM LUB a more general result. [5]

Assumptions Made in Liu Layland Paper[1]

1. The request for all tasks for which hard deadlines exist 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 derivation steps in RM LUB derivation[1]**Key Step 1**

It is mentioned in the paper that “Let T_1 and T_2 be the request periods of the tasks, with $T_1 < T_2$. If we let Task 1 be the higher priority task, then, according to Theorem 1, the following inequality must be satisfied:

$$\lfloor T_2/T_1 \rfloor C_1 + C_2 \leq T_2$$

This equation was a little difficult to comprehend just from the liu layland but after reading the chapter 3 of “REAL-TIME EMBEDDED COMPONENTS AND SYSTEMS with LINUX and RTOS” by Dr. Sam Siewert and John Pratt, the equation makes more sense after having a look at the graphical representation shown below

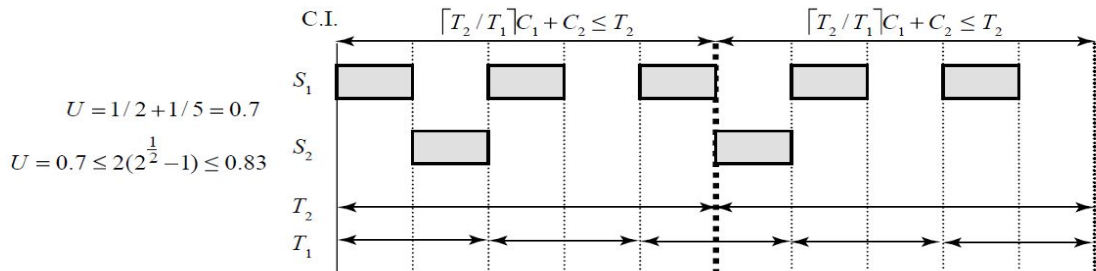


FIGURE 3.1 Example of Two-Service Feasibility Testing by Examination

Here both the task request service at the same time that is the critical instant. The graphical view makes it clear how the above equation is derived.

Note: The equation for the case where critical instant occurs the formula mentioned slightly differs. In paper floor of the ratio T_2/T_1 is taken. Whereas, in the book the ceiling of that ratio is considered for the same case.

Key step 2

There are 2 cases directly mentioned in the paper while proving Theorem 3 in the paper that is

THEOREM 3. For a set of two tasks with fixed priority assignment, the least upper bound to the processor utilization factor is $U = 2(2^{1/2} - 1)$.

It was difficult to grasp what exactly the 2 case situations were and how were the equations derived as there was no graphical explanations and the equations were directly stated without any explanation. The two cases and equations mentioned in the paper were

• Case 1

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

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

• Case 2

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

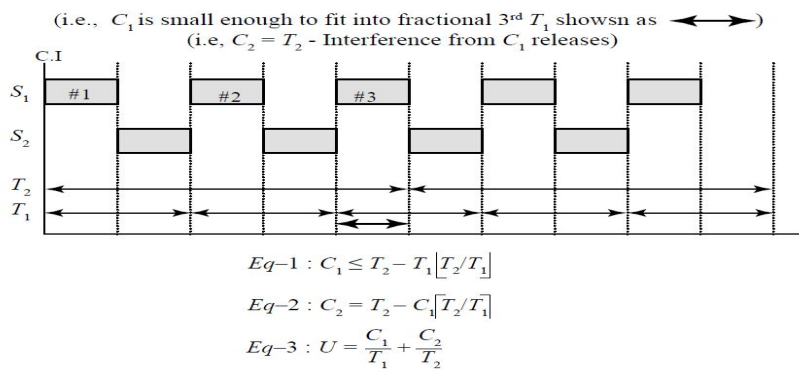
$$C_2 = -C_1 \lfloor T_2/T_1 \rfloor + T_1 \lfloor T_2/T_1 \rfloor$$

and the corresponding utilization factor is

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

These 2 cases were properly explained in the book and the equations in each case were supported by graphical representation of each case which gave a clear idea of how the equations were written down.

Case 1:



Case 2:

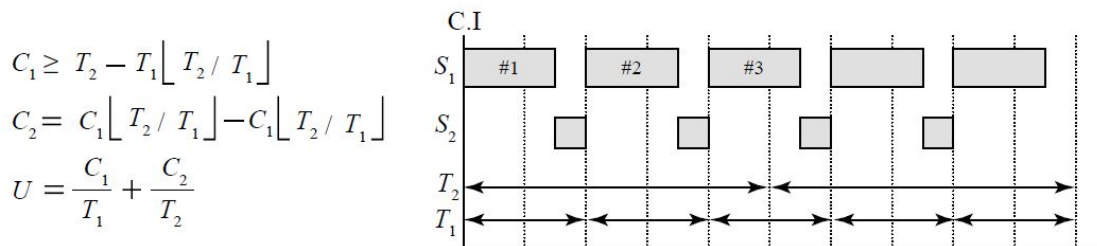


FIGURE 3.7 Case 2 Overrun of Critical Time Zone by S_1

Key Step 3

The Derivative step is not shown in the Liu Layland paper the equation used is

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

After this step, it is directly mentioned that the minimum value of the equation is calculated and then the following expression is written down.

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

The process for minimization is well documented and explained in the 3rd chapter of the book. After we take the partial of the above formula with respect to f the formula for U is derived to be

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

The use of partial derivation is quite clearly explained in the book whereas this step is skipped in the paper. It was a little difficult to comprehend how the equation was minimized just by reading through the paper.

References:

1. <https://www.cs.ru.nl/~hooman/DES/liu-layland.pdf>
2. <http://www.seas.upenn.edu/~lee/09cis480/lec-intro-rt-sched.pdf>
3. [http://dx.doi.org/10.1016/S0967-0661\(97\)00088-9](http://dx.doi.org/10.1016/S0967-0661(97)00088-9)
4. <https://www.sciencedirect.com/science/article/pii/S0967066197000889?via%3Dihub>
5. <https://www.youtube.com/watch?v=XrwqY205Zfo>
6. **Real Time Embedded Components and Systems By Dr Sam Siewert**

Exercise Staff

Harsimransingh Bindra	Srishti khare
Graduate Student Embedded Systems Engineering University Of Colorado Boulder Harsimransingh.Bindra@colorado.edu	Graduate Student Embedded Systems Engineering University Of Colorado Boulder Srishti.Khare@colorado.edu