CRITICAL SCHEDULING FOR SOFT REAL-TIME OPERATING SYSTEM

A Thesis submitted to Gujarat Technological University

for the Award of

Doctor of Philosophy

in

Computer /IT Engineering

by

Donga Jaynaben K.

179999913010

under the supervision of

Dr. Mehfuza S. Holia

GUJARAT TECHNOLOGICAL UNIVERSITY

AHMEDABAD

[June–2022]

DECLARATION

I declare that the thesis entitled **Critical Scheduling for Soft Real-Time Operating System** submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me during the period from **February 2018 to June 2022** under the supervision of **Dr. Mehfuza S. Holia** and this has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other institution of higher learning.

I further declare that the material obtained from other sources has been duly acknowledged in the thesis. I shall be solely responsible for any plagiarism or other irregularities, if noticed in the thesis.

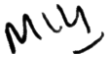
Signature of the Research Scholar:  Date: 18-6-2022

Name of Research Scholar: **Donga Jaynaben K**

Place: **Anand**

CERTIFICATE

I certify that the work incorporated in the thesis **Critical Scheduling for Soft Real-Time Operating System** submitted by **Smt. Donga Jaynaben K** was carried out by the candidate under my supervision/guidance. To the best of my knowledge: (i) the candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship or other similar titles (ii) the thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and (iii) the thesis represents independent research work on the part of the Research Scholar.

Signature of the Supervisor:  Date: 18-6-2022

Name of Supervisor: **Dr. Mehfuza S. Holia**

Place: **Anand**

Course-work Completion Certificate

This is to certify that **Mrs. Donga Jaynaben K.** enrolment no. **179999913010** is a PhD scholar enrolled for PhD program in the branch **Computer / IT Engineering** of Gujarat Technological University, Ahmedabad.

(Please tick the relevant option(s))

☐ She has been exempted from the course-work (successfully completed during M.Phil Course)

☐ She has been exempted from Research Methodology Course only (successfully completed during M.Phil Course)

☑ She has successfully completed the PhD course work for the partial requirement for the award of PhD Degree. Her performance in the course work is as follows-

Grade Obtained in Research Methodology (PH001)	Grade Obtained in Self Study Course (Core Subject) (PH002)
BB	**AB**

MUY

Supervisor's Sign

(Dr. Mehfuza S. Holia)

Originality Report Certificate

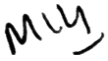
It is certified that PhD Thesis titled **Critical Scheduling for Soft Real-Time Operating System** by **Donga Jaynaben K.** has been examined by us. We undertake the following:

a. Thesis has significant new work / knowledge as compared already published or areunder consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.
b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented asAuthor own work.
c. There is no fabrication of data or results which have been compiled / analysed.
d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.
e. The thesis has been checked using **turnitin software** (copy of originality report attached) and found within limits as per GTU Plagiarism Policy and instructions issued from time to time (i.e. permitted similarity index <10%).

Signature of the Research Scholar:  Date: 18-6-2022

Name of Research Scholar: **Donga Jaynaben K.**

Place: **Anand**

Signature of the Supervisor:  Date: 18-6-2022

Name of Supervisor: **Dr. Mehfuza S. Holia**

Place: **Anand**

Thesis

ORIGINALITY REPORT

5%	4%	3%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Ms Jayna Donga, M.S. Holia. "A Hybrid Multiprocessor Scheduler for Soft Real-Time System with Processor Affinity Concept", Journal of Physics: Conference Series, 2021 Publication	1%
2	ebin.pub Internet Source	<1%
3	www-users.cs.york.ac.uk Internet Source	<1%
4	www.cse.chalmers.se Internet Source	<1%
5	drops.dagstuhl.de Internet Source	<1%
6	K.-Y. Chen, A. Liu, C.-H.L. Lee. "A multiprocessor real-time process scheduling method", Fifth International Symposium on Multimedia Software Engineering, 2003. Proceedings., 2003 Publication	<1%
7	www.ukessays.com	

PhD THESIS Non-Exclusive License to GUJARAT TECHNOLOGICAL UNIVERSITY

In consideration of being a Research Scholar at Gujarat Technological University, and in the interests of the facilitation of research at the University and elsewhere, I, **Mrs. Donga Jaynaben K.** having (Enrollment No.) enrolment no. **179999913010** hereby grant a nonexclusive, royalty free and perpetual license to the University on the following terms:

a. The University is permitted to archive, reproduce and distribute my thesis, in whole or in part, and/or my abstract, in whole or in part (referred to collectively as the "Work") anywhere in the world, for non-commercial purposes, in all forms of media;
b. The University is permitted to authorize, sub-lease, sub-contract or procure any of the acts mentioned in paragraph (a);
c. The University is authorized to submit the Work at any National / International Library, under the authority of their "Thesis Non-Exclusive License";
d. The Universal Copyright Notice (©) shall appear on all copies made under the authority of this license;
e. I undertake to submit my thesis, through my University, to any Library and Archives. Any abstract submitted with the thesis will be considered to form part of the thesis.
f. I represent that my thesis is my original work, does not infringe any rights of others, including privacy rights, and that I have the right to make the grant conferred by this non-exclusive license.
g. If third party copyrighted material was included in my thesis for which, under the terms of the Copyright Act, written permission from the copyright owners is required, I have obtained such permission from the copyright owners to do the acts mentioned in paragraph (a) above for the full term of copyright protection.
h. I understand that the responsibility for the matter as mentioned in the paragraph (g) rests with the authors / me solely. In no case shall GTU have any liability for any acts / omissions / errors / copyright infringement from the publication of the said thesis or otherwise.
i. I retain copyright ownership and moral rights in my thesis, and may deal with the

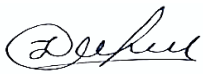
copyright in my thesis, in any way consistent with rights granted by me to my University in this non-exclusive license.

j. GTU logo shall not be used /printed in the book (in any manner whatsoever) being published or any promotional or marketing materials or any such similar documents.

k. The following statement shall be included appropriately and displayed prominently in the book or any material being published anywhere: "The content of the published work is part of the thesis submitted in partial fulfilment for the award of the degree of Ph.D. in **Computer / IT Engineering** of the Gujarat Technological University".

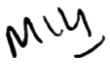
l. I further promise to inform any person to whom I may hereafter assign or license my copyright in my thesis of the rights granted by me to my University in this nonexclusive license. I shall keep GTU indemnified from any and all claims from the Publisher(s) or any third parties at all times resulting or arising from the publishing or use or intended use of the book / such similar document or its contents.

m. I am aware of and agree to accept the conditions and regulations of Ph.D. including all policy matters related to authorship and plagiarism.

Date: 18-6-2022 Signature of the Research Scholar: 

Place: **Anand**

Recommendation of the Supervisor: **Strongly Accepted**

Signature of the Supervisor: 

Thesis Approval Form

The viva-voce of the PhD Thesis submitted by **Smt. Donga Jaynaben K.** (Enrollment No. **179999913010**) entitled **Critical Scheduling for Soft Real-Time Operating System** was conducted on **Saturday, 18/6/2022** (day and date) at Gujarat Technological University.

(Please tick any one of the following options)

☑ The performance of the candidate was satisfactory. We recommend that he/she be awarded the PhD degree.

☐ Any further modifications in research work recommended by the panel after 3 months from the date of first viva-voce upon request of the Supervisor or request of Independent Research Scholar after which viva-voce can be re- conducted by the same panel again

(briefly specify the modifications suggested by the panel)

☐ The performance of the candidate was unsatisfactory. We recommend that he/she should not be awarded the PhD degree.

(The panel must give justifications for rejecting the research work)

Name and Signature of Supervisor : **Dr. Mehfuza S. Holia**

External Examiner 1: **Dr. Pancham Shukla**

External Examiner 2: **Dr. Sumantra Dutta Roy**

ABSTRACT

The scheduling approaches have largely been analyzed so far, but the attention given to multi-processor environments and real-time operating system is very less. Mostly all the approaches which are optimal are appropriate for the general-purpose operating system which cannot accomplish the real-time systems requirements as the RTS is closely concerned about the constraints of timing. The correctness of output is extremely affected by the deadlines means the output must be generated within a given time-bound. Nowadays, the real-time systems are multi-tasking and multiprocessing, so the scheduler plays an important role for controlling the functioning into the system. Different researchers had designed numerous techniques for real-time task scheduling to meet the varying needs of different environments like uniprocessor systems and multiprocessor systems. The selection of scheduling approach is an essential task as the output is greatly affected by which approach is being used in which environment. Furthermore, existing schedulers designed for supporting RTS aren't absolutely optimal for multiprocessor system. The schedulers designed earlier had focused on priority assignment which decides only the execution order of a task in the task set.

Nowadays, we are entering into the era of the high-performance computing, which demands for the multi-processor environment. The uniprocessor system and the multiprocessor system; both are having their individual requirements and challenges as per their architecture. In the multiprocessor system the scheduler has to address both the problems like process selection (Priority assignment) as well as processor selection (set Processor affinity).All the existing approaches for multi-processor real-time system described in literature are divided in two categories; 1) Global scheduling approach 2) Partitioned scheduling approach and they all are application dependent approaches.

The concept of affinity is used widely in the non-real-time environment to improve the cache performance but not in SRTS. In current era, the concept of APA for scheduling real-time tasks in hard real-time systems are used by the few RTOS which can give more flexible and generalized scheduler rather than the application based traditional algorithms described in literature. The context switching can be reduce and performance of cache may be improved by putting restrictions on the migration but it creates very huge effect on other parameters which are most essential for the any RTOS that is the overall system

schedulability will be degraded and increasing the deadline miss ratio. Meeting the deadline for improving the schedulability is the very important aspect in any real-time system.

The work presented in this research has implemented G-JLFP-APA scheduler for soft real-time system which is global approach using processor affinity and full migration with job-level-fix-priority approach. This research also presents another P-FP-APA scheduler for soft real-time system which is partitioned approach using processor affinity and no migration with fix-priority assignment approach. After analyzing the performance of G-JLFP-APA and P-FP-APA schedulers to combine the advantages of both the approach; A new scheduling approach called PrASWM is implemented which is using affinity concept for processor selection and without Flexible Migration Policy along with JLDP. After that by combining the flexible migration policy proposed a generalized scheduler PrAMS (multi-processor scheduler based on processor affinity) for the soft real-time system which focuses on three different aspects: i) processor allocation which is done using static affinity assignment to achieve generalized approach ii) Process selection (priority assignment) which is done using JLDP(Job Level Dynamic Priority) to improve the efficiency of algorithm iii)Flexible Migration policy which is done by task shifting with priority reprioritization mechanism for improving the system's schedulability and to solve the priority inversion problem. The performance measure parameters taken into consideration are Schedulabilty, No. of Context Switches, Tardiness, Deadline Miss Ratio and CPU_Utilization. The soft real-time system's overall schedulability is improved by the proposed scheduler (PrAMS) as compared to conventional scheduling algorithms, reduces the average deadline miss ratio, reduces the tardiness and improves the CPU_Utilization in the multiprocessor system.

Acknowledgement

As a firm believer in the power and blessings of the **ALMIGHTY GOD**, I firmly believe that he is the **Lord Krishna** who has done this gigantic task as he is **'Karta- the doer'** of everything and whose wishes run supreme to any action or happening in the known and unknown spheres of universe. I certainly feel blessed that my sincere, often heart touching and tear-some prayers have been answered by his presence which I feel was more pertinent during my long hours of work. I personally attribute it as a single point source from whom I have drawn inspiration from.

This thesis is the result of a journey of work whereby I have received contributions, help and support from a number of friends, associates, researchers, and my family members. I have now the great opportunity to express my sincere gratitude and appreciation to all of them.

Firstly, I am extremely grateful to my honourable supervisor **Dr. Mehfuza Holia**, Electronics Department, Birla Vishvakarma Mahavidyalaya, V. V. Nagar, for her continuous guidance, motivation, encouragement and support throughout my research work. Her dedication towards work, method of problem solving and helping nature has inspired me and will continue to inspire me for the rest of my career. She helped me and motivated a lot during my pregnancy so I could complete my work timely. I cannot imagine of having a better advisor and mentor for my research work. I am extremely thankful to her, for the technical guidance, help and conveniences provided to me for successful completion of my thesis and research work.

I would like to thank my 1st doctoral progress committee member **Dr. Jagadish M. Rathod,** Professor, Electronics Department, BVM for his insightful comments and encouragement during entire period of research work and guidance regarding publications.

I would like to extend my sincere regards and thanks to my 2nd doctoral progress committee member **Dr. Narendra M. Patel**, Associate Professor, CE Department BVM for his insightful comments continuous guidance and whole hearted cooperation since I started journey of this research work. Words are inadequate to appreciate the knowledge, insight and patience with which he guided me during this work. I am very much thankful to him for sparing his precious time whenever I asked without any hesitation.

It is an honour for me to express my deep sense of gratitude to late Shri **Dr. C. L. Patel Sir**, Chairman, Charutar Vidya Mandal (CVM) for giving permission to pursue in-service Ph.D. I would like to thank **MBIT Family** and **GTU Team** for giving me the opportunity, facilities and inspiration to complete this thesis on time. Sincere thanks to the executives and management of MBIT, CVM and GTU who placed their faith in my ability to deliver innovative research and who stood with me, when I needed them.

Sentiments from the core of the heart, when translated into words, seldom convey what one truly wishes to express for prayers, love, blessings, encouragement and support provided to me by my **Guruji Shri Kishorchandara Maharajaji**, **Shri Krishn das Balyogiji**, all the **Saints and Devotees** of my Lord. No formal words can fully convey my thanks to them.

"The essence of Bhagavat Gita lies in man's devotion to work and service to mankind. If this is ensured, rest all follows". I found it true in **Naineshbhai Saheb**. His selfless help has not only removed many obstacles from my path but has also given new meaning to my life as a whole. I with full respects accept his love, blessings and best wishes.

My special thanks to friend and a helping hand, **Rahul Goradiya** for his technical support, continuous guidance and unconditional help throughout my work. This accomplishment would not have been possible without his prayers and dedicated help inspite of his busy schedule. My special thanks to best **friends Dhaval , Rima and Mrs. Sweta** for their moral support.

I feel indebted to my respected parents for providing their continuous inspiration and motivation to complete this research work. I firmly believe that person's first Guru is his/her Mother and it was her blessings only that kept me going. I take this opportunity to give my heartfelt tributes to my mother and father who has always supported and applauded my academic efforts constantly.

Words cannot express how grateful I am to my family members for all the sacrifices they have made for me. I owe deepest gratitude to my parents, my sister, my sister in-law and in-laws for their mental support. My heartiest thanks to my beloved husband and my backbone who inspired me to start this research work, **Dr. Vatsal** for his all sacrifices. This work would not have been completed without his unconditional love and support during entire period of my work. I am short of words to express my loving gratitude to my adorable son **Madhav**, whose innocent smile and love has given me mental support during the entire

work. Children are innocent and Innocence is loved by Lord". I have felt the divine presence of the almighty whenever my young nephews **Isha, Mann and Vraj** have prayed for my success and goodwill.

I am highly obliged to all the persons who helped me visibly or invisibly for the successful completion of this Ph.D. work.

Donga Jaynaben K.

Table of Contents

List of Abbreviations

RTOS	Real-time Operating System
RTS	Real-time Systems
OS	Operating System
HRTS	Hard Real-time Systems
SRTS	Soft Real-time Systems
FRTS	Firm Real-time Systems
RM	Rate Monotonic
DM	Deadline Monotonic
EDF	Earliest Deadline First
RTA	Real-Time Application
G-JLFP-APA	Global-Job level fixed priority-APA
PrAMS	Processor Affinity based Multiprocessor Scheduler
P-FP-APA	Partitioned-Fixed Priority-APA
APA	Arbitrary Processor Affinity
FP	Fixed Priority
JLFP	Job Level Fixed Priority
JLDP	Job Level Dynamic Priority
PrASWM	Processor affinity based scheduler without flexible migration
LITMUSRT	Linux Testbed for Multiprocessor Scheduling in Real-Time Systems
WCET	Worst case Execution Time
SRTN	Shortest Remaining Time Next
FCFS	First Come First Served
SJF	Shortest Job First
HRN	Highest Ratio Next
LLF	Least Laxity First
CP-EDF	Controlled Pre-emptive EDF
DC-gEDF	Domain-Cluster-gEDF
gEDF	Group EDF

List of Figures

List of Tables

Chapter 1:
Introduction

1.1 Real-Time Systems

In the past eras, the real-time systems were being used infrequently and limited to some specific applications like defense and space. Nowadays in our day-to-day life, embedded systems had become an necessary part of our life. As an example, we have numerous consumer products like smart phones, smartcards, music players, video conferencing applications, routers, video games, digital cameras, microwave ovens, VoIP or other electronic devices are rapidly making change in our lives; we are using office products like security systems, fax machines, laser printers. Moreover, in hospitals also we are having use of real-time systems in the form of medical treatment tools and imaging system. There are huge number of real-time system based tools or gadgets we are using indirectly in our daily life like base stations in cellular systems, industrial robots, industrial plants automation systems [62][92]. Based on one survey it is estimated that 70% of processors among all the manufactured worldwide are deployed for real-time applications [46][109].

An embedded system is nothing but a group of various components like computer hardware, software, electrical and mechanical components put to gather to achieve a specific goal on the spot at particular time instant. Because of timing constraint, sometimes they can be called RTS also. In all the real-time system the first and foremost requirement is to complete the task within a stipulated time, the scheduling is the important mechanism to accomplish the requirement and the RTOS helps to real-time system by providing scheduling mechanism. So RTOS becomes the foundation of any Embedded Systems. Due to this reason, to design an appropriate scheduler is the challenging task for any RTS and the scheduling became an important research topic since the beginning of the RTS.

A RTOS can be defined as, **it is one kind of OS that will help to the real-time systems by applying scheduling mechanism for getting an exact result within a stipulated time slot and this time duration is said the deadline [33].**

TABLE 1.1: Differentiation of General OS and Real-time OS [119]

	General Purpose OS	Real-Time OS
Determinism	Non-deterministic	Deterministic
Pre-emptive kernel	Not Compulsory	Preempt-able all the kernel operations
Priority Inversion problem	No such technique is present	Having approaches for preventing the priority inversion problem
Scheduling of Tasks	Scheduling based on Task	Scheduling based on Time
Application	Used in desktop PC or other general purpose PCs	Typically used in embedded real-time applications

The TABLE 1.1 presents the differentiation of general purpose OS and a real-time OS based on their properties like kernel constraints, priority inversion problem, process scheduling, determinism and applications [86][119]. In RTOS the task scheduling is highly deterministic. The system can be called the real-time system if it contains at least one task which is real-time task. The major distinction between non-real-time job and real-time job is that a real-time job is always being distinguished by task's deadline which tells the system that the particular job must complete within a stipulated time.

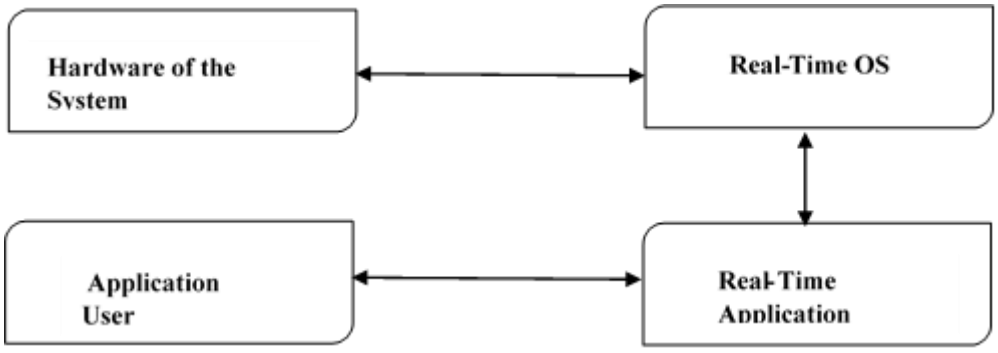


FIGURE 1.1: Relationship of Embedded system and real-time applications [52]

There are three different categories of real-time system based on its deadline meeting criticality [3][9][91]:

1) **Hard RTS:** A system is said a HRTS (hard Real-time system) if the task doesn't produce the output within the particular deadline; due to this the whole system fails and

may create some disaster like loss of property or loss of life. Nature of several hard real-time system is safety-critical. For example, Anti-missile system, Air traffic control, Nuclear Power plant control etc.[74][94].

2) **Firm RTS:** A system is said a FRTS (firm Real-time system) if the task doesn't produce the output within the particular deadline then the output is not useful for the system but there is nothing like disastrous. It is not safety critical but its zero utility if produced after deadline. For example, a video conferencing application, satellite-based surveillance applications etc.[49][116].
3) **Soft RTS:** A system is said a SRTS (soft Real-time system) even though the doesn't produce the output within the particular deadline; the output is still useful for the system but the utility of a result decreases with time after the deadline. If several tasks miss the deadline, then the system performance will be degraded. The example for SRTS are multimedia applications, railway reservation system, web browsing, all interactive applications etc.[15][47].

1.2 Features of RTOS

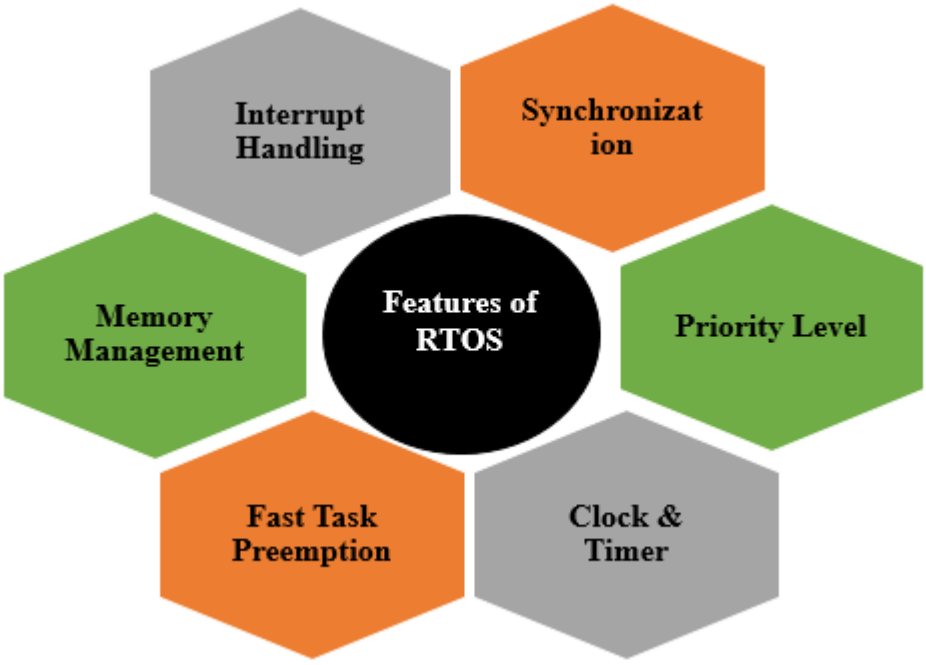


FIGURE 1.2: Features of RTOS [52] [91]

- **Synchronization:**

  Synchronization of tasks is the crucial part for the real-time system as we are working on multitasking system and resources shared between all the tasks with mutual

exclusion. In this, the inter-process communication and synchronization policies are compulsory for working in multitasking environment [17] [60] [103].

- **Interrupt Handling:**
  In RTS, an Interrupt Service Routine (ISR) is very important to handle the interrupts. The interrupt latency can be defined as the time interval from interrupt generation to the execution of the respective ISR [58].

- **Timers and clocks:**
  The services related to timers and clocks along with suitable resolution are the important part of any RTOS as meeting the deadline is the first requirement of RTS and for that the clock and timer facilities with appropriate determination are significant aspects of each real-time OS [58] [85].

- **Priority Levels for Real-Time Task:**
  Every RTOS contains the facility of real-time priority levels so that the jobs can meet the deadlines. The priority levels in each RTOS are assigned by the developers and can't be modified by the operating system [63].

- **Fast Task Preemption:**
  Whenever higher priority job arrives in the RTS, the CPU must be taken back from the lower priority currently running task and it must be assigned to the higher priority task for efficient work of the RTS. Always the task with highest priority must be scheduled first so that task can meet its deadline which is the requirement of any RTS [39] [58].

- **Memory Management:**
  The applications which are having gigantic size may be offered the virtual memory in real-time operating system [71]. RTOS are not responsible to fulfill only the real-time task's demand but also it has to assist the demands of non-real-time applications like browsers, different types of editors, etc. The RTOS are having very small size memory which includes only the required functionalities for the applications [93][95][96].

1.3 The Real-Time System Characteristics

The properties of RTS are as following [12] [81] [87]

- **Timeliness:**

  Output can't be called correct if it is only logically correct but it must be produced within predefined time interval. To meet the deadline, there must be some kernel mechanism for time management and to handle the task sets with the timing constraints which should address its criticality.

- **Predictability:**

  The system should be capable to guess the afterward effects for every decision taken regarding scheduling to accomplish an expected output. For any safety critical application, before pushing system into action; all the time constraints must be analyzed in advance. If any task can't provide guarantee of completion before its timing limits then the system must be informed this reality in prior as the exception handling can be planned by any other substitute activities.

- **Efficiency:**

  The management of all the allotted resources must be done by the OS very efficiently to get a required output as the small devices along with unadorned constraints like weight, computational power, space, memory etc. will be embedded in almost all the RTS.

- **Robustness:**

  In the uttermost load conditions, RTS must not fall down so the RTS will be designed in such a way that it can manage all the probable workload situations. Overload management and adaptation behavior are important characteristics to handle variable resource needs and higher load variations in the systems.

- **Fault tolerance:**

  The critical modules which are critical in the real-time system must be fault tolerant as the failure of single component (hardware or software) should not lead to the whole system towards the crash.

- **Easy to Maintain:**
  The structure of a RTS must be modular structure to make the variations in the system easier.

1.4 Applications of Real-Time Systems

Some examples of applications for each area are shown as following [73] [74] [92].

- **Industrial Applications:** The examples are automated Car Assembly plant.

- **Medical Applications:** The example is a Robot which is used in recovery of Displaced Radioactive Material to the patients.

- **Peripheral Equipment:** Different peripheral equipment which are attached to a computer have embedded systems on them. For example, laser printer and the digital cameras.

- **Automotive and Transportation:** The example is the multi-point fuel Injection (MPFI) system. Embedded systems are used heavily in transportation. Nowadays, all the cars are road worthy and they use an MPFI system because it reduces the pollution. The principle idea behind MPFI system is that when a computer controls the exact fuel quantity and the time of injection of the fuel, the car will run at maximum efficiency.

- **Telecommunication Application:** In the telecommunication applications, important example is the cellular system (mobile phones). The base stations receive signal from the mobile phone and give signal to the mobile phone. They use real-time operating systems and handle many tasks at any time. For example, handling SMS messages, call establishment, keeping track of billing, hand off to other base stations etc.

- **Aerospace:** The example is the air traffic controller in the aircraft. Most of the aerospace applications are essentially have computers on board, because it reduces the interference of the pilot. Sometimes, even without pilot they can fly the aircraft.

- **Consumer Electronics:** Consumer electronics items like phones, digital cameras,

camcorders are embedded RTS. In defense application, the embedded system is used heavily including the wireless sensor networks.

- **Internet and Multimedia Applications:** The internet is also use various embedded systems; the example is the router. In Multimedia applications, such as video conferencing, it involves handling the signals, compressing, transmitting, receiving it on the other side in real-time. In videoconferencing, there is frame to frame translation, so, if the time delay is not proper, it usually results in glitches and we say that video is not working properly, so these are also real-time systems.

1.5 Problem Statement

To design and develop a generalized scheduler for multiprocessor soft RTOS using the concept of processor affinity with dynamic priority assignment which can increase the overall schedulability, maximize the CPU_Utilization, reduce the Tardiness and reduce the Deadline miss ratio using the flexible mechanism for the migration which can be implemented with task shifting using priority reprioritization mechanism.

1.6 Research Objectives

The scheduling in real-time systems for multiprocessor environment is the important area of research which is still unconquered as real-time applications enforces various necessities on scheduler than the general purpose apps. In The RTS, the important need is the investigation of scheduling techniques that is used for deciding whether the real-time application's execution time will be confined to meet the timing restrictions (deadline). The key objectives of this research work is that to design the generalized multiprocessor scheduler using the concept of processor affinity and to provide flexible migration policy with priority reprioritization which optimize the different parameters of the scheduling approach for the multiprocessor soft RTS and they are defined as the following:

- **Improve the CPU_Utilization:**

FP along with partitioned approach is not able to utilize more % of the offered resource. Whereas in some cases the global along with EDF priority approach is

capable of utilizing resources up to 100% but it is having very high overheads are typically prohibitive [70][78]. The approaches like partitioned and global described in literature are application based so to design a processor affinity based generalized scheduling algorithm which can provide advantages of both the approaches and improve the CPU_Utilization is required.

- **Improve the Schedulability:**

  The practical study has proved that a huge breach amongst the best suitable schedulability condition and presently existing for partitioning P-FP and global G-FP scheduling [9] [39] [82]. Also the approaches discussed in literature are application based. At the present scenario the affinity concept is used by few RTOS like VxWorks,QNX and LynxOS [82]. By keeping the constraints on migration of real-time tasks in the scheduling approach may improve the performance of the cache, balance the load but the main disadvantage of it is that there will be the decrement in overall schedulability [35][76]. So a new generalized multiprocessor scheduler based on affinity should be designed to improve the overall system schedulability.

- **Analysis of Switching Overhead:**

  The structural design of cache is having a large effect on the migration cost whether it is at the task level or job level. Current investigational applications [45] [111] on multi-processor environment presented that the migrations expenses, switching overhead and run-queue handling are the main issues whenever designing any multiprocessor scheduler. Such overheads should be taken into account whenever the research related to scheduling approaches and analysis is done. So it should be considered as an important parameter and must be analyzed the no. of context switches.

- **Reduce Deadline Miss Ratio:**

  In all the real-time applications, meeting the deadline is the prime goal [57] [60]. However, because of varying workload, meeting the deadline of all the task is difficult and changing priority of the applications [57] [81] but more no. of tasks meeting the deadline then the deadline miss ratio is low. So, analysis of the deadline miss ratio is the important parameter which should be reduced for better system performance.

- **Reduce Tardiness:**

  In the SRTS, if the result is produced after deadline then it has less utilization rather than zero like in hard real-time system [50] [108]. The system performance is degraded as much as more time is taken by a task after the deadline [50] [87]. So, the tardiness is the important parameter to be reduced in soft real-time OS for better system performance.

1.7 Research Contributions

In this research, different types of multiprocessor soft real-time schedulers are implemented and compared with different approaches. The details of each Multi-processor scheduling approaches are given below:

- **Implementation of P-FP-APA scheduler with Processor Affinity and No Migration:** The P-FP is the partitioning scheduling approach for processor selection in multiprocessor real-time system in that each task is fixed to the particular processor with no migration. This conventional approach P-FP mentioned in the literature is application based and not implemented using processor affinity in soft real-time system. So the first contribution is the implementation of partitioned approach (P-FP-APA) along with fixed priority algorithm using processor affinity in soft real-time system. Here the fixed priority means every jobs of a task will be allotted the equal priorities which will be the same as that task's priority.

- **Implementation of G-JLFP-APA scheduler with Processor Affinity and Full migration:** The G-JLFP (G-EDF) is the global scheduling approach for processor selection in multiprocessor real-time system in that any job can be scheduled on any available CPU with full migration. This conventional approach G-JLFP mentioned in the literature is application based and not implemented using processor affinity in soft real-time system. So in this research G-JLFP-APA is the implemented along with JLFP (job-level-fixed-priority) approach using processor affinity in soft real-time system. Here the JLFP assignment means the different jobs of a same task can be assigned dissimilar priority which will not be changed throughout its lifetime.

- **Implementation of proposed scheduling approach PrASWM:** The PrASWM

(Processor Affinity based Scheduler without flexible migration) is the proposed scheduling approach in that the attention is given on processor selection policy. The PrASWM is using affinity concept for processor selection and migration of a task is possible within its affinity as per the conventional migration approach based on its priority. In PrASWM, the processor selection policy is combined with JLDP (Job Level Dynamic Priority Assignment) priority assignment algorithm which improves the results as compared to P-FP-APA and G-JLFP-APA. So, PrASWM is the Scheduler with Processor Affinity and without Flexible Migration Policy along with JLDP.

- **Designed and Implemented Multiprocessor soft real-time scheduler PrAMS (Processor Affinity based Multi-processor Scheduler) along with the three approaches:**

  Any multiprocessor algorithm addresses three different problems as mentioned below:

  i. **Processor Selection Policy:** In this research the proposed scheduler PrAMS is using arbitrary processor affinity concept for processor selection which gives more flexible generalized scheduling approach rather than the application based conventional approaches.

  ii. **Process Selection (priority assignment)**: In PrAMS for the priority assignment JLDP (Job-Level-Dynamic-Priority) is used which is fully dynamic approach. Here, in JLDP a task's jobs will be allocated the various priorities during job's lifetime based on its deadline.

  iii. **Flexible Task Migration Policy:** Instead of restricted migration, the PrAMS provides flexible migration policy by task shifting with the help of priority reprioritization mechanism which solves the priority inversion problem, improve the overall system's schedulability, decrease the deadline-miss-ratio, decreases the tardiness and maximizes the CPU_Utilization.

1.8 Organization of Thesis

The next thesis chapters are organized into six chapters as shown below:

Chapter 2: Literature Review: This chapter includes basic terminologies related to scheduling and explores the static and dynamic priority algorithms for uniprocessor and multiprocessor algorithms provided by RTOS along with their mechanisms in section 2.1.

After that this chapter includes the survey of prior work related to this research in section 2.2. After that Section 2.3 describes performance measure parameters. Furthermore, after the studying all the existing approaches in detail this chapter presents the summary of literature review in section 2.4.

Chapter 3: PrASWM a Real-Time Multiprocessor Scheduling Approach: In this chapter, the focus is to provide and implement an effective generalized processor affinity based processor selection approach PrASWM for multiprocessor soft real-time system without using flexible migration policy. Furthermore the comparative analysis is done with P-FP-APA and G-JLFP-APA which are also implemented using the processor affinity concept.

Chapter 4: A Multiprocessor Scheduler PrAMS for Soft Real-Time systems: In this chapter, the focus is to provide an effective generalized scheduler for SRTS in multiprocessor environment which uses static processor affinity assignment for processor selection, Priority assignment using JLDP policy along with flexible migration policy using priority reprioritization concept.

Chapter 5: LITMUS^{RT} and Experimental Results Analysis: Firstly, this chapter introduces a simulator for scheduling algorithm called LITMUS^{RT}. LITMUS^{RT} is the Linux extension with the emphasis on scheduling in real-time and synchronization for multiprocessor environment. The next sections discusses the task set generation theory and experimental setup taken into consideration to take the results. After that the section 5.4 presents results and comparative analysis of processor affinity based scheduling algorithm PrAMS with existing conventional scheduling approaches tested for the 100 task sets of random size and more no. of processors along with five parameters that is Tardiness, Schedulability, CPU_utilization, No. of context switches and Deadline miss ratio.

Chapter 6: Conclusions and Future Scope: This chapter offers the conclusions observed based on the outcomes of this research work. Furthermore, it presents the several ways for additional possible research in the future also.

Chapter 2:
Literature Review

This chapter presents the background theory and prior work done in real-time scheduling. Here, the basic theory related to scheduling in real-time is discussed. Afterward, the analysis of different types of real-time scheduling approaches for uniprocessor as well as for multiprocessor and related schedulability analysis is done which provides the base for this research work. Furthermore the prior work done by various researchers and performance measure parameters will be discussed. At last the summary of the literature survey along with limitations of existing approaches is presented.

2.1 Background Theory

This section will present the basic theory and various terminologies regarding scheduling in real-time system

2.1.1 Real-Time Task Model

In any real-time systems, the simultaneous computational events are done for that the rightness of result doesn't depend on logical output only but it also depends on the time at which it is produced. The event is said to be tasks and during its life cycle every task can have a job sequence .Thus a sequential unit of work is called the job. Time constraints for the task is bounded by the deadlines that indicates the task execution must be completed before given time. This section presents the real-time task model, definition and assumptions that had been taken to perform the analysis. We assume the classical sporadic task model followed by the task [37] [92]. Here it is assumed that the real-time system contains n real-time tasks $T_1, T_2, \ldots, T_n$, that forms a taskset $\boldsymbol{\tau} = \{ \mathbf{T_1}, \mathbf{T_2}, \ldots\ldots \mathbf{T_n} \}$.Every task spawns a sequence of m jobs an instance of task $T_{i.1}, T_{i.2}, T_{i.3}, \ldots. T_{i.m}$.

Absolute deadline (d_{ab}): The duration between the time instance 0 and the real instance of time where the deadline occurs is called the absolute deadline $\boldsymbol{d_{ab}}$ [18].

Relative deadline (d_i): Relative deadline $\boldsymbol{d_i}$ is the time duration between the release time of the task to the instant at which the actual deadline occurs [49] [92]. FIGURE 2.1 represents

the difference between absolute deadline and relative deadline.

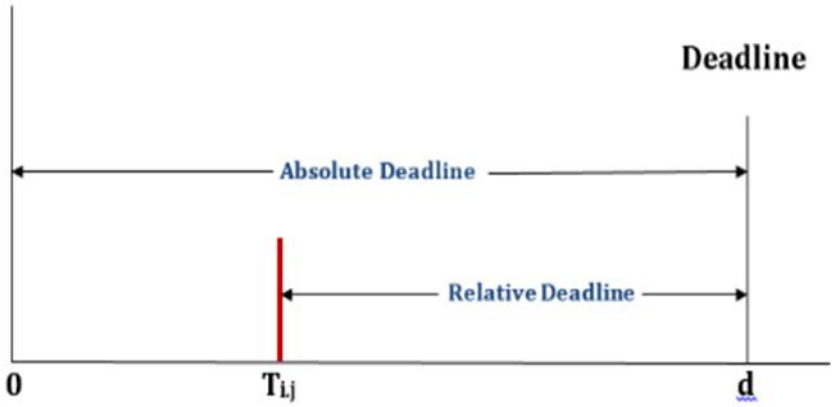


FIGURE 2.1: Absolute Deadline and Relative Deadline [37][49][92]

Arrival time(t_a): It is the time instant at which the job comes in the ready to run state due to the occurrence of an event/condition [20] [32].

Start time(t_s): It is the time instant at which the task's first job starts its execution for the first time [88] [92].

Computation time(t_c): It is the duration in which the task require the processor to complete the execution of the job without interruption [64] [92].

Finish time/Completion time(t_f): It is the time instant at which the task produces the output [36] [66] [115].

Response time(t_r): It is the duration between task arrivel time to the task completion time:$t_r = t_f - t_a$ [18] [115].

Worst-case execution time (C_i): The worst-case execution time is the maximum computation time required amongst all the jobs of task T_i: $C_i = \max(C_i, k)$. It can be shortened with WCET [1] [91] [107].

Lateness(L_i): It is the difference between job completion and its deadline: $L_i = t_f - d_i$. If the job finishes execution before its deadline then lateness is negative but when the job finishes the execution after the deadline then its lateness is greater than zero and this case is called a deadline miss event [17] [92] [100].

Phase for a periodic task (Φ_i): It is the time from 0 till the occurrence of the first instance of the task which is denoted by Φ. Example: The track correction task starts at 200ms, after the launch of the rocket, periodically recurs at every 30ms, each instance of the task requires a processing time 6ms and its relative deadline is 50ms. So it can be represented by (Φ_i, p_i, C_i, d_i) = (200ms, 30ms, 6ms, 50ms) [80] [94].

2.1.2 Multiprocessor System Classification

From the scheduling point of view, the multiprocessor system is categorized into three classes [66] [80]:

i. **Heterogeneous Multi-Processor system**: The architecture of all processors will be different, so the execution rate of the task depends on both the type of processor and the task. In addition to that each task cannot be executed on all types of processors [18] [80].

ii. **Homogeneous Multi-Processor system**: Here, all the processors have identical structure so the execution rate of all tasks will be the equal on all the processors [27] [80].

iii. **Uniform Multi-Processor system**: In this environment, the execution rate of the task depends on the processor speed only. So the processor of speed 2 can have exactly double the rate of a processor of speed 1 [49] [80].

2.1.3 Classification of Tasks

i. **Periodic Task:** The Periodic Task appears after a fixed amount of time duration and recurs according to a timer. Majority all the real-time tasks are periodic; Example: Temperature sensor sense at fixed interval and give input to the RTOS [83][92].

ii. **Aperiodic Task:** Aperiodic Task is one that recurs at random instance and they are having soft deadlines; Example: any interactive commands issued can be handled by aperiodic task like keyboard presses, mouse movements etc. [43] [92].

iii. **Sporadic Task:** A sporadic task Recur randomly but they are having hard real-time deadlines. An aperiodic task is in many ways similar to Sporadic Task. Two or more instances of an aperiodic task might take place at the same time instance; Example: In a factory, the task handles the fire conditions [20] [92] [110].

2.1.4 Scheduling in Real-Time Systems

Majority of real-time systems handle concurrent tasks. Concurrent task are not easily handled by linear software design paradigm like super loop. To manage real-time system we need to work with environment which provides us concurrent task handling and abstraction for development of control system. Operating system has this functionalities to provide multitasking and abstraction. Operating system is called hard real-time OS if it can deal with the set of tasks which are hard in nature [23] [36] [78]. The real-time system with number of resources and all these resources are shared between more than one processes running simultaneously for meeting the several objectives.

To take the decision for running various tasks on currently available processor in which order is called the scheduling technique and the portion of operating system who is taking this decision is said to be a scheduler [6] [19] [67]. The functionalities related to scheduling like time-management routines, expectedness of interrupt falls under very important features category of real-time operating systems [89] [104]. The fundamental design issue in any multi-tasking real-time OS is the scheduling. The scheduling is a decision making process that accomplishes the supply of various resources to various tasks during their execution based on their demand to achieve necessary outputs [11]. To meet the dead-line is the important task in any real-time application and RTOS is helping them in meeting their deadline with the help of different scheduling policies [4] [113].

2.1.5 Classification of Scheduling Approaches

Currently, the multi-tasking computer systems are used by us. In the multi-tasking system we can execute more than one task simultaneously. There are number of resources in our computer system that is either hardware or software. All the currently present tasks into the systems will share these resources. Which task will get which resource in which duration is the important decision in any computer system [60] [76] [115]. The part of OS which is responsible to take decision regarding allocation and De-allocation of resources during its execution is called scheduler [25] [78]. Scheduling is the decision making process which performs the dissemination of different resources to various tasks as per their requirement. So a critical design issue in any operating system is to design an efficient scheduling algorithm. The scheduling approaches are designed with aim to balance the load, maximize resource utilization, minimize Tardiness, minimize deadline miss ratio, minimize no. of

context switches and maximize schedulability [25] [119]. The scheduling approaches designed for real-time systems so far can be categorized into two different classes based on the number of processor a real-time system contains.

2.1.5.1 Categorization with User's Context

From the user's point of view the scheduling approaches can be classified as the following:

i. **Interactive scheduling approach:**

In this approach the scheduling of the task can be done with interactive mechanisms and hence the scheduling approaches using for this is said an iterative scheduling algorithm [69][78]. The examples for this approach are SRTN, Round-Robin algorithm and lottery scheduling.

ii. **Batch scheduling approach:**

The queue is containing certain similar jobs which is called the batch. In this approach the jobs will be scheduled batch-wise. The algorithms used for this approach are said the batch scheduling algorithm. The examples of this batch scheduling are the FCFS, SJF, and HRN [78].

iii. **Real-Time scheduling approach:**

In this approach the rightness of the result doesn't relays only on the logical rightness but it must be produced within the predefined time called the deadline. The algorithms used for this approach are called the Real-Time scheduling approaches which will schedule the Real-Time tasks [25]. The examples of the real-time approaches for scheduling are RM (Rate Monotonic), EDF (Earliest Deadline First), DM (Deadline Monotonic) and LLF (Least Laxity First) [22][25].

2.1.5.2 Scheduling Location based Classification

The location based classification can be done like they are scheduled from a centralized location or from distributed locations [69].

i. **Centralized Scheduling approach:**

There is one central system which is responsible to take decisions related to the scheduling [63] [69]. The whole distributed system may fail if the central system will

go down. In this approach, central system needs to collect latest information from each node periodically it causes the communication bottleneck.

ii. Distributed Scheduling approach:

In this approach, scheduling decisions will be taken by different nodes and for that the chaining of different jobs will be maintained into a harmonized workflow. Here, the scheduling decision is taken by numerous nodes not by single centralized server only. This scheduling approach is useful in non-interactive processes in distributed systems [38] [58] [69].

2.1.5.3 Pre-emption based Categorization

The Pre-emption based Categorization depending on that either the pre-emption will be permitted or not by a specific scheduling approach after assigning processor to a task.

i. Preemptive Scheduling approach:

In Preemptive scheduling , A scheduler will be permitted to take back a CPU from the task which is in execution if a process with high priority will be arrived or task has completed its time slice and algorithms used for this approach are said as pre-emptive scheduling algorithms [60] [78]. All the scheduling approaches for real-time systems and the RTOS' kernel are pre-emptive in nature. The examples of preemptive scheduling are Round-Robin and the SRTN.

ii. Non-Preemptive Scheduling approach:

In a Non-Preemptive scheduling approach, the scheduler will not be permitted to take processor forcefully back from the running task till either the task will release the CPU voluntarily or the task's execution gets completed [26] [78]. The examples of Non-preemptive scheduling are the FCFS and SJF.

2.1.6 Real-Time Scheduling in Uniprocessor System

In a single processor system, all the currently ready-to-run processes can be scheduled on a standalone processor one by one in a particular order so that maximum tasks can be completed before their deadline. As shown in FIGURE 2.2 the scheduling approaches can be divided into two classes: 1) Static approach 2) Dynamic approach. In the first one, the task priorities are kept static which is assigned at the time of algorithm designing time

whereas in second approach the priorities of the task are allocated during the execution time depending on various factors and also it may be altered during its lifetime. Further the dynamic scheduling can be divided into two categories fixed priority and dynamic priority. The sample for dynamic scheduling along with fixed priority method is the Rate Monotonic (RM) and Deadline Monotonic (DM) approach [2] [33] [79] whereas the examples of for the dynamic scheduling method with dynamic priority are the Earliest Deadline First (EDF), Least Laxity First (LLF) [79] [118].

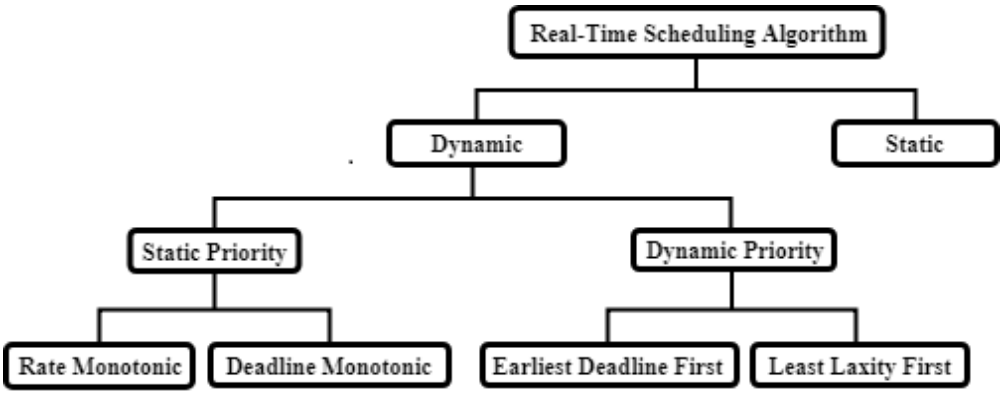


FIGURE 2.2: Classification of Scheduling Algorithms

2.1.6.1 Dynamic Real-Time Scheduling Approaches using Fixed Priorities

i. Rate Monotonic (RM) Algorithm [11] [71]

For the uniprocessor real-time system the Rate Monotonic (RM) is an optimal fixed priority scheduler. It can handle all the processes in the system those are having periodicity in nature and the deadline and the period are equal. This algorithm will assign the fixed priority to each task based on its rate and the rate of a task is the inverse of task's period. Higher the priority of a task if its rate is higher and lower the priority of a task if its rate is lower. For meeting the deadline of a task set with n no. of tasks; the required and mandatory condition to this method is as follow [5]:

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq \mathrm{n}\,(2^{1/n} - 1)$$

Where c_i is the worst-case execution time, p_i is the period of a task

ii. Deadline Monotonic (DM) Algorithm [11] [87]

The important constraint in RM is that it can't be an optimal approach if for any real-time task the deadline of a task is not equal to its period. The deadline Monotonic is

an optimal algorithm for a single processor system using the static priority. The limitation of RM is overcome in DM as deadline monotonic will also schedule all the periodic real-time tasks whose deadline does not equal to task's period. This algorithm will assign a fixed priority to a task which is in proportion to task's relative deadline. The RM can be said as the special case of DM where the task's deadline is equal to its period.

2.1.6.2 Dynamic Real-Time Scheduling Approaches using Dynamic Priority

i. Earliest Deadline First (EDF) Algorithm [87] [118]

The EDF is an optimal uniprocessor real-time scheduling approach which is using dynamic priority. EDF can schedule only those tasks of the system that are periodic in nature by taking the absolute deadline into consideration. The process which is having the shortest deadline will be assigned the maximum priority and that task will get first chance to be scheduled. The necessary condition for task set of n tasks to meet their deadline is as follow [94]:

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$$

Where c_i is the worst-case execution time, p_i is the period of a task

ii. Least Laxity First (LLF) Algorithm [13] [87]

The optimal and efficient approach for the single processor real-time system is the LLF which is a dynamic algorithm with dynamic priority. In this method the process would be assigned a priority based on the task's slack time. The task which is having more laxity will be assigned lower priority and the small laxity task will have higher priority. The Task's LAXITY can be defined as the difference of Deadline and Worst-Case Run Time. The laxity of a task will represent the time by that the process can delay and still run into its deadline. The Laxity is also said as the Slack-Time. Laxity of the Task T_i is:

$$\mathbf{L_i = d_i - C_i}$$

Where,

$\mathbf{d_i}$ is a Task deadline for task T_i,

$\mathbf{C_i}$ is worst-case running time for a Task T_i

2.1.7 Real-Time Scheduling in Multiprocessor System

In the multiprocessor system, more than one ready tasks in the system will be scheduled on different CPUs simultaneously in a particular order so that the maximum tasks can finish their execution within their deadline [96]. So there should be appropriate approach for the selection of process and the processor amongst them. Two different issues addressed in the Multiprocessor scheduling are as follows [32] [103]:

1) **The Processor selection problem**: It represents the task will be executed on which processor that is processor selection [32] [95].

   Processor allocation problem further categories in three classes

   i. **No migration:** The task is fixed on particular processor and it must complete execution on the same processor no further migration will be allowed.
   ii. **Task-level migration:** Various jobs of the same task would be scheduled on different CPUs but every job can only be scheduled on the single processor.
   iii. **Job-level migration:** The migration of a sole job is permitted and that job is able to be scheduled on various processors.

2) **The Priority assignment / The Process selection problem:** It represents the order of execution for particular job of a task with reference to the other jobs of different tasks it should be executed.

   Priority assignment problem further can be classified into [32] [113]

   i. **Fixed task priority:** Every task is having a single static priority which is applicable to all the jobs of that task. The example is Rate Monotonic (RM) scheduler
   ii. **Fixed job priority:** The different jobs of the same process can contain dissimilar priority levels but every job is having its single fixed priority. The example is the Earliest Deadline First (EDF) scheduler.
   iii. **Dynamic priority:** A solo job of a process can be allocated different priority levels during its life-time. The example is Least Laxity First (LLF) scheduler.

   There are various multiprocessor scheduling algorithms designed by the researchers but it can be divided into the two categories [25] [30] [103], any processor selection policy can be selected as per our application and it can be combined with any appropriate process selection algorithm. There are main two categories of the multiprocessor scheduling algorithm as described below:

2.1.7.1 Partitioned Scheduling Approach

In partitioning scheduling, the tasks are partitioned between system processors and every task is allocated a fixed CPU on which it can be scheduled. The task cannot be executed on any other processor even though it is free but wait for the processor which is already fixed for the task. There is no migration is permitted on other processor in partitioned approach. This methodology is modest and easy to implement.it is having very less switching overhead but it may not provide feasible schedule for the given task set [89] [67]. The purpose is that, because of their fixed task allotment and restricted migration, the partitioned approach can simplify the problem of multi-processor real-time system into the uniprocessor ones [12] [67].

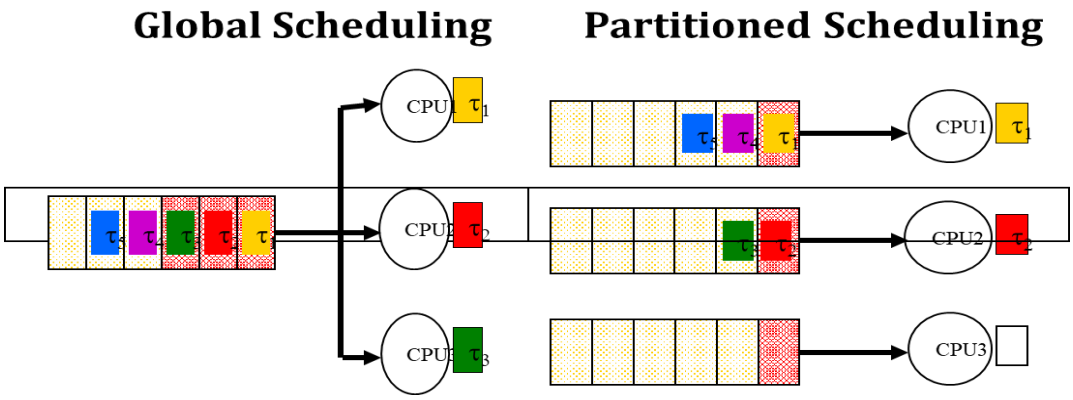


FIGURE 2.3: Difference between two Multiprocessor Scheduling Approaches [87]

Advantages of the Partitioned scheduling as compared to global scheduling:

- If any task exceeds its worst-case execution time then it will affect only those tasks that are scheduled on the same processor [76].
- In this approach the task executes only on a single processor so there is no cost of migration i.e. a task that is running on processor P1 and after some duration if it is pre-empted and resumed on another processor P2 will have to restore its context on processor P2 which will increases the switching overhead [87].
- In this approach, a separate run-queue is used per processor instead of a single global queue. The overheads of operating a single global queue in the large systems can be too much [10] [67].
- The important advantage of a partitioned approach is that once the assignment of the tasks to processors has been done, there can be applied any uniprocessor real-time

scheduling techniques [30].

2.1.7.2 Global Scheduling Approach

The migration of the tasks on various processors in their execution duration is allowed in global scheduling and any task can be executed on any available processor [42] [67]. The global scheduling approach ensures the feasible schedule of a feasible schedule for the given task set but gives considerable switching overheads also due to its unrestricted migration mechanism [67]. The FIGURE 2.3 represents the architecture difference between partitioned approach and the global approach.

Advantages of the Global scheduling as compared to partitioned scheduling:

- If any task completes its execution in less than its worst-case execution time than that processor becomes free so we can schedule any other task on it who wants to execute [26].
- If any task exceeds its worst-case execution time then also the lower probability of missing the deadline as it can be migrated on another available processor [84].
- For open systems, Global scheduling approach is more suitable as there is no requirement of running the load balancing / task allocation algorithms at the time of task set changes [69] [72].

2.1.7.3 Clustered Scheduling Approach

To take the advantages of both the existing approaches a hybrid clustered multiprocessor scheduling approach has been designed [39] [44]. The advantage of the partitioned approach is that it is having minimum overhead as all the tasks are fixed on a specific processor. Whereas, the benefit of the global approach is that it provides more optimal schedule due to its migration proficiency. This approach generally creates a clusters of processors. All the tasks of a system are assigned to a particular cluster like partitioned scheduling and then task is scheduled like global scheduling within that cluster which is already assigned to it [76] [99].

2.2 Literature Survey

Now a days, the numerous real-time applications are using multiprocessor systems extensively. A semi-partitioned algorithm for multiprocessor soft real-time system for

optimizing QoS has been presented by Behnaz Sanati et al. [108] whose aim is to decrease the number of misses. A semi partitioned scheduling algorithm is the extension of partitioned scheduling approach with allowing migration to very few tasks of a task set thus it reduces the number of context switching. The greedy approach and load balancing two dynamic approximation approaches are used in proposed approach with an online semi partitioning policy in the soft real-time system with periodic task set. The aim is to reduce deadline miss ratio for improving the Quality of Service.

The GPEDF scheduling approach presented by Li, Q. & Ba, W et. al.[88] for SRT. At the time of priority assignment, in some cases the task set wants priority levels beyond the capacity of the system. This algorithm GPEDF addressed that issue by allocating the per group priority in that the group with the shortest deadline can have the maximum priority and it will be scheduled first. It is compared with conventional approaches EDF and the g-EDF. The outcomes proved that the GPEDF has less number of context switches, the short response time, reduced deadline miss ratio with very few priority levels.

A multi-processor soft real-time Scheduler has been published by Hitham Alhussian et al. [3] has published in that complete relaxation given on the fairness rule. In the multiprocessor environment, Most of the present real-time scheduling approaches with optimal schedulability follow the fairness rule by providing no. of migrations and job pre-emptions that greatly affects the expectedness of the approach. This approach uses the global approach and provides fairness rule relaxation so that the no. of context switches and deadline miss ratio can be reduced.

The hybrid scheduler for weakly hard real-time system for multiprocessor environment has been presented by Habibah Ismail et.al.[72] that has less context switching overhead. The goal is to increase the global schedulability of a system by giving the hybrid approach with an effective assignment of jobs to the processor for taking benefit of both the approaches like partitioned and the global scheduling.

The EDF Scheduling of Uniprocessor Systems along with new pre-emption policy has been presented by Jinkyu Lee et. al. [83] for meeting the deadline. EDF is the broadly analysed, efficient and simple algorithm for single processor real-time system but the downside is extra overhead due to the pre-emption with the existing pre-emption policies. This paper presented the new and more effective pre-emption policy which decreases the

deadline miss ratio and increases the schedulability of EDF approach up to 7.4%. This policy is designed for the Uniprocessor system only. This paper had proved that the proposed approach (CP-EDF) is giving extra schedulable task sets as compared to EDF's currently existing pre-emption policies.

R. Kalpana et. al. [74] has presented DC-gEDF policy for soft real-time system with zero context switching over head as it uses the non-pre-emptive policy. In this approach, based on task's domain description and its deadline requirements the clusters of the tasks will be created. After that the tasks will be scheduled which belongs to that group. The performance analysis is done based on the deadline miss ratio of G-EDF and DC-gEDG. It is observed that that the improvement deadline miss ratio using proposed algorithm Domain-Cluster-gEDF.

A SRT scheduling algorithm CP-EDF (Controlled Pre-emptive EDF) has been presented by Keerthanaa C et. al. [76] in that restricted the number of pre-emptions for minimizing the number of context switches. The FP-EDF and Non-Preemptive EDF approaches compared with CP-EDF and it is proved that the CP-EDF gives improved results than existing two by reducing context switches, deadline miss ratio but it also decreases the CPU-Utilization.

A fault-tolerant mechanism for real-time tasks scheduling in multiprocessor real-time system with dynamic approach has been presented by Mohammad H. Mottaghi et. al. [101]. The algorithm in this paper is designed to increase the schedulability and to reduce the all over execution time of running jobs. The redundancy technique is used in this paper for fault tolerant: 1) Redundancy depending on Hardware by keeping the several replicas of a job on dissimilar hardware 2) Redundancy depending on Software with rollback recovery. This paper has used task utilization to define the criticality of every task. Based on its criticality the tasks are divided into two groups; the critical task group and non-critical task group. The categories of processes either it is critical or Non-critical will be used for the fault tolerant policy selection dynamically at the time of its execution like non-critical tasks will be scheduled on a standalone machine and rollback recovery by the check pointing and the replicas of critical processes will be kept on the dissimilar processors for improving the possibility of task execution before its deadline in case of any fault in the system.

The schedulability study of Linux scheduler based on APA concept in hard real-time

system has been done by A. Gujarati et. al. [67] that mainly focuses on processor selection issue not on the process selection. They had formulated that the conventional approaches like the global, clustered, and partitioned scheduling can be strictly dominated by the APA-based scheduling and the paper proved that the processor affinities are advantageous and it should be analysed properly. To design the enhanced analysis technique for scheduling using APA and strong scheduling approach by providing flexible migration policy can be an important area for the research.

The scheduling of soft real-time sporadic task systems under global EDF on an identical multiprocessor has been published by U.M.C. Devi et. al. [51]. In the past research, the main focus was on hard real-time systems for global EDF but for the soft real-time systems the scenario is different as it tolerates bounded tardiness. The tardiness bounds for pre-emptive and non pre-emptive global EDF in the multiprocessor system has been derived in this paper. It is proved that global EDF performs better than partitioned EDF for multiprocessor-based soft real-time systems. If any task completes a little bit late in soft real-time system then the result will not likely to be disastrous, but the tardiness must be bounded.

A SRT scheduler G–EDF-like (GEL) has been presented by Jeremy P. Erickson [56] that discover the broader category of G-EDF-like (GEL) schedulers which is having equal overhead characteristics to G-EDF. They had shown that the selection of GEL schedulers gives the better tardiness than G-EDF.

In the real-time system, mostly the researchers had focused on priority selection algorithms but whenever we are working on multiprocessor environment the processor selection issue must be addressed as it creates a huge effect on the system performance. A. Gujarati et. al. [68] had designed Processor Affinity based scheduling approach for multiprocessor hard Real-Time system which increases the overall schedulability of the system and mainly focuses on processor selection problem.

In the big data era and 5G era, many tasks have higher and higher requirement for the latency and data, traditional cloud computing paradigm is gradually unable to handle such real-time scenarios with large number of tasks, tremendous data volume and high latency requirements. In order to solve these problems, mobile edge computing has become the focus of attention. However, although low latency is a major feature of mobile edge computing, some real-time tasks still cannot be completed on time due to the limitation of network and

computing resources. To reduce the penalty of tardiness of tasks Xi Chen et. al. [56] had proposed a SRT heuristic algorithm based on the urgency of deadline of tasks to reduce the loss caused by task timeout. They have taken the mobile edge computing system as a soft real-time system and discussed how to assign the task to a server and how to schedule the task on the server.

Shiva Nejati et. al. [102] had analysed the CPU_Utilization limitations and presented a framework which is used to maximize the CPU_Utilization. They proposed a novel model for investigating the CPU utilization. The CPU_Utilization formulated by them as a constraint optimization problem and had given an implementation of their approach using optimization tool.

The problem of multiprogramming scheduling for Uniprocessor in SRTS has been studied by C.L. LIU et. al. [90] which shows that for large task sets the CPU_Utilization has an upper bound that is as low as 70% in optimum fixed priority scheduler. They concluded that the full CPU_utilization could be achieved by dynamic priority assignment based on urgency of their current deadlines and also it can be analyzed for multi-processor real-time system.

Davis RI, Burns A et. al. [47] has done the survey for hard real-time schedulers and schedulability investigation techniques for homogeneous multi-processor environment which studied the key results in the real-time scheduling since the 1960s to 2009. A taxonomy of the various scheduling approaches and consideration of different performance measure parameters are provided with comparison. A comprehensive survey has been given which covers partitioned, global, and hybrid scheduling approaches. The open issues, likely research directions and key research challenges are given by the review.

Baker TP, Baruah SK et. al. [17] has presented the schedulability analysis for multiprocessor sporadic task system on identical processors. In this paper the proposed approach investigated the related notion of schedulability and a notion that is called feasibility and proved that the discrete-time schedules are as powerful as continuous-time schedules.

Anderson JH et. al. [8] has published an EDF based multi-processor scheduler for soft real-time systems has been published by which proposed techniques and heuristics for reducing the tardiness. As per the information this paper is the first which investigating EDF

scheduling in multiprocessor system for the soft real-time system. The esurience of bounded tardiness is the important contribution of a novel EDF based strategy. Here, the restricted per_task_utilizations but there is no need for restricting the overall utilization.

Bado B et. al. [15] has presented a semi-partitioning algorithm on n identical processors for in parallel soft real-time system using EDF. In this approach a series of phases is used to define every periodic task which will be possibly parallelized. The semi-partitioned technique is used with migrations at native deadlines assigned to every phase. They considered the phase parallelism which is extension of the common job parallelism. The well-known uni-processor EDF feasibility condition for asynchronous periodic tasks has been taken into consideration for deciding the schedulability of a Multi-Thread task.

The Semi-partitioned scheduling of sporadic task systems on multiprocessors with arbitrary deadlines on identical multiprocessor environment has been presented by Kato S, Yamasaki N, Ishikawa Y et. al.[75] in that almost all the tasks will be assigned to particular fixed processors except very few tasks that can migrate on any available processor in the system. The priority assignment policy is EDF along with less number of context switches.

Dorin F et. al. [53] has published the semi-partitioning hard real-time scheduler with constrained migrations for identical multiprocessor environment which gives feasible substitute between the two extremes global approach and partitioned approach for periodic tasks. The proposed approach use the idea of semi-partitioning scheduling which provides constrained migrations by not allowing jobs to migrate but two distinct jobs of a task are being allotted on various processors.

The EDF scheduling approach for heterogeneous multiprocessor environment presented by Funk SH [61]. Mainly this research expands the scheduling for different types of systems available for real-time applications. It shows the test cases which proved that the tasks are meeting the deadlines whenever scheduled on heterogeneous multiprocessor system using EDF scheduling algorithm. He has presented the schedulability tests which exists for the Earliest Deadline First (EDF) scheduling algorithm in heterogeneous multiprocessor environment along with various migration policies like full migration, No migration and restricted migration.

The feasibility analysis for multi-processor SRT recurrent task systems using specified processor affinities has been presented by Baruah SK, Brandenburg BB et. al [91] that raises

the question of finding the task system that may be implemented to always meet all deadlines, with affinity mask constraints. They proposed an algorithm which derived the answer of this question effectively that the runtime is in polynomial.

Foong A et. al. [69] had done a comprehensive study to show the effect of processor affinity on network performance and they demonstrated that affinitization has a substantial effect on protocol processing efficiency and with the different affinitization, the performance bottleneck of the network receive process can be changed significantly.

The Optimal virtual cluster-based multiprocessor soft real-time scheduling with constrained deadline on multiprocessor platforms has been proposed by Easwaran A, Shin I, Lee I et. al [54] which gives more general approach called cluster-based scheduling to take advantages of both the conventional approaches partitioned and global scheduling. In the proposed algorithm each task instance will be assigned to a specific cluster of CPUs and it may migrate inside that cluster. They have given research direction for cluster scheduling with the goal of improving the CPU_Utilization bounds.

A heuristic algorithm based on the earliness of deadline of tasks has been proposed by Xi Chen et. al. [41] to decrease the penalty of tardiness of tasks. The mobile edge computing system is taken into consideration as a soft real-time system and shows that how the task can be assigned and scheduled to a server. The performance measure parameters which addressed here are deadlines miss ratio, throughput, and CPU_Utilization and Memory_Utilization.

2.3 Performance Measure Parameters

Performance of multiprocessor real-time scheduler can be measured mostly by schedulability and deadline miss ratio and also using the measures like tardiness, no. of context switches, CPU_Utilization. Each of them is discussed in the following sections.

2.3.1 Tardiness(ns)

The Tardiness is defined as " After the deadline, how long a job has taken to complete its work" [50]. In other words, tardiness specifies the time amount for that a task has missed its deadline. The tardiness is used to measure the delay in task execution completion before

given deadline, so it is very important performance measure parameter for the any SRTS as into SRTS the usability of the result depends on the time a particular task has taken after deadline. As delay increases it degrades the soft real-time system performance. We can define tardiness with respect to Lateness which is completion time minus deadline [41][51]; if lateness is positive then it is called tardiness and if lateness is negative then it is called earliness. The Tardiness formula is given as following :

$$\textbf{Tardiness}(T) = \max(0, t_f - d_i)$$

Where $T \geq 0$

t_f = A point of time on which task has completed the execution,

d_i = The deadline of the task T_i

2.3.2 CPU_Utilization (%)

The Utilization of CPU can be defined as "The sum of work load handled by a particular processor" [114]. It is used to estimate system performance especially whenever task scheduling is concerned. CPU_Utilization depends on the type of computing task because some tasks are CPU_Bound which requires almost all the time CPU during its execution but I/O_Bound tasks require CPU for less amount of time. Another name of CPU time is the processing time which is the time amount taken by the processor for executing the task instructions. The clock ticks or milliseconds are used to measures the CPU time. CPU_Utilization presents the load on a CPU in the percentage and based on that some actions can be taken into the system for increasing the utilization. The CPU_utilization formula is given as following:

$$\text{U}=\frac{R}{C} \quad \text{OR} \quad \text{CPU\_Utilization} = \frac{\text{BUSY TIME}}{(\text{BUSY TIME}+\text{IDLE TIME})} \times 100$$

Where U = CPU_Utilization

R = BUSY TIME (Actual Time taken for processing)

C = BUSY TIME + IDLE TIME (Total capacity of a processor)

2.3.3 Schedulability(%)

The important performance measure parameter for HRTS or SRTS is its capability of

finding the feasible schedule for the given task set if any such schedules are in existance [90]. The Schedulability is defined as " the number of tasks from the task set executed sucessfully out of total tasks in a taskset" [8][17]. Here it is considered that the tasks T_{CS} of taskset whose single job has not missed the deadline. In other words it specifies the % number of tasks schedulable with particular scheduler. The schedulability is used to measure the success rate of a system so it is very important performance measure parameter for the real-time system. As the schedulability increases it improves the real-time system performence. The Schedulability formula is given by following equation:

$$\text{Schedulability of a Task set}(S_\tau) = \frac{T_{CS}}{T_{total}} \times 100$$

Where,

$\boldsymbol{S_\tau}$ = Schedulability for a taskset τ

$\mathbf{T_{CS}}$ = The no. of tasks who has completed its execution successfully whose single job has not missed the deadline

$\mathbf{T_{total}}$ = The total no. of tasks in the taskset

2.3.4 Deadline miss ratio (%)

The Deadline-miss-ratio can be written as "the ratio between the no. of tasks those missed their deadlines to the total no. of tasks entered into the RTS" [82][114]. It is a very essential performance measure parameter for any real-time system which specifies the ratio of deadline misses to total admitted tasks into the system. The deadline miss ratio shows the task's deadline misses from a given taskset in terms of percentage. The formula for the deadline miss ratio is given as follows:

$$\mathbf{D} = \frac{T_n}{(T_n + T_m)} \times \mathbf{100}$$

Where D=Deadline Miss Ratio

T_n = The total tasks who had missed the deadlines

T_m= The total tasks who had met their deadlines

2.3.5 Context Switch

The Context Switch can be defined as "the procedure to store the state of a Task /Job for the future so it may be restored and resume the task's execution later on" [29] [115]. In

other words, when we transfer the CPU control from one Task to the another task it needs to save the context (current state information) of the presently executing task and to load the context of another ready task. The mechanism to save and restore is called the context switch [104]. The main disadvantage of the context switch is that it includes the vast cost to the system in terms of CPU time as the system will not perform any fruitful task during the context switch. So whatever number of context switch happens into the system using particular scheduling approach that should be counted and analyzed the overhead.

2.4 Summary of the Literature Survey

As per the analysis of the literature review done so far, it is seen that very less or negligible work for multi-processor scheduling is done using processor affinity based approach in soft real-time environment so with this findings in this research the proposed scheduler presented which is affinity based for soft real-time system.

In last five decades, many scheduling algorithms have been proposed but the main focus was uniprocessor scheduling and optimal algorithms exists for uniprocessor real-time system so still research scope for the multi-processor scheduling. Multi-processor real-time schedulers are important for the applications of this computing era. The research work is proposed for multiprocessor scheduling.

Each existing scheduler has its own pros and cons, but they are trying to improve schedulability, reduce the deadline miss ratio, reduce the tardiness, reduce the number of context switches but no more focus on CPU_Utilization especially in the real-time environment so I have tried to analyse all five performance measure parameters in this research.

Most of the algorithms in literature addressed the issue of process selection/priority assignments but very less focused on processor selection direction so in the proposed approach the main aim is to select appropriate processor so it will increase the overall schedulability. In existing multiprocessor schedulers P-FP and G-JLFP, the priority assignment taken into consideration is either fully static or job level fixed priority but it can be test with Job-Level-Dynamic priority assignment to improve the schedulability as well as CPU_Utilization. The traditional approaches like partitioned and global shown in literature are application based but modern RTOS supports the concept of arbitrary processor affinity which realizes any of the approach as per our application and it is more flexible and

generalized than the existing one so the proposed scheduler uses the affinity concept in this research to make it generalized scheduler for SRTS. The existing scheduler provides restricted migration which limits the overall system schedulability but this proposed research is providing additional flexibility in migration along with APA by task shifting using priority reprioritization and dynamic priority assignment which may improve the schedulability, improve the cpu_utilization, reduce the deadline- miss-ratio and reduce the tardiness.

Chapter 3:
PrASWM a Real-Time Multiprocessor Scheduling Approach

In this chapter, a new scheduling approach along with JLDP priority assignment, processor affinity based processor selection policy and without using flexible migration policy is proposed which has been designed and implemented for multiprocessor soft real-time system. It has been named as **PrASWM** (processor affinity based scheduler without flexible migration).This approach has been designed to provide generalized multiprocessor soft real-time scheduling approach with improved overall Schedulability, maximize CPU_Utilization, minimize the Deadline miss ratio and minimize the Tardiness. This chapter describes the introduction to processor affinity, structure of proposed processor selection approach using affinity, the experimental results and comparison between P-FP-APA, G-EDF-APA and PrASWM for the five performance measure parameters like Deadline Miss Ratio, Schedulability, Tardiness, No. of Context Switches and CPU_Utilization.

3.1 Introduction to Affinity

This section presents the introduction to processor affinity, importance of processor affinity, relation between real-time task scheduling and processor affinity and demonstration for generality of the affinity.

3.1.1 Arbitrary Processor Affinity

The processor affinity means an arbitrary set of processors on which we can execute a process. It allows us to bind a task to a random subgroup of processors inside the system. The example of it is Linux kernel which supports a kernel API called *sched_set_affinity ()* that will be used to set the affinity of the task / job with specification that the task/job can be scheduled on a CPU that is in task's processor_affinity. FIGURE 3.1 demonstrates the processor affinity of a task T_i which is $\{P_8, P_9, P_{11}, P_{12}\}$; it means that the task T_i can be executed on any one of the processor which is free amongst its affinity set but not on any other processor out of its affinity.

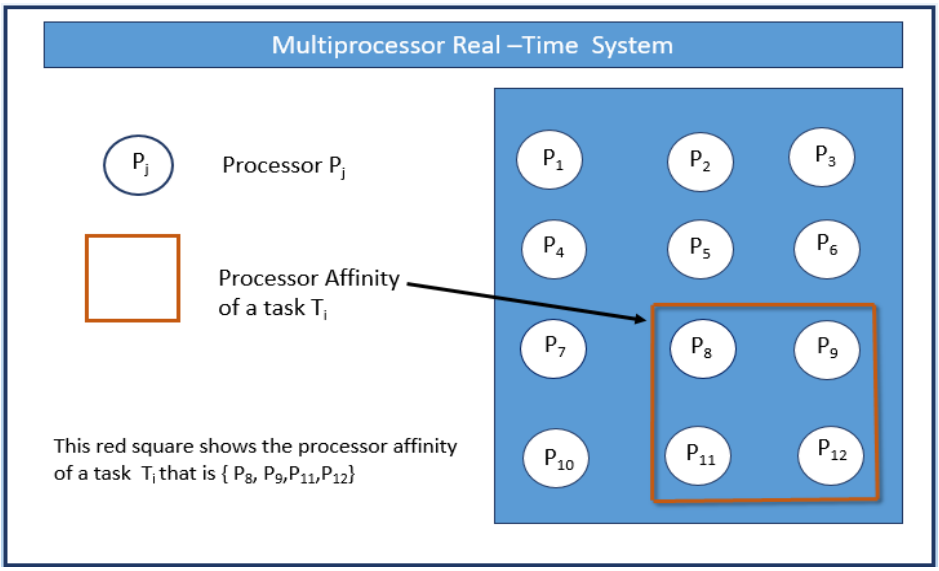


FIGURE 3.1: Illustration of Arbitrary Processor Affinity

3.1.2 Significance of Processor Affinity

Generally, to enhance an application's performance processor affinity is used in output-oriented computing [68]. Processor affinity is also useful in realizing the global scheduling, partitioning scheduling and clustered scheduling like we can achieve partitioning scheduling by setting every task's affinity to strictly single processor and global can be presented by setting affinity to all processor for every task into the system [68]. The scheduling point of view this is very interesting feature as the arbitrary processor affinity (APA) assignment is possible on a task-by-task basis which will allow to get more flexible migration policies and to design a generalized approach rather than that presented in the literature [76]. The separation of the RTAs from the non-RTAs is possible using processor affinity by allocating them to different CPUs. To reduce the power ingestion and improve the performance a task can be executed on particular CPU with the help of processor affinity in heterogeneous environment [7] [68].

3.1.3 Processor Affinity and Task Scheduling in Real-Time Systems

The contemporary RTOS like Lynux, QNX are using the concept of affinity assignment for processor selection instead of conventional methods described in prior researches to implement its scheduling approach that provides additional generalized techniques in multiprocessor environment. Several such OS had used the processor affinity mechanism with constrained migration in HRTS. The processor affinity ($\boldsymbol{\alpha_k}$; where$\boldsymbol{\alpha_k} \subseteq \boldsymbol{\pi}$) of any task presents the set of processors on which the job of that task can

execute[46][47]. The processor affinity can be changed with the help of the kernel APIs; an important example is the Linux kernel that offers sched_get_affinity( ) to reading the affinity and sched_set_affinity( ) to set the affinity for the task correspondingly [1] [21] [34] [67]. In the research it has been assumed that the affinity of a task won't be altered throughout task's entire lifecycle. The existing approaches of migration described in literature will be presented using affinity concept. By setting the CPU affinity of every tasks to all processors the global scheduling approach can be realized and by setting the affinity of every task to a one fixed CPU can realize the partitioned approach [22] [46] [67]. The processor selection approach based on affinity can be joined with any process selection methodology.

3.1.4 Generalized approach with Processor Affinity based Scheduling

With the watchful allocation of processor affinity, one can enhance the output, simplification of load balancing can be done and also it will be used for isolation of the applications from each other. This research focuses on the schedulability profits of APA scheduling in contrast to global scheduling and partitioning scheduling. It shows that arbitrary processor affinity is a valuable construct from the perception of the scheduling in real-time also.

A constrained-migration model can be represented by the arbitrary processor affinity (APA) scheduling which will restrict the migration and scheduling of the task on a random processor set. With a suitable assignment of affinity, the job will be either permitted for migration between all CPUs similar to global approach, permitted for migration amongst a processor's subgroup similar to clustered or no migration permitted at all similar to the partitioning scheduling. If we find feasible schedule for a task set using global approach, partitioned approach or clustered approach then it will be also possible to schedule using APA scheduling. The Affinity scheduling is able to imitate global, clustered and partitioning with appropriate assignment of processor affinity to every task into the system. Conversely, The disjoint processor affinity of a task is not necessary requirement using APA scheduling like it is required in clustered scheduling, it means that there are two different tasks T_i and T_k may have unequal processor affinities α_i and α_k in such a way that $\alpha_i \cap \alpha_k = \varphi$.

So, there may be the task sets those can be scheduled using affinity implementation but not possible to schedule using global, clustered and partitioning approaches.

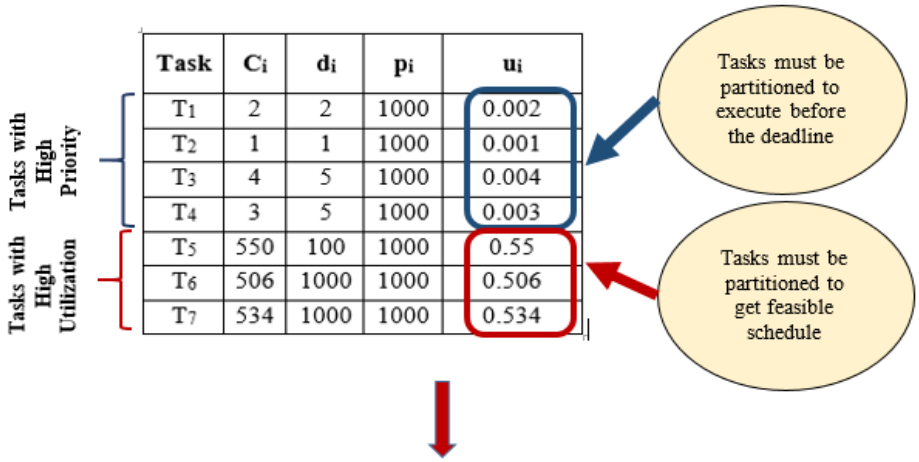
FIGURE 3.2 proves that the global, partitioned, and clustered along with JLFP is firmly dominated by the processor affinity with Job-Level-Fixed-Priority scheduling approach. Here it is proved that the task set may exist which can be scheduled under APA scheduling but it cannot be scheduled under the global approach, partitioned approach or clustered approach along with job-level-fixed-priority approach of scheduling. A task set shown in FIGURE 3.2 that will be scheduled in two CPUs system. Here the job-level-fixed-priority approach is taken into consideration to assign the priorities to the tasks and T_1 is having arrival time 1 whereas the timing of arrival for all other tasks is 0. Here we are trying to schedule the given task set on 2 CPUs with global, partitioning and APA scheduling along with job level fixed priority. Here the clustered scheduling is not shown separately as with two processors if we create cluster of size one then it presents portioning and if we take cluster size two then it is global scheduling.

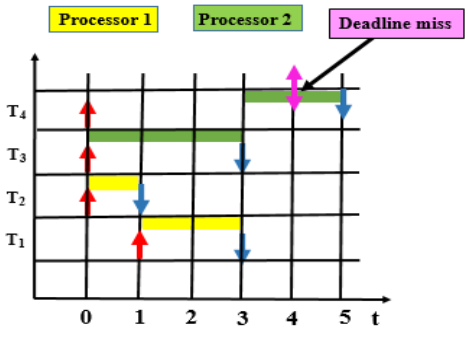
Applying Global scheduling approach: There is two processor system and tasks T_1 and T_2 have densities one and hence to acquire a feasible schedule with no deadline miss; all the jobs of tasks T_1 and T_2 should be assigned the maximum priorities. In addition to that, tasks T_3 and T_4 are having comparatively very small deadline than the worst-case-running-time of tasks T_5, T_6, and T_7 so the jobs of tasks T_3 and T_4 should be allocated greater priority as per the comparison with the jobs of the T_5, T_6, and T_7. Because of this limitations, either jobs of T_3 will be allotted the 3^{rd} and jobs of T_4 will be allotted the 4^{th} uppermost priority or the reverse is also possible. In both the cases, either T_3 or T_4 whoever is assigned the low priority will miss the deadline as demonstrated in FIGURE 3.2 (a)-(b). Therefore, it has been demonstrated that it is not possible to schedule the task set under the global scheduling.

Applying Partitioned scheduling: First and foremost condition to get a feasible schedule is that the total task utilization of a partition must be less than or equals to 1. As shown in FIGURE 3.2 the task utilizations of T_5, T_6, and T_7 are 0.501, 0.5001, and 0.5 respectively. So the dividing of the tasks T_5, T_6, and T_7 into two different portion is not possible as it causes the total task_utilization greater than 1 in this example for two processor system. Therefore it can be demonstrated that it is not possible to schedule a taskset under the partitioned scheduling.

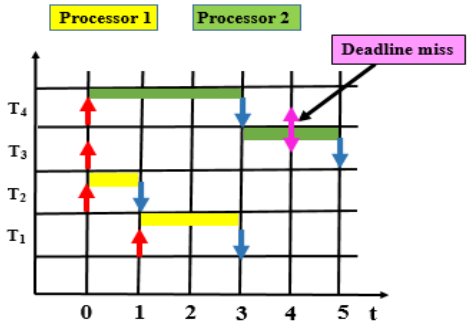
Processor Affinity scheduling approach: The global scheduling failure is suggesting that T_3 and T_4 must be scheduled on different processors and also tasks T_1 and T_2 should be scheduled on separate processor due to their unit densities. This division is not possible with

partitioned scheduling as tasks T_5, T_6, and T_7 violates the condition for feasible schedule as task utilization is greater than 1. Hence, with the help of processor affinities as shown in FIGURE 3.2 (d), the partition of T_1, T_2, T_3 tasks, T_4, T_5 and T_6 tasks can be done but the T_7 task will be allowed to migrate and FIGURE 3.2 (c) shows the solution using affinity. Hence it is shown in FIGURE 3.2 (c) that using APA scheduling the task set is possible to schedule.

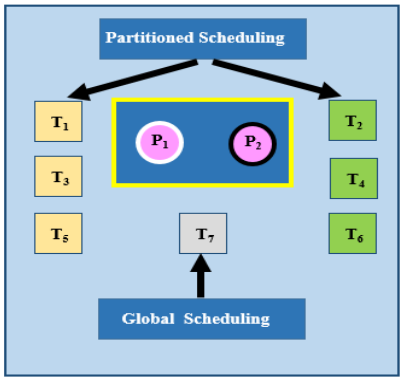




a) Deadline miss of T_4 with Global Approach



b) Deadline miss of T_3 with Global Approach



c) Solved with Processor Affinity

Task	Processor Affinity Set
T_1	π_1
T_2	π_2
T_3	π_1
T_4	π_2
T_5	π_1
T_6	π_2
T_7	π_1, π_2

d) Processor Affinity Assignment

FIGURE 3.2: Traditional approaches Vs. the generality of Processor Affinity Scheduling

3.2 Structure of Proposed Scheduling Approach without Migration Policy: PrASWM

In this section the structure of G-JLFP-APA, P-FP-APA and proposed scheduling approach PrASWM has been shown. The first part describes the pseudo code of "Affinity assignment for G-JLFP-APA, P-FP-APA and PrASWM" and second part describes the pseudo code of "Task Priority Assignment for G-JLFP-APA, P-FP-APA and PrASWM".

3.2.1 Pseudocode for Affinity Assignment

The Algorithm 3.1 shows the steps for affinity assignment in G-JLFP-APA, P-FP-APA and PrASWM. In global approach, the affinity is assigned to all processors means that the ready task T_k can be executed on any available processor in set α and hence here full migration is allowed as the affinity of each task is all processors in affinity set α.In Partitioned approach, the affinity of each task T_k is single fixed processor from the affinity set α and hence here migration is not allowed at all. In PrASWM, the affinity of each task is assigned that is arbitrary processor affinity rather than all or single processor. Here in this research the affinity assignment is kept static.

Algorithm 3.1: Affinity Assignment in different Scheduling Approaches

```
BEGIN
IF the Global_approach THEN
    Set_Affinity()
        Affinity_Assignment_to_All_Processors in α
        WHILE task_becomes_ready_execute (TK )
          Check For all processor π in α
            IF any processor π is free THEN
                Schedule Task Tk on available processor πi
            ELSE IF no free processor in α THEN
                IF Any Running Task Ti on πi with priority < Tk THEN
                    Preempt Ti from πi and schedule Tk on πi
                ENDIF
            ELSE
                TK has to wait for processor π in α to become free
            ENDIF
        ENDWHILE
ENDIF
```

```
IF the Partitioned_approach THEN
    Set_Affinity()
        Affinity_Assignment_Single task_to_Single_Processor(π_k) in α
        WHILE task_becomes_ready_execute (T_K)
            Check For only single processor π in α which is in affinity of T_K
            IF assigned processor π_k is free THEN
                Schedule Task T_k on assigned processor π_k
            ELSE IF assigned processor π_k is not free THEN
                IF Any Task (T_i) running on π_k with priority < T_k THEN
                    Preempt T_i from π_k and schedule T_k on π_k
                ENDIF
            ELSE
                T_K has to wait until processor π_k in α to become free after executing T_i
            ENDIF
        ENDWHILE
ENDIF
IF the PrASWM_approach THEN
    Set_Affinity()
        Affinity_Assignment_to_Arbitrary_Processor_Affinity in α
        WHILE task_becomes_ready_execute (T_K)
            Check For processors π in α which are in affinity of task (T_K)
                IF any processor π amongst the affinity of T_K is free THEN
                    Schedule Task T_k on available processor π_i
            ELSE IF processor in affinity of T_K is not free THEN
                IF Any Running Task T_i on π_i which is in affinity of T_K with priority < T_k THEN
                    Preempt T_i from π_i and schedule T_k on π_i
                ENDIF
            ELSE
                T_K has to wait until any processor π in α which are in affinity of T_k to become free
            ENDIF
        ENDWHILE
ENDIF
END
```

3.2.2 Pseudocode for Priority Assignment

The Algorithm 3.2 shows the steps for priority assignment in G-JLFP-APA, P-FP-APA and PrASWM. In global approach along with JLFP the priority is assigned that means the different priority levels can be assigned to the different jobs of T_k task based on its

deadline but it remains constant during its execution time. In partitioned approach along with FP the priority is assigned that is task-level-fixed-priority means that all the jobs of T_k task are assigned the similar priority that is task T_k's priority and it remains fixed. In PrASWM along with JLDP priority is assigned that is job-level-dynamic-priority means that different jobs of T_k task will have the dissimilar priorities based on its timing constraints and that will be permitted to modify based on task's deadline throughout its lifetime hence it can be said as fully dynamic approach.

Algorithm 3.2: Priority Assignment in different Scheduling Approaches

```
BEGIN
IF the Partitioned_FP_Approach THEN
    Set_Task_Level_Fixed _Priority(Task_for_assignmet T_K)
        Priority of the Task T_k is assigned to all its jobs which remain fixed during its lifetime (no
        migration)
ENDIF
IF the Global_JLFP_Approach THEN
    Set_Job_Level_Fixed_Priority (Task_for_assignmet T_K)
        All the jobs of the T_k Task is allotted different priority levels based on its deadline but it
        remain fixed during its lifetime (Full migration)
ENDIF
IF the PrASWM_approach THEN
            Set_Job_Level_Dynamic_Priority (Task_for_assignmet T_K)
        All the jobs of the T_k Task is allotted different priority levels and it may be changed during
        its lifetime based on the shortest deadline(Migration within Task's Affinity)
ENDIF
END
```

3.3 Results

This section presents the experimental setup and the comparative result analysis of different scheduling approaches for various cases.

3.3.1 Experimental Setup

In this research, experiment and result analysis are done for five parameters that is deadline miss ratio, tardiness, number of context switches, schedulability analysis and CPU_Utilization for multiprocessor real-time system. TABLE 3.1 presents task set generation criteria and the value taken into consideration. The SimSo API is used for Task

set generation along with "Random Vectors with fixed Sum" algorithm as explained in section 5.2.3 [42]. To implement the different schedulers and to perform experiments the LITMUS^{RT} simulator has been used [24] [25] [30] [105]; the scheduler analysis is done by the Sched_trace APIs of the LITMUS-RT and trace analysis is done using Feather_Trace APIs of LITMUS-RT. Lib LITMUS Library used for various APIs [24] [30]. The random task periods has been taken in range of (10ms to 100ms) using the log-uniform period allocation method [46]. The Affinity assignment is done using arbitrary processor affinity with static approach [25] [24] [68]. In this research we have taken constrained deadline that is ($d_i < p_i$) not taken implicit deadline.

In this research, the Job-Level-Dynamic Priority assignment along with PrASWM approach and for that the DkC priority assignment approach is used as mentioned in section 5.2.4 [46]. The section 3.3.2 presents the result analysis for various cases in which PrASWM compared with existing methods for 100 Task sets with various Task set size (n=4, 8,16, 32, 64) and the No. of Processors (m=2, 4) up to the 4.

TABLE 3.1: Task Set Generation Criteria and the value taken into consideration

Criteria	**Value**
Type of RTS	Soft Real-Time System
Priority Assignment	JLDP
Task set Count	100
Running time	60 seconds
Period Range	10ms to 100ms
Task set Generation Algorithm	Random Vectors with fixed Sum algorithm
No of processors	2,4
No of Tasks in a task set range	(4,8,16,32,64)
Task Utilization	For all Taskset more than 75% of m
Processor Selection	Fixed Affinity Assignment
Relative Deadline	$d_i < p_i$
Migration	Within its affinity without priority reprioritization
Priority Assignment	JLDP

3.3.2 PrASWM on 100 Tasksets with different Taskset size

In this case we have taken results for single taskset which has been tested on no. of processors (m=2, 4) and the taskset size (n=4, 8, 16, 32, 64).The taskset size is kept fixed which is 2^i (where the $1< i < 7$, m is no. of processors and n is taskset size).

There are three different test categories have been selected.

1. The test has been taken with two Processor and taskset size (n=4, 8, 16, 32, 64).
2. The test has been taken with four Processor and taskset size (n=4, 8, 16, 32, 64).

There are five parameters considered for the results.

- Parameter 1: Average Deadline Miss Ratio should be minimum.
- Parameter 2: Average Tardiness (ns) should be as low as possible.
- Parameter 3: Average No. of Context Switches should be minimum.
- Parameter 4: Average Schedulability (%) should be as high as.
- Parameter 5: Average CPU_Utilization (%) should be maximum.

Here, following three different scheduling algorithms have been compared.

1. P-FP-APA (Partitioned Approach with fixed priority using Affinity)
2. G-JLFP-APA (Global Approach with Job-Level-Fixed priority using Affinity)
3. Proposed Approach (PrASWM without flexible migration policy)

All the tests have been taken on LITMUS^{RT}. The details about simulator LITMUS^{RT} has been discussed in Chapter 5.

The results mentioned here is the average of all the tasks of a 100 Tasksets. One Taskset contains the no. of tasks with the range (n=4, 8, 16, 32, 64) and many jobs of each task can appear within the given duration as per the given period. Task generation detail is given in TABLE 3.1.

Case-1 PrASWM on the 100 Task sets for Average Deadline Miss Ratio

1. Average of 100 Tasksets with two Processor and Taskset Size (n=4, 8, 16, 32, 64)

TABLE 3.2: Average Deadline Miss Ratio on Two Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	0.0068	0.0045	0.0018
n=8	0.0720	0.0515	0.0351
n=16	0.3904	0.2165	0.0146
n=32	0.5902	0.4366	0.2165
n=64	0.8041	0.5954	0.4437

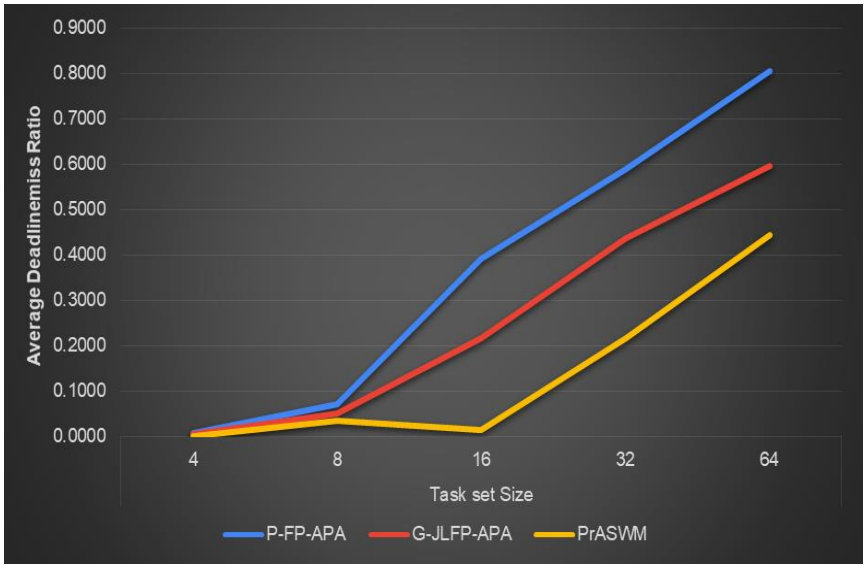


FIGURE 3.3: Average Deadline Miss Ratio on Two Processors

As shown in FIGURE 3.3, the PrASWM meets the maximum number of deadlines, so it has the lowest **Deadline miss ratio** as compared to all other schedulers. The P-FP-APA has missed the maximum no. of deadlines, so it's deadline miss ratio is high. In this case deadline miss happening for all schedulers are greater than four processors (FIGURE3.4). One more observation is that the deadline miss ratio is increasing as the no. of tasks increases for the same no. of processors.

2. Average of 100 Tasksets with Four Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.3: Average Deadline Miss Ratio on Four Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	0.0047	0.0032	0.0009
n=8	0.0509	0.0265	0.0370
n=16	0.2632	0.1890	0.0608
n=32	0.4065	0.3248	0.1822
n=64	0.6504	0.4068	0.2389

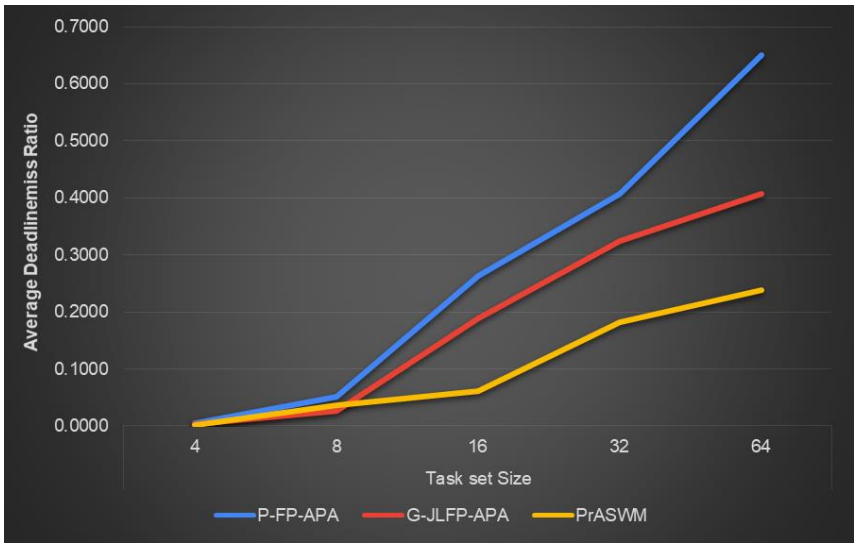


FIGURE 3.4: Average Deadline Miss Ratio on Four Processors

As shown in FIGURE 3.4, the PrASWM meets the maximum number of deadlines, so it has the lowest **Deadline miss ratio** as compared to all other schedulers. The P-FP-APA has missed the maximum no. of deadlines, so its deadline miss ratio is high. In this case deadline miss ratio is less for all schedulers than two processors (FIGURE3.3). One more observation is that the deadline miss ratio is increasing as the no. of tasks increases for the same no. of processors.

Case-2 PrASWM on the 100 Task sets for Average Tardiness

1. Average of 100 Tasksets with two Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.4: Average Tardiness on Two Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	25.65	21.63	10.85
n=8	29.56	26.70	17.20
n=16	34.98	30.66	19.67
n=32	42.09	34.29	26.54
n=64	48.61	39.52	31.55

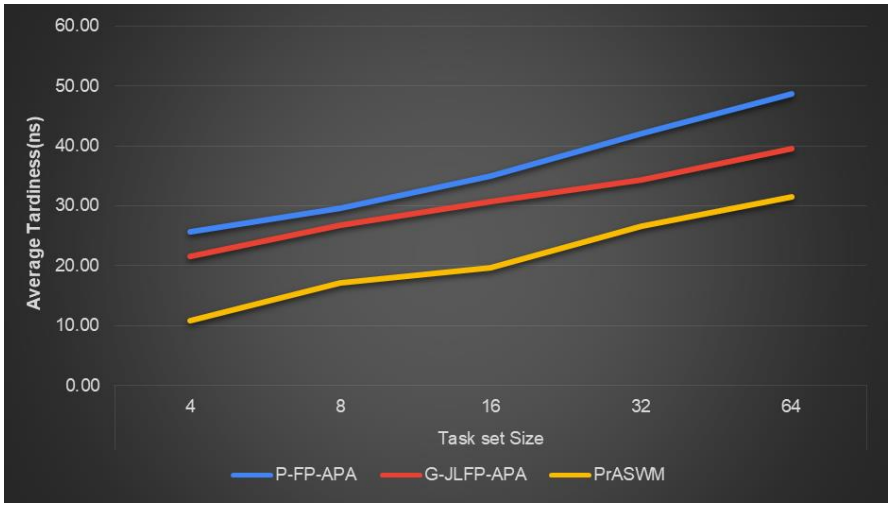


FIGURE 3.5: Average Tardiness on Two Processors

As shown in FIGURE 3.5, the PrASWM gives the less **Tardiness** as compared to all other schedulers, so PrASWM improves all over system performance as the more tardiness means time taken after deadline is more and in the soft real-time system the performance degradation happens in proportion to the time elapsed after the deadline. It also shows that the tardiness is increasing as the no. of tasks in a taskset size is being increased and decreases with increase in no. of processor (FIGURE3.6).

2. Average of 100 Tasksets with Four Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.5: Average Tardiness on Four Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	22.47	19.51	8.58
n=8	26.04	24.22	14.53
n=16	31.78	28.48	17.07
n=32	39.85	31.63	22.63
n=64	44.56	37.25	28.45

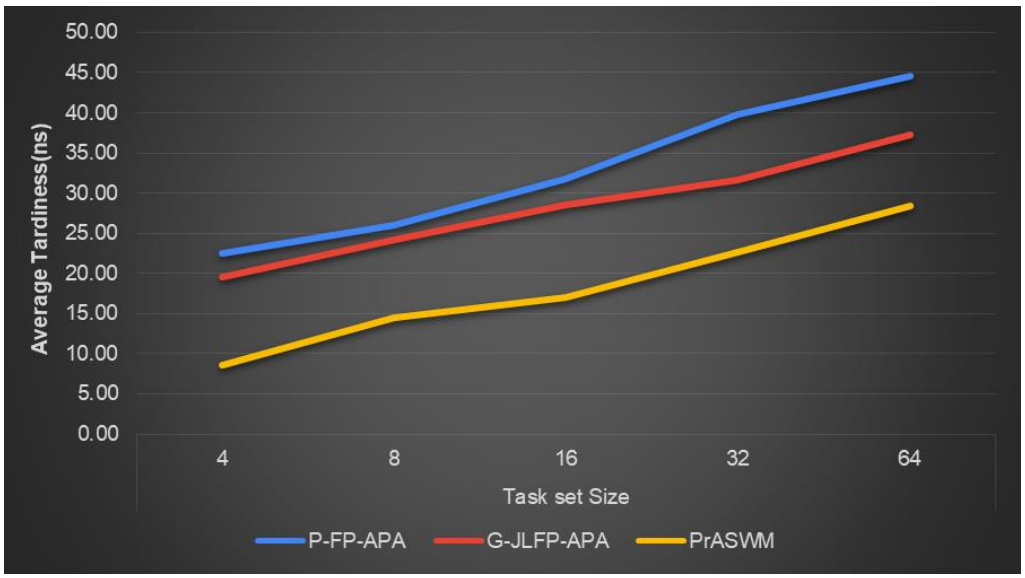


FIGURE 3.6: Average Tardiness on Four Processors

As shown in FIGURE 3.6, in this case also the PrASWM is giving less **Tardiness** as compared to the all other multiprocessor schedulers due to the PrASWM's dynamic priority assignment policy (JLDP) which is making more no. of tasks schedulable and reducing the deadline misses. It can be seen that as the taskset size is being increased the tardiness is also increasing.

Case-3 PrASWM on the 100 Task sets for Average No. of Context Switch

1. Average of 100 Tasksets with two Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.6: Average No. of Context Switch on Two Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	67.25	99.25	92.75
n=8	98.38	102.13	95.25
n=16	123.31	145.81	168.81
n=32	147.34	183.25	205.25
n=64	177.58	211.45	195.52

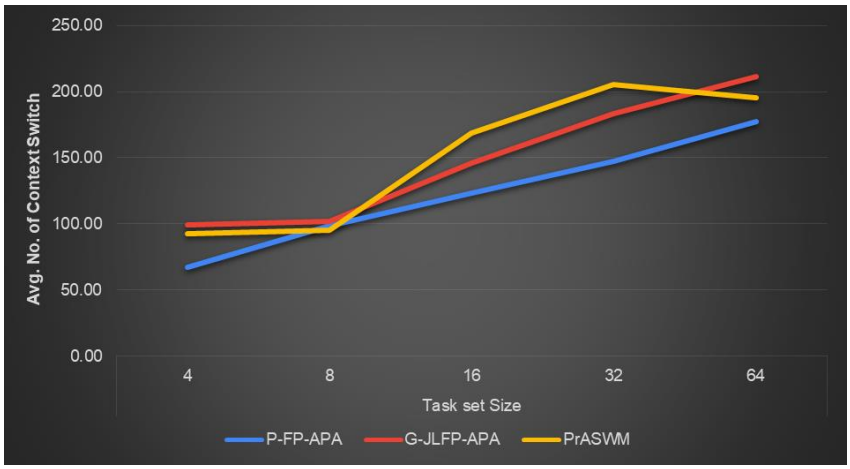


FIGURE 3.7: Average number of Context Switch on Two Processors

The FIGURE 3.7 shows that in all the cases the PrASWM has taken more no. of **Context switches** as compared to P-FP-APA but it is either less than or the greater than G-JLFP-APA due to PrASWM's dynamic priority assignment policy and P-FP-APA has almost no migration kind situation as the processors are strictly partitioned for all the tasks into the system. Another observation is that the no. of context switches are getting increased as the taskset size is increased.

2. Average of 100 Tasksets with Four Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.7: Average No. of Context Switch on Four Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	99.25	126.25	119.50
n=8	112.13	138.25	153.13
n=16	124.50	163.56	156.38
n=32	164.19	194.22	213.25
n=64	180.66	269.02	247.02

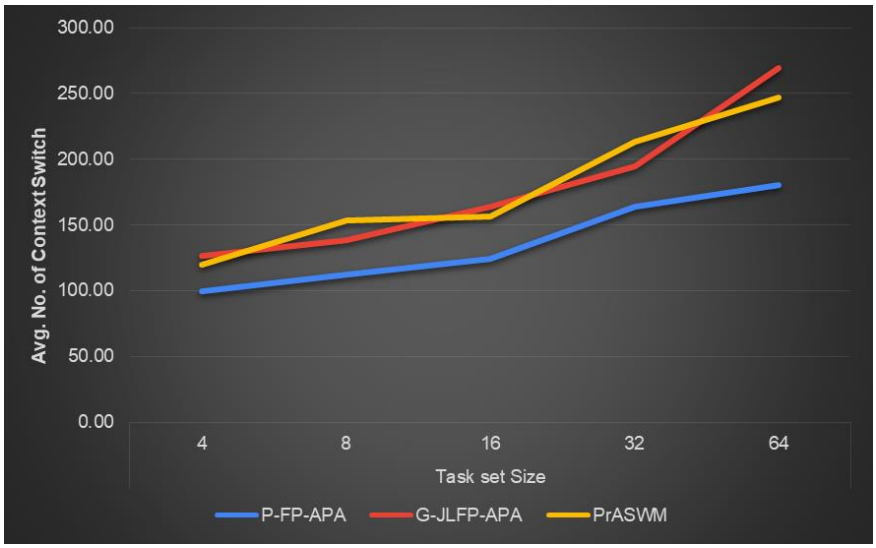


FIGURE 3.8: Average number of Context Switch on Four Processors

The FIGURE 3.8 shows that in all the cases the PrASWM and G-JLFP-APA have taken more no. of **Context switches** as compared to P-FP-APA as PrASWM provides job-level-dynamic priority assignment. Except few cases the PrASWM has taken less no. of context switches than G-JLFP-APA. One more thing in this case also the no. of context switches increased as the taskset size is increased with the same no. of processors.

Case-4 PrASWM on the 100 Tasksets for Average Schedulability

1. Average of 100 Tasksets with two Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.8: Average Schedulability on Two Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	71.50	81.50	84.00
n=8	67.13	78.13	82.50
n=16	64.88	76.56	79.63
n=32	62.09	73.16	76.06
n=64	57.48	64.47	72.02

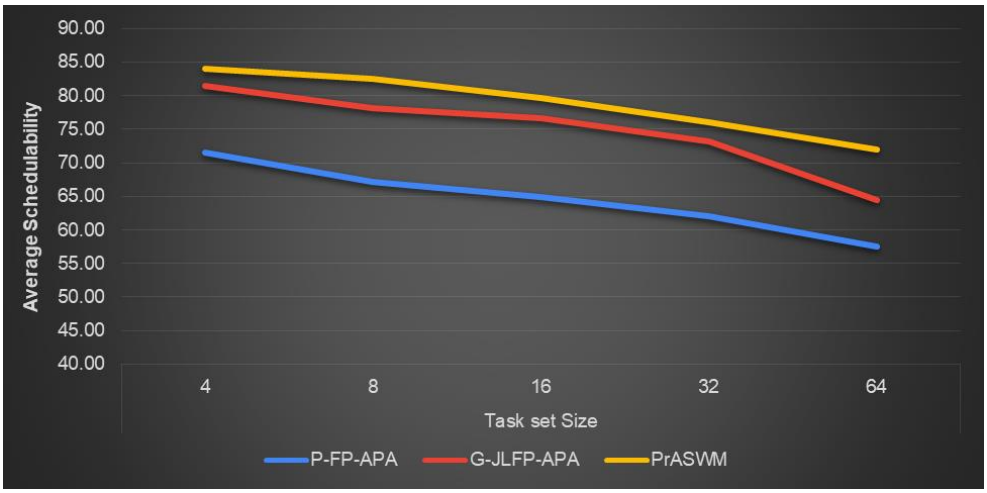


FIGURE 3.9: Average Schedulability on Two Processors

As shown in FIGURE 3.9, the PrASWM gives the more **Schedulability** that is more no. of tasks can be executed before its deadline as compared to all other multiprocessor scheduler due to PrASWM's fully dynamic priority assignment. It also shows that the schedulability is decreasing as the no. of tasks in a taskset size is being increased with the same no. of processors but if it is compared with FIGURE 3.10. It can be concluded that the schedulability is increasing if no. of processors are increased.

2. Average of 100 Tasksets with Four Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.9: Average Schedulability on Four Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	73.50	84.25	86.25
n=8	68.75	80.88	83.13
n=16	67.19	78.06	80.31
n=32	64.03	74.06	78.13
n=64	60.33	67.89	74.05

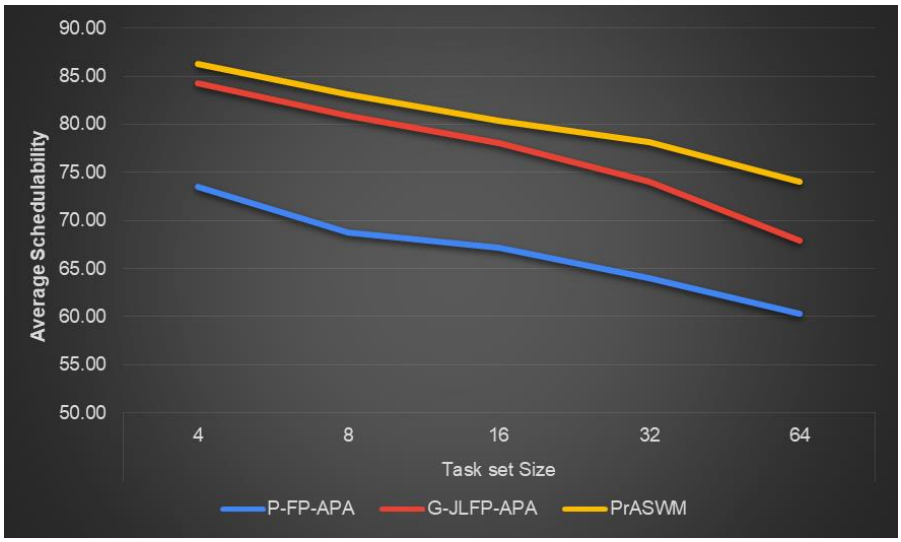


FIGURE 3.10: Average Schedulability on Four Processors

The FIGURE 3.10 shows that in all the cases the PrASWM increases overall **Schedulability** of the system as compared to all other scheduler as it provides fully dynamic priority assignment policy along with affinity based processor selection mechanism and P-FP-APA has lowest schedulability. One more thing in this case also is the schedulability decreased as the taskset size is increased and the schedulability increased with the no. of processors are increased.

Case-5 PrASWM on the 100 Task sets for Average CPU_Utilization

1. Average of 100 Tasksets with two Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.10: Average CPU_Utilization on Two Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	51.46	53.96	55.29
n=8	54.25	59.53	63.11
n=16	57.33	63.36	64.28
n=32	61.40	64.32	68.71
n=64	65.47	71.45	73.77

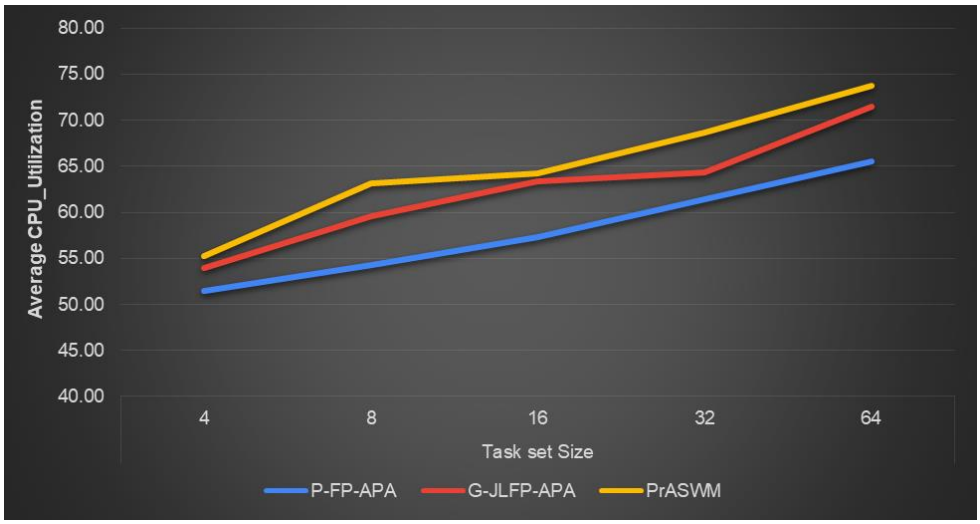


FIGURE 3.11: Average CPU_Utilization on Two Processors

As shown in FIGURE 3.11, the PrASWM gives the more **CPU_Utilization** that is more no. of tasks get executed before its deadline as compared to all other multiprocessor scheduler due to PrASWM's dynamic priority assignment. It also shows that the CPU_Utilization is increasing as the no. of tasks in a taskset size is being increased with fixed no. of processors but if it is compared with FIGURE 3.12 it can be concluded that the CPU_Utilization is decreasing if no. of processor are increased.

2. Average of 100 Tasksets with Four Processor and Taskset Size (n=4, 8, 16, 32, 64).

TABLE 3.11: Average CPU_Utilization on Four Processors

Taskset Size	Algorithms		
	P-FP-APA	G-JLFP-APA	PrASWM
n=4	48.25	50.22	53.43
n=8	51.78	56.21	59.59
n=16	54.34	58.17	61.92
n=32	57.54	61.98	64.57
n=64	61.33	65.67	70.64

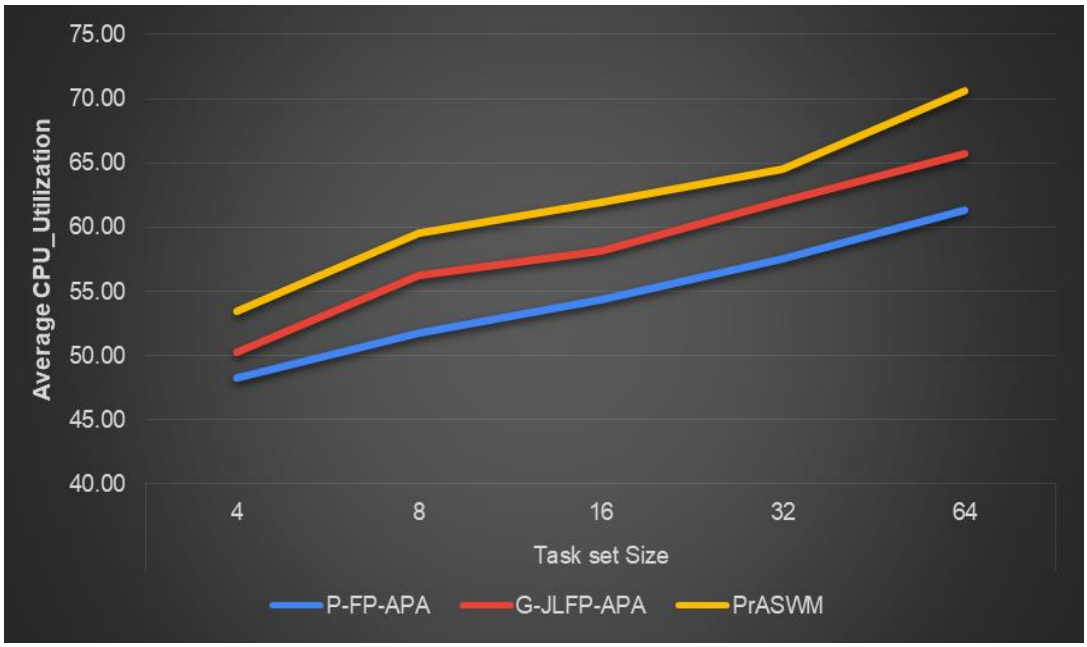


FIGURE 3.12: Average CPU_Utilization on Four Processors

The FIGURE 3.12 shows that in all the cases the PrASWM increases **CPU_Utilization** of the system as compared to all other scheduler as it provides affinity implementation along with fully dynamic priority assignment mechanism and P-FP-APA has lowest CPU_Utilization. One more thing in this case also is the CPU_Utilization is increased as the taskset size is increased and the CPU_Utilization decreased with the no. of processors are increased.

Chapter 4:
A Multiprocessor Scheduler PrAMS for Soft Real-Time Systems

This chapter presents a novel processor affinity based scheduler along with JLDP priority assignment approach and flexible migration policy is proposed which has been designed and implemented for multiprocessor soft real-time system. It has been named as **PrAMS** (Processor Affinity based Multiprocessor Scheduler). This scheduler has been designed to improve overall schedulability, maximize CPU_Utilization, reduce the Deadline-miss-ratio and minimize the Tardiness. The Arbitrary processor affinity, Illustration of Conventional and Hybrid Migration Approaches, the real-time task migration scenarios in real-time system has been demonstrated to understand the problem in existing schedulers and requirement to develop this novel scheduler will be explained first and then the structure of PrAMS will be discussed in this chapter.

4.1 Arbitrary Processor Affinity and Linux Scheduler

We can understand the concept of processor affinity to select processor for the task with the help of an illustration of Linux Scheduler in multiprocessor environment. To accomplish scheduling quickness in priority based scheduling for real-time task the runqueue is divided into many number of linked list per priority one list. Each list made up of doubly linked list of tasks ready to run with its priority. In multiprocessor environment each processor is having its own runqueue. At a time any task may be only on one runqueue. At the time of new task arrival, the incoming task is to be inserted into any one processor's runqueue based on its processor affinity and priority number [67] [77]. When task gets completed, the task will be removed from the queue and runqueue will be restructured. If necessary, the task can be shifted from one CPU runqueue to another CPU runqueue with following the strict affinity constraints of a task. Updating the runqueue of any processor generates various scheduling cases.

When any lower priority task awakens on a runqueue that is running higher priority task or the lower priority task would be preempted by higher priority task then the scheduler

push the lower priority task to another runqueue. The runqueue on which the task is to be pushed should have lower priority tasks.

Whenever any currently running higher priority task completes its execution and lower priority task is going to be scheduled, the scheduler examines every processor's runqueue for pulling highest priority task from the other runqueue if any available. To push task on different processor's runqueue or to pull task from other processor's runqueue must not violate the affinity constraints [77]. The jobs are said to be suitable for pushing on the other queue if it is not scheduled currently. On the other hand the tasks that has been already scheduled cannot be pulled from any runqueue. The task in execution can be removed only when the higher priority task arrives. The running task will release its processor either voluntarily when execution gets complete or its time quantum is over in time sharing system. The load balancing requirements for runqueue is also checked periodically by Linux scheduler so maximum CPU Utilization can be achieved. However, Linux scheduler can't be flexible to move higher priority task on another processor for lower priority task which could not be executed elsewhere because of its processor affinity constraint; so restricted migration mechanism fails to attain higher schedulability. In this research this scenario explained in detail and presented novel processor affinity based real-time scheduling algorithm with flexible migration policy to overcome these limitations and improve the overall schedulability of the real-time system.

4.2 Illustration of Conventional and Hybrid Migration Approaches

The degree of migration allowed to a task is the key parameter for classifying the multiprocessor real-time scheduler. The two excessive ends of this spectrum are the global scheduling approach and the portioned approach. In global scheduling, there will be a single common queue between all processors and ready to run tasks will be assigned dynamically to any available processor according to its priority while in partitioned scheduling, every task has assigned fixed processor on which it will be executed without any migrations. The hybrid approaches are also designed by the researchers but one of the prominent hybrid approaches is clustered scheduling, in this approach the system processors are classified into the separate clusters and every task will be assigned to a single cluster and "global" approach will be applied inside the cluster.

The recent RTOS like VxWorks, LynxOS, QNX, and the real-time extended version of the Linux are using the processor affinity concept which is providing more flexible

migration and generalized algorithm instead of implementing the traditional methods described in the literature. FIGURE 4.1 illustrates the different migration strategies like global scheduling is with full Migration policy, partitioned scheduling is with no migration, clustered scheduling is with partial migration and the proposed approach with flexible migration for which schedulability analysis has been done in this thesis.

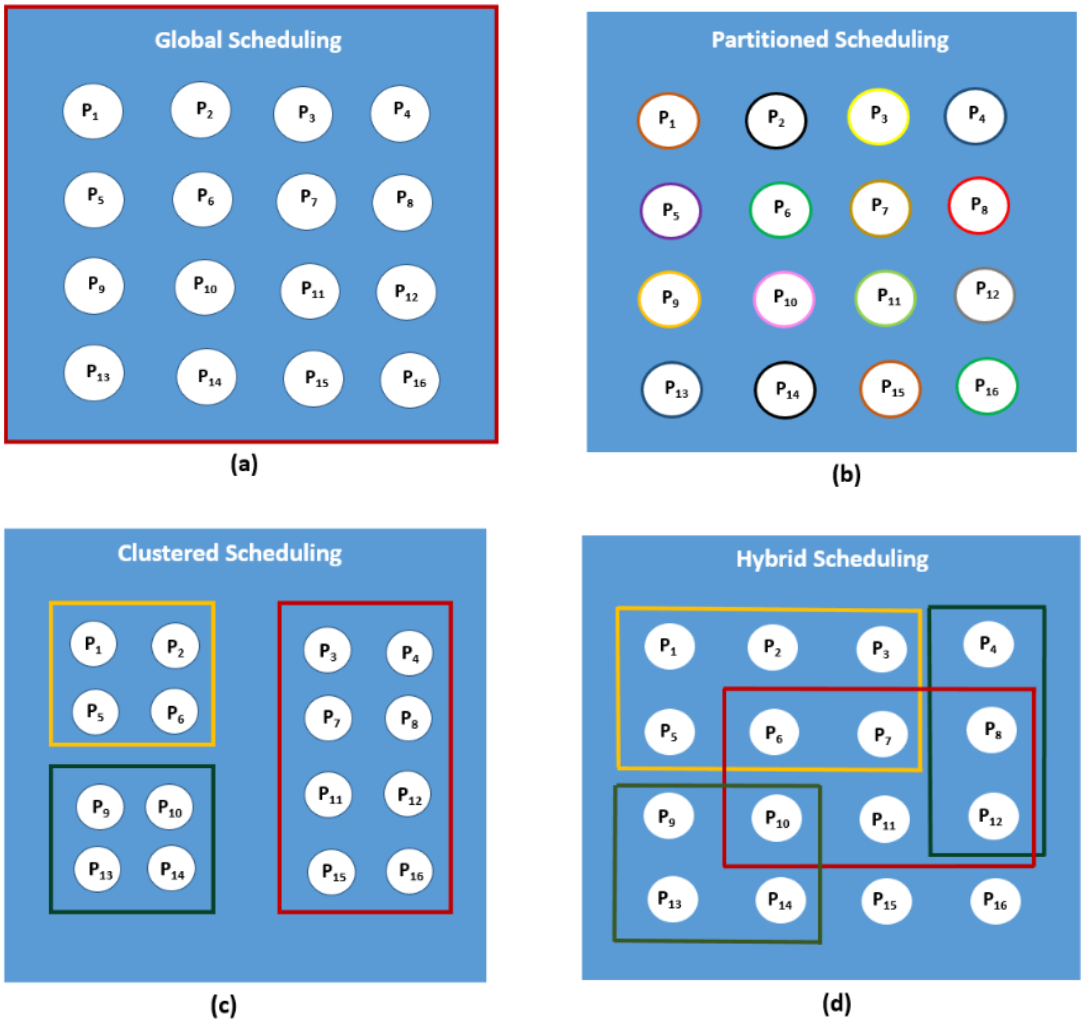


FIGURE 4.1: Illustration of different migration strategies:

a) Full Migration policy with Global scheduling **b)** No migration with Partitioned Scheduling **c)** Partial migration with Clustered Scheduling **d)** Proposed Approach for which the schedulability analysis is done in this research

4.3 The Task Migration Scenarios for Real-Time Tasks

The uniprocessor system faces the problem of priority inversion which is the case when the tasks with high priority have to wait for the tasks with the lower priority for starting its execution. The same situation may occur in multiprocessor system when the low priority job T_k holds a processor that is not in the affinity set α_j of task T_j, but another higher priority

task T_i has common processor affinity set α_i with α_j and α_k (i.e., also T_k holds processor that is common with T_i, where $p_i > p_j > p_k$. Let us understand this case with scenarios given below.

Case 1:

In FIGURE 4.1, various migration opportunities have been considered. A RTS with three CPUs and taskset of four tasks along with $P_1 > P_2 > P_3 > P_4$ priority order and processor affinity sets $\alpha_1 = \{\pi_1, \pi_2, \pi_3\}$, $\alpha_2 = \{\pi_2, \pi_3\}$, $\alpha_3 = \{\pi_1\}$, $\alpha_4 = \{\pi_2, \pi_3\}$ are taken into consideration. As shown in FIGURE 4.2 (a) at an instance of time where task P_3 is waiting and tasks P_1, P_2, P_4 are running. Now P_3 becomes ready to execute but scheduler will not allow run it on π_1 as it is running P_1 that is having high priority than the P_3.whereas the task P_4 is being executed on processor π_3 that is in the affinity of P_1 task. There may be the solution to preempt task P_4 as the priority order is P_3>P_4 and we can shift P_1 the higher priority task to processor π_3 to schedule task P_3 on π_1.One likely solution will be the pulling P_4 from the π_3 and pushing the P_1 (P_1>P_3) from processor π_1 to processor π_3.

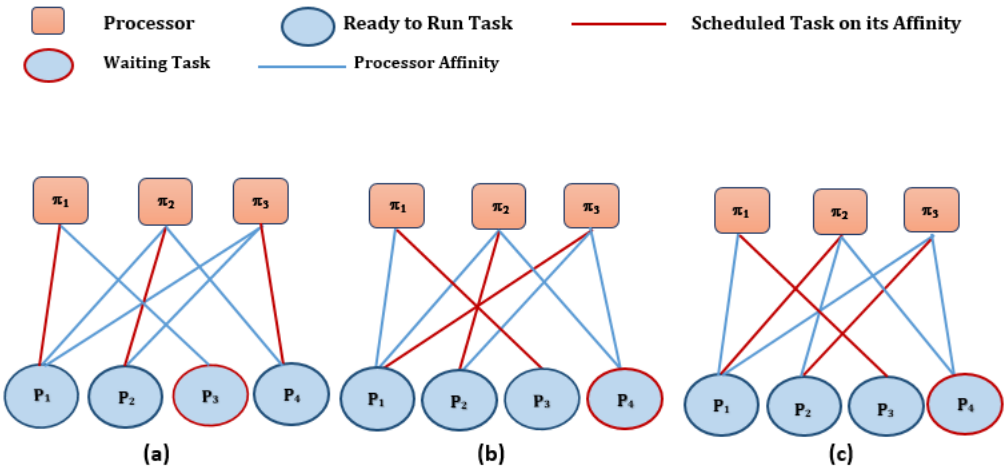


FIGURE 4.2: Problem in Existing approach and proposed solution

Figures (a)-(c) present's tasks scheduling states at various point of time along with task's affinity and opportunities for shifting of a task. (a) Primary state t_0 where P_1, P_2 and P_4 are the already scheduled tasks. (b) Possible state in that P_3 becomes ready and it can be scheduled on processor π_1 by task shifting of P1 on processor π_3. (c) Possible state where task P_3 became ready and it is scheduled on processor π_1 by shifting P_2 on processor π_3 and shifting P_1 on processor π_2

This allocation-preemption operation is making possible to execute the P_3 on π_1 which may improve the overall system schedulability as shown in FIGURE 4.2 (b). Another possible solution is presented in FIGURE 4.2 (c); here the P_1 is assigned to π_2 processor

and P_2 is assigned to π_3 processor and therefore this approach requires two preemptions with two context switches. While the approach shown in FIGURE 4.2 (b) requires one context switch only hence that is called the cost effective solution.

Case 2:

In FIGURE 4.3 (a)-(c),there is one real-time system having a task set which contains two tasks P_1 and P_2 and two processors. The task P_1 is having the lower priority than the task P_2 ($P_2 > P_1$). The processors π_1 and π_2 both are in affinity of task P_2 that is $\alpha_2 = \{\pi_1, \pi_2\}$; whereas only processor π_1 is in affinity of the task P_1 which is $\alpha_1 = \{\pi_1\}$. Consider FIGURE 4.3 (a)-(c) where initially task P_1 and P_2 both are in waiting state then at another time instant P_2 becomes ready and scheduled on processor π_1. Now see the FIGURE 4.3 (c) where the P_1 becomes ready and P_2 is already being executed on processor π_1. The Task P_1 will be blocked till task P_2 finishes its execution on π_1 as the priority of task P_2 higher than task P_1 and task P_1 can be executed only on processor π_1 due to its affinity constraint.

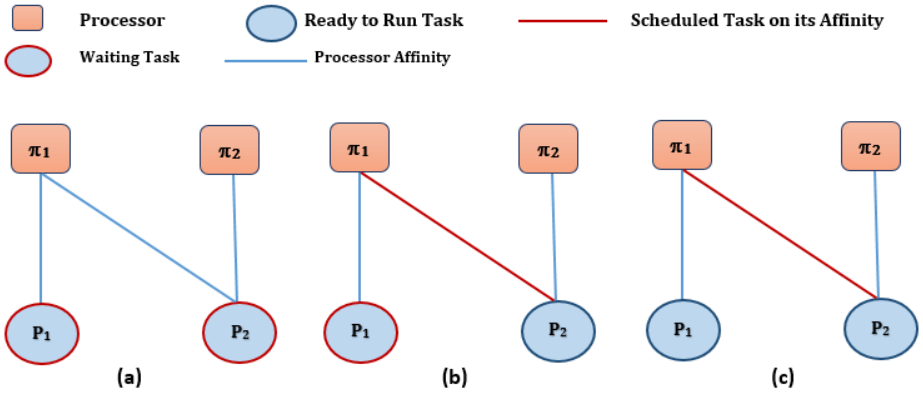


FIGURE 4.3: scheduler state at various point of time with its affinity and current state of execution. (a) Initial state at time t_0 when P_1 and P_2 both are waiting. (b) State at time t_1 when P_2 becomes ready and scheduled on π_1. (c) State at time t_2 when task P_1 becomes ready but blocked as all the processors from its affinity is busy to execute higher priority task.

These two cases illustrates that because of the restrictive migration mechanism of conventional APA based scheduling which is failing to obtain more schedulability by not preempting the higher priority running task and shifting it on another processor which is running lower priority task. The schedulability will be increased by preempting the task with low priority and after that the task with high priority will be migrated on the same CPU. To find the feasible schedule like which task will be shifted to obtain improved schedulability; here the proposed approach is used for checking opportunities of the task migration.

4.4 Pseudocode of Proposed Scheduler: PrAMS

In this section pseudo code of proposed scheduler PrAMS has been shown. The first part described the pseudo code of "Check for Priority Inversion problem" and then pseudo code of "Task Priority Reprioritization".

Algorithm 4.1: Check for Priority Inversion Problem

checking_Priority_Inversion(task_to_be_scheduled T_k)

For all π in α

If π is free or priority of process on processors < p_k

Add π to list processors_to_check

For each π_i in α_k

Task running on π_i is T_i

If $(\alpha_j \cap processors\_to\_check) \neq \emptyset$

For all tasks T_j in $(\alpha_i \cap processors\_to\_check)$

Add T_j & π_i in list tasks_to_check

If tasks_to_check list not empty

Target_Processor = lowest priority task processor from list

Target_Task = task running on Target_Processor

Algorithm 4.2: Task Priority Reprioritization

Scheduling_Backlogged_Task(Target_Task,Target_Processor,task_to_be_scheduled)

pseudo_priority = priority of Target_Task + 1

pseudo_affinity = Target_Processor

$p_{task\_to\_be\_scheduled}$ = *pseudo_priority*

$\alpha_{\text{task\_to\_be\_scheduled}}$ = Target_Processor

schedule(*cur_task*)

Set task *pseudo_affinity* and *pseudo_priority* to original

The earlier section 4.3 shows the constraints in conventional APA based scheduling approaches. The section 4.4 shows a proposed scheduler using affinity to solve the problem exists in conventional scheduler and also improves the overall system schedulability.

To illustrate the proposed approach, let's take case-1 into consideration, in which P_3 is ready to execute but the CPU π_1 which falls in P_3's affinity isn't available and executing the higher priority task P1.we can preempt P1 from π_1 only when the priority of task P_3 is higher than the priority of P1 and that is done by priority reassignment approach. According to proposed approach the priority of P_3 will be set higher than P1 so it will be preempted and P3 will be scheduled on π_1.P1 is ready to run task so it will be scheduled on processor π_3 as it is running lowest priority task P4 so it should be preempted. On completion of the push-pull (migration) operation the task P_3 will be set to its original priority. The priority ceiling protocol or priority inheritance protocol that swaps the priority of the tasks at the time of priority inversion problem so the problem can be solved and higher priority task does not have to wait for lower priority task. This solution must ensure the successful execution of all higher priority tasks.

The priority inversion problem may arise when any processor is executing the lower priority process (processor π_3 and the task P_4 in FIGURE 4.2) with respect to the task which is going to be scheduled (P_3 in FIGURE 4.2) and it is in the affinity of higher priority task (task P_1 and Processor π_1 in FIGURE 4.2) so lower priority process blocks the higher priority task from owning the resources(processor). For generalization, one can find a processor who is executing the lower priority task than the task to be scheduled and to it is in its affinity mask which is executing on processors in the affinity of the blocked task.

First of all the algorithm 4.1 try to check the probability of occurrence for the priority inversion problem. If priority inversion problem arise then the priority reassignment is done using algorithm 4.2 for increasing the overall schedulability.

Priority Inversion Problem: It appears into the system when the task to be scheduled is ready but cannot be executed as the processor in its affinity is running higher priority process and the processor which is running lowest priority process is in the affinity of that running higher priority process so the task to be scheduled is blocked and lowest priority process is running.

Once priority inversion is found simply algorithm 4.1 approach will identify a Target_Processor and a Target_Task and then Target_Task is pushed to the target_processor

only by priority reassignment that is increase the priority of task to be scheduled (T3) at kernel level.

The algorithm 4.2 is used for priority reassignments. The priority reassignment allows to assimilate solution without altering any kernel API. Another approach allows to change processor affinity of the task to be scheduled to Target_Processor so it pulls Target_Task. For example, if task to be scheduled has processor affinity $\{\pi_i, \pi_j, \pi_k\}$ and tasks on these processors are $\{p_i, p_j, p_k\}$ respectively, where $p_i > p_j > p_k$. Now Target_Task is p_j and priority of task to be scheduled is set higher than p_j. In that case, instead of preempting p_j it preempts p_k. Setting affinity to only π_j it only preempts p_j.In this research the affinity is kept fixed not dynamic as it may affect the cache performance.

Chapter 5:

LITMUS^{RT} and Experimental Results Analysis

In this chapter, firstly, the architecture of LITMUS^{RT} is shown in section 5.1, after that section 5.2 presents the taskset generation theory, the section 5.3 shows the experimental setup and result analysis of different test cases for 100 task sets are presented in section 5.4.

5.1 LITMUS^{RT}

In a real-time operating system, for simulating the various test cases of scheduling many simulators are available [25][97-99][105][117]. The wellknown RTOS are LITMUS^{RT}, Mark3, rtsim, etc. The LITMUS^{RT} is the extended version of the Linux kernel in real-time which focuses on scheduling in real-time and synchronization for multiprocessor environment. The LITMUS^{RT} is providing a valuable experimental platform for the realistic research in RTS. The aim of modifying the kernel of Linux is to provide provision for the tasks which are sporadic, scheduler plugins and reservation-based scheduling. The LITMUS^{RT} is supporting conventional multiprocessor approaches like global scheduling, partitioning scheduling, clustered scheduling, and Semi partitioned scheduling [30][88]. The LITMUS^{RT} provides such platform so researchers can design their own real-time scheduler and it can be deployed on it. LITMUS^{RT} offers various Kernel APIs which are useful to design any new real-time scheduler [26][40].

5.1.1 Design and Evolution of LITMUS^{RT}

Initially the launching of LITMUS^{RT} was done at Chapel Hill in University of North Carolina under guidance of James H. Anderson and lateron it had been benefited from the numerious researchers by their contributions over the years [40][66]. The main inventor of LITMUS^{RT} was Bjorn Brandenburg of the Max Planck Institute of Software Systems (MPI-SWS) [40]. In 2006, Calandrino et al. has developed the first model of LITMUS^{RT} by altering the Linux 2.6.9 [40] which was not made public . The first LITMUS^{RT} public version based on Linux 2.6.20 is LITMUS^{RT} 2007.1 and it was released in May 2007 [29]. LITMUS^{RT} is being constantly maintained by Björn Brandenburg and his team since 2006 to 2017 [40]. The letest version of LITMUS^{RT} is 2017.1 which is based on Linux 4.9.3 and

it released in May 2017 [40]. LITMUS^{RT}'s implementation is basically extention of Linux with additional new functionality [26].

5.1.2 The LITMUS^{RT} Architecture

The architecture of LITMUS^{RT} mainly contains four different parts as shown in FIGURE 5.1 [26]. These four parts are the core infrastructure, plugins for scheduler, the user-space interfaces and the user-space library and tools.

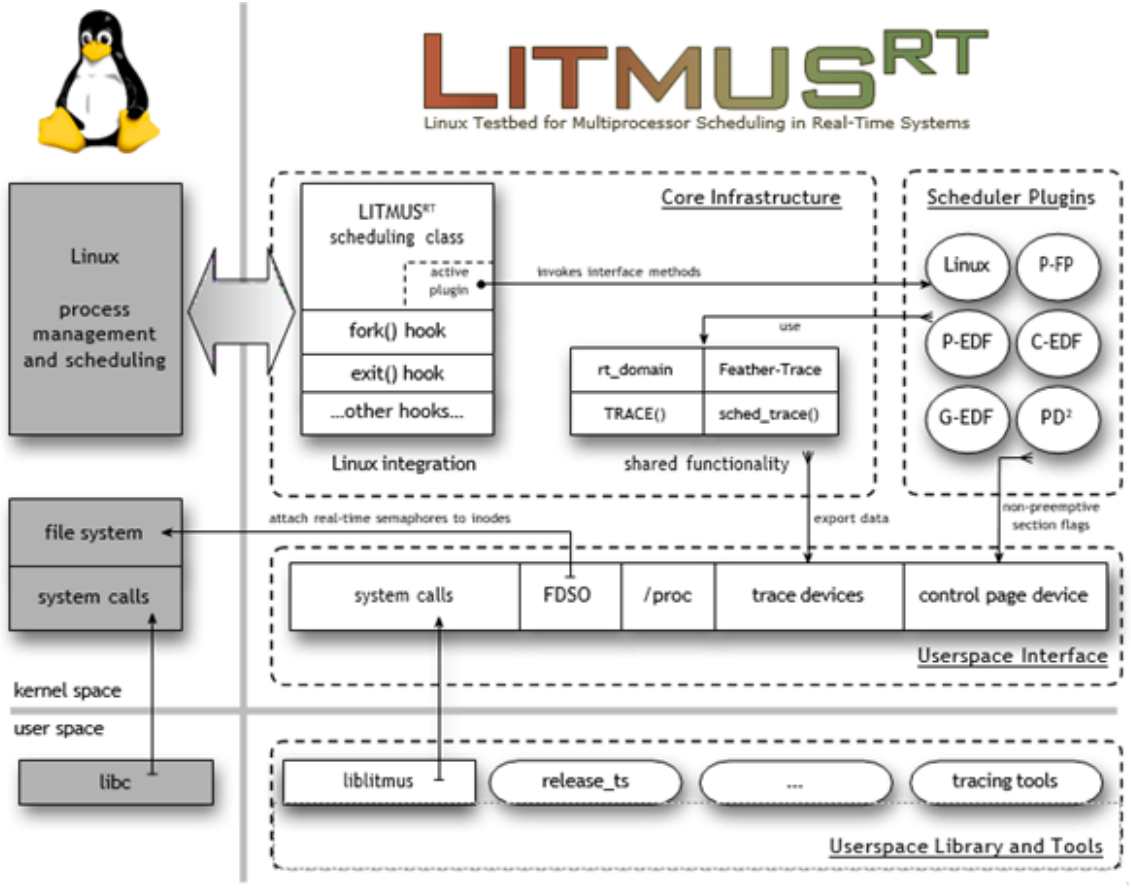


FIGURE 5.1: The LITMUS^{RT} Architecture with four important modules [26]

5.1.2.1 Core Infrastructure of the LITMUS^{RT}

The scheduling classes are the core infrastructure of the LITMUS^{RT} through which it is coupling with the framework of Linux scheduling. In the Linux scheduling hierarchy, the LITMUS^{RT} scheduling class will be added with the utmost priority for example, first_checked_scheduling_class() which is allowing LITMUS^{RT} to overrule the Linux's usual decision of scheduling [26][30]. The standard scheduling classes of Linux's were not changed since the LITMUS^{RT} was designed but only the change is made in such a way so

the tasks admitted in the LITMUS^{RT} scheduling class are being considered as real-time tasks [26][40]. Here, the advantage is that the system will behave same as a usual Linux system whenever no real-time task is present. The LITMUS^{RT} scheduling classes themselves don't implement any specific procedure but it points to the presently active scheduler for all the scheduling decisions.

5.1.2.2 Scheduler Plugins

For the development of new scheduler, LITMUS^{RT} is providing a facility of plugin interface which will allow to implement new scheduling mechanisms without going in depth of Linux kernel. The plugin for scheduler is provided for simplification in development of scheduler therefore the investigators who has not Linux expertize can develop the scheduler into a RTOS kernel [26][40]. There are few scheduler plugins shown in TABLE 5.1.

TABLE 5.1: The Methods list for LITMUS^{RT} scheduler plugins [26]

Method	Purpose
schedule()	Select next task to be schedule
tick()	Called to trigger the scheduler at the completion time of a quantum
release_at()	Prepare future job release
finish_switch()	Bookkeeping after context switch
task_block()	Remove task from the ready queue
task_wake_up()	Add task to the ready queue
admit_task()	Check if task is correctly configured or not
complete_job()	Prepare next periodic job release
task_new()	Allocate and initialize pre-process scheduler state
task_exit()	Free per-process scheduler state
activate_plugin()	Allocate and initialize plugin state
deactivate_plugin()	Free plugin state
allocate_lock()	Allocate real-time semaphore for task

5.1.2.3 User-Space Interface

LITMUS^{RT} provides many extra system calls along with some virtual character devices. These system calls are classified into different categories based on its functionality as shown below [26]:

1. get and set the sporadic task parameters like ei, pi, and di and CPU assignment
2. Controlling jobs of the real-time Tasks
3. overhead measurement of the system calls
4. create, lock, and unlock of real-time semaphores
5. to provide provision for task set releases synchronously

The implementation and purpose of the categories 1-3 are straightforward; here short description of the categories 4 and 5 are as following:

- **Real-time semaphores:** To enforce the priority ceiling and priority inheritance protocols with Blocking-by-suspending needs kernel support. So, every shared resource can be molded as entity inside the kernel that will holds current statistics such as the maximum assigned priority,unfulfilled requests, *etc.* All tasks which will share a resource should acquire the same location to object inside the same kernel space. A design space of the applications so it can be transparent and accessible to all the task boundaries that means the separate address space must not prevent sharing of resources as LITMUS^{RT} is not putting constraints on it.

- **Release of the task set synchronously [26]:** The synchronous release of the task set means that the first job for every task in that task set will be dispatched at the similar point of time. LITMUS^{RT} provides the additional API ts_release( ) for synchronous task set release [26][40] and that API will make them possible to hang up till the tasks release happens synchronously. Notably, the numerous real-time tasks in waiting state are exported using the kernel in the format of a virtual file that will be located inside the file system called proc [40].

- **Virtual devices:** There are three types of virtual devices provided by the LITMUS^{RT} which are located in the proc file system of the Linux; The first one is the TRACE( ) device which is to debug the data; second on is the Feather_Trace( ) which is used to calculate the system overheads and sched_trace( ) for tracing the scheduling events

[26][28][31][40]. In LITMUS^{RT}, Feather-Trace( ) is used for recording the timestamp during the execution of the schedulers at different time instances [26][40]. The couples of timestamps before and after an event can be recorded and based on that the overhead can be calculated by checking the total time taken by an event. By checking this overhead, the statistics can be derived like average overhead and maximum overheads. Using the sched_trace( ) framework the tracing of whole event will be recorded since starting of event to the end of the events for example job statistics like processor on which it has been executed, deadline misses, tardiness, job releases, preemptions, job completions, etc. [26]. By applying the sched_trace( ) whatever data we can derive are not directly understandable to the person so it can be plotted in the form of graph for visualization using evince( ) and st_draw( ) of the TRACE( ) and it can be also used for finding the scheduling errors automatically [28].

5.1.2.4 Library and Tools for Userspace

In LITMUS^{RT}, real-time tasks are like usual processes of the Linux. The libLitmus, a library for user-space includes the necessary system APIs and numerous suitable methods and macros which is making easy real-time task programming easy. Moreover, there are a numerous testing and helping utilities for communicating with the LITMUS^{RT} kernel are based on libLitmus and the release_ts( ) tool is the important example of it which will be useful in attaining the synchronous release of the set of tasks [26][40]. Additionally, the LITMUS^{RT} provides some mechanisms which communicate to trace device and the collected data post-processing will be performed [30][40]. With this we conclude our brief overview for the LITMUS^{RT} [26].

5.2 Taskset Generation Theory

To know the perfect efficiency and effectiveness of any algorithm, there should be a remarkable task set as an input. This section describes the Real-Time Task model and basic theory for task set generation.

5.2.1 Taskset Model

In Real time systems taskset is set of n different tasks $\tau = \{T_1, T_2, \ldots, T_n\}$ and each task can be characterized by $T_i = (c_i, d_i, p_i)$ as mentioned in Liu Layland model [89], where c_i stands for the worst_case_execution_time, d_i stands for the relative deadline, and p_i is a

period of the task. The task can have either implicit deadline of the tasks are same as its period ($d_i = p_i$) or explicit deadline ($d_i <$) or any arbitrary deadline [55][68][89]. The task_utilization U_i of a task T_i can be defined as $\frac{c_i}{p_i}$ and the task set utilization can be defined as $U_\tau = \sum_{i=1}^{n} \frac{c_i}{p_i}$.

For generating any real-time task required parameters are WCET(c_i), period (p_i), deadline(d_i) and its priority.

5.2.2 Taskset Utilization for generating tasks

Consider FIGURE 5.2 showing results of taskset utilization for 4 processor and 7 tasks. Task set utilization on X-axis ranges from [0.05, 4] with incremental steps of 0.05. Maximum value is 4 as it is maximum achievable value with 4 processors. For each utilization point we can generate multiple task sets and each task set needs to be different than another [15].

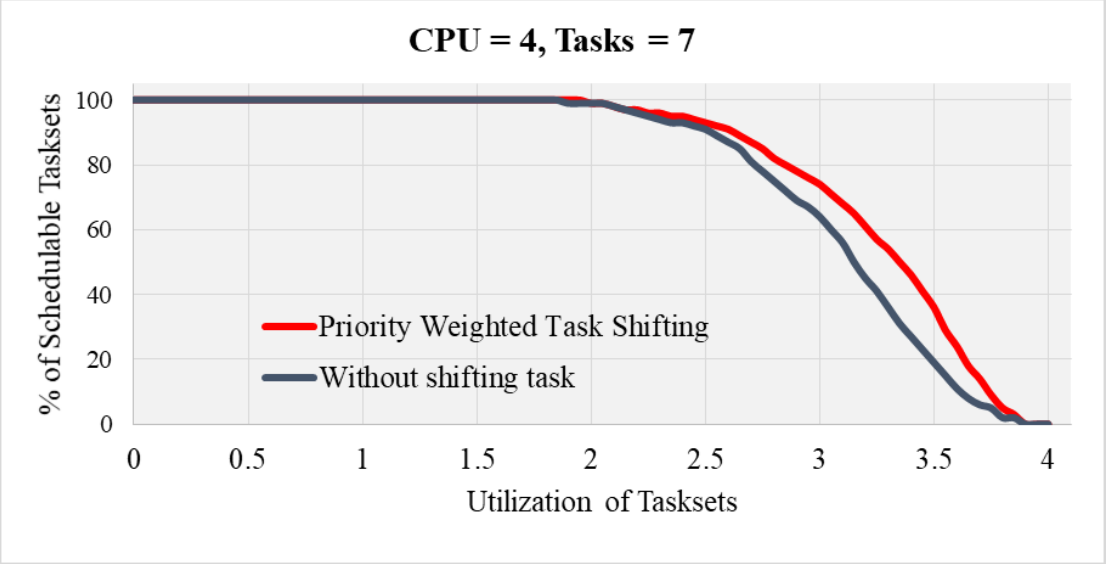


FIGURE 5.2: Understanding the Taskset Utilization for experimental setup [15]

In the further sections, the taskset generation for single taskset with utilization = m has been taken into consideration. Similar method can be used to generate any taskset. In each taskset we have taken n tasks and total task_utilization of the taskset is m. This requirement leads to not only creating n random tasks but also such n random tasks whose sum of utilization is m.

5.2.3 Generating WCET

Let us assume the case where utilization value is m in FIGURE 5.2. In this case we need to create such tasksets which have n tasks with total utilization equals m. Using R. Stafford's algorithm "Random Vectors with fixed Sum", we can create vector of n numbers having sum m [112]. Each of these n random numbers represent utilization value of one task. The utilization of a single task T_i can be represented as $U_i = \frac{c_i}{p_i}$. If we have already generated random values of period p_i and the values of U_i. Then one can calculate the value of c_i, which are random and not biased. Implementation and explanation of this algorithm are available online in provided reference and this algorithm is widely used by researchers to generate random values with fixed sum.

5.2.4 Priority Assignment

The schedulers can assign the priorities to different jobs/ tasks of a task set and it may be changed during its execution period. Particularly, various priority allotment mechanisms exists in real-time system scheduling are categorized in three classes;

1) **Fixed (Static) priority Approach (FP):** In the Fixed Priority approach, the algorithm will allocates an exclusive priority level for every task and it remains same for all the jobs of that task during the whole execution; the examples for this group are the Rate Monotonic (RM) [89] and Deadline Monotonic (DM) [11][87].

2) **Job-Level Fixed priority Approach (JLFP):** In the JLFP approach, the algorithm allocates the static priority level to every job but the different jobs of the same task can contain different priority levels. An example of this group is the Earliest Deadline First (EDF) priority assignment policy [89].

3) **Job-level dynamic priority Approach (JLDP):** In the JLDP approach, a job is permitted to carry different priorities during its whole execution time (one job is having different priority at different time of instance); The example of this group is the Least_Laxity_First algorithm [48].

Processor selection with affinity scheduling may be used along with any of the above discussed priority allotment mechanism. This research has compared processor affinity based scheduler with partitioned and global scheduler along with FP and JLFP policies respectively. The proposed Scheduler PrAMS tested with JLDP policy. The comparison between P-FP-APA-APA, G-JLFP-APA and PrAMS-JLDP is done for five different parameters.

The Heuristic used for priority assignment is the DkC priority assignment [46][106]. Deadline d_i for each task is random value generated either implicit deadline which is $d_i = p_i$ or it is constrained deadline $d_i < p_i$. Authors suggested using DkC heuristic which outperforms all other priority assignment method in terms of generating effective utilization values.

Here DkC means, priority is $d_i - k*c_i$, where k is the important variable which is calculated depending on the processor numbers in the system and formula for that is shown as follows [106]:

$$K = \frac{(2 * m - 1) + \sqrt{(5 * m^2) - (6 * m) + 1}}{(2 * m)}$$

5.3 Experimental Setup

In this research, experiment and result analysis is done for five parameters that is deadline miss ratio, tardiness, number of context switches, schedulability analysis and CPU_Utilization for multiprocessor real-time system. TABLE 5.2 presents task set generation criteria and the value taken into consideration. The SimSo API is used for Task set generation along with "Random Vectors with fixed Sum" algorithm as explained in section 5.2.3 [42]. Taskset size n is taken random within the range (m+1 to 4m). To implement the different schedulers and to perform experiments the LITMUS^{RT} simulator has been used [24][25][30][105]; the scheduler analysis is done by the Sched_trace APIs of the LITMUS^{RT} and trace analysis is done using Feather_Trace APIs of LITMUS^{RT}. Lib LITMUS Library used for various APIs [24][30]. The random task periods has been taken in range of (10ms to 100ms) using the log-uniform period allocation method [46]. The Affinity assignment is static using graph search approach [24][25] [68]. In this research we have taken constrained deadline that is ($d_i < p_i$) not taken implicit deadline.

In this research, the Job-Level-Dynamic Priority assignment along with PrAMS approach and for that the DkC priority assignment approach is used as mentioned in section 5.2.4 [46]. The section 5.4 presents the result analysis for various cases in which PrAMS compared with existing methods for 100 Task sets with different Task set size (n = m+1 to 4m) and the No. of Processors(m=8,12,16).

TABLE 5.2: Task Set Generation Criteria and the value taken into consideration

Criteria	Value
Type of RTS	Soft Real-Time System
Priority Assignment	JLDP
Task set Count	100
Running time	60 seconds
Period Range	10ms to 100ms
Task set Generation Algorithm	Random Vectors with fixed Sum algorithm
No of processors	8,12,16
The task set size range	(m+1 to 4m)
Task Utilization	For all Taskset more than 75% of m
Processor Selection	Static Affinity Assignment
Relative Deadline	$d_i < p_i$
Migration	Within its affinity with priority reprioritization

5.4 Experimental Results Analysis

In this section the results for PrAMS on 100 Tasksets with Random Taskset size (n= (m+1 to 4m)) and more number of Processors (m= 8, 12, 16) along with three different test cases have been considered for five different performance measure parameters which are mentioned below:

Case-1: Existing Schedulers and PrAMS with Fixed No. of Processors and Variable Tasksetsize (m+1,4m)

In this case we have kept fix no. of processors (m) and the variable taskset size (n) has been taken into consideration. The taskset size is kept random with the range from m+1 to 4m (where the m is no. of processors and n is taskset size).

There are three different test categories have been selected.

1. The test has been taken with fixed No. of Processors 8 and variable taskset size (n=10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32).
2. The test has been taken with fixed No. of Processors 12 and variable taskset size (n=16, 20, 24, 28, 32, 36, 40, 44, 48).
3. The test has been taken with fixed No. of Processors 16 and variable taskset size (n=20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64).

There are five parameters considered for the results.

- Parameter 1: Average Deadline Miss Ratio should be minimum.
- Parameter 2: Average Tardiness (ns) should be as low as possible.
- Parameter 3: Average No. of Context Switches should be minimum.
- Parameter 4: Average Schedulability (%) should be as high as.
- Parameter 5: Average CPU_Utilization (%) should be maximum.

Here, following three different scheduling algorithms have been compared.

1. P-FP- APA (Partitioned Approach with fixed priority using Affinity)
2. G-JLFP-APA (Global Approach with Job-Level-Fixed priority using Affinity)
3. Proposed Approach (PrAMS along with affinity for processor selection, job level dynamic priority for priority assignment and flexible migration policy using priority reprioritization)

All the tests have been taken on LITMUS^{RT}. The details about the simulator LITMUS^{RT} has been discussed in section 5.1.

The results mentioned here is the average of 100 Tasksets. One Taskset can contains any no. of tasks with the range m+1 to 4m and many jobs of each task can appear within the given duration as per the given period. Here results are shown is the average of 100 tasksets. Task generation detail is given in TABLE 5.2. The Detailed results of 100 attempts for each case have been mentioned in Annexure-I, Annexure-II and Annexure-III.

5.4.1 Average Deadline Miss Ratio (Fixed No. of Processors and Vary Taskset size)

1. Average of 100 Tasksets with Fixed no. of Processors 8 and variable Taskset Size

TABLE 5.3: Average Deadline Miss Ratio for 100 Tasksets with 8 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	0.0945	0.0696	0.0226
m=8, n=12	0.1449	0.1229	0.0847
m=8, n=14	0.2048	0.1841	0.1447
m=8, n=16	0.2661	0.2538	0.2160
m=8, n=18	0.3963	0.3248	0.2872
m=8, n=20	0.4666	0.4141	0.3663
m=8, n=22	0.5440	0.4927	0.4357
m=8, n=24	0.6144	0.5739	0.5238
m=8, n=26	0.6558	0.5955	0.5359
m=8, n=28	0.6975	0.6261	0.5862
m=8, n=30	0.7837	0.7059	0.6754
m=8, n=32	0.8842	0.7971	0.7762

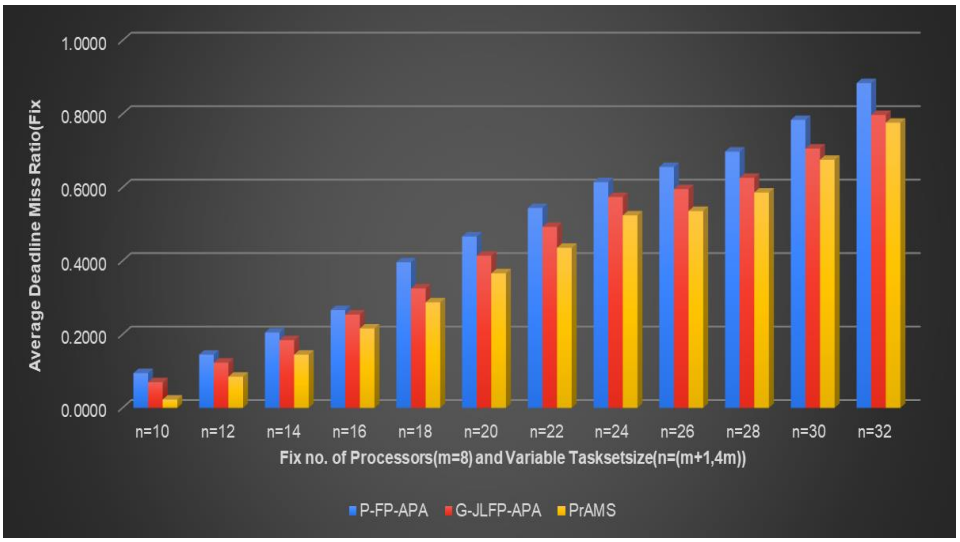


FIGURE 5.3: Average Deadline Miss Ratio with 8 Processors

As shown in FIGURE 5.3, the PrAMS is having minimum **Deadline miss ratio** as compared to all other multiprocessor schedulers, and the P-FP-APA has missed the maximum deadlines. It also shows that as taskset size increases the deadline miss ratio is also increasing for all the schedulers.

2. Average of 100 Tasksets with Fixed no. of Processors 12 and variable Taskset Size

TABLE 5.4: Average Deadline Miss Ratio for 100 Tasksets with 12 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=12, n=16	0.1941	0.0929	0.0468
m=12, n=20	0.2863	0.2444	0.1476
m=12, n=24	0.4011	0.3745	0.2876
m=12, n=28	0.5342	0.4856	0.3823
m=12, n=32	0.6739	0.6033	0.4866
m=12, n=36	0.7344	0.6962	0.6245
m=12, n=40	0.8061	0.7835	0.7160

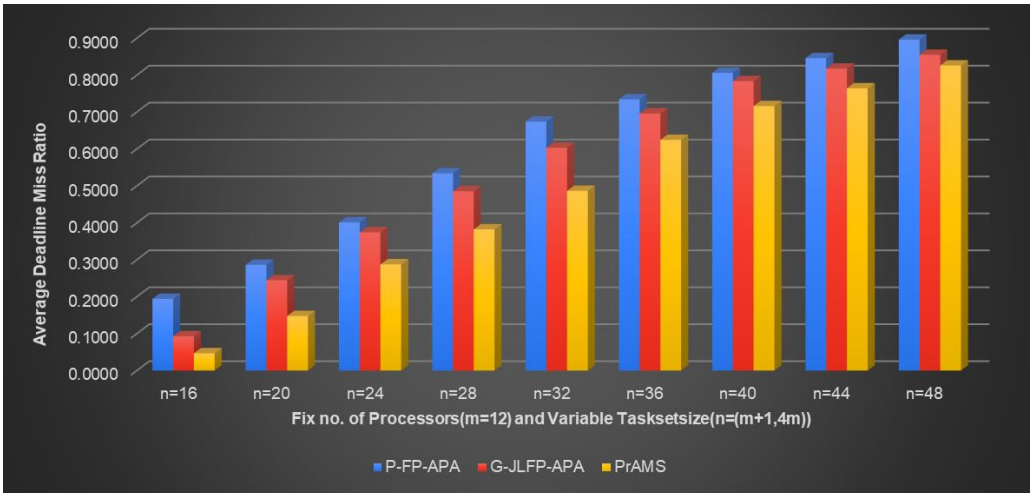


FIGURE 5.4: Average Deadline Miss Ratio with 12 Processors

As shown in FIGURE 5.4, the PrAMS meets the maximum number of deadlines so it has the lowest **Deadline miss ratio** as compared to all other schedulers. The P-FP-APA has missed the maximum no. of deadlines so its deadline miss ratio is high. It also shows that as taskset size increases the deadline miss ratio is also increasing for fixed no. of processors.

3. Average of 100 Tasksets with Fixed no. of Processors 16 and variable Taskset Size

TABLE 5.5: Average Deadline Miss Ratio for 100 Tasksets with 16 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=16, n=20	0.0873	0.0453	0.0034
m=16, n=24	0.1427	0.0938	0.0360
m=16, n=28	0.1963	0.1664	0.0852
m=16, n=32	0.2841	0.2450	0.1472
m=16, n=36	0.3723	0.3231	0.2043
m=16, n=40	0.4548	0.3960	0.2845
m=16, n=44	0.5330	0.4851	0.3560
m=16, n=48	0.6143	0.5450	0.4233
m=16, n=52	0.6835	0.6418	0.5422
m=16, n=56	0.7172	0.6755	0.6243
m=16, n=60	0.7339	0.6919	0.6458
m=16, n=64	0.7580	0.7407	0.7335

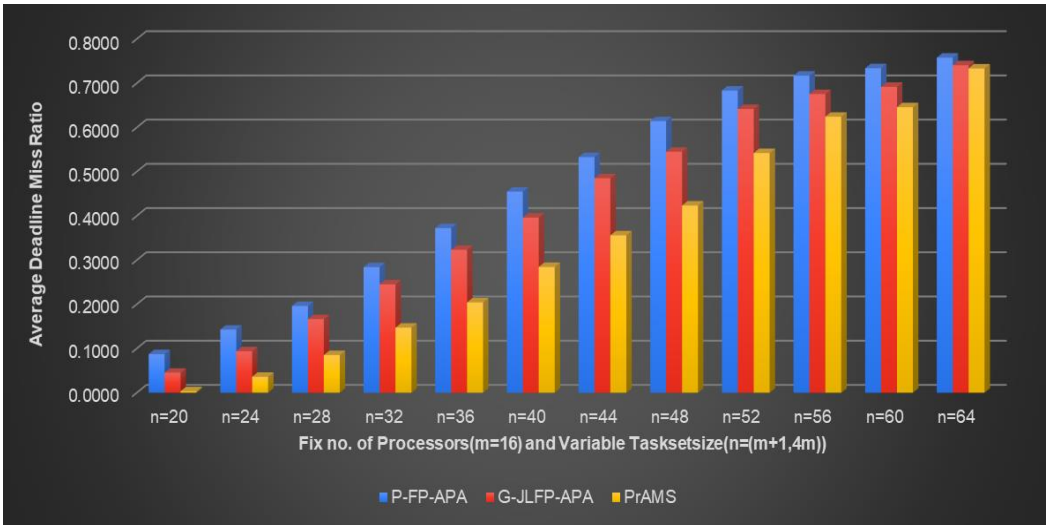


FIGURE 5.5: Average Deadline Miss Ratio with 16 Processors

As shown in FIGURE 5.5, the PrAMS is giving the result as per the expectation that is minimum **Deadline miss ratio** as compared to all other multiprocessor schedulers and the P-FP-APA has the deadlines miss ratio more than G-JLFP-APA and PrAMS both as P-FP-APA provides fixed priority and processors are strictly partitioned between all the tasks. Here also the deadline miss ratio increasing as the no. of tasks in a taskset is being increased.

5.4.2 Average Tardiness (Fixed No. of Processors and Vary Taskset size)

1. Average of 100 Tasksets with Fixed no. of Processors 8 and variable Taskset Size

TABLE 5.6: Average Tardiness for 100 Tasksets with 8 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	19.51	11.36	8.51
m=8, n=12	21.49	15.38	10.33
m=8, n=14	23.53	19.72	13.50
m=8, n=16	24.45	20.07	14.58
m=8, n=18	25.49	20.54	15.41
m=8, n=20	25.89	20.94	15.71
m=8, n=22	26.19	21.15	15.99
m=8, n=24	27.05	21.59	16.90
m=8, n=26	27.46	21.67	17.38
m=8, n=28	28.32	22.16	18.24
m=8, n=30	28.57	23.53	18.79
m=8, n=32	31.38	26.63	21.55

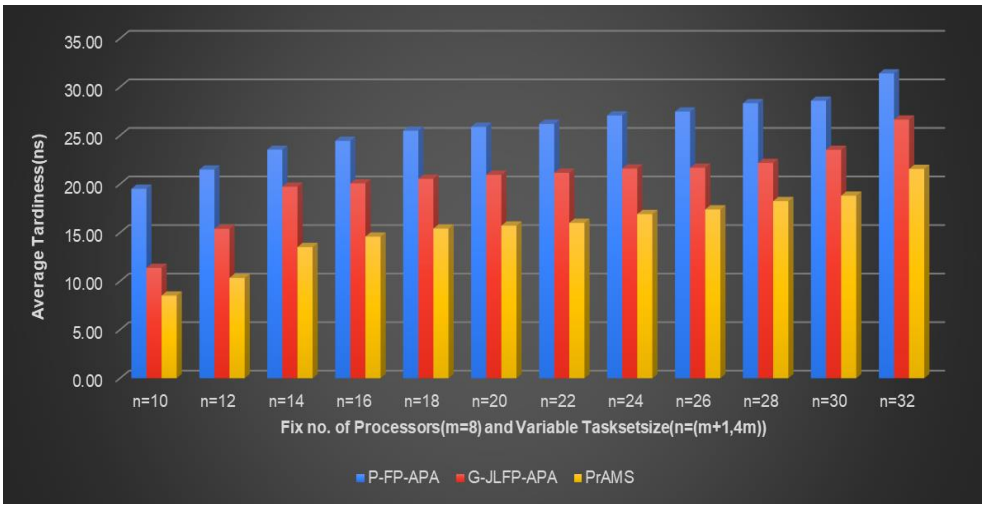


FIGURE 5.6: Average Tardiness with 8 Processors

As shown in FIGURE 5.6, the PrAMS is having less **Tardiness** as compared to other schedulers like G-JLFP-APA and the P-FP-APA as the PrAMS' flexible migration policy making more task schedulable and reducing the deadline misses. The P-FP-APA has given the maximum tardiness. It can be observed that the tardiness is increasing as the taskset size is being increased.

2. Average of 100 Tasksets with Fixed no. of Processors 12 and variable Taskset Size

TABLE 5.7: Average Tardiness for 100 Tasksets with 12 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=12, n=16	15.60	13.59	10.68
m=12, n=20	22.42	20.43	14.64
m=12, n=24	26.10	21.32	15.49
m=12, n=28	27.52	21.86	17.67
m=12, n=32	29.56	24.56	19.38
m=12, n=36	32.39	28.34	22.44
m=12, n=40	32.56	30.61	24.54
m=12, n=44	34.35	29.53	25.85
m=12, n=48	36.54	28.44	27.90

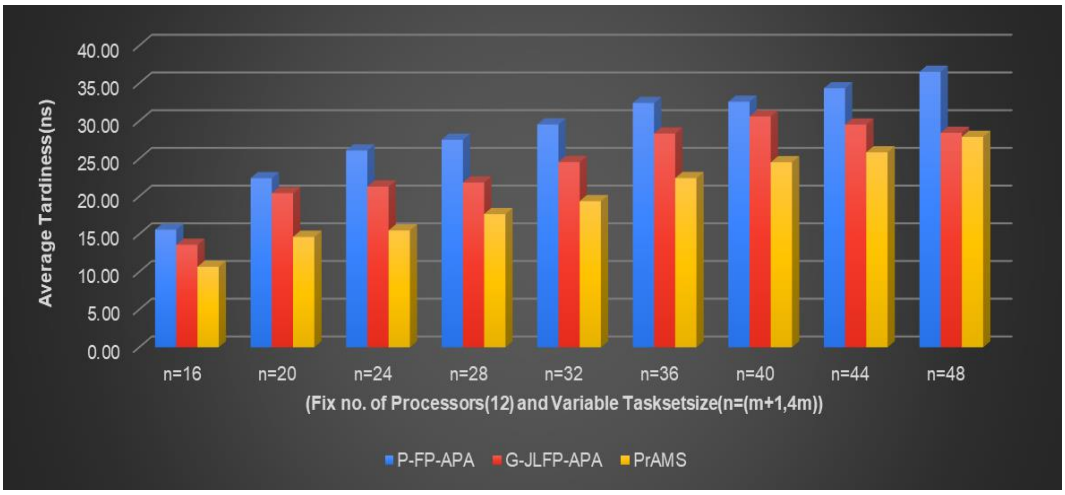


FIGURE 5.7: Average Tardiness with 12 Processors

As shown in FIGURE 5.7, the PrAMS gives the less **Tardiness** as compared to all other schedulers so PrAMS improves all over system performance as the more tardiness means time taken after deadline is more and in the SRTS the performance degradetion happens in proportion to the time elased after the deadline. It also shows that the tardiness is increasing as the no. of tasks in a taskset size is being increased.

3. Average of 100 Tasksets with Fixed no. of Processors 16 and variable Taskset Size

TABLE 5.8: Average Tardiness for 100 Tasksets with 16 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=16, n=20	14.04	13.08	11.09
m=16, n=24	16.39	15.27	12.64
m=16, n=28	19.37	17.44	14.59
m=16, n=32	21.54	20.33	16.66
m=16, n=36	23.63	22.38	18.31
m=16, n=40	26.50	24.81	20.38
m=16, n=44	28.46	25.53	22.45
m=16, n=48	31.70	27.46	23.47
m=16, n=52	34.35	29.54	25.37
m=16, n=56	34.78	30.33	28.61
m=16, n=60	35.71	32.15	31.53
m=16, n=64	38.43	33.73	31.59

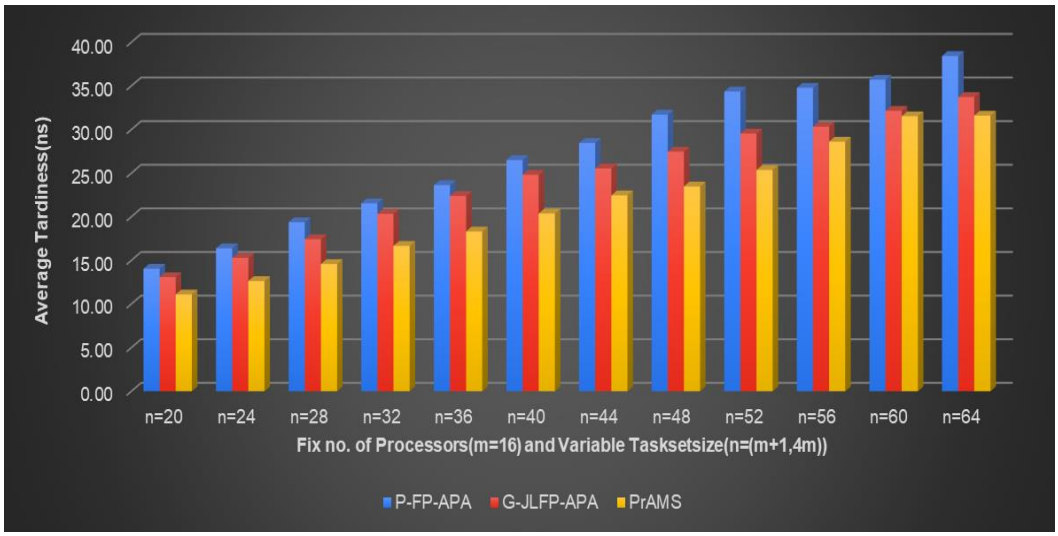


FIGURE 5.8: Average Tardiness with 16 Processors

As shown in FIGURE 5.8, in this case also the PrAMS is giving less **Tardiness** as compared to the all other multiprocessor schedulers due to the PrAMS' flexible migration policy which is making more no. of tasks schedulable and reducing the deadline misses. It can be seen that as the taskset size is being increased the tardiness is also increasing.

5.4.3 Average No. of Context Switches (Fixed No. of Processors and Vary Tasksetsize)

1. Average of 100 Tasksets with Fixed no. of Processors 8 and variable Taskset Size

TABLE 5.9: Average No. of Context Switches for 100 Tasksets with 8 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	59.48	67.56	71.24
m=8, n=12	71.45	83.53	81.52
m=8, n=14	83.54	99.23	91.60
m=8, n=16	87.60	107.73	99.38
m=8, n=18	91.47	114.16	106.57
m=8, n=20	93.26	118.38	114.37
m=8, n=22	96.55	120.38	121.45
m=8, n=24	98.41	125.61	129.53
m=8, n=26	106.63	153.18	132.33
m=8, n=28	113.51	197.81	134.64
m=8, n=30	143.40	209.59	169.45
m=8, n=32	165.77	213.77	193.47

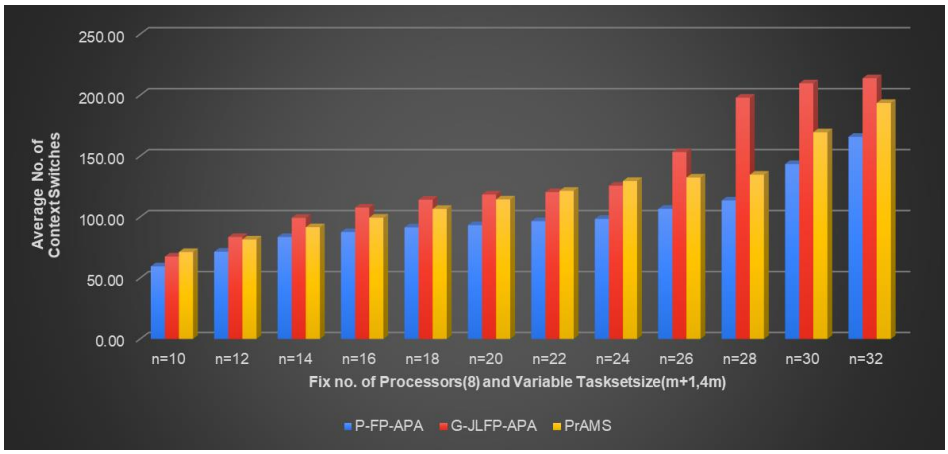


FIGURE 5.9: Average No. of Context Switches with 8 Processors

As shown in FIGURE 5.9, In most of the cases, the PrAMS is having less no. of **Context switches** as compared to G-JLFP-APA but always more than P-FP-APA because PrAMS provides flexible migration policy which is increasing no. of context switches. It can be observed that in some cases PrAMS is having more no. of context switches than G-JLFP-APA also. One more observation is that as the taskset size is being increased the no. of context switches are also increasing.

2. Average of 100 Tasksets with Fixed no. of Processors 12 and variable Taskset Size

TABLE 5.10: Average No. of Context Switches for 100 Tasksets with 12 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=12, n=16	88.73	107.67	123.99
m=12, n=20	112.32	178.62	156.35
m=12, n=24	139.74	189.29	167.25
m=12, n=28	154.52	199.69	180.54
m=12, n=32	172.58	219.77	203.31
m=12, n=36	180.45	224.38	231.64
m=12, n=40	194.25	244.45	268.29
m=12, n=44	215.77	278.59	279.70
m=12, n=48	236.62	312.23	289.87

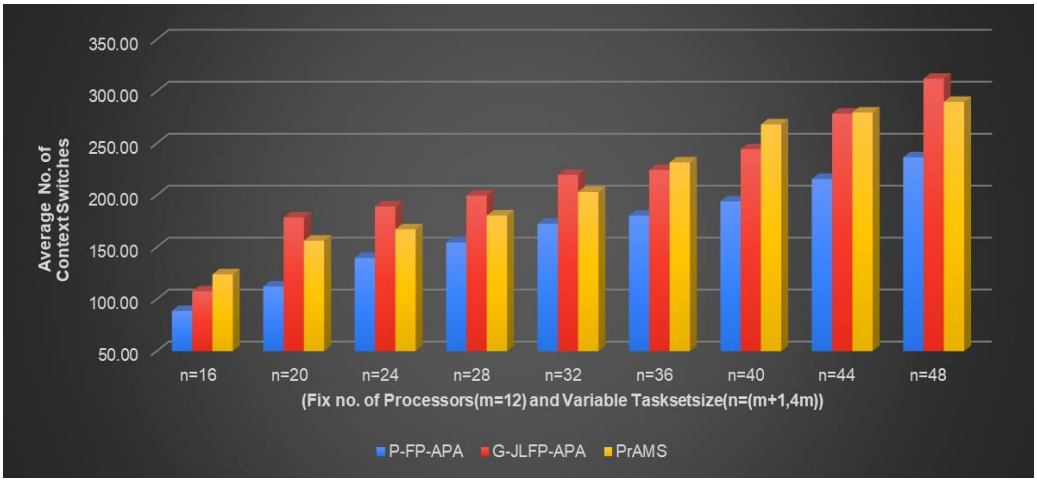


FIGURE 5.10: Average No. of Context Switches with 12 Processors

The FIGURE 5.10 shows that in all the cases the PrAMS has taken more no. of **Context switches** as compared to P-FP-APA but in most of the cases it is less than the G-JLFP-APA due to PrAMS' flexible migration policy and P-FP-APA has almost no migration kind situation as the processors are strictly partitioned for all the tasks into the system. Another conclusion is that the no. of context switches are getting increased as the taskset size and the no. of processors are increased.

3. Average of 100 Tasksets with Fixed no. of Processors 16 and variable Taskset Size

TABLE 5.11: Average No. of Context Switches for 100 Tasksets with 16 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=16, n=20	118.5	203.54	189.40
m=16, n=24	145.23	225.21	205.34
m=16, n=28	165.71	246.63	221.51
m=16, n=32	200.06	288.53	254.76
m=16, n=36	234.27	325.45	287.50
m=16, n=40	244.60	339.26	313.55
m=16, n=44	253.39	354.22	337.94
m=16, n=48	263.79	368.23	364.58
m=16, n=52	272.91	382.59	389.20
m=16, n=56	297.62	397.92	407.11
m=16, n=60	321.66	412.43	414.60
m=16, n=64	346.90	427.85	438.22

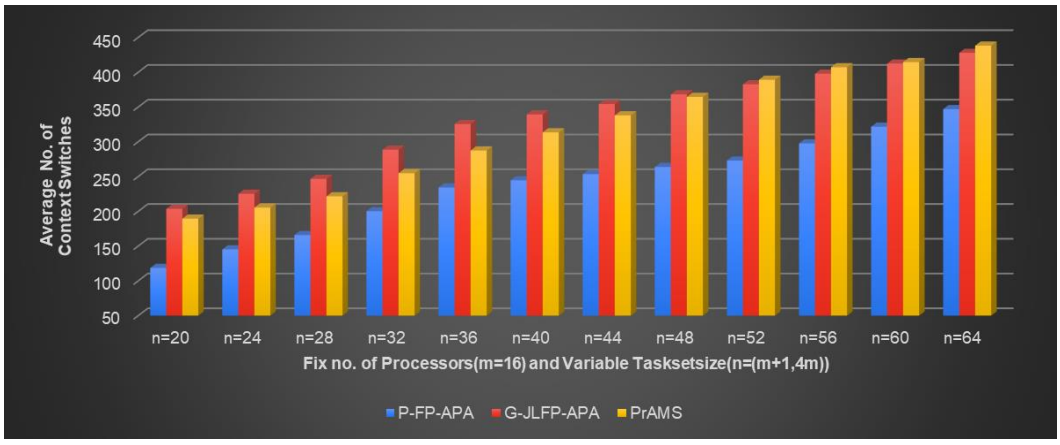


FIGURE 5.11: Average No. of Context Switches with 16 Processors

The FIGURE 5.11 shows that in all the cases the PrAMS and G-JLFP-APA have taken more no. of **Context switches** as compared to P-FP-APA as PrAMS provides flexible migration by task shifting with the priority reassignment. Except few cases the PrAMS has taken less no. of context switches than G-JLFP-APA. One more thing in this case also the no. of context switches increased as the taskset size and the no. of processors are increased.

5.4.4 Average Schedulability (Fixed No. of Processors and Vary Taskset size)

1. Average of 100 Tasksets with Fixed no. of Processors 8 and variable Taskset Size

TABLE 5.12: Average Schedulability for 100 Tasksets with 8 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	82.020	86.340	89.622
m=8, n=12	79.810	83.855	87.382
m=8, n=14	77.600	81.370	84.685
m=8, n=16	76.575	80.025	84.446
m=8, n=18	75.550	78.680	83.340
m=8, n=20	74.517	77.580	83.200
m=8, n=22	73.483	76.480	82.407
m=8, n=24	72.450	75.380	81.457
m=8, n=26	70.445	73.695	79.679
m=8, n=28	68.440	72.010	77.629
m=8, n=30	64.970	68.385	75.452
m=8, n=32	61.500	64.760	73.525

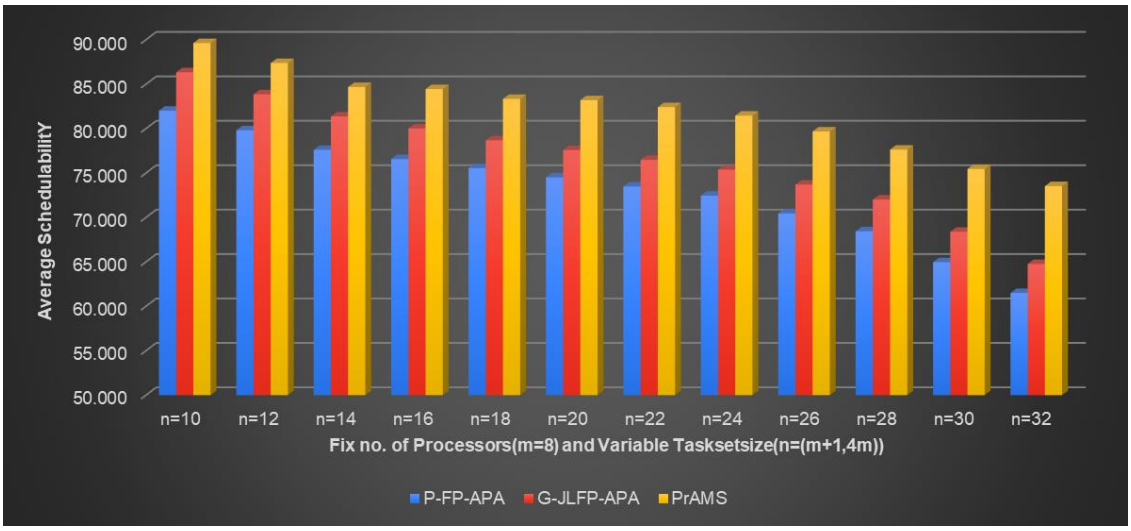


FIGURE 5.12: Average Schedulability with 8 Processors

As shown in FIGURE 5.12, in all the cases, the PrAMS has the more **Schedulability** as compared to P-FP-APA and G-JLFP-APA both which is the main goal of any real-time system. For all the cases P-FP-APA has less schedulability. It is also observed that the schedulability is decreasing as no. of task in a taskset is increasing with the fixed no. of processors.

2. Average of 100 Tasksets with Fixed no. of Processors 12 and variable Taskset Size

TABLE 5.13: Average Schedulability for 100 Tasksets with 12 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=12, n=16	86.54	88.49	93.31
m=12, n=20	82.44	87.72	89.51
m=12, n=24	78.50	81.55	86.37
m=12, n=28	78.45	80.64	84.45
m=12, n=32	77.66	79.62	82.40
m=12, n=36	75.45	78.42	80.69
m=12, n=40	74.78	77.70	78.46
m=12, n=44	70.67	75.45	77.67
m=12, n=48	66.64	73.64	75.37

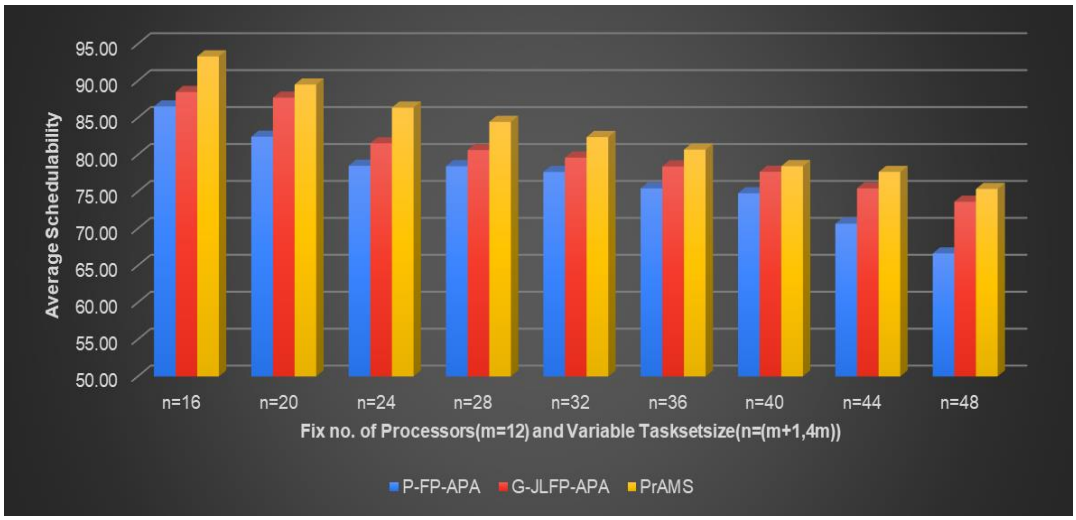


FIGURE 5.13: Average Schedulability with 12 Processors

As shown in FIGURE 5.13, the PrAMS gives the more **Schedulability** that is more no. of tasks get executed before its deadline as compared to all other multiprocessor scheduler due to PrAMS' flexible migration policy. It also shows that the schedulability is decreasing as the no. of tasks in a taskset size is being increased with fixed no. of processors but if it is compared with FIGURE 5.14, it can be concluded that the schedulability is increasing if no. of processor are increased.

3. Average of 100 Tasksets with Fixed no. of Processors 16 and variable Taskset Size

TABLE 5.14: Average Schedulability for 100 Tasksets with 16 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=16, n=20	87.400	89.587	94.286
m=16, n=24	86.599	88.571	92.236
m=16, n=28	84.312	87.401	91.568
m=16, n=32	81.657	86.280	89.358
m=16, n=36	78.645	84.500	87.453
m=16, n=40	78.483	83.610	86.632
m=16, n=44	77.552	82.466	85.542
m=16, n=48	76.676	81.281	84.658
m=16, n=52	75.416	80.449	83.444
m=16, n=56	75.232	79.420	82.446
m=16, n=60	74.500	79.302	81.490
m=16, n=64	69.480	75.572	77.502

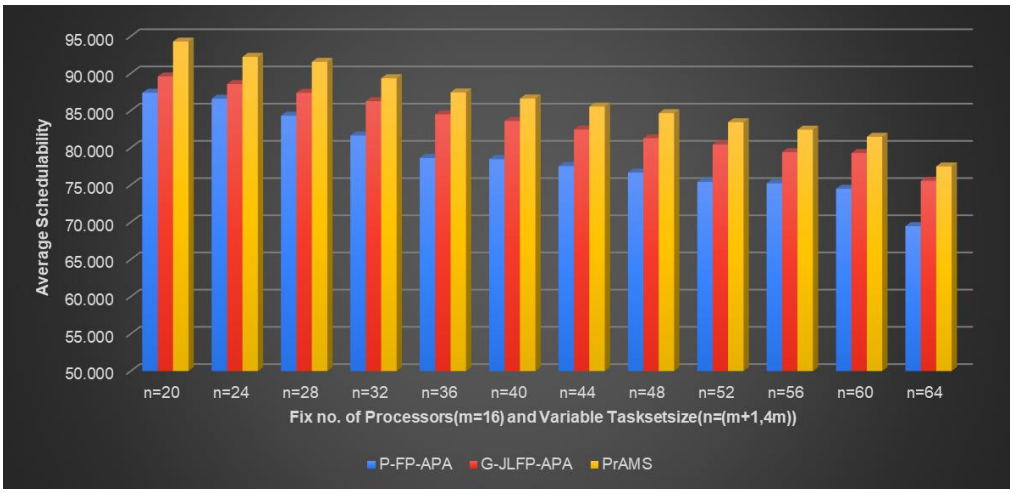


FIGURE 5.14: Average Schedulability with 16 Processors

The FIGURE 5.14 shows that in all the cases the PrAMS increases overall **Schedulability** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest schedulability. One more thing in this case also is the schedulability decreased as the taskset size is increased and the schedulability increased with the no. of processors are increased.

5.4.5 Average CPU_Utilization (Fixed No. of Processors and Vary Taskset size)

1. Average of 100 Tasksets with Fixed no. of Processors 8 and variable Taskset Size

TABLE 5.15: Average CPU_Utilization for 100 Tasksets with 8 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	59.28	64.37	66.32
m=8, n=12	59.44	64.50	67.39
m=8, n=14	60.35	65.43	68.32
m=8, n=16	61.38	67.34	70.39
m=8, n=18	61.60	69.35	72.30
m=8, n=20	62.46	70.36	73.29
m=8, n=22	63.50	70.53	73.43
m=8, n=24	64.61	71.50	74.25
m=8, n=26	67.41	73.33	76.45
m=8, n=28	69.42	75.81	78.54
m=8, n=30	71.51	77.51	79.34
m=8, n=32	73.42	79.34	80.46

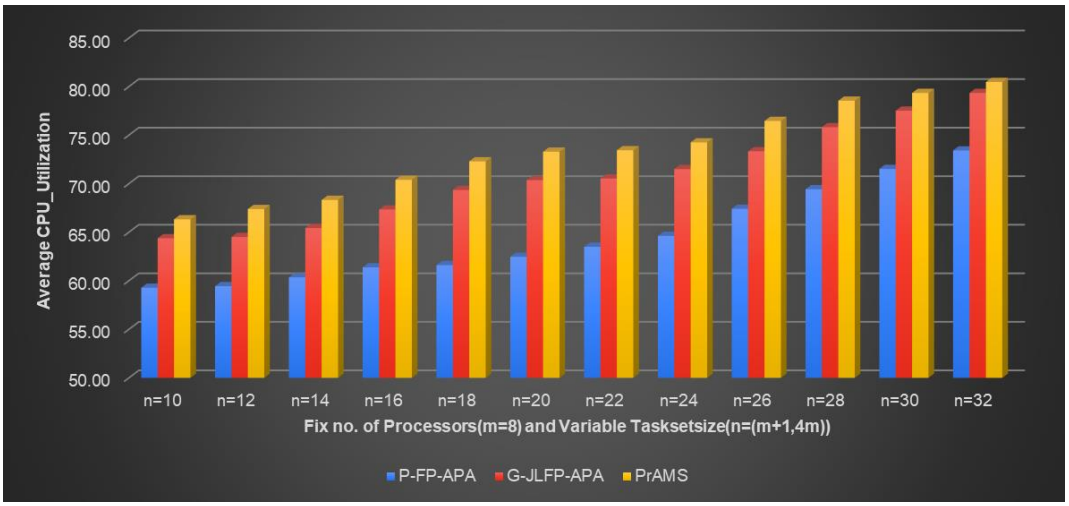


FIGURE 5.15: Average CPU_Utilization with 8 Processors

As shown in FIGURE 5.15, in all the cases, the PrAMS has the more **CPU_Utilization** as compared to P-FP-APA and G-JLFP-APA both as the PrAMS' schedulability improvement increases overall execution of tasks which utilize the more no. of CPUs. P-FP-APA is having minimum CPU_Utilization. For all the cases P-FP-APA has less schedulability. It is also observed that the CPU_Utilization is increasing as no. of tasks in a taskset is increasing with the fixed no. of processors.

2. Average of 100 Tasksets with Fixed no. of Processors 12 and variable Taskset Size

TABLE 5.16: Average CPU_Utilization for 100 Tasksets with 12 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=12, n=16	55.59	62.54	64.52
m=12, n=20	59.57	63.57	65.54
m=12, n=24	62.41	65.56	69.49
m=12, n=28	65.45	66.61	70.42
m=12, n=32	67.77	68.50	71.87
m=12, n=36	68.57	70.43	73.64
m=12, n=40	69.56	73.37	74.40
m=12, n=44	70.44	74.46	75.78
m=12, n=48	72.49	76.40	77.45

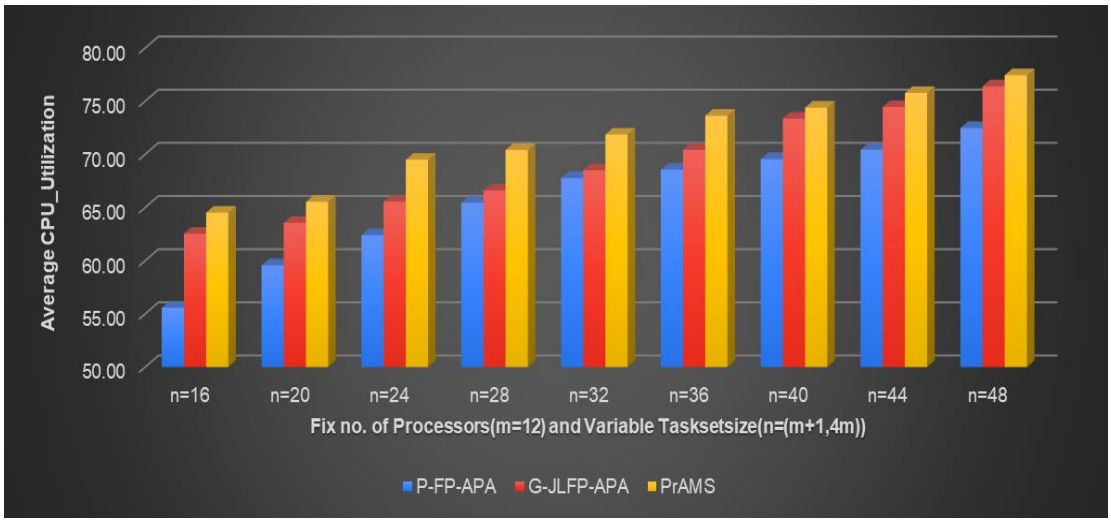


FIGURE 5.16: Average CPU_Utilization with 12 Processors

As shown in FIGURE 5.16, the PrAMS gives the more **CPU_Utilization** that is more no. of tasks get executed before its deadline as compared to all other multiprocessor scheduler due to PrAMS' flexible migration policy. It also shows that the CPU_Utilization is increasing as the no. of tasks in a taskset size is being increased with fixed no. of processors but if it is compared with FIGURE 5.17 it can be concluded that the CPU_Utilization is decreasing if no. of processor are increased.

3. Average of 100 Tasksets with Fixed no. of Processors 16 and variable Taskset Size

TABLE 5.17: Average CPU_Utilization for 100 Tasksets with 16 Processors

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=16, n=20	51.62	55.56	58.69
m=16, n=24	53.62	57.18	59.43
m=16, n=28	55.89	59.36	61.55
m=16, n=32	56.40	61.34	62.68
m=16, n=36	58.60	63.40	64.36
m=16, n=40	59.39	64.51	65.39
m=16, n=44	61.47	65.60	67.29
m=16, n=48	62.45	67.56	68.59
m=16, n=52	64.43	68.53	70.50
m=16, n=56	65.59	69.32	71.68
m=16, n=60	66.64	71.61	72.45
m=16, n=64	69.68	74.43	75.75

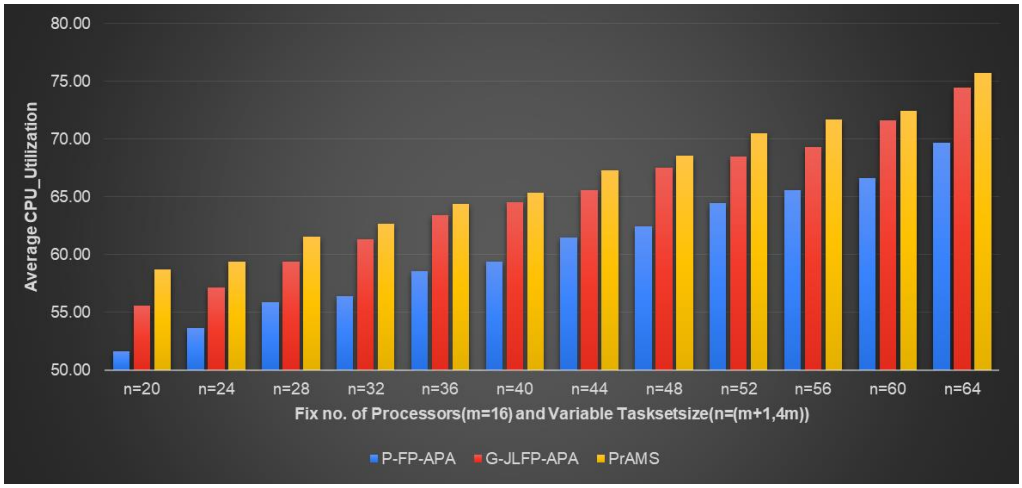


FIGURE 5.17: Average CPU_Utilization with 16 Processors

The FIGURE 5.17 shows that in all the cases, the PrAMS increases **CPU_Utilization** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest CPU_Utilization. One more thing in this case also is the CPU_Utilization is increased as the taskset size is increased and the CPU_Utilization decreased with the no. of processors are increased.

Case-2: Existing Schedulers and PrAMS with Variable No. of Processors and Fixed Tasksetsize (m+1,4m)

In this case we have kept variable no. of processors (m) and the fixed taskset size (n) has been taken into consideration. The taskset size is kept random with the range from m+1 to 4m (where the m is no. of processors and n is taskset size).

There are three different test categories have been selected, each test has been taken with

1. Variable No. of Processors (m=8, 12, 16) and fixed taskset size (n=20).
2. Variable No. of Processors (m=8, 12, 16) and fixed taskset size (n=24).
3. Variable No. of Processors (m=8, 12, 16) and fixed taskset size (n=28).
4. Variable No. of Processors (m=8, 12, 16) and fixed taskset size (n=32).

There are five parameters considered for the results.

- Parameter 1: Average Deadline Miss Ratio should be minimum.
- Parameter 2: Average Tardiness (ns) should be as low as possible.
- Parameter 3: Average No. of Context Switches should be minimum.
- Parameter 4: Average Schedulability (%) should be as high as.
- Parameter 5: Average CPU_Utilization (%) should be maximum.

Here, following three different scheduling algorithms have been compared.

1. P-FP- APA (Partitioned Approach with fixed priority using Affinity)
2. G-JLFP-APA (Global Approach with Job-Level-Fixed priority using Affinity)
3. Proposed Approach (PrAMS along with affinity for processor selection, job level dynamic priority for priority assignment and flexible migration policy using priority reprioritization)

All the tests have been taken on LITMUS^{RT}. The details about simulator LITMUS^{RT} has been discussed in section 5.1.

The results mentioned here is the average of 100 Tasksets. One Taskset can contains any no. of tasks with the range m+1 to 4m and many jobs of each task can appear within the given duration as per the given period. Here results are shown is the average of 100 tasksets. Task generation detail is given in TABLE 5.2. The Detailed results of 100 attempts for each case have been mentioned in Annexure-I, Annexure-II and Annexure-III.

5.4.6 Average Deadline Miss Ratio (Vary number of Processors and Fixed Taskset size)

1. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 20

TABLE 5.18: Average Deadline Miss Ratio for 100 Tasksets with Taskset size 20

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=20	0.4666	0.4141	0.3663
m=12, n=20	0.2863	0.2444	0.1476
m=16, n=20	0.0873	0.0453	0.0034

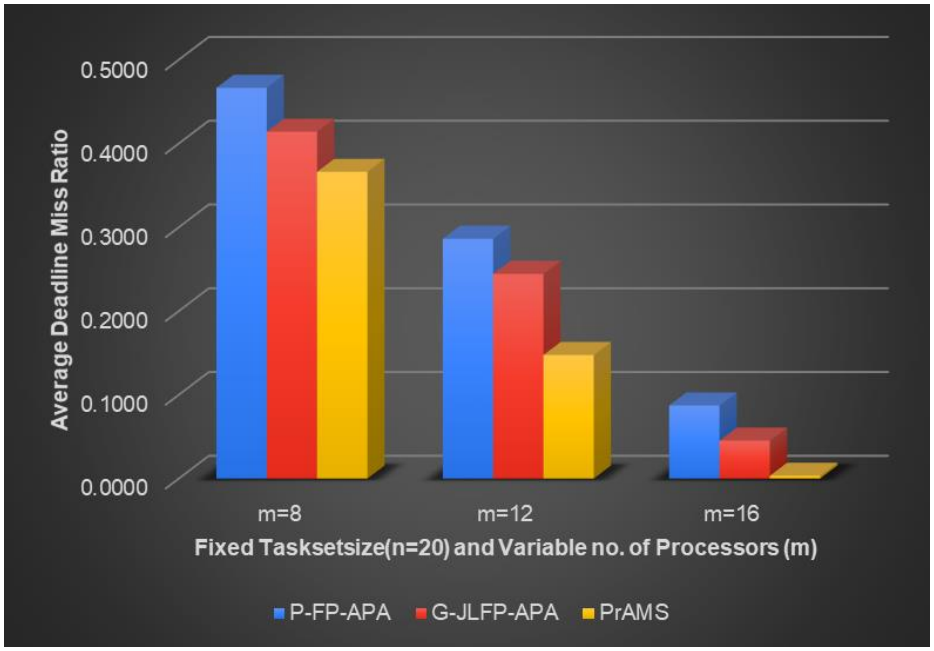


FIGURE 5.18: Average Deadline Miss Ratio with Taskset size 20

As shown in FIGURE 5.18, the PrAMS has minimum **Deadline miss ratio** as compared to P-FP-APA and G-JLFP-APA and the P-FP-APA has missed the maximum deadlines. It also shows that as No. of processors increases, the deadline miss ratio is decreasing for all the schedulers.

2. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 24

TABLE 5.19: Average Deadline Miss Ratio for 100 Tasksets with Taskset size 24

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=24	0.6144	0.5739	0.5238
m=12, n=24	0.4011	0.3745	0.2876
m=16, n=24	0.1427	0.0938	0.0360

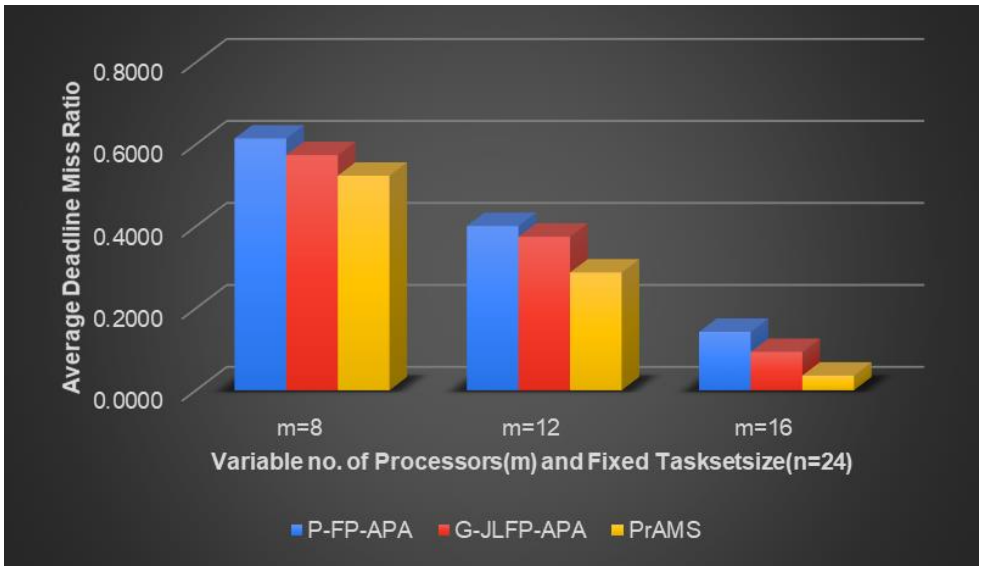


FIGURE 5.19: Average Deadline Miss Ratio with Taskset size 24

As shown in FIGURE 5.19, the PrAMS meets the maximum number of deadlines so it has the lowest **Deadline miss ratio** as compared to all other schedulers. The P-FP-APA has missed the maximum no. of deadlines so its deadline miss ratio is high. It also shows that as No. of processors increases, the deadline miss ratio is decreasing for fixed taskset size.

3. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 28

TABLE 5.20: Average Deadline Miss Ratio for 100 Tasksets with Taskset size 28

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=28	0.6975	0.6261	0.5862
m=12, n=28	0.5342	0.4856	0.3823
m=16, n=28	0.1963	0.1664	0.0852

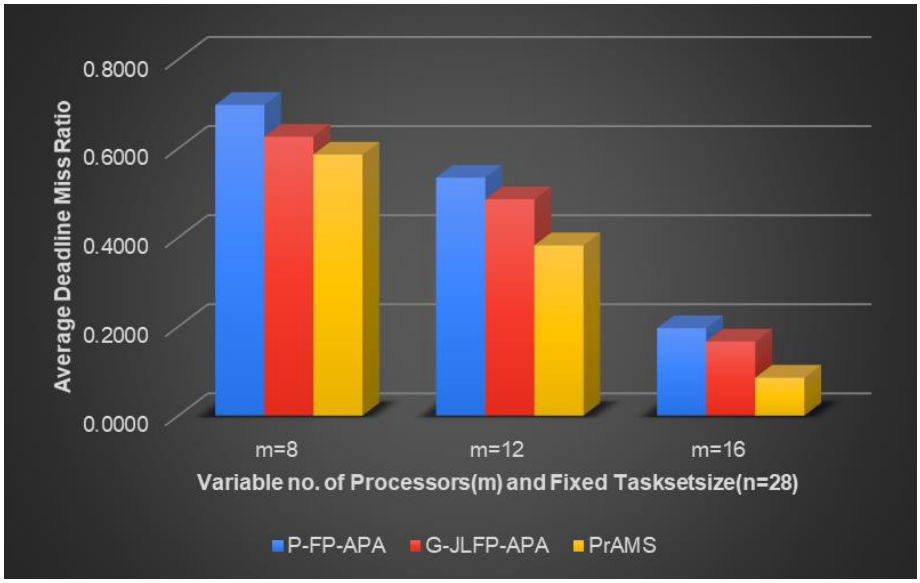


FIGURE 5.20: Average Deadline Miss Ratio with Taskset size 28

As shown in FIGURE 5.20, the PrAMS is giving the results as per the expectation that is minimum **Deadline miss ratio** as compared to all other multiprocessor schedulers and the P-FP-APA has the deadlines miss ratio more than G-JLFP-APA and PrAMS both as P-FP-APA provides fixed priority and processors are strictly partitioned between all the tasks. Here also the deadline miss ratio decreases as the No. of processors with fixed taskset size is there.

4. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 32

TABLE 5.21: Average Deadline Miss Ratio for 100 Tasksets with Taskset size 32

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=32	0.8842	0.7971	0.7762
m=12, n=32	0.6739	0.6033	0.4866
m=16, n=32	0.2841	0.2450	0.1472

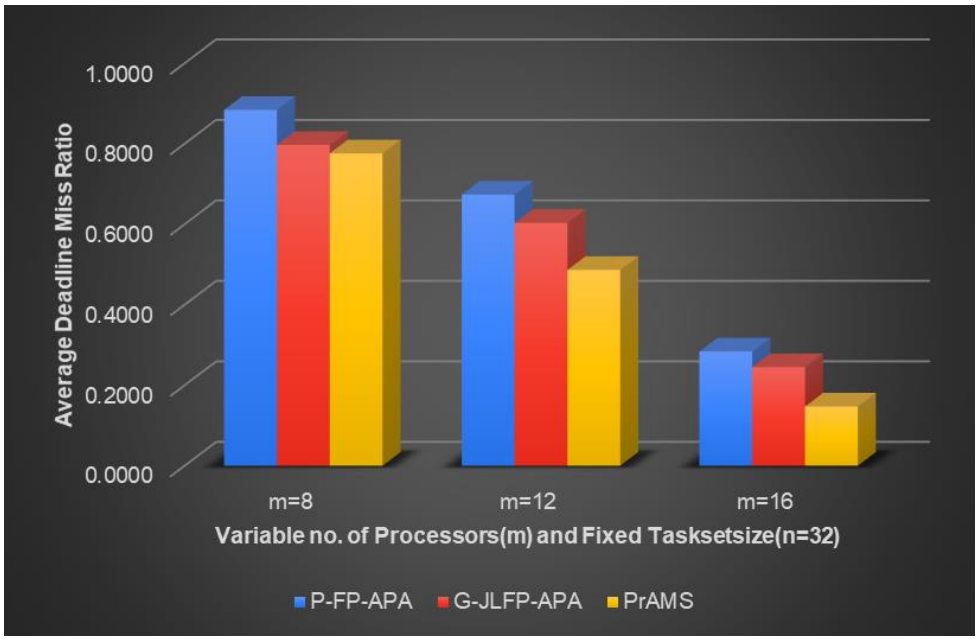


FIGURE 5.21: Average Deadline Miss Ratio with Taskset size 32

As shown in FIGURE 5.21, the PrAMS is giving the results as per the expectation that is minimum **Deadline miss ratio** as compared to all other multiprocessor schedulers and the P-FP-APA has the deadlines miss ratio more than G-JLFP-APA and PrAMS both as P-FP-APA provides fixed priority and processors are strictly partitioned between all the tasks. Here also the deadline miss ratio decreasing as the No. of processors with fixed taskset size is there.

5.4.7 Average Tardiness (Vary No. of Processors and Fixed Taskset size)

1. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 20

TABLE 5.22: Average Tardiness for 100 Tasksets with Taskset size 20

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=20	25.89	20.94	15.71
m=12, n=20	22.42	20.43	14.64
m=16, n=20	14.04	13.08	11.09

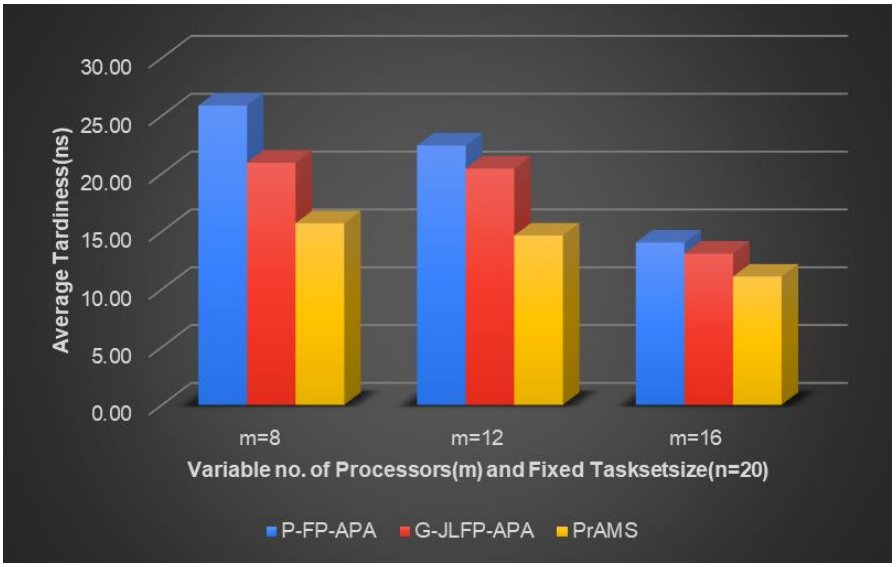


FIGURE 5.22: Average Tardiness with Taskset size 20

As shown in FIGURE 5.22, the PrAMS is having less **Tardiness** as compared to other schedulers like G-JLFP-APA and the P-FP-APA as the PrAMS' flexible migration policy making more task schedulable and reducing the deadline misses. The P-FP-APA has given the maximum tardiness. It can be observed that the tardiness is decreasing as the No. of processors are being increased.

2. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 24

TABLE 5.23: Average Tardiness for 100 Tasksets with Taskset size 24

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=24	27.05	21.59	16.90
m=12, n=24	26.10	21.32	15.49
m=16, n=24	16.39	15.27	12.64

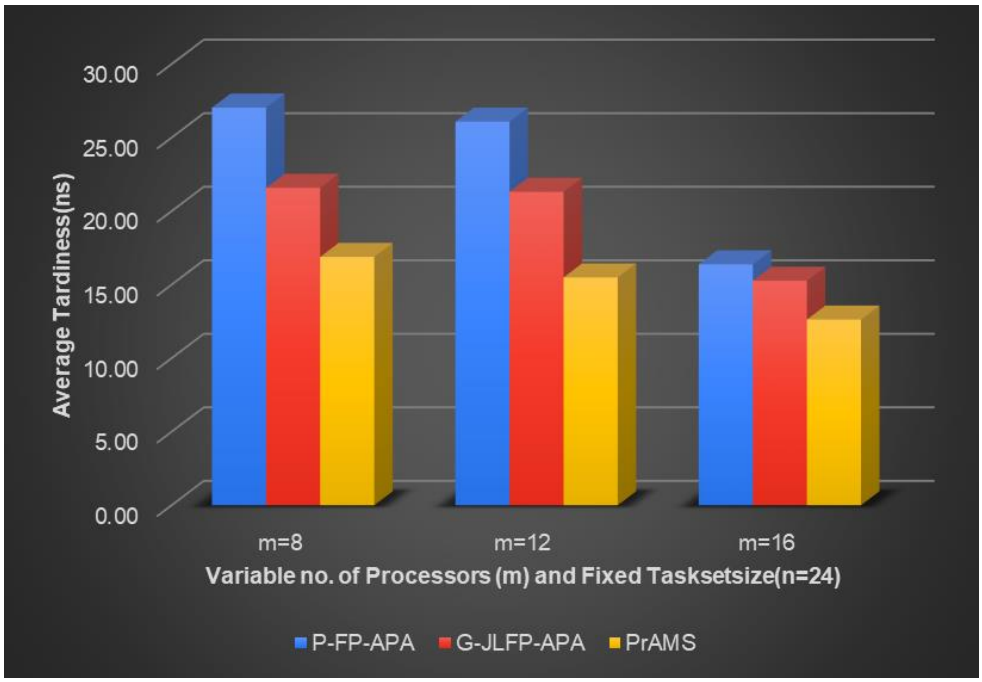


FIGURE 5.23: Average Tardiness with Taskset size 24

As shown in FIGURE 5.23, the PrAMS gives the less **Tardiness** as compared to all other schedulers so PrAMS improves all over system performance as the more tardiness means time taken after deadline is more and in the soft real-time system the performance degradation happens in proportion to the time elapsed after the deadline. It also shows that the tardiness is decreasing as the no. of processors in a system size is being increased with the fixed taskset size.

3. Average of 100 Tasksets with Variable number of Processors (m=8,12,16) and Taskset Size 28

TABLE 5.24: Average Tardiness for 100 Tasksets with Taskset size 28

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=28	28.32	22.16	18.24
m=12, n=28	27.52	21.86	17.67
m=16, n=28	19.37	17.44	14.59

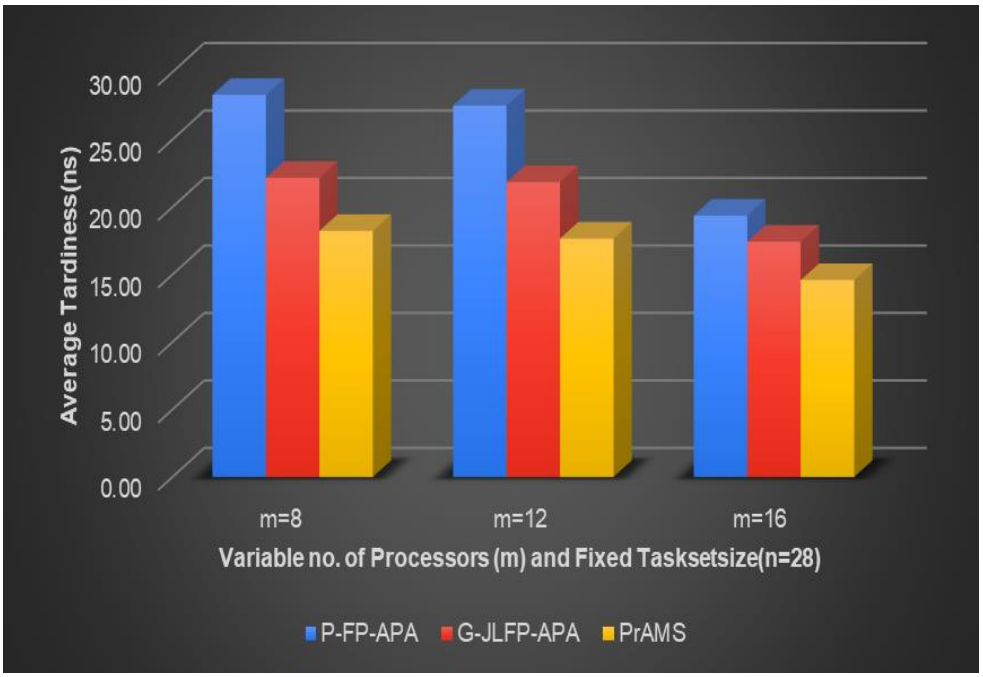


FIGURE 5.24: Average Tardiness with Taskset size 28

As shown in FIGURE 5.24, in this case also the PrAMS is giving less **Tardiness** as compared to the all other multiprocessor schedulers due to the PrAMS' flexible migration policy which is making more no. of tasks schedulable and reducing the deadline misses. It can be seen that as the No. of processors are being increased the tardiness is reducing.

4. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 32

TABLE 5.25: Average Tardiness for 100 Tasksets with Taskset size 32

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=32	31.38	26.63	21.55
m=12, n=32	29.56	24.56	19.38
m=16, n=32	21.54	20.33	16.66

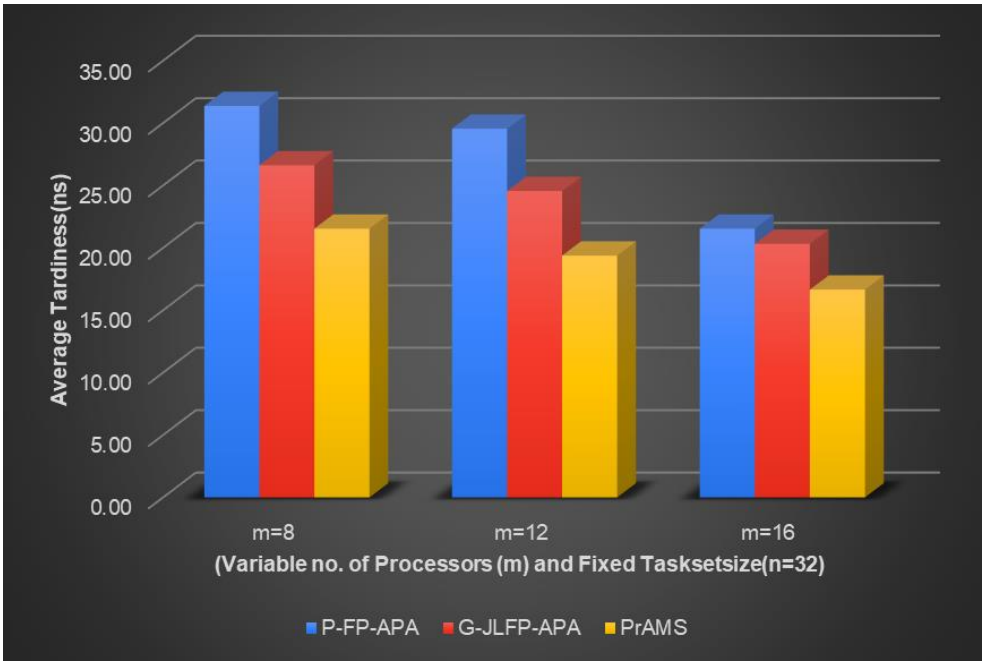


FIGURE 5.25: Average Tardiness with Taskset size 32

As shown in FIGURE 5.25, in this case also the PrAMS is giving less **Tardiness** as compared to the all other multiprocessor schedulers due to the PrAMS' flexible migration policy which is making more no. of tasks schedulable and reducing the deadline misses. It can be seen that as the No. of processors are being increased the tardiness is reducing.

5.4.8 Average No. of Context Switches (Vary No. of Processors and Fixed Taskset size

1. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 20

TABLE 5.26: Avg. No. of Context Switches for 100 Tasksets with

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=20	93.26	118.38	114.37
m=12, n=20	112.32	178.62	156.35
m=16, n=20	118.50	203.54	189.40

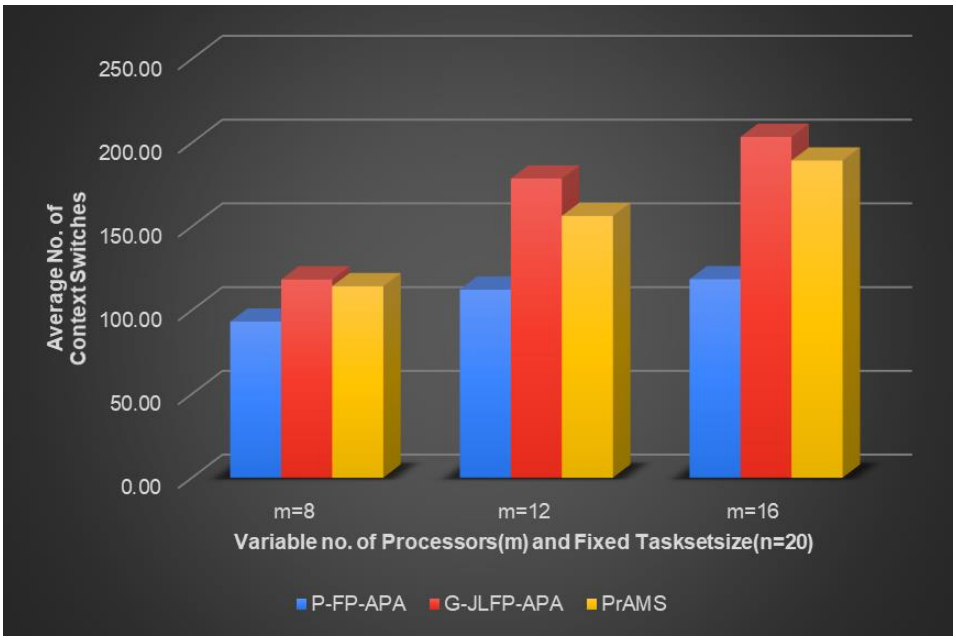


FIGURE 5.26: Average No. of Context Switches with Taskset size 20

As shown in FIGURE 5.26, in most of the cases, the PrAMS is having less no. of **Context switches** as compared to G-JLFP-APA but always more than P-FP-APA because PrAMS provides flexible migration policy which is increasing no. of context switches. It can be observed that in some cases PrAMS is having more no. of context switches than G-JLFP-APA also. One more observation is that as the No. of processors is being increased the no. of context switches are also increasing for all the schedulers.

2. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 24

TABLE 5.27: Average No. of Context Switches for 100 Tasksets with Taskset size 24

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=24	98.41	125.61	129.53
m=12, n=24	143.23	189.29	167.25
m=16, n=24	145.23	225.21	205.34

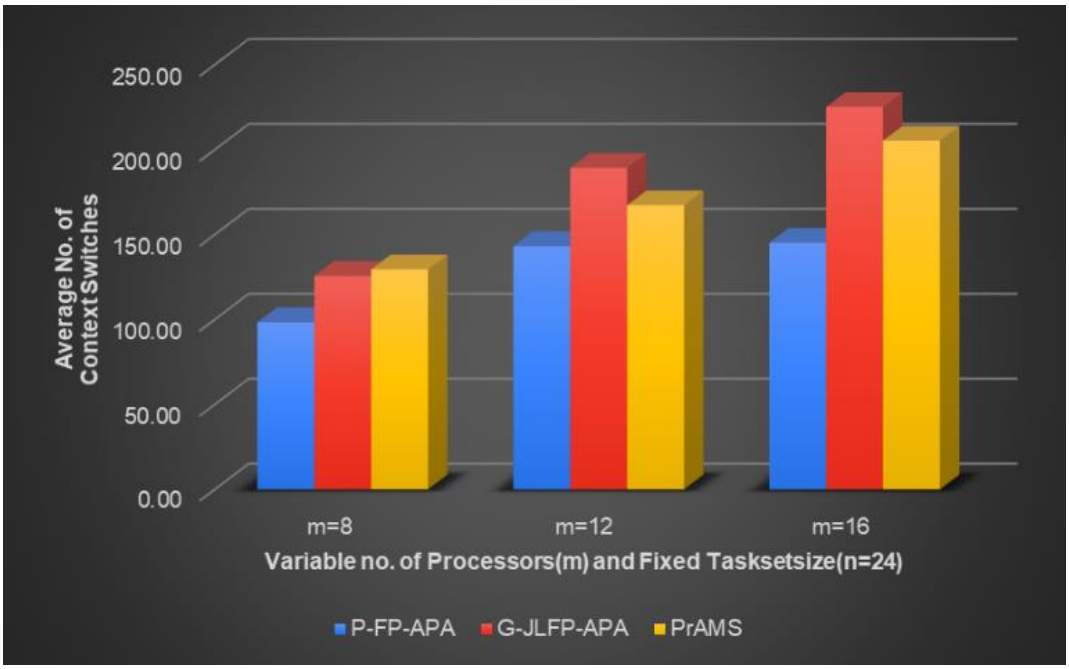


FIGURE 5.27: Average No. of Context Switches with Taskset size 24

The FIGURE 5.27 shows that in all the cases the PrAMS has taken more no. of **Context switches** as compared to P-FP-APA but in most of the cases it is less than the G-JLFP-APA due to PrAMS' flexible migration policy and P-FP-APA has almost no migration kind situation as the processors are strictly partitioned for all the tasks into the system. Another conclusion is that the no. of context switches are getting increased as the no. of processors and the no. of tasks in a taskset are increased.

3. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 28

TABLE 5.28: Average No. of Context Switches for 100 Tasksets with Taskset size 28

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=28	113.51	197.81	134.64
m=12, n=28	154.52	199.69	180.54
m=16, n=28	165.71	246.63	221.51

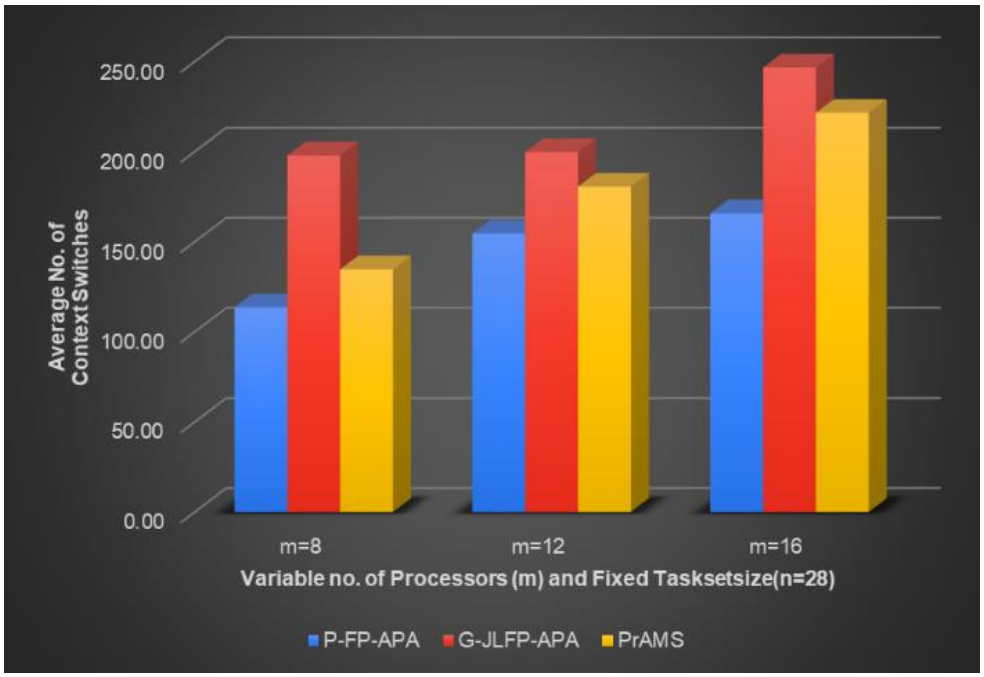


FIGURE 5.28: Average No. of Context Switches with 28

The FIGURE 5.28 shows that in all the cases the PrAMS and G-JLFP-APA have taken more no. of **Context switches** as compared to P-FP-APA as PrAMS provides flexible migration by task shifting with the priority reassignment. Except few cases the PrAMS has taken less no. of context switches than G-JLFP-APA. One more thing in this case also the no. of context switches increased as the taskset size and the no. of processors are increased.

4. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 32

TABLE 5.29: Average No. of Context Switches for 100 Tasksets with Taskset size 32

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=32	165.77	213.77	193.47
m=12, n=32	172.58	219.77	203.31
m=16, n=32	200.06	288.53	254.76

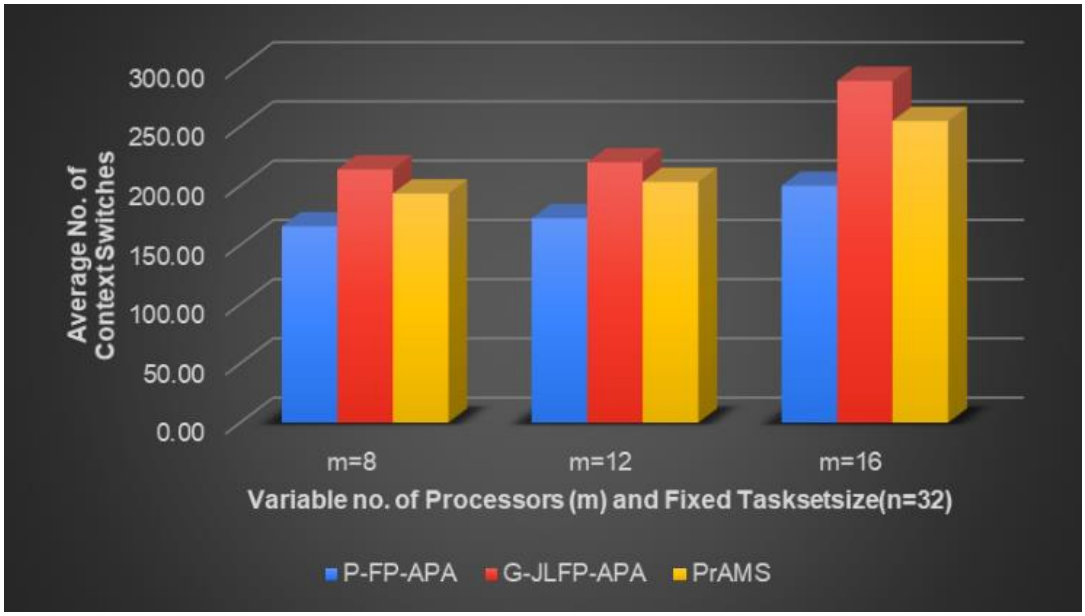


FIGURE 5.29: Average No. of Context Switches with 32

The FIGURE 5.29 shows that in all the cases the PrAMS and G-JLFP-APA have taken more no. of **Context switches** as compared to P-FP-APA as PrAMS provides flexible migration by task shifting with the priority reassignment. Except few cases the PrAMS has taken less no. of context switches than G-JLFP-APA. One more thing in this case also the no. of context switches increased as the taskset size and the no. of processors are increased.

5.4.9 Average Schedulability (Vary No. of Processors and Fixed Taskset size)

1. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 20

TABLE 5.30: Average Schedulability for 100 Tasksets with Taskset size 20

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=20	74.517	77.580	83.200
m=12, n=20	82.44	87.72	89.51
m=16, n=20	87.400	89.587	94.286

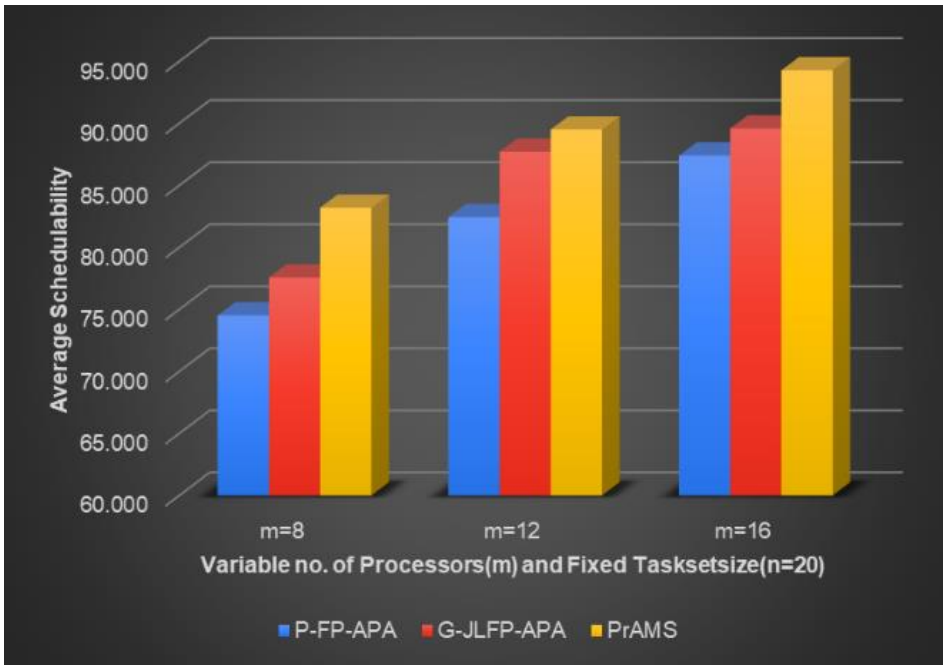


FIGURE 5.30: Average Schedulability with Taskset size 20

As shown in FIGURE 5.30, in all the cases, the PrAMS has the more **Schedulability** as compared to P-FP-APA and G-JLFP-APA both which is the main goal of any real-time system. For all the cases P-FP-APA has less schedulability. It is also observed that the schedulability is increasing as no. of processors increasing with the fixed taskset size.

2. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 24

TABLE 5.31: Average Schedulability for 100 Tasksets with Taskset size 24

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=24	72.450	75.380	81.457
m=12, n=24	78.50	81.55	86.37
m=16, n=24	86.599	88.571	92.236

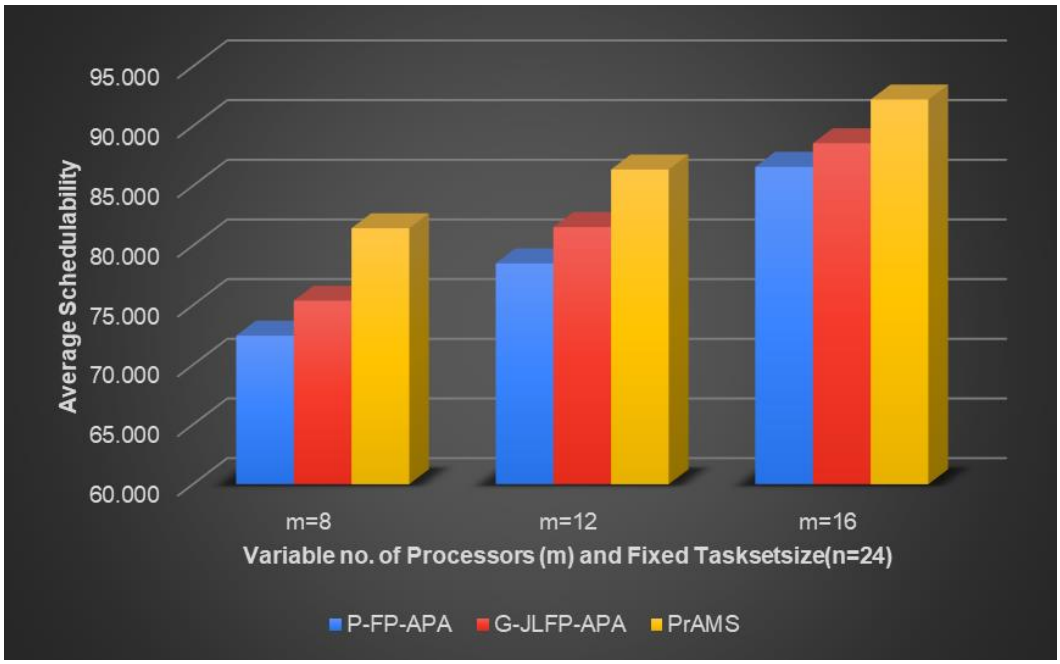


FIGURE 5.31: Average Schedulability with Taskset size 24

As shown in FIGURE 5.31, the PrAMS gives the more **Schedulability** that is more no. of tasks get executed before its deadline as compared to all other multiprocessor scheduler due to PrAMS' flexible migration policy. It also shows that the schedulability is increasing as the no. of processors is being increased with fixed taskset size but if it is compared with FIGURE 5.32 it can be concluded that the schedulability is decreasing if no. of tasks in a taskset are increased.

3. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 28

TABLE 5.32: Average Schedulability for 100 Tasksets with Taskset size 28

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=28	68.440	72.010	77.629
m=12, n=28	78.45	80.64	84.45
m=16, n=28	84.312	87.401	91.568

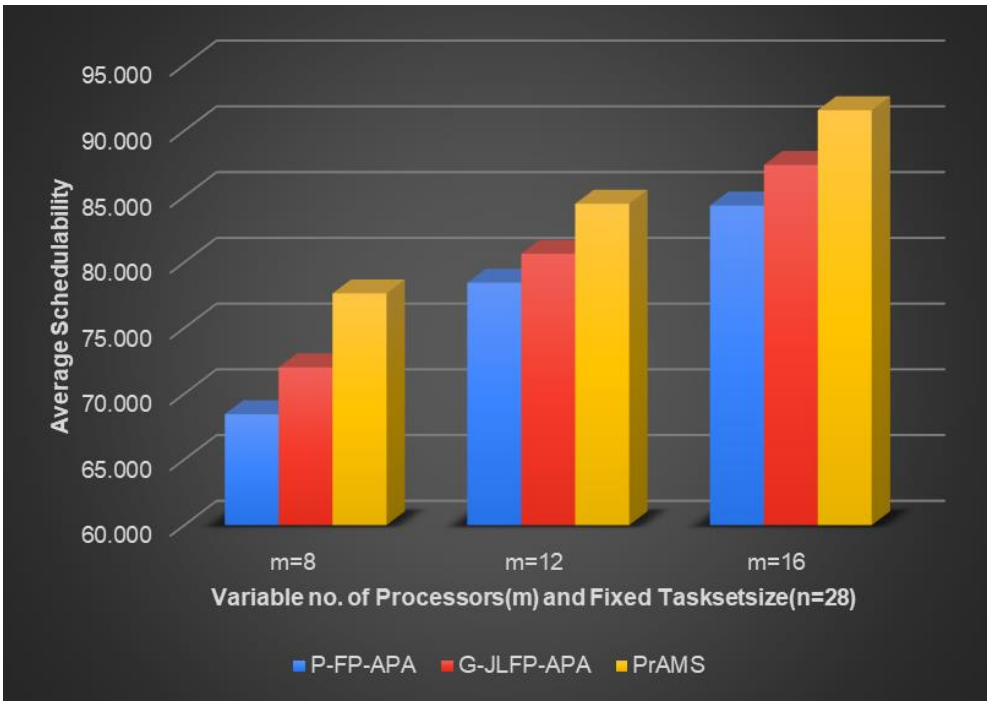


FIGURE 5.32: Average Schedulability with Taskset size 28

The FIGURE 5.32 shows that in all the cases the PrAMS increases overall **Schedulability** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest schedulability. One more thing in this case also is the schedulability increased as the no. of processors are increased and the schedulability decreased with the no. tasks in a taskset size are increased.

4. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 32

TABLE 5.33: Average Schedulability for 100 Tasksets with Taskset size 32

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=32	61.500	64.760	73.525
m=12, n=32	77.66	79.62	82.40
m=16, n=32	81.657	86.280	89.358

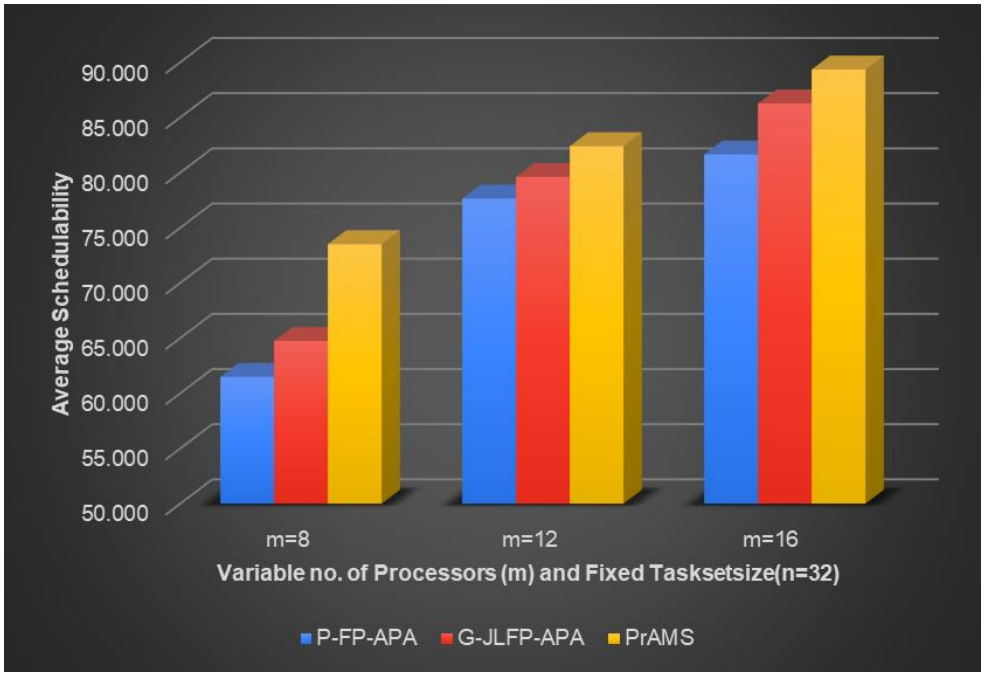


FIGURE 5.33: Average Schedulability with Taskset size 32

The FIGURE 5.33 shows that in all the cases the PrAMS increases overall **Schedulability** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest schedulability. One more thing in this case also is the schedulability increased as the no. of processors are increased and the schedulability decreased with the no. tasks in a taskset size are increased.

5.4.10 Average CPU_Utilization (Vary No. of Processors and Fixed Taskset size)

1. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 20

TABLE 5.34: Average CPU_Utilization for 100 Tasksets with Taskset size 20

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=20	62.46	70.36	73.29
m=12, n=20	59.57	63.57	65.54
m=16, n=20	51.62	55.56	58.69

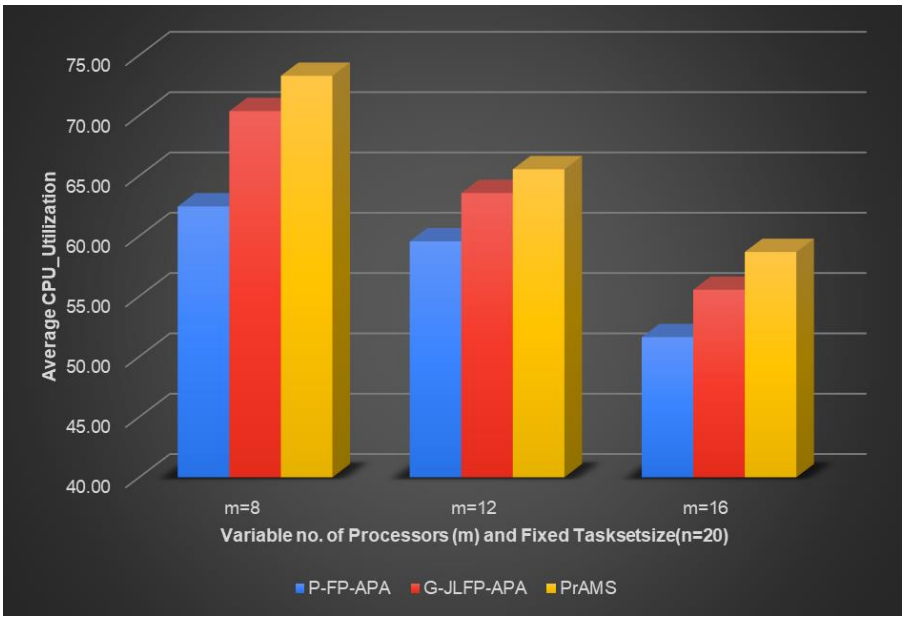


FIGURE 5.34: Average CPU_Utilization with Taskset size 20

As shown in FIGURE 5.34, in all the cases, the PrAMS has the more **CPU_Utilization** as compared to P-FP-APA and G-JLFP-APA both as the PrAMS' schedulability improvement increases overall execution of tasks which utilize the more no. of CPUs. P-FP-APA is having minimum CPU_Utilization. For all the cases P-FP-APA has less schedulability. It is also observed that the CPU_Utilization is decreasing as no. of processors in a system is increasing with the fixed no. of tasks in a taskset.

2. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 24

TABLE 5.35: Average CPU_Utilization for 100 Tasksets with Taskset size 24

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=24	64.61	71.50	74.25
m=12, n=24	62.41	65.56	69.49
m=16, n=24	53.62	57.18	59.43

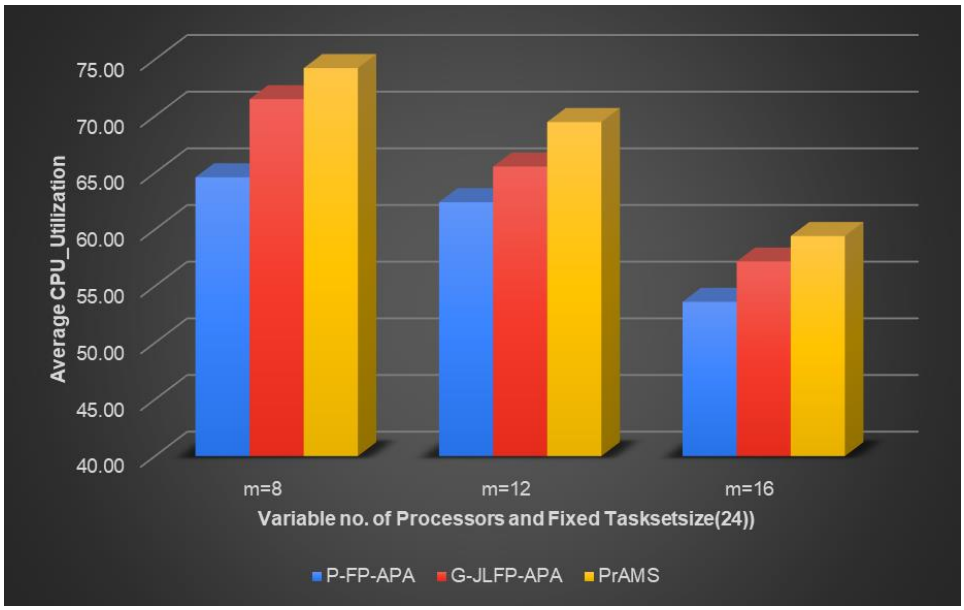


FIGURE 5.35: Average CPU_Utilization with Taskset size 24

As shown in FIGURE 5.35, the PrAMS gives the more **CPU_Utilization** that is more no. of tasks get executed before its deadline as compared to all other multiprocessor scheduler due to PrAMS' flexible migration policy. It also shows that the CPU_Utilization is decreasing as the no. of tasks in the no. of processors are being increased with fixed no. of taskset size but if it is compared with FIGURE 5.36 it can be concluded that the CPU_Utilization is increasing if no. of tasks in a taskset is increased.

3. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 28

TABLE 5.36: Average CPU_Utilization for 100 Tasksets with Taskset size 28

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=28	69.42	75.81	78.54
m=12, n=28	65.45	66.61	70.42
m=16, n=28	55.89	59.36	61.55

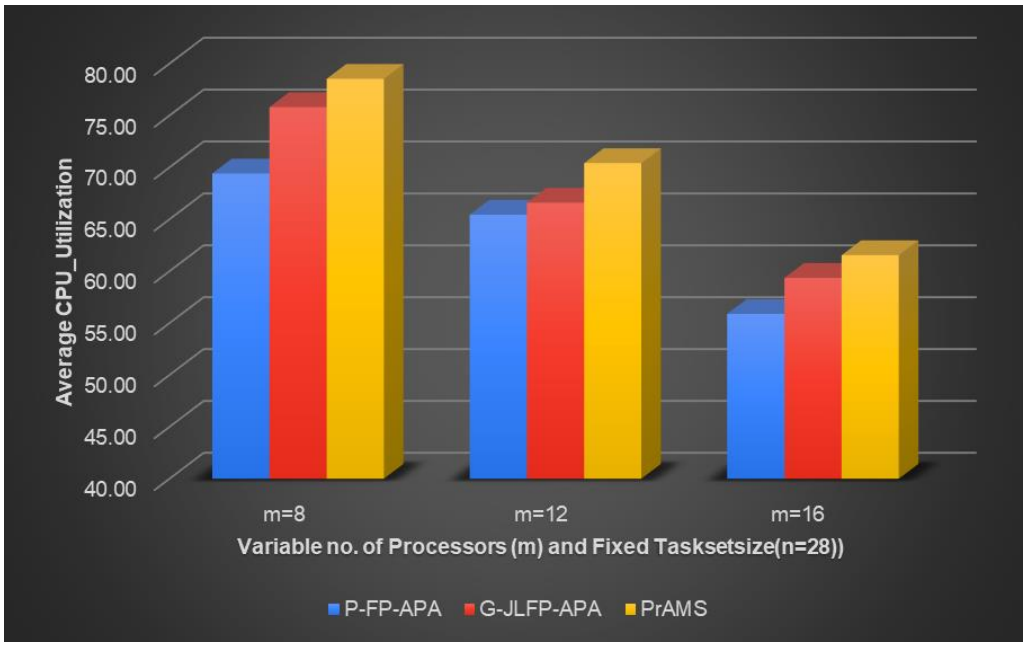


FIGURE 5.36: Average CPU_Utilization with Taskset size 28

The FIGURE 5.36 shows that in all the cases the PrAMS increases **CPU_Utilization** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest CPU_Utilization. One more thing in this case also is the CPU_Utilization is decreased as the no. of processors are increased and the CPU_Utilization increased with the taskset size is increased.

4. Average of 100 Tasksets with Variable no. of Processors (m=8,12,16) and Taskset Size 32

TABLE 5.37: Average CPU_Utilization for 100 Tasksets with Taskset size 32

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=32	73.42	79.34	80.46
m=12, n=32	67.77	68.50	71.87
m=16, n=32	56.40	61.34	62.68

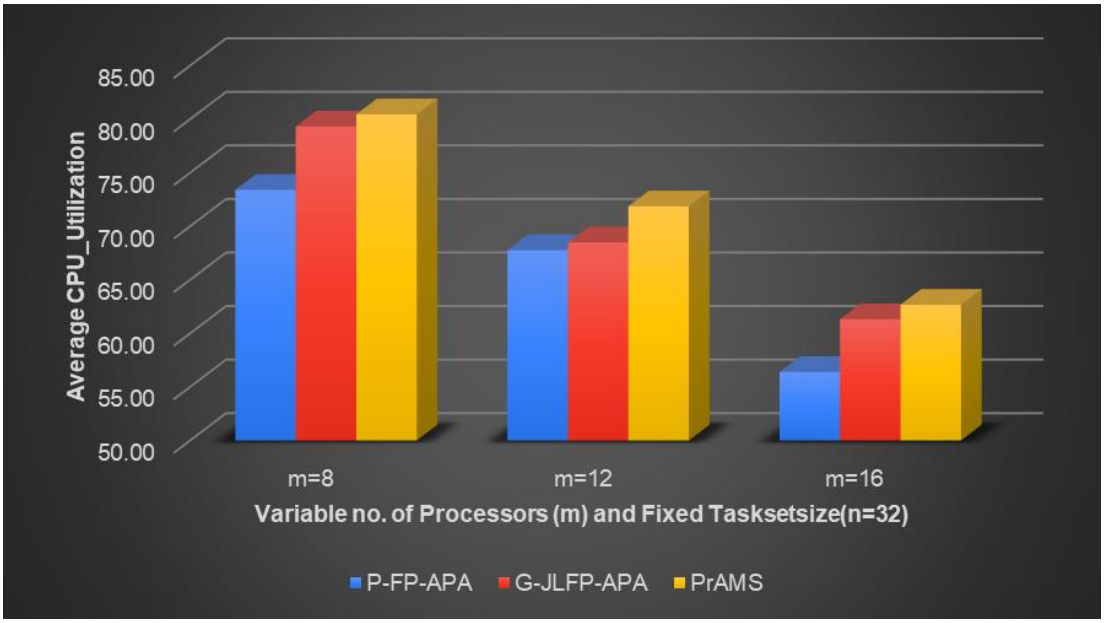


FIGURE 5.37: Average CPU_Utilization with Taskset size 32

The figure 5.37 shows that in all the cases the PrAMS increases **CPU_Utilization** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest CPU_Utilization. One more thing in this case also is the CPU_Utilization is decreased as the no. of processors are increased and the CPU_Utilization increased with the taskset size is increased.

Case-3: Existing Schedulers and PrAMS with No. of Processors and Tasksetsize both variable (m+1,4m)

In this case we have kept variable no. of processors (m) and the variable taskset size (n) has been taken into consideration. The taskset size is kept random with the range from m+1 to 4m (where the m is no. of processors and n is taskset size).

The test has been taken with variable No. of Processors (m=8, 12, 16) and variable taskset size (n=10, 12, 14, 16, 20, 24, 28, 32, 48, 52, 56, 60).

There are five parameters considered for the results.

- Parameter 1: Average Deadline Miss Ratio should be minimum.
- Parameter 2: Average Tardiness (ns) should be as low as possible.
- Parameter 3: Average No. of Context Switches should be minimum.
- Parameter 4: Average Schedulability (%) should be as high as.
- Parameter 5: Average CPU_Utilization (%) should be maximum.

Here, following three different scheduling algorithms have been compared.

1. P-FP- APA (Partitioned Approach with fixed priority using Affinity)
2. G-JLFP-APA (Global Approach with Job-Level-Fixed priority using Affinity)
3. Proposed Approach (PrAMS along with affinity for processor selection, job level dynamic priority for priority assignment and flexible migration policy using priority reprioritization)

All the tests have been taken on LITMUS^{RT} The details about simulator LITMUS^{RT} has been discussed in section 5.1.

The results mentioned here is the average of 100 Tasksets. One Taskset can contains any no. of tasks with the range m+1 to 4m and many jobs of each task can appear within the given duration as per the given period. Here results are shown is the average of 100 tasksets. Task generation detail is given in TABLE 5.2. The Detailed results of 100 attempts for each case have been mentioned in Annexure-I, Annexure-II and Annexure-III.

5.4.11 Average Deadline Miss Ratio (Variable No. of Processors and Taskset size)

Average of 100 Tasksets with variable No. of Processors (m=8, 12, 16) and variable taskset size (n=10, 12, 14, 16, 20, 24, 28, 32, 48, 52, 56, 60)

TABLE 5.38: Average Deadline Miss Ratio with variable No. of Processors and Taskset size

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	0.0945	0.0696	0.0226
m=8, n=12	0.1449	0.1229	0.0847
m=8, n=14	0.2048	0.1841	0.1447
m=8, n=16	0.2661	0.2538	0.2160
m=12, n=20	0.2863	0.2444	0.1476
m=12, n=24	0.4011	0.3745	0.2876
m=12, n=28	0.5342	0.4856	0.3823
m=12, n=32	0.6739	0.6033	0.4866
m=16, n=48	0.6143	0.5450	0.4233
m=16, n=52	0.6835	0.6418	0.5422
m=16, n=56	0.7172	0.6755	0.6243
m=16, n=60	0.7339	0.6919	0.6458

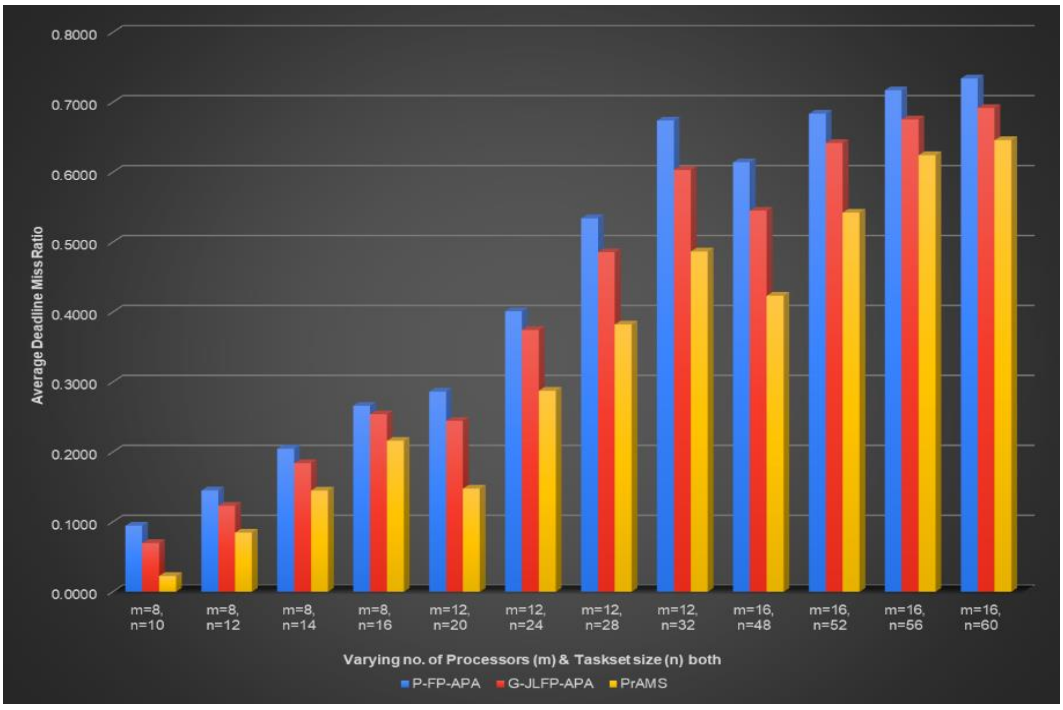


FIGURE 5.38: Average Deadline Miss Ratio with variable No. of Processors and Taskset size

As shown in below FIGURE 5.38, the PrAMS is giving the result as per the expectation that is minimum **Deadline miss ratio** as compared to all other multiprocessor schedulers and the P-FP-APA has the deadlines miss ratio more than G-JLFP-APA and PrAMS both as P-FP-APA provides fixed priority and processors are strictly partitioned between all the tasks. Here also the deadline miss ratio increasing as the no. of tasks in a taskset is being increased with fixed processor but is both are changing than it may depend on no. of processors and taskset size.

5.4.12 Average Tardiness (Variable No. of Processors and Taskset size)

Average of 100 Tasksets with variable No. of Processors (m=8, 12, 16) and variable taskset size (n=10, 12, 14, 16, 20, 24, 28, 32, 48, 52, 56, 60)

TABLE 5.39: Average Tardiness with variable No. of Processors and Taskset size

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	19.51	11.36	8.51
m=8, n=12	21.49	15.38	10.33
m=8, n=14	23.53	19.72	13.50
m=8, n=16	24.45	20.07	14.58
m=12, n=20	22.42	20.43	14.64
m=12, n=24	26.10	21.32	15.49
m=12, n=28	27.52	21.86	17.67
m=12, n=32	29.56	24.56	19.38
m=16, n=48	31.70	27.46	23.47
m=16, n=52	34.35	29.54	25.37
m=16, n=56	34.78	30.33	28.61
m=16, n=60	35.71	32.15	31.53

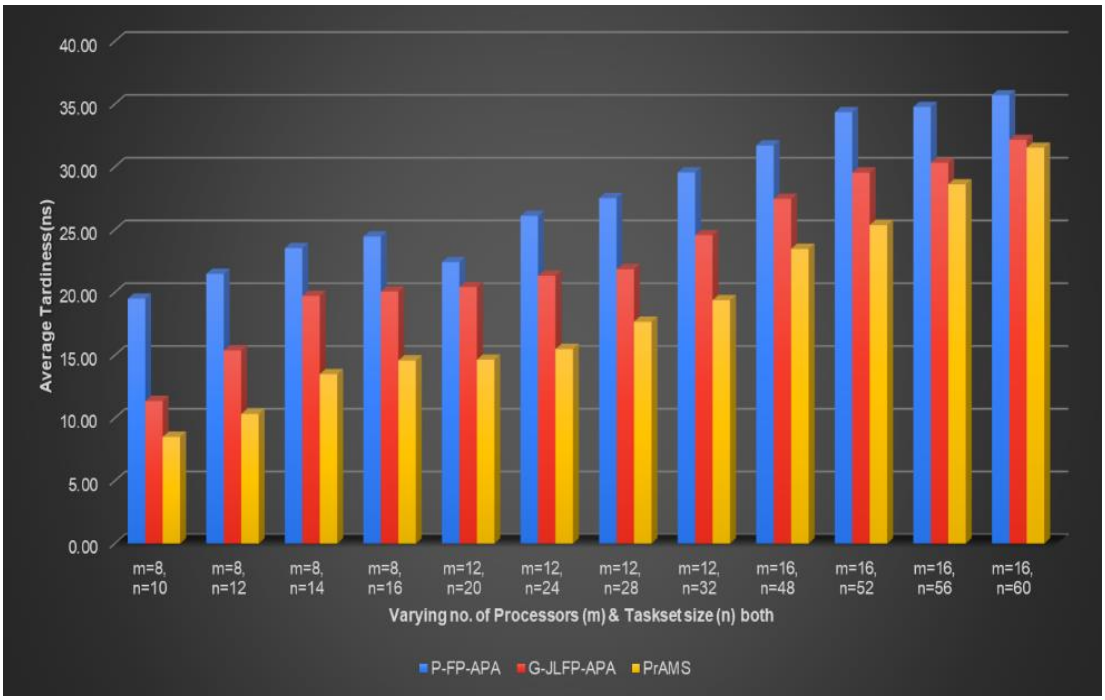


FIGURE 5.39: Average Tardiness with variable No. of Processors and Taskset size

As shown in FIGURE 5.39, the PrAMS gives the less **Tardiness** as compared to all other schedulers so PrAMS improves all over system performance as the more tardiness means time taken after deadline is more and in the soft real-time system the performance degradetion happens in proportion to the time elased after the deadline. It also shows that the tardiness is increasing as the no. of tasks in a taskset size is being increased.

Here also the deadline miss ratio increasing as the no. of tasks in a taskset is being increased with fixed processor but is both the parameters are variable than it depends on no. of processors and taskset size.

5.4.13 Average No. of Context Switches (Variable No. of Processors and Tasksetsize)

Average of 100 Tasksets with variable No. of Processors (m=8, 12, 16) and variable taskset size (n=10, 12, 14, 16, 20, 24, 28, 32, 48, 52, 56, 60)

TABLE 5.40: Average No. of Context Switches with variable No. of Processors and Taskset size

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	59.48	67.56	71.24
m=8, n=12	71.45	83.53	81.52
m=8, n=14	83.54	99.23	91.60
m=8, n=16	87.60	107.73	99.38
m=12, n=20	112.32	178.62	156.35
m=12, n=24	139.74	189.29	167.25
m=12, n=28	154.52	199.69	180.54
m=12, n=32	172.58	219.77	203.31
m=16, n=48	263.79	368.23	364.58
m=16, n=52	272.91	382.59	389.20
m=16, n=56	297.62	397.92	407.11
m=16, n=60	321.66	412.43	414.60

The FIGURE 5.40 shows that in all the cases the PrAMS has taken more no. of **Context switches** as compared to P-FP-APA but in most of the cases it is less than the G-JLFP-APA due to PrAMS' flexible migration policy and P-FP-APA has almost no migration kind situation as the processors are strictly partitioned for all the tasks into the system. Another conclusion is that the no. of context switches are getting increased in all three cases; the taskset size is increased and the no. of processors are fixed or the no. of processors are increased or the taskset size and the no. of processors both are increasing.

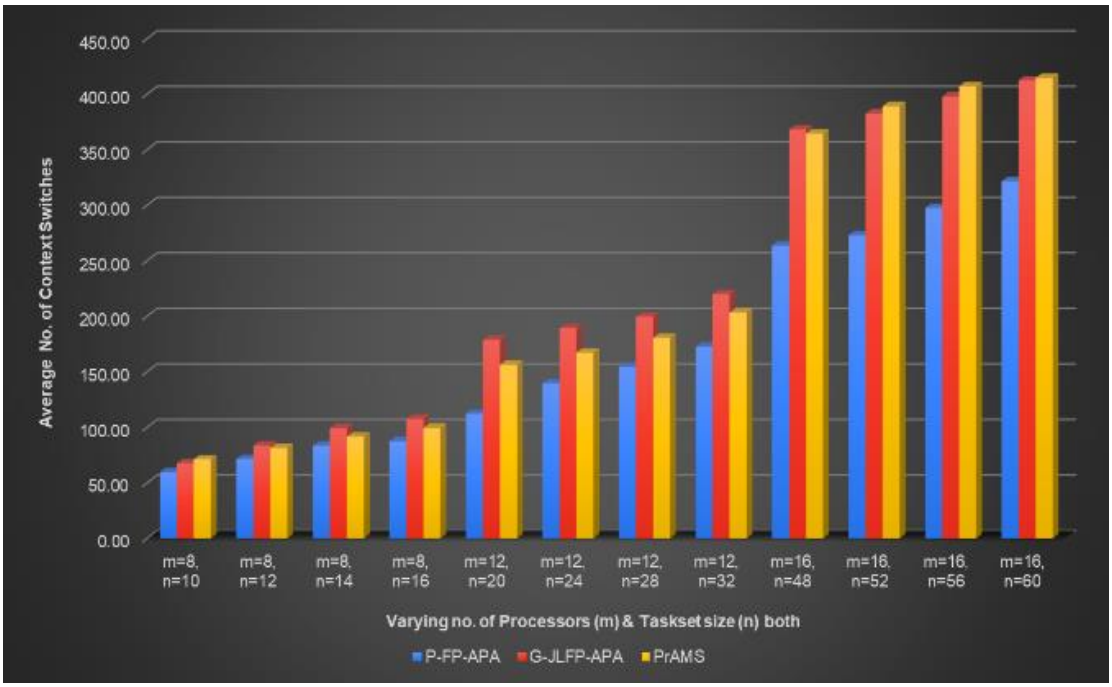


FIGURE 5.40: Average No. of Context Switches with variable No. of Processors and Taskset size

5.4.14 Average Schedulability (Variable No. of Processors and Taskset size)

Average of 100 Tasksets with variable No. of Processors (m=8, 12, 16) and variable taskset size (n=10, 12, 14, 16, 20, 24, 28, 32, 48, 52, 56, 60)

TABLE 5.41: Average Schedulability with variable No. of Processors and Taskset size

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	82.020	86.340	89.622
m=8, n=12	79.810	83.855	87.382
m=8, n=14	77.600	81.370	84.685
m=8, n=16	76.575	80.025	84.446
m=12, n=20	82.44	87.72	89.51
m=12, n=24	78.50	81.55	86.37
m=12, n=28	78.45	80.64	84.45
m=12, n=32	77.66	79.62	82.40
m=16, n=48	76.676	81.281	84.658
m=16, n=52	75.416	80.449	83.444
m=16, n=56	75.232	79.420	82.446
m=16, n=60	74.500	79.302	81.490

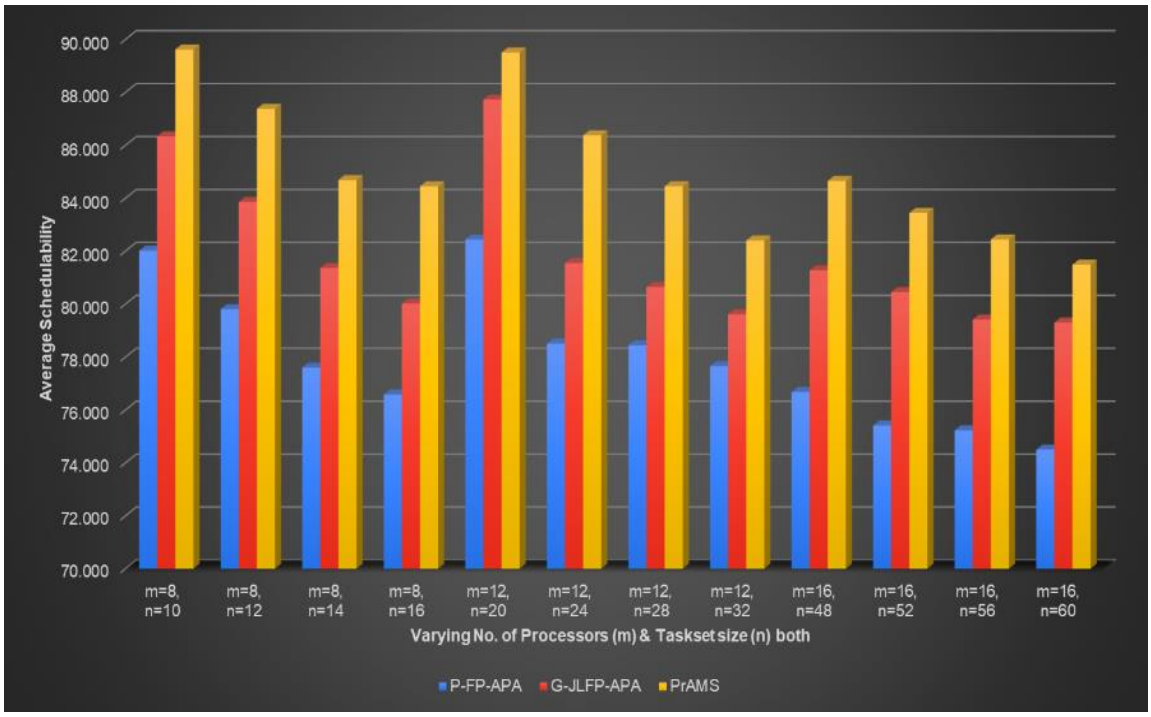


FIGURE 5.41: Average Schedulability with variable No. of Processors and Taskset size

The FIGURE 5.41 shows that in all the cases the PrAMS increases overall **Schedulability** of the system as compared to all other scheduler as it provides flexible task migration policy by priority reprioritization mechanism and P-FP-APA has lowest schedulability. One more thing in this case also is the schedulability decreased as the taskset size is increased and the schedulability increased with the no. of processors are increased but the increase/decrease depends on no. of processors and taskset size if both parameters are variable.

5.4.15 Average CPU_Utilization (Variable No. of Processors and Taskset size)

Average of 100 Tasksets with variable No. of Processors (m=4, 8, 12, 16) and variable taskset size (n=10, 12, 14, 16, 20, 24, 28, 32, 48, 52, 56, 60)

As shown in FIGURE 5.42, in all the cases, the PrAMS has the more **CPU_Utilization** as compared to P-FP-APA and G-JLFP-APA both as the PrAMS' schedulability improvement increases overall execution of tasks which utilize the more no. of CPUs. P-FP-APA is having minimum CPU_Utilization. For all the cases P-FP-APA has less CPU_Utilization. It is also observed that the CPU_Utilization is increasing as no. of tasks in

a taskset is increasing with the fixed no. of processors but if there is variable no. of processors and taskset size than result depends on no. of processors and no. of tasks in a taskset.

TABLE 5.42: Average CPU_Utilization with variable No. of Processors and Taskset size

No. of Processors, No. of Tasks	Algorithms		
	P-FP-APA	G-JLFP-APA	PrAMS
m=8, n=10	59.28	64.37	66.32
m=8, n=12	59.44	64.50	67.39
m=8, n=14	60.35	65.43	68.32
m=8, n=16	61.38	67.34	70.39
m=12, n=20	59.57	63.57	65.54
m=12, n=24	62.41	65.56	69.49
m=12, n=28	65.45	66.61	70.42
m=12, n=32	67.77	68.50	71.87
m=16, n=48	62.45	67.56	68.59
m=16, n=52	64.43	68.53	70.50
m=16, n=56	65.59	69.32	71.68
m=16, n=60	66.64	71.61	72.45

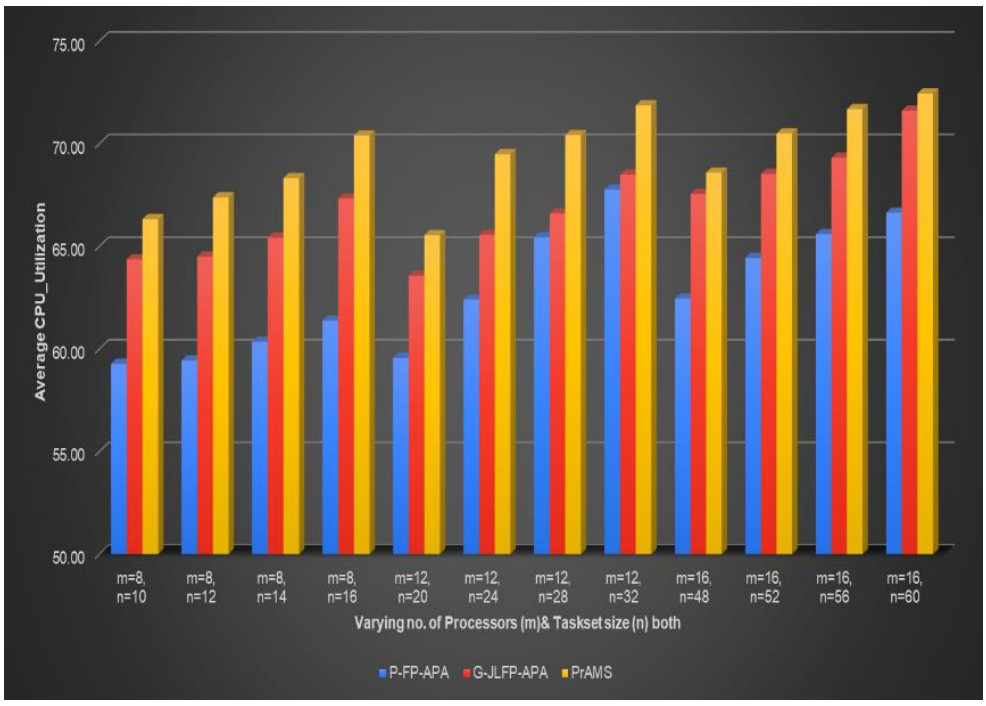


FIGURE 5.42: Average CPU_Utilization with variable No. of Processors and Taskset size

Chapter 6:
Conclusion & Future Scope

This chapter describes the summary of the thesis and points out the probable expansion for future work. The chapter is separated into two sections. The first section 6.1 concludes the thesis by mentioning the major contributions and also the limitations of the system proposed while the second section 6.2 presents future directions for further research.

6.1 Conclusion

The generalized multi-processor real-time scheduler presented in the thesis primarily focuses on three different problems. First is the processor selection problem in a multiprocessor system which is based on affinity concept and processor affinity is kept static in this experiment. The second problem addressed is the process selection that is priority assignment for task execution order which is done with Job-Level-Dynamic-Priority (JLDP) approach. The third is to provide flexible migration policy with the task shifting by reprioritizing the task. This research also addresses the priority inversion problem perfectly which is the big problem in any real-time system. The proposed processor affinity based scheduler improves overall schedulability of the system by providing the flexible migration policy. The proposed scheduler provides consistent and optimum results for different performance measure parameters like less deadline miss ratio, less tardiness, Maximum CPU_Utilization and improved schedulability as compared to traditional approaches discussed in the literature.

- The processor affinity based scheduling approach PrASWM without flexible migration policy has been proposed for multiprocessor real-time system which provides consistent and optimum results in all the cases. Scheduling tasks with proposed approach with dynamic priority assignment policy improves schedulability and CPU_Utilization reduces the deadline miss ratio and tardiness as compared to conventional approaches like G-JLFP-APA and P-FP-APA.
- The processor affinity based scheduler PrAMS along with flexible migration using priority reprioritization and dynamic priority assignment has been proposed for multiprocessor soft real-time system which provides consistent and optimum results

in all the cases. Scheduling tasks with the proposed approach with flexible migration and dynamic priority assignment policy improves schedulability and CPU_Utilization as compared to existing approaches. Scheduling tasks with proposed approach minimize the Deadline miss ratio as well as Tardiness as compared to traditional approaches. Proposed approach gives always more no. of context switches than partitioned approach but when we compare it with global then in some cases it has less no. of context switches than the Global approach and in some cases it is more than global approach.

The common observations for all schedulers based on result analysis are as following:

Parameter	Observations for all Schedulers
Deadline Miss Ratio	i. The deadline miss ratio increases when n-m increases as the number of tasks are higher than processors which causes misses more frequently.
	ii. The deadline miss ratio decreased if the number of processors are increased with the same taskset size that performs better for n is lower and m is constant.
	iii. If m increases and n is kept constant then PrAMS' performance is getting improved (ratio gets lower) as we have more choices to migrate our tasks.
Tardiness	i. The Tardiness increases when n-m increases (more misses as tasks are higher than processors and if more no. of misses than time elapsed after deadline is more)
	ii. The Tardiness is decreasing if number of processors are increased with fixed taskset size (constant n) as we have more choices to migrate our tasks and due to that less deadline miss appears in the system which reduces the tardiness.
No. of Context Switches	i. The number of context switches increases in both the cases either n or m any value increases as if n increases according to priority requirement it needs to shift that's why increases and if

	m increases it gets more chance to migrate so it should increases
	ii. PrAMS always has more no. of context switches than the Partitioned approach and in some cases (like more no. of processors) it also has more than global approach or almost same as global (for less no. of processors).
	iii. In any case partitioned approach has less no. of context switches than the global approach and proposed approach.
Schedulability	i. Schedulability decreases when n-m increases (more misses as tasks are higher than processors and if more no. of misses then lower the schedulability)
	ii. The PrAMS performs better in all three cases; a) No. of processors fixed and variable taskset size b) No. of processors variable and constant taskset size c) Both variable.
	iii. The schedulability is increasing when m increases and n is constant as we have more choice to migrate our tasks and fewer misses.
CPU_Utilization	i. CPU_Utilization increases when n-m increases as more tasks will be executed on same no. of processors.
	ii. CPU_Utilization decreases when m increases for constant n as the same number of tasks are being executed on more no. of processors.
	iii. In all the cases global and PrAMS are having more CPU_Utilization than partitioned approach as in partitioned approach the task must be executed on predefined processor even though other processors free in our system.
	iv. The PrAMS performs better than the global and partitioned approach in all the cases as it is possible to schedule more number of tasks for execution due to its flexible migration policy.

6.2 Limitations and Future Scope

The following section presents the limitations & future scope for this research work.

6.2.1 Limitations

The major problem faced in implementing the scheduler is the time required to test the large number of task sets on more number of processors and the infrastructure to support such execution. With existing resources and data structure as per best of my knowledge, experiments are performed to achieve better results which are mentioned in the thesis.

The proposed approach has taken more no. of context switches as compared to partitioned approach in all the cases and in some cases it is also greater than the global approach due to its flexible migration policy.

6.2.2 Future Scope

The analysis presented here several areas for future work.

1. The implementation presented here has used the static processor affinity assignment; so one can reform the scheduler which can apply dynamic processor affinity so schedulability can be improved.
2. The proposed approach has taken more no. of context switches as compared to partitioned in all the cases and in some cases it is greater than the global also; so one can reduce the no. of context switches by careful processor affinity assignment.
3. Synchronization for sharing the resource is a significant thought in the proposal of a scheduling algorithm which supports concurrency and scalability. But, for real-time systems, it is essential to consider memory allocation algorithm with the synchronization. So one can design a memory allocator which can be integrated with PrAMS.
4. The proposed algorithm is kept limited to the one task shift only, so one can improve the schedulability by using multiple task shifts with the help of advanced data structure.

List of References

[1] A. Bastoni, B. B., & Anderson, J. (2011). Is semi-partitioned scheduling practical? . *Proceedings of the 23rd Euromicro Conference on Real-Time Systems*, (pp. 125-135).

[2] A. Silberschatz, P. B., & Gagne, G. (2012). *Operating System Concepts* (8th ed.). Wiley.

[3] Alhussian, H., Zakaria, N., & Hussin, F. A. (2014, January). An Efficient Real-Time Multiprocessor Scheduling Algorithm. *Journal of Convergence Information Technology*.

[4] Anderson, J. H., & Srinivasan, A. (2004). Mixed pfair/erfair scheduling of asynchronous periodic tasks. *Journal of Computer and System Sciences, 68*(1), 157–204.

[5] Anderson, J. H., Bud, V., & Devi, U. C. (2008). An EDF -based restricted-migration scheduling algorithm for multiprocessor soft real-time systems. *Real-Time Systems, 38*(2), 85-131.

[6] Anderson, J. H., Erickson, J. P., Devi, U. C., & Casses, B. N. (2014). Optimal semi-partitioned scheduling in soft real-time systems. *Proceedings of the 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.*

[7] Anderson, J., & Srinivasan, A. (2000). Early-release fair scheduling. *Proceedings of the 12th Euromicro Conference on Real-Time Systems*, (pp. 35-43).

[8] Anderson, J., Bud, V., & Devi, U. (2005). An EDF-based scheduling algorithm for multiprocessor soft real-time systems. *Proceedings of the 17th Euromicro Conference on Real-Time Systems*, (pp. 199–208).

[9] Andersson, B., Raravi, G., & Bletsas, K. (2010).) Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors. *Proceedings of the 31st IEEE Real-Time Systems Symposium*, (pp. 239-248).

[10] Audsley, N. (2001, May). On priority assignment in fixed priority scheduling. *Information Processing Letters, 79*(1), 39-44.

[11] Audsley, N. C., Burns, A., Richardson, M. F., & Wellings, A. J. (1991). Hard Real-Time Scheduling: The Deadline Monotonic Approach. *Proceedings of the 8th IEEE Workshop on Real-Time Operating Systems and Software*, (pp. 133-137).

[12] Audsley, N. (n.d.). *Optimal priority assignment and feasibility of static priority tasks with arbitrary start times.* Technical Report YCS 164, University of York, UK, Dept. Computer Science.

[13] Audsley, N. (September 1990). *Deadline Monotonic Scheduling.* Technical Report, University of York, Department of Computer Science.

[14] Ba, Q. L. (2014). A group priority EDF scheduling algorithm. *Frontiers of Computer Science, 16*(5), 26-57.

[15] Bado, B., L, L. G., P, P. C., & Goossens, J. (2012). A semi-partitioned approach for parallel real-time scheduling. *Proceedings of the 20th International Conference on Real-Time and Network Systems*, (pp. 151-160).

[16] Baker, T. P., & Baruah, S. K. (2009). Sustainable multiprocessor scheduling of sporadic task systems. *proceedings of the Euromicro Conference on Real-Time Systems*, (pp. 141-150).

[17] Baker, T., & Baruah, S. (2007). *Schedulability analysis of multiprocessor sporadic task systems.* Chapman Hall/CRC.

[18] Barabanov, M. (1997). *A Linux-based Real-Time Operating System.* Master's thesis, New Mexico Institute of Mining and Technology, Socorro New Mexico.

[19] Baruah S. K., M. A., & Rosier, L. E. (1990). Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor. *proceedings of the IEEE Real-Time System Symposium*, (pp. 182-190).

[20] Baruah, S. (2007). Techniques for multiprocessor global schedulability analysis. *28th IEEE International Real-Time Systems Symposium* (pp. 119-128). Washington, DC, USA: IEEE Computer Society.

[21] Baruah, S., & Brandenburg, B. B. (2013). Multiprocessor feasibility analysis of recurrent task systems with specified processor affinities. *Proceedings of the 34th IEEE Real-Time Systems Symposium*, (pp. 160–169).

[22] Bastoni, A., Brandenburg, B., & Anderson, J. (2010). An empirical comparison of global, partitioned, and clustered multiprocessor EDF schedulers. *Real-Time Systems Symposium (RTSS).* 31st IEEE Conference.

[23] Bertogna, M., & Cirinei, M. (2007). Response-time analysis for globally scheduled symmetric multiprocessor platforms. *Real-Time Systems Symposium* (pp. 149–160). 28th IEEE International Conference.

[24] Block, A., & Kelley, W. (2015). Implementing Adaptive Clustered Scheduling in LITMUSRT. *Proceedings of the 11th Annual Workshop on Operating Systems Platforms for Embedded Real-Time applications* , (pp. 33-35).

[25] Bovet, D. P., & Cesati, M. (2000). *Understanding the Linux Kernel.* O'Reilly Online Catalogue.

[26] Brandenburg, B. B. (2011). *Scheduling and Locking in Multiprocessor Real-Time Operating Systems.* PhD Thesis, The University of North Carolina, Chapel Hill.

[27] Brandenburg, B. B., & Anderson, J. H. (2009). The implementation of global real- time schedulers. *Proceedings of the 30th IEEE Real-Time Systems Symposium, RTSS'09* (pp. 214–224). Washington, DC, USA: IEEE Computer Society.

[28] Brandenburg, B., & Anderson, J. (2007). Feather Trace: A light-weight event tracing toolkit. *Proceedings of the Third International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, (pp. 19–28).

[29] Brandenburg, B., & Anderson, J. (2007). Integrating hard/soft real-time tasks and best-effort jobs on multiprocessors. *Proceedings of the 19th Euromicro Conference on Real-Time Systems*, (pp. 61–70).

[30] Brandenburg, B., & Anderson, J. (2008). A Comparison of the M-PCP, D-PCP, and FMLP on LITMUS. *proceedings of the International Conference on Principles of Distributed Systems.*

[31] Brandenburg, B., & Anderson, J. (2008). An implementation of the PCP, SRP, D-PCP, M-PCP, and FMLP real-time synchronization protocols in LITMUSRT. *Proceedings of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium*, (pp. 185–194).

[32] Brandenburg, B., Calandrino, J., & Anderson, J. (2008). On the scalability of real- time scheduling algorithms on multicore platforms: A case study. *Real-Time Systems Symposium*, (pp. 157-169).

[33] Burns, A., & Wellings, A. J. (n.d.). *Real-Time Systems and Programming Languages* (3rd ed.). Addison Wesley.

[34] Burns, A., Davis, R. I., Wang, P., & Zhang, F. (2012). Partitioned EDF scheduling for multiprocessors using a C=D task splitting. *Real-Time Systems Journal*, *48*(1), 3-33.

[35] Buttazzo, G. C. (2000). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications* (3rd ed.). Springer.

[36] Buttazzo, G. C. (2004). Hard Real-time Computing System: Predictable Scheduling Algorithms and Apps. *Real-Time System Series*.

[37] Buttazzo, G. C. (2011). Hard real-time computing systems: predictable scheduling algorithms and applications. *24*.

[38] Buttazzo, G. C., & Stankovic, J. A. (1995). Adding robustness in dynamic preemptive scheduling. *The Springer International Series in Engineering and Computer Science*, 67–88.

[39] Calandrino, J. M., Anderson, J. H., & Baumberger, D. P. (2007). A hybrid real-time scheduling approach for large-scale multicore platforms. *19th Euromicro Conference on Real-Time Systems*, (pp. 247–258).

[40] Calandrino, J. M., Leontyev, H., Block, A., Devi, U. C., & Anderson, J. H. (2006). LITMUS-RT: A testbed for empirically comparing real-time multiprocessor schedulers. *Proceedings of the 27th IEEE International Real-Time Systems Symposium* (pp. 111-126). Washington, DC, USA: IEEE Computer Society.

[41] Chen, X., Xu, H., & Huang, L. (2019). Online Scheduling Strategy to Minimize Penalty of Tardiness for Real-Time Tasks in Mobile Edge Computing Systems. *4th International Conference on Big Data and Computing.* ACM Digital Library.

[42] Chéramy, M., Hladik, P.-E., & Déplanche, A.-M. (2014). SimSo: A Simulation Tool to Evaluate Real-Time. *5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems.* Madrid, Spain: hal-01052651.

[43] Chwa, H. S., Seo, J., Lee, J., & Shin, I. (2015). Optimal Real-Time Scheduling on Two-Type Heterogeneous Multicore Platforms. *IEEE Real-Time Systems Symposium.*

[44] Collette, S., Cucu, L., & Goossens, J. (2007). Algorithm and complexity for the global scheduling of sporadic tasks on multiprocessors with work-limited parallelism. *proceedings of the International Conference on Real-Time and Network Systems*, (pp. 123-128).

[45] Collette, S., Cucu, L., & Goossens, J. (2008, May). Integrating Job Parallelism in Real-Time Scheduling Theory. *Information Processing Letters, 106*(5), pp. 180-187.

[46] Davis, R., & Burns, A. (2011). Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Journal of Real-Time Systems, 47*(1), 1-40.

[47] Davis, R., & Burns, A. P. (2011). A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys, 43*(4), 1-44. doi:10.1145/1978802.1978814

[48] Dertouzos, M., & Mok, A. (1989). Multiprocessor online scheduling of hard-real-time tasks. *IEEE Transactions on Software Engineering, 15*(12), 1497 –1506.

[49] Devi, U. C. (2003). An improved schedulability test for uniprocessor periodic task systems. *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, (pp. 23-30).

[50] Devi, U. C., & Anderson, J. H. (2008). Tardiness bounds under global edf scheduling on a multiprocessor. *Real-Time Systems, 38*(2), 133–189.

[51] Devi, U., & Anderson, J. (2006). Tardiness bounds under global EDF scheduling on a multiprocessor. *26th IEEE International Real-Time Systems Symposium* (pp. 1052-8725). IEEE Xplore.

[52] Diwase, D., Shah, S., Diwase, T., & Rathod, P. (2012). Survey Report on Memory Allocation Strategies for Real-time Operating System in Context with Embedded Devices. *International Journal of Engineering Research and Applications, 2*(3), 1151-1156.

[53] Dorin, F., Yomsi, P. M., Goossens, J., & Richard, P. (2010, june 14). *Semi-Partitioned Hard Real-Time Scheduling with Restricted Migrations upon Identical Multiprocessor Platforms.* Retrieved from https://arxiv.org/abs/1006.2637

[54] Easwaran, A., Shin, I., & Lee, I. (2009). Optimal virtual cluster-based multiprocessor scheduling. *Real-Time Systems, 43*(1), 25-59.

[55] Emberson, P., R, R. S., & Davis, R. (2010). Techniques for the synthesis of multiprocessor tasksets. *Proceedings of the 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, (pp. 6–11).

[56] Erickson, J. P. (2014). *Managing Tardiness Bounds and Overload in Soft Real-Time Systems.* PhD Thesis, The University of North Carolina, Chapel Hill.

[57] Erickson, J., Coombe, G., & Anderson, J. (2012). Soft real-time scheduling in google earth. *Proceedings of the 18th IEEE Real-Time and Embedded Technology and Applications Symposium*, (pp. 141–150).

[58] Eßer, H.-G. (n.d.). Combining memory management and filesystems in an operating systems course. *Proceedings of the 16th Annual {SIGCSE} Conference on Innovation and Technology in Computer Science Education.* Darmstadt, Germany.

[59] Foong, A., Fung, J., & Newell, D. (2004). An in-depth analysis of the impact of processor affinity on network performance. *Proceedings of the 12th IEEE International Conference on Networks*, *1*, pp. 244-250.

[60] Fredkin, E. (1960). *Trie memory. Commun. ACM, 3*(9), pp. 490–499.

[61] Funk, S. (2004). *EDF scheduling on heterogeneous multiprocessors.* Chapel Hill: The University of North Carolina .

[62] Ga´lvez, J., Ruiz, P., & Skarmeta, A. (2010). *Heuristics for scheduling on restricted identical machines.* Technical Report, University of Murcia, Spain.

[63] Gloger, W. (2001). *Dynamic memory allocator implementations in Linux system libraries.* Munich.

[64] Gracioli, G., & Fröhlich, A. A. (2013). An experimental evaluation of the cache partitioning impact on multicore real-time schedulers. *19th International Conference on Embedded and Real-Time Computing Systems and Applications* (pp. 72-81). IEEE. doi:10.1109/RTCSA.2013.6732205

[65] Gracioli, G., Fröhlich, A. A., Pellizzoni, R., & Fischmeister, S. (2012). Analysis of global and partitioned scheduling in a RTOS. *Real-Time Systems, 47*(2), 166–174.

[66] Gracioli, G., Fröhlich, A. A., Pellizzoni, R., & Fischmeister, S. (2013). Implementation and evaluation of global and partitioned scheduling in a real-time OS. *Real-Time Systems, 49*(6), 669-714. doi:10.1007/s11241-013-9183-3

[67] Gujarati, A., Cerqueira, F., & Brandenburg, B. B. (2015). Schedulability analysis of the linux push and pull scheduler with arbitrary processor affinities. *Proceedings of the 25th Euromicro Conference on Real-Time Systems*, (pp. 69–79).

[68] Gujarati, A., Cerqueira, F., & Brandenburg, B. B. (2016). Multiprocessor Real-Time Scheduling with Arbitrary Processor Affinities: From Practice to Theory. *Proceedings of the 25th Euromicro Conference on Real-Time Systems*, (pp. 69-79).

[69] Harkut, D. G., & Agrawal, A. M. (2014, July). Comparison of Different Task Scheduling Algorithms in RTOS: A Survey. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(7), 1236-1240.

[70] Herman, J., Kenna, C., Mollison, M., Anderson, J., & Johnson, D. (2012). RTOS support for multicore mixed-criticality systems. *Proceedings of the 18th IEEE Real-Time and Embedded Technology and Applications Symposium*, (pp. 197–208).

[71] Ionescu, F. (2000). Application-Level Virtual Memory Management in Real-Time Multiprocessor Systems. *Proceedings of the {ACM} Symposium on Applied Computing*, (pp. 610-614). Villa Olmo, Via Cantoni 1, 22100 Como, Italy.

[72] Ismail, H., & Jawawi, D. N. (2017). A Hybrid Multiprocessor Scheduling Approach for Weakly Hard Real-Time Tasks. *17th Asia Simulation Conference* (pp. 666–678). Melaka, Malaysia: Springer.

[73] Jang, H., & Jin, H. (2009). MiAMI: Multi-core aware processor affinity for TCP/IP over multiple network interfaces. *Proceedings of the 17th IEEE Symposium on High Performance Interconnects*, (pp. 73–82).

[74] Kalpana, R., & Keerthika, S. (2014, May). An Efficient Non-Preemptive Algorithm for Soft Real-Time Systems using Domain Cluster–Group EDF. *International Journal of Computer Applications, 93*(20), 0975-8887.

[75] Kato, S., Yamasaki, N., & Ishikawa, Y. (2009). Semi-partitioned scheduling of sporadic task systems on multiprocessors. *Proceedings of the 21st Euromicro Conference on Real-Time Systems*, (pp. 249-258).

[76] Keerthanaa, C., & Poongothai, M. (2016). Improved Priority based Scheduling Algorithm for Real-Time Embedded Systems. *IEEE xplore for International Conference on Circuit, Power and Computing Technologies.*

[77] Kim, N., Tang, S., Otterness, N., Anderson, J., Smith, F., & Porter, D. (2018). Supporting I/O and IPC via Fine-Grained OS Isolation for Mixed-Criticality Real-Time Tasks.

Proceedings of the 26th International Conference on Real-Time Networks and Systems, (pp. 191-201).

[78] Kopetz, H. (2011). *Real-Time Systems: Design Principles for Distributed Embedded Applications* (2nd ed.). Springer.

[79] Kotecha, K., & Shah, A. (2008). ACO based dynamic scheduling algorithm for real-time operating system. *International Conference on Artificial Intelligence and Pattern Recognition.* Florida.

[80] Krishna, C., & Shin, K. (n.d.). *Real-Time Systems.* 2000: McGraw-Hill.

[81] Kurose, J. D., & Towsley, D. (1995). *Further results in affinity-based scheduling of parallel networking.* University of Massachusetts, Department of Computer Science, Amherst, MA 01003.

[82] Kyoung-Don, K., Sang, H. S., & Stankovic, J. (2004). Managing Deadline Miss Ratio and Sensor Data Freshness in Real-Time Databases. *IEEE Transactions on Knowledge And Data Engineering*, *16*(10).

[83] Lee, J., & Shin, K. G. (2014). Preempt a Job or Not in EDF Scheduling of Uniprocessor Systems. *IEEE Transactions On Computers*, *63*(5), 1197-1205.

[84] Lelli, J., Faggioli, D., Cucinotta, T., & Lipari, G. (2012). An experimental comparison of different real-time schedulers on multicore systems. *Journal of Systems and Software*, *85*(10), 2405–2416.

[85] Lelli, J., Scordino, C., Abeni, L., & Faggioli, D. (2016). Deadline scheduling in the linux kernel. *Journal of Software:Practice and Experience*, 821-839. doi:10.1002/spe.2335

[86] Leung, J., & Li, C. (2008). Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, *116*(2), 251-262.

[87] Leung, J., & Whitehead, J. (1982). The Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks. *Performance Evaluation*, 237-250.

[88] Li, Q., & Ba, W. (2012). A group priority earliest deadline first scheduling algorithm. *Frontiers of Computer Science*, *6*(5), 560–567.

[89] Liu, C. L., & Layland, J. W. (1973). Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 46–61. doi:10.1145/321738.321743

[90] Liu, C., & Layland, J. W. (2002). Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. In G. {. Micheli}, & R. E. Wolf (Eds.), *Readings in Hardware/Software Co-Design* (pp. 179-194). San Francisco: Morgan Kaufmann.

[91] Liu, J. (2000). *Real-Time Systems.* Pearson.

[92] Mall, R. (2007). *Real-Time Systems Theory and Practice* (1st ed.). Pearson.

[93] Markatos, E., & LeBlanc, T. (1992). Using processor affinity in loop scheduling on shared-memory multiprocessors. *Proceedings of Supercomputing'92*, (pp. 104-113).

[94] Marwedel, P. (2006). *Embedded System Design* (2nd ed.). Berlin: Springer.

[95] McDonald, I. (1999). Distributed, Configurable Memory Management in an Operating System Supporting Quality of Service. *Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems*, (pp. 191-196).

[96] McDonald, I. (2001). *Memory management in a distributed system of single address space operating systems supporting quality of service.* PhD Thesis, University of Glasgow, UK.

[97] Mehta, H., Owens, R., Irwin, M., Chen, R., & Ghosh, D. (1997). Techniques for low energy software. *International Symposium in Low Power Electronics and Design*, (pp. 72-75).

[98] MicroQuill. (2012). *shbench benchmark tool*. Retrieved from http://www.microquill.com

[99] Mochel, P. (2005). The sysfs filesystem. *In Linux Symposium*, (pp. 313-326). Ottawa, Ontario, Canada.

[100] Mollison, M., Brandenburg, B., & Anderson, J. (2009). Towards unit testing real-time schedulers in LITMUSRT. *Proceedings of the 5th Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, (pp. 33-39).

[101] Mottaghi, M. H., & Zarandi, H. R. (2014). DFTS: A dynamic fault-tolerant scheduling for real-time tasks in multicore processors. *Elsevier Microprocessors and Microsystems, 38*, 88–97.

[102] Nejati, S., Alesio, S., Sabetzadeh, M., & Briand, L. (2012). Modeling and Analysis of CPU Usage in Safety-Critical Embedded Systems to Support Stress Testing. In R. France, *Model Driven Engineering Languages and Systems* (Vol. 7590, pp. 759-775). SpringerNature.

[103] Putchala, M. K., & Bryant, A. R. (2016). Synchron-ITS: An Interactive Tutoring System to Teach Process Synchronization and Shared Memory Concepts in an Operating Systems Course. *International Conference on Collaboration Technologies and Systems CTS*, (pp. 180-187).

[104] Reddy, D., Koufaty, D. A., Brett, P., & Hahn, S. W. (2011). Bridging functional heterogeneity in multicore architectures. *SIGOPS Operating Systems Review, 45*(1), 21-33. doi: 10.1145/1945023.1945028

[105] Rossbach, C. J., Hofmann, O. S., Porter, D. E., Ramadan, H. E., Aditya, B., & Witchel, E. (2007). TxLinux: using and managing hardware transactional memory in an operating system. *Proceedings of the 21st {ACM} Symposium on Operating Systems Principles SOSP*, (pp. 87-102).

[106] Rubin, J. M. (2011, November 1). *Can a computer generate a truly random number?* Retrieved from MIT Engineering Home: https://engineering.mit.edu/engage/ask-an-engineer/can-a-computer-generate-a-truly-random-number/#:~:text=Not%20all%20randomness%20is%20pseudo,can%20generate%20truly%20random%20numbers.&text=There%20are%20devices%20that%20generate,rather%20than%20human%2D

[107] Saini, G. (2005). Application of fuzzy logic to real-time scheduling. *Real -Time Conference.* IEEE-NPSS.

[108] Sanati, B., Albert, M., & Cheng, K. (2016). Online Semi-Partitioned Multiprocessor Scheduling of Soft Real-Time Periodic Tasks for QoS Optimization. *Real-Time and Embedded Technology and Applications Symposium.* IEEE.

[109] Santo, B. (2001, December). Embedded battle royale. *IEEE Spectrum*, pp. 36-42.

[110] Srinivasan, A. (2003). *Efficient and Flexible Fair Scheduling of Real-time Tasks on Multiprocessors.* PhD thesis, University of North Carolina, Chapel Hill.

[111] Srinivasan, A., & Anderson, J. (2006). Optimal rate-based scheduling on multiprocessors. *Journal of Computer and System Science, 72*(6), 1094-1117.

[112] Stafford, R. (2021). *Random Vectors with Fixed Sum*. Retrieved from Mathworks: https://www.mathworks.com/matlabcentral/fileexchange/9700-random-vectors-with-fixed-sum

[113] Su, H., & Zhu, D. (2013). An elastic mixed-criticality task model and its scheduling algorithm. *Proceedings of the 2013 Design, Automation Test in Europe Conference Exhibition*, (pp. 147-152).

[114] Suzhen, L. (2005). *Feedback-based task scheduling in real-time systems.* PhD thesis, The Iowa State University.

[115] Tanenbaum, A. S. (n.d.). *Modern Operating Systems* (3rd ed.). Pearson Education Publication.

[116] Thoen, F., Goossens, G., Hugo, & Cornero, M. (2002). Real-Time Multi-Tasking in Software Synthesis for Information Processing Systems. In G. {. Micheli}, & R. E. Wolf (Eds.), *Readings in Hardware/Software Co-Design* (pp. 389-394). San Francisco: Morgan Kaufmann.

[117] Torvalds, L. (2011). *Source codes of linux kernel v3.0.4.* Retrieved from https://github.com/erik96/Linux-Kernel-3.4

[118] V. Singh, K. V., & Khatri, S. (2014). Analytical Study on Scheduling Algorithms for Real Time Static System. *Proceedings of the International Conference on Recent Advances in Communication, VLSI & Embedded Systems*, (pp. 179-182).

[119] Wei, P., Yue, L., Liu, Z., & Xiaoyan, X. (2008). Flash memory management based on predicted data expiry-time in embedded real-time systems. *Symposium on Applied Computing (SAC)* (pp. 1477-1481). Fortaleza, Ceara, Brazil: Proceedings of the 2008 {ACM}. Retrieved March 16-20, 2008

Publications & Patent

1. Donga, J., & Holia, M. (October, 2018). The Comparative Analysis of Scheduling Approaches in Real-Time Systems. *International Journal of Technical Innovation in Modern Engineering & Science* (*IJTIMES*), *4*(10), (pp. 433-438). Impact Factor: 5.22 (SJIF-2017), (ISSN: e-ISSN: 2455-2585). *(UGC Approved)*
2. Donga, J., & Holia, M. (December, 2019). An Analysis of scheduling algorithms in Real-Time operating system. *Lecture Notes in Networks and Systems 98 (LNNS) with Inventive Computation Technologies* (ISSN: 2367- 3370, ISSN 2367-3389 (electronic),ISBN 978-3-030-33845-9, ISBN 978-3-030-33846-6, https://doi.org/10.1007/978-3-030-33846-6), Springer Series. (pp. 374-381). *(SCOPUS, INSPEC, WTI Frankfurt eG, zbMATH, SCImago).*
3. Donga, J., Holia, M., & Patel., N. (April, 2021). The Classification and Comparative study of Real-Time Task Scheduling Algorithms based on various parameters. *Algorithms for Intelligent Systems (AIS)* (ISSN: 2524-7565, ISSN 2524-7573 (electronic), ISBN 978-981-33-4603-1 ISBN 978-981-33-4604-8, https://doi.org/10.1007/978-981-33-4604-8), Springer Series. (pp.239-246).
4. Donga, J., & Holia, M. (December, 2020). Processor Affinity based Multiprocessor Scheduling algorithm for Soft Real-Time System. *Solid State Technology* (ISSN: 0038-111X), 63(5). *(Google Scholar, SCOPUS, Ei Compendex)*.
5. Donga, J., & Holia, M. (December, 2020). A Hybrid Multiprocessor Scheduler for Soft Real-Time System with Processor Affinity Concept. *International Conference on Applied Mathematics, Modeling and Simulation in Engineering 2021 (AMSE-2021) published in IOP SCIENCE (JPCS)* [doi:10.1088/I Online ISSN: 1742-6596 /Print ISSN: 1742- 6588], Volume 2089-*(Scopus indexed).*

Patent

1. Donga, J., Holia, M., & Shah., V. “PrAMS for Scheduling in Real-Time Operating System using LITMUSRT” Indian Patent, Application No: 202121019254 A, Published date: 02 July 2021.

Annexure - I

Detailed Results to find Deadline Miss Ratio

Case 1.1

Average Deadlinemiss Ratio for 100 Task Sets with 8 Processors and Tasksetsize=10

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	0.0921	0.0697	0.0231		**51**	0.0960	0.0680	0.0237
2	0.0959	0.0709	0.0247		**52**	0.0928	0.0715	0.0222
3	0.0957	0.0680	0.0226		**53**	0.0934	0.0713	0.0240
4	0.0934	0.0703	0.0227		**54**	0.0934	0.0686	0.0202
5	0.0960	0.0713	0.0240		**55**	0.0936	0.0687	0.0246
6	0.0936	0.0687	0.0209		**56**	0.0951	0.0690	0.0235
7	0.0953	0.0678	0.0229		**57**	0.0933	0.0694	0.0221
8	0.0964	0.0712	0.0208		**58**	0.0929	0.0718	0.0241
9	0.0933	0.0700	0.0201		**59**	0.0926	0.0691	0.0220
10	0.0936	0.0700	0.0235		**60**	0.0922	0.0686	0.0210
11	0.0948	0.0697	0.0233		**61**	0.0947	0.0689	0.0233
12	0.0970	0.0682	0.0249		**62**	0.0932	0.0717	0.0209
13	0.0925	0.0706	0.0221		**63**	0.0930	0.0678	0.0233
14	0.0928	0.0691	0.0216		**64**	0.0926	0.0684	0.0217
15	0.0927	0.0717	0.0238		**65**	0.0962	0.0705	0.0235
16	0.0924	0.0672	0.0214		**66**	0.0926	0.0713	0.0200
17	0.0943	0.0678	0.0226		**67**	0.0935	0.0688	0.0219
18	0.0946	0.0702	0.0248		**68**	0.0966	0.0671	0.0232
19	0.0932	0.0698	0.0214		**69**	0.0931	0.0673	0.0218
20	0.0947	0.0685	0.0200		**70**	0.0967	0.0698	0.0233
21	0.0968	0.0712	0.0201		**71**	0.0952	0.0676	0.0246
22	0.0954	0.0716	0.0230		**72**	0.0933	0.0719	0.0202
23	0.0950	0.0700	0.0207		**73**	0.0922	0.0696	0.0227
24	0.0939	0.0711	0.0241		**74**	0.0954	0.0717	0.0201
25	0.0943	0.0712	0.0219		**75**	0.0925	0.0702	0.0220
26	0.0952	0.0712	0.0245		**76**	0.0968	0.0685	0.0250
27	0.0929	0.0680	0.0223		**77**	0.0958	0.0715	0.0224
28	0.0927	0.0715	0.0201		**78**	0.0957	0.0684	0.0201
29	0.0946	0.0709	0.0223		**79**	0.0931	0.0688	0.0245
30	0.0939	0.0681	0.0236		**80**	0.0959	0.0692	0.0250
31	0.0924	0.0686	0.0226		**81**	0.0966	0.0708	0.0238
32	0.0926	0.0672	0.0233		**82**	0.0946	0.0704	0.0224
33	0.0951	0.0703	0.0209		**83**	0.0921	0.0691	0.0217
34	0.0954	0.0701	0.0240		**84**	0.0934	0.0686	0.0224
35	0.0930	0.0671	0.0250		**85**	0.0956	0.0700	0.0238
36	0.0953	0.0701	0.0230		**86**	0.0944	0.0708	0.0236
37	0.0958	0.0691	0.0209		**87**	0.0927	0.0695	0.0206
38	0.0942	0.0704	0.0203		**88**	0.0961	0.0687	0.0218
39	0.0937	0.0680	0.0220		**89**	0.0929	0.0708	0.0215
40	0.0961	0.0689	0.0237		**90**	0.0970	0.0719	0.0249
41	0.0939	0.0693	0.0234		**91**	0.0957	0.0698	0.0225
42	0.0944	0.0671	0.0230		**92**	0.0964	0.0708	0.0214
43	0.0936	0.0671	0.0239		**93**	0.0966	0.0703	0.0243
44	0.0956	0.0699	0.0235		**94**	0.0956	0.0712	0.0248
45	0.0933	0.0714	0.0228		**95**	0.0944	0.0687	0.0201
46	0.0962	0.0675	0.0246		**96**	0.0960	0.0709	0.0216
47	0.0970	0.0698	0.0239		**97**	0.0967	0.0720	0.0237
48	0.0962	0.0675	0.0208		**98**	0.0961	0.0678	0.0237
49	0.0959	0.0680	0.0207		**99**	0.0926	0.0699	0.0215
50	0.0957	0.0698	0.0248		**100**	0.0964	0.0685	0.0221
Avg of 100 Attempts	**0.0945**	**0.0696**	**0.0226**					

Case 1.12

Average Deadlinemiss Ratio for 100 Task Sets with 8 Processors and Tasksetsize=32

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	0.9084	0.7824	0.7815		**51**	0.8744	0.7851	0.7973
2	0.8700	0.7784	0.7971		**52**	0.8749	0.8014	0.7764
3	0.9080	0.7858	0.7782		**53**	0.8612	0.7717	0.7837
4	0.8861	0.7914	0.7559		**54**	0.8749	0.7730	0.7786
5	0.8661	0.8138	0.7747		**55**	0.8743	0.8175	0.7560
6	0.8641	0.7949	0.7733		**56**	0.8960	0.7915	0.7899
7	0.8827	0.7969	0.7967		**57**	0.8749	0.7840	0.7737
8	0.8861	0.7742	0.7691		**58**	0.8810	0.7731	0.7636
9	0.8882	0.7944	0.7527		**59**	0.9032	0.8154	0.7735
10	0.8692	0.7911	0.7560		**60**	0.8831	0.8052	0.7918
11	0.8649	0.7853	0.7660		**61**	0.9096	0.7882	0.7678
12	0.8814	0.7906	0.7779		**62**	0.9055	0.8035	0.7686
13	0.8773	0.8165	0.7982		**63**	0.9040	0.8153	0.7588
14	0.8693	0.7916	0.7525		**64**	0.8910	0.7833	0.7523
15	0.9029	0.8179	0.7779		**65**	0.8960	0.7896	0.7826
16	0.9100	0.8178	0.7596		**66**	0.8646	0.8081	0.7713
17	0.9036	0.7958	0.7774		**67**	0.8706	0.8001	0.7871
18	0.8658	0.8129	0.7773		**68**	0.8707	0.8015	0.7877
19	0.8665	0.7710	0.7901		**69**	0.8650	0.7798	0.7823
20	0.8981	0.8044	0.7709		**70**	0.8973	0.7714	0.7865
21	0.9028	0.7884	0.7737		**71**	0.8719	0.8014	0.7602
22	0.9099	0.8133	0.7522		**72**	0.8861	0.7937	0.7806
23	0.9077	0.7736	0.7960		**73**	0.8891	0.7937	0.7985
24	0.8934	0.8200	0.7926		**74**	0.8850	0.7956	0.7936
25	0.8826	0.7965	0.7890		**75**	0.8893	0.8179	0.7995
26	0.9000	0.8152	0.7671		**76**	0.8840	0.7897	0.7663
27	0.8625	0.8105	0.7905		**77**	0.9028	0.8146	0.7929
28	0.8948	0.7957	0.7502		**78**	0.8742	0.8012	0.7999
29	0.9001	0.8162	0.7680		**79**	0.8733	0.8114	0.7888
30	0.8885	0.8107	0.7980		**80**	0.9053	0.8006	0.7720
31	0.8600	0.8000	0.7898		**81**	0.8635	0.8083	0.7581
32	0.8998	0.7898	0.7658		**82**	0.8982	0.7762	0.7520
33	0.8658	0.8021	0.7902		**83**	0.8922	0.8163	0.7537
34	0.8940	0.7718	0.7840		**84**	0.8908	0.8116	0.7752
35	0.8638	0.7752	0.7995		**85**	0.9093	0.8056	0.7837
36	0.8956	0.7995	0.7669		**86**	0.8665	0.7863	0.7715
37	0.8940	0.7950	0.7921		**87**	0.8916	0.8071	0.7531
38	0.8738	0.7818	0.7992		**88**	0.8893	0.7965	0.7901
39	0.8709	0.8134	0.7884		**89**	0.9083	0.7975	0.7956
40	0.8805	0.7979	0.7534		**90**	0.8917	0.8009	0.7671
41	0.9066	0.7815	0.7681		**91**	0.8652	0.8170	0.7623
42	0.8674	0.8126	0.7806		**92**	0.8750	0.7858	0.7507
43	0.8667	0.8139	0.7903		**93**	0.8803	0.8031	0.7514
44	0.8668	0.8060	0.7699		**94**	0.8734	0.7932	0.7757
45	0.8681	0.7957	0.7973		**95**	0.8787	0.7767	0.7805
46	0.8810	0.7952	0.7632		**96**	0.9072	0.8023	0.7517
47	0.8810	0.7990	0.7805		**97**	0.8987	0.7882	0.7992
48	0.8734	0.8125	0.7591		**98**	0.8688	0.8067	0.7836
49	0.8753	0.7715	0.7738		**99**	0.8793	0.7858	0.7628
50	0.8846	0.8148	0.7541		**100**	0.8791	0.7945	0.7901
Avg of 100 Attempts	**0.8842**	**0.7971**	**0.7762**					

Similarly, Case 1.2 to **Case 1.11**

Average Deadlinemiss Ratio for 100 Task Sets with 8 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Deadline Miss Ratio For, All Result, **Scan QR Code:** Proc 8_Case 1.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 1.2	**12**	**Click Here**
Case 1.3	**14**	**Click Here**
Case 1.4	**16**	**Click Here**
Case 1.5	**18**	**Click Here**
Case 1.6	**20**	**Click Here**
Case 1.7	**22**	**Click Here**
Case 1.8	**24**	**Click Here**
Case 1.9	**26**	**Click Here**
Case 1.10	**28**	**Click Here**
Case 1.11	**30**	**Click Here**

Detailed Results to find Average Tardiness

Case 2.1

Average Tardiness for 100 Task Sets with 8 Processors and Tasksetsize=10

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	20.88	11.11	6.96		**51**	20.19	12.95	8.88
2	18.46	9.04	8.34		**52**	21.17	13.75	6.60
3	18.28	11.27	7.11		**53**	19.09	12.71	6.02
4	19.98	9.53	8.38		**54**	19.25	12.14	10.79
5	17.51	10.04	6.41		**55**	18.84	11.16	8.77
6	17.57	13.62	7.50		**56**	18.35	10.49	6.53
7	19.80	11.37	9.41		**57**	19.46	9.81	10.69
8	18.68	10.83	6.74		**58**	18.88	13.73	10.30
9	19.16	12.09	6.39		**59**	20.32	10.37	8.57
10	18.26	10.47	8.38		**60**	19.39	13.84	10.18
11	19.96	10.75	10.51		**61**	18.01	11.03	9.62
12	17.18	12.32	10.60		**62**	17.46	13.25	6.08
13	19.84	10.97	10.35		**63**	19.02	11.73	10.39
14	21.38	9.06	9.40		**64**	18.13	9.66	8.22
15	18.83	12.45	10.34		**65**	17.71	13.03	6.85
16	18.56	11.84	7.41		**66**	17.46	13.23	8.80
17	20.91	13.80	9.91		**67**	20.84	12.31	9.54
18	21.14	12.96	7.60		**68**	17.28	10.01	10.82
19	19.26	13.15	7.45		**69**	18.18	10.00	8.75
20	19.26	9.56	8.34		**70**	20.34	9.17	6.45
21	19.22	10.90	7.80		**71**	19.86	13.25	7.33
22	19.32	9.34	8.50		**72**	17.02	13.87	7.03
23	18.71	10.42	9.59		**73**	19.14	11.31	10.00
24	19.34	13.80	7.11		**74**	17.51	9.47	6.97
25	20.04	9.64	9.74		**75**	18.99	10.24	8.72
26	18.10	11.20	8.70		**76**	21.04	12.69	6.22
27	17.70	9.75	9.70		**77**	21.90	10.59	6.36
28	18.74	13.12	9.54		**78**	18.47	12.36	7.79
29	21.41	9.49	8.78		**79**	20.26	13.25	10.59
30	19.26	11.75	6.51		**80**	17.36	13.40	6.59
31	19.34	12.15	7.15		**81**	17.67	12.18	9.51
32	18.69	12.75	7.33		**82**	17.30	10.74	6.33
33	21.83	9.48	7.40		**83**	21.06	10.30	7.33
34	21.78	9.26	10.59		**84**	19.86	11.16	9.58
35	18.51	13.67	6.69		**85**	20.75	9.54	8.12
36	20.38	10.75	8.96		**86**	17.66	9.16	10.57
37	17.63	11.92	7.25		**87**	20.94	10.65	8.79
38	19.64	11.20	10.00		**88**	20.32	9.70	9.54
39	21.59	10.45	8.15		**89**	21.66	11.54	9.25
40	20.20	13.42	10.78		**90**	20.00	13.34	6.32
41	21.49	9.17	7.15		**91**	21.89	13.96	10.21
42	21.58	11.53	7.23		**92**	20.55	12.06	8.91
43	19.61	11.18	10.77		**93**	19.15	11.38	10.04
44	18.07	11.98	7.61		**94**	21.71	9.66	8.13
45	17.67	11.77	8.62		**95**	20.77	11.53	9.48
46	18.88	10.52	6.07		**96**	20.39	10.49	10.96
47	19.68	12.90	8.42		**97**	20.05	10.51	7.16
48	20.84	9.10	10.66		**98**	19.73	10.50	9.90
49	21.81	12.94	7.39		**99**	20.17	10.07	10.64
50	20.18	11.29	8.60		**100**	21.85	9.29	9.05
Avg of 100 Attempts	**19.51**	**11.36**	**8.51**					

Case 2.12

Average Tardiness for 100 Task Sets with 8 Processors and Tasksetsize=32

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	31.36	26.90	17.28		**51**	30.80	30.38	20.83
2	30.95	26.77	18.96		**52**	28.61	25.48	25.57
3	31.28	30.65	19.70		**53**	35.19	30.58	25.21
4	31.35	24.80	21.41		**54**	28.16	23.36	23.20
5	28.05	23.44	18.22		**55**	30.23	25.62	18.03
6	31.40	26.09	18.86		**56**	31.93	24.38	19.44
7	27.99	22.36	17.55		**57**	31.37	24.76	27.27
8	29.88	22.13	20.13		**58**	34.55	23.11	21.60
9	33.46	23.05	19.06		**59**	31.54	30.85	21.38
10	27.96	30.81	21.09		**60**	30.95	30.83	19.53
11	30.44	26.14	21.19		**61**	34.16	30.32	21.48
12	29.80	25.52	25.19		**62**	36.22	26.77	25.36
13	30.61	22.66	25.58		**63**	28.06	25.63	25.35
14	30.25	22.74	18.63		**64**	30.32	30.49	25.50
15	29.81	22.08	18.91		**65**	35.49	23.62	24.16
16	28.23	25.54	21.15		**66**	32.43	25.21	20.37
17	27.47	30.29	18.71		**67**	28.08	24.59	20.05
18	29.65	24.48	20.64		**68**	35.64	26.11	19.39
19	28.98	26.59	19.52		**69**	30.26	30.55	25.07
20	28.53	24.79	20.38		**70**	33.02	26.79	20.87
21	31.82	23.91	21.63		**71**	34.13	25.83	19.24
22	35.77	22.04	18.99		**72**	32.71	28.70	21.90
23	29.01	26.81	25.62		**73**	35.88	25.07	21.48
24	27.82	26.28	20.95		**74**	34.78	30.45	25.66
25	29.84	26.95	25.51		**75**	28.69	30.65	21.76
26	34.38	22.45	18.20		**76**	35.05	26.98	25.16
27	27.97	26.53	25.76		**77**	30.56	30.79	22.00
28	31.88	23.17	20.33		**78**	27.61	24.78	25.11
29	31.28	26.36	25.59		**79**	29.56	30.32	20.68
30	31.39	22.42	19.43		**80**	34.77	24.39	17.65
31	34.67	26.05	18.17		**81**	32.01	30.32	23.86
32	27.99	22.31	19.14		**82**	31.33	25.04	20.66
33	28.00	22.25	18.41		**83**	35.48	30.36	23.30
34	28.70	22.13	21.69		**84**	31.34	30.97	21.90
35	29.38	30.86	19.33		**85**	29.49	26.14	20.27
36	35.62	22.83	25.05		**86**	31.57	30.10	18.42
37	28.38	30.44	20.94		**87**	33.98	30.16	22.93
38	30.70	23.17	20.23		**88**	31.40	25.66	21.46
39	31.48	22.18	23.50		**89**	28.17	30.58	25.79
40	30.07	25.47	22.32		**90**	35.30	28.63	25.62
41	35.65	30.33	21.50		**91**	34.44	30.73	22.72
42	36.93	26.75	17.95		**92**	30.91	25.30	21.06
43	31.86	25.32	18.25		**93**	28.36	28.38	21.77
44	30.98	26.89	19.80		**94**	29.49	24.55	21.47
45	30.93	30.79	17.85		**95**	33.17	24.10	25.03
46	36.59	24.99	20.34		**96**	31.35	30.67	20.50
47	29.92	26.72	25.68		**97**	34.42	26.04	19.38
48	30.82	25.28	20.17		**98**	31.29	28.69	21.89
49	36.34	28.06	23.73		**99**	29.38	30.37	22.66
50	30.05	30.97	20.25		**100**	30.21	30.45	25.89
Avg of 100 Attempts	**31.38**	**26.63**	**21.55**					

Similarly, Case 2.2 to **Case 2.11**

Average Tardiness for 100 Task Sets with 8 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average Tardiness For, All Result, **Scan QR Code:** Proc 8_Case 2.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 2.2	**12**	**Click Here**
Case 2.3	**14**	**Click Here**
Case 2.4	**16**	**Click Here**
Case 2.5	**18**	**Click Here**
Case 2.6	**20**	**Click Here**
Case 2.7	**22**	**Click Here**
Case 2.8	**24**	**Click Here**
Case 2.9	**26**	**Click Here**
Case 2.10	**28**	**Click Here**
Case 2.11	**30**	**Click Here**

Detailed Results to find Average No. of Context Switch

Case 3.1

Average No. of Context Switch for 100 Task Sets with 8 Processors and Tasksetsize=10

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	57	66	71		**51**	60	66	73
2	59	68	70		**52**	60	66	69
3	62	69	74		**53**	58	66	72
4	58	67	70		**54**	59	67	72
5	62	67	70		**55**	59	70	69
6	61	69	70		**56**	59	65	74
7	58	70	74		**57**	59	67	71
8	62	68	70		**58**	62	70	71
9	58	70	72		**59**	61	67	72
10	60	70	71		**60**	61	69	74
11	58	67	69		**61**	62	70	71
12	59	70	70		**62**	58	70	69
13	60	66	71		**63**	61	67	73
14	57	65	69		**64**	60	67	69
15	59	69	71		**65**	60	70	70
16	60	68	69		**66**	59	70	71
17	57	65	73		**67**	60	67	71
18	61	66	69		**68**	60	69	70
19	61	65	73		**69**	60	66	73
20	61	67	71		**70**	60	66	72
21	60	69	72		**71**	59	67	70
22	59	67	73		**72**	59	69	71
23	58	68	72		**73**	58	66	74
24	62	65	70		**74**	60	66	73
25	61	67	70		**75**	59	67	72
26	61	66	70		**76**	60	65	71
27	62	66	74		**77**	57	66	73
28	58	69	72		**78**	61	67	71
29	59	66	73		**79**	59	65	72
30	57	70	71		**80**	58	68	71
31	59	66	71		**81**	61	66	73
32	61	70	72		**82**	59	67	70
33	60	67	69		**83**	61	65	69
34	59	66	70		**84**	59	67	70
35	60	70	70		**85**	58	66	69
36	59	68	73		**86**	61	69	74
37	59	70	71		**87**	59	68	71
38	60	67	71		**88**	59	69	70
39	62	69	73		**89**	57	67	71
40	59	67	70		**90**	60	69	70
41	60	70	72		**91**	59	67	70
42	59	68	73		**92**	58	65	72
43	60	69	72		**93**	57	68	72
44	59	65	69		**94**	58	65	72
45	61	69	71		**95**	58	68	73
46	59	69	71		**96**	61	69	71
47	58	70	71		**97**	59	68	71
48	61	70	71		**98**	57	66	72
49	57	69	71		**99**	58	68	72
50	61	65	70		**100**	61	69	73
Avg of 100 Attempts	**59.48**	**67.56**	**71.24**					

Case 3.12

Average No. of Context Switch for 100 Task Sets with 8 Processors and Tasksetsize=32

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	168	213	197		**51**	165	206	192
2	164	220	194		**52**	168	201	194
3	162	206	191		**53**	169	220	191
4	166	201	195		**54**	166	215	194
5	168	204	196		**55**	164	205	194
6	169	218	195		**56**	169	217	191
7	165	217	196		**57**	168	220	192
8	167	216	196		**58**	163	214	193
9	167	204	196		**59**	163	216	192
10	165	215	190		**60**	167	208	193
11	163	217	193		**61**	168	204	190
12	163	212	195		**62**	162	215	194
13	164	207	194		**63**	165	212	192
14	164	206	190		**64**	169	220	192
15	169	208	195		**65**	164	214	196
16	164	218	194		**66**	165	218	192
17	165	217	197		**67**	164	212	190
18	163	220	195		**68**	163	219	190
19	162	204	195		**69**	168	215	197
20	163	218	197		**70**	164	216	192
21	169	208	191		**71**	168	218	195
22	168	205	193		**72**	166	220	196
23	167	204	193		**73**	168	220	196
24	164	204	192		**74**	165	212	192
25	166	217	192		**75**	167	207	197
26	165	216	197		**76**	168	218	192
27	168	220	194		**77**	165	216	191
28	165	215	192		**78**	168	215	192
29	168	217	196		**79**	165	214	195
30	163	212	193		**80**	166	222	193
31	164	219	193		**81**	164	217	194
32	164	206	194		**82**	165	214	193
33	168	208	196		**83**	168	213	191
34	164	218	196		**84**	166	216	195
35	166	217	194		**85**	168	215	193
36	169	216	196		**86**	163	215	195
37	166	218	195		**87**	166	217	192
38	163	215	194		**88**	167	216	190
39	163	217	195		**89**	164	220	194
40	165	212	197		**90**	168	215	193
41	164	215	194		**91**	169	217	192
42	162	206	191		**92**	165	212	190
43	164	208	196		**93**	169	218	195
44	164	218	197		**94**	169	213	190
45	165	212	196		**95**	167	214	194
46	168	216	191		**96**	167	216	192
47	167	220	196		**97**	163	209	190
48	167	214	191		**98**	164	205	191
49	167	218	190		**99**	165	220	192
50	168	216	194		**100**	168	218	191
Avg of 100 Attempts	**165.77**	**213.77**	**193.47**					

Similarly, Case 3.2 to **Case 3.11**

Average No. of Context Switch for 100 Task Sets with 8 Processors and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average No. of Context Switch For, All Result, **Scan QR Code:** Proc 8_Case 3.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 3.2	**12**	**Click Here**
Case 3.3	**14**	**Click Here**
Case 3.4	**16**	**Click Here**
Case 3.5	**18**	**Click Here**
Case 3.6	**20**	**Click Here**
Case 3.7	**22**	**Click Here**
Case 3.8	**24**	**Click Here**
Case 3.9	**26**	**Click Here**
Case 3.10	**28**	**Click Here**
Case 3.11	**30**	**Click Here**

Detailed Results to find Average Schedulability

Case 4.1

Average Schedulability for 100 Task Sets with 8 Processors and Tasksetsize=10

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	81.37	86.34	87.68		**51**	80.52	84.52	91.14
2	81.66	88.00	88.53		**52**	84.88	85.46	89.24
3	80.70	86.57	89.12		**53**	83.84	84.60	88.93
4	81.41	84.62	87.26		**54**	82.76	88.53	88.85
5	84.44	87.52	90.06		**55**	79.98	85.98	89.46
6	80.82	88.91	91.20		**56**	82.47	84.93	87.27
7	80.12	87.29	87.28		**57**	80.85	85.80	89.41
8	81.86	88.39	90.06		**58**	80.85	88.08	90.46
9	84.18	85.45	91.32		**59**	84.27	88.01	91.58
10	83.47	85.38	91.36		**60**	81.63	85.14	89.47
11	81.27	87.09	91.22		**61**	80.15	86.19	89.00
12	84.67	84.95	91.12		**62**	82.72	84.18	89.20
13	84.58	88.55	91.24		**63**	80.01	85.20	90.00
14	81.25	85.95	89.89		**64**	83.83	88.88	89.49
15	82.99	84.55	87.17		**65**	80.65	86.92	88.63
16	80.59	85.61	88.22		**66**	80.14	86.02	91.58
17	81.69	84.62	87.29		**67**	80.95	88.42	89.00
18	81.86	85.49	91.50		**68**	82.58	85.70	90.41
19	81.91	86.69	90.36		**69**	83.08	85.12	91.47
20	84.89	88.37	88.03		**70**	83.20	84.93	91.36
21	81.21	85.62	90.76		**71**	81.27	86.90	91.85
22	82.19	85.00	89.89		**72**	80.06	84.36	90.12
23	80.12	86.12	90.69		**73**	82.08	86.59	91.62
24	81.07	86.67	90.90		**74**	81.75	85.39	87.50
25	81.44	87.49	91.46		**75**	80.83	87.80	88.90
26	84.42	87.26	89.28		**76**	83.52	85.89	89.53
27	80.31	87.80	88.43		**77**	79.88	84.52	91.82
28	80.46	85.24	88.07		**78**	83.67	85.02	88.85
29	83.39	85.17	88.77		**79**	80.88	87.52	88.70
30	84.91	87.35	87.51		**80**	79.08	87.47	91.13
31	83.41	87.07	87.11		**81**	83.18	84.16	88.60
32	83.80	85.45	87.56		**82**	82.64	83.98	88.09
33	80.23	88.86	87.87		**83**	84.71	86.57	87.08
34	79.94	86.16	90.71		**84**	80.48	84.64	91.63
35	83.43	86.38	89.23		**85**	80.71	88.58	91.55
36	83.08	87.42	87.18		**86**	83.89	85.60	90.36
37	83.20	87.56	89.07		**87**	79.25	86.43	88.88
38	84.95	84.51	87.37		**88**	84.83	85.20	87.27
39	79.73	86.94	91.34		**89**	83.53	87.30	89.85
40	84.24	87.86	89.22		**90**	86.94	86.16	90.42
41	82.38	85.98	88.45		**91**	79.27	88.84	91.70
42	84.80	88.63	89.43		**92**	80.20	87.91	89.20
43	82.64	88.03	87.99		**93**	81.20	84.23	91.26
44	79.99	85.53	91.06		**94**	82.89	85.29	88.89
45	82.90	84.60	88.40		**95**	80.26	86.85	89.95
46	82.17	88.80	91.48		**96**	80.87	88.33	91.09
47	80.55	86.92	90.09		**97**	80.55	86.17	90.15
48	81.82	84.12	90.36		**98**	80.21	84.28	90.93
49	80.53	85.45	89.63		**99**	84.18	88.28	91.08
50	81.09	86.21	87.71		**100**	79.80	84.88	91.28
Avg of 100 Attempts	**82.02**	**86.34**	**89.62**					

Case 4.12

Average Schedulability for 100 Task Sets with 8 Processors and Tasksetsize=32

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	61.40	64.22	75.53		**51**	61.04	63.71	71.28
2	60.57	65.95	72.74		**52**	63.53	64.03	72.78
3	60.14	66.32	71.42		**53**	59.47	63.66	71.17
4	62.07	66.68	71.80		**54**	61.71	63.32	74.15
5	62.46	64.08	74.87		**55**	59.98	64.85	72.49
6	63.92	66.98	72.96		**56**	60.56	66.54	75.64
7	60.25	64.43	73.17		**57**	60.67	62.53	73.44
8	59.89	66.89	75.49		**58**	60.66	65.39	74.99
9	63.69	63.92	74.34		**59**	61.70	62.51	72.63
10	59.37	63.31	73.52		**60**	63.24	63.47	75.86
11	62.76	64.47	71.45		**61**	63.44	64.09	74.02
12	60.11	63.59	71.96		**62**	59.12	66.34	73.13
13	60.40	65.17	75.75		**63**	60.49	62.33	74.51
14	60.02	64.71	72.98		**64**	60.73	62.58	74.26
15	61.51	63.55	72.82		**65**	59.09	65.00	75.79
16	63.40	64.34	73.77		**66**	62.60	65.83	73.30
17	62.21	62.76	72.37		**67**	59.04	63.26	75.90
18	63.96	65.69	75.96		**68**	61.33	66.68	72.13
19	60.37	62.58	74.76		**69**	60.53	63.04	71.95
20	59.31	63.00	74.87		**70**	59.51	65.55	73.47
21	59.38	66.15	72.64		**71**	59.96	65.35	72.59
22	63.01	62.56	71.13		**72**	62.92	65.43	73.03
23	63.66	66.83	75.30		**73**	63.90	62.56	73.03
24	61.11	64.77	71.94		**74**	62.91	66.86	71.04
25	63.31	64.17	74.05		**75**	62.98	65.35	72.82
26	59.66	66.22	71.34		**76**	63.19	63.98	74.65
27	62.26	63.01	71.66		**77**	63.06	66.16	74.52
28	62.15	62.78	75.09		**78**	63.33	64.45	71.25
29	59.80	62.73	73.59		**79**	61.96	62.45	72.91
30	61.92	64.31	72.08		**80**	61.38	65.78	74.78
31	63.28	66.78	72.61		**81**	62.58	65.51	74.40
32	62.03	64.23	73.73		**82**	60.18	66.46	71.48
33	62.75	63.95	75.13		**83**	60.92	66.19	71.32
34	62.72	65.03	73.67		**84**	59.88	67.16	75.13
35	62.09	64.05	71.35		**85**	60.29	65.57	72.61
36	63.22	66.90	75.79		**86**	60.98	65.62	71.77
37	60.22	63.85	73.08		**87**	59.41	64.90	74.05
38	62.31	64.56	75.29		**88**	62.96	67.86	74.85
39	61.11	66.31	75.50		**89**	63.40	64.67	75.59
40	63.78	62.91	74.41		**90**	60.63	65.81	74.18
41	63.57	64.97	73.29		**91**	63.83	65.77	71.20
42	60.53	65.40	74.96		**92**	59.06	66.10	72.52
43	60.75	66.65	75.81		**93**	59.85	64.15	71.01
44	60.17	63.25	72.03		**94**	63.18	63.58	74.17
45	62.18	62.65	72.62		**95**	61.95	64.83	75.63
46	60.27	62.85	73.16		**96**	62.88	66.10	75.29
47	60.40	65.47	71.58		**97**	63.54	66.09	71.30
48	61.63	65.92	75.77		**98**	60.64	64.74	75.00
49	61.23	62.95	75.59		**99**	60.77	65.40	71.27
50	60.63	65.11	74.00		**100**	60.40	66.75	75.49
Avg of 100 Attempts	**61.50**	**64.76**	**73.52**					

Similarly, Case 4.2 to **Case 4.11**

Average Schedulability for 100 Task Sets with 8 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average Schedulability For, All Result, **Scan QR Code:** Proc 8_Case 4.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 4.2	**12**	**Click Here**
Case 4.3	**14**	**Click Here**
Case 4.4	**16**	**Click Here**
Case 4.5	**18**	**Click Here**
Case 4.6	**20**	**Click Here**
Case 4.7	**22**	**Click Here**
Case 4.8	**24**	**Click Here**
Case 4.9	**26**	**Click Here**
Case 4.10	**28**	**Click Here**
Case 4.11	**30**	**Click Here**

Detailed Results to find Average CPU_Utilization

Case 5.1

Average CPU_Utilization for 100 Task Sets with 8 Processors and Tasksetsize=10

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	58.94	64.39	64.29		**51**	58.94	65.93	64.97
2	61.59	66.00	66.64		**52**	57.80	66.92	65.81
3	61.49	64.70	64.93		**53**	61.42	66.28	65.59
4	61.60	62.11	64.58		**54**	60.35	65.59	68.86
5	58.32	66.22	67.11		**55**	60.69	64.26	65.20
6	57.78	66.62	64.21		**56**	59.74	64.79	65.81
7	58.38	64.13	68.63		**57**	57.81	65.99	66.65
8	61.58	64.72	65.32		**58**	59.00	64.47	67.22
9	61.31	66.08	68.37		**59**	59.10	62.52	65.91
10	59.22	64.33	68.17		**60**	61.88	63.41	68.25
11	57.10	63.05	65.45		**61**	57.54	64.56	67.83
12	60.82	62.27	66.70		**62**	57.22	63.08	67.58
13	57.75	62.00	65.18		**63**	57.05	62.73	66.22
14	61.60	62.95	67.22		**64**	59.02	66.99	67.99
15	57.80	63.72	65.15		**65**	57.57	63.34	68.01
16	60.51	65.91	65.53		**66**	59.62	65.03	67.21
17	59.02	66.78	65.28		**67**	59.68	63.33	64.52
18	61.59	66.50	67.33		**68**	58.95	63.99	68.26
19	60.88	65.53	64.07		**69**	58.21	65.67	64.13
20	61.90	62.26	68.60		**70**	59.54	63.17	67.33
21	58.21	63.81	67.64		**71**	59.99	65.73	65.83
22	61.03	62.54	66.01		**72**	58.74	63.43	65.42
23	60.31	66.13	66.64		**73**	61.50	63.76	67.25
24	57.84	63.39	66.27		**74**	60.47	64.24	68.17
25	58.23	64.61	65.21		**75**	58.98	62.19	67.68
26	57.59	64.41	64.50		**76**	58.06	63.34	64.16
27	61.77	65.72	65.80		**77**	57.66	63.95	65.48
28	59.46	62.47	67.22		**78**	57.74	63.63	66.49
29	61.37	64.92	68.51		**79**	57.94	64.71	64.06
30	59.94	66.54	67.45		**80**	57.40	63.75	64.05
31	59.00	64.89	65.02		**81**	58.42	66.54	65.72
32	61.13	63.44	67.69		**82**	58.21	65.77	66.80
33	58.66	64.47	64.47		**83**	59.52	64.48	68.92
34	57.36	66.28	68.01		**84**	58.01	63.64	67.72
35	59.36	64.47	68.46		**85**	57.65	63.00	65.74
36	57.35	62.48	64.35		**86**	58.00	65.84	66.01
37	57.85	64.01	66.05		**87**	61.83	62.96	60.07
38	58.15	65.62	68.04		**88**	59.25	61.97	66.68
39	58.49	64.99	67.50		**89**	57.74	62.73	66.19
40	58.99	64.73	68.54		**90**	59.34	63.79	66.51
41	57.75	66.41	66.89		**91**	61.54	62.90	66.79
42	61.67	65.10	64.75		**92**	58.46	61.64	65.40
43	61.90	66.00	64.18		**93**	59.49	64.02	68.69
44	60.15	66.77	65.03		**94**	58.43	64.42	65.08
45	57.60	64.20	64.15		**95**	60.79	66.98	65.86
46	61.76	63.23	64.10		**96**	58.35	63.96	66.51
47	60.79	64.26	68.98		**97**	58.32	62.95	67.76
48	57.75	65.65	67.04		**98**	58.42	62.16	67.25
49	59.32	64.28	66.12		**99**	57.89	64.17	67.80
50	60.25	63.19	65.90		**100**	58.35	63.02	65.68
Avg of 100 Attempts	**59.28**	**64.37**	**66.32**					

Case 5.12

Average CPU_Utilization for 100 Task Sets with 8 Processors and Tasksetsize=32

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	75.17	78.36	78.62		**51**	71.93	81.40	80.57
2	71.20	79.15	82.27		**52**	72.00	77.60	79.13
3	74.03	79.63	78.14		**53**	71.87	80.41	78.62
4	73.54	81.95	80.67		**54**	75.07	78.15	81.86
5	73.59	77.04	79.58		**55**	74.50	78.01	78.25
6	72.57	78.07	80.76		**56**	75.56	80.95	78.64
7	71.96	78.97	78.84		**57**	72.14	77.00	80.74
8	73.21	78.94	79.02		**58**	71.66	77.17	78.16
9	73.34	78.87	78.65		**59**	71.23	80.14	82.90
10	73.92	78.36	82.46		**60**	71.31	81.57	78.74
11	71.24	80.29	81.29		**61**	74.48	77.90	80.05
12	72.19	80.82	80.12		**62**	71.75	78.85	80.47
13	74.32	77.36	80.92		**63**	73.70	79.08	79.78
14	75.82	77.41	82.87		**64**	74.71	80.54	78.62
15	74.61	77.41	79.15		**65**	71.87	77.86	82.50
16	72.04	77.71	81.41		**66**	75.81	81.45	80.52
17	71.37	77.23	81.86		**67**	75.99	78.68	80.27
18	74.02	78.07	78.05		**68**	71.35	80.80	78.34
19	71.54	77.72	79.93		**69**	71.22	77.23	81.12
20	72.83	77.29	81.95		**70**	71.10	81.95	81.40
21	73.85	78.24	82.74		**71**	75.59	78.95	80.43
22	71.99	79.76	80.89		**72**	73.00	77.16	79.28
23	73.62	81.66	82.03		**73**	73.22	81.64	80.73
24	72.55	80.63	80.45		**74**	73.86	80.52	81.89
25	73.12	80.03	78.15		**75**	74.17	81.85	81.18
26	74.54	78.97	82.63		**76**	75.46	77.45	81.11
27	74.23	80.98	81.30		**77**	71.26	79.12	80.53
28	73.96	77.23	79.54		**78**	72.85	81.30	79.68
29	75.03	80.56	82.65		**79**	74.69	80.53	82.01
30	75.10	77.17	78.03		**80**	74.99	81.86	80.15
31	75.52	80.69	82.19		**81**	71.56	77.81	79.21
32	75.52	80.91	80.15		**82**	74.43	79.37	79.39
33	73.33	78.02	79.30		**83**	71.23	80.34	82.43
34	72.56	80.71	80.82		**84**	72.82	80.77	81.73
35	74.46	80.87	80.13		**85**	74.42	81.94	80.99
36	72.91	79.09	78.01		**86**	73.17	80.58	78.99
37	75.62	78.45	81.60		**87**	74.77	79.11	81.50
38	73.30	80.10	79.70		**88**	74.06	79.02	82.00
39	74.00	78.60	82.00		**89**	75.94	77.17	81.55
40	72.61	77.38	82.97		**90**	75.08	81.87	82.44
41	71.00	78.71	79.55		**91**	75.15	79.95	80.35
42	72.48	77.92	78.73		**92**	74.39	79.60	82.42
43	75.60	79.51	79.42		**93**	72.64	80.91	81.30
44	74.93	79.56	82.33		**94**	73.97	79.83	81.89
45	71.85	80.18	78.71		**95**	75.34	80.56	78.70
46	75.32	80.59	80.97		**96**	72.01	80.41	81.30
47	75.18	77.43	80.20		**97**	74.13	81.08	78.57
48	71.25	79.44	82.87		**98**	71.64	77.61	80.01
49	72.00	79.56	80.94		**99**	71.60	79.65	80.20
50	71.63	77.58	78.10		**100**	73.47	77.76	79.98
Avg of 100 Attempts	**73.42**	**79.34**	**80.46**					

Similarly, Case 5.2 to **Case 5.11**

Average CPU_Utilization for 100 Task Sets with 8 Processors

and Taskset size = n

Detailed Results are available as follow)

Detailed Results to find Average CPU_Utilization For, All Result, **Scan QR Code:** Proc 8_Case 5.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 5.2	**12**	**Click Here**
Case 5.3	**14**	**Click Here**
Case 5.4	**16**	**Click Here**
Case 5.5	**18**	**Click Here**
Case 5.6	**20**	**Click Here**
Case 5.7	**22**	**Click Here**
Case 5.8	**24**	**Click Here**
Case 5.9	**26**	**Click Here**
Case 5.10	**28**	**Click Here**
Case 5.11	**30**	**Click Here**

Annexure - II

Detailed Results to find Deadline Miss Ratio

Case 1.1

Average Deadlinemiss Ratio for 100 Task Sets with 12 Processors and Tasksetsize=16

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	0.2179	0.0946	0.0367		**51**	0.2140	0.1062	0.0593
2	0.2051	0.0961	0.0689		**52**	0.1919	0.0713	0.0541
3	0.1827	0.0895	0.0349		**53**	0.1968	0.0873	0.0620
4	0.1920	0.0801	0.0545		**54**	0.2086	0.1013	0.0321
5	0.1947	0.0745	0.0611		**55**	0.1714	0.0835	0.0603
6	0.2197	0.0800	0.0550		**56**	0.2087	0.0745	0.0213
7	0.1840	0.0931	0.0290		**57**	0.1892	0.0853	0.0320
8	0.1738	0.0914	0.0647		**58**	0.2127	0.1092	0.0315
9	0.1987	0.0957	0.0658		**59**	0.1713	0.1031	0.0471
10	0.1755	0.0716	0.0597		**60**	0.1930	0.1156	0.0254
11	0.2087	0.1071	0.0521		**61**	0.2101	0.0898	0.0645
12	0.2089	0.1117	0.0384		**62**	0.1880	0.1019	0.0581
13	0.1973	0.0798	0.0491		**63**	0.2042	0.0722	0.0384
14	0.1922	0.0921	0.0663		**64**	0.1794	0.0908	0.0242
15	0.1765	0.0827	0.0451		**65**	0.2120	0.0930	0.0266
16	0.1791	0.0969	0.0453		**66**	0.2127	0.1133	0.0319
17	0.1910	0.0802	0.0656		**67**	0.1898	0.0995	0.0631
18	0.1702	0.0740	0.0332		**68**	0.1795	0.0705	0.0623
19	0.2114	0.0828	0.0465		**69**	0.1702	0.1107	0.0362
20	0.2152	0.0968	0.0402		**70**	0.1814	0.0868	0.0327
21	0.2030	0.1145	0.0627		**71**	0.2096	0.0752	0.0424
22	0.1968	0.0729	0.0654		**72**	0.1931	0.0968	0.0583
23	0.1990	0.1077	0.0345		**73**	0.1872	0.0892	0.0564
24	0.1958	0.0993	0.0455		**74**	0.2098	0.0810	0.0577
25	0.2128	0.1193	0.0290		**75**	0.2044	0.0882	0.0214
26	0.2125	0.0722	0.0675		**76**	0.2200	0.0820	0.0472
27	0.1767	0.0956	0.0635		**77**	0.1774	0.1193	0.0558
28	0.1932	0.0751	0.0308		**78**	0.1742	0.0979	0.0482
29	0.1883	0.1159	0.0689		**79**	0.1867	0.0769	0.0360
30	0.1919	0.0802	0.0359		**80**	0.1751	0.0845	0.0237
31	0.1876	0.0767	0.0333		**81**	0.1978	0.0737	0.0505
32	0.1874	0.0741	0.0507		**82**	0.1775	0.1103	0.0240
33	0.1736	0.1074	0.0403		**83**	0.1846	0.0821	0.0642
34	0.1875	0.0806	0.0457		**84**	0.2058	0.0715	0.0524
35	0.2047	0.0970	0.0512		**85**	0.2111	0.1028	0.0236
36	0.2109	0.0831	0.0522		**86**	0.1841	0.0801	0.0569
37	0.1739	0.1002	0.0645		**87**	0.1848	0.1162	0.0376
38	0.2019	0.0771	0.0679		**88**	0.2165	0.0756	0.0309
39	0.1743	0.0736	0.0405		**89**	0.2178	0.0872	0.0411
40	0.1932	0.1045	0.0547		**90**	0.1967	0.1142	0.0611
41	0.1852	0.1174	0.0281		**91**	0.1858	0.0890	0.0698
42	0.1791	0.1030	0.0657		**92**	0.1730	0.0774	0.0531
43	0.1900	0.1169	0.0222		**93**	0.1816	0.0934	0.0527
44	0.1787	0.1088	0.0426		**94**	0.2155	0.0957	0.0486
45	0.1910	0.1091	0.0354		**95**	0.1914	0.0734	0.0382
46	0.2121	0.1054	0.0221		**96**	0.2107	0.1157	0.0576
47	0.1814	0.0956	0.0225		**97**	0.1743	0.1117	0.0470
48	0.1834	0.0923	0.0607		**98**	0.1893	0.1150	0.0535
49	0.2089	0.0847	0.0573		**99**	0.2133	0.1021	0.0494
50	0.2172	0.1085	0.0309		**100**	0.1924	0.1116	0.0514
Avg of 100 Attempts	**0.1941**	**0.0929**	**0.0468**					

Case 1.9

Average Deadlinemiss Ratio for 100 Task Sets with 12 Processors and Tasksetsize=48

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	0.8993	0.8575	0.8405		**51**	0.8790	0.8399	0.8059
2	0.8900	0.8696	0.8354		**52**	0.8924	0.8594	0.8329
3	0.9060	0.8686	0.8005		**53**	0.9085	0.8335	0.8230
4	0.8979	0.8454	0.8428		**54**	0.9169	0.8673	0.8175
5	0.8887	0.8460	0.8450		**55**	0.8807	0.8571	0.8387
6	0.8992	0.8435	0.8344		**56**	0.9095	0.8608	0.8055
7	0.9084	0.8419	0.8304		**57**	0.8896	0.8420	0.8208
8	0.8940	0.8438	0.8291		**58**	0.9155	0.8563	0.8129
9	0.8989	0.8666	0.8298		**59**	0.8806	0.8318	0.8004
10	0.8738	0.8624	0.8222		**60**	0.9147	0.8332	0.8319
11	0.9197	0.8619	0.8007		**61**	0.9123	0.8589	0.8136
12	0.8857	0.8446	0.8255		**62**	0.8791	0.8333	0.8234
13	0.8771	0.8660	0.8498		**63**	0.9178	0.8460	0.8170
14	0.8737	0.8366	0.8204		**64**	0.8967	0.8472	0.8090
15	0.8999	0.8400	0.8050		**65**	0.8911	0.8728	0.8146
16	0.9007	0.8445	0.8024		**66**	0.9015	0.8452	0.8112
17	0.8708	0.8391	0.8471		**67**	0.8814	0.8615	0.8263
18	0.8732	0.8450	0.8245		**68**	0.9030	0.8626	0.8230
19	0.9102	0.8500	0.8368		**69**	0.8846	0.8722	0.8093
20	0.9009	0.8538	0.8106		**70**	0.8958	0.8351	0.8496
21	0.8928	0.8536	0.8049		**71**	0.9093	0.8727	0.8410
22	0.9138	0.8530	0.8019		**72**	0.9008	0.8774	0.8299
23	0.8719	0.8534	0.8200		**73**	0.9161	0.8618	0.8317
24	0.9173	0.8426	0.8282		**74**	0.9193	0.8599	0.8327
25	0.9200	0.8550	0.8359		**75**	0.8976	0.8373	0.8141
26	0.9054	0.8774	0.8349		**76**	0.8823	0.8658	0.8179
27	0.8737	0.8786	0.8283		**77**	0.9183	0.8673	0.8344
28	0.8965	0.8798	0.8459		**78**	0.9006	0.8419	0.8326
29	0.8752	0.8669	0.8329		**79**	0.9064	0.8752	0.8498
30	0.9109	0.8631	0.8207		**80**	0.8851	0.8623	0.8277
31	0.8855	0.8786	0.8329		**81**	0.8923	0.8387	0.8044
32	0.9012	0.8563	0.8308		**82**	0.8854	0.8539	0.8115
33	0.8768	0.8779	0.8227		**83**	0.8821	0.8390	0.8039
34	0.9016	0.8433	0.8266		**84**	0.9163	0.8307	0.8301
35	0.8724	0.8521	0.8330		**85**	0.8806	0.8633	0.8222
36	0.8873	0.8563	0.8420		**86**	0.9017	0.8624	0.8178
37	0.8799	0.8795	0.8479		**87**	0.9069	0.8480	0.8242
38	0.8728	0.8616	0.8214		**88**	0.9080	0.8480	0.8167
39	0.8735	0.8394	0.8296		**89**	0.9054	0.8595	0.8405
40	0.8938	0.8730	0.8321		**90**	0.9043	0.8653	0.8150
41	0.9047	0.8663	0.8295		**91**	0.8924	0.8580	0.8391
42	0.9139	0.8568	0.8275		**92**	0.9068	0.8378	0.8449
43	0.8855	0.8567	0.8321		**93**	0.8885	0.8510	0.8388
44	0.8729	0.8446	0.8274		**94**	0.9172	0.8528	0.8349
45	0.9032	0.8471	0.8450		**95**	0.9017	0.8717	0.8470
46	0.9023	0.8631	0.8228		**96**	0.8712	0.8723	0.8241
47	0.9078	0.8707	0.8354		**97**	0.8785	0.8597	0.8038
48	0.8827	0.8396	0.8310		**98**	0.9098	0.8322	0.8100
49	0.8768	0.8501	0.8417		**99**	0.9117	0.8681	0.8117
50	0.9000	0.8544	0.8388		**100**	0.9164	0.8378	0.8337
Avg of 100 Attempts	**0.8960**	**0.8551**	**0.8261**					

Similarly, Case 1.2 to **Case 1.8**

Average Deadlinemiss Ratio for 100 Task Sets with 12 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Deadline Miss Ratio For, All Result, **Scan QR Code:** Proc 12_Case 1.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 1.2	**20**	**Click Here**
Case 1.3	**24**	**Click Here**
Case 1.4	**28**	**Click Here**
Case 1.5	**32**	**Click Here**
Case 1.6	**36**	**Click Here**
Case 1.7	**40**	**Click Here**
Case 1.8	**44**	**Click Here**

Detailed Results to find Average Tardiness

Case 2.1

Average Tardiness for 100 Task Sets with 12 Processors and Tasksetsize=16

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	15.59	13.93	12.67		**51**	17.48	12.62	11.63
2	16.28	15.77	10.73		**52**	14.56	14.66	11.37
3	14.79	14.20	10.36		**53**	16.99	15.79	9.34
4	14.11	14.30	12.62		**54**	13.31	12.40	10.86
5	16.03	12.76	10.60		**55**	14.98	15.94	11.73
6	17.53	14.18	10.03		**56**	17.35	15.84	9.52
7	17.80	13.43	12.39		**57**	16.18	11.10	11.95
8	14.39	14.59	11.04		**58**	16.14	12.44	10.33
9	17.66	15.93	10.05		**59**	17.88	11.60	9.64
10	15.97	13.44	12.31		**60**	17.94	11.93	10.78
11	14.69	12.73	10.53		**61**	15.00	12.51	12.02
12	16.95	13.84	8.24		**62**	17.52	12.59	12.85
13	14.63	13.64	11.33		**63**	13.88	13.35	10.88
14	13.23	15.77	11.95		**64**	16.11	15.32	12.31
15	15.08	14.73	11.11		**65**	17.51	11.09	11.72
16	14.72	12.68	8.40		**66**	13.35	14.41	9.64
17	17.31	14.92	9.93		**67**	15.47	15.94	9.96
18	13.17	12.61	9.37		**68**	14.00	11.53	8.50
19	16.08	14.56	10.59		**69**	14.86	13.54	12.67
20	15.91	12.57	12.58		**70**	16.21	11.89	9.33
21	17.62	14.95	10.71		**71**	17.82	15.53	11.12
22	14.04	13.78	10.35		**72**	14.13	11.21	11.25
23	17.20	14.86	8.11		**73**	16.34	15.86	8.44
24	13.51	14.23	10.50		**74**	16.05	13.18	10.86
25	16.37	15.98	11.97		**75**	13.54	11.16	11.49
26	15.47	11.45	11.97		**76**	16.14	12.53	11.12
27	13.01	14.08	8.55		**77**	17.13	11.96	12.22
28	13.97	14.98	10.62		**78**	13.62	13.91	10.83
29	13.90	12.15	9.77		**79**	15.15	11.52	12.17
30	14.40	12.03	8.15		**80**	13.39	15.95	10.28
31	14.39	15.51	9.92		**81**	16.41	13.28	8.61
32	17.17	13.82	8.22		**82**	14.17	14.05	8.41
33	17.50	11.91	12.29		**83**	14.76	12.52	9.84
34	14.03	13.44	11.81		**84**	16.92	14.46	8.28
35	13.64	12.23	10.31		**85**	13.60	15.78	10.70
36	14.29	12.30	11.85		**86**	15.91	15.70	12.54
37	13.99	14.41	11.83		**87**	15.44	12.56	10.60
38	15.84	14.59	9.26		**88**	13.05	14.79	9.60
39	14.11	14.65	9.77		**89**	17.41	12.45	9.41
40	17.57	15.86	9.18		**90**	15.62	14.41	9.92
41	16.63	14.31	10.76		**91**	17.99	15.00	10.98
42	13.39	14.26	12.26		**92**	15.70	11.73	9.01
43	16.18	12.21	12.17		**93**	17.62	15.17	11.78
44	15.24	11.16	12.95		**94**	14.18	13.88	10.54
45	16.65	11.63	12.64		**95**	15.22	13.34	8.11
46	17.86	14.20	11.43		**96**	16.82	12.42	11.76
47	17.57	12.09	12.36		**97**	16.14	11.90	9.54
48	16.06	12.84	12.87		**98**	16.50	13.50	9.35
49	13.46	13.56	11.44		**99**	17.70	11.72	8.68
50	15.54	11.73	12.36		**100**	16.19	15.09	10.53
Avg of 100 Attempts	**15.60**	**13.59**	**10.68**					

Case 2.9

Average Tardiness for 100 Task Sets with 12 Processors and Tasksetsize=48

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	37.54	28.88	26.44		**51**	37.93	27.90	28.26
2	37.98	30.34	29.03		**52**	34.14	28.16	26.33
3	37.37	26.09	28.90		**53**	35.23	26.49	27.75
4	35.88	28.54	26.62		**54**	37.28	28.51	26.40
5	36.81	29.84	27.08		**55**	38.15	26.79	29.70
6	38.86	29.72	29.34		**56**	35.26	27.12	27.65
7	38.84	27.52	27.31		**57**	34.22	27.87	28.73
8	37.26	26.44	25.22		**58**	34.50	29.78	27.02
9	34.60	28.02	26.51		**59**	35.67	27.91	27.12
10	34.35	30.45	28.73		**60**	37.08	26.33	29.28
11	36.27	28.66	27.25		**61**	34.59	29.26	28.93
12	37.75	30.58	29.87		**62**	37.64	26.81	27.39
13	36.79	29.40	28.44		**63**	36.14	27.96	28.66
14	36.53	27.57	28.52		**64**	37.28	29.93	27.97
15	38.70	28.28	29.42		**65**	38.07	27.23	29.85
16	35.20	29.64	29.53		**66**	37.36	29.45	28.96
17	38.02	28.27	25.87		**67**	34.72	29.90	27.08
18	37.01	30.92	26.28		**68**	35.22	29.93	29.05
19	34.46	28.32	27.31		**69**	37.12	30.92	27.36
20	37.36	27.84	25.64		**70**	34.15	26.40	28.50
21	38.03	27.85	29.35		**71**	34.87	26.66	28.19
22	38.65	28.22	27.65		**72**	36.22	26.85	29.55
23	36.21	28.56	27.85		**73**	34.78	27.16	29.39
24	34.26	28.02	27.69		**74**	38.81	28.20	28.78
25	38.67	27.16	26.33		**75**	37.50	26.02	30.00
26	37.65	28.99	27.25		**76**	35.67	27.45	26.42
27	38.90	26.96	27.66		**77**	38.06	29.93	28.39
28	38.79	30.83	28.38		**78**	35.24	29.21	26.82
29	37.71	28.99	28.22		**79**	34.30	29.38	28.78
30	35.80	30.84	25.82		**80**	35.64	29.39	28.12
31	36.98	26.53	25.12		**81**	35.70	28.75	29.56
32	37.04	29.68	28.94		**82**	36.97	29.11	26.83
33	34.22	28.42	29.62		**83**	36.20	26.02	28.93
34	38.41	28.76	28.36		**84**	36.81	29.67	28.56
35	37.10	30.94	28.52		**85**	37.48	26.71	26.56
36	38.40	27.58	28.97		**86**	36.29	29.79	29.67
37	37.12	27.65	26.55		**87**	35.04	28.47	27.79
38	35.04	30.33	27.67		**88**	37.30	30.54	27.90
39	38.19	26.03	27.34		**89**	37.29	27.07	27.45
40	35.21	28.84	26.83		**90**	35.76	29.54	29.94
41	35.45	29.90	28.96		**91**	38.41	30.27	27.09
42	36.36	28.55	26.09		**92**	34.36	26.22	26.68
43	36.66	27.26	29.25		**93**	37.63	28.98	28.12
44	34.33	27.36	28.96		**94**	38.50	27.26	28.96
45	34.60	29.45	25.78		**95**	35.10	30.06	29.13
46	36.34	28.14	25.57		**96**	37.20	26.22	27.27
47	38.23	27.27	25.49		**97**	36.01	28.65	28.31
48	36.41	28.90	29.01		**98**	34.87	28.18	25.06
49	36.24	27.76	26.72		**99**	36.91	29.42	27.87
50	38.35	27.85	29.01		**100**	34.04	29.25	29.70
Avg of 100 Attempts	**36.54**	**28.44**	**27.90**					

Similarly, Case 2.2 to **Case 2.8**

Average Tardiness for 100 Task Sets with 12 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average Tardiness For, All Result, **Scan QR Code:** Proc 12_Case 2.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 2.2	**20**	**Click Here**
Case 2.3	**24**	**Click Here**
Case 2.4	**28**	**Click Here**
Case 2.5	**32**	**Click Here**
Case 2.6	**36**	**Click Here**
Case 2.7	**40**	**Click Here**
Case 2.8	**44**	**Click Here**

Detailed Results to find Average No. of Context Switch

Case 3.1

Average No. of Context Switch for 100 Task Sets with 12 Processors and Tasksetsize=16

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	89	107	122		**51**	86	111	124
2	88	110	127		**52**	88	108	128
3	91	107	123		**53**	92	105	125
4	88	106	124		**54**	87	107	124
5	85	108	122		**55**	87	104	127
6	89	106	121		**56**	90	109	125
7	87	105	125		**57**	88	106	123
8	88	107	122		**58**	86	106	122
9	90	108	122		**59**	92	107	125
10	91	104	125		**60**	91	110	124
11	89	109	121		**61**	87	111	125
12	88	108	127		**62**	89	107	122
13	87	105	125		**63**	91	110	126
14	91	109	123		**64**	91	107	126
15	86	109	123		**65**	92	105	124
16	90	106	123		**66**	91	108	124
17	90	110	124		**67**	92	110	122
18	90	110	122		**68**	87	108	127
19	91	111	124		**69**	89	108	126
20	91	108	123		**70**	86	109	126
21	85	111	125		**71**	88	105	126
22	87	106	123		**72**	87	105	121
23	91	104	125		**73**	89	104	124
24	91	111	121		**74**	91	109	123
25	85	106	123		**75**	85	109	125
26	88	109	123		**76**	90	108	127
27	90	107	127		**77**	91	106	122
28	88	106	124		**78**	87	104	121
29	85	107	121		**79**	88	108	127
30	87	111	121		**80**	90	111	124
31	85	105	123		**81**	89	109	126
32	90	109	123		**82**	86	107	126
33	86	105	124		**83**	89	110	125
34	90	111	126		**84**	87	104	128
35	89	110	123		**85**	90	105	122
36	91	104	123		**86**	90	106	123
37	89	110	121		**87**	91	109	125
38	90	107	125		**88**	87	106	125
39	92	109	125		**89**	89	110	121
40	89	108	121		**90**	86	104	122
41	86	108	121		**91**	91	108	125
42	90	109	128		**92**	91	107	121
43	88	108	122		**93**	92	106	122
44	91	110	125		**94**	89	106	123
45	87	107	128		**95**	85	110	124
46	92	108	122		**96**	86	108	121
47	92	105	127		**97**	85	108	124
48	86	110	128		**98**	89	109	124
49	87	110	127		**99**	91	109	125
50	85	108	124		**100**	91	109	125
Avg of 100 Attempts	**88.73**	**107.67**	**123.99**					

Case 3.9

Average No. of Context Switch for 100 Task Sets with 12 Processors and Tasksetsize=48

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	236	313	291		**51**	236	312	294
2	234	317	286		**52**	235	310	290
3	233	311	293		**53**	232	311	285
4	239	310	286		**54**	233	309	294
5	240	309	285		**55**	236	312	292
6	239	313	287		**56**	239	317	288
7	236	312	291		**57**	239	311	287
8	241	309	292		**58**	235	309	289
9	233	309	286		**59**	239	315	289
10	234	313	290		**60**	238	316	289
11	238	308	291		**61**	241	311	294
12	241	308	292		**62**	235	310	290
13	239	317	293		**63**	233	312	291
14	240	315	293		**64**	234	310	287
15	234	312	290		**65**	237	317	293
16	232	313	288		**66**	240	311	287
17	239	316	290		**67**	239	310	291
18	239	316	288		**68**	238	315	290
19	238	311	291		**69**	234	316	293
20	239	315	292		**70**	233	311	291
21	235	310	291		**71**	235	311	294
22	234	311	289		**72**	238	316	292
23	233	311	291		**73**	234	313	288
24	241	310	291		**74**	235	309	289
25	238	312	293		**75**	233	314	289
26	241	310	288		**76**	237	309	287
27	240	313	291		**77**	236	316	290
28	240	316	291		**78**	240	310	294
29	235	310	293		**79**	238	315	286
30	236	311	288		**80**	234	312	286
31	232	308	289		**81**	239	309	294
32	234	313	288		**82**	237	310	292
33	235	313	293		**83**	233	312	285
34	238	312	289		**84**	238	312	288
35	235	309	292		**85**	240	310	293
36	236	310	287		**86**	241	310	285
37	234	314	287		**87**	239	316	292
38	236	314	288		**88**	240	310	291
39	237	311	291		**89**	233	313	287
40	233	316	289		**90**	239	312	288
41	238	315	291		**91**	237	313	293
42	238	316	294		**92**	239	311	286
43	240	315	286		**93**	237	311	293
44	234	314	288		**94**	233	313	285
45	240	312	294		**95**	233	315	293
46	232	311	288		**96**	236	309	292
47	236	315	293		**97**	241	308	292
48	235	315	285		**98**	237	313	286
49	234	315	286		**99**	237	311	292
50	236	313	291		**100**	237	313	291
Avg of 100 Attempts	**236.62**	**312.23**	**289.87**					

Similarly, Case 3.2 to **Case 3.8**

Average No. of Context Switch for 100 Task Sets with 12 Processors and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average No. of Context Switch

For, All Result, **Scan QR Code:**

Proc 12_Case 3.pdf

Case No.	Taskset size (n)	Link
Case 3.2	**20**	**Click Here**
Case 3.3	**24**	**Click Here**
Case 3.4	**28**	**Click Here**
Case 3.5	**32**	**Click Here**
Case 3.6	**36**	**Click Here**
Case 3.7	**40**	**Click Here**
Case 3.8	**44**	**Click Here**

Detailed Results to find Average Schedulability

Case 4.1

Average Schedulability for 100 Task Sets with 12 Processors and Tasksetsize=16

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	84.70	86.88	93.05		**51**	85.39	86.50	91.31
2	87.63	89.08	95.30		**52**	88.51	89.80	92.52
3	85.67	89.05	91.97		**53**	84.69	89.92	92.54
4	88.13	89.92	91.99		**54**	85.96	86.49	91.28
5	85.54	88.28	91.45		**55**	86.21	87.48	94.50
6	87.22	86.51	93.19		**56**	84.78	90.73	91.42
7	88.50	87.83	95.02		**57**	88.01	86.77	95.97
8	85.01	89.18	93.39		**58**	85.74	90.11	95.36
9	88.39	88.26	91.89		**59**	84.54	86.97	95.93
10	88.46	87.41	93.42		**60**	84.81	87.33	92.49
11	88.08	87.40	94.89		**61**	88.01	88.02	91.41
12	87.26	89.16	95.47		**62**	85.14	88.45	92.15
13	85.49	90.16	95.80		**63**	84.42	88.18	95.91
14	86.60	87.94	94.21		**64**	88.74	90.50	91.46
15	86.32	90.50	91.35		**65**	84.31	90.91	95.48
16	85.57	87.09	95.68		**66**	88.94	89.08	92.36
17	86.15	86.80	92.49		**67**	85.14	86.60	95.44
18	85.61	88.75	92.99		**68**	84.99	89.57	91.81
19	87.78	88.21	94.46		**69**	86.74	88.00	92.01
20	86.12	89.26	94.67		**70**	85.55	90.04	92.66
21	84.57	90.39	91.06		**71**	86.50	86.37	94.98
22	87.69	87.82	91.47		**72**	84.35	87.06	92.00
23	88.22	90.97	93.05		**73**	84.94	90.21	91.83
24	87.93	90.17	94.61		**74**	87.01	89.66	94.29
25	88.10	87.04	93.20		**75**	87.54	87.20	92.13
26	87.31	86.42	92.90		**76**	85.05	89.07	94.13
27	88.57	86.69	94.60		**77**	88.91	88.80	95.49
28	84.57	88.67	91.36		**78**	87.38	88.83	91.43
29	86.09	87.15	94.68		**79**	87.59	89.07	94.87
30	86.59	89.32	91.60		**80**	88.36	88.35	93.87
31	84.34	87.42	92.88		**81**	86.18	88.91	92.85
32	84.26	87.10	91.59		**82**	88.68	90.29	93.24
33	85.71	90.87	94.39		**83**	85.16	88.07	93.89
34	87.66	86.53	91.99		**84**	85.21	86.60	95.53
35	84.72	87.08	95.12		**85**	88.27	87.83	93.07
36	85.22	87.43	94.93		**86**	88.67	90.07	92.73
37	86.40	90.69	94.39		**87**	86.52	87.00	94.51
38	85.49	89.33	91.83		**88**	85.96	87.10	94.30
39	88.40	86.65	92.55		**89**	84.69	88.03	92.82
40	87.81	88.71	91.65		**90**	88.54	90.24	94.59
41	84.67	90.18	93.67		**91**	85.96	90.37	95.48
42	84.49	86.84	92.25		**92**	86.07	88.35	93.05
43	85.20	86.23	92.60		**93**	86.95	88.33	91.09
44	87.39	90.00	94.66		**94**	85.33	88.12	91.44
45	88.88	87.64	92.75		**95**	86.64	88.37	91.91
46	87.71	87.11	92.03		**96**	88.89	90.33	95.39
47	88.03	88.79	91.54		**97**	85.26	89.41	91.38
48	86.87	90.53	92.25		**98**	88.81	89.17	95.18
49	86.05	88.15	93.84		**99**	88.26	88.90	93.56
50	85.12	90.14	95.91		**100**	87.28	88.20	91.75
Avg of 100 Attempts	**86.54**	**88.49**	**93.31**					

Case 4.9

Average Schedulability for 100 Task Sets with 12 Processors and Tasksetsize=48

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	66.76	73.50	76.30		**51**	64.12	73.32	75.70
2	66.41	72.69	75.33		**52**	67.39	73.31	77.78
3	68.39	74.17	77.85		**53**	64.85	72.92	74.04
4	66.11	71.64	73.93		**54**	68.56	75.17	77.39
5	65.05	71.71	77.90		**55**	65.54	75.36	75.43
6	65.22	75.08	77.13		**56**	67.72	71.70	74.46
7	65.82	71.12	77.29		**57**	64.23	74.06	73.52
8	66.46	73.37	76.82		**58**	68.00	73.24	73.86
9	68.59	72.98	73.35		**59**	68.42	73.52	74.49
10	66.10	74.16	74.97		**60**	64.87	72.08	76.16
11	66.32	74.41	74.37		**61**	64.64	72.14	74.49
12	67.65	75.95	76.47		**62**	66.48	73.50	73.65
13	66.64	71.61	75.56		**63**	64.90	71.37	74.45
14	67.64	75.79	73.29		**64**	65.37	71.63	76.16
15	65.76	72.18	77.36		**65**	67.73	75.99	77.39
16	68.65	73.54	73.44		**66**	66.14	71.63	77.71
17	68.22	75.08	76.10		**67**	64.78	74.16	73.69
18	65.33	74.92	76.12		**68**	68.50	72.10	73.28
19	66.62	74.87	74.79		**69**	68.05	73.31	73.62
20	68.01	75.41	73.10		**70**	68.56	75.83	77.55
21	68.60	73.69	73.83		**71**	68.87	71.83	73.79
22	67.75	73.95	77.47		**72**	67.16	72.36	76.27
23	65.21	72.65	75.90		**73**	66.29	74.26	73.36
24	65.93	71.56	76.02		**74**	66.68	73.39	76.43
25	65.30	73.76	73.53		**75**	66.71	71.23	73.02
26	67.47	75.57	75.04		**76**	65.89	72.30	74.10
27	68.77	71.95	76.18		**77**	65.06	71.33	77.29
28	64.74	73.53	74.48		**78**	64.07	73.75	73.94
29	64.26	73.69	75.44		**79**	66.60	73.25	77.38
30	65.89	75.70	77.41		**80**	68.97	73.71	76.97
31	64.99	71.58	75.65		**81**	65.27	74.95	77.47
32	66.62	73.80	76.38		**82**	67.57	73.06	77.32
33	68.14	75.44	73.42		**83**	65.23	73.31	73.92
34	67.24	75.89	76.42		**84**	66.47	72.35	75.97
35	68.40	74.03	73.91		**85**	66.81	74.59	76.98
36	67.34	72.31	75.83		**86**	65.93	75.78	76.75
37	67.88	72.21	75.06		**87**	68.96	71.64	73.74
38	68.16	75.29	75.84		**88**	68.86	75.90	76.45
39	68.68	75.66	74.34		**89**	64.17	75.27	73.32
40	68.91	73.86	77.86		**90**	65.14	73.54	76.58
41	64.61	73.87	77.84		**91**	65.09	74.59	76.02
42	66.98	74.18	73.47		**92**	67.63	75.87	73.44
43	64.23	72.44	74.53		**93**	66.33	73.88	74.05
44	68.05	71.76	74.11		**94**	64.22	74.79	77.83
45	65.41	72.82	74.18		**95**	68.24	75.61	75.10
46	64.80	75.01	75.73		**96**	67.55	74.23	73.20
47	64.29	75.80	75.73		**97**	68.29	71.51	74.35
48	67.69	74.57	76.80		**98**	67.62	71.64	74.30
49	68.15	75.47	74.57		**99**	69.00	73.57	77.65
50	66.24	73.26	73.88		**100**	64.02	74.75	73.07
Avg of 100 Attempts	**66.64**	**73.64**	**75.37**					

Similarly, Case 4.2 to **Case 4.8**

Average Schedulability for 100 Task Sets with 12 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average Schedulability For, All Result, **Scan QR Code:** Proc 12_Case 4.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 4.2	**20**	**Click Here**
Case 4.3	**24**	**Click Here**
Case 4.4	**28**	**Click Here**
Case 4.5	**32**	**Click Here**
Case 4.6	**36**	**Click Here**
Case 4.7	**40**	**Click Here**
Case 4.8	**44**	**Click Here**

Detailed Results to find Average CPU_Utilization

Case 5.1

Average CPU_Utilization for 100 Task Sets with 12 Processors and Tasksetsize=16

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	56.42	61.65	63.24		**51**	53.25	60.48	66.90
2	56.49	61.88	62.82		**52**	53.53	62.83	66.10
3	54.04	63.27	64.41		**53**	54.80	64.42	64.82
4	53.76	63.03	63.79		**54**	57.11	64.85	62.48
5	53.02	61.55	64.19		**55**	56.62	62.40	66.22
6	54.10	61.90	64.71		**56**	53.08	63.77	64.00
7	56.36	60.22	62.82		**57**	54.26	62.94	66.38
8	55.20	64.30	62.05		**58**	54.51	64.80	66.60
9	58.00	60.70	62.76		**59**	57.99	62.35	66.25
10	56.47	60.86	64.83		**60**	53.79	64.05	62.59
11	56.93	63.62	62.91		**61**	56.06	63.85	63.12
12	57.57	64.14	65.55		**62**	57.39	64.98	62.21
13	55.03	60.30	62.60		**63**	56.72	60.56	62.60
14	54.57	63.53	64.22		**64**	57.09	63.37	63.33
15	56.83	62.71	63.77		**65**	56.06	62.94	62.17
16	56.30	62.20	65.08		**66**	55.64	60.46	63.09
17	54.25	60.38	63.64		**67**	53.03	60.84	62.21
18	53.80	60.97	62.20		**68**	55.95	60.37	65.97
19	53.10	61.39	62.61		**69**	56.10	61.21	66.73
20	53.80	60.16	66.29		**70**	55.96	61.99	66.50
21	55.71	60.18	65.52		**71**	56.00	64.86	63.56
22	57.86	63.89	62.13		**72**	54.43	64.06	65.79
23	54.05	61.24	65.61		**73**	56.54	60.41	64.00
24	57.12	63.78	66.97		**74**	56.22	62.48	66.20
25	57.88	60.85	62.74		**75**	57.73	61.30	66.08
26	56.53	61.03	65.72		**76**	55.02	61.79	66.90
27	53.40	64.34	66.96		**77**	56.98	61.11	65.87
28	57.60	64.66	63.36		**78**	57.01	62.09	63.63
29	57.33	63.64	64.54		**79**	54.15	64.96	65.81
30	57.36	60.11	62.29		**80**	56.67	60.49	66.49
31	53.94	64.62	63.29		**81**	54.05	61.17	65.48
32	53.99	62.96	63.20		**82**	56.18	62.91	65.05
33	53.30	62.94	65.83		**83**	57.61	61.81	66.83
34	54.93	64.37	62.18		**84**	56.15	61.31	62.18
35	53.50	62.13	65.68		**85**	56.63	63.68	64.20
36	57.11	62.09	63.24		**86**	54.24	63.42	66.67
37	53.63	63.47	63.29		**87**	56.52	64.80	66.09
38	57.19	63.83	63.28		**88**	55.08	64.73	66.07
39	57.83	64.95	66.98		**89**	55.29	61.16	65.70
40	53.13	64.16	63.79		**90**	56.00	60.26	62.87
41	54.53	62.10	63.63		**91**	56.40	64.85	64.77
42	56.21	63.97	66.58		**92**	56.51	62.20	65.58
43	54.58	64.49	64.72		**93**	55.17	60.67	63.25
44	55.33	60.53	65.05		**94**	55.58	60.43	66.79
45	56.43	64.22	66.47		**95**	53.73	62.76	64.11
46	53.05	64.84	62.53		**96**	56.45	63.26	66.00
47	54.27	60.64	66.18		**97**	57.50	61.64	65.98
48	53.50	64.49	63.87		**98**	56.86	60.96	63.78
49	57.57	61.46	62.38		**99**	55.72	63.19	64.91
50	55.79	62.38	63.71		**100**	56.99	64.28	64.97
Avg of 100 Attempts	**55.59**	**62.54**	**64.52**					

Case 5.9

Average CPU_Utilization for 100 Task Sets with 12 Processors and Tasksetsize=48

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	73.10	74.25	76.45		**51**	74.01	75.02	75.47
2	72.86	74.96	78.41		**52**	71.49	74.12	75.21
3	73.88	78.96	76.94		**53**	74.98	74.58	75.68
4	72.42	76.15	75.46		**54**	73.10	76.04	79.06
5	70.54	78.59	77.73		**55**	70.42	76.10	76.91
6	71.22	77.57	78.99		**56**	74.34	74.84	79.06
7	73.77	74.36	79.11		**57**	70.53	75.79	78.67
8	70.52	74.29	78.84		**58**	73.54	77.85	78.28
9	72.48	76.92	75.69		**59**	70.85	74.84	77.57
10	70.96	74.41	78.60		**60**	74.63	74.13	77.69
11	74.33	75.54	75.79		**61**	71.36	76.90	79.18
12	70.27	74.17	79.37		**62**	70.97	78.63	76.31
13	73.39	76.64	79.95		**63**	72.85	76.91	78.99
14	70.46	75.37	75.32		**64**	71.94	74.59	75.27
15	70.94	77.57	77.25		**65**	73.62	78.96	78.81
16	70.11	75.05	79.55		**66**	72.91	77.17	77.94
17	71.25	78.26	78.03		**67**	71.49	75.44	76.45
18	72.56	76.07	79.76		**68**	73.51	76.50	79.56
19	73.00	76.81	78.86		**69**	72.85	78.11	77.78
20	72.10	74.89	76.98		**70**	70.80	74.91	76.65
21	73.33	77.49	79.81		**71**	73.17	75.17	77.11
22	70.05	78.41	79.08		**72**	70.36	77.74	75.35
23	72.79	78.44	76.49		**73**	70.98	77.16	79.04
24	70.97	76.32	78.34		**74**	72.87	76.31	78.31
25	70.36	77.15	76.52		**75**	72.83	78.63	76.60
26	73.94	75.98	75.57		**76**	71.66	76.56	76.70
27	70.41	78.34	76.36		**77**	72.06	77.05	75.05
28	74.58	75.53	75.99		**78**	70.33	78.12	76.29
29	73.14	76.37	79.01		**79**	72.09	74.16	79.04
30	72.64	76.14	79.22		**80**	73.27	78.31	79.10
31	72.40	74.12	77.86		**81**	72.89	76.34	76.30
32	72.74	76.93	79.79		**82**	72.28	77.20	77.42
33	74.03	75.66	75.01		**83**	74.48	76.45	77.11
34	71.72	76.07	75.37		**84**	71.75	78.74	77.91
35	72.59	74.69	75.25		**85**	74.03	77.79	75.84
36	73.35	77.26	76.05		**86**	72.36	77.07	77.62
37	70.11	76.57	77.32		**87**	74.40	78.34	75.27
38	71.79	76.51	78.32		**88**	71.55	76.27	78.90
39	72.90	78.74	77.77		**89**	71.14	77.46	76.23
40	74.81	77.30	75.22		**90**	73.44	75.17	79.80
41	74.76	78.87	79.04		**91**	71.11	77.89	78.13
42	71.91	75.14	76.30		**92**	70.38	74.32	78.84
43	73.35	75.51	77.82		**93**	74.37	74.12	77.58
44	74.40	75.16	78.59		**94**	71.10	77.09	78.05
45	74.05	74.98	77.21		**95**	73.88	75.61	75.92
46	74.71	77.01	76.36		**96**	72.67	75.51	75.80
47	74.45	77.15	78.39		**97**	74.60	77.60	77.53
48	73.33	76.47	79.92		**98**	72.67	74.14	77.58
49	72.24	77.60	76.41		**99**	71.55	75.39	77.87
50	74.48	75.25	75.19		**100**	71.63	78.84	75.27
Avg of 100 Attempts	**72.49**	**76.40**	**77.45**					

Similarly, Case 5.2 to **Case 5.8**

Average CPU_Utilization for 100 Task Sets with 12 Processors

and Taskset size = n

Detailed Results are available as follow)

Detailed Results to find Average CPU_Utilization For, All Result, **Scan QR Code:** Proc 12_Case 5.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 5.2	**20**	**Click Here**
Case 5.3	**24**	**Click Here**
Case 5.4	**28**	**Click Here**
Case 5.5	**32**	**Click Here**
Case 5.6	**36**	**Click Here**
Case 5.7	**40**	**Click Here**
Case 5.8	**44**	**Click Here**

Annexure - III

Detailed Results to find Deadline Miss Ratio

Case 1.1

Average Deadlinemiss Ratio for 100 Task Sets with 16 Processors and Tasksetsize=20

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	0.0638	0.0675	0.0021		**51**	0.0790	0.0526	0.0025
2	0.0915	0.0375	0.0040		**52**	0.0917	0.0366	0.0027
3	0.0892	0.0302	0.0060		**53**	0.1084	0.0305	0.0048
4	0.0685	0.0211	0.0029		**54**	0.0948	0.0285	0.0053
5	0.0695	0.0393	0.0025		**55**	0.0846	0.0343	0.0016
6	0.0948	0.0358	0.0059		**56**	0.0854	0.0436	0.0019
7	0.0863	0.0352	0.0040		**57**	0.0861	0.0586	0.0019
8	0.0780	0.0689	0.0032		**58**	0.1098	0.0581	0.0054
9	0.0697	0.0321	0.0043		**59**	0.1041	0.0477	0.0031
10	0.1065	0.0445	0.0058		**60**	0.0972	0.0253	0.0026
11	0.1071	0.0431	0.0042		**61**	0.0934	0.0558	0.0026
12	0.0930	0.0626	0.0021		**62**	0.1072	0.0561	0.0036
13	0.0717	0.0423	0.0036		**63**	0.1098	0.0391	0.0059
14	0.0963	0.0591	0.0045		**64**	0.0839	0.0562	0.0022
15	0.0819	0.0535	0.0018		**65**	0.0686	0.0635	0.0024
16	0.0680	0.0417	0.0059		**66**	0.0691	0.0454	0.0038
17	0.0632	0.0407	0.0029		**67**	0.1043	0.0497	0.0025
18	0.1054	0.0384	0.0031		**68**	0.0915	0.0284	0.0019
19	0.0976	0.0333	0.0056		**69**	0.0730	0.0590	0.0017
20	0.0668	0.0394	0.0020		**70**	0.0749	0.0696	0.0025
21	0.1036	0.0693	0.0037		**71**	0.1024	0.0328	0.0044
22	0.0923	0.0631	0.0014		**72**	0.0706	0.0504	0.0052
23	0.0885	0.0330	0.0035		**73**	0.0636	0.0203	0.0022
24	0.0973	0.0515	0.0020		**74**	0.0878	0.0215	0.0042
25	0.0928	0.0499	0.0043		**75**	0.0661	0.0305	0.0016
26	0.0832	0.0494	0.0029		**76**	0.1061	0.0624	0.0019
27	0.0629	0.0458	0.0038		**77**	0.0897	0.0434	0.0042
28	0.0638	0.0295	0.0050		**78**	0.0606	0.0506	0.0050
29	0.0968	0.0275	0.0037		**79**	0.0612	0.0203	0.0058
30	0.1019	0.0595	0.0019		**80**	0.0632	0.0284	0.0021
31	0.1034	0.0281	0.0046		**81**	0.0803	0.0396	0.0044
32	0.0827	0.0573	0.0042		**82**	0.1060	0.0654	0.0018
33	0.0951	0.0221	0.0019		**83**	0.0714	0.0474	0.0030
34	0.0840	0.0471	0.0037		**84**	0.0804	0.0449	0.0024
35	0.1044	0.0634	0.0036		**85**	0.1040	0.0642	0.0025
36	0.1055	0.0556	0.0014		**86**	0.0939	0.0457	0.0029
37	0.0786	0.0622	0.0014		**87**	0.0849	0.0635	0.0053
38	0.0915	0.0506	0.0031		**88**	0.0924	0.0635	0.0057
39	0.0959	0.0394	0.0025		**89**	0.0825	0.0252	0.0025
40	0.0757	0.0518	0.0012		**90**	0.0822	0.0336	0.0022
41	0.0886	0.0506	0.0025		**91**	0.1032	0.0633	0.0035
42	0.0606	0.0671	0.0051		**92**	0.0890	0.0269	0.0034
43	0.0985	0.0615	0.0030		**93**	0.0914	0.0275	0.0055
44	0.0775	0.0640	0.0042		**94**	0.0956	0.0678	0.0055
45	0.0952	0.0575	0.0050		**95**	0.1074	0.0460	0.0010
46	0.0952	0.0350	0.0014		**96**	0.0979	0.0412	0.0038
47	0.1062	0.0365	0.0052		**97**	0.0949	0.0448	0.0059
48	0.0924	0.0679	0.0021		**98**	0.0959	0.0455	0.0048
49	0.0690	0.0386	0.0040		**99**	0.0629	0.0205	0.0042
50	0.0688	0.0270	0.0022		**100**	0.1034	0.0302	0.0025
Avg of 100 Attempts	**0.0873**	**0.0453**	**0.0034**					

Case 1.12

Average Deadlinemiss Ratio for 100 Task Sets with 16 Processors and Tasksetsize=64

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	0.7360	0.7098	0.7480		**51**	0.7481	0.7722	0.7476
2	0.7316	0.7360	0.7256		**52**	0.7629	0.7795	0.7444
3	0.7389	0.7190	0.7338		**53**	0.7953	0.7378	0.7350
4	0.7594	0.7437	0.7451		**54**	0.7492	0.7384	0.7575
5	0.7541	0.7393	0.7564		**55**	0.7868	0.7430	0.7585
6	0.7529	0.7343	0.7376		**56**	0.7583	0.7780	0.7351
7	0.7715	0.6954	0.7279		**57**	0.7518	0.7573	0.7435
8	0.7453	0.7384	0.7509		**58**	0.7360	0.7749	0.7536
9	0.7541	0.7658	0.7373		**59**	0.7730	0.7653	0.7170
10	0.7670	0.7190	0.7534		**60**	0.7666	0.7785	0.7171
11	0.7416	0.6364	0.7482		**61**	0.7762	0.7234	0.7309
12	0.7301	0.7143	0.7124		**62**	0.7688	0.7509	0.7380
13	0.7351	0.6806	0.7392		**63**	0.7778	0.7322	0.7132
14	0.7591	0.7024	0.7184		**64**	0.7401	0.7455	0.7126
15	0.7668	0.7166	0.7337		**65**	0.7559	0.7238	0.7522
16	0.7482	0.7458	0.7163		**66**	0.7768	0.7380	0.7263
17	0.7660	0.7625	0.7361		**67**	0.7327	0.7738	0.7351
18	0.7516	0.7300	0.7161		**68**	0.7346	0.7304	0.7419
19	0.7344	0.7385	0.7489		**69**	0.7542	0.7058	0.7593
20	0.7620	0.7260	0.7454		**70**	0.7578	0.7393	0.7426
21	0.7510	0.7692	0.7534		**71**	0.7343	0.7344	0.7230
22	0.7631	0.7009	0.7375		**72**	0.7733	0.7381	0.7147
23	0.7616	0.7318	0.7266		**73**	0.7603	0.7300	0.7290
24	0.7582	0.7284	0.7463		**74**	0.7303	0.7510	0.7320
25	0.7473	0.7406	0.7351		**75**	0.7492	0.7490	0.7311
26	0.7664	0.7426	0.7568		**76**	0.7700	0.7738	0.7460
27	0.7379	0.7299	0.7286		**77**	0.7771	0.7703	0.7212
28	0.7472	0.7644	0.7402		**78**	0.7497	0.7735	0.7245
29	0.7402	0.7278	0.7535		**79**	0.7496	0.7308	0.7219
30	0.7486	0.7799	0.7470		**80**	0.7470	0.7587	0.7263
31	0.7547	0.7546	0.7480		**81**	0.7362	0.7074	0.7225
32	0.7650	0.7373	0.7417		**82**	0.7704	0.7281	0.7482
33	0.7339	0.7504	0.7426		**83**	0.7584	0.7214	0.7316
34	0.7635	0.7593	0.7235		**84**	0.7426	0.7454	0.7156
35	0.7561	0.7472	0.7215		**85**	0.7613	0.7144	0.7393
36	0.7934	0.7581	0.7184		**86**	0.7658	0.7347	0.7031
37	0.7435	0.7455	0.7292		**87**	0.7740	0.7506	0.7326
38	0.7534	0.7491	0.7397		**88**	0.7506	0.7695	0.7310
39	0.7457	0.7401	0.7048		**89**	0.7662	0.7402	0.7312
40	0.7702	0.7685	0.7324		**90**	0.7463	0.7196	0.7169
41	0.7687	0.7640	0.7156		**91**	0.7780	0.7004	0.7166
42	0.7395	0.7531	0.7290		**92**	0.7507	0.7327	0.7451
43	0.7554	0.7471	0.7216		**93**	0.7764	0.7105	0.7342
44	0.7840	0.7425	0.7220		**94**	0.7569	0.7193	0.7106
45	0.7889	0.7569	0.7379		**95**	0.7610	0.7366	0.7558
46	0.7710	0.7687	0.7350		**96**	0.7697	0.7208	0.7469
47	0.7509	0.7602	0.7180		**97**	0.7889	0.7738	0.7361
48	0.7702	0.7688	0.7176		**98**	0.7675	0.7519	0.7384
49	0.7680	0.7719	0.7123		**99**	0.7971	0.7301	0.7353
50	0.7676	0.7321	0.7247		**100**	0.7670	0.7188	0.7318
Avg of 100 Attempts	**0.7580**	**0.7407**	**0.7335**					

Similarly, Case 1.2 to **Case 1.11**

Average Deadlinemiss Ratio for 100 Task Sets with 16 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Deadline Miss Ratio

For, All Result, **Scan QR Code:**

Proc 16_Case 1.pdf

Case No.	Taskset size (n)	Link
Case 1.2	**24**	**Click Here**
Case 1.3	**28**	**Click Here**
Case 1.4	**32**	**Click Here**
Case 1.5	**36**	**Click Here**
Case 1.6	**40**	**Click Here**
Case 1.7	**44**	**Click Here**
Case 1.8	**48**	**Click Here**
Case 1.9	**52**	**Click Here**
Case 1.10	**56**	**Click Here**
Case 1.11	**60**	**Click Here**

Detailed Results to find Average Tardiness

Case 2.1

Average Tardiness for 100 Task Sets with 16 Processors and Tasksetsize=20

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	14.89	14.45	9.42		**51**	13.14	11.24	11.86
2	15.09	13.19	12.68		**52**	15.49	13.83	10.35
3	13.17	13.85	9.81		**53**	13.99	11.20	11.45
4	12.53	14.25	12.40		**54**	14.86	14.30	10.45
5	12.40	12.99	10.47		**55**	14.69	12.14	10.54
6	13.34	11.05	12.70		**56**	15.46	12.52	11.02
7	12.47	13.02	12.73		**57**	14.34	13.77	9.83
8	15.61	13.90	11.61		**58**	15.37	14.66	9.66
9	13.93	13.09	10.18		**59**	14.72	12.22	10.72
10	12.16	14.81	11.47		**60**	15.43	12.15	12.08
11	12.48	12.43	10.53		**61**	15.53	14.80	12.71
12	12.59	12.47	10.36		**62**	14.86	12.12	11.04
13	12.35	14.87	11.50		**63**	15.11	12.18	12.49
14	12.99	13.77	12.11		**64**	13.00	13.36	11.35
15	15.27	12.69	11.10		**65**	13.79	14.92	12.10
16	13.31	14.26	10.15		**66**	13.41	12.63	10.86
17	15.85	13.55	10.83		**67**	15.46	13.71	11.16
18	12.51	14.17	12.22		**68**	12.79	11.99	9.61
19	14.28	11.01	11.64		**69**	15.60	12.75	9.32
20	12.57	13.86	11.84		**70**	13.79	14.55	11.75
21	12.60	13.55	11.64		**71**	12.18	13.76	12.30
22	13.65	11.03	11.12		**72**	12.45	11.27	11.93
23	13.25	13.84	11.35		**73**	15.69	11.91	12.11
24	14.87	11.79	9.60		**74**	14.84	11.15	12.90
25	14.45	14.93	9.97		**75**	12.22	11.42	10.86
26	12.45	12.63	11.45		**76**	13.69	13.18	11.57
27	15.17	12.61	9.64		**77**	14.28	14.97	9.69
28	14.36	13.90	11.28		**78**	12.74	11.21	12.88
29	15.27	12.16	11.01		**79**	14.25	14.26	9.16
30	13.75	14.56	12.94		**80**	14.43	12.77	9.94
31	13.23	12.93	10.22		**81**	12.76	12.99	11.98
32	16.86	13.76	12.91		**82**	12.07	13.80	9.44
33	13.38	14.36	9.37		**83**	13.13	13.96	12.97
34	15.96	12.22	12.01		**84**	15.26	14.43	12.55
35	15.61	14.96	11.94		**85**	14.51	14.40	9.45
36	13.79	14.10	12.73		**86**	14.14	12.42	12.49
37	15.56	12.06	11.01		**87**	13.91	11.69	12.44
38	12.97	12.79	11.46		**88**	15.57	13.13	9.89
39	13.01	13.72	9.18		**89**	15.61	13.28	12.81
40	12.55	12.07	11.82		**90**	15.46	11.44	10.88
41	13.57	13.84	11.69		**91**	16.80	14.08	9.35
42	13.60	13.46	10.69		**92**	14.81	11.64	9.77
43	13.10	12.81	10.83		**93**	14.09	11.35	10.76
44	12.11	13.36	11.33		**94**	14.77	12.85	9.94
45	12.32	13.37	9.69		**95**	15.77	13.63	11.08
46	13.12	13.46	9.92		**96**	13.47	12.48	9.47
47	13.26	14.18	12.82		**97**	15.89	11.13	12.82
48	14.83	11.14	10.35		**98**	12.99	11.46	9.33
49	15.62	12.57	9.41		**99**	14.19	13.02	10.07
50	14.74	14.98	12.54		**100**	12.34	12.75	10.52
Avg of 100 Attempts	**14.04**	**13.08**	**11.09**					

Case 2.12

Average Tardiness for 100 Task Sets with 16 Processors and Tasksetsize=64

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	37.16	32.49	31.60		**51**	39.89	35.61	29.50
2	37.46	32.90	30.84		**52**	36.23	34.48	29.36
3	38.70	32.65	32.56		**53**	38.81	35.54	30.79
4	39.58	33.93	30.56		**54**	36.00	32.35	30.67
5	38.20	34.10	33.58		**55**	36.67	35.73	30.44
6	37.04	35.82	33.39		**56**	40.80	32.92	32.40
7	37.13	35.70	32.59		**57**	36.21	31.84	33.29
8	36.55	34.50	33.82		**58**	39.08	35.72	30.54
9	40.14	33.54	32.56		**59**	38.08	34.94	29.08
10	36.97	33.87	29.86		**60**	39.93	35.74	29.79
11	38.44	32.48	32.05		**61**	39.41	33.97	33.45
12	37.88	35.87	32.75		**62**	36.78	33.02	30.36
13	38.77	33.53	31.52		**63**	36.88	33.97	32.87
14	39.92	34.03	31.10		**64**	38.84	32.00	33.94
15	38.66	32.22	33.99		**65**	39.91	35.06	33.84
16	36.06	35.50	29.80		**66**	40.17	34.72	33.23
17	39.89	34.82	33.71		**67**	38.99	34.32	33.05
18	38.16	33.04	31.61		**68**	36.47	35.95	29.19
19	41.00	34.71	31.87		**69**	36.14	33.85	30.83
20	36.11	34.56	33.01		**70**	36.99	33.40	30.20
21	37.82	33.02	33.02		**71**	40.60	35.66	29.86
22	39.28	34.57	31.45		**72**	38.78	33.73	32.02
23	37.58	35.23	31.76		**73**	36.02	31.03	29.78
24	36.97	32.62	30.78		**74**	37.74	35.25	33.62
25	40.32	33.52	32.06		**75**	36.07	32.62	31.19
26	36.67	32.40	30.80		**76**	39.09	33.89	33.84
27	37.61	33.07	30.09		**77**	36.40	31.73	32.62
28	37.23	32.58	31.57		**78**	40.78	31.09	29.37
29	38.31	34.75	32.98		**79**	40.44	31.02	32.81
30	38.17	34.57	32.87		**80**	40.13	32.93	32.94
31	36.42	34.37	32.56		**81**	36.10	32.74	31.32
32	40.03	35.13	30.35		**82**	37.43	35.97	33.18
33	40.66	31.05	30.20		**83**	38.18	32.25	33.54
34	39.88	32.48	29.15		**84**	39.57	32.19	29.91
35	38.44	31.52	32.48		**85**	40.45	32.06	31.67
36	37.42	34.92	33.51		**86**	39.31	34.62	32.03
37	39.94	35.68	33.35		**87**	37.95	35.59	29.73
38	38.49	35.21	29.12		**88**	37.15	32.76	29.77
39	36.25	31.65	30.94		**89**	38.10	34.67	32.22
40	40.97	35.56	33.20		**90**	40.18	32.14	32.55
41	37.38	31.90	30.49		**91**	39.52	31.58	32.64
42	40.28	33.42	32.54		**92**	40.99	35.22	29.36
43	36.95	33.50	29.43		**93**	36.46	34.12	30.67
44	39.64	31.53	30.47		**94**	39.29	33.16	31.97
45	39.05	35.09	30.51		**95**	36.70	31.03	30.85
46	40.77	35.69	33.08		**96**	39.32	35.83	30.84
47	39.73	31.21	33.17		**97**	40.72	31.73	33.07
48	37.57	33.66	29.77		**98**	39.59	34.13	29.32
49	38.77	32.65	33.75		**99**	38.09	35.13	29.99
50	40.47	34.96	29.46		**100**	36.34	34.22	32.26
Avg of 100 Attempts	**38.43**	**33.73**	**31.59**					

Similarly, Case 2.2 to **Case 2.11**

Average Tardiness for 100 Task Sets with 16 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average Tardiness For, All Result, **Scan QR Code:**  Proc 16_Case 2.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 2.2	**24**	**Click Here**
Case 2.3	**28**	**Click Here**
Case 2.4	**32**	**Click Here**
Case 2.5	**36**	**Click Here**
Case 2.6	**40**	**Click Here**
Case 2.7	**44**	**Click Here**
Case 2.8	**48**	**Click Here**
Case 2.9	**52**	**Click Here**
Case 2.10	**56**	**Click Here**
Case 2.11	**60**	**Click Here**

Detailed Results to find Average No. of Context Switch

Case 3.1

Average No. of Context Switch for 100 Task Sets with 16 Processors and Tasksetsize=20

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	117	201	190		**51**	122	202	190
2	121	206	192		**52**	121	201	188
3	114	199	193		**53**	115	200	185
4	117	203	194		**54**	114	207	186
5	121	201	188		**55**	123	208	193
6	118	206	189		**56**	114	202	185
7	118	205	185		**57**	116	206	186
8	116	205	189		**58**	121	205	187
9	114	206	191		**59**	119	204	190
10	118	204	191		**60**	117	203	189
11	115	206	191		**61**	120	202	191
12	123	202	192		**62**	118	200	187
13	122	200	190		**63**	122	202	188
14	116	206	190		**64**	122	207	187
15	121	199	192		**65**	116	207	189
16	115	200	186		**66**	118	202	187
17	122	201	191		**67**	115	202	185
18	117	205	194		**68**	115	202	192
19	121	202	187		**69**	123	202	188
20	115	200	190		**70**	118	199	186
21	115	204	194		**71**	115	205	191
22	121	200	190		**72**	114	207	186
23	122	200	187		**73**	119	201	187
24	117	208	192		**74**	118	200	192
25	123	203	191		**75**	120	203	190
26	122	206	190		**76**	119	200	193
27	119	202	188		**77**	119	206	186
28	120	204	186		**78**	114	203	190
29	119	201	190		**79**	114	204	190
30	115	206	190		**80**	121	206	190
31	119	207	190		**81**	118	208	189
32	118	208	193		**82**	117	206	186
33	114	206	185		**83**	119	199	185
34	123	202	193		**84**	119	203	187
35	118	202	189		**85**	121	205	194
36	114	207	194		**86**	119	205	189
37	114	201	193		**87**	116	201	191
38	120	207	192		**88**	120	207	190
39	121	205	188		**89**	115	201	193
40	117	207	185		**90**	122	200	188
41	123	208	186		**91**	115	202	188
42	121	205	189		**92**	118	199	190
43	123	202	193		**93**	120	208	193
44	116	205	190		**94**	121	199	187
45	123	207	191		**95**	121	208	187
46	123	208	192		**96**	121	202	187
47	118	206	187		**97**	121	205	187
48	115	203	193		**98**	120	200	190
49	121	206	188		**99**	121	202	186
50	114	201	191		**100**	115	202	194
Avg of 100 Attempts	**118.50**	**203.54**	**189.40**					

Case 3.12

Average No. of Context Switch for 100 Task Sets with 16 Processors and Tasksetsize=64

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	348	426	438		**51**	342	424	439
2	345	426	440		**52**	351	426	441
3	351	427	435		**53**	345	427	434
4	342	429	439		**54**	347	427	439
5	347	431	436		**55**	342	428	440
6	346	426	438		**56**	341	424	434
7	342	431	434		**57**	349	428	438
8	346	425	436		**58**	344	424	441
9	349	425	443		**59**	343	429	443
10	347	431	435		**60**	344	425	439
11	352	432	438		**61**	345	428	440
12	349	427	435		**62**	346	425	438
13	342	426	438		**63**	348	427	437
14	346	431	435		**64**	351	424	441
15	351	431	434		**65**	348	431	442
16	344	432	439		**66**	345	432	438
17	346	424	439		**67**	351	430	442
18	350	426	435		**68**	348	431	438
19	342	429	435		**69**	343	432	437
20	352	424	441		**70**	344	427	437
21	346	424	437		**71**	351	428	439
22	345	424	435		**72**	351	432	443
23	351	428	440		**73**	349	431	439
24	348	431	442		**74**	349	424	435
25	344	428	439		**75**	344	430	442
26	348	428	438		**76**	350	429	434
27	348	426	441		**77**	351	431	437
28	348	432	441		**78**	348	429	435
29	342	428	436		**79**	349	427	436
30	350	427	441		**80**	343	428	442
31	348	428	442		**81**	344	425	442
32	346	428	440		**82**	349	428	435
33	349	426	438		**83**	351	431	442
34	348	428	440		**84**	343	431	439
35	342	426	441		**85**	351	432	437
36	349	431	440		**86**	345	428	437
37	350	430	440		**87**	348	430	438
38	343	427	436		**88**	345	423	440
39	347	423	440		**89**	350	425	439
40	350	427	438		**90**	346	425	442
41	344	429	439		**91**	346	430	435
42	346	431	435		**92**	350	426	438
43	345	426	439		**93**	351	424	436
44	348	430	435		**94**	348	430	438
45	349	428	434		**95**	342	431	436
46	349	430	440		**96**	348	427	437
47	346	432	436		**97**	347	425	434
48	348	428	438		**98**	345	430	440
49	342	430	442		**99**	351	425	435
50	343	424	435		**100**	349	424	441
Avg of 100 Attempts	**346.90**	**427.85**	**438.22**					

Similarly, Case 3.2 to **Case 3.11**

Average No. of Context Switch for 100 Task Sets with 16 Processors and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average No. of Context Switch For, All Result, **Scan QR Code:** Proc 16_Case 3.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 3.2	**24**	**Click Here**
Case 3.3	**28**	**Click Here**
Case 3.4	**32**	**Click Here**
Case 3.5	**36**	**Click Here**
Case 3.6	**40**	**Click Here**
Case 3.7	**44**	**Click Here**
Case 3.8	**48**	**Click Here**
Case 3.9	**52**	**Click Here**
Case 3.10	**56**	**Click Here**
Case 3.11	**60**	**Click Here**

Detailed Results to find Average Schedulability

Case 4.1

Average Schedulability for 100 Task Sets with 16 Processors and Tasksetsize=20

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	88.04	89.94	96.74		**51**	89.62	90.77	92.99
2	89.57	89.10	93.14		**52**	89.14	89.15	92.91
3	87.83	88.57	93.21		**53**	88.65	89.80	93.82
4	88.45	89.54	92.18		**54**	89.02	87.57	93.08
5	89.20	90.86	93.86		**55**	89.00	90.53	93.02
6	85.58	89.64	94.37		**56**	85.52	91.32	94.70
7	86.28	90.45	94.52		**57**	89.49	89.42	93.81
8	88.46	88.91	96.15		**58**	85.86	90.89	95.20
9	86.60	90.08	96.75		**59**	85.43	90.76	94.37
10	89.69	91.48	92.60		**60**	90.00	89.64	95.22
11	85.03	89.50	93.93		**61**	87.87	87.15	92.59
12	88.88	91.60	93.23		**62**	87.29	90.60	96.17
13	85.67	90.45	93.25		**63**	86.89	91.15	96.18
14	89.60	87.61	95.73		**64**	86.95	88.30	96.42
15	89.47	88.31	92.92		**65**	85.79	88.10	96.00
16	89.51	91.75	95.73		**66**	88.12	88.09	92.61
17	87.33	90.68	94.24		**67**	86.68	87.53	95.45
18	88.48	89.75	92.93		**68**	86.89	89.12	94.33
19	86.68	89.99	94.75		**69**	86.30	87.27	96.08
20	86.93	91.20	93.61		**70**	86.08	89.61	94.31
21	85.33	90.59	94.16		**71**	87.64	87.94	92.16
22	88.62	87.53	92.57		**72**	85.97	91.94	96.06
23	87.64	88.63	94.81		**73**	85.59	88.06	92.45
24	85.95	87.67	94.44		**74**	87.68	91.54	95.05
25	85.31	90.66	94.35		**75**	87.33	88.08	96.25
26	89.83	88.76	96.34		**76**	85.77	87.26	94.88
27	87.85	88.61	93.32		**77**	87.20	91.10	94.62
28	89.17	89.99	94.54		**78**	85.70	89.76	93.78
29	88.31	90.71	94.74		**79**	85.50	90.52	95.74
30	89.83	90.72	92.07		**80**	86.09	88.14	96.01
31	87.00	87.24	94.50		**81**	86.49	88.40	93.29
32	88.58	87.08	92.79		**82**	88.26	91.17	95.85
33	85.53	89.31	95.68		**83**	89.29	91.34	94.29
34	87.13	89.89	96.16		**84**	85.76	88.92	94.58
35	89.70	89.44	93.76		**85**	86.04	90.65	93.85
36	86.59	91.22	94.63		**86**	86.41	88.01	95.34
37	88.29	90.67	92.04		**87**	89.14	91.96	95.63
38	85.22	91.89	94.14		**88**	88.63	89.53	92.78
39	85.92	88.39	95.53		**89**	86.19	90.84	92.43
40	87.36	90.09	92.47		**90**	89.61	91.50	96.01
41	87.70	89.78	93.32		**91**	85.86	90.18	92.17
42	89.36	88.85	92.71		**92**	85.79	90.17	94.07
43	85.30	91.48	94.98		**93**	88.84	89.43	92.33
44	89.11	89.14	96.75		**94**	85.64	90.91	93.31
45	87.19	88.90	95.72		**95**	85.73	88.61	96.70
46	89.88	87.49	94.93		**96**	86.11	87.45	93.99
47	87.38	91.15	92.30		**97**	87.81	89.14	93.28
48	87.33	87.40	93.55		**98**	85.82	91.82	96.63
49	89.78	89.24	92.54		**99**	86.64	89.96	93.45
50	85.68	87.61	94.14		**100**	85.76	88.04	94.50
Avg of 100 Attempts	**87.40**	**89.59**	**94.29**					

Case 4.12

Average Schedulability for 100 Task Sets with 16 Processors and Tasksetsize=64

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	67.02	75.44	75.04		**51**	67.44	74.21	79.67
2	69.27	75.70	76.05		**52**	69.83	75.99	78.62
3	71.10	75.77	75.03		**53**	68.90	75.64	76.29
4	70.15	73.29	75.64		**54**	67.57	76.06	75.58
5	69.46	73.76	75.69		**55**	70.80	74.01	76.24
6	70.07	73.24	75.93		**56**	70.32	73.36	79.38
7	71.93	76.23	78.23		**57**	69.58	74.65	79.73
8	71.02	73.35	75.52		**58**	69.78	75.15	77.26
9	69.21	76.92	77.29		**59**	71.03	73.83	79.52
10	71.85	76.40	77.89		**60**	69.76	73.46	76.55
11	71.42	76.92	77.43		**61**	69.93	75.54	77.12
12	69.16	76.97	76.28		**62**	69.29	77.95	78.96
13	68.40	73.59	75.64		**63**	67.53	76.02	77.56
14	69.72	75.65	75.71		**64**	71.62	75.97	76.50
15	70.42	73.14	75.77		**65**	68.94	76.44	76.87
16	69.21	75.08	79.17		**66**	68.22	74.74	76.53
17	70.53	73.66	77.67		**67**	67.05	77.00	79.39
18	67.15	77.39	75.82		**68**	68.09	76.87	77.10
19	67.54	74.93	79.36		**69**	69.55	75.68	79.98
20	67.65	77.47	77.32		**70**	71.99	73.30	77.80
21	69.70	75.92	78.17		**71**	69.24	74.19	76.37
22	70.11	77.31	79.66		**72**	71.98	77.40	75.97
23	71.76	75.92	76.12		**73**	70.68	76.72	75.98
24	67.92	73.20	76.73		**74**	67.83	76.56	79.71
25	69.90	77.72	78.48		**75**	68.13	75.37	75.41
26	68.18	75.11	76.57		**76**	70.30	73.07	75.61
27	67.75	76.67	76.48		**77**	70.77	75.52	79.55
28	69.67	73.19	76.58		**78**	68.86	75.47	78.66
29	68.64	73.39	76.58		**79**	68.58	75.26	78.11
30	69.62	75.87	76.28		**80**	69.53	75.82	79.41
31	67.62	75.24	79.68		**81**	71.02	74.46	78.42
32	70.88	74.83	76.98		**82**	67.15	76.85	78.96
33	68.16	75.88	75.92		**83**	70.00	74.22	75.45
34	69.52	74.31	79.63		**84**	68.46	76.03	77.42
35	67.94	77.94	78.47		**85**	68.76	75.25	76.09
36	71.87	73.08	76.99		**86**	71.22	77.16	79.36
37	69.23	77.85	79.12		**87**	68.26	76.45	75.10
38	68.12	77.36	78.67		**88**	69.35	74.20	78.68
39	70.94	73.86	79.56		**89**	67.27	74.98	79.85
40	71.36	74.08	79.29		**90**	67.78	77.70	76.81
41	67.52	76.68	75.20		**91**	69.47	76.42	78.63
42	71.55	76.76	78.47		**92**	70.78	75.04	77.55
43	67.03	76.71	75.99		**93**	70.66	75.68	77.82
44	68.44	77.91	77.67		**94**	71.21	77.40	78.39
45	71.14	77.52	79.93		**95**	67.48	76.16	78.06
46	71.54	77.94	75.13		**96**	70.16	76.58	77.26
47	68.36	76.09	76.58		**97**	67.01	75.13	78.16
48	69.70	75.89	75.76		**98**	68.45	76.89	76.13
49	71.81	74.68	79.92		**99**	71.59	74.49	79.23
50	71.62	75.20	78.62		**100**	68.92	75.80	79.71
Avg of 100 Attempts	**69.48**	**75.57**	**77.50**					

Similarly, Case 4.2 to **Case 4.11**

Average Schedulability for 100 Task Sets with 16 Processors

and Taskset size = n

(Detailed Results are available as follow)

Detailed Results to find Average Schedulability For, All Result, **Scan QR Code:**  Proc 16_Case 4.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 4.2	**24**	**Click Here**
Case 4.3	**28**	**Click Here**
Case 4.4	**32**	**Click Here**
Case 4.5	**36**	**Click Here**
Case 4.6	**40**	**Click Here**
Case 4.7	**44**	**Click Here**
Case 4.8	**48**	**Click Here**
Case 4.9	**52**	**Click Here**
Case 4.10	**56**	**Click Here**
Case 4.11	**60**	**Click Here**

Detailed Results to find Average CPU_Utilization

Case 5.1

Average CPU_Utilization for 100 Task Sets with 16 Processors and Tasksetsize=20

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	49.89	57.42	59.09		**51**	52.90	53.63	59.02
2	50.31	56.67	57.07		**52**	49.29	56.17	56.38
3	52.83	56.30	58.54		**53**	53.93	57.57	56.88
4	53.04	57.38	58.53		**54**	49.93	56.62	58.48
5	51.91	55.17	56.60		**55**	51.95	55.16	56.41
6	49.20	57.24	57.17		**56**	49.23	53.30	58.55
7	52.06	57.35	56.71		**57**	51.27	55.66	60.96
8	53.68	56.18	59.85		**58**	49.98	57.99	60.67
9	49.42	57.26	60.28		**59**	51.49	56.16	59.10
10	50.01	53.97	59.16		**60**	52.43	53.33	56.36
11	52.25	54.67	56.65		**61**	51.01	57.46	58.37
12	52.19	53.27	56.91		**62**	52.67	53.51	58.23
13	51.61	57.70	60.68		**63**	53.98	56.05	58.72
14	52.25	56.69	59.19		**64**	49.90	53.02	56.39
15	53.12	56.50	58.13		**65**	50.95	55.21	60.97
16	49.21	57.21	57.61		**66**	53.48	54.01	57.65
17	49.98	55.25	57.23		**67**	50.22	57.51	59.54
18	51.22	53.18	60.54		**68**	51.16	54.53	56.43
19	53.81	57.28	59.33		**69**	50.84	54.31	56.52
20	49.87	53.04	60.11		**70**	51.65	57.18	60.54
21	52.54	54.59	57.74		**71**	49.68	57.26	56.76
22	49.73	54.11	56.79		**72**	51.96	53.15	60.18
23	52.91	56.05	59.96		**73**	51.70	53.32	56.91
24	53.66	56.14	59.37		**74**	52.94	53.41	58.61
25	50.47	57.57	59.78		**75**	49.32	56.16	60.01
26	53.42	53.11	58.46		**76**	53.72	57.72	57.56
27	51.13	54.48	59.79		**77**	51.21	56.96	60.08
28	53.12	53.89	60.09		**78**	52.87	56.86	59.64
29	50.37	54.61	58.05		**79**	52.67	53.55	59.73
30	53.35	57.81	57.70		**80**	50.02	53.30	59.85
31	53.36	53.91	58.24		**81**	51.52	57.24	60.51
32	51.45	56.40	59.89		**82**	52.29	53.96	60.47
33	52.32	57.17	60.18		**83**	52.06	55.45	56.05
34	51.27	54.18	58.67		**84**	50.78	55.31	60.01
35	50.38	53.80	59.47		**85**	52.31	55.56	56.99
36	49.91	56.31	56.56		**86**	51.29	53.47	58.45
37	52.49	54.91	59.64		**87**	51.30	56.98	58.55
38	53.24	56.80	59.45		**88**	51.07	55.81	58.71
39	53.75	54.54	60.20		**89**	50.79	53.66	57.61
40	50.31	54.30	60.88		**90**	50.31	54.67	56.00
41	50.73	54.30	59.11		**91**	51.47	57.72	58.77
42	52.66	56.18	60.82		**92**	53.03	57.49	59.02
43	53.40	54.56	60.55		**93**	51.60	55.64	56.65
44	50.05	54.69	58.23		**94**	52.54	56.09	57.62
45	50.79	56.78	58.30		**95**	51.45	56.13	57.99
46	52.50	57.53	58.63		**96**	51.42	54.33	57.27
47	52.09	54.91	58.23		**97**	51.68	55.48	60.55
48	52.37	55.56	58.91		**98**	49.21	53.91	60.31
49	51.26	56.93	59.42		**99**	52.62	57.37	60.42
50	53.40	56.36	58.91		**100**	52.72	57.08	60.03
Avg of 100 Attempts	**51.62**	**55.56**	**58.69**					

Case 5.12

Average CPU_Utilization for 100 Task Sets with 16 Processors and Tasksetsize=64

Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS		Taskset No.	P-FP-APA	G-JLFP-APA	PrAMS
1	67.08	75.40	77.40		**51**	71.38	74.78	75.28
2	68.56	74.97	76.14		**52**	69.62	73.57	75.97
3	70.69	75.46	75.74		**53**	68.81	72.66	77.65
4	68.85	73.78	74.68		**54**	69.73	74.87	77.93
5	71.10	73.44	76.77		**55**	70.74	73.87	77.96
6	70.72	74.58	76.38		**56**	68.71	75.60	76.72
7	69.47	76.30	73.94		**57**	71.51	76.51	74.40
8	70.95	74.61	74.94		**58**	67.33	73.38	74.66
9	70.98	76.29	75.81		**59**	71.25	73.32	74.53
10	71.10	76.39	75.85		**60**	68.45	72.34	77.79
11	71.77	72.55	77.46		**61**	67.62	76.71	73.54
12	68.14	72.37	76.82		**62**	68.74	76.79	74.25
13	70.89	74.62	73.76		**63**	71.20	75.77	75.46
14	70.48	75.24	76.12		**64**	71.05	74.39	73.83
15	67.04	75.07	75.29		**65**	69.67	76.87	76.03
16	67.72	73.55	76.28		**66**	70.18	74.65	75.33
17	69.27	72.72	73.16		**67**	70.61	76.01	73.28
18	67.08	75.06	76.56		**68**	69.59	72.97	75.26
19	69.48	74.98	73.58		**69**	71.91	72.47	76.17
20	67.57	75.82	74.79		**70**	67.64	72.12	75.76
21	67.80	75.44	76.93		**71**	70.12	75.80	73.90
22	70.63	76.55	77.63		**72**	70.93	74.69	75.38
23	67.12	74.20	74.87		**73**	71.64	72.76	76.74
24	71.18	74.49	75.74		**74**	70.76	76.06	73.05
25	71.64	73.34	76.75		**75**	69.34	73.31	76.14
26	71.45	74.47	77.03		**76**	68.50	75.12	73.29
27	67.36	76.30	77.53		**77**	70.66	74.91	77.01
28	69.88	76.75	76.56		**78**	71.91	74.56	73.14
29	69.85	76.84	76.26		**79**	70.24	72.59	73.23
30	70.24	74.93	75.93		**80**	69.09	75.34	77.75
31	68.36	72.70	74.84		**81**	71.26	74.37	76.89
32	68.58	72.01	76.08		**82**	67.50	72.62	77.58
33	68.34	74.89	76.64		**83**	67.59	73.02	76.35
34	70.88	74.15	75.84		**84**	68.82	75.69	77.76
35	68.38	73.64	77.72		**85**	70.40	74.55	75.15
36	68.50	74.80	76.36		**86**	70.40	73.54	77.29
37	67.55	72.69	76.26		**87**	69.76	72.53	77.53
38	70.72	72.19	75.76		**88**	69.54	75.12	77.84
39	69.83	72.78	76.50		**89**	68.43	76.61	77.04
40	71.90	73.26	74.34		**90**	71.29	75.72	74.24
41	71.99	74.24	75.28		**91**	67.29	73.87	73.26
42	67.35	75.25	74.91		**92**	71.07	76.27	73.86
43	71.29	73.98	76.82		**93**	67.89	76.76	75.06
44	71.50	74.43	76.04		**94**	67.18	72.15	74.02
45	69.98	72.13	73.19		**95**	71.48	72.92	74.63
46	67.24	73.30	77.86		**96**	71.75	72.73	77.44
47	71.17	75.82	74.75		**97**	68.30	73.82	76.22
48	69.79	74.01	74.52		**98**	70.24	75.39	76.30
49	69.59	73.54	77.61		**99**	70.23	76.61	73.22
50	69.06	72.80	76.47		**100**	70.23	74.78	77.31
Avg of 100 Attempts	**69.68**	**74.43**	**75.75**					

Similarly, Case 5.2 to **Case 5.11**

Average CPU_Utilization for 100 Task Sets with 16 Processors

and Taskset size = n

Detailed Results are available as follow)

Detailed Results to find Average CPU_Utilization For, All Result, **Scan QR Code:** Proc 16_Case 5.pdf		
Case No.	**Taskset size (n)**	**Link**
Case 5.2	**24**	**Click Here**
Case 5.3	**28**	**Click Here**
Case 5.4	**32**	**Click Here**
Case 5.5	**36**	**Click Here**
Case 5.6	**40**	**Click Here**
Case 5.7	**44**	**Click Here**
Case 5.8	**48**	**Click Here**
Case 5.9	**52**	**Click Here**
Case 5.10	**56**	**Click Here**
Case 5.11	**60**	**Click Here**