Resource Optimization by Scheduling

A dissertation submitted towards the partial fulfillment of requirement

for the award of degree of

Master of Technology

in

VLSI Design & CAD

Submitted by:

Pankaj Udvanshi

Roll No. : 601161027

Under the guidance of:

Mr. Ajay Kakkar

Assistant Professor, ECED

Thapar University, Patiala

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled, **"Resource Optimization by Scheduling"** in partial fulfilment of the requirement for the award of degree of Master of Technology in VLSI Design submitted in the department of Electronics and Communication Engineering of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of **Mr. Ajay Kakkar, Assistant Professor, ECED** and refers other researcher's work which are duly listed in the reference section.

The matter presented in this dissertation has not been submitted in any other University/Institute for the award of degree.

Date: 15/07/2013

(PANKAJ UDVANSHI)

Roll No: 601161027

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

15/7/13

(Mr. Ajay Kakkar)

Assistant Professor

ECED, Thapar University

Countersigned by:

Head

ECED, Thapar University

Patiala-147004

Dean of Academic Affairs

Thapar University

Patiala- 147004

i

Acknowledgement

First of all, I would like to express my gratitude to **Mr. Ajay Kakkar, Assistant Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala for his patient guidance and support throughout my work. I am truly very fortunate to have the opportunity to work with him. I found his guidance to be extremely valuable.

I am also thankful to our **Head of the Department, Dr. Rajesh Khanna,** Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department. I would also like to thank my friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would like to thank my parents for their unconditional support and encouragement.

Pankaj Udvanshi
Roll No. 601161027

Abstract

Scheduling refers to a set of policies and mechanisms supported by operating system that controls the order in which the work to be done is completed. Scheduling is the best method to improve the performance of the network by re-arranging the system parameters. The aim of scheduling is to keep the CPU busy as much as possible by executing a (user) process until it must wait for an event, and then switch to another process. Various scheduling algorithm which are suitable only in linear system have been studied. Shortest distance between source and destination in a network has also been calculated. Results have been observed using "TORA". Two nodes failure condition have also been considered: a) Single node failure and b) Multiple nodes. Finally, observing from the simulation results has been drawn.

Table of Contents

List of Figures

Page No.

List of Tables

CHAPTER

1

INTRODUCTION

1.1 Introduction

Scheduling refers to a set of policies and mechanisms supported by operating system which controls the order of task to be completed. The main objective of scheduling is to increase CPU utilization and provide higher throughput. Throughput is the amount of work accomplished in a given time interval [13]. CPU scheduling is the basis of operating system which supports multiprogramming concepts. It also improves the overall efficiency of the computer system by completing more work in less time [10]. From all the resources in a computer system that are scheduled before use, the CPU is by far the most important. The aim is to utilization resources in an optimized way and is possible by executing a process until it must wait for an event, and then switch to another process.

1.2 Need of scheduling

Scheduling is done on the basis of two things new process and current process. In multiprogramming system; when there are more than one variable process, operating system must decide which one is activated [24]. To activate the operating system specific time is known scheduling. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm. Initially there was no need for scheduling, since the users of computers are lined up in front of the computer room or gave their job to an operator. It means they have only single task at a one time, after that the batch processing is come [47]. In the batch processing, one can handle multiple tasks simultaneously but one drawback of batch processing is that they execute the task which first comes. So, it is also known as first come first serve processing. To complete the entire task in less time, one needs the scheduling. The scheduling is basically used to cure the deadlock which occurs in the system [30].

1.3 Types of Scheduling:

To improve the system performance like response time, throughput and processing efficiency one uses scheduling. In many system scheduling activity can be categorized into three ways [41] that is Long term scheduling, Medium term scheduling and Short term scheduling.

1.3.1 Long-Term Scheduling:

This type of scheduling is performing when the new process is design [41]. If the number of processes in the ready queue is very high, mean overhead on operating system will be observed. Long-term scheduler determines which program is admitted into the system for the process. When a program is admitted, then it becomes process and it is added to the queue for short-term scheduling. It may also be very important that the long term scheduler may take a careful selection of process. Process should be a combination of CPU and input/output bound types. Generally, most processes can be put into any of the two categories: CPU bound or input/output bound. If all processes are input/output bound, the ready queue will always be empty and the short term scheduler will have nothing to do. If all processes are CPU bound, no process will be waiting for input/output operation and again the system will be unbalanced [10]. The long term scheduler provides good performance by selecting a combination of CPU bound and input/output bound processes. In some system, a newly created process begins in a swapped-out condition in which case it is added to a queue for the medium term scheduler.

1.3.2 Medium -Term Scheduling:

This type of scheduling is the part of swapping function which means transfers the job in ready queue to main memory because there is no sufficient memory in the medium term scheduling [41]. It means if a system is running multiple programs and has very large memory, there is no need of medium term scheduling. Medium term scheduling may decide to swap out to process which are not activated for some time. Saving of the suspended process is said to be swapped out or rolled out. The process is swapped in and swapped out by the medium term scheduler. The medium term scheduler has nothing to do with the suspended processes but the moment the suspending condition is fulfilled, the medium term scheduler get activated to allocate the memory and swapped in the process and make it ready for execution. The medium term scheduling is also helpful for reducing

the degree of multiprogramming, when long term scheduler is absent or minimal. Medium term scheduling mainly based on the requirement of resources. Most of the processes require some input/output operation. In such case, it may become suspended for input/output operation after running a while. It is beneficial to remove suspended processes from main memory to hard disk to make room for other processes. At some later time these processes can be reloaded into memory and continued where from it was left earlier.

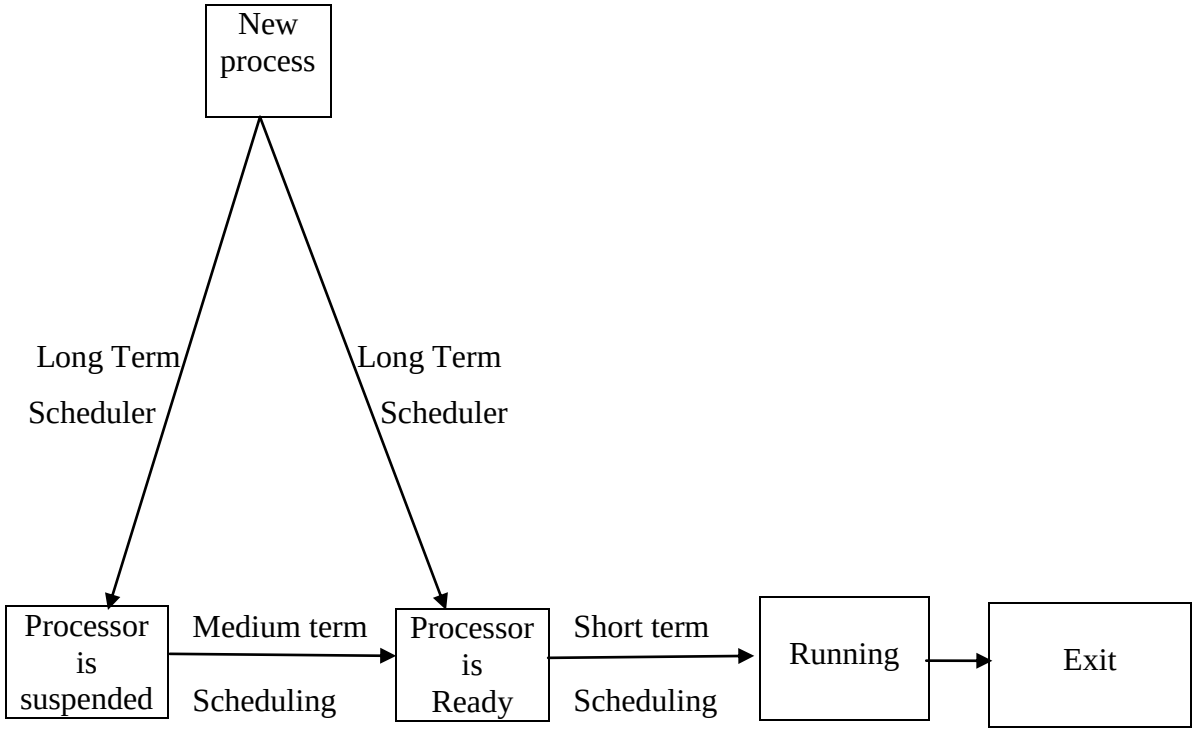


Fig.1 Type of Scheduling [41]

1.3.3 Short –Term Scheduling:

This type of scheduling is also called dispatch and is very fast. Short-term scheduling is invoked whenever an event is occurs, that may lead to the interruption of current running process [41]. The short term scheduler is also called CPU scheduler. Whenever the CPU becomes idle the operating system must select one of the processes in the ready queue to execute.

1.4 Types of CPU schedulers:

The CPU scheduler selects a process from the ready queue and lets it run on the CPU. There are basically two type of scheduler in CPU scheduler; (a) Non Pre-emptive and (b) Pre-emptive scheduler.

1.4.1 Non-Pre-emptive:

Non pre-emptive scheduler executes only when the process is terminated. It means if there is another task of higher priority, the current executed process not suspended it is executed until this process is completed [18]. Non pre-emptive scheduler is easy to implement but unsuitable for time-sharing systems.

1.4.2 Pre-emptive scheduler:

Pre-emptive scheduler executes at times when process is created. It means if there is another task of higher priority, the current executed process suspended and it executed the newly higher priority process [18]. After the completed the higher priority process pre-emptive scheduler again executes the suspended process. On the pre-emptive scheduler more overhead as compare to Non pre-emptive. Pre-emptive scheduler also uses memory for the current suspended process.

1.5 Scheduling Criteria:

Different CPU-scheduling algorithms have different properties and may favour one class of processes over another. In selecting the algorithm to use in a particular situation, one must consider the properties of the various algorithms. The following criteria were suggested for comparing CPU-scheduling algorithms.

1.5.1 CPU Utilization:

The key idea is that if the CPU is busy all the time, the utilization factor [10]of all the components of the system will be also high. CPU utilization may range from 0 to 100%.

1.5.2 Throughput:

It refers to the amount of work completed in a unit of time. One way to measure throughput is by means of the number of processes that are completed in a unit of time [40]. The higher the number of processes, the more work apparently being done by the

system. But this approach is not very useful for comparison because this is dependent on the characteristics and resource requirement of the process being executed.

1.5.3 Turn-around Time:

Turnaround time is defined as the interval from the time of submission of a process to the time of its completion [44]. The important criterion is how long it takes to execute that process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and input/ output operations.

1.5.4 Waiting Time:

In a multiprogramming operating system several jobs reside at a time in memory. CPU executes only one job at a time. The rest of jobs wait for the CPU [41]. The waiting time may be expressed as turnaround time less the actual processing time and is given as:
Waiting time = turnaround time - processing time.

But the scheduling algorithm affects the amount of time that a process spends waiting in a ready queue. The CPU-scheduling algorithm does not affect the amount of time during which a process executes [44]. So rather than looking at turnaround time; waiting time is the sum of the periods spent waiting in the ready queue.

1.5.5 Response Time:

It is most frequently considered in time sharing and real time operating systems. However, its characteristics differ in the two ways. In time sharing system, it may be defined as interval from the time the last character of a command line of a program or transaction is entered to the time the last result appears on the terminal. In real time system it may be defined as interval from the time an internal or external event is signalled to the time the first instruction of the respective service routine is executed [28]. One of the problems in designing schedulers and selecting a set of its performance criteria is that they often conflict with each other. For example, the fastest response time in time sharing and real time system may result in low CPU utilization. Throughput and CPU utilization may be increased by executing large number of processes, but then response time may suffer. So, the design of a scheduler usually requires a balance of all the different requirements and constraints.

1.6 Objectives of the Work:

The objective of the work is given as:

a) To study the various scheduling algorithms used for real time tasks.

b) To design an algorithms used to handle linear and non linear tasks in real time.

c) Comparison of proposed algorithm with existing algorithms.

1.7 Organization of the report:

The report contains five chapters; first chapter contain the introduction, needs, type and criteria of scheduling. Work done by various researchers, considering the research paper in static and dynamic scheduling real time tasks have been introduced in second chapter. The third chapter include the various scheduling algorithms on the basis of all scheduling algorithms and the performance of proposed system has been observed. Comparison of proposed model has also been done with various scheduling algorithm. Simulation result for the scheduling algorithm using the "TORA" tool has been shown in the fourth chapter. Finally, the fifth chapter include conclusion and future scope of this work.

CHAPTER 2

LITERATURE SURVEY

This chapter involves the work done by the various researchers in the field of scheduling of real time tasks to handle embedded system.

Jane W.S. Liu et al. [1] proposed an algorithm for scheduling imprecise computations. According to them, imprecise computation techniques provide scheduling flexibility by trading off result quality to meet computation deadlines. They also reviewed, imprecise computation scheduling problems such as workload models and algorithms. Imprecise computation techniques minimize the bad effect of timing fault. They also suggested to minimization the number of discarded optional tasks and minimization of total error.

Christopher chase et al. [2] proposed a real time scheduling policies for flexible manufacturing system. They consider a model that was proposed by Perkins and Kumar [43] for real time control of flexible manufacturing system. Machine processed a finite number of part types at specifed rates but only one part was processed at a time in that model. To use multiple part, machine uses the feedback rule time to time from one part type to another.

John A. Stankovic [3] proposed many faces of multi-level real-time scheduling. The complex real time system usually have two or more scheduling algorithms due to the system size, the vastly different requirements of various sets of tasks, and the different metrics used for different functions or subsystems. They focused on the issue of multilevel scheduling. Multi-level scheduling arises for various important reasons. They discussed multi-level scheduling examples from a process/thread model, local/distributed scheduling, manufacturing and multimedia.

Y. X. Dai et al. [4] proposed an alternate approach for real-time scheduling. They proposed self scheduling based on the philosophy of competition, which improved the scheduling efficiency when jobs were increasing in automatic manufacturing system. Implementation problems and support technology, such as parallel processing and dynamic data exchange were also discussed. They addressed the issue of information systems development, production control and performance measures in an integrated manner.

P. Bizzarri et al. [5] proposed a scheduling algorithm for a periodic group of tasks in distributed real-time systems. In their analysed work the problem of scheduling periodic groups of tasks in distributed systems was also proposed. Two contributions; a) distributed scheduling algorithm and b) analysis of the behaviour of the proposed scheduling algorithm by applying a holistic approach were also considered. The sensitivity of the response time of the algorithm to the parameters involved was also studied.

D. Montana et al. [6] proposed a genetic algorithm for complex, real-time scheduling applications. They considered applications that require real-time scheduling of large scale problems in complex domains present a number of difficulties for search and optimization techniques. These difficulties include search space whose size grows exponentially with the size of the problem. Tasks were constantly changing due to a changing environment and users. There was need to trade off between a variety of different criteria measuring the relative fitness interaction and of a particular schedule.

M. Abe et al. [7] proposed an artificial neural network optimized by a genetic algorithm for real time flow-shop scheduling. They developed a former genetic artificial neural network scheduling method to a practical real-time scheduling system for flow-shop process problems. Adaptability of artificial neural network to change of environments was proved to be effective for real-time rescheduling. They investigated the effect of improvements of the genetic artificial neural network scheduling system on the efficiency of rescheduling when new jobs were append in a chemical process with some buffer tanks. In results, the former genetic artificial neural network scheduling method could be developed to a practical real-time scheduling system for process problems.

Tei-Wei Kuo et al. [8] proposed a class of rate-based real-time scheduling algorithms. They investigated a class of rate-based real-time scheduling algorithms based on the idea of General Processor Sharing (GPS). They extended the GPS frame work for periodic and sporadic process scheduling and showed the optimality of GPS-based scheduling. They also proposed the Earliest-Completion-Time GPS (EGPS) scheduling algorithm to simulate the GPS algorithm with much lower run-time overheads. The schedulability of each process was enforced by a guaranteed CPU service rate, independent of the demands of other processes. They also provide a theoretical foundation to assign proper CPU service rates to processes to satisfy their individual stringent response time requirements. They also proposed a GPS-based scheduling mechanism for jitter control. Finally, the performance of the proposed algorithms was studied using a generic avionics platform. Simulation experiments on jitter control and mixed soft and hard real-time process scheduling were also done.

Kwei-Jay Lin et al. [9] proposed designing multimedia applications used for real-time systems with symmetric multi-processors architecture. Existing real-time scheduling algorithms designed for single-processor systems such as Rate Monotonic (RM) and Earliest Deadline First algorithm (EDF) showed very poor performance on multiprocessor systems. They analyzed different ways used to design real-time applications on symmetric multi-processors machines. Many simulator systems were adopted multiprocessor architecture to handle the large number of system events and the large volume of computation data that were typical in this type of systems. With an appropriate system design, the SMP (Symmetric Multi-Processors) architecture allows a system to be more scalable and more reliable. SMP-based systems were more scalable since the number of processors could be increased with the growing needs of the application. SMP-based systems were more reliable since the unavailability of one processor could not affect the execution of the other processors.

Deji Chen et al. [10] proposed an utilization bound revisited. They defined the utilization bound as a function of the information about the task set. By making use of more than just the number of tasks, better utilization bound over the Liu and Layland [10] bound can be achieved. There was still no known polynomial algorithm to calculate the exact utilization bound given the task periods. They proposed the exact bound for some special cases.

They also suggested a simpler proof for the harmonic chain bound. The algorithms were tested and compared with randomly generated task periods.

Sanjoy K. Baruah et al. [11] proposed a rate monotonic scheduling method for uniform multiprocessors. Rate-Monotonic scheduling was the one of the most popular algorithm for the scheduling system of periodic real-time task. They obtained the first nontrivial feasibility test for a static-priority scheduling algorithm that adopts a global approach to task allocation upon uniform multiprocessors. They also studied the behaviour of algorithm RM-one such previously defined static priority global scheduling algorithm-upon uniform multiprocessor platforms. They obtained simple sufficient conditions for determining whether any given periodic task system could be successfully scheduled by RM Algorithm upon a given uniform multiprocessor platform.

C.M. Krishna et al. [12] proposed a voltage-clock-scaling adaptive scheduling technique for low power in hard real-time systems. They described simple algorithms for voltage scaling in real-time systems. These algorithms exploit the fact that power consumption tends to drop quadratically with voltage, while circuit delays increase only linearly. Proposed algorithm was compared to work upon offline and online component. The offline component assumes that the tasks run to their worst-case execution times and computes the voltage settings to minimize energy consumption. The online component starts with the offline voltage settings as a base and then reclaims any time resources that were released by tasks which finish ahead of their predicted worst-case execution times, thus making for a further round of energy savings. Results indicate that significant energy savings were possible, while guaranteeing that all tasks could be continued to meet their deadlines.

Kuan-Yu Chen et al. [13] proposed a multiprocessor real-time process scheduling method. They proposed a method for multiprocessor real-time scheduling algorithm applicable for both computer science and operation research. Method proposed by them was general enough to solve different scheduling problems such as wafer lot dispatching and scheduling for behaviours of a robot soccer player. In multimedia systems like, video-on-demand systems required a good scheduling method to improve their services because of their real-time requirements.

Peng Li et al. [14] suggested a fast, best-effort real-time scheduling algorithm. They presented two fast, best-effort real-time scheduling algorithms called Modified Dependent Activity Scheduling Algorithm (MDASA) and Modified Locke's Best Effort Scheduling Algorithm (MLBESA). Furthermore, the task response times under MDASA and MLBESA were found to be very close to the values under their counterpart scheduling algorithms. While MDASA performs better than MLBESA and had a better worst-case complexity, MLBESA guarantees the optimal schedule during under load situations.

Dan Ma et al. [15] proposed dynamic scheduling algorithm for parallel real-time jobs in heterogeneous system. The aim was to building a real-time scheduling model that was applicable to dynamically scheduling multiple. In the model, an assigned processor which was called centre scheduler was responsible for dynamically scheduling the real-time jobs when they arrive. Based on the proposed real-time scheduling model they presented a new dynamic scheduling algorithm. Some simulation experiments were made and the results of these experiments show the proposed scheduling algorithm was a valid dynamic scheduling algorithm with better performance.

Pengliu Tan et al. [16] proposed a hybrid scheduling scheme for hard, soft and non-real-time tasks. They proposed a hybrid scheduling scheme for hard, soft and non-real-time tasks. The scheme created a constant utilization server for each real-time scheduling policy and a total bandwidth server for all the non-real-time tasks with time sharing scheduling policy. Each server had a scheduler, called server scheduler, which was responsible for executing the ready-to-run tasks with the same scheduling policy as the server. All the servers were scheduled by the bottom operating system scheduler according to EDF algorithm. The schedulability test was also presented by them. The hybrid scheme was used to implement different real time systems with different goals and provides the real-time services with different task modes by adjusting every constant utilization server's size. This scheme also simplified the schedulability analysis and validates the schedulability of the tasks with one scheduling policy scheduling independent of the tasks with other scheduling policies. The experimental results showed that the hybrid scheme was an efficient scheduling scheme.

Yang-ping Chen et al. [17] presented a novel task scheduling algorithm for real-time multiprocessor systems. A novel task scheduling algorithm for real-time multiprocessor

systems was also given, which took task's height and particle's position as the task's priority values, and applies the list scheduling strategy to generate the feasible solutions. Genetic algorithm produces encouraging results in terms of quality of solution and time complexity.

Octav Chipara et al. [18] introduced a real-time query scheduling for wireless sensor networks. A novel approach to conflict-free transmission scheduling for real-time queries in wireless sensor networks was also proposed. An inherent trade-off between prioritization and throughput in conflict-free query scheduling was observed. Three types of new real-time scheduling algorithms were presented. a) The non preemptive query scheduling algorithm achieved high throughput while introducing priority inversions. b) The preemptive query scheduling algorithm eliminated priority inversion at the cost of reduced throughput. c) The slack stealing query scheduling algorithm combines the benefits of preemptive and nonpreemptive scheduling by improving the throughput while meeting query deadlines. Finally, it was observed that there exists a trade off between throughput and prioritization under conflict-free query scheduling. Design and schedulability analysis of three new real-time scheduling algorithms for prioritized transmission scheduling was also presented.

Jian-Jia Chen Chuan-Yue Yang et al. [19] proposed approximation algorithms for multiprocessor energy-efficient scheduling of periodic real-time tasks with uncertain task execution time. A task partition was derived which minimized the expected energy consumption for completing all the given tasks in time. They proposed an efficient approximation algorithm and a Polynomial-Time Approximation Scheme (PTAS) to provide worst-case guarantees for the strongly non pre-emptive-hard problem. Experimental results showed that the algorithms could be effectively minimizing the expected energy consumption.

Robert Glaubius et al. [20] presented a scheduling design and verification for open soft real-time systems. Three main contributions to the state of the art were made in scheduling open soft real-time systems: a) it defined a novel representation of the scheduling state space that was more compact and more expressive than the model defined in previous work; b) it exploits regular structure of that representation to allow efficient verification of properties involving both discrete and continuous system state

variables under specific scheduling policies; and c) it removes the unnecessary use of a time horizon in previous approach, thus allowing the more precise specification and enforcement of a wider range of scheduling policies for open soft real-time systems. Techniques for efficiently creating scheduling polices based on a given utilization specification were also presented.

Lars Lundberg et al. [21] proposed a slack-based global multiprocessor scheduling of a periodic tasks in parallel embedded real-time systems. They proved that if the load on the multiprocessor stays below (3-root of 5)/2 ≈38.197%, the server accepts an incoming a periodic task. This was better than the current state-of the-art algorithm where the priorities of light tasks are based on deadlines. The traditional priority assignment strategy for light tasks from shortest deadline was replaced with least slack first. It turns out that the worst-case behaviour of the slack based approach was better when multi processors.

Yi-Hsiung Chao et al. [22] presented on a schedulability issues for EDZL scheduling on real-time multiprocessor systems. They disproved this conjecture and show that the utilization bound of EDZL was no greater than m $(1 - 1/e) \approx 0.6321m$, where $e \approx 2.718$ was the Euler's number. EDZL (Earliest Deadline first until Zero Laxity) was an efficient and practical scheduling algorithm on multiprocessor systems. It had a comparable number of context switches to EDF (Earliest Deadline First) and its schedulable utilization seems to be higher than that of EDF. Previously, there was a conjecture that the utilization bound of EDZL is $3m/4 = 0.75m$ for m processors.

Zhu Xiang bin et al. [23] proposed an embedded computing on a dynamic window-constrained scheduling algorithm for multiprocessor real-time systems. An improved heuristic algorithm, which had a new heuristic function for window-constrained real time tasks were also proposed. The improved algorithm considers not only the deadlines and the resource requirements of a task, but also the processing time of the task. The most important metric for real-time scheduling algorithms was scheduling success ratio. To evaluate the effectiveness of the improved algorithm had done extensive simulation studied. The simulation results showed that the scheduling success ratio of the improved heuristic algorithm was superior to that of myopic algorithm.

Ming Xiong et al. [24] proposed a deferrable scheduling for maintaining real-time data freshness algorithms, analysis and results. Deferrable Scheduling algorithm for Fixed-Priority (DS-FP) transactions, approach for minimizing update workload while maintaining the temporal validity of real-time data was also proposed. In contrast to prior work on maintaining data freshness periodically, update transactions followed a periodic task model in the deferrable scheduling algorithm. The deferrable scheduling algorithm exploits the semantics of temporal validity constraint of real-time data by judiciously deferring the sampling times of updated transaction jobs as late as possible. They also proposed a theoretical estimation of the processor utilization of DS-FP, which was verified in experimental study. It was possible for the same concept to be used in the scheduling of update transactions with dynamic priority.

Euiseong Seo et al. [25] proposed an energy efficient scheduling of real-time tasks on multicore processors. They tackled the problem of reducing power consumption in a periodic real-time system using dynamic voltage scaling on a multicore processor. The processor was assumed to have the limitation that all cores must run at the same performance level. To reduce the dynamic power consumption of such a system they suggested an algorithm based on dynamic repartitioning and dynamic core scaling. Dynamic repartitioning tries to maintain balance in the performance demands by migrating tasks between cores during execution accompanying with deadline guarantee.

Farooq Muhammad et al. [26] proposed a slack-conserving based scheduling of periodic real-time tasks. They proposed an algorithm called Real Urgency First scheduling (RUF) algorithm. According to this algorithm, critical tasks were always guaranteed to meet their real time constraints while improving the execution of non critical tasks. Real time scheduling algorithms provide a mean to schedule tasks on processor such that real time constraints were guaranteed. Fixed priority real time scheduling algorithms had low runtime complexity and their behaviour was categorized a prior but they do not support dynamic systems very well. A transient overload in dynamic priority scheduling algorithms may cause a critical task to fail but they were capable of supporting dynamic systems. Hybrid scheduling algorithms behaviour could be categorized a prior and they support dynamic systems as well but they may unnecessarily cause non critical tasks to miss their deadlines even when all critical tasks are schedulable.

Aihan Yin et al. [27] suggested a dynamic programmable scheduling mechanism of gigabit-ethernet passive optical network. They also proposed a dynamic programmable scheduling mechanism that was effectively to solve the problems of the polling mechanism. At the same time, the dynamic programmable scheduling mechanism could provides a greater degree of flexibility that enables service providers to decide and achieve their own bandwidth allocation logic after the network was fixed. In addition, to a large extent, the proposed mechanism could be promoting interoperability among different gigabit-ethernet passive optical network systems.

Enrico Bini et al. [28] considered a response-time bound in fixed-priority scheduling with arbitrary deadlines. They identified three desirable properties of estimates of the exact response times, continuity with respect to system parameters, efficient computability and approximability. They also derived a continuous upper bound on response time for task systems that were scheduled upon a preemptive uniprocessor.

Marko Bertogna et al. [29] analyzed a schedulability of global scheduling algorithms on multiprocessor platforms. They addressed the schedulability problem of periodic and sporadic real-time task sets with constrained deadlines preemptively scheduled on a multiprocessor platform. They assumed that a global work-conserving scheduler was used and migration from one processor to another was allowed during a task lifetime. The analysis would be applied to two typical scheduling algorithms: Earliest Deadline First (EDF) and Fixed Priority (FP). The effectiveness of the proposed test was showed through an extensive set of synthetic experiments.

Fengxiang Zhang et al. [30] analyzed a schedulability for real-time systems with Earliest Deadline First (EDF) scheduling. They proposed new results on necessary and sufficient schedulability analysis for EDF scheduling. These results reduce the calculation times, in all situations exponentially, for schedulable task sets. They also addressed and solved the problem of providing fast schedulability analysis which was necessary and sufficient for EDF scheduling with arbitrary relative deadlines. The experimental results for quick convergence processor-demand analysis were similar for all kinds of task sets. Quick convergence processor-demand analysis reduces the required number of calculations exponentially in almost all situations.

Xian-Bo He et al. [31] proposed a fuzzy Earliest Deadline First (EDF) scheduling algorithm which was suitable for embedded soft real-time systems in the uncertain environments. All tasks were periodic and a tasks criticality and deadline distance were also described with fuzzy set. In scheduling algorithm tasks with shorter fuzzy deadline distance were scheduled first to those with same fuzzy deadline distance level, their criticalities were considered first. The simulation test shows that fuzzy EDF model had less deadline missing ratios than those of traditional EDF algorithm and the important tasks had less deadline missing ratios than that of others tasks in an overloaded uncertain embedded soft real-time system.

Paulo Baltarejo Sousa et al. [32] implemented a slot-based task-splitting multiprocessor scheduling. They discussed and proposed some modifications to the slot-based task splitting algorithm driven by implementation concerns, and they report the first implementation of this family of algorithms in a real operating system running linux kernel version 2.6.34. They also conducted an extensive range of experiments on a 4-core multicore desktop PC running task-sets with utilizations of up to 88%. The results showed that the behaviour of implementation was in line with the theoretical framework behind it. In spite of the unpredictability of the linux kernel observed a good correspondence between theory and practice.

Dong-Song Zhang et al. [33] proposed an energy-efficient scheduling algorithm for sporadic real-time tasks in multiprocessor systems. They proposed an energy-efficient real-time scheduling algorithm named local remaining voltage frequency scaling-efficient area control error real-time scheduling algorithm for sporadic tasks. The experimental results proved that the existing algorithms has not only guaranteed the optimal feasibility of sporadic tasks, but also achieve more energy savings in all cases, especially in the case of high workloads.

Huang-Ming Huang et al. [34] implemented and evaluate mixed-criticality scheduling approaches for periodic tasks. They also presented the first side-by-side evaluation scheme of those approaches, for periodic mixed-criticality tasks on uniprocessor systems, under a mixed-criticality scheduling model that was common to all three approaches. To make a fair evaluation of zero-slack scheduling, they also addressed two previously open issues; a) how to accommodate execution of a task after its deadline, and b) how to

account for previously unidentified forms of interference between mixed-criticality tasks. Simulations showed that while priority assignment and period transformation were most likely to be able to schedule a randomly selected task set, a small fraction of the task sets were schedulable only under the zero-slack approach. Empirical evaluation demonstrates that user-space implementations of mechanisms to enforce period transformation and zero-slack scheduling could be achieved on linux without kernel modification; with suitably low overhead for mixed-criticality real-time task sets.

Wei Tong et al. [35] proposed hard-real-time scheduling on weakly programmable multi-core processor with application to multi-standard channel decoding. They addressed the channel decoding part of a radio. Each channel decoding application had its own throughput requirement. It consists of a set of tasks that process data in a streaming fashion. They focused on the channel decoding part of SDR. They used Synchronous Data Flow (SDF) and Cyclo-Static Data Flow (CSDF) graphs to model channel decoding functions.

Liu Dun-nan et al. [36] considered a fuzzy systems and knowledge discovery (FSKD) on real-time scheduling feedback fuzzy control system based on area control error and power generation error. Based on the real-time scheduling system, they increased Area Control Error (ACE) indicators on area control deviation and feedback control part of generating deviation indicators. They also established the acquisition and feedback control systems of the index based on power grid. They connected the objective of real-time scheduling with assessment criteria of the power grid, and meet the needs of assessment of the power grid. They also established the acquisition and feedback control system of the power generation deviation based on real-time power assessment in generation plant. They analyzed power delay because of technical factors and human-induced factors prevalent existing in reality, and improved the effect of real time scheduling.

Arnoldo Diaz-Ramirez et al. [37] presented a multiprocessor real-time scheduling simulation tool. They introduced a tool name was RealtssMP to perform scheduling analysis and simulation of multiprocessor real-time scheduling algorithms. Result were helpful in the evaluation of the performance of existing and new scheduling algorithms. RealtssMP was fully integrated with Kiwi, a visualization tool used to evaluate the

execution chronograms of the task sets. The proposed tool was flexible enough, allowing the seamlessly integration of additional scheduling algorithms and schedulability tests.

Jin Cui et al. [38] proposed fast high-level event-driven thermal estimator for dynamic thermal aware scheduling. They developed an iteration free calibration which accounts for leakage power. Thermal estimator was validated by experimentation and showed to be suitable for dynamic thermal aware scheduling proposed by them. They developed a predictive future thermal map which used to proposed several online heuristic scheduling policies. These policies were show to produce better overall thermal and real-time effects than previous dynamic thermal aware scheduling solutions.

2.1 OBSERVATIONS:

From the literature survey it has been observed that the proposed scheduling algorithm was not suitable for non linear system and have suffer from poor speed. So there is need to develop an algorithm which is suitable for both linear and non linear system. Scheduling is also help in the optimization of common resources in a multi node network. From these observations the objectives of thesis work have been drawn and stated in chapter 1.

CHAPTER 3

SCHEDULING ALGORITHMS

CPU scheduling deal with the problem of deciding which of the process in ready queue is to be allocated the CPU. There are several scheduling algorithms already exists. Scheduling algorithms or policies are mainly used for short term scheduling [47]. List of Commonly used scheduling algorithms are as follow

3.1 First Come First Served (FCFS) Algorithm

3.2 Short Job First (SJF) Scheduling Algorithm

3.3 Shortest Remaining Time (SRT) Scheduling Algorithm

3.4 Round Robin (RR) Scheduling Algorithm

3.5 Priority Scheduling Algorithm

3.6 Multi level Queue Scheduling Algorithm

Description of each scheduling algorithm one by one is given below.

3.1 First Come First Served (FCFS) Scheduling Algorithm

This is the simplest scheduling algorithm. This algorithm is also called first in and first out (FIFO) algorithm [44]. This algorithm is non pre-emptive. Table 3.1.1 has been taken to understand this algorithm. In the table 3.1.1, it can be seen that each process has its own burst time and arrival time. For example the process P1 has arrival time ‘0’ and his burst time ‘6’. Similarly process P2 has ‘0’ arrival time and ‘8’ burst time. Similarly again all process has their own arrival time and burst time.

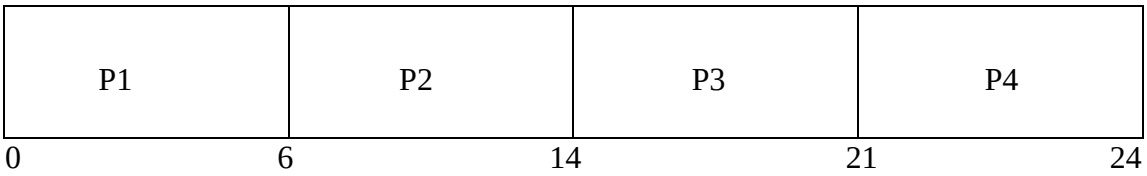
Table 3.1.1: Example for FCFS scheduling algorithm

Process	P1	P2	P3	P4

Burst time	6	8	7	3
Arrival time	0	0	0	0

- Arrival time: It is the time at which the process is arrives in the queue [41]. If the process is not being meeting at burst time, error was occurred in that process.
- Burst time: It is the time at which the process executed [22]. It means that the time at which the any process comes into the execution process queue.

Table 3.1.2: Gantt chart for FCFS algorithm



This algorithm helps to realize that which process first come that's first serve. Process P1 appears first in the execution queue; therefore it first executed first that is called first come first serve. After that process P2 come before all other process, so executed first, as compare to all other process. This process of execution continues until all the processes are executed. Here process P4 come at the end so, it executed lastly that's why it is called first come first serve scheduling algorithm. Average waiting time is calculated as: (0+6+14+21)/4 =10.25ns

Advantages

- This scheduling algorithm is better for long process.
- This is very Simple method.

Disadvantages

- Small process should wait for its turn to utilize the CPU.
- Throughput is not emphasized.

3.2 Short Job First (SJF) Scheduling Algorithm

Short job first (SJF) scheduling algorithm enters the short job first for execution. It means job which have less time to execution first executed in comparison to all other jobs [44]. In this scheduling algorithm the over heads are increased as compare to First Come First serves (FCFS) scheduling algorithm. Same example has been used to understand this scheduling algorithm and is given below:

Table 3.2.1: Example to understand this scheduling algorithm

Process	P1	P2	P3	P4
Burst time	6	8	7	3
Arrival time	0	0	0	0

Process P4 takes short time to execute in this algorithm and is shown in table 3.2.2 below:

Table 3.2.2: Gantt diagram for shortest job first scheduling algorithm

P4	P1	P3	P2

0 3 9 16 24

Average waiting time is calculated as:

(0+3+9+16)/4 =7ns

It has been observed that the average waiting time of short job first (SJF) scheduling algorithm is less in comparison to FCFS scheduling algorithm.

Advantage

- It provides high throughput high.
- It gives the superior turn- around time performance.

Disadvantage

- It increase the overhead on our processor

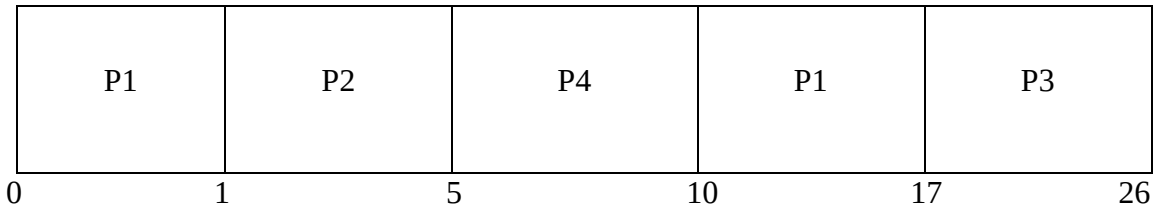
3.3 Shortest Remaining Time (SRT) Scheduling Algorithm

Shortest remaining time (SRT) scheduling algorithm first executes the process which has shortest remaining time [41]. Assume four process and they have different arrival time and burst time.

Table 3.3.1: Example for the SRT scheduling algorithm

Process	Arrival time	Burst time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Table 3.3.2: Gantt diagram for this type algorithm



Here table 3.3.2 reveals that the process P1 is started at time 0, since it is only process in the queue. Process P2 arrived at time 1. The remaining time for process P1 is larger than the time required by P2, so P1 is suspend and P2 is scheduled. This is also call pre-emptive. If the remaining time of process P1 is less as compare to P2 and all other process, P1 is continues executes but in this case process P1 has large remaining time, so

it is interrupted. Process P2 is fully executed because the execution time is less in comparison to all other process. After completing the execution of P2, process P4 is executed due to the same reason. Finally the Process P3 is executed because the remaining time of process P3 is large so process P3 is executed at the end. The average waiting time for this example is given as:

{(10-0) + (1-1) + (17-2) + (5-3)} / 4 = 26 / 6 = 6.5ns

3.4 Round Robin (RR) Scheduling Algorithm

In this algorithm a time slot is taken, that is called time slicing. So this algorithm is also called time slicing algorithm [41]. Following example has been used to understand the working of this algorithm.

Table 3.4.1: Example table for RR scheduling algorithm

PROCESS	P1	P2	P3
BURST TIME	24	3	3
ARRIVAL TIME	0	0	0

Here time slice of 4ns has been considered and after that using Round Robin algorithm all the process were organized in a manner.

Table 3.4.2 : Gantt chart for Round Robin scheduling algorithm

P1	P2	P3	P1	P1	P1	P1	P1

0 4 7 10 14 18 22 26 30

In Round Robin (RR) scheduling algorithm, there are three processes P1, P2 and P3. Process P1 first executed for 4ns because time slice indented is 4ns. After completing 4ns, process P2 executed for next 4ns and after that Process P3 will be executed for next 4ns. This is happens until the all process are completely executed. Average waiting time in the Round Robin scheduling algorithm is given as

{(0+4+7) + (10-4)}/3 = 18/3 = 6ns

3.5 Priority Scheduling Algorithm

In this type of scheduling algorithm, the priority is given for each process according to their necessity [41]. To understand priority scheduling an example is taken, which is shown below:

Table 3.5.1: Example for priority scheduling algorithm

Process	**Burst Time**	**Arrival Time**	**Priority**
P1	10	0	3
P2	1	0	1
P3	2	0	4
P4	1	0	5
P5	5	0	2

In this example there are five processes P1, P2, P3, P4 and P5. Each process has its own priority, arrival time and burst time which is shown in the table 3.5.1.

Table 3.5.2: Gantt chart for priority scheduling algorithm

P2	P5	P1	P3	P4

0 1 6 16 18 19

Table 3.5.2 reveals that process P2 is first executed due to their first priority shown in the table 3.5.1. After that process P5 is executed because second priority is given to this process. Similarly, all remaining process is executed according to which priority is given to that. In this algorithm the CPU is allocate to process with higher priority after completing the present running process. This is as first come first serve scheduling algorithm.

Advantage

- It provides better response for highest priority process.

Disadvantage

- Star vision may be possible for the lowest priority process only.
- Pre-emptive scheduling algorithm

3.6 Multi level Queue (MLQ) Scheduling Algorithm

Multi level queue scheduling algorithm partitions the ready queue into several separate queue. Processes are permanently assigned to each queue, usually based upon properties such as memory size or process type [44]. Each queue has its own scheduling algorithm. The interactive queue might be scheduling by a round robin algorithm while batch queue follows FCFS scheduling algorithm as an example of multi level queue scheduling one simple approach to partitioning of ready queue into system processes, interactive processes and batch processes and it creates a three ready queue [47]. It can be seen that in the figure 3.6 there are three scheduler, high priority scheduler, round robin scheduler and first come first out scheduler can be switch to CPU by using switch.

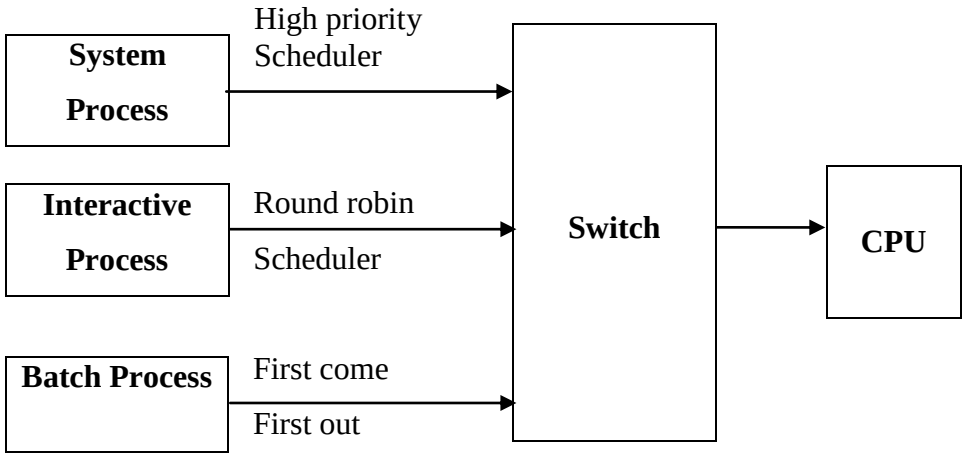


Fig. 3.6 multi level queue scheduling algorithm [47]

The comparison on the basis of performance measuring parameters of various algorithms has been shown in the following table 3.7

Table 3.7: Comparison of various scheduling algorithms

Scheduling Algorithm	**Parameters**			
	CPU Overhead	**Throughput**	**Turnaround Time**	**Response Time**
First In First Serve	Low	Low	High	Low
Shortest Job First	Medium	High	Medium	Medium
Shortest Remaining Time	Low	Medium	Medium	Medium
Priority Based	Medium	Low	High	High
Round Robin	High	Medium	Medium	High
Multi level Queue Scheduling	High	High	Medium	Medium

CHAPTER 4

SIMULATION AND RESULT

There is a need to optimize the method of finding shortest path for saving the resources for data communication between various nodes. The simulation results have been achived using “TORA” software.

4.1 Shortest path algorithms:

The shortest path is the path through which data or information is communicated from source node to destination node as compared to other paths. In other way a shortest path is the path between the two vertices in such a way that the sum of the weights of its constituent edges is minimized. There are many variations depending on the directivity of the graph [6]. For undirected weighted graph (graph with a set of V vertices, a set of E edges and a real valued weight function f: E→R) and elements v and v' of V, find a path P from v to v' so that

$\sum_{p \in P} f(p)$---------- (1)

is minimal along all paths connecting v to v'. There are some variations in the shortest path problems which are described below:

4.2 Single source shortest path problem:

When a single source is used to transmit data to all other nodes in a circuit, the problem is known as single source shortest path problem [17]. This problem is basically used to finding the shortest path from a single source to all the nodes or vertices in the graph. To find the possible shortest path from a particular source node to all other node, all the possibe paths are traversed.

4.3 Single destinaton shortest path problem:

In the single destination shortest path problem, a node with shortest path from all other possible node in the graph for the data or information transfer has to be found [19].

Shortest paths from all the vertices in the directed graph to a single vertex v has to be found. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.

All-pairs shortest path problem

Shortest path between every pair of vertices v and v' has to be calculated and thus the name is All-pairs shortest path problem.

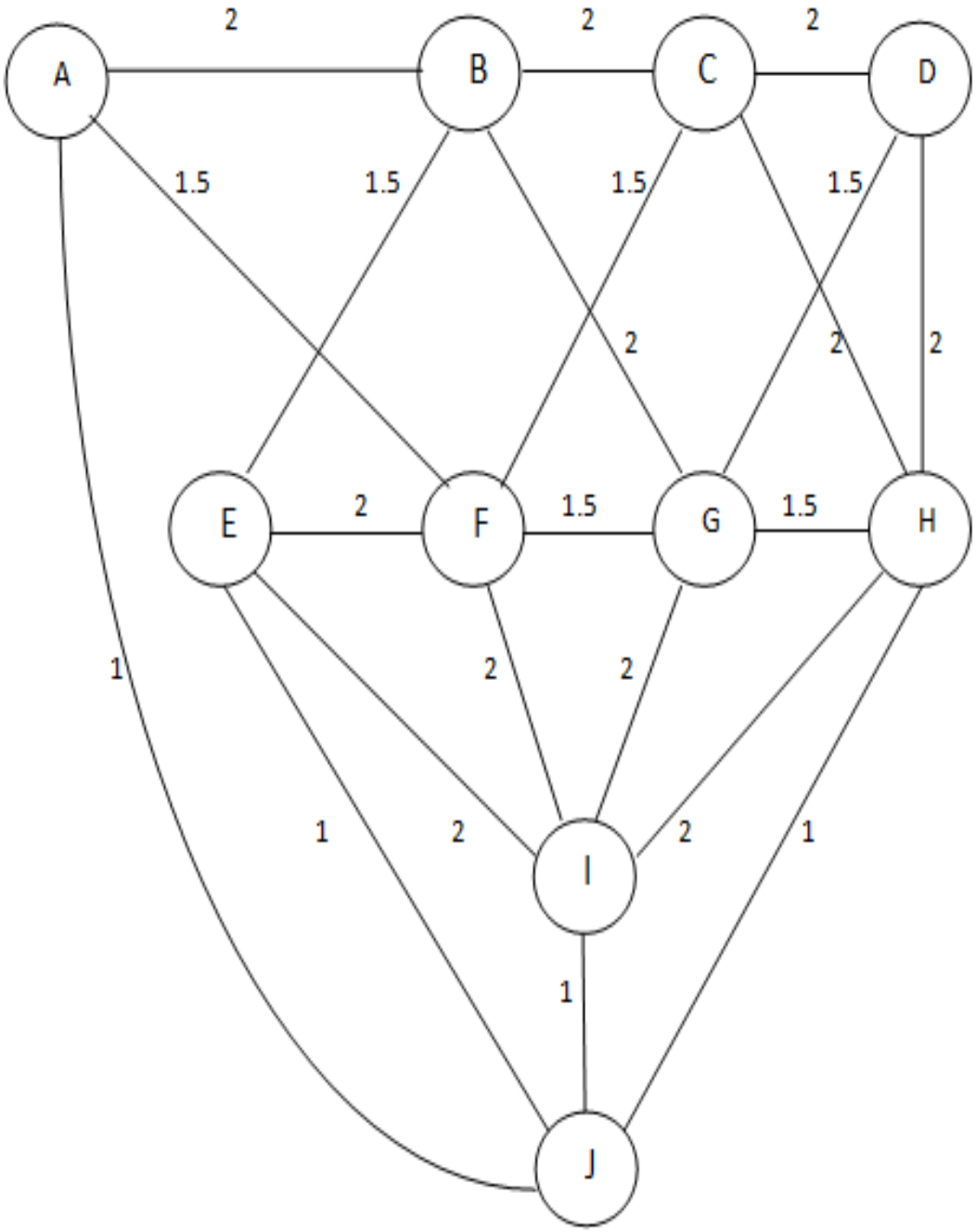


Figure 4.0 Undirected graph

The figure 4.0 is undirected graph. Basically there are two type of graph: a) Directed graph and b) Undirected graph. In the directed graph, data is transferred only in one direction and in the undirected graph, data is transferred in one direction to another direction and vice versa. Example of directed graph is shown in figure 4.0.1

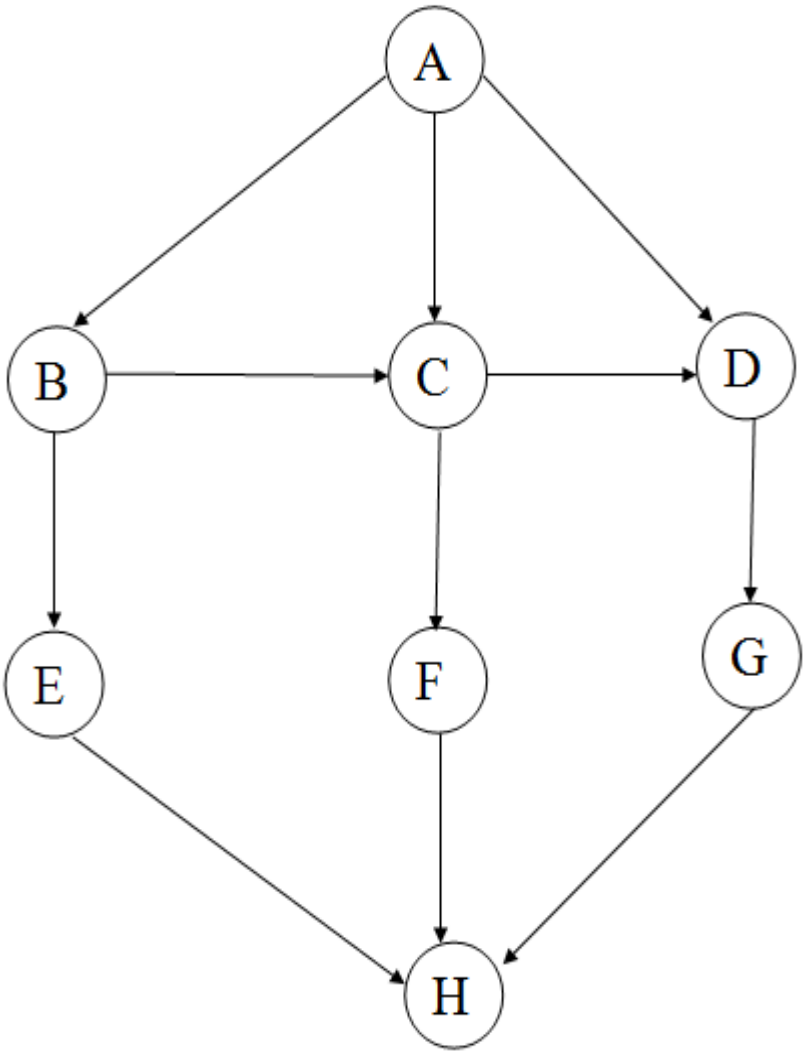


Figure 4.0.1 : Directed Graph

In above figure 4.0.1 there is an arrow direction in each vertics. The connected path between two or more node is called vertics. Vertics play an improrant role to send data or information from source node to destination node. In figure 4.0.1 data can be tranferred from node A to node C but not from node C to node A. If data is need to be transferred from node A to node H, then there are many paths like (i) A to C to F to H (ii) A to B to E to H and (iii) A to B to C to D to G to H. But there is no path to transfer the data from node H to A. So it reveal that the directed garph works like a one way road in which data transfer from one node to another node but it can not be followed back.

Table 4.1: Manual inspection table for undirected graph

S.No.	Node	Path	Delay	Intermediate Nodes
1.	A	$A \xrightarrow{1} J$	1	0
		$A \xrightarrow{1.5} F \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$A \xrightarrow{2} B \xrightarrow{2} G \xrightarrow{1.5} H \xrightarrow{1} J$	6.5	3
		$A \xrightarrow{1.5} F \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	5.5	3
2.	B	$B \xrightarrow{1.5} E \xrightarrow{1} J$	2.5	1
		$B \xrightarrow{2} A \xrightarrow{1} J$	3	1
		$B \xrightarrow{2} G \xrightarrow{1.5} G \xrightarrow{1} I$	4.5	2
3.	C	$C \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$C \xrightarrow{1.5} F \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$C \xrightarrow{2} D \xrightarrow{2} H \xrightarrow{1} J$	5	2
		$C \xrightarrow{2} H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	5.5	3
4.	D	$D \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$D \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	4	2
		$D \xrightarrow{2} C \xrightarrow{1.5} F \xrightarrow{2} I \xrightarrow{1} J$	6.5	3
5.	E	$E \xrightarrow{1} J$	1	0

		$E \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$E \xrightarrow{2} F \xrightarrow{1.5} A \xrightarrow{1} J$	3.5	2
6.	F	$F \xrightarrow{1.5} A \xrightarrow{1} J$	2.5	1
		$F \xrightarrow{2} E \xrightarrow{1} J$	3	1
		$F \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	4	2
		$F \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
7.	G	$G \xrightarrow{1.5} H \xrightarrow{1} J$	2.5	1
		$G \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$G \xrightarrow{1.5} F \xrightarrow{1.5} A \xrightarrow{1} J$	4	2
8.	H	$H \xrightarrow{1} J$	1	0
		$H \xrightarrow{2} I \xrightarrow{1.5} J$	2.5	1
		$H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$H \xrightarrow{1.5} G \xrightarrow{1.5} F \xrightarrow{1.5} A \xrightarrow{1} J$	5.5	3
9.	I	$I \xrightarrow{1} J$	1	0
		$I \xrightarrow{2} E \xrightarrow{1} J$	3	1

		$I \xrightarrow{2} G \xrightarrow{1.5} H \xrightarrow{1} J$	4.5	2
		$I \xrightarrow{2} F \xrightarrow{2} E \xrightarrow{1} J$	5	2

On manual inspection of the undirected graph is shown above table 4.1. Various paths have been observed for single source shortest path problem. For the node J, different paths have been observed. For node A and B there are various paths directed to other nodes and many nodes have to be traversed for reaching that node. For the transmission of the data to a particular node from A and B there can be multiple paths but a path with minimum delay or minimum intermediate nodes is needed.

From	To	Distance	Route
1-A	2-B	2.00	1- 2
1-A	3-C	3.00	1- 6- 3
1-A	4-D	4.00	1- 10- 8- 4
1-A	5-E	2.00	1- 10- 5
1-A	6-F	1.50	1- 6
1-A	7-G	3.00	1- 6- 7
1-A	8-H	2.00	1- 10- 8
1-A	9-I	2.00	1- 10- 9
1-A	10-J	1.00	1- 10
2-B	1-A	2.00	2- 1
2-B	3-C	2.00	2- 3
2-B	4-D	3.50	2- 7- 4
2-B	5-E	1.50	2- 5
2-B	6-F	3.50	2- 1- 6
2-B	7-G	2.00	2- 7
2-B	8-H	3.50	2- 7- 8
2-B	9-I	3.50	2- 5- 9
2-B	10-J	2.50	2- 5- 10

Figure 4.1 Shortest paths to all other nodes from node A and B

From the results obtained above it has been observed that transmitting the data from node A to node D takes maximum time is 4ns and has two intermediate nodes. If another path is considered from node A to node D, time taken has been found greater than or equal to $4\,ns$. If a random path $A \rightarrow F \rightarrow G \rightarrow D$ is taken, it will take 4.5ns and number of intermediate nodes will be two. Hence, there will not be any change in the terms of

intermediate nodes but time taken for the data transfer from source node to destination node has been reduced by:

$$\frac{.5}{4.5} \times 100 = 11.11\%.$$

The probability of error for node failure is same because there is no change in the intermediate in the random path taken. So shortest paths to all the nodes has been calculated which are helpful in resolving timing constraints and saving of resources. The following results have been achieved for the single destination problem, if C and D be the source. All the shortest paths has been found for all other nodes.

From	To	Distance	Route
3-C	1-A	3.00	3- 6- 1
3-C	2-B	2.00	3- 2
3-C	4-D	2.00	3- 4
3-C	5-E	3.50	3- 2- 5
3-C	6-F	1.50	3- 6
3-C	7-G	3.00	3- 6- 7
3-C	8-H	2.00	3- 8
3-C	9-I	3.50	3- 6- 9
3-C	10-J	3.00	3- 8- 10
4-D	1-A	4.00	4- 8- 10- 1
4-D	2-B	3.50	4- 7- 2
4-D	3-C	2.00	4- 3
4-D	5-E	4.00	4- 8- 10- 5
4-D	6-F	3.00	4- 7- 6
4-D	7-G	1.50	4- 7
4-D	8-H	2.00	4- 8
4-D	9-I	3.50	4- 7- 9
4-D	10-J	3.00	4- 8- 10

Figure 4.2 Shortest paths from node C and D to all other nodes

It has been observed that maximum amount of time to reach node I, from node C which is 3.5ns and one intermediate nodes are transversed. If a random path C → H → I, taken, then data will takes 4 ns to reach node I from node C and intermediate nodes remain 1. Hence, the time needed to send the data from source node to destination node is reduced by:

$$\frac{.5}{4} \times 100 = 12.5\%$$

In this case, there is no change in the number of traversed nodes or the number of intermediate nodes to reach node I from node C but the time required to reach that node

reduces by 12.5%. In this case, there is no probability of error for node failure because of the intermediate node is not changed. The following result, have been observed for the single destination problem, if E and F be the source. The following result describes the shortest path from node E and F to all other nodes.

From	To	Distance	Route
5-E	1-A	2.00	5- 10- 1
5-E	2-B	1.50	5- 2
5-E	3-C	3.50	5- 2- 3
5-E	4-D	4.00	5- 10- 8- 4
5-E	6-F	2.00	5- 6
5-E	7-G	3.50	5- 2- 7
5-E	8-H	2.00	5- 10- 8
5-E	9-I	2.00	5- 9
5-E	10-J	1.00	5- 10
6-F	1-A	1.50	6- 1
6-F	2-B	3.50	6- 1- 2
6-F	3-C	1.50	6- 3
6-F	4-D	3.00	6- 7- 4
6-F	5-E	2.00	6- 5
6-F	7-G	1.50	6- 7
6-F	8-H	3.00	6- 7- 8
6-F	9-I	2.00	6- 9
6-F	10-J	2.50	6- 1- 10

Figure 4.3 Shortest paths from node E and F to all other nodes

It has been observed from above figure 4.3 that the path from node E to node D has maximum time of 4ns. There are two intermediate nodes. If the other path is taken from node E to node D, that is E → B → C → D, then the time needed to reach node D from node E is 4.5 ns and intermediate nodes to be traversed are same. So, the time needed is reduced by:

$$\frac{.5}{4.5} \times 100 = 11.11\%$$

In this case there is no probability of error for node failure because the reason is same, no change in intermediate node. If there is change in intermediate node than the change in probability of error for node failure has been caculated by the formula which is given in following equation:

$$\sum_{k=1}^{n} {}^{\mathrm{n}}_{\mathrm{k}}\mathrm{c} \left(\frac{1}{10}\right)^{k} \left(\frac{9}{10}\right)^{n-k} \text{----------(2)}$$

The following result gives the shortest path to all the nodes from node G and H.

From	To	Distance	Route
7-G	1-A	3.00	7- 6- 1
7-G	2-B	2.00	7- 2
7-G	3-C	3.00	7- 6- 3
7-G	4-D	1.50	7- 4
7-G	5-E	3.50	7- 2- 5
7-G	6-F	1.50	7- 6
7-G	8-H	1.50	7- 8
7-G	9-I	2.00	7- 9
7-G	10-J	2.50	7- 8- 10
8-H	1-A	2.00	8- 10- 1
8-H	2-B	3.50	8- 7- 2
8-H	3-C	2.00	8- 3
8-H	4-D	2.00	8- 4
8-H	5-E	2.00	8- 10- 5
8-H	6-F	3.00	8- 7- 6
8-H	7-G	1.50	8- 7
8-H	9-I	2.00	8- 9
8-H	10-J	1.00	8- 10

Figure 4.4 Shortest paths from node G and H to all other nodes

From the above result, it has been observed that time taken by data to reach from node H to node B is maximum which is 3.5 ns. The number of intermediate nodes which has to be crossed for node H to node B is one . If the other path is taken from node H, H → D → C → B, the time taken to reach from node H to node B will be 6 ns and number of intermediate nodes is two. So the time is reduced by:

$$\frac{2.5}{6} \times 100 = 41.66\%$$

Thus result shows that there is change in the intermediate nodes and the time taken is reduced by 41.66%. Here the change in intermediate node is occured so the probability of error for node failure is calculated by the formula given below:

$$\sum_{k=1}^{n} {}^{n}_{k}\mathrm{c} \left(\frac{1}{10}\right)^{k} \left(\frac{9}{10}\right)^{n-k}$$

The probabilty of error for node failure is 0.19.

The shortest path to all other nodes from node I and J is shown below.

From	To	Distance	Route
9-I	1-A	2.00	9- 10- 1
9-I	2-B	3.50	9- 5- 2
9-I	3-C	3.50	9- 6- 3
9-I	4-D	3.50	9- 7- 4
9-I	5-E	2.00	9- 5
9-I	6-F	2.00	9- 6
9-I	7-G	2.00	9- 7
9-I	8-H	2.00	9- 8
9-I	10-J	1.00	9- 10
10-J	1-A	1.00	10- 1
10-J	2-B	2.50	10- 5- 2
10-J	3-C	3.00	10- 8- 3
10-J	4-D	3.00	10- 8- 4
10-J	5-E	1.00	10- 5
10-J	6-F	2.50	10- 1- 6
10-J	7-G	2.50	10- 8- 7
10-J	8-H	1.00	10- 8
10-J	9-I	1.00	10- 9

Figure 4.5 Shortest paths from node I and J to all other nodes

From the above result, it has been observed that the path form node I to node B has maximum time of 3.5ns. The number of intermediate nodes which is needed to be crossed on this path is one. If the optimized path $I \rightarrow G \rightarrow B$ is considered for node I to node B, then the time taken is 4ns and number of intermediate nodes is same. Now the optimized path as compared to the other arbitrary path is reduced by:

$$\frac{0.5}{4} \times 100 = 12.5\%$$

When one node is failed, effect on data transsmission can be analyzed by manual inspection and simulation result. By assuming node B is failed in graph, all the shortest paths from one node to all other nodes have been calculated. If node B is failed, it means that node B is not working, it neither recieves any kind of data nor send. Table 4.2 shows the manual inspection. It shows the data tranmission from one node to all other possible nodes in the graph. In the manual inspection the data trasmission path from node B is taken as infinity because the node B is failed in the graph so all the paths from this node is considered as infinity.

Table 4.2 : The manual inspecton table for the one node failuar condition.

S.No.	Node	Path	Delay	Intermediate Nodes
1.	A	$A \xrightarrow{1} J$	1	0
		$A \xrightarrow{1.5} F \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$A \xrightarrow{1.5} F \xrightarrow{1.5} C \xrightarrow{2} H \xrightarrow{1} J$	6	3
		$A \xrightarrow{1.5} F \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	5.5	3
2.	B	Infinite	–	–
3.	C	$C \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$C \xrightarrow{1.5} F \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$C \xrightarrow{2} D \xrightarrow{2} H \xrightarrow{1} J$	5	2
		$C \xrightarrow{2} H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	5.5	3
4.	D	$D \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$D \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	4	2
		$D \xrightarrow{2} C \xrightarrow{1.5} F \xrightarrow{2} I \xrightarrow{1} J$	6.5	3
5.	E	$E \xrightarrow{1} J$	1	0
		$E \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$E \xrightarrow{2} F \xrightarrow{1.5} A \xrightarrow{1} J$	3.5	2
6.	F	$F \xrightarrow{1.5} A \xrightarrow{1} J$	2.5	1

		$F \xrightarrow{2} E \xrightarrow{1} J$	3	1
		$F \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	4	2
		$F \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
7.	G	$G \xrightarrow{1.5} H \xrightarrow{1} J$	2.5	1
		$G \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$G \xrightarrow{1.5} F \xrightarrow{1.5} A \xrightarrow{1} J$	4	2
8.	H	$H \xrightarrow{1} J$	1	0
		$H \xrightarrow{2} I \xrightarrow{1.5} J$	2.5	1
		$H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$H \xrightarrow{1.5} G \xrightarrow{1.5} F \xrightarrow{1.5} A \xrightarrow{1} J$	5.5	3
9.	I	$I \xrightarrow{1} J$	1	0
		$I \xrightarrow{2} E \xrightarrow{1} J$	3	1
		$I \xrightarrow{2} G \xrightarrow{1.5} H \xrightarrow{1} J$	4.5	2
		$I \xrightarrow{2} F \xrightarrow{2} E \xrightarrow{1} J$	5	2

Figure 5.1 is first simulation result taking the condition of one node failure. In this figure, the distance from node B to other node, and other node to B is infinity, which shows that node is failure or is not in working condition. All the shortest path from node A and B to all other node is shown in figure 5.1.

From	To	Distance	Route
1-A	2-B	infinity	
1-A	3-C	3.00	1- 6- 3
1-A	4-D	4.00	1- 10- 8- 4
1-A	5-E	2.00	1- 10- 5
1-A	6-F	1.50	1- 6
1-A	7-G	3.00	1- 6- 7
1-A	8-H	2.00	1- 10- 8
1-A	9-I	2.00	1- 10- 9
1-A	10-J	1.00	1- 10
2-B	1-A	infinity	
2-B	3-C	infinity	
2-B	4-D	infinity	
2-B	5-E	infinity	
2-B	6-F	infinity	
2-B	7-G	infinity	
2-B	8-H	infinity	
2-B	9-I	infinity	
2-B	10-J	infinity	

Figure 5.1 Shortest paths from node A and B to all other nodes

When one node in failure condition

From above result, it has been observed that the path from node A to node D has maximum time of 4ns. The number of intermediate nodes which has to be crossed from node A to node D is two. If the other path is taken from node A , $A \rightarrow F \rightarrow C \rightarrow D$, then the time taken by data to reach from node A to node D will be 5ns and number of intermediate nodes is two. Thus the time needed with selected path is reduced by:

$$\frac{1}{5} \times 100 = 20\%.$$

Thus there is no change in the intermediate nodes but the time taken is reduced by 20%. So, by using simulation result 20% time has beens reduced to transmit data from node A to node D. This improves the system performance.

All the shortest path from node C and D to all other node is shown in figure 5.2.

From	To	Distance	Route
3-C	1-A	3.00	3- 6- 1
3-C	2-B	infinity	
3-C	4-D	2.00	3- 4
3-C	5-E	3.50	3- 6- 5
3-C	6-F	1.50	3- 6
3-C	7-G	3.00	3- 6- 7
3-C	8-H	2.00	3- 8
3-C	9-I	3.50	3- 6- 9
3-C	10-J	3.00	3- 8- 10
4-D	1-A	4.00	4- 8- 10- 1
4-D	2-B	infinity	
4-D	3-C	2.00	4- 3
4-D	5-E	4.00	4- 8- 10- 5
4-D	6-F	3.00	4- 7- 6
4-D	7-G	1.50	4- 7
4-D	8-H	2.00	4- 8
4-D	9-I	3.50	4- 7- 9
4-D	10-J	3.00	4- 8- 10

Figure 5.2 Shortest paths from node C and D to all other nodes

When one node in failure condition

The above result reflects that the path from node A to node D and from node D to node E have maximum time of 4ns. The number of intermediate nodes which has to be crossed from node D to node E is two . If the other path, D → H → J → A → F → E is considered from node D, then the time taken by data to reach from node E to node D is 7.5ns and number of intermediate nodes is four. Thus the time needed with selected path reduces by:

$$\frac{3.5}{7.5} \times 100 = 46.66\%$$

Thus it has been observed that there is no change in the intermediate nodes but the time taken is reduced by 46.66%. So, by using simulation result 46.66% time has been reduced to transmit data from node D to node E.

All the shortest path from node E and F to all other node is shown in figure 5.3.

From	To	Distance	Route
5-E	1-A	2.00	5- 10- 1
5-E	2-B	infinity	
5-E	3-C	3.50	5- 6- 3
5-E	4-D	4.00	5- 10- 8- 4
5-E	6-F	2.00	5- 6
5-E	7-G	3.50	5- 6- 7
5-E	8-H	2.00	5- 10- 8
5-E	9-I	2.00	5- 9
5-E	10-J	1.00	5- 10
6-F	1-A	1.50	6- 1
6-F	2-B	infinity	
6-F	3-C	1.50	6- 3
6-F	4-D	3.00	6- 7- 4
6-F	5-E	2.00	6- 5
6-F	7-G	1.50	6- 7
6-F	8-H	3.00	6- 7- 8
6-F	9-I	2.00	6- 9
6-F	10-J	2.50	6- 1- 10

Figure 5.3 Shortest paths from node E and F to all other nodes

When one node in failure condition

From the above result, it has been observed that the path from node E to node D has maximum time of 4ns. The number of intermediate nodes which has to be crossed from node E to node D is two . If the other path, E → F → A → J → H → D is considered from node E, then the time taken to reach from node E to node D is 7.5 ns and number of intermediate nodes is four. So, the time needed with selected path is reduced by:

$$\frac{3.5}{7.5} \times 100 = 46.66\%$$

So result shows that there is no change in the intermediate nodes but the time taken is reduced by 46.66%. So, simulation result time reduce by 46.66% to transmit data from node E to node D.

All the shortest path from node E and F to all other node is shown in figure 5.4.

From	To	Distance	Route
7-G	1-A	3.00	7- 6- 1
7-G	2-B	infinity	
7-G	3-C	3.00	7- 6- 3
7-G	4-D	1.50	7- 4
7-G	5-E	3.50	7- 6- 5
7-G	6-F	1.50	7- 6
7-G	8-H	1.50	7- 8
7-G	9-I	2.00	7- 9
7-G	10-J	2.50	7- 8- 10
8-H	1-A	2.00	8- 10- 1
8-H	2-B	infinity	
8-H	3-C	2.00	8- 3
8-H	4-D	2.00	8- 4
8-H	5-E	2.00	8- 10- 5
8-H	6-F	3.00	8- 7- 6
8-H	7-G	1.50	8- 7
8-H	9-I	2.00	8- 9
8-H	10-J	1.00	8- 10

Figure 5.4 Shortest paths from node G and H to all other nodes When one node in failure condition

From the result, it has been observed that path takes maximum time to reach node A from node G is 3.5ns. The number of intermediate nodes which has to be crossed for node G to node A is one. If any other random path has been taken from node G, G → I → J → A, then the time taken to reach node A from node G is 4ns and intermediate nodes is two. So, the time needed with selected path is reduced by:

$$\frac{.5}{4} \times 100 = 12.5\%$$

So result shows that there is no change in the intermediate nodes but the time taken is reduced by 12.5%. So, simulation result time reduce by 12.5% to tramission data from node G to node A.

All the shortest path from node E and F to all other node is show in the figure 5.5

From	To	Distance	Route
9-I	1-A	2.00	9- 10- 1
9-I	2-B	infinity	
9-I	3-C	3.50	9- 6- 3
9-I	4-D	3.50	9- 7- 4
9-I	5-E	2.00	9- 5
9-I	6-F	2.00	9- 6
9-I	7-G	2.00	9- 7
9-I	8-H	2.00	9- 8
9-I	10-J	1.00	9- 10
10-J	1-A	1.00	10- 1
10-J	2-B	infinity	
10-J	3-C	3.00	10- 8- 3
10-J	4-D	3.00	10- 8- 4
10-J	5-E	1.00	10- 5
10-J	6-F	2.50	10- 1- 6
10-J	7-G	2.50	10- 8- 7
10-J	8-H	1.00	10- 8
10-J	9-I	1.00	10- 9

Figure 5.5 Shortest paths from node I and J to all other nodes

When one node in failure condition

From the result, it has been observed that path takes maximum time to reach node C from node I which is 3.5ns and From node I to node D has same time 3.5ns. The path node I to node C concider here. The intermediate nodes which has to be crossed for node I to node C is one. If any other path, I → j → H → D → C is considered from node I , then the time taken to reach node C from node I will be $6ns$ and the number of intermediate nodes is three. Thus the time needed with selected path is reduced by:

$$\frac{2.5}{6} \times 100 = 41.66\%.$$

So result shows that there is no change in the number of intermediate nodes but the time taken is reduced by 41.66%. So by using simulation result 41.66% time is reduced to tramission data from node I to node C. When multiple node is failed simultaneouly, then effect on data transsmission can be analyzed by manual inspection and simulation result. Here node B and F in failure condition. Manual inspection and simulation analyzed again

for this condition . First of all manual inpection carried out which is show in the table 4.3. In manual inspection node B and node F show the infinity path.

Table 4.3: Manual inspection table for undirected graph when multiple node are failed

S.No.	Node	Path	Delay	Intermediate Nodes
1.	A	$A \xrightarrow{1} J$	1	0
2.	B	Infinite	–	–
3.	C	$C \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$C \xrightarrow{2} D \xrightarrow{2} H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	8.5	4
		$C \xrightarrow{2} D \xrightarrow{2} H \xrightarrow{1} J$	5	2
		$C \xrightarrow{2} H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	5.5	3
4.	D	$D \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$D \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	4	2
		$D \xrightarrow{2} C \xrightarrow{2} H \xrightarrow{2} I \xrightarrow{1} J$	7	3
5.	E	$E \xrightarrow{1} J$	1	0
		$E \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$E \xrightarrow{2} I \xrightarrow{2} H \xrightarrow{1} J$	5	2

6.	F	Infinite	–	–
7.	G	$G \overset{1.5}{\rightarrow} H \overset{1}{\rightarrow} J$	2.5	1
		$G \overset{2}{\rightarrow} I \overset{1}{\rightarrow} J$	3	1
		$G \overset{1.5}{\rightarrow} D \overset{2}{\rightarrow} H \overset{1}{\rightarrow} J$	4.5	2
8.	H	$H \overset{1}{\rightarrow} J$	1	0
		$H \overset{2}{\rightarrow} I \overset{1.5}{\rightarrow} J$	2.5	1
		$H \overset{1.5}{\rightarrow} G \overset{2}{\rightarrow} I \overset{1}{\rightarrow} J$	4.5	2
		$H \overset{1.5}{\rightarrow} G \overset{2}{\rightarrow} I \overset{2}{\rightarrow} E \overset{1}{\rightarrow} J$	6.5	3
9.	I	$I \overset{1}{\rightarrow} J$	1	0
		$I \overset{2}{\rightarrow} E \overset{1}{\rightarrow} J$	3	1
		$I \overset{2}{\rightarrow} G \overset{1.5}{\rightarrow} H \overset{1}{\rightarrow} J$	4.5	2
		$I \overset{2}{\rightarrow} G \overset{1.5}{\rightarrow} D \overset{2}{\rightarrow} H \overset{1}{\rightarrow} J$	6.5	3

Table 4.3 show the manual inspection when two node is simultaneously failed in the graph.

All the shortest path from node A and B to all other node is show in the Figure 6.1.

From	To	Distance	Route
1-A	2-B	infinity	
1-A	3-C	4.00	1- 10- 8- 3
1-A	4-D	4.00	1- 10- 8- 4
1-A	5-E	2.00	1- 10- 5
1-A	6-F	infinity	
1-A	7-G	3.50	1- 10- 8- 7
1-A	8-H	2.00	1- 10- 8
1-A	9-I	2.00	1- 10- 9
1-A	10-J	1.00	1- 10
2-B	1-A	infinity	
2-B	3-C	infinity	
2-B	4-D	infinity	
2-B	5-E	infinity	
2-B	6-F	infinity	
2-B	7-G	infinity	
2-B	8-H	infinity	
2-B	9-I	infinity	
2-B	10-J	infinity	

Figure 6.1 Shortest paths from node A and B to all other nodes

When the two node are simultaneously failed.

From the result, it has been observed that path takes maximum time to reach node C from node A is 4ns and From node A to node D is the same time 4ns. Consider the path node A to node C. The number of intermediate nodes which has to be crossed for node A to node C is two. If other random path, A → j → H → D → C is considered from node A, time taken to reach node C from node I is 6 ns and the number of intermediate nodes is three. Thus the time needed with random selected path is reduced by:

$$\frac{2}{6} \times 100 = 33.33\%$$

There is no change in the intermediate nodes but the time taken is reduced by 33.33%. To calculate the probability of error for node failure is by usin the formula which is shown below. In this case there is the value of n is 2.

$$\sum_{k=1}^{n} {}^{n}_{k}\mathrm{c}\left(\frac{1}{10}\right)^{k}\left(\frac{9}{10}\right)^{n-k}$$

Hence the probability of error in this codition is 0.19.

All the shortest path from node C and D to all other node is show in the figure 6.2.

From	To	Distance	Route
3-C	1-A	4.00	3- 8- 10- 1
3-C	2-B	infinity	
3-C	4-D	2.00	3- 4
3-C	5-E	4.00	3- 8- 10- 5
3-C	6-F	infinity	
3-C	7-G	3.50	3- 4- 7
3-C	8-H	2.00	3- 8
3-C	9-I	4.00	3- 8- 9
3-C	10-J	3.00	3- 8- 10
4-D	1-A	4.00	4- 8- 10- 1
4-D	2-B	infinity	
4-D	3-C	2.00	4- 3
4-D	5-E	4.00	4- 8- 10- 5
4-D	6-F	infinity	
4-D	7-G	1.50	4- 7
4-D	8-H	2.00	4- 8
4-D	9-I	3.50	4- 7- 9
4-D	10-J	3.00	4- 8- 10

Figure 6.2 Shortest paths from node C and D to all other nodes

When the two node are simultaneously failed.

From the result, it has been observed that it takes maximum time to reach node I from node C which is 4ns and From node D to node A has same time 4ns. If a random path node C to node I is reduced. The number of intermediate nodes which has to be crossed for node C to node I is two. If other path, C → D → H → j → I is taken from node I the time taken to reach node I from node C will be $6ns$ and there is three intermediate nodes. Thus the time needed with selected path is reduced by:

$$\frac{2}{6}\times 100 = 33.33\%$$

There is no change in the intermediate nodes but the time taken is reduced by 33.33%. To calculate the probability of error for node failure is by using the formula which is show in below. In this case there is the value of n is 2.

$$\sum_{k=1}^{n} {}^{n}_{k}\mathrm{c}\left(\frac{1}{10}\right)^{k}\left(\frac{9}{10}\right)^{n-k}$$

Hence the probability of error in this codition is 0.19.

All the shortest path from node E and F to all other node is show in the figure 6.3.

From	To	Distance	Route
5-E	1-A	2.00	5- 10- 1
5-E	2-B	infinity	
5-E	3-C	4.00	5- 10- 8- 3
5-E	4-D	4.00	5- 10- 8- 4
5-E	6-F	infinity	
5-E	7-G	3.50	5- 10- 8- 7
5-E	8-H	2.00	5- 10- 8
5-E	9-I	2.00	5- 9
5-E	10-J	1.00	5- 10
6-F	1-A	infinity	
6-F	2-B	infinity	
6-F	3-C	infinity	
6-F	4-D	infinity	
6-F	5-E	infinity	
6-F	7-G	infinity	
6-F	8-H	infinity	
6-F	9-I	infinity	
6-F	10-J	infinity	

Figure 6.3 Shortest paths from node E and F to all other nodes

When the two node are simultaneously failed.

From the result, it has been observed that path it takes maximum time to reach node C from node E which is 4ns and From node E to node D path has same time 4ns. If consider the random path node E to node D the intermediate nodes crossed for node E to node D is 2 . If any other random path has been taken from node E, E → I → G → H → D, the time taken to reach node D from node E will be 7.5ns and there were three intermediate nodes. Thus the time needed with selected path is reduced by:

$$\frac{3.5}{7.5} \times 100 = 46.66\%$$

There is no change in the intermediate nodes but the time reduced by 46.66%. It improves overll circuit performance and efficiency. To calculate the probability of error for node failure the formula which is given in below. In this case there is the value of n is 2.

$$\sum_{k=1}^{n} {}^{n}_{k}c \left(\frac{1}{10}\right)^{k} \left(\frac{9}{10}\right)^{n-k}$$

Hence the probability of error in this codition is 0.19.

All the shortest path from node E and F to all other node is show in the figure 6.4.

From	To	Distance	Route
7-G	1-A	3.50	7-8-10-1
7-G	2-B	infinity	
7-G	3-C	3.50	7-4-3
7-G	4-D	1.50	7-4
7-G	5-E	3.50	7-8-10-5
7-G	6-F	infinity	
7-G	8-H	1.50	7-8
7-G	9-I	2.00	7-9
7-G	10-J	2.50	7-8-10
8-H	1-A	2.00	8-10-1
8-H	2-B	infinity	
8-H	3-C	2.00	8-3
8-H	4-D	2.00	8-4
8-H	5-E	2.00	8-10-5
8-H	6-F	infinity	
8-H	7-G	1.50	8-7
8-H	9-I	2.00	8-9
8-H	10-J	1.00	8-10

Figure 6.4 Shortest paths from node G and H to all other nodes

When the two node are simultaneously failed.

From the result, it has been observed that it takes maximum time to reach node A from node G which is 3.5ns and From node G to node C path has same time 3.5ns but the diffirence is that there is less number of intermediate node so prefer the the path G node to node C. If path G node to C node consider, the intermediate nodes which has to be crossed for node G to node C is 1. If any other random path has been taken from node G,

G → I → H → D → C, the time taken to reach node C from node G will be $8ns$ and there were 3 intermediate nodes. Thus the time needed with our selected path is reduced by:

$$\frac{4.5}{8} \times 100 = 56.25\%$$

There is no change in the intermediate nodes but the time taken is reduced by 56.25%. It improves overall circuit performance and efficiency. The Probability of error for node failure is calculated by the formula which is given below:

$$\sum_{k=1}^{n} {}^{n}_{k}\mathrm{c} \left(\frac{1}{10}\right)^{k} \left(\frac{9}{10}\right)^{n-k}$$ here the value of n is 1.

Hence the probability of error in this codition is 0.1.

All the shortest path from node E and F to all other node is show in the figure 6.5.

From	To	Distance	Route
9-I	1-A	2.00	9- 10- 1
9-I	2-B	infinity	
9-I	3-C	4.00	9- 8- 3
9-I	4-D	3.50	9- 7- 4
9-I	5-E	2.00	9- 5
9-I	6-F	infinity	
9-I	7-G	2.00	9- 7
9-I	8-H	2.00	9- 8
9-I	10-J	1.00	9- 10
10-J	1-A	1.00	10- 1
10-J	2-B	infinity	
10-J	3-C	3.00	10- 8- 3
10-J	4-D	3.00	10- 8- 4
10-J	5-E	1.00	10- 5
10-J	6-F	infinity	
10-J	7-G	2.50	10- 8- 7
10-J	8-H	1.00	10- 8
10-J	9-I	1.00	10- 9

Figure 6.5 Shortest paths from node I and J to all other nodes When the two node are simultaneously failed.

From the result, it has been observed that it takes maximum time to reach node C from node I which is 4ns . The path from node I to node C is consider here. The number of intermediate nodes which has to be crossed for node I to node C is one. If any other path has been taken from node I, I → j → H → D → C, the time taken to reach node C from

node I will be $6ns$ and there is three intermediate nodes. Thus the time needed with selected path is reduced by:

$$\frac{2.5}{6} \times 100 = 41.66\%$$

Thus it has been seen that there is change in number intermediate nodes and the time taken is reduced by 41.66%. Probability of error for node failure is calculated by the formula which is show blow

$$\sum_{k=1}^{n} {}^{\mathrm{n}}_{\mathrm{k}}\mathrm{c} \left(\frac{1}{10}\right)^{k} \left(\frac{9}{10}\right)^{n-k}$$

Hence the probability of error in this codition is 0.19. That improve our system performance.

CHAPTER 5

CONCLUSIONS AND FUTURE SCOPE

The literature survey has been carried out by considering the various parameters such as processing speed, accuracy and scheduling algorithms. The observations from the literature survey have been stated in the chapter 2 which clearly highlights that the performance of the network to handle real time tasks enhanced by employing the scheduling algorithms. The various scheduling algorithm is analysed and also compared to all scheduling algorithm. The implementation of shortest path algorithm has been evaluated for the tasks. The problem for finding the shortest path in the communication network has also been taken and simulated accordingly. Evaluation of shortest path and probability of error for different nodes have been done. Resources optimization has been achieved in our technique as it only involves the optimal path between two nodes using the "TORA" software.

LIST OF PUBLICATION

- Pankaj Udvanshi, Ajay Kakkar, "Scheduling of Real Time Tasks", IOSR Journal ISSN: 2278-8719 Vol. 3, pp. 44-58, June 2013.
- Pankaj Udvanshi, Ajay Kakkar, “Scheduling Algorithms”, International Conference on Electronics, Communication and Information Technology, Thapar University 4-5 July 2013 (communicated).

REFERENCES

[1] Jane W.S. Liu, Kwei-Jay Lin, Wei-Kuan Shih, and Albert Chuang-shi Yu, "Algorithms for Scheduling Imprecise Computations", IEEE Transactions on Computer, Vol. 24, pp. 58-68, 1991.

[2] Christopher Chase and Peter J. Ramadge "On Real-Time Policies for Flexible Manufacturing System", IEEE Transaction on Automatic Control,Vol. 37, No.4, pp. 491-496, 1992.

[3] John A. Stankovic "The Many Faces of Multi-Level Real-Time Scheduling", Proceeding of Real-Time Systems and Applications, pp. 2-5, 1995.

[4] Y.X. Dai and B.S. Chen "An Alternate Approach for Real-time Scheduling", Proceedings of the IEEE International Conference on Industrial Technology, pp. 189-193, December 1996.

[5] Paolo Bizzarri, Andrea Bondavalli and Felicita Di Giandomenico "A Scheduling Algorithm for Aperiodic Groups of Tasks in Distribute-Time Systems and its Holistic Analysis", IEEE Proceeding on Distributed Computing Systems, pp. 296-301, 1997.

[6] David Montana, Garrett Bidwell and Sean Moore "Genetic Algorithms for Complex, Real-Time Scheduling Applications", IEEE Transactions Network Operations and Management Symposium, Vol. 1, pp. 245-248, 1998.

[7] Masahiko Abe, Hideyuki Matsumoto and Chiaki Kuroda "An Artificial Neural Network Optimized by a Genetic Algorithm for Real-time Flow-shop Scheduling" International Conference on knowledge-Bad Intelligent Engineering System & Allied Technologies, Brighton, UK, Vol. 1, pp. 329-332, September 2000.

[8] Tei-Wei Kuo, Wang-Ru Yang, and Kwei-Jay Lin "A Class of Rate-Based Real Time Scheduling Algorithms", IEEE Transactions on Computers, Vol. 51, No. 6, pp.708-720, 2002.

[9] Kwei-Jay Lin, Yu-Chung Wang, Ting-Hua Chien, Yaa-Jyun Yeh "Designing Multimedia Applications on Real-Time Systems with SMP Architecture", Proceeding of 4th International Symposium on Multimedia Software Engineering (ISMSE 2002), pp. 17-24, 2002.

[10] Deji Chen, Aloysius K. Mok, and Tei-Wei Kuo "Utilization Bound Revisited", IEEE Transactions on Computers, Vol. 52, No. 3,pp. 351-361, 2003.

[11] S. K. Baruah and Joel Goossens "Rate-Monotonic Scheduling on Uniform multi-processors", IEEE Transactions on Computers, Vol. 52, No. 7, pp. 966-970, 2003.

[12] C.M. Krishna and Yann-Hang Lee "Voltage-Clock-Scaling Adaptive Scheduling Techniques For Low Power in Hard Real-Time Systems", IEEE Transactions on Computers, Vol. 52, No.12, pp. 1586-1593, 2003.

[13] Kuan-Yu Chen, Alan Liu and Chiung-Hui Leon Le "A Multiprocessor Real-Time Process Scheduling Method", IEEE 5th International Symposium on Multimedia Software Engineering, pp. 29-36, 2003.

[14] Peng Li and Binoy Ravindran "Fast, Best-Effort Real-Time Scheduling Algorithms" IEEE Transactions on Computers, Vol. 53, No. 9, pp. 1159-1175, 2004.

[15] Dan Ma, Wei Zhang and Qinghua Li "Dynamic Scheduling Algorithm for Parallel Real-time Jobs in Heterogeneous System", 4th International Conference on Computer and Information Technology, pp. 462-466, September 2004.

[16] Pengliu Tan, Hai Jin, Minghu Zhang ''A Hybrid Scheduling Scheme for Hard, Soft and Non-Real-time Tasks", IEEE International Symposium on Object and Component-Oriented Real-Time Distributed computing, 2006.

[17] Yang-ping Chen, Lai-xiong Wang, and Shi-tan Huang Xi "A Novel Task Scheduling Algorithm for Real-Time Multiprocessor Systems", IEEE International Conference on Control and Automation Guangzhou, CHINA, pp. 271-275, June 2007.

[18] Octav Chipara, Chenyang and Gruia-Catalin Roman "Real-time Query Scheduling for Wireless Sensor Networks", 28th IEEE International Real-Tim Systems Symposium pp. 389-399, 2007.

[19] Jian-Jia Chen , Chuan-Yue Yang, Hsueh-I Lu and Tei-Wei Kuo "Approximation Algorithms for Multiprocessor Energy-Efficient Scheduling of Periodic Real-Time Tasks with Uncertain Task Execution Time", Real-Time and Embedded Technology and Applications Symposium, pp. 13-23, 2008.

[20] Robert Glaubius, Terry Tidwell, William D. Smart and Christopher Gill "Scheduling Design and Verification for Open Soft Real-timeSystems", Real-Time System Symposium, pp. 505-514, 2008.

[21] Lars Lundberg and Hakan Lennerstad "Slack-Based Global Multiprocessor Scheduling of Aperiodic Tasks in Parallel Embedded Real-Time

Systems", IEEE/ACS International Conference on Computer Systems and Applications, pp. 465-472, 2008.

[22] Yi-Hsiung Chao, Shun-Shii Lin and Kwei-Jay Lin "Schedulability issues for EDZL scheduling on real -time multiprocessor systems", Elsevier Information Processing Letters, Vol. 107, pp. 158-164, 2008.

[23] Zhu Xiang bin "A Dynamic Window-Constrained Scheduling Algorithm for Multiprocessor Real-Time Systems", 5th IEEE International Symposium on Embedded Computing, pp. 3-8, 2008.

[24] Ming Xiong, Song Han, Kam-Yiu Lam and Deji Chen "Deferrable Scheduling for Maintaining Real-Time Data Freshness Algorithms, Analysis and Results", IEEE Transactions on Computers, Vol. 57, No. 7, pp. 952-962, 2008.

[25] Euiseong Seo, Jinkyu Jeong, Seonyeong Park, and Joonwon Lee "Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors", IEEE Transactions on Parallel and Distributed Systems, Vol. 19, No. 11, pp. 1540-1552, 2008.

[26] F. Muhammad, Fabrice Muller and Michel Auguin "Slack-Conserving based Scheduling of Periodic Real-Time Tasks", 5th IEEE International Symposium on Embedded Computing, pp. 37-42, 2008.

[27] Aihan Yin, Peizhou Zhang, Wei Zhou, and Yueli Jiao "Dynamic Programmable Scheduling Mechanism of GEPON", Journal of Lightwave Technology, Vol. 27, No. 19, pp. 4289-4296, 2009.

[28] Enrico Bini, Thi Huyen Chau Nguyen, Pascal Richard, and Sanjoy K. Baruah "A Response-Time Bound in Fixed-Priority Scheduling with Arbitrary Deadlines", IEEE Transactions on Computers, Vol. 58, No. 2, pp. 279-286, 2009.

[29] M. Bertogna, Michele Cirinei, and Giuseppe Lipari "Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms", IEEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 4, pp. 952-964, 2009.

[30] Fengxiang Zhan and Alan Burns "Schedulability Analysis for Real-Time Systems with EDF Scheduling", IEEE Transactions on Computers, Vol. 58, No. 9, pp. 1250-1258, 2009.

[31] Xian-Bo He, Gang-Yuan Zhang, Min Liu, Yu-Ping Zhao,Wei Li "A Fuzzy EDF Scheduling Algorithm Being Suitable for Embedded Soft Real-time Systems in the Uncertain Environments", International Conference on Advanced Computer Control (ICACC),, Vol. 5, pp. 583-587, 2010.

[32] Paulo Baltarejo Sousa, Bjorn Andersson and Eduardo Tovar "Implementing Slot-Based Task-Splitting Multiprocessor Scheduling", 6th IEEE International Symposium on Industrial Embedded Systems (SIES), pp. 256-265, 2011.

[33] Dong-Song Zhang, Fang-Yuan Chen, Hong-Hua Li, Shi-Yao Jin and De-Ke Guo "An Energy-Efficient Scheduling Algorithm for Sporadic Real-Time Tasks in Multiprocessor Systems", IEEE International Conference on High Performance Computing and Communications, pp. 187-194, June 2011.

[34] Huang-Ming Huang, Christopher Gill and Chenyang Lu "Implementation and Evaluation Mixed-Criticality Scheduling Approaches for Periodic Tasks", Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 23-32, 2012.

[35] Wei Tong, Orlando Moreira, Rick Nas and Kees van Berkel "Hard-Real-Time Scheduling on a Weakly Programmable Multi-core Processor with Application to Multi-standard Channel Decoding", Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 151-160, 2012.

[36] Liu Dun-nan, Jiang Xin-fan, Hu Bin-qi, Zhang Si-yuan "Real-time Scheduling Feedback Fuzzy Control System based on Area Control Error and Power Generation Error", 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 352- 355, May 2012.

[37] A. Diaz-Ramirez, Dulce Orduno and Pedro Mejia-Alvarez "A Multiprocessor Real-Time Scheduling Simulation Tool", International Conference on Electrical Communications and Computers (CONIELECOMP), pp. 157-161, February 2012.

[38] Jin Cui and Douglas L. Maskell "A Fast High-Level Event-Driven Thermal Estimator for Dynamic Thermal Aware Scheduling", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 31, No. 6, pp. 904-917, 2012.

[39] Qin Xiao, Pang Liping, Li Shengli and Han Zongfen "Efficient Scheduling Algorithm with Fault-tolerance for Real-time Tasks in Distributed Systems", CiteSeerX, 1999.

[40] Wan Yeon Lee "Energy-Efficient Scheduling of Periodic Real-Time Tasks on Lightly Loaded Multicore Processors", IEEE Transactions on Parallel and Distributed Systems, Vol. 23, No.3, pp. 530-537, 2012.

[41] CPU Scheduling", http://www.os-concepts.thiyagaraaj.com/cpu-process scheduling.

[42] Marlin Litoiu, Traian C. lonescu and Jusus Labarta "Dynamic Task Scheduling in distributed Real Time System using Fuzzy rule", Microprocessors and Microsystems journal Vol. 21, pp. 299–311, 1998.

[43] J. Perkins, P. R. Kumar "Distributed Scheduling of Flexible Manufacturing Systems: Stability and Performance", IEEE Transaction Robotics Automation 1994.

[44] William Stallings, "Operating Systems", 5th Edition.

[45] Li Jie, Guo Ruifeng and Shao Zhixiang "The Research of Scheduling Algorithms in Real-time System", International Conference on Computer and Communication Technologies in Agriculture Engineering, Vol. 1, pp. 333–336, June 2010.

[46] Lalatendu Behera and Durga Prasad Mohapatra "Schedulability Analysis of Task Scheduling in Multiprocessor Real-Time Systems Using EDF Algorithm", International Conference on Computer Communication and Informatics Coimbatore, India, pp. 1-9, January 2012.

[47] Peter Brucker, "Scheduling Algorithms", Springer, 5th edition.