Dynamic Scheduling in Real Time for Resource Optimization

Thesis Submitted in the partial fulfillment of requirement for the award of degree of

Master of Technology

in

VLSI Design & CAD

Submitted by:

Robin Garg

Roll No. : 601061020

Under the guidance of:

Mr. Ajay Kakkar

Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

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

PATIALA – 147004 (PUNJAB)

2012

Declaration

I, Robin Garg, hereby certify that the work which is being presented in this thesis entitled "Dynamic Scheduling in Real Time for Resource Optimization" by me in partial fulfillment of the requirement for the award of degree of Master of Technology in VLSI Design & CAD from Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Mr. Ajay Kakkar.

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

Robin

Date: 25/09/12

Robin Garg
Roll. No. 601061020

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

Date: 25/09/12

Mr. Ajay Kakkar
Assistant Professor
Thapar University, Patiala

Countersigned by:

(Dr. Rajesh Khanna)
Professor and Head ECED
Thapar University, Patiala
Date:

(Dr. S. K. Mohapatra)
Dean of Academic Affairs
Thapar University, Patiala
Date:

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 this 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 member, 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.

Robin Garg
601061020

Abstract

Scheduling is a decision making process which deals with the allocation of resources to tasks over given time periods. Scheduling means how the processes can be assigned on the available CPU. It is a key feature in multitasking, multiprocessing and real-time operating system design. Scheduling is done by scheduler and dispatcher. A scheduler is a person or machine that organizes or maintains schedules. A dispatcher is a module which gives control of CPU to the process selected by the scheduler. Scheduling problem involves jobs that must schedule on machines subject to certain constraints to optimize some objective functions. In real time systems the correctness of the results depends not only on the logical computations but also on the time at which results are produced. In real time environments, scheduler needs to ensure that processes meet the deadlines that are crucial for keeping the system safe. The tasks which result in catastrophic conditions on missing the deadline are the hard real time tasks and the tasks whose results are useful even after missing the deadline are known as soft real time systems. In real time system, processes can dynamically increase or decrease in priority depending on whether it has been serviced already, or it is waiting extensively. The priority assignment schemes assign priorities to different tasks which can be static or dynamic. The study of various scheduling algorithms along with the comparison between different real time scheduling algorithms has been done in the thesis. Resource optimization is of main concern in case of scheduling in real time. Shortest path algorithms achieve resource optimization between different nodes by finding the shortest path using dynamic scheduling. Shortest distance between the nodes results in resource optimization and the tasks are scheduled through that optimized path in order to achieve feasible and efficient schedule. The shortest paths between all the nodes along with the calculation of probability of error in all the paths have been evaluated using dynamic scheduling by considering two cases: (i) when node $N4$ gets failed (ii) when critical node $N12$ gets failed. From the above two cases it has been observed that the probability of error or failure comes out to be the most for nodes $N13$, $N12$ and B.

Table of Contents

List of Figure

List of Tables

Chapter 1: Introduction

Introduction

Scheduling is concern with the optimal allocation of resources to activities in a specified time. It is a key feature in multitasking, multiprocessing and real-time operating system design. It is a decision making process that deals with the allocation of common resources to various tasks at different time periods to achieve multiple objectives. Scheduling problem involves jobs that must schedule on machines subject to certain constraints to optimize some objective functions. It is done by means of scheduler and dispatcher. Scheduler selects the processes that are ready to execute. A dispatcher is a module that performs the work of passing the CPU to the next selected process. The time to do this is called dispatch latency. In real time environments, such as embedded systems for automatic control in industry, the scheduler needs to ensure that processes meet the deadlines that are crucial for keeping the system safe. The resources and task can be of different forms in homogeneous/heterogeneous organization. Scheduling helps processes to perform input/output operation in the normal course of computation. Since input/output operation require more time to complete than CPU instructions, multiprogramming system allocate the CPU to process which invokes an input/output operation.

1.1 Types of Processor Scheduling

The aim of processor scheduling is to assign processes to be executed by the processor in a way that meets system objectives, such as response time, throughput, and processor efficiency. In many systems, this scheduling activity is broken into three separate functions (i) Long term scheduling (ii) Medium term scheduling (iii) Short term scheduling

1.1.1 Long term scheduling:

The long term scheduler determines which programs are admitted to the systems for processing, thus, it controls the degree of multiprogramming [5]. Once admitted, a job or user program becomes a process and is added to the queue for the short term scheduler. In a batch system, long term scheduler creates processes and forms the queue wherever it is possible. The more processes are created; the smaller is the percentage of time for which each process can be executed. Long term schedulers may limit the degree of multiprogramming to provide satisfactory service to the current set of processes.

1.1.2 Midterm scheduling:

The scheduling of processes is mainly based on the requirement of the resources. It is essentially concern with memory management and often design as a memory management subsystem of an operating system [5,25]. It temporarily removes a process from the main memory which is of low priority or has been inactive for a long time. This is known as "swamping out" of a process. The scheduler may decide to swamp out the process which is frequently page-faulting or a process which is taking large amount of memory. Its efficient interaction with the short term scheduler is very essential for the performance of the systems with virtual memory.

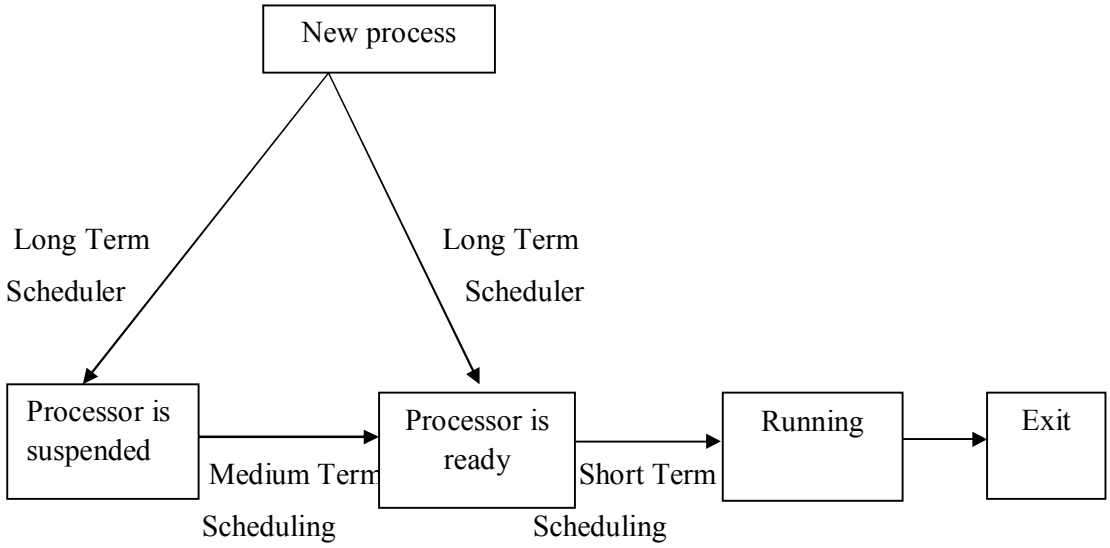

Figure 2.1: Types of Scheduling

1.1.3 Short Term Scheduling:

In terms of frequency of execution, the long term scheduler executes relatively infrequently and makes the coarse grained decision of whether or not to take on a new process and which one to take. Short term scheduler is invoked whenever an event occurs that may lead to the blocking of the current process that may provide an opportunity to preempt a currently running process in favor of another. CPU scheduling decisions can occur on the given conditions: (a) either the running process changes from running to waiting state or when the running process terminates (b) The waiting process becomes ready and (c) the current process switches from running to ready state.

1.2 Goals of Scheduling Algorithm for Different Systems

There are some goals that must be achieved in order to perfectly schedule the task on the processor. Some of these goals are mentioned below:

(i) Fairness: Fairness is important under all circumstances. A scheduler makes sure that each process gets its fair share of the CPU and no process suffer indefinite postponement. Note that giving equivalent or equal time is not fair [25].

(ii) Policy Enforcement: The scheduler has to make sure that system's policy is enforced. For example, if the local policy is safety then the safety control processes must be able to run whenever they want to, even if it means delay in payroll processes.

(iii) Efficiency: Scheduler should keep the system busy cent percent of the time when possible. If the CPU and the entire input/output device run for entire time, more work gets done per second [25].

(iv) Meeting deadlines: Scheduler should finish all its processes before the deadline of process otherwise catastrophic results can occur [8].

1.3 Performance Criteria

In order to achieve an efficient processor management, OS tries to select the most appropriate process from the ready queue. The terms associated with a process are:

(i) Processor Utilization: It is the ratio of busy time of the processor to the total time passes for processes to finish. Processor needs to remain as busy as possible in order to perform the defined task [13].

(ii) Throughput: It is defined as the number of processes that complete their execution per time unit. One way to measure throughput is by mean of the number of processes that are completed in a unit time.

(iii) Turnaround Time: It is the time interval from submission of job till its completion. It includes actual processing time plus time spent on waiting for resources, including the processor.

(iv) Waiting Time: It is the time spent in ready queue by a process to get the CPU for completion of process. Processor scheduling algorithms only affect the time spent waiting in the ready queue [13].

(v) Response Time: It is the time taken from the time when a request was submitted to till the first response is produced. Often a process can begin producing some output to the user while continuing to process the request.

1.4 Types of Scheduling Systems:

We have different systems like batch system, interactive system, real time system etc. to process a job and these systems are explained as following:

1.4.1 Batch Systems:

The batch systems do not need user interaction, and hence they often run in the background. Since such processes do not need to be very responsive, they are often penalized by the scheduler [28]. Typical batch programs are programming language compilers, database search engines, and scientific computations.

1.4.2 Interactive Systems:

These systems interact constantly with their users, and therefore spend a lot of time waiting for key presses and mouse operations. When input is received, the process must be woken up quickly, otherwise the user will find the system to be unresponsive. Typically, the average delay must fall between 50 and 150 ms. The variance of such delay must also be bounded, otherwise the user will find the system to be erratic. Typical interactive programs are command shells, text editors, and graphical applications.

1.4.3 Real-Time Systems:

Real-time systems are those systems whose correctness depends not only on logical results of computations, but also on the time at which the results are produced. These systems have very strong scheduling requirements [8]. Such processes should never be blocked by lower-priority processes. These systems should have a short response time and, more importantly response time should have a minimum variance. Typical real-time programs are video and sound applications, robot controllers, and programs that collect data from physical sensors.

1.5 Objectives of the work:

The main objective of my work is to achieve resource optimization and resolving timing constraints in scheduling the task dynamically in real time.

(i) To study the various scheduling algorithms and their applications for real time environment.

(ii) To study the real time systems based upon dynamic scheduling.

(iii) Evaluation of shortest path using dynamic scheduling algorithm with minimum delay and less probability of error for a particular path.

1.6 Organization of the Report

The report consists of five chapters in which the first chapter involves the introduction, types and goals of scheduling. Work done by various researchers, by considering the research paper in static and dynamic scheduling real time task have been included in the second chapter. Third chapter involves introduction of real time system, various real time scheduling algorithms on the basis of different categorization and their comparison. Simulation results obtained for the dynamic scheduling algorithm using the TORA tool is shown in fourth chapter. Finally the fifth chapter involves conclusion and future scope work.

Chapter 2: Literature Survey

This chapter has the survey of the work done by various researchers in the field of scheduling tasks in real time. The possibility of improvement in the work and motivation has been stated clearly. The work has been carried out into two different categories (i) Fixed Priority Scheduling Algorithms in Real Time Tasks (ii) Dynamic Priority Scheduling Algorithms in Real Time Tasks.

2.1 Fixed Priority Scheduling Algorithms in Real Time Tasks

This section shows the work done by the various researchers over a period of time in the field of fixed priority scheduling algorithms in real time tasks scheduling.

C. L. Liu et. al. [1] studied the problem of multiprogramming scheduling on a single processor. They proved that an optimum fixed priority scheduler possesses an upper bound to processor utilization which might be as low as 70% for large task sets. The dynamic deadline driven scheduling algorithm was shown to be globally optimum and capable of achieving full processor utilization. A combination of these two scheduling algorithms was considered which provides the benefits of the deadline driven scheduling algorithm.

C. V. Ramamoorthy et. al. [2] presented an optimal algorithm for scheduling requests on interleaved memories and allowed a finite set of requests to be processed in the minimum expected completion time. The average completion time for servicing a finite set of randomly generated requests was proved to be minimum. Two alternative organizations were investigated (i) A common set of fixed size buffers used to store conflicting requests (ii) Individual fixed size buffers used for each module. These two organizations were shown to be equivalent as far as the average utilization and waiting cycles were concerned. The basic assumptions for the analysis were the dependency effects which were ignored. Moreover, the request rate was also very high, so that, any empty buffers could be filled immediately. The work not includes the nonrandom requests and their performances were not evaluated.

A. F. Bashir et. al. [3] proved the experimental results on the performance of simple level scheduling algorithm for unit time task systems. This efficient linear time algorithm produces an optimal schedule for unit execution time and sense the probability of producing an optimal

schedule. By using this efficient linear time algorithm the probability was at least 0.9 for over 700 cases randomly constructed in their experiment. The results provided additional evidence to support previous empirical studies indicating that the performance of this level algorithm is indeed very good.

Pauline Markenscoff [4] analyzed the operation and performance of a multiple-processor system having shared buses. The model consists of two pipelined tasks in which the first task is partitioned into a number of independent subtasks on separate processors. These processors transmit their output data to the processor executing the second task over shared buses. It was shown that the system having a single shared bus becomes periodic after a number of task executions. Two schemes: (i) Fixed Allocation Bus Scheduling (FABS) scheme and (ii) Non Fixed Allocation Bus Scheduling (NFABS) scheme were proposed for scheduling the data transmission on a multiple-bus system to minimize the corresponding cycle times. A method named as the swapping technique was used in order to schedule the data transmission according to an NFABS scheme. If any of the buses was saturated, the swapping technique provided higher throughput than the FABS scheme.

Almut Burchard et. al. [6] presented optimal scheduling of real-time tasks on multiprocessor systems. For assigning real-time tasks to a multiprocessor system any practical scheduling algorithm presents a trade-off between its computational complexity and its performance. New schedulability conditions were presented for homogeneous multiprocessor systems where individual processor executes the rate-monotonic scheduling algorithm. These conditions were used to develop two task assignment schemes for Multiprocessor systems, named Rate-Monotonic Small Task (RMST) and Rate-Monotonic General Tasks (RMGT) for assigning real-time tasks to processors. For each scheme, they obtained upper bound and lower bound for the average processor utilization. The performance of the new strategies was shown significantly better than earlier strategies.

Alan A. Bertossi et. al. [7] showed that Hard-real-time systems require predictable performance despite the occurrence of failures. Fault tolerance was implemented by using a novel duplication technique where each task scheduled on a processor had either an active backup copy or a passive backup copy scheduled on a different processor. An active copy is always executed, while a passive copy is executed only in the case of a failure. They considered the Rate-Monotonic scheduling algorithm to meet the deadlines of periodic tasks

in the presence of a processor failure. In particular, the Completion Time Test was extended to check the schedulability of a task set including backup copies on a single processor. They extended the well-known Rate-Monotonic First-Fit Assignment algorithm, where all the task copies, including the backup copies, are considered by Rate-Monotonic priority order and assigned to the first processor in which they fit. Remarkable saving of processors was achieved with respect to those needed by the usual active duplication approach in which the schedule of the non-fault-tolerant case is duplicated on two sets of processors.

Sanjoy K. Baruah et. al. [9] presented a schedulability analysis of Rate Monotonic (RM) scheduling algorithm to determine schedulability of periodic task set on a particular uniform multiprocessor. A simple, sufficient test was presented for determining whether a given periodic task system would be successfully scheduled by this algorithm upon a particular uniform multiprocessor platform. This test generalized earlier results concerning RM scheduling upon identical multiprocessor platforms. RM assigns each task a priority inversely proportional to its period, the smaller the period, the higher the priority, with ties broken arbitrarily but in a consistent manner. They also proved that all the deadlines will meet if task set is scheduled using RM algorithm.

Krijn Van Der Raadt et. al. [10] presented multiround algorithms for scheduling divisible loads on star networks. They developed the XMI algorithm, which provides a new closed-form solution for multi installment scheduling on homogeneous star platforms with communication and computation latencies. They also introduced a new multiround algorithm, UMR (Uniform Multi-Round) which sends a fixed amount of work to each worker within each round. This restriction makes it possible to compute a near-optimal number of rounds, which was not possible for previously proposed algorithms. UMR was the first proposed multiround algorithm that was amenable to heterogeneous platforms. Simulation results demonstrated that UMR's ability to compute a near-optimal number of rounds outweighed, on average, the penalty due the restriction it imposes on chunk sizes, when compared with the XMI algorithm.

From the above literature survey of fixed priority scheduling algorithms in real time tasks the following observations have been made:

1. Static scheduling refers to the fact that the scheduling algorithm has complete knowledge regarding the task set and its constraints, such as, deadlines, computation times, precedence constraints, and future release times. This set of assumptions is realistic for many real-time systems.
2. Fixed priority scheduler possesses an upper bound to processor utilization which might be as low as 70 percent for large task sets.
3. In static approach the runtime overhead does not depend on the complexity of the scheduling algorithm. This allows very sophisticated algorithms to be used to solve complex problems or find optimal scheduling sequences.

2.2 Dynamic Scheduling of Real Time Task

This section shows the work done by the various researchers over a period of time in the field of dynamic scheduling of real time task

Wei Z Hao et. al. [11] considered the problem of scheduling a set of pre-emptable tasks in a system. Each task had an arbitrary, determined and worst case processing time based upon a fixed deadline. They also developed algorithms for determining a set of preemptive tasks which were schedulable in a real-time system. This scheduling problem was known to as computationally intensive. In many real-time applications, dynamic tasks are scheduled in order to keep run-time cost low. The computational complexity of their algorithms for scheduling a set of pre-emptable tasks in a system was much lower than that of known optimal algorithms. However the computational overheads introduced by such mechanisms might offset the improve performance.

Don Towsley et. al. [12] considered the problem of analyzing Maximum Laxity (ML) and Earliest Deadline (ED) for systems in which jobs have real-time constraints. They observed that the Markov chains associated with these policies were not good for exact analysis. Hence, they modified the Markov chains in order to develop two families of Markov chains which provide bounds on the performance of ML and ED. The resulting Markov chains were easier to solve numerically. Moreover, the Markov chains that produced the pessimistic bounds on ML(n) and ED(n) where n corresponds to the maximum number of jobs at the head of the queue were scheduled according to the ML and ED rules. The remaining jobs

were treated in a First in First out (FIFO) manner. They observed that ML(3) and ED(3) provide a 5% to 15% improvement in performance over a policy such as FIFO that does not use any laxity or deadline information.

Jia Xu et. al. [15] presented an algorithm which has an optimal schedule on a single processor for a given set of processes where each process start executing after its release time and complete its computation before its deadline. A given set of precedence relations and a given set of exclusion relations defined on ordered pairs of process segments were satisfied. The previously unsolved problem of automated pre-run-time scheduling of processes with arbitrary precedence and exclusion relations was solved.

Krithi Ramamritham et. al. [14] developed scheduling algorithms for real-time multiprocessor systems. Hard real-time systems require both functionally correct executions and results are achieve in a specified time. The tasks are characterized by worst case computation times, deadlines, and resources requirements. The algorithm actively directs the search for a feasible schedule or they help to choose the task that extends the current partial schedule. Two scheduling algorithms were developed (i) this algorithm considers, at each step of the search, the tasks that were yet to be scheduled as candidates (ii) this algorithm focus its attention on a small subset of tasks having the shortest deadlines. It was shown that the second algorithm is very effective when the maximum allowable scheduling overhead is fix. Hence, this algorithm is appropriate for dynamic scheduling in real-time systems

Karsten Schwan et. al. [16] investigated the dynamic scheduling of tasks with well-defined timing constraints for hard real-time systems. They presented a dynamic uniprocessor scheduling algorithm with a $o(nlog(n))$ worst case complexity. The preemptive scheduling performed by algorithm for the tasks which might be related by precedence constraints and might have arbitrary deadlines was of higher efficiency than other known algorithms. An analytic model used for explanation of the experimental results was also presented which validates with actual system measurements. Efficiency was attained by careful design of the algorithms data structures which records the scheduling information. The results proved that the dynamic algorithm might be used for run-time scheduling of a large number of tasks.

Kwang S. Hong et. al. [18] presented an optimal on-line scheduler for a set of real time tasks with one common deadline on *m* processors. It was shown that no optimal on-line scheduler

exist for tasks with two distinct deadlines on m processors. An optimal on-line scheduler was given for situations where processors can go down unexpectedly. They also proposed algorithm B whish was modified to yield an optimal on-line scheduler for the situation where the duration of processor down times was unknown. However, the modified algorithm was not able to detect infeasible task systems until it reaches the common deadline of the non urgent tasks.

Marco Spuri et. al. [19] extended formal results for precedence constrained, real-time scheduling of unit time tasks to arbitrary time tasks with preemption and developed an algorithm for dynamic systems in more real-time system situations. These results were integrated with a well-known protocol which handles real-time scheduling of tasks with shared resources, but don't consider precedence constraints. This results in a practical algorithm for many real-time uniprocessor systems and schedulability formulas for task sets which allows preemption, shared resources, and precedence constraints.

Kang G. Shin et. al. [20] considered the problem of scheduling both periodic and aperiodic tasks in real-time systems. A new algorithm, called Reservation-Based (RB), was proposed for ordering the execution of real-time tasks. This algorithm guarantees all periodic-task deadlines while minimizing the probability of missing aperiodic-task deadlines. In this algorithm the periodic tasks are scheduled according to the Rate Monotonic Priority Algorithm (RMPA) and aperiodic tasks are schedule by utilizing the processor time left unused by periodic tasks in each unit cycle. The RB algorithm outperforms all other scheduling algorithms in meeting aperiodic task deadlines.

Wu Chun Feng et. al. [21] described algorithm for scheduling preemptive, imprecise, composite tasks in real-time. Each composite task consists of a chain of component tasks, and each component task was made up of a mandatory part and an optional part. Whenever a component task use imprecise input, the processing times of its mandatory and optional parts becomes larger. The composite tasks are scheduled by a two-level scheduler. At the high level, the composite tasks are scheduled preemptively on one processor according to an existing algorithm and at the low-level, scheduler distributes the time budgeted for each composite task across its component task to minimize the output error of the composite task.

G. Manimaran et. al. [22] proposed an efficient algorithm for non pre-emptive scheduling of dynamically arriving aperiodic real-time tasks in multiprocessor systems. A real-time task is characterize by its deadline, resource requirements, and worst case computation time. They used parallelism to obtain better schedulability than non parallelizable task scheduling algorithms. The schedulability of a preemptive algorithm is always higher than its non preemptive version. However, the higher schedulability of a preemptive algorithm obtains at the cost of higher scheduling overhead. Parallelizable task scheduling tries to meet the conflicting requirements of high schedulability and low overhead. The success ratio offered by this algorithm was higher than myopic algorithm for a wide variety of task parameters. They proved that the impact of number of backtracks on the success ratio was less significant in comparison to other parameters.

Amit Sinha et. al. [24] analyzed Real-time scheduling on processors which support dynamic voltage and frequency scaling. The Slacked Earliest Deadline First (SEDF) algorithm was proposed and it was shown that the algorithm is optimal in minimizing processor energy consumption and maximum lateness. An upper bound on the processor energy savings was also derived. Real-time scheduling of periodic tasks was also analyzed and optimal voltage and frequency allocation for a given task set was determined that guarantees schedulability and minimizes energy consumption.

Ali Manzak et. al. [26] presented variable voltage task scheduling algorithms which minimize energy when the task arrival time, deadline time, execution time, periods and switching activities are given. They considered aperiodic Earliest Due Date, aperiodic Earliest Deadline First and periodic Rate Monotonic scheduling algorithms. To minimize the energy they used Lagrange multiplier method and determined the relation between the task voltages. It was shown that the voltage assignment obtained by the proposed low-complexity algorithm was very close to that of the optimal energy (0.1% error) and optimal peak power (1% error) assignment. However the application of this procedure increases the complexity of the scheduling algorithms. So, there is a tradeoff between energy savings and the complexity of the algorithms.

Peng Li et. al. [27] presented two fast, best-effort real-time scheduling algorithms named Modified Dependent Activity Scheduling Algorithm (MDASA) and Modified Locke's Best Effort Scheduling Algorithm (MLBESA). MDASA and MLBESA mimic the behavior of the

DASA and LBESA scheduling algorithms, but are faster than DASA and LBESA. The task response time under MDASA and MLBESA is very close to the values under their counterpart scheduling algorithms. Thus, MDASA and MLBESA substituted for DASA and LBESA algorithms, respectively, in adaptive resource allocation techniques for asynchronous real-time distributed systems where DASA and LBESA had previously been serious bottlenecks on computational costs.

Euiseong Seo [29] presented an energy efficient technique for scheduling real time tasks on multicore processors to lower the power consumption and increase the throughput. The two techniques suggested by them were: (i)Dynamic Repartitioning, which tries to keep the performance demands of each core balanced by migrating tasks from the core with the highest demand to the one with the lowest demand. (ii) Dynamic core scaling algorithm, which reduces leakage power consumption by adjusting the number of active cores. Simulation results showed that dynamic repartitioning produces energy savings of about 8% even with the best energy-efficient partitioning algorithm. The results also showed that dynamic core scaling reduces energy consumption by about 26% under low load conditions.

Feng Xiang Zhang et. al. [30] presented schedulability analysis for real time systems with Earliest Deadline First (EDF) scheduling. They proposed new results on necessary and sufficient schedulability analysis for EDF scheduling. The new result reduces the calculation times for schedulable task sets in all situations. For example, a 16-task system that in the previous analysis had to check 858,331 points or deadlines could, with the new analysis, check at just 12 points. There were no restrictions on the new results: each task can be periodic or sporadic, with relative deadline, which can be less than, equal to, or greater than its period, and task parameters could range over many orders of magnitude.

Xian-Bo [23] proposed an improved fuzzy EDF scheduling model based on fuzzy inference which was more suitable for embedded soft real-time systems in an uncertain environment. In this scheduling model, all tasks were periodic and a task's criticality and deadline distance were described with fuzzy set. In this scheduling algorithm, a task's scheduling priority is gotten by looking up the inference rule table with its fuzzy deadline distance and fuzzy criticality patterns. Tasks having shorter fuzzy deadline distance and higher fuzzy criticality are scheduled first. The simulation test showed that proposed scheduling model had less deadline missing ratio than traditional EDF algorithm and the important tasks had less

deadline missing ratio than that of others tasks in an overloaded uncertain embedded soft real-time system.

Wann-Yun Shieh et. al. [32] presented Dynamic Voltage Scaling (DVS) technique for the dual-core real-time systems. Such systems have low power consumption demands. They proposed an energy-efficient task scheduling algorithm and gave two approaches: off-line and on-line. For the off-line approach, they proposed an Integer Linear Programming (ILP) based algorithm to find the optimal scheduling. For the on-line approach, they proposed a heuristic algorithm. They used two decision functions to reduce core's energy consumption and maintain task's performance. The core decision left more time space in one core during scheduling tasks; therefore the tasks get larger time space to execute with low voltage to reduce energy consumption.

A. Burns et. al. [31] suggested partitioned EDF scheduling for multiprocessors using a new task splitting scheme. For m processors at most $(m - 1)$ tasks are split. The first part of a split task is constrained to have a deadline equal to its computation time. The second part of the task then had the maximum time available to complete its execution on a different processor. The advantage of this scheme is that no special run-time mechanisms are required and the overheads are kept to a minimum.

From the above literature survey of dynamic priority scheduling algorithms in real time tasks the following observations have been drawn:

1. A dynamic scheduling algorithm has complete knowledge of the currently active set of tasks, but new arrivals may occur in the future, not known to the algorithm at the time it is scheduling the current set.
2. Dynamic priority scheduling performs the task scheduling during run time and it helps in the processor utilization optimization.
3. Dynamic scheduling algorithm uses dynamic repartitioning and dynamic core scaling algorithm which minimize energy when the task arrival time, deadline time, execution time, periods, and switching activities are known.

From the literature survey the following observations have been drawn:

1. Static scheduling algorithm has complete knowledge regarding the task set and its constraints, such as, deadlines, computation times, precedence constraints, and future release times.
2. Static priority scheduling performs the task scheduling during compilation time. It possesses an upper bound to processor utilization which might be as low as 70 percent because the scheduling is done during compilation time and therefore, if any higher priority task comes processor will have to wait for lower priority task to complete its execution.
3. Dynamic scheduling can perform the task pre-emptively also in which the currently executing process can be stopped and the process with higher priority can start executing. The lower priority task can resume later. The preemptive scheduling performed by algorithm is of higher efficiency than other algorithms.
4. Energy efficient techniques for scheduling real time tasks on multicore processors to lower the power consumption were proposed. Devices can perform better and able to produce better results, if power consumption is low. At the same time, the heat dissipation also reduce, hence, designer has the provision to use ON-chip components.
5. Dynamic priority scheduling performs the task scheduling during run time and it helps in the processor utilization optimization. It can be achieved by dynamic scheduling and it also increases the CPU overheads.

It has been observed that a lot of work has been done on dynamic scheduling, but still there is a scope to design and implement a scheduling algorithm which can reduce the overhead using dynamic scheduling. Furthermore, the algorithm also provides efficient results for dynamic tasks in real time environment.

Chapter 3: Real Time Scheduling Algorithm

Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced. This chapter involves the introduction of real time system, its types, features and various real time scheduling algorithms.

3.1 Real Time Scheduling

When designing an operating system, a programmer must consider which scheduling algorithm will perform best for the use the system is going to see. There is no universal scheduling algorithm, and many operating systems use extended or combinations of the scheduling algorithms. For example, Windows NT/XP/Vista uses a multilevel feedback queue, a combination of Fixed Priority Preemptive Scheduling, Round Robin and First In First Out. In real time system, processes can dynamically increase or decrease in priority depending on whether it has been serviced already, or it is waiting extensively. Every priority level is represented by its own queue, with Round-Robin scheduling among the highest priority processes and FIFO among the lower ones. In this sense, response time is short for most processes and critical system processes get completed very quickly. Since processes can only use single time unit of the round robin in the highest priority queue, starvation can be a problem for longer high priority processes.

3.2 Types of Real Time System Task

Real time systems are the computing systems that must react within the precise time constraints to events in the environment. The efficient scheduling of tasks on these systems becomes necessary in order to receive accurate and timely response. From the work done by various researchers it has been observed that there is still a need to optimize scheduling techniques, so that, the various constraints of real time systems can be met and performance can be enhanced. Depending on the consequences that may occur because of a missed deadline, a real time task can be distinguished in two categories; (i) Soft Real-Time Tasks (ii) Hard Real-Time Tasks

3.2.1 Soft Real-Time Tasks: soft real-time tasks can miss some deadlines and the system could still work correctly [8]. However, missing some deadlines for soft real-time tasks will lead to paying penalties.

3.2.2 Hard Real-Time Tasks: hard real-time tasks cannot miss any deadline, otherwise, undesirable or fatal results will be produced in the system.

3.3 Features of Real-Time Systems

A real-time system must have following features for the efficient working of the system:
(a)Fast Response and Predictability**:** The systems should respond predictably fast to the urgent events so as their deadlines are met. The timely generation of results is essential for the accurate working of the system. (b) High degree of schedulability**:** The timing requirements must be satisfied at high degrees of resource usage [8]. The constraints should be met even if all the resources are being utilized i.e. high degree of schedulability should be achieved.(c) Stability under transient overload: The system must be stable under overloaded conditions which mean that the deadlines of the selected critical tasks must be met even if the system is overloaded and other deadlines could not be met. The system should be robust enough to deal with peak overload conditions. (d) Efficiency: As most real time systems are embedded into small devices with space, weight, memory, power constraints, an efficient management of available resources is essential for achieving a desired performance. (e) Fault Tolerance: The system should not crash on the occurrence of single hardware or software failure. So, the critical components have to be designed to be fault tolerant. (f)Maintainability: Architecture should be in modular form in order to perform easy modifications on the system and ensuring the enhancement of the system.

3.4 Real Time Scheduling Approaches

Scheduling problems for a real-time system is to determine a schedule for the execution of the tasks so that they satisfy their timing constraints. The appropriate scheduling approach needs to be designed which are based on the properties of the system and the tasks occurring in it. These approaches are as follows:

3.4.1 Soft/Hard Real-Time Tasks

Soft Real-Time tasks can miss some deadlines and the system will still work correctly. However, missing some deadlines for soft real-time tasks will lead to paying penalties [8].

Hard real-time tasks cannot miss any deadline, otherwise, undesirable or fatal results will be produced in the system.

3.4.2 Periodic/Aperiodic/Sporadic tasks

Periodic real-time tasks are activated regularly at fixed periods. A majority of sensory processing is periodic in nature. For example, radar which tracks flights produces data at a fixed rate. Aperiodic real-time tasks are activated irregularly at some unknown and possibly unbounded rate. The time constraint is usually a deadline. Sporadic real-time tasks are activated irregularly with some known bounded rate. The bounded rate is characterized by a minimum inter-arrival period which is a minimum interval of time between two successive activations.

3.4.3 Preemptive/Non-preemptive tasks

In some real-time scheduling algorithms, a task can be preempted means if another task of higher priority becomes ready then currently executing process can be stopped and the process with higher priority can start executing [25,8]. In contrast, the execution of a non preemptive task should be completed without interruption once it is started. It means higher priority task will have to wait for lower priority task's execution to start executing.

3.4.4 Off-line/Online.

A scheduling algorithm is used off line if it is executed on the entire task set before tasks activation. The schedule generated in this way is stored in a table and later executed by a dispatcher. A scheduling algorithm is used online if scheduling decisions are taken at run time every time a new task enters the system or when a running task terminates.

3.4.5 Fixed/Dynamic priority tasks

In priority driven scheduling, a priority is assigned to each task. Assigning the priorities can be done statically or dynamically while the system is running [13,8]. The real-time scheduling algorithms are categorized, based on their priority assignment method, into fixed and dynamic priority scheduling algorithms.

3.5 Scheduling Algorithms

There can be many bases on which Scheduling algorithms classified but Scheduling algorithms are mainly classified on two bases (i) On the basis of clock interrupts (ii) On the basis of environment they are being used.

3.5.1 Categorization on the basis of Clock Interrupts:

On the basis of clock interrupts scheduling algorithms can be broadly class into two types: (a) Non Preemptive Scheduling**:** A scheduling algorithm is non preemptive if the execution of a particular process cannot be interrupted [25, 8]. The process once started will complete its execution and other processes will have to wait in the ready queue until currently executing task gets completed. (b) Preemptive Scheduling**:** It is the technique in which we temporarily interrupt the execution of the task and resume its processing at a later stage. This is also regarded as context switch. Preemptive scheduler carry out this task as it has the power to pre-empt a task and resume it later. It lets a pending high-priority job to halt the execution of currently running low priority job by taking away the resources from the low priority job.

3.5.2 Categorization on the Basis of Environment they are being used:

Scheduling algorithm on the basis of environment they are being used are:

3.5.2.1 Round-Robin Scheduling:

Each process gets a small unit of CPU time called time quantum usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue. Ready queue is treated as a circular queue.CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of 1 time quantum. Ready queue is a FIFO queue of processes [13]. New processes are added to the tail of the ready queue and the CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum and dispatches the process. If the process has a CPU burst of less than 1 time quantum, it releases the CPU voluntarily. On the other hand, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed and the process will be put at the tail of the ready queue. The CPU scheduler then picks up the next process in the ready queue. If there are n processes in the ready queue and the time quantum is q, each process gets 1/*n* of the CPU time in chunks of at most q time units at once.

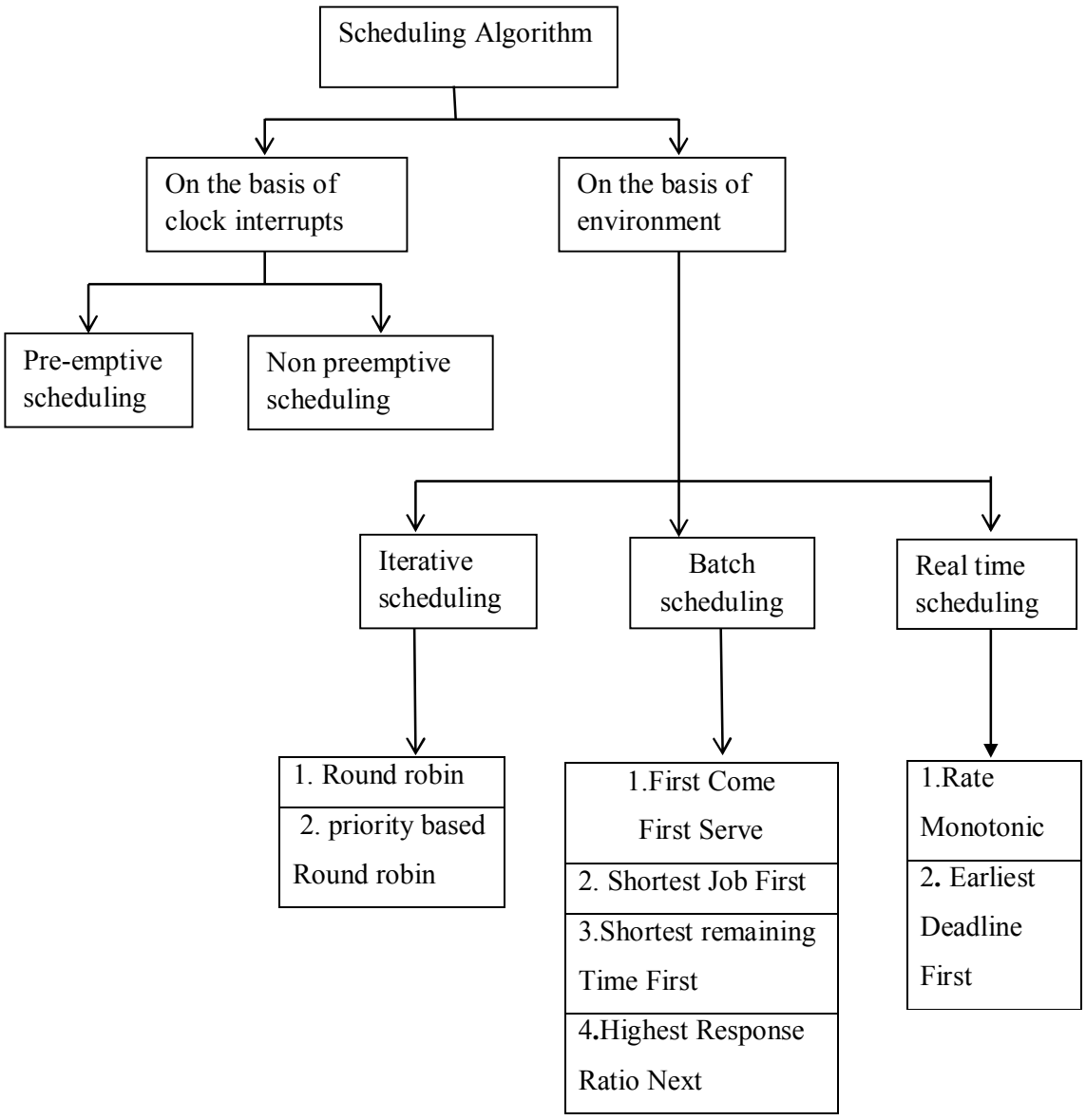

Figure 3.1: Scheduling Algorithms

3.5.2.2 Priority Based Round Robin

This type of scheduling is similar to round robin technique but it allows hierarchy of processes. The processes are assigned to different queues on the basis of some property of the process. Waiting time and turnaround time depends on the priority of the processes [13]. It is simple to implement and have reasonable support for priorities. Starvation is very rare as almost all the privileged processes are blocked for input/output. Scheduling must be done between the queues. Priority Based Round Robin is a fixed priority time slice scheduling where each queue gets a certain amount of CPU time which it can schedule among its processes.

3.5.2.3 First Come First Served Scheduling

In this type of scheduling the process that request the CPU first will get a share of the CPU first. First In First Out (FIFO) scheme is a non-preemptive. It is fair in the human sense of fairness but unfair in the sense that long jobs make short jobs wait or important jobs might get held up because of unimportant jobs [13].

3.5.2.4 Shortest-Job First Scheduling (SJF)

In this method, the processor is assigned to the process with the smallest execution time. This requires the knowledge of execution time. In our examples, it is given as a table but actually these burst times are not known to the OS. So it makes prediction [17]. One approach for this prediction is using the previous processor burst times for the processes in the ready queue. The algorithm selects the shortest predicted next processor burst time.

3.5.2.6 Multilevel Feedback Queue Scheduling

A process can move between the various queues; aging can be implemented this way. This algorithm aims to minimize turnaround time along with reducing the response time. The idea is to separate processes with different CPU-burst characteristics. In this technique there are multiple queues with different priorities. The job in the higher priority queue is executed first and the jobs within the same queue are scheduled using round robin algorithm [25]. The priority is varied on the basis of observed behavior. A new process in the system is placed in the highest order priority queue and if uses entire time interval during execution then the priority of that process is reduced and it is dropped by level in the priority queues. If it completes its execution before the end of time interval then it remains at the same priority level. First come first server (FCFS) technique is used in all the queues but the bottom level

queue or the lowest priority queue uses Round robin technique. I/O-bound processes will stay in higher priority queue

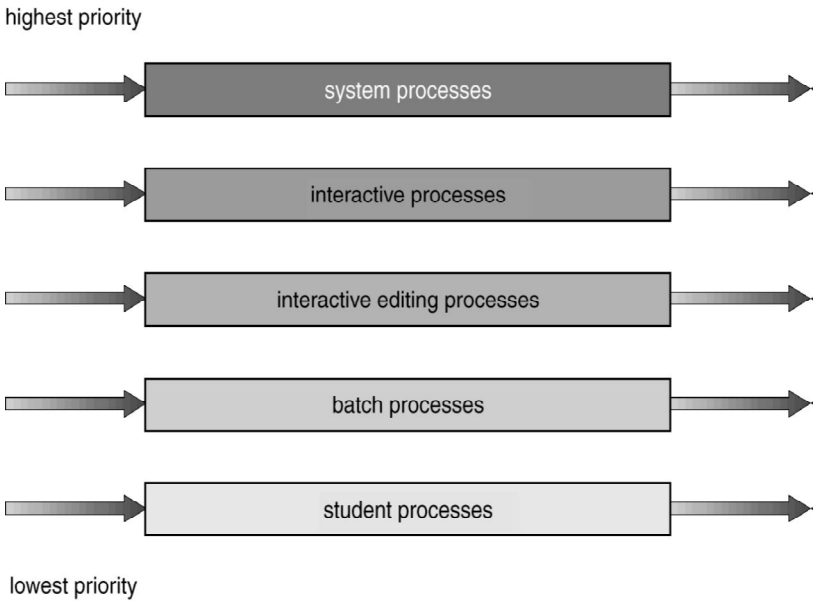

Figure 3.2: Multilevel Feedback Queue Scheduling

3.5.2.7 Highest Response Ratio Next

This technique was developed by Brinch J Hansen to correct certain weaknesses in Shortest Job First algorithm including the difficulty in estimating the run time. It is a priority based non-preemptive technique in which priority is dependent on the estimated run time and waiting time of the process and processes cannot be interrupted in between their execution. Jobs with longer waiting time get higher priority and hence the problem of starvation is removed.

3.5.2.8 Rate Monotonic Scheduling algorithm (RMS)

It is a dynamic preemptive algorithm for scheduling set of independent hard real time tasks. This was published in 1973 by Liu and Layland [8]. The algorithm was based on static task priorities. The assumptions made about the task are (a) the request for all the task sets, for which hard deadlines should be met, is periodic. (b) All tasks are independent of each other. No precedence constraints or mutual exclusion constraints exist between any pair of tasks. (c) The deadline interval of every task is equal to its period. (d) The required maximum computation time is known beforehand and is constant. (e) Time required for context switching can be ignored.

Rate monotonic scheduling algorithm assigns the priority of each task according to its period, so that the shorter the period the higher the priority. If a task set cannot be scheduled using the RMS algorithm, it cannot be scheduled using any fixed-priority algorithm. One major limitation of fixed-priority scheduling is that it is not always possible to fully utilize the CPU. Even though RMS is the optimal fixed-priority scheme, it has a worst-case schedule bound

$$Wn = n(2^{(1/n)} - 1)$$

where n is the number of tasks in a system.

As we would expect, the worst-case schedulable bound for one task is 100%. But, as the number of tasks increases, the schedulable bound decreases.

3.5.2.9 Deadline Monotonic (DM)

This technique is an extension of Rate Monotonic scheduling algorithm. This was first proposed in 1982 by Leung and Whiteland. This is also fully preemptive technique used for scheduling tasks with static priorities. The third assumption mentioned in rate monotonic technique that says the deadline interval of every task is same and equal to its period; has been relaxed. The tasks have constrained deadlines i.e. relative deadlines can be less than or equal to its period. Each task is assigned a fixed priority inversely proportional to its relative deadline. So, at any instant task with the shortest deadline is executed. As relative deadlines are constant, DM is a static priority assignment technique.

3.5.2.10 Earliest Deadline First algorithm (EDF)

It is the optimal dynamic pre-emptive algorithm for single processor systems which are based on dynamic priorities. The task with the earliest deadline is given the highest priority [8]. That is why this algorithm is also known as Deadline Driven Scheduling Algorithm (DDSA). The jobs in the task set are put in the ready queue on the basis of their priorities. The priorities of all the jobs in the ready queue are fixed. So, this algorithm is a job level fixed-priority algorithm. A task set can be scheduled by this algorithm if the utilization factor is less than 1.

3.6 Comparison of various scheduling algorithms

Comparison of various scheduling algorithms is shown as follows:

Scheduling Algorithm	**CPU Overhead**	**Throughput**	**Turnaround Time**
First In First Out	Low	Low	High
Shortest Job First	Medium	High	Medium
Round-robin scheduling	High	Medium	Medium
Priority based scheduling	Medium	Low	High
Multilevel Queue Scheduling	High	High	Medium

Table 3.5: Comparison of various scheduling algorithms

Chapter 4: Results and Discussion

From the literature survey, it has been observed that static and dynamic scheduling techniques can be used to schedule the tasks in a communication network where we find the shortest path from each node to every other node. Shortest paths from each node to every other node have been found using the TORA software and the Simulation result which includes delay and the intermediate nodes between two nodes during scheduling have also been obtained. 19 nodes have been taken for the dynamic scheduling of the following network as shown in figure 4.1:

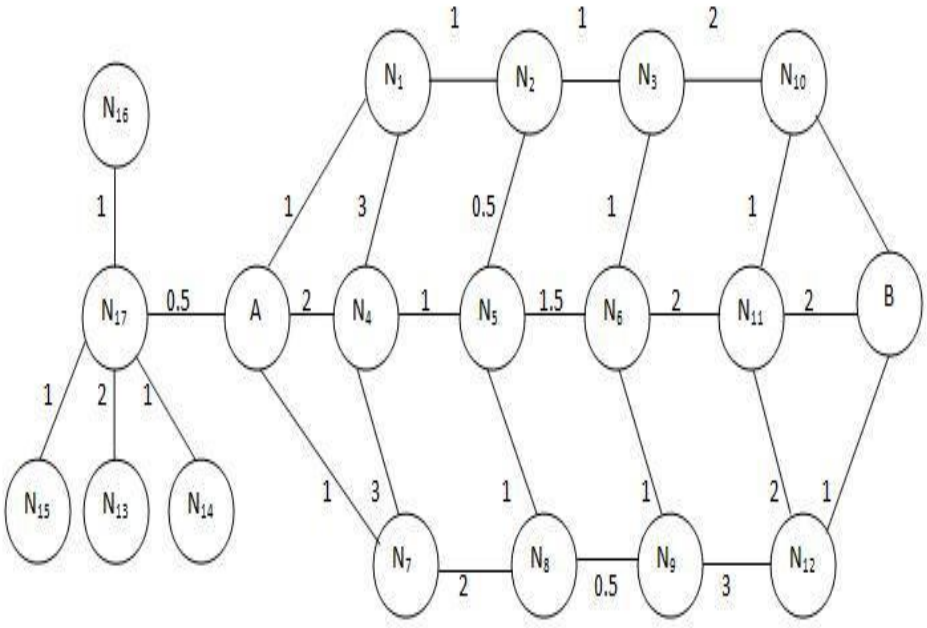

Figure 4.1: Network used for dynamic scheduling

Simulation results have been obtained by considering two cases: the nodes $n4$ and n12 are the failed nodes in case 1 and case 2 respectively. These are the critical nodes because maximum amount of time is required to transport the data from source to destination via n4 and n12.

Case 1: when node N4 assume to be failed or disconnected

The network has been critically analyzed and the shortest path for node $n1$ to all other nodes has been achieved. The table 4.1 shows the name of the destination node, the various paths from source node $n1$ to destination node, the delay along the path and the number of intermediate nodes.

S. NO.	Destination Node	Paths	Delay(ns)	Number of Intermediate Node
1.	N2	N1 $\xrightarrow{1}$ N2	1	0
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{1}$ N5 $\xrightarrow{0.5}$ N2	5.5	4
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{0.5}$ N9 $\xrightarrow{1}$ N6 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N2	7.5	6
2.	N3	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3	2	1
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{1}$ N3	4	3
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\longrightarrow$ N9 $\xrightarrow{1}$ N6 $\xrightarrow{1}$ N3	6.5	5
3.	N4	infinity	-	
4.	N5	N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5	1.5	1
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{1}$ N5	5.0	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{1.5}$ N5	4.5	3
5.	N6	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6	3	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6	3	2
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{0.5}$ N9 $\xrightarrow{1}$ N6	5.5	4
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10 $\xrightarrow{1}$ N11 $\xrightarrow{2}$ N6	7	4
6.	N7	N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7	2	1
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.0}$ N8 $\xrightarrow{2}$ N7	4.5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{1}$ N9 $\xrightarrow{0.5}$ N8 $\xrightarrow{2}$ N7	6.5	5
7.	N8	N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8	4	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.0}$ N8	2.5	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1.0}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{1.0}$ N9 $\xrightarrow{0.5}$ N8	4.5	4

8.	N9	N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{0.5}$ N9	4.5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1.0}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{1.0}$ N9	4	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1}$ N8 $\xrightarrow{0.5}$ N9	3	3
9.	N10	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10	4	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{2}$ N11 $\xrightarrow{1}$ N10	6	4
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{2}$ N11 $\xrightarrow{1}$ N10	6	4
10.	N11	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10 $\xrightarrow{1}$ N11	5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{2}$ N11	5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{2}$ N11	5	3
11.	N12	N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{0.5}$ N9 $\xrightarrow{3}$ N12	7.5	4
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1}$ N8 $\xrightarrow{0.5}$ N9 $\xrightarrow{3}$ N12	6	4
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{1.0}$ N9 $\xrightarrow{3}$ N12	7	4
12.	N13	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{2}$ N13	3.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{2}$ N13	7.5	3
13.	N14	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N14	2.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{2}$ N14	6.5	3
14.	N15	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N15	2.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N15	6.5	3
15.	N16	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N16	2.5	2
16.	N17	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17	1.5	1

17.	A	N1 $\xrightarrow{1}$ A	1	0
18.	B	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10 $\xrightarrow{1}$ B	5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{2}$ N11 $\xrightarrow{2}$ B	7	4

Table 4.1: various paths from source node N1 to destination nodes when N4 failed

Similarly, shortest paths from the remaining nodes have been found. By knowing the number of intermediate nodes and the shortest paths between all the nodes in the network, the path between any two nodes has been optimized. The subsequent section shows the optimized paths from every node to all other nodes when node $n4$ and critical node $n12$ get failed and also contains the calculation of probability of error in case of failure of any one of the intermediate node.

For node $n1$, there are various paths to the other nodes, but from all the possible paths we need that path in which there is minimum delay or the intermediate nodes are minimum for the effective transmission of data.

From	To	Distance	Route
1-N1	2-N2	1.0	1-2
1-N1	3-N3	2.0	1-2-3
1-N1	4-N4	infinity	
1-N1	5-N5	1.5	1-2-5
1-N1	6-N6	3.0	1-2-3-6
1-N1	7-N7	2.0	1-18-7
1-N1	8-N8	2.5	1-2-5-8
1-N1	9-N9	3.0	1-2-5-8-9
1-N1	10-N10	4.0	1-2-3-10
1-N1	11-N11	5.0	1-2-3-6-11
1-N1	12-N12	6.0	1-2-5-8-9-12
1-N1	13-N13	3.5	1-18-17-13
1-N1	14-N14	2.5	1-18-17-14
1-N1	15-N15	2.5	1-18-17-15
1-N1	16-N16	2.5	1-18-17-16
1-N1	17-N17	1.5	1-18-17
1-N1	18-A	1.0	1-18
1-N1	19-B	5.0	1-2-3-10-19

Figure 4.2: Shortest paths from source node $n1$ to other nodes

It has been observed from the figure 4.2 that node $n1$ to destination node $n4$ shows infinite delay because the destination node $n4$ is failed. From the results obtained above it is very clear that the transmission of data from $n1$ to node $n12$ takes maximum time i.e. $6ns$ and has 4 intermediate nodes. If another path is taken from $n1$ to node $n12$, the time taken has been found greater than or equal to $6ns$. If we choose other path ($n1$ A $n7$ $n8$ $n9$ $n12$), the data will take 7.5 ns but the number of intermediate nodes remains 4. There is no change in the number of intermediate nodes but the time require to reach node $n12$ from $n1$ reduces by, 1.5/7.5×100=20.00%. Considering equal probability of failure of all nodes to be 1/10, the probability of failure in optimized path which has 4 intermediate nodes from node $n1$ to node $n12$, is given by

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure of the schedule comes out to be 0.34.

Similarly for node $n2$, there are various paths to the other nodes and from all the possible paths we need that path in which either delay or the intermediate nodes are minimum for the effective transmission of data.

From	To	Distance	Route
2-N2	1-N1	1.0	2-1
2-N2	3-N3	1.0	2-3
2-N2	4-N4	infinity	
2-N2	5-N5	0.5	2-5
2-N2	6-N6	2.0	2-3-6
2-N2	7-N7	3.0	2-1-18-7
2-N2	8-N8	1.5	2-5-8
2-N2	9-N9	2.0	2-5-8-9
2-N2	10-N10	3.0	2-3-10
2-N2	11-N11	4.0	2-3-6-11
2-N2	12-N12	5.0	2-5-8-9-12
2-N2	13-N13	4.5	2-1-18-17-13
2-N2	14-N14	3.5	2-1-18-17-14
2-N2	15-N15	3.5	2-1-18-17-15
2-N2	16-N16	3.5	2-1-18-17-16
2-N2	17-N17	2.5	2-1-18-17
2-N2	18-A	2.0	2-1-18
2-N2	19-B	4.0	2-3-10-19

Figure 4.3: Shortest paths from source node $n2$ to other nodes

It has been observed from the figure 4.3 that node $n2$ to destination node $n4$ shows infinite delay because the destination node $n4$ is failed. From the results obtained above it is very clear that the transmission of data from node $n2$ to node $n12$ takes maximum time i.e. $5ns$ and has 3 intermediate nodes. If we choose other path ($n2$ $n1$ A $n7$ $n8$ $n9$ $n12$), the data will take $8.5ns$ to reach $n12$ from node $n2$ and intermediate nodes increases to 5. Hence, the time needed is reduced by, 3.5/8.5×100=41.18%. There is no change in the number of nodes traversed in order to reach node $n12$ from $n2$ but the time require to reach that node reduces by 41.18%. The probability of failure of the schedule in the optimized path which has 3 intermediate nodes from node $n2$ to node $n12$, is given by

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

Here n=3, so the probability of failure of the schedule in the worst case comes out to be 0.27 approximately and this is lesser than the probability of failure on the route from node $n1$ node $n12$, as observed in previous case.

Similarly for node $n3$, there are various paths to the other nodes and from all the possible paths we need that path in which either delay or the intermediate nodes are minimum for the effective transmission of data.

From	To	Distance	Route
3-N3	1-N1	2.0	3-2-1
3-N3	2-N2	1.0	3-2
3-N3	4-N4	infinity	
3-N3	5-N5	1.5	3-2-5
3-N3	6-N6	1.0	3-6
3-N3	7-N7	4.0	3-2-1-18-7
3-N3	8-N8	2.5	3-2-5-8
3-N3	9-N9	2.0	3-6-9
3-N3	10-N10	2.0	3-10
3-N3	11-N11	3.0	3-6-11
3-N3	12-N12	4.0	3-10-19-12
3-N3	13-N13	5.5	3-2-1-18-17-13
3-N3	14-N14	4.5	3-2-1-18-17-14
3-N3	15-N15	4.5	3-2-1-18-17-15
3-N3	16-N16	4.5	3-2-1-18-17-16
3-N3	17-N17	3.5	3-2-1-18-17
3-N3	18-A	3.0	3-2-1-18
3-N3	19-B	3.0	3-10-19

Figure 4.4: Shortest paths from source node $n3$ to other nodes

It has been observed from the figure 4.4 that the path from node $n3$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n3$ to node $n13$ takes maximum time i.e. $5.5ns$ and has 4 intermediate nodes. If we choose other path ($n3$ $n6$ $n9$ $n8$ $n7$ A $n17$ $n13$), the data will take $8ns$ to reach node $n13$ from $n3$ and intermediate nodes increases to 6. Hence, the time require to reach node $n13$ from $n3$ reduces by, 2.5/8×100=31.25%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n3$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure of the schedule comes out to be 0.34.

For node $n4$, there is no path to the other nodes because node $n4$ has been failed and hence scheduling for this node is not possible. In other words transmission of data is not possible from node $n4$.

From	To	Distance	Route
4-N4	1-N1	infinity	
4-N4	2-N2	infinity	
4-N4	3-N3	infinity	
4-N4	5-N5	infinity	
4-N4	6-N6	infinity	
4-N4	7-N7	infinity	
4-N4	8-N8	infinity	
4-N4	9-N9	infinity	
4-N4	10-N10	infinity	
4-N4	11-N11	infinity	
4-N4	12-N12	infinity	
4-N4	13-N13	infinity	
4-N4	14-N14	infinity	
4-N4	15-N15	infinity	
4-N4	16-N16	infinity	
4-N4	17-N17	infinity	
4-N4	18-A	infinity	
4-N4	19-B	infinity	

Figure 4.5: No Shortest path from node $n4$ to other nodes

Since node $n4$ has been failed so all the distances obtained from the simulation result comes out to be infinity.

For node $n5$, there are various paths to the other nodes and for the effective transmission of data from all the possible paths we need that path in which either delay or the intermediate nodes is minimum.

From	To	Distance	Route
5-N5	1-N1	1.5	5-2-1
5-N5	2-N2	0.5	5-2
5-N5	3-N3	1.5	5-2-3
5-N5	4-N4	infinity	
5-N5	6-N6	1.5	5-6
5-N5	7-N7	3.0	5-8-7
5-N5	8-N8	1.0	5-8
5-N5	9-N9	1.5	5-8-9
5-N5	10-N10	3.5	5-2-3-10
5-N5	11-N11	3.5	5-6-11
5-N5	12-N12	4.5	5-8-9-12
5-N5	13-N13	5.0	5-2-1-18-17-13
5-N5	14-N14	4.0	5-2-1-18-17-14
5-N5	15-N15	4.0	5-2-1-18-17-15
5-N5	16-N16	4.0	5-2-1-18-17-16
5-N5	17-N17	3.0	5-2-1-18-17
5-N5	18-A	2.5	5-2-1-18
5-N5	19-B	4.5	5-2-3-10-19

Figure 4.6: Shortest paths from node $n5$ to all other nodes

It has been observed from the figure 4.6 that the path from node $n5$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n5$ to node $n13$ takes maximum time i.e. $5ns$ and has 4 intermediate nodes. If we choose other path ($n5$ $n8$ $n7$ A $n17$ $n13$), the data will take time $6.5ns$ to reach node $n13$ from node $n5$ but the number of intermediate nodes remains 4. Hence, the time require to reach node $n13$ from node $n5$ reduces by, 1.5/6.5×100=27.27%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n5$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure of the schedule comes out to be 0.34.

For node $n6$, there are various paths to the other nodes and for effective transmission of data from all the possible paths we need that path in which either delay or the intermediate nodes is minimum.

From	To	Distance	Route
6-N6	1-N1	3.0	6-3-2-1
6-N6	2-N2	2.0	6-3-2
6-N6	3-N3	1.0	6-3
6-N6	4-N4	infinity	
6-N6	5-N5	1.5	6-5
6-N6	7-N7	3.5	6-9-8-7
6-N6	8-N8	1.5	6-9-8
6-N6	9-N9	1.0	6-9
6-N6	10-N10	3.0	6-3-10
6-N6	11-N11	2.0	6-11
6-N6	12-N12	4.0	6-9-12
6-N6	13-N13	6.5	6-3-2-1-18-17-13
6-N6	14-N14	5.5	6-3-2-1-18-17-14
6-N6	15-N15	5.5	6-3-2-1-18-17-15
6-N6	16-N16	5.5	6-3-2-1-18-17-16
6-N6	17-N17	4.5	6-3-2-1-18-17
6-N6	18-A	4.0	6-3-2-1-18
6-N6	19-B	4.0	6-3-10-19

Figure 4.7: Shortest paths from node $n6$ to all other nodes

It has been observed from the figure 4.7 that the path from node $n6$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n6$ to node $n13$ takes maximum time i.e. $6.5ns$ and has 5 intermediate nodes. If we choose other path ($n6\ n5\ n8\ n7\ A\ n17\ n13$), the data will take $8ns$ to reach node $n13$ from node $n6$ but the number of intermediate nodes remains 5. Hence the time require to reach node $n13$ from node $n6$ reduces by, 1.5/8×100=18.75%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n6$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure of the schedule comes out to be 0.4.

For node $n7$, there are various paths to the other nodes and for effective transmission of data from all the possible paths we need that path in which either delay or the intermediate nodes is minimum.

From	To	Distance	Route
7-N7	1-N1	2.0	7-18-1
7-N7	2-N2	3.0	7-18-1-2
7-N7	3-N3	4.0	7-18-1-2-3
7-N7	4-N4	infinity	
7-N7	5-N5	3.0	7-8-5
7-N7	6-N6	3.5	7-8-9-6
7-N7	8-N8	2.0	7-8
7-N7	9-N9	2.5	7-8-9
7-N7	10-N10	6.0	7-18-1-2-3-10
7-N7	11-N11	5.5	7-8-9-6-11
7-N7	12-N12	5.5	7-8-9-12
7-N7	13-N13	3.5	7-18-17-13
7-N7	14-N14	2.5	7-18-17-14
7-N7	15-N15	2.5	7-18-17-15
7-N7	16-N16	2.5	7-18-17-16
7-N7	17-N17	1.5	7-18-17
7-N7	18-A	1.0	7-18
7-N7	19-B	6.5	7-8-9-12-19

Figure 4.8: Shortest paths from node $n7$ to all other nodes

It has been observed from the figure 4.8 that the path from node $n7$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n7$ to node B takes maximum time i.e. $6.5ns$ and has 3 intermediate nodes. If we choose other path ($n7\ n8\ n5\ n6\ n11\ B$), the data will take $8.5ns$ to reach node B from node $n7$ and the number of intermediate nodes increases to 4. Hence, the time require to reach node B from node $n7$ reduces by, 2/8.5×100=23.53%. The probability of failure of the schedule in the optimized path which has 3 intermediate nodes from node $n7$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=3, so the probability of failure of the schedule comes out to be 0.27.

For node $n8$, there are various paths to the other nodes for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
8-N8	1-N1	2.5	8-5-2-1
8-N8	2-N2	1.5	8-5-2
8-N8	3-N3	2.5	8-5-2-3
8-N8	4-N4	infinity	
8-N8	5-N5	1.0	8-5
8-N8	6-N6	1.5	8-9-6
8-N8	7-N7	2.0	8-7
8-N8	9-N9	0.5	8-9
8-N8	10-N10	4.5	8-5-2-3-10
8-N8	11-N11	3.5	8-9-6-11
8-N8	12-N12	3.5	8-9-12
8-N8	13-N13	5.5	8-7-18-17-13
8-N8	14-N14	4.5	8-7-18-17-14
8-N8	15-N15	4.5	8-7-18-17-15
8-N8	16-N16	4.5	8-7-18-17-16
8-N8	17-N17	3.5	8-7-18-17
8-N8	18-A	3.0	8-7-18
8-N8	19-B	4.5	8-9-12-19

Figure 4.9: Shortest paths from node $n8$ to all other nodes

It has been observed from the figure 4.9 that the path from node $n8$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n8$ to node $n13$ takes maximum time i.e. $5.5ns$ and has 3 intermediate nodes. If we choose other path ($n8$ $n5$ $n2$ $n1$ A $n17$ $n13$), the data will take $6ns$ to reach node $n13$ from node $n8$ and the number of intermediate nodes increases to 5. Hence, the time require to reach node $n13$ from node $n8$ reduces by, 0.5/6×100=8.33%. The probability of failure of the schedule in the optimized path which has 3 intermediate nodes from node $n8$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=3, so the probability of failure of the schedule comes out to be 0.27.

For node $n9$, there are various paths to the other nodes for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
9-N9	1-N1	3.0	9-8-5-2-1
9-N9	2-N2	2.0	9-8-5-2
9-N9	3-N3	2.0	9-6-3
9-N9	4-N4	infinity	
9-N9	5-N5	1.5	9-8-5
9-N9	6-N6	1.0	9-6
9-N9	7-N7	2.5	9-8-7
9-N9	8-N8	0.5	9-8
9-N9	10-N10	4.0	9-6-3-10
9-N9	11-N11	3.0	9-6-11
9-N9	12-N12	3.0	9-12
9-N9	13-N13	6.0	9-8-7-18-17-13
9-N9	14-N14	5.0	9-8-7-18-17-14
9-N9	15-N15	5.0	9-8-7-18-17-15
9-N9	16-N16	5.0	9-8-7-18-17-16
9-N9	17-N17	4.0	9-8-7-18-17
9-N9	18-A	3.5	9-8-7-18
9-N9	19-B	4.0	9-12-19

Figure 4.10: Shortest paths from node $n9$ to all other nodes

It has been observed from the figure 4.10 that the path from node $n9$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n9$ to node $n13$ takes maximum time i.e. $6ns$ and has 4 intermediate nodes. If some choose other path ($n9$ $n8$ $n5$ $n2$ $n1$ A $n17$ $n13$), the data will take $6.5ns$ to reach node $n13$ from node $n9$ and the number of intermediate nodes increases to 6. Hence, the time require to reach node $n13$ from node $n9$ reduces by, 0.5/6.5×100=7.69%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n9$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

For node $n10$, there are various paths to the other nodes for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
10-N10	1-N1	4.0	10-3-2-1
10-N10	2-N2	3.0	10-3-2
10-N10	3-N3	2.0	10-3
10-N10	4-N4	infinity	
10-N10	5-N5	3.5	10-3-2-5
10-N10	6-N6	3.0	10-3-6
10-N10	7-N7	6.0	10-3-2-1-18-7
10-N10	8-N8	4.5	10-3-2-5-8
10-N10	9-N9	4.0	10-3-6-9
10-N10	11-N11	1.0	10-11
10-N10	12-N12	2.0	10-19-12
10-N10	13-N13	7.5	10-3-2-1-18-17-13
10-N10	14-N14	6.5	10-3-2-1-18-17-14
10-N10	15-N15	6.5	10-3-2-1-18-17-15
10-N10	16-N16	6.5	10-3-2-1-18-17-16
10-N10	17-N17	5.5	10-3-2-1-18-17
10-N10	18-A	5.0	10-3-2-1-18
10-N10	19-B	1.0	10-19

Figure 4.11: Shortest paths from node $n10$ to all other nodes

It has been observed from the figure 4.11 that the path from node $n10$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n10$ to node $n13$ takes maximum time i.e. $7.5ns$ and has 5 intermediate nodes. If we choose other path ($n10\ n11\ n6\ n5\ n8\ n7\ A\ n17\ n13$), the data will take $11ns$ to reach node $n13$ from node $n10$ and the number of intermediate nodes increases to 6. Hence, the time require to reach node $n13$ from node $n10$ reduces by, 3.5/11×100=31.82%.The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n10$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation result of shortest paths from source node $n11$ to other nodes is

From	To	Distance	Route
11-N11	1-N1	5.0	11-6-3-2-1
11-N11	2-N2	4.0	11-6-3-2
11-N11	3-N3	3.0	11-6-3
11-N11	4-N4	infinity	
11-N11	5-N5	3.5	11-6-5
11-N11	6-N6	2.0	11-6
11-N11	7-N7	5.5	11-6-9-8-7
11-N11	8-N8	3.5	11-6-9-8
11-N11	9-N9	3.0	11-6-9
11-N11	10-N10	1.0	11-10
11-N11	12-N12	2.0	11-12
11-N11	13-N13	8.5	11-6-3-2-1-18-17-13
11-N11	14-N14	7.5	11-6-3-2-1-18-17-14
11-N11	15-N15	7.5	11-6-3-2-1-18-17-15
11-N11	16-N16	7.5	11-6-3-2-1-18-17-16
11-N11	17-N17	6.5	11-6-3-2-1-18-17
11-N11	18-A	6.0	11-6-3-2-1-18
11-N11	19-B	2.0	11-19

Figure 4.12: Shortest paths from node $n11$ to all other nodes

It has been observed from the figure 4.12 that the path from node $n11$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n11$ to node $n13$ takes maximum amount of time i.e. $8.5ns$ and has 6 intermediate nodes. If we choose other path ($n11\ n6\ n5\ n8\ n7\ A\ n17\ n13$), the data will take $10ns$ to reach node $n13$ from node $n11$ but the number of intermediate nodes remain 6. Hence, the time require to reach node $n13$ from node $n11$ reduces by, 1.5/10×100=15.00%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node $n11$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Simulation result of shortest paths from source node $n12$ to other nodes is

Figure 4.13: Shortest paths from node $n12$ to all other nodes

From	To	Distance	Route
12-N12	1-N1	6.0	12-9-8-5-2-1
12-N12	2-N2	5.0	12-9-8-5-2
12-N12	3-N3	4.0	12-19-10-3
12-N12	4-N4	infinity	
12-N12	5-N5	4.5	12-9-8-5
12-N12	6-N6	4.0	12-9-6
12-N12	7-N7	5.5	12-9-8-7
12-N12	8-N8	3.5	12-9-8
12-N12	9-N9	3.0	12-9
12-N12	10-N10	2.0	12-19-10
12-N12	11-N11	2.0	12-11
12-N12	13-N13	9.0	12-9-8-7-18-17-13
12-N12	14-N14	8.0	12-9-8-7-18-17-14
12-N12	15-N15	8.0	12-9-8-7-18-17-15
12-N12	16-N16	8.0	12-9-8-7-18-17-16
12-N12	17-N17	7.0	12-9-8-7-18-17
12-N12	18-A	6.5	12-9-8-7-18
12-N12	19-B	1.0	12-19

It has been observed from the figure 4.13 that the path from node $n12$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n12$ to node $n13$ takes maximum amount of time i.e. $9ns$ and has 5 intermediate nodes. If we choose other path ($n12$ $n11$ $n6$ $n5$ $n8$ $n7$ A $n17$ $n13$), the data will take $12ns$ to reach node $n13$ from node $n12$ and the number of intermediate nodes increase to 6. Hence, the time require to reach node $n13$ from node $n12$ reduces by, 3/12×100=25%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n12$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation result of shortest paths from source node $n13$ to other nodes is

From	To	Distance	Route
13-N13	1-N1	3.5	13-17-18-1
13-N13	2-N2	4.5	13-17-18-1-2
13-N13	3-N3	5.5	13-17-18-1-2-3
13-N13	4-N4	infinity	
13-N13	5-N5	5.0	13-17-18-1-2-5
13-N13	6-N6	6.5	13-17-18-1-2-3-6
13-N13	7-N7	3.5	13-17-18-7
13-N13	8-N8	5.5	13-17-18-7-8
13-N13	9-N9	6.0	13-17-18-7-8-9
13-N13	10-N10	7.5	13-17-18-1-2-3-10
13-N13	11-N11	8.5	13-17-18-1-2-3-6-11
13-N13	12-N12	9.0	13-17-18-7-8-9-12
13-N13	14-N14	3.0	13-17-14
13-N13	15-N15	3.0	13-17-15
13-N13	16-N16	3.0	13-17-16
13-N13	17-N17	2.0	13-17
13-N13	18-A	2.5	13-17-18
13-N13	19-B	8.5	13-17-18-1-2-3-10-19

Figure 4.14: Shortest paths from node $n13$ to all other nodes

It has been observed from the figure 4.14 that the path from node $n13$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n13$ to node $n12$ takes maximum amount of time i.e. $9ns$ and has 5 intermediate nodes. If we choose other path ($n13\ n17\ A\ n1\ n2\ n3\ n10\ n11\ n12$), the data will take $10.5ns$ to reach node $n12$ from node $n13$ and the number of intermediate nodes increase to 7. Hence, the time require to reach node $n12$ from node $n13$ reduces by, 1.5/10.5×100=14.28%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n13$ to node $n12$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation results of shortest paths from source node $n14$ to other nodes is

From	To	Distance	Route
14-N14	1-N1	2.5	14-17-18-1
14-N14	2-N2	3.5	14-17-18-1-2
14-N14	3-N3	4.5	14-17-18-1-2-3
14-N14	4-N4	infinity	
14-N14	5-N5	4.0	14-17-18-1-2-5
14-N14	6-N6	5.5	14-17-18-1-2-3-6
14-N14	7-N7	2.5	14-17-18-7
14-N14	8-N8	4.5	14-17-18-7-8
14-N14	9-N9	5.0	14-17-18-7-8-9
14-N14	10-N10	6.5	14-17-18-1-2-3-10
14-N14	11-N11	7.5	14-17-18-1-2-3-6-11
14-N14	12-N12	8.0	14-17-18-7-8-9-12
14-N14	13-N13	3.0	14-17-13
14-N14	15-N15	2.0	14-17-15
14-N14	16-N16	2.0	14-17-16
14-N14	17-N17	1.0	14-17
14-N14	18-A	1.5	14-17-18
14-N14	19-B	7.5	14-17-18-1-2-3-10-19

Figure 4.15: Shortest paths from node $n14$ to all other nodes

It has been observed from the figure 4.15 that the path from node $n14$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n14$ to node $n12$ takes maximum amount of time i.e. $8ns$ and has 5 intermediate nodes. If we choose other path ($n14\ n17\ A\ n1\ n2\ n3\ n10\ n11\ n12$), the data will take $9.5ns$ to reach node $n12$ from node $n14$ and the number of intermediate nodes increase to 7. Hence, the time require to reach node $n12$ from node $n14$ reduces by, 1.5/9.5×100=15.79%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n14$ to node $n12$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

In this case n=5, so the probability of failure comes out to be 0.4.

For node $n15$, there are various paths to the other nodes for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
15-N15	1-N1	2.5	15-17-18-1
15-N15	2-N2	3.5	15-17-18-1-2
15-N15	3-N3	4.5	15-17-18-1-2-3
15-N15	4-N4	infinity	
15-N15	5-N5	4.0	15-17-18-1-2-5
15-N15	6-N6	5.5	15-17-18-1-2-3-6
15-N15	7-N7	2.5	15-17-18-7
15-N15	8-N8	4.5	15-17-18-7-8
15-N15	9-N9	5.0	15-17-18-7-8-9
15-N15	10-N10	6.5	15-17-18-1-2-3-10
15-N15	11-N11	7.5	15-17-18-1-2-3-6-11
15-N15	12-N12	8.0	15-17-18-7-8-9-12
15-N15	13-N13	3.0	15-17-13
15-N15	14-N14	2.0	15-17-14
15-N15	16-N16	2.0	15-17-16
15-N15	17-N17	1.0	15-17
15-N15	18-A	1.5	15-17-18
15-N15	19-B	7.5	15-17-18-1-2-3-10-19

Figure 4.16: Shortest paths from node $n15$ to all other nodes

It has been observed from the figure 4.16 that the path from node $n15$ to destination node $n4$ shows infinite delay because the destination node N4 has been failed. From the results obtained above it is very clear that the transmission of data from node $n15$ to node $n12$ takes maximum amount of time i.e. $8ns$ and has 5 intermediate nodes. If we choose other path ($n15\ n17\ A\ n1\ n2\ n3\ n10\ n11\ n12$), the data will take $9.5ns$ to reach node $n12$ from node $n15$ and the number of intermediate nodes increase to 7. Hence, the time require to reach node $n12$ from node $n15$ reduces by, 1.5/9.5×100=15.79%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n15$ to node $n12$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

For node $n16$, there are various paths to the other nodes for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
16-N16	1-N1	2.5	16-17-18-1
16-N16	2-N2	3.5	16-17-18-1-2
16-N16	3-N3	4.5	16-17-18-1-2-3
16-N16	4-N4	infinity	
16-N16	5-N5	4.0	16-17-18-1-2-5
16-N16	6-N6	5.5	16-17-18-1-2-3-6
16-N16	7-N7	2.5	16-17-18-7
16-N16	8-N8	4.5	16-17-18-7-8
16-N16	9-N9	5.0	16-17-18-7-8-9
16-N16	10-N10	6.5	16-17-18-1-2-3-10
16-N16	11-N11	7.5	16-17-18-1-2-3-6-11
16-N16	12-N12	8.0	16-17-18-7-8-9-12
16-N16	13-N13	3.0	16-17-13
16-N16	14-N14	2.0	16-17-14
16-N16	15-N15	2.0	16-17-15
16-N16	17-N17	1.0	16-17
16-N16	18-A	1.5	16-17-18
16-N16	19-B	7.5	16-17-18-1-2-3-10-19

Figure 4.17: Shortest paths from node $n16$ to all other nodes

It has been observed from the figure 4.17 that the path from node $n16$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n16$ to node $n12$ takes maximum amount of time i.e. $8ns$ and has 5 intermediate nodes. If we choose other path ($n16\ n17\ A\ n1\ n2\ n3\ n10\ n11\ n12$), the data will take $9.5ns$ to reach node $n12$ from node $n16$ and the number of intermediate nodes increase to 7. Hence, the time require to reach node $n12$ from node $n16$ reduces by, 1.5/9.5×100=15.79%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n16$ to node $n12$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

For node $n17$, there are various paths to the other nodes for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
17-N17	1-N1	1.5	17-18-1
17-N17	2-N2	2.5	17-18-1-2
17-N17	3-N3	3.5	17-18-1-2-3
17-N17	4-N4	infinity	
17-N17	5-N5	3.0	17-18-1-2-5
17-N17	6-N6	4.5	17-18-1-2-3-6
17-N17	7-N7	1.5	17-18-7
17-N17	8-N8	3.5	17-18-7-8
17-N17	9-N9	4.0	17-18-7-8-9
17-N17	10-N10	5.5	17-18-1-2-3-10
17-N17	11-N11	6.5	17-18-1-2-3-6-11
17-N17	12-N12	7.0	17-18-7-8-9-12
17-N17	13-N13	2.0	17-13
17-N17	14-N14	1.0	17-14
17-N17	15-N15	1.0	17-15
17-N17	16-N16	1.0	17-16
17-N17	18-A	0.5	17-18
17-N17	19-B	6.5	17-18-1-2-3-10-19

Figure 4.18: Shortest paths from node $n17$ to all other nodes

It has been observed from the figure 4.18 that the path from node $n17$ to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n17$ to node $n12$ takes maximum amount of time i.e. $7ns$ and has 4 intermediate nodes. If we choose other path ($n17\ A\ n1\ n2\ n3\ n10\ n11\ n12$) the data will take $8.5ns$ to reach node $n12$ from node $n17$ and the number of intermediate nodes increase to 6. Hence, the time require to reach node $n12$ from node $n17$ reduces by, 1.5/8.5×100=17.65%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n17$ to node $n12$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

For node A, there are various paths to the other nodes and all for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
18-A	1-N1	1.0	18-1
18-A	2-N2	2.0	18-1-2
18-A	3-N3	3.0	18-1-2-3
18-A	4-N4	infinity	
18-A	5-N5	2.5	18-1-2-5
18-A	6-N6	4.0	18-1-2-3-6
18-A	7-N7	1.0	18-7
18-A	8-N8	3.0	18-7-8
18-A	9-N9	3.5	18-7-8-9
18-A	10-N10	5.0	18-1-2-3-10
18-A	11-N11	6.0	18-1-2-3-6-11
18-A	12-N12	6.5	18-7-8-9-12
18-A	13-N13	2.5	18-17-13
18-A	14-N14	1.5	18-17-14
18-A	15-N15	1.5	18-17-15
18-A	16-N16	1.5	18-17-16
18-A	17-N17	0.5	18-17
18-A	19-B	6.0	18-1-2-3-10-19

Figure 4.19: Shortest paths from node A to all other nodes

It has been observed from the figure 4.19 that the path from node A to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node A to node $n12$ takes maximum amount of time i.e. $6.5ns$ and has 3 intermediate nodes. If we choose other path ($A\ n1\ n2\ n3\ n10\ n11\ n12$) the data will take $8ns$ to reach node $n12$ from node A and the number of intermediate nodes increase to 5. Hence, the time require to reach node $n12$ from node A reduces by, 1.5/8×100=16.66%. The probability of failure of the schedule in the optimized path which has 3 intermediate nodes from node A to node $n12$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=3, so the probability of failure comes out to be 0.27.

.

For node B, there are various paths to the other nodes and all for effective transmission of data. From all the possible paths we need that path in which either delay or the intermediate nodes are minimum.

From	To	Distance	Route
19-B	1-N1	5.0	19-10-3-2-1
19-B	2-N2	4.0	19-10-3-2
19-B	3-N3	3.0	19-10-3
19-B	4-N4	infinity	
19-B	5-N5	4.5	19-10-3-2-5
19-B	6-N6	4.0	19-10-3-6
19-B	7-N7	6.5	19-12-9-8-7
19-B	8-N8	4.5	19-12-9-8
19-B	9-N9	4.0	19-12-9
19-B	10-N10	1.0	19-10
19-B	11-N11	2.0	19-11
19-B	12-N12	1.0	19-12
19-B	13-N13	8.5	19-10-3-2-1-18-17-13
19-B	14-N14	7.5	19-10-3-2-1-18-17-14
19-B	15-N15	7.5	19-10-3-2-1-18-17-15
19-B	16-N16	7.5	19-10-3-2-1-18-17-16
19-B	17-N17	6.5	19-10-3-2-1-18-17
19-B	18-A	6.0	19-10-3-2-1-18

Figure 4.20: Shortest paths from node B to all other nodes

It has been observed from the figure 4.20 that the path from node B to destination node $n4$ shows infinite delay because the destination node $n4$ has been failed. From the results obtained above it is very clear that the transmission of data from node B to node $n13$ takes maximum amount of time i.e. $8.5ns$ and has 6 intermediate nodes. If we choose other path (B $n12$ $n9$ $n8$ $n7$ A $n17$ $n13$) the data will take $10ns$ to reach node $n13$ from node B and number of intermediate nodes remains 6. Hence, the time require to reach node $n13$ from node B reduces by 1.5/10×100=15.00%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node B to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Case 2: when critical node $n12$ assume to be failed

The network has been critically analyzed and the shortest path for node $n1$ to all other nodes has been achieved when critical node $n12$ is failed. The table 4.1 shows the name of the destination node, the various paths from source node $n1$ to destination node, the delay along the path and the number of intermediate nodes.

S. NO.	Destination Node	Paths	Delay(ns)	Number of Intermediate Nodes
1.	N2	N1 $\xrightarrow{1}$ N2	1	0
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{1}$ N5 $\xrightarrow{0.5}$ N2	5.5	4
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{1}$ N5 $\xrightarrow{0.5}$ N2	4.5	2
2.	N3	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3	2	1
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{1}$ N3	4	3
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{1}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{1}$ N3	6.5	5
3.	N4	N1 $\xrightarrow{3}$ N4	3	0
		N1 $\xrightarrow{1}$ A $\xrightarrow{2}$ N4	3	1
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{1}$ N5 $\xrightarrow{0.5}$ N2	2.5	2
4.	N5	N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5	1.5	1
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N4 $\xrightarrow{1}$ N5	4.0	1
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{1}$ N5	4.0	2
5.	N6	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6	3	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6	3	2
		N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{0.5}$ N9 $\xrightarrow{1}$ N6	5.5	4
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10 $\xrightarrow{1}$ N11 $\xrightarrow{2}$ N6	7	4
6.	N7	N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7	2	1
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{3}$ N7	6	1

		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.0}$ N8 $\xrightarrow{2}$ N7	4.5	3
7.	N8	N1 $\xrightarrow{1}$ A $\xrightarrow{1}$ N7 $\xrightarrow{2}$ N8	4	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.0}$ N8	2.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{3}$ N7 $\xrightarrow{2}$ N8	8	2
8.	N9	N1 $\xrightarrow{3}$ N4 $\xrightarrow{3}$ N7 $\xrightarrow{2}$ N8 $\xrightarrow{0.5}$ N9	8.5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1.0}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{1.0}$ N9	4	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1}$ N8 $\xrightarrow{0.5}$ N9	3	3
9.	N10	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10	4	2
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{2}$ N11 $\xrightarrow{1}$ N10	6	4
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{2}$ N11 $\xrightarrow{1}$ N10	6	4
10.	N11	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10 $\xrightarrow{1}$ N11	5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{0.5}$ N5 $\xrightarrow{1.5}$ N6 $\xrightarrow{2}$ N11	5	3
		N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{2}$ N11	5	3
11.	N12	-		
12.	N13	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{2}$ N13	3.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{2}$ N13	7.5	3
13.	N14	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N14	2.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{2}$ N14	6.5	3
14.	N15	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N15	2.5	2
		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N15	6.5	3
15.	N16	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N16	2.5	2

		N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17 $\xrightarrow{1}$ N16	6.5	3
16.	N17	N1 $\xrightarrow{1}$ A $\xrightarrow{0.5}$ N17 N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A $\xrightarrow{0.5}$ N17	1.5 5.5	1 2
17.	A	N1 $\xrightarrow{1}$ A N1 $\xrightarrow{3}$ N4 $\xrightarrow{2}$ A	1 5	0 1
18.	B	N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{2}$ N10 $\xrightarrow{1}$ B N1 $\xrightarrow{1}$ N2 $\xrightarrow{1}$ N3 $\xrightarrow{1}$ N6 $\xrightarrow{2}$ N11 $\xrightarrow{2}$ B	5 7	3 4

Table 4.2: Various paths from source node N1 to destination nodes when N12 failed

Simulation result of shortest paths from source node $n1$ to other nodes is

From	To	Distance	Route
1-N1	2-N2	1.0	1-2
1-N1	3-N3	2.0	1-2-3
1-N1	4-N4	2.5	1-2-5-4
1-N1	5-N5	1.5	1-2-5
1-N1	6-N6	3.0	1-2-3-6
1-N1	7-N7	2.0	1-18-7
1-N1	8-N8	2.5	1-2-5-8
1-N1	9-N9	3.0	1-2-5-8-9
1-N1	10-N10	4.0	1-2-3-10
1-N1	11-N11	5.0	1-2-3-6-11
1-N1	12-N12	infinity	
1-N1	13-N13	3.5	1-18-17-13
1-N1	14-N14	2.5	1-18-17-14
1-N1	15-N15	2.5	1-18-17-15
1-N1	16-N16	2.5	1-18-17-16
1-N1	17-N17	1.5	1-18-17
1-N1	18-A	1.0	1-18
1-N1	19-B	5.0	1-2-3-10-19

Figure 4.21: Shortest paths from source node $n1$ to other nodes

It has been observed from the figure 4.21 that the path from node $n1$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n1$ to node $n11$ and B takes maximum amount of time i.e. $5ns$ and has 3 intermediate nodes. If we choose other path ($n1$ $n4$ $n5$ $n6$ $n11$), the data will take $7.5ns$ to reach node $n11$ from node $n1$, but the number of intermediate nodes remains 3. Hence, the time require to reach node $n11$ from node $n1$ reduces by, 2.5/7.5×100=33.33%. The probability of failure of the schedule in the optimized path which has 3 intermediate nodes from node $n1$ to node $n11$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=3, so the probability of failure comes out to be 0.27.

Simulation result of shortest paths from source node $n2$ to other nodes is

From	To	Distance	Route
2-N2	1-N1	1.0	2-1
2-N2	3-N3	1.0	2-3
2-N2	4-N4	1.5	2-5-4
2-N2	5-N5	0.5	2-5
2-N2	6-N6	2.0	2-3-6
2-N2	7-N7	3.0	2-1-18-7
2-N2	8-N8	1.5	2-5-8
2-N2	9-N9	2.0	2-5-8-9
2-N2	10-N10	3.0	2-3-10
2-N2	11-N11	4.0	2-3-6-11
2-N2	12-N12	infinity	
2-N2	13-N13	4.5	2-1-18-17-13
2-N2	14-N14	3.5	2-1-18-17-14
2-N2	15-N15	3.5	2-1-18-17-15
2-N2	16-N16	3.5	2-1-18-17-16
2-N2	17-N17	2.5	2-1-18-17
2-N2	18-A	2.0	2-1-18
2-N2	19-B	4.0	2-3-10-19

Figure 4.22: Shortest paths from source node $n2$ to other nodes

It has been observed from the figure 4.22 that the path from node $n2$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n2$ to node $n13$ takes maximum amount of time i.e. $4.5ns$ and has 3 intermediate nodes. If we choose other path ($n2$ $n5$ $n4$ A $n17$ $n13$), the data will take $6ns$ to reach node $n13$ from node $n2$ and the number of intermediate nodes increases to 4. Hence, the time require to reach node $n13$ from node $n2$ reduces by, 1.5/6×100=25.00%. The probability of failure of the schedule in the optimized path which has 3 intermediate nodes from node $n2$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=3, so the probability of failure comes out to be 0.27.

Simulation result of shortest paths from source node $n3$ to other nodes is

From	To	Distance	Route
3-N3	1-N1	2.0	3-2-1
3-N3	2-N2	1.0	3-2
3-N3	4-N4	2.5	3-2-5-4
3-N3	5-N5	1.5	3-2-5
3-N3	6-N6	1.0	3-6
3-N3	7-N7	4.5	3-2-5-8-7
3-N3	8-N8	2.5	3-2-5-8
3-N3	9-N9	2.0	3-6-9
3-N3	10-N10	2.0	3-10
3-N3	11-N11	3.0	3-6-11
3-N3	12-N12	infinity	
3-N3	13-N13	5.5	3-2-1-18-17-13
3-N3	14-N14	4.5	3-2-1-18-17-14
3-N3	15-N15	4.5	3-2-1-18-17-15
3-N3	16-N16	4.5	3-2-1-18-17-16
3-N3	17-N17	3.5	3-2-1-18-17
3-N3	18-A	3.0	3-2-1-18
3-N3	19-B	3.0	3-10-19

Figure 4.23: Shortest paths from source node $n3$ to other nodes

It has been observed from the figure 4.23 that the path from node $n3$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n3$ to node $n13$ takes maximum amount of time i.e. $5.5ns$ and has 4 intermediate nodes. If we choose other path ($n3$ $n6$ $n5$ $n4$ A $n17$ $n13$), the data will take $8ns$ to reach node $n13$ from node $n3$ and the number of intermediate nodes increase to 5. Hence, the time require to reach node $n13$ from node $n3$ reduces by, 2.5/8×100=31.25%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n3$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

Simulation result of shortest paths from source node $n4$ to other nodes is

From	To	Distance	Route
4-N4	1-N1	2.5	4-5-2-1
4-N4	2-N2	1.5	4-5-2
4-N4	3-N3	2.5	4-5-2-3
4-N4	5-N5	1.0	4-5
4-N4	6-N6	2.5	4-5-6
4-N4	7-N7	3.0	4-7
4-N4	8-N8	2.0	4-5-8
4-N4	9-N9	2.5	4-5-8-9
4-N4	10-N10	4.5	4-5-2-3-10
4-N4	11-N11	4.5	4-5-6-11
4-N4	12-N12	infinity	
4-N4	13-N13	4.5	4-18-17-13
4-N4	14-N14	3.5	4-18-17-14
4-N4	15-N15	3.5	4-18-17-15
4-N4	16-N16	3.5	4-18-17-16
4-N4	17-N17	2.5	4-18-17
4-N4	18-A	2.0	4-18
4-N4	19-B	5.5	4-5-2-3-10-19

Figure 4.24: Shortest paths from node $n4$to all other nodes

It has been observed from the figure 4.24 that the path from node $n4$ to destination $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n4$ to node B takes maximum amount of time i.e. $5.5ns$ and has 4 intermediate nodes. If we choose other path ($n4$ $n5$ $n6$ $n11$ B), the data will take $6.5ns$ to reach node B from node $n4$ but the number of intermediate nodes reduces to 3. Hence, the time require to reach node B from node $n4$ reduces by 1/6.5×100=15.38%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n4$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

.

Simulation result of shortest paths from source node $n5$ to other nodes is

From	To	Distance	Route
5-N5	1-N1	1.5	5-2-1
5-N5	2-N2	0.5	5-2
5-N5	3-N3	1.5	5-2-3
5-N5	4-N4	1.0	5-4
5-N5	6-N6	1.5	5-6
5-N5	7-N7	3.0	5-8-7
5-N5	8-N8	1.0	5-8
5-N5	9-N9	1.5	5-8-9
5-N5	10-N10	3.5	5-2-3-10
5-N5	11-N11	3.5	5-6-11
5-N5	12-N12	infinity	
5-N5	13-N13	5.0	5-2-1-18-17-13
5-N5	14-N14	4.0	5-2-1-18-17-14
5-N5	15-N15	4.0	5-2-1-18-17-15
5-N5	16-N16	4.0	5-2-1-18-17-16
5-N5	17-N17	3.0	5-2-1-18-17
5-N5	18-A	2.5	5-2-1-18
5-N5	19-B	4.5	5-2-3-10-19

Figure 4.25: Shortest paths from node $n5$ to all other nodes

It has been observed from the figure 4.25 that the path from node $n5$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n5$ to node $n13$ takes maximum amount of time i.e. $5ns$ and has 4 intermediate nodes. If we choose other path ($n5$ $n4$ A $n17$ $n13$), the data will take $5.5ns$ to reach node $n13$ from node $n5$ but the number of intermediate nodes reduces to 3. Hence, the time require to reach node $n13$ from node $n5$ reduces by, 0.5/5.5×100=9.09%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n5$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

Simulation result of shortest paths from source node $n6$ to other nodes is

From	To	Distance	Route
6-N6	1-N1	3.0	6-3-2-1
6-N6	2-N2	2.0	6-3-2
6-N6	3-N3	1.0	6-3
6-N6	4-N4	2.5	6-5-4
6-N6	5-N5	1.5	6-5
6-N6	7-N7	3.5	6-9-8-7
6-N6	8-N8	1.5	6-9-8
6-N6	9-N9	1.0	6-9
6-N6	10-N10	3.0	6-3-10
6-N6	11-N11	2.0	6-11
6-N6	12-N12	infinity	
6-N6	13-N13	6.5	6-3-2-1-18-17-13
6-N6	14-N14	5.5	6-3-2-1-18-17-14
6-N6	15-N15	5.5	6-3-2-1-18-17-15
6-N6	16-N16	5.5	6-3-2-1-18-17-16
6-N6	17-N17	4.5	6-3-2-1-18-17
6-N6	18-A	4.0	6-3-2-1-18
6-N6	19-B	4.0	6-3-10-19

Figure 4.26: Shortest paths from node $n6$ to all other nodes

It has been observed from the figure 4.26 that the path from node $n6$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n6$ to node $n13$ takes maximum amount of time i.e. $6.5ns$ and has 5 intermediate nodes. If we choose other path ($n6$ $n5$ $n4$ A $n17$ $n13$), the data will take $7ns$ to reach node $n13$ from node $n6$ but the number of intermediate nodes reduces to 4. Hence, the time require to reach node $n13$ from node $n6$ reduces by, 0.5/7×100=7.14%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n6$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation result of shortest paths from source node $n7$ to other nodes is

From	To	Distance	Route
7-N7	1-N1	2.0	7-18-1
7-N7	2-N2	3.0	7-18-1-2
7-N7	3-N3	4.0	7-18-1-2-3
7-N7	4-N4	3.0	7-4
7-N7	5-N5	3.0	7-8-5
7-N7	6-N6	3.5	7-8-9-6
7-N7	8-N8	2.0	7-8
7-N7	9-N9	2.5	7-8-9
7-N7	10-N10	6.0	7-18-1-2-3-10
7-N7	11-N11	5.5	7-8-9-6-11
7-N7	12-N12	infinity	
7-N7	13-N13	3.5	7-18-17-13
7-N7	14-N14	2.5	7-18-17-14
7-N7	15-N15	2.5	7-18-17-15
7-N7	16-N16	2.5	7-18-17-16
7-N7	17-N17	1.5	7-18-17
7-N7	18-A	1.0	7-18
7-N7	19-B	7.0	7-18-1-2-3-10-19

Figure 4.27: Shortest paths from node $n7$ to all other nodes

From the figure 4.27, it has been observed that the path from node $n7$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n7$ to node B takes maximum amount of time i.e. $7ns$ and has 5 intermediate nodes. If we choose other path ($n7$ $n8$ $n5$ $n6$ $n11$ B), the data will take $8.5ns$ to reach node B from node $n7$ but the number of intermediate nodes reduces to 4. Hence, the time require to reach node B from node $n7$ reduces by, 1.5/8.5×100=17.64%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n7$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation result of shortest paths from source node $n8$ to other nodes is

From	To	Distance	Route
8-N8	1-N1	2.5	8-5-2-1
8-N8	2-N2	1.5	8-5-2
8-N8	3-N3	2.5	8-5-2-3
8-N8	4-N4	2.0	8-5-4
8-N8	5-N5	1.0	8-5
8-N8	6-N6	1.5	8-9-6
8-N8	7-N7	2.0	8-7
8-N8	9-N9	0.5	8-9
8-N8	10-N10	4.5	8-5-2-3-10
8-N8	11-N11	3.5	8-9-6-11
8-N8	12-N12	infinity	
8-N8	13-N13	5.5	8-7-18-17-13
8-N8	14-N14	4.5	8-7-18-17-14
8-N8	15-N15	4.5	8-7-18-17-15
8-N8	16-N16	4.5	8-7-18-17-16
8-N8	17-N17	3.5	8-7-18-17
8-N8	18-A	3.0	8-7-18
8-N8	19-B	5.5	8-5-2-3-10-19

Figure 4.28: Shortest paths from node $n8$ to all other nodes

It has been observed from the figure 4.28 that the path from node $n8$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n8$ to node B and $n13$ takes maximum amount of time i.e. $5.5ns$ and has 4 and 3 intermediate nodes. If we choose other path ($n8$ $n5$ $n6$ $n11$ B), the data will take $6.5ns$ to reach node B from node $n8$ but the number of intermediate nodes reduces to 3. Hence, the time require to reach node B from node $n8$ reduces by, 1/6.5×100=15.36%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n8$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

Simulation result of shortest paths from source node $n9$ to other nodes is

From	To	Distance	Route
9-N9	1-N1	3.0	9-8-5-2-1
9-N9	2-N2	2.0	9-8-5-2
9-N9	3-N3	2.0	9-6-3
9-N9	4-N4	2.5	9-8-5-4
9-N9	5-N5	1.5	9-8-5
9-N9	6-N6	1.0	9-6
9-N9	7-N7	2.5	9-8-7
9-N9	8-N8	0.5	9-8
9-N9	10-N10	4.0	9-6-3-10
9-N9	11-N11	3.0	9-6-11
9-N9	12-N12	infinity	
9-N9	13-N13	6.0	9-8-7-18-17-13
9-N9	14-N14	5.0	9-8-7-18-17-14
9-N9	15-N15	5.0	9-8-7-18-17-15
9-N9	16-N16	5.0	9-8-7-18-17-16
9-N9	17-N17	4.0	9-8-7-18-17
9-N9	18-A	3.5	9-8-7-18
9-N9	19-B	5.0	9-6-3-10-19

Figure 4.29: Shortest paths from node $n9$ to all other nodes

It has been observed from the figure 4.29 that the path from node $n9$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n9$ to node $n13$ takes maximum amount of time i.e. $6ns$ and has 4 intermediate nodes. If we choose other path ($n9$ $n6$ $n5$ $n4$ A $n17$ $n13$), the data will take $8ns$ to reach node $n13$ from node $n9$ and the number of intermediate nodes increase to 5. Hence, the time require to reach node $n13$ from node $n9$ reduces by, 2/8×100=25.00%. The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node $n9$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

Simulation result of shortest paths from source node $n10$ to other nodes is

From	To	Distance	Route
10-N10	1-N1	4.0	10-3-2-1
10-N10	2-N2	3.0	10-3-2
10-N10	3-N3	2.0	10-3
10-N10	4-N4	4.5	10-3-2-5-4
10-N10	5-N5	3.5	10-3-2-5
10-N10	6-N6	3.0	10-3-6
10-N10	7-N7	6.0	10-3-2-1-18-7
10-N10	8-N8	4.5	10-3-2-5-8
10-N10	9-N9	4.0	10-3-6-9
10-N10	11-N11	1.0	10-11
10-N10	12-N12	infinity	
10-N10	13-N13	7.5	10-3-2-1-18-17-13
10-N10	14-N14	6.5	10-3-2-1-18-17-14
10-N10	15-N15	6.5	10-3-2-1-18-17-15
10-N10	16-N16	6.5	10-3-2-1-18-17-16
10-N10	17-N17	5.5	10-3-2-1-18-17
10-N10	18-A	5.0	10-3-2-1-18
10-N10	19-B	1.0	10-19

Figure 4.30: Shortest paths from node $n10$ to all other nodes

It has been observed from the figure 4.30 that the path from node $n10$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n10$ to node $n13$ takes maximum amount of time i.e. $7.5ns$ and has 5 intermediate nodes. If we choose other path ($n10$ $n11$ $n6$ $n5$ $n4$ A $n17$ $n13$), the data will take $10ns$ to reach node $n13$ from node $n10$ and the number of intermediate nodes increase to 6. Hence, the time require to reach node $n13$ from node $n10$ reduces by, 2.5/10×100=25.00%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n10$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation result of shortest paths from source node $n11$ to other nodes is

From	To	Distance	Route
11-N11	1-N1	5.0	11-6-3-2-1
11-N11	2-N2	4.0	11-6-3-2
11-N11	3-N3	3.0	11-6-3
11-N11	4-N4	4.5	11-6-5-4
11-N11	5-N5	3.5	11-6-5
11-N11	6-N6	2.0	11-6
11-N11	7-N7	5.5	11-6-9-8-7
11-N11	8-N8	3.5	11-6-9-8
11-N11	9-N9	3.0	11-6-9
11-N11	10-N10	1.0	11-10
11-N11	12-N12	infinity	
11-N11	13-N13	8.5	11-6-3-2-1-18-17-13
11-N11	14-N14	7.5	11-6-3-2-1-18-17-14
11-N11	15-N15	7.5	11-6-3-2-1-18-17-15
11-N11	16-N16	7.5	11-6-3-2-1-18-17-16
11-N11	17-N17	6.5	11-6-3-2-1-18-17
11-N11	18-A	6.0	11-6-3-2-1-18
11-N11	19-B	2.0	11-19

Figure 4.31: Shortest paths from node $n11$ to all other nodes

It has been observed from the figure 4.31 that the path from node $n11$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n11$ to node $n13$ takes maximum amount of time i.e. $8.5ns$ and has 6 intermediate nodes. If we choose other path ($n11$ $n6$ $n5$ $n4$ A $n17$ $n13$), the data will take $9ns$ to reach node $n13$ from node $n11$ but the number of intermediate nodes reduces to 5. Hence, the time require to reach node $n13$ from node $n11$ reduces by, 0.5/9×100=5.55%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node $n11$ to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Simulation result of shortest paths from source node $n12$ to other nodes is

From	To	Distance	Route
12-N12	1-N1	infinity	
12-N12	2-N2	infinity	
12-N12	3-N3	infinity	
12-N12	4-N4	infinity	
12-N12	5-N5	infinity	
12-N12	6-N6	infinity	
12-N12	7-N7	infinity	
12-N12	8-N8	infinity	
12-N12	9-N9	infinity	
12-N12	10-N10	infinity	
12-N12	11-N11	infinity	
12-N12	13-N13	infinity	
12-N12	14-N14	infinity	
12-N12	15-N15	infinity	
12-N12	16-N16	infinity	
12-N12	17-N17	infinity	
12-N12	18-A	infinity	
12-N12	19-B	infinity	

Figure 4.32: Shortest paths from node $n12$ to all other nodes

It has been observed from the figure 4.32 that for node $n12$, there is no path to the other nodes because node $n12$ has been failed and hence scheduling for this node is not possible. In other words transmission of data is not possible from node $n12$.

Simulation result of shortest paths from source node $n13$ to other nodes is

From	To	Distance	Route
13-N13	1-N1	3.5	13-17-18-1
13-N13	2-N2	4.5	13-17-18-1-2
13-N13	3-N3	5.5	13-17-18-1-2-3
13-N13	4-N4	4.5	13-17-18-4
13-N13	5-N5	5.0	13-17-18-1-2-5
13-N13	6-N6	6.5	13-17-18-1-2-3-6
13-N13	7-N7	3.5	13-17-18-7
13-N13	8-N8	5.5	13-17-18-7-8
13-N13	9-N9	6.0	13-17-18-7-8-9
13-N13	10-N10	7.5	13-17-18-1-2-3-10
13-N13	11-N11	8.5	13-17-18-1-2-3-6-11
13-N13	12-N12	infinity	
13-N13	14-N14	3.0	13-17-14
13-N13	15-N15	3.0	13-17-15
13-N13	16-N16	3.0	13-17-16
13-N13	17-N17	2.0	13-17
13-N13	18-A	2.5	13-17-18
13-N13	19-B	8.5	13-17-18-1-2-3-10-19

Figure 4.33: Shortest paths from node $n13$ to all other nodes

It has been observed from the figure 4.33 that the path from node $n13$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n13$ to node B and node $n11$ takes maximum amount of time i.e. $8.5ns$ and has 6 intermediate nodes. If we choose other path ($n13$ $n17$ A $n4$ $n5$ $n6$ $n11$ B), the data will take $11ns$ to reach node B from node $n13$ but the number of intermediate nodes remains 6. Hence, the time require to reach node B from node $n13$ reduces by, 2.5/11×100=22.72%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node $n13$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Simulation result of shortest paths from source node $n14$ to other nodes is

From	To	Distance	Route
14-N14	1-N1	2.5	14-17-18-1
14-N14	2-N2	3.5	14-17-18-1-2
14-N14	3-N3	4.5	14-17-18-1-2-3
14-N14	4-N4	3.5	14-17-18-4
14-N14	5-N5	4.0	14-17-18-1-2-5
14-N14	6-N6	5.5	14-17-18-1-2-3-6
14-N14	7-N7	2.5	14-17-18-7
14-N14	8-N8	4.5	14-17-18-7-8
14-N14	9-N9	5.0	14-17-18-7-8-9
14-N14	10-N10	6.5	14-17-18-1-2-3-10
14-N14	11-N11	7.5	14-17-18-1-2-3-6-11
14-N14	12-N12	infinity	
14-N14	13-N13	3.0	14-17-13
14-N14	15-N15	2.0	14-17-15
14-N14	16-N16	2.0	14-17-16
14-N14	17-N17	1.0	14-17
14-N14	18-A	1.5	14-17-18
14-N14	19-B	7.5	14-17-18-1-2-3-10-19

Figure 4.34: Shortest paths from node $n14$ to all other nodes

It has been observed from the figure 4.34 that the path from node $n14$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n14$ to node B and node $n11$ takes maximum amount of time i.e. $7.5ns$ and has 6 intermediate nodes. If we choose other path ($n14$ $n17$ A $n4$ $n5$ $n6$ $n11$ B), the data will take $10ns$ to reach node B from node $n14$ but the number of intermediate nodes remains 6. Hence, the time require to reach node B from node $n14$ reduces by, 2.5/10×100=25.00%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node $n14$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Simulation result of shortest paths from source node $n15$ to other nodes is

From	To	Distance	Route
15-N15	1-N1	2.5	15-17-18-1
15-N15	2-N2	3.5	15-17-18-1-2
15-N15	3-N3	4.5	15-17-18-1-2-3
15-N15	4-N4	3.5	15-17-18-4
15-N15	5-N5	4.0	15-17-18-1-2-5
15-N15	6-N6	5.5	15-17-18-1-2-3-6
15-N15	7-N7	2.5	15-17-18-7
15-N15	8-N8	4.5	15-17-18-7-8
15-N15	9-N9	5.0	15-17-18-7-8-9
15-N15	10-N10	6.5	15-17-18-1-2-3-10
15-N15	11-N11	7.5	15-17-18-1-2-3-6-11
15-N15	12-N12	infinity	
15-N15	13-N13	3.0	15-17-13
15-N15	14-N14	2.0	15-17-14
15-N15	16-N16	2.0	15-17-16
15-N15	17-N17	1.0	15-17
15-N15	18-A	1.5	15-17-18
15-N15	19-B	7.5	15-17-18-1-2-3-10-19

Figure 4.35: Shortest paths from node $n15$ to all other nodes

It has been observed from the figure 4.35 that the path from node $n15$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n15$ to node B and $n11$ takes maximum amount of time i.e. $7.5ns$ and has 6 intermediate nodes. If we choose other path ($n15$ $n17$ A $n4$ $n5$ $n6$ $n11$ B), the data will take $10ns$ to reach node B from node $n15$ but the number of intermediate nodes remains 6. Hence, the time require to reach node B from node $n15$ reduces by, 2.5/10×100=25.00%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node $n15$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Simulation result of shortest paths from source node $n16$ to other nodes is

From	To	Distance	Route
16-N16	1-N1	2.5	16-17-18-1
16-N16	2-N2	3.5	16-17-18-1-2
16-N16	3-N3	4.5	16-17-18-1-2-3
16-N16	4-N4	3.5	16-17-18-4
16-N16	5-N5	4.0	16-17-18-1-2-5
16-N16	6-N6	5.5	16-17-18-1-2-3-6
16-N16	7-N7	2.5	16-17-18-7
16-N16	8-N8	4.5	16-17-18-7-8
16-N16	9-N9	5.0	16-17-18-7-8-9
16-N16	10-N10	6.5	16-17-18-1-2-3-10
16-N16	11-N11	7.5	16-17-18-1-2-3-6-11
16-N16	12-N12	infinity	
16-N16	13-N13	3.0	16-17-13
16-N16	14-N14	2.0	16-17-14
16-N16	15-N15	2.0	16-17-15
16-N16	17-N17	1.0	16-17
16-N16	18-A	1.5	16-17-18
16-N16	19-B	7.5	16-17-18-1-2-3-10-19

Figure 4.36: Shortest paths from node $n16$ to all other nodes

It has been observed from the figure 4.36 that the path from node $n16$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n16$ to node B and $node$ $n11$ takes maximum amount of time i.e. $7.5ns$ and has 6 intermediate nodes. If we choose other path ($n16$ $n17$ A $n4$ $n5$ $n6$ $n11$ B), the data will take $10ns$ to reach node B from node $n16$ but the number of intermediate nodes remains 6. Hence, the time require to reach node B from node $n16$ reduces by, 2.5/10×100=25.00%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node $n16$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=6, so the probability of failure comes out to be 0.45.

Simulation result of shortest paths from source node $n17$ to other nodes is

From	To	Distance	Route
17-N17	1-N1	1.5	17-18-1
17-N17	2-N2	2.5	17-18-1-2
17-N17	3-N3	3.5	17-18-1-2-3
17-N17	4-N4	2.5	17-18-4
17-N17	5-N5	3.0	17-18-1-2-5
17-N17	6-N6	4.5	17-18-1-2-3-6
17-N17	7-N7	1.5	17-18-7
17-N17	8-N8	3.5	17-18-7-8
17-N17	9-N9	4.0	17-18-7-8-9
17-N17	10-N10	5.5	17-18-1-2-3-10
17-N17	11-N11	6.5	17-18-1-2-3-6-11
17-N17	12-N12	infinity	
17-N17	13-N13	2.0	17-13
17-N17	14-N14	1.0	17-14
17-N17	15-N15	1.0	17-15
17-N17	16-N16	1.0	17-16
17-N17	18-A	0.5	17-18
17-N17	19-B	6.5	17-18-1-2-3-10-19

Figure 4.37: Shortest paths from node $n17$ to all other nodes

It has been observed from the figure 4.37 that the path from node $n17$ to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node $n17$ to node B and $n11$ takes maximum amount of time i.e. $6.5ns$ and has 5 intermediate nodes. If we choose other path ($n17$ A $n4$ $n5$ $n6$ $n11$ B), the data will take $9ns$ to reach node B from node $n17$ but the number of intermediate nodes remains 5. Hence, the time require to reach node B from node $n17$ reduces by, 2.5/9×100=27.77%. The probability of failure of the schedule in the optimized path which has 5 intermediate nodes from node $n17$ to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=5, so the probability of failure comes out to be 0.4.

Simulation result of shortest paths from source node A to other nodes is

From	To	Distance	Route
18-A	1-N1	1.0	18-1
18-A	2-N2	2.0	18-1-2
18-A	3-N3	3.0	18-1-2-3
18-A	4-N4	2.0	18-4
18-A	5-N5	2.5	18-1-2-5
18-A	6-N6	4.0	18-1-2-3-6
18-A	7-N7	1.0	18-7
18-A	8-N8	3.0	18-7-8
18-A	9-N9	3.5	18-7-8-9
18-A	10-N10	5.0	18-1-2-3-10
18-A	11-N11	6.0	18-1-2-3-6-11
18-A	12-N12	infinity	
18-A	13-N13	2.5	18-17-13
18-A	14-N14	1.5	18-17-14
18-A	15-N15	1.5	18-17-15
18-A	16-N16	1.5	18-17-16
18-A	17-N17	0.5	18-17
18-A	19-B	6.0	18-1-2-3-10-19

Figure 4.38: Shortest paths from node A to all other nodes

It has been observed from the figure 4.38 that the path from node A to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node A to node B and $node$ $n11$ takes maximum amount of time i.e. $6ns$ and has 4 intermediate nodes. If we choose other path (A $n4$ $n5$ $n6$ $n11$ B), the data will take $8.5ns$ to reach node B from node A but the number of intermediate nodes remains 4. Hence, the time require to reach node B from node A reduces by, 2.5/8.5×100=29.41% The probability of failure of the schedule in the optimized path which has 4 intermediate nodes from node A to node B, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.34.

Simulation result of shortest paths from source node B to other nodes is

From	To	Distance	Route
19-B	1-N1	5.0	19-10-3-2-1
19-B	2-N2	4.0	19-10-3-2
19-B	3-N3	3.0	19-10-3
19-B	4-N4	5.5	19-10-3-2-5-4
19-B	5-N5	4.5	19-10-3-2-5
19-B	6-N6	4.0	19-10-3-6
19-B	7-N7	7.0	19-10-3-2-1-18-7
19-B	8-N8	5.5	19-10-3-2-5-8
19-B	9-N9	5.0	19-10-3-6-9
19-B	10-N10	1.0	19-10
19-B	11-N11	2.0	19-11
19-B	12-N12	infinity	
19-B	13-N13	8.5	19-10-3-2-1-18-17-13
19-B	14-N14	7.5	19-10-3-2-1-18-17-14
19-B	15-N15	7.5	19-10-3-2-1-18-17-15
19-B	16-N16	7.5	19-10-3-2-1-18-17-16
19-B	17-N17	6.5	19-10-3-2-1-18-17
19-B	18-A	6.0	19-10-3-2-1-18

Figure 4.39: Shortest paths from node B to all other nodes

It has been observed from the figure 4.39 that the path from node B to destination node $n12$ shows infinite delay because the destination node $n12$ has been failed. From the results obtained above it is very clear that the transmission of data from node B to $n13$ takes maximum amount of time i.e. $8.5ns$ and has 6 intermediate nodes. If we choose other path (B $n11$ $n6$ $n5$ $n4$ A 17 $n13$), the data will take $11ns$ to reach node $n13$ from node B but the number of intermediate nodes remains 6. Hence, the time require to reach node $n13$ from node B reduces by, 2.5/11×100=22.72%. The probability of failure of the schedule in the optimized path which has 6 intermediate nodes from node B to node $n13$, is given by,

$$\sum_{r=1}^{n} \binom{n}{r} \left(\frac{1}{10}\right)^{r} \left(\frac{9}{10}\right)^{n-r}$$

Here n=4, so the probability of failure comes out to be 0.45.

The following results have been drawn from the simulation results obtained

The shortest paths between all the nodes along with the calculation of probability of error in all the paths have been evaluated using dynamic scheduling. Since in a network a node can fail at any time so the dynamic scheduling has been achieved by considering two cases:

Result drawn when node $N4$ gets failed

It has been observed that no path exist from node $N4$ to any other node because node $N4$ has failed. Node $N13$ and node $N12$ are critical nodes as maximum amount of time is require to reach these nodes from most of other nodes. From nodes $N1$, $N2$, $N13$, $N14$, $N15$, $N16$, $N17$ and A, the data takes maximum amount of time to reach node $N12$ and from nodes $N3$, $N5$, $N6$, $N8$, $N9$, $N10$, $N11$, $N12$ and B, the data takes maximum amount of time to reach node $N13$ and the data from node $N7$ takes maximum time to reach node B. The probability of failure of schedule because of the failure of intermediate nodes has been evaluated and it has been found that scheduling the tasks to nodes $N13$ and $N12$ is most likely to fail. It has also been observed that as the number of intermediate nodes increases, the probability of failure also increases which in turn makes the schedule along the path less desirable. The optimization of the paths and the evaluation of the probability of failure have suggested that least critical tasks should be scheduled to the nodes $N12$ and $N13$ so that the critical tasks don't miss their deadlines and the feasible schedule is obtained. The above observations justify that nodes $N12$ and $N13$ are the most critical nodes and optimization of paths is highly require for these two nodes.

Result drawn when node critical node $N12$ gets failed

It has been observed that no path exist from node $N12$ to any other node because node $N12$ has failed. From nodes $N4$, $N7$, $N8$, $N13$, $N14$, $N15$, $N16$, $N17$ and A, the data takes maximum amount of time to reach node B and from nodes $N2$, $N3$, $N5$, $N6$, $N9$, $N10$, $N11$and B, the data takes maximum amount of time to reach node $N13$ and the data from node $N1$ takes maximum time to reach node B and $N11$. The probability of failure of schedule because of the failure of intermediate nodes has been evaluated and it has been found that scheduling the tasks to nodes $N13$ and B is most likely to fail. The optimization of the paths and the evaluation of the probability of failure have suggested that least critical tasks should be scheduled to the nodes B and $N13$ so that the critical tasks don't miss their deadlines and the feasible schedule is obtained. The above observations justify

that nodes $N12$ and $N13$ are the most critical nodes and optimization of paths is highly require for these two nodes.

It has been observed that in a network when a node gets failed the critical nodes also changes and least critical tasks should be scheduled to the nodes. This optimization has been achieved by dynamic scheduling and the delay has been minimized.

Chapter 5: Conclusion and Future Scope

The study of various real time systems and their classifications has been done. The scheduling algorithms have also been discussed in detail along with the comparison of different real time scheduling algorithms. The implementation of shortest path algorithm in real time has been done for the tasks. In chapter number 4 results and discussion, the problem for finding the shortest path in the communication network has been taken and simulated accordingly. The shortest paths between all the nodes along with the calculation of probability of error in all the paths have been evaluated using dynamic scheduling by considering two cases: (i) when node $N4$ gets failed (ii) when critical node $N12$ gets failed. From the above two cases it has been observed that the probability of error or failure comes out to be the most for nodes $N13$, $N12$ and B. Since the probability of failure is high for these nodes so the least critical tasks should be schedule across these nodes so that the critical tasks don't miss their deadlines and the feasible schedule is obtained. Resource optimization has been achieved in our technique as it only involves the optimal paths between two nodes.

The work can be extended by exploiting the priority constraints for static as well as dynamic priorities. The limited pre-emption techniques can be exploited for the efficient working of real-time systems. The overheads can also be managed by using partly pre-emptive systems. The techniques can be tested on the different platforms or software so as to ensure its optimality for all systems.

References:

[1] C. L. Liu and James W. Layland, “Scheduling Algorithm for Multiprogramming In A Hard Real Time Environment”, Journal of the Association of Computing Machinery, Vol. 20, No. 1, 1973, pp. 46-61.

[2] C. V. Ramamoorthy and Benjamin W. Wah, “An Optimal Algorithm for Scheduling Requests on Interleaved Memories for a Pipelined Processor”, IEEE Transactions on Computers, Vol. C-30, No. 10, 1981, pp. 787-799.

[3] A. F. Bashir, V. Susarla, and K. Vairavan, “A Statistical Study of the Performance of a Task Scheduling Algorithm”, IEEE Transactions on Computers, Vol. C-32. No. 8, 1983, pp. 774 – 777.

[4] Pauline Markenscoff, “Bus scheduling for a multiple-processor system with shared buses”, IEEE Proceedings, Vol. 134, No. 6, 1987, pp. 288-294.

[5] Franco Callari, "Types of Scheduling - Long Term and Medium Term Scheduling".

[6] Almut Burchard, Jorg Liebeherr and Sang H. Son, “New Strategies for Assigning Real Time Tasks to Multimocessor Svstems”, IEEE Transactions on Computers, Vol. 44, No. 12, 1995, pp.1429-1442.

[7] Alan A. Bertossi, Luigi V. Mancini and Federico Rossini, “Fault-Tolerant Rate Monotonic First-Fit Scheduling In Hard-Real-Time Systems”, IEEE Transactions on Parallel and Distributed Systems, Vol. 10, No. 9, 1999, pp. 934-945.

[8] Giorgio C. Buttazzo, "Hard Real Time Computing Systems: Predictable Scheduling Algorithms and Applications", Springer, Third edition.

[9] Sanjoy K. Baruah and Joel Goossens, “Rate-Monotonic Scheduling on Uniform Multiprocessors", IEEE Transactions on Computers, Vol. 52, No. 7, July 2003, pp. 966-970.

[10] Yang Yang, Krijn Van Der Raadt and Henri Casanova, “Multiround Algorithms for Scheduling Divisible Loads”, IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 11, 2005, pp. 1192-1203.

[11] Wei Zhao, Krithi Ramamritham, and John A. Stankovic, “Preemptive Scheduling Under Time and Resource Constraints”, IEEE Transactions on Computers, Vol. C-36, No. 8, 1987, pp. 949-960.

[12] Jia Wei Hong, Xiaonan Tan, and Don Towsley, “A Performance Analysis of Minimum Laxity and Earliest Deadline Scheduling in A Real-Time System”, IEEE Transactions on Computers, Vol. 38, No. 12, 1989, pp. 1736-1744.

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

[14] Krithi Ramamritham, John A. Stankovic and Perng Fe1 Shiah, “Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems”, IEEE Transactions on Parallel and Distributed Systems, Vol. 1, No. 2, 1990, pp. 184-194.

[15] Jia Xu and David Lorze Parnas, “Scheduling Processes with Release Times, Deadlines, Precedence and Exclusion Relations”, IEEE Transactions on Software Engineering, Vol. 16. No.3, 1990, pp. 360-369.

[16] Karsten Schwan and Hongyi Zhou, “Dynamic Scheduling of Hard Real-Time Tasks and Real-Time Threads”, IEEE Transactions on Software Engineering, Vol. 18, No. 8, 1992, pp. 736-748.

[17] Howard Hamilton, Kimberly Lemieux, Allan Tease, "Scheduling of Processes", http://www2.cs.uregina.ca/~hamilton/courses/330/notes/scheduling/scheduling.html.

[18] Kwang S. Hong and Joseph Y. T. hung, “On-Line Scheduling of Real-Time Tasks” IEEE Transactions on Computers, Vol. 41, No. 10, 1992, pp. 1326-1331.

[19] Marco Spuri and John A. Stankovic, “How to Integrate Precedence Constraints and Shared Resources in Real-Time Scheduling” IEEE Transactions on Computers, Vol. 43, No. 12, 1994, pp. 1407-1412.

[20] Kang G. Shin and Yi Chieh Chang, “ A Reservation-Based Algorithm for Scheduling Both Periodic and Aperiodic Real-Time Tasks”, IEEE Transactions on Computers, Vol. 44, No. 12, 1995, pp. 1405-1419.

[21] Wu Chun Feng and Jane W. S. Liu, “Algorithms for Scheduling Real-Time Tasks with Input Error and End-to-End Deadlines”, IEEE Transactions on Software Engineering, Vol. 23, No. 2, 1997, pp. 93-106.

[22] G. Manimaran and C. Siva Ram Murthy, “An Efficient Dynamic Scheduling Algorithm for Multiprocessor Real-Time Systems”, IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 3, 1998, pp. 312-319.

[23] Xian Bo He, Gang-Yuan Zhang, Min Liu, Yu-Ping Zhao and Wei Li, “A Fuzzy EDF Scheduling Algorithm Being Suitable for Embedded Soft Real-time Systems in the Uncertain Environments”, IEEE, 2010, pp. 583-587.

[24] Amit Sinha and Anantha P. Chandrakasan, “Energy Efficient Real-Time Scheduling”, IEEE Transactions on Software Engineering, Vol. 24, No. 2, 2001, pp. 458-463.

[25] Daniel P. Bovet and Marco Cesati, "Understanding the Linux Kernel", O'Reilly Online Catalogue, 2000.

[26] Ali Manzak and Chaitali Chakrabarti, “Variable Voltage Task Scheduling Algorithms for Minimizing Energy”, IEEE Transactions on Very Large Scale Integration Systems, Vol. 11, No. 2, 2003, pp. 270-276.

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

[28] Michael L. Pinedo, “Scheduling: Theory, Algorithms, and Systems”, Springer, Third Edition

[29] 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, 2008, pp. 1540-1552.

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

[31] A. Burns, R.I. Davis and· P. Wang · F. Zhang, “Partitioned EDF scheduling for multiprocessors using a C = D task splitting scheme”, Springer Science+Business Media, 2011.

[32] Wann Yun Shieh and Bo Wei Chen, “Energy-Efficient Tasks scheduling Algorithm for Dual-core Real-time Systems”, IEEE Transactions on Computers, Vol. 24, No. 2, 2001, pp. 978-985.