IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719
Vol. 3, Issue 6 (June. 2013), ||V2|| PP 44-58

Scheduling of Real Time Tasks

Pankaj Udvanshi, Ajay Kakkar
Research Scholar, Thapar University, Patiala
Assistant Professor, Thapar University, Patiala

Abstract: **-** Scheduling refers to a set of policies and mechanisms supported by operating system that controls the order in which the work to be done is completed. Scheduling is the best method to improve the performance of the network by re-arranging the system parameters. The basic of need of scheduling is to keep the CPU busy as much as possible by executing a (user) process until it must wait for an event, and then switch to another process. This paper involves the various issues on which the performance of the network depends.

Key Words: *Scheduling, Network performance, efficiency, throughput.*

I. INTRODUCTION

A scheduler is an operating system program (module) that selects the next job admitted for execution. The main objective of scheduling is to increase CPU utilization and higher throughput. Throughput is the amount of work accomplished in a given time interval. CPU scheduling is the basis of operating system which supports multiprogramming concepts. This mechanism improves the overall efficiency of the computer system by getting more work done in less time. Scheduling refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. Of all the resources in a computer system that are scheduled before use, the CPU is by far the most important. Multiprogramming is the (efficient) scheduling of the CPU. Processes alternate between consuming CPU cycles (CPU-burst) and performing (input/output-burst)

II. NEED OF SCHEDULING

In multiprogramming system, when there is more than one variable process, operating system must decide which one is activated. This design is made by the part of scheduling. In multiprogramming systems, when there is more than one runable process (i.e., ready), the operating system must decide which one to activate. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm.

• In the beginning—there was no need for scheduling, since the users of computers lined up in front of the computer room or gave their job to an operator.
• Batch processing—the jobs were executed in first come first served manner.
• Multiprogramming—life became complicated! The scheduler is concerned with deciding policy, not providing a mechanism.

The scheduling is basically used to cure the deadlock which occurs in the system. So scheduling is very important in the multiprogramming system.

III. TYPES OF SCHEDULING

There are many types of scheduling:

(a) Long -Term Scheduling

This type of scheduling is performing when the new process is design. If the number of processes in the ready queue is very high, then there be overhead on operating system. For maintaining long list, contact switch and dispatching increases; therefore, allows only limited number of process into ready queue.

Long -term scheduler determine which program are admitted into the system for the process. Once when admit a program, it become process and is added to the queue for short – term scheduling. It may also be very important that the long term scheduler should take a careful selection of process, i.e. process should be a combination of CPU and I/O bound types. Generally, most processes can be put into any of the two categories: CPU bound or I/O bound. If all process is I/O bound, the ready queue will always be empty and the short term scheduler will have nothing to do. If all process are CPU bound, no process will be waiting for I/O operation and again the system will be unbalanced. therefore , the long term scheduler provide good performance by selecting a combination of CPU bound and I/O bound processes. In some system, a newly created process begins in a swapped – out condition in which case it is added to a queue for the medium term scheduler.

(b) Medium -Term Scheduling

This type of scheduling is the part of swapping function. Medium -term scheduling may decide to swap out to process which has not been activated for some time. Saving of the suspended process is said to be swapped out or rolled out. The process is swapped in and swapped out by the medium term scheduler. The medium term scheduler has nothing to do with the suspended processes. But the moment the suspending condition is fulfilled, the medium term scheduler get activated to allocate the memory and swapped in the process and make it ready for execution. The medium term scheduling is also helpful for reducing the degree of multiprogramming, when long term scheduler is absent or minimal.

(c) Short –Term Scheduling

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

IV. SCHEDULING CRITERIA

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

(a) CPU Utilization

The key idea is that if the CPU is busy all the time, the utilization factor of all the components of the system will be also high. CPU utilization may range from 0 to100 percent.

(b) Throughput

It refers to the amount of work completed in a unit of time. One way to measure throughput is by means of the number of processes that are completed in a unit of time. The higher the number of processes, the more work apparently being done by the system. But this approach is not very useful for comparison because this is dependent on the characteristics and resource requirement of the process being executed. Therefore to compare throughput of several scheduling algorithms it should be fed the process with similar requirements.

(c) Turn-around Time

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

(d) Waiting Time

In a multiprogramming operating system several jobs reside at a time in memory. CPU executes only one job at a time. The rest of jobs wait for the CPU. The waiting time may be expressed as turnaround time less the actual processing time, i.e.

- Waiting time = turnaround time - processing time.

But the scheduling algorithm affects or considers the amount of time that a process spends waiting in a ready queue (the CPU-scheduling algorithm does not affect the amount of time during which a process executes or does (input/output). Thus rather than looking at turnaround time waiting time is the sum of the periods spent waiting in the ready queue.

(e) Response Time

It is most frequently considered in time sharing and real time operating systems. However it's characteristics differs in the two systems. In time sharing system it may be defined as interval from the time the last character of a command line of a program or transaction is entered to the time the last result appears on the terminal. In real time system it may be defined as interval from the time an internal or external event is signalled to the time the first instruction of the respective service routine is executed.

One of the problems in designing schedulers and selecting a set of its performance criteria is that they often conflict with each other. For example, the fastest response time in time sharing and real time system may result in low CPU utilization. Throughput and CPU utilization may be increased by executing large number of processes, but then response time may suffer. Therefore, the design of a scheduler usually requires a balance of all the different requirements and constraints.

V. LITERATURE SURVEY

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

Yang-ping Chen et. al. [1] proposed in 2007 on A Novel Task Scheduling Algorithm for Real-Time Multiprocessor Systems .They presents a novel task scheduling algorithm for real-time multiprocessor systems, which takes task's height and particle's position as the task's priority values, and applies the list scheduling strategy to generate the feasible solutions. Simulation results demonstrate that the proposed algorithm, compared with genetic algorithm produces encouraging results in terms of quality of solution and time complexity.

Jian-Jia Chen Chuan-Yue Yang et .al. [5] is proposed in 2008 on Approximation Algorithms for Multiprocessor Energy-Efficient Scheduling of Periodic Real-Time Tasks with Uncertain Task Execution Time. They derive a task partition which minimizes the expected energy consumption for completing all the given tasks in time. We give an efficient 1.13-approximation algorithm and a polynomial-time approximation scheme (PTAS) to provide worst-case guarantees for the strongly NP-hard problem. Experimental results show that the algorithms can effectively minimize the expected energy consumption.

Lars Lundberg et. al. [2] proposed in 2008 on Slack-Based Global Multiprocessor Scheduling of A periodic Tasks in Parallel Embedded Real-Time Systems. They prove that if the load on the multiprocessor stays below (3 - root of 5) / 2 ≈38.197% , the server can accept an incoming a periodic task and guarantee that the deadlines of all accepted tasks will be met. This is better than the current state-of the-art algorithm where the priorities of light tasks are based on deadlines (the corresponding bound is in that case 35.425%). they have replaced the traditional priority assignment strategy for light tasks from shortest deadline to least slack first. It turns out that the worst-case behaviour of the slack based approach is better when there are three or more processors.

Yi-Hsiung Chao et. al. [7] in 2008 on Schedulability issues for EDZL scheduling on real-time multiprocessor systems. They disprove this conjecture and show that the utilization bound of EDZL is no greater than m $(1 - 1/e) \approx 0.6321m$, where $e \approx 2.718$ is the Euler's number. EDZL (Earliest Deadline first until Zero Laxity) is an efficient and practical scheduling algorithm on multiprocessor systems. It has a comparable number of context switches to EDF (Earliest Deadline First) and its schedulable utilization seems to be higher than that of EDF. Previously, there was a conjecture that the utilization bound of EDZL is $3m/4 = 0.75m$ for m processors.

Paulo Baltarejo Sousa et. al. [4] proposed in 2010 Implementing Slot-Based Task-Splitting Multiprocessor Scheduling. They discuss and propose some modifications to the slot-based task splitting algorithm driven by implementation concerns, and we report the first implementation of this family of algorithms in a real operating system running Linux kernel version 2.6.34. We have also conducted an extensive range of experiments on a 4-core multicore desktop PC running task-sets with utilizations of up to 88%. The results show that the behaviour of our implementation is in line with the theoretical framework behind it. They have conducted a range of experiments with a 4-core multi-core desktop PC utilized to 88% with real-time tasks executing empty for loops that took approximately 37 hours. In spite of the unpredictability of the Linux kernel we observed a good correspondence between theory and practice.

Sanjoy K. Baruah et. al. Member of IEEE are proposed in July 2003 on Rate Monotonic Scheduling on Uniform Multiprocessors. Rate-Monotonic Scheduling is the one of the most popular algorithm for the scheduling system of periodic real-time task. They obtained here the first nontrivial feasibility test for a static-priority scheduling algorithm that adopts a global approach to task allocation upon uniform multiprocessors. They also have studied the behaviour of Algorithm RM—one such previously defined static priority global scheduling algorithm—upon uniform multiprocessor platforms. They obtained simple sufficient conditions for determining whether any given periodic task system will be successfully scheduled by Algorithm RM upon a given uniform multiprocessor platform.

Peng Li, et. al. Proposed on Fast, Best-Effort Real-Time Scheduling Algorithms at September 2004 .They presents two fast, best-effort real-time scheduling algorithms called MDASA and MLBESA. MDASA and MLBESA are novel in the way that they heuristically, yet accurately, mimic the behaviour of the DASA and LBESA scheduling algorithms. For a highly bursty workload, MLBESA is found to perform worse than LBESA. The task response times under MDASA and MLBESA are very close to the values under their counterpart scheduling algorithms. They show in experimental results that MDASA and MLBESA perform almost as well as DASA and LBESA, respectively, unless the workload is highly burst and the system is heavily overloaded. Furthermore, the task response times under MDASA and MLBESA are found to be very close to the values under their counterpart scheduling algorithms. While MDASA performs better than MLBESA and has a better worst-case complexity, MLBESA guarantees the optimal schedule during under load situations.

Ming Xiong, et. al. Member IEEE, proposed on Deferrable Scheduling for Maintaining Real-Time Data Freshness Algorithms, Analysis, and Results in July 2008. They propose a deferrable scheduling algorithm for fixed-priority transactions, a novel approach for minimizing update workload while maintaining the

temporal validity of real-time data. In contrast to prior work on maintaining data freshness periodically, update transactions follow an a periodic task model in the deferrable scheduling algorithm. The deferrable scheduling algorithm exploits the semantics of temporal validity constraint of real-time data by judiciously deferring the sampling times of update transaction jobs as late as possible. They present a theoretical estimation of its processor utilization and a sufficient condition for its Schedulability. In his experimental results were verify the theoretical estimation of the processor utilization. They demonstrate through the experiments that the deferrable scheduling algorithm is an effective approach and it significantly outperforms the More-Less scheme in terms of reducing processor workload. They presents a sufficient condition for its Schedulability. They also propose a theoretical estimation of the processor utilization of DS-FP, which is verified in our experimental studies. It is possible for the same concept to be used in the scheduling of update transactions with dynamic priority, for example, in the Earliest Deadline scheduling of update transactions.

Euiseong Seo, et. al. proposed on Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors in November 2008. They tackles the problem of reducing power consumption in a periodic real-time system using DVS on a multicore processor. The processor is assumed to have the limitation that all cores must run at the same performance level. To reduce the dynamic power consumption of such a system, They suggest two algorithms: Dynamic Repartitioning and Dynamic Core Scaling. The former is designed to reduce mainly the dynamic power consumption, and the latter is for the reduction of the leakage power consumption. In the assumed environment, the best case dynamic power consumption is obtained when all the processor cores have the same performance demand. Dynamic Repartitioning tries to maintain balance in the performance demands by migrating tasks between cores during execution accompanying with deadline guarantee. Leakage power is more important in multicore processors than in traditional unicore processors due to their vastly increased number of integrated circuits. Indeed, a major weakness of multicore processors is their high leakage power under low loads. To relieve this problem, Dynamic Core Scaling deactivates excessive cores by exporting their assigned tasks to the other activated cores.

Marko Bertogna, et.al. Member of IEEE proposed Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms in April 2009.They addresses the Schedulability problem of periodic and sporadic real-time task sets with constrained deadlines preemptively scheduled on a multiprocessor platform composed by identical processors.They assume that a global work-conserving scheduler is used and migration from one processor to another is allowed during a task lifetime. The analysis will be applied to two typical scheduling algorithms: Earliest Deadline First (EDF) and Fixed Priority (FP). Then, the derived Schedulability conditions will be tightened, refining the analysis with a simple and effective technique that significantly improves the percentage of accepted task sets. The effectiveness of the proposed test is shown through an extensive set of synthetic experiments. We developed a new schedulability analysis of real-time systems globally scheduled on a platform composed by identical processors. They presented sufficient Schedulability algorithms that are able to check in polynomial and pseudopolynomial time whether a periodic or sporadic task set can be scheduled on a multiprocessor platform.

Enrico Bini, et. al. Proposed A Response-Time Bound in Fixed-Priority Scheduling with Arbitrary Deadlines in February 2009.They identify in his research the three desirable properties of estimates of the exact response times, continuity with respect to system parameters, efficient computability, and approximability. They have argued that continuity with respect to system parameters is a very desirable property of response-time bounds if such bounds are to be used as an integral part of an incremental interactive design process. Accordingly, we have derived a continuous upper bound on response time for task systems that are scheduled upon a preemptive uniprocessor using the DM scheduling algorithm.

Fengxiang Zhang et. al. Proposed Schedulability Analysis for Real-Time Systems with EDF Scheduling on September 2009. They propose new results on necessary and sufficient Schedulability analysis for EDF scheduling; the new results reduce, exponentially, the calculation times, in all situations, for schedulable task sets, and in most situations, for unschedulable task sets. In this paper, we have addressed and solved the problem of providing fast Schedulability analysis which is necessary and sufficient for EDF scheduling with arbitrary relative deadlines. They saw that a number of factors can significantly affect the experimental results of the old methods; in some circumstances, they have exponential growth. The experimental results for QPA are similar for all kinds of task sets, and QPA reduces the required number of calculations exponentially in almost all situations.

From mention literature survey, it has been observed that there is a need to design and implement a scheduling algorithm in order to optimize the speed of processor. Furthermore, the algorithm also provides an experiment result for linear and non linear task in lead this environment. They also let solution related to dynamic problem. The comparison on the basis of performance measuring parameters of various algorithms has been shown in the following table 1.

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

Table 1: Comparison table for all types scheduling algorithm

Simulation result

There is a need to optimize the method of finding shortest path in order to save the resources for transmitting data from one node to the other. The results have been obtained for that problem.

Shortest path algorithms

A shortest path problem is the problem of finding path between the two vertices in such a way that the sum of the weights of its constituent edges is minimized. There are many variations depending on the directivity of the graph [7,10]. For undirected weighted graph (graph with a set of V vertices, a set of E edges and a real valued weight function f: E→R) and elements v and v' of V, find a path P from v to v' so that

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

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

Single source shortest path problem

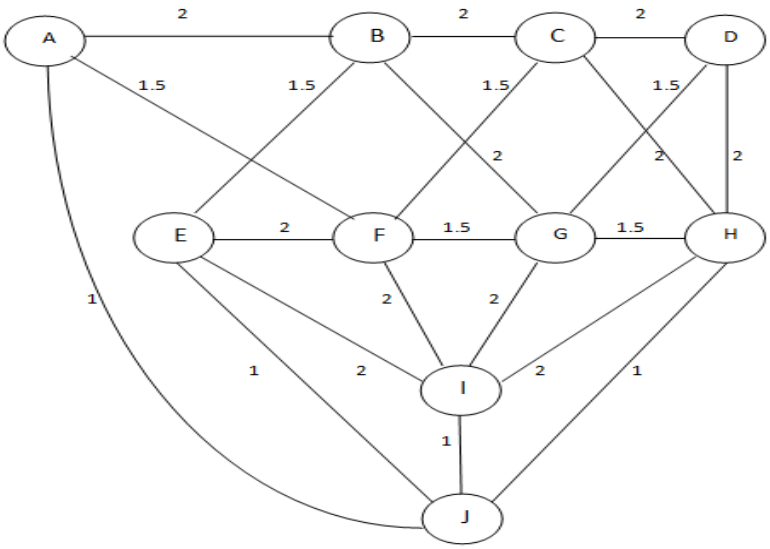
If a single source is used to transmit data to all other nodes then the problem is known as single source shortest path problem. This problem deals with finding the shortest path from a single source to all the nodes or vertices in the graph.

Single destinaton shortest path problem

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

All-pairs shortest path problem

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



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

Figure Undirected graph

On manual inspection of the undirected graph shown above, following path has been observed for single source shortest path problem. For the node J, following paths has been observed. For node A and B there are various paths to the other nodes and many nodes has to be traversed for reaching that node. For the transmission of the data to a particular node from A and B there can be multiple paths but we need that path in which there is minimum delay or the intermediate nodes are minimum.

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

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

From the results obtained above it has been observed that transmitting the data from node A to node D takes maximum time i.e. 4 ns and has 1 intermediate nodes. If another path has been taken from A to node D then time taken has been found greater than or equal to 4 ns. If we take path $A \rightarrow F \rightarrow G \rightarrow D$, it will take 4.5 ns and number of intermediate nodes will be 2 only. Hence there will not be any change in the terms of intermediate nodes but the time taken has been reduced by,

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

So shortest paths to all the nodes has been found which are helpful in resolving timing constraints and saving of resources.

The following result is for the single destination problem from the source C and D be the source. All the shortest paths has been found for all other nodes.

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

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

It has been observed that it takes maximum amount of time to reach node I, from node C which is 3.5 ns and 1 intermediate nodes are transversed. If we choose any other path like $c \rightarrow H \rightarrow I$, then data will take 4 ns to reach I from C and intermediate nodes will remain 1 only. Hence, the time needed is reduced by,

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

In this case, there is no change in the number of nodes traversed in order to reach node I from C but the time required to reach that node is reduced by 12.5%.

The node E and F connects all other nodes through the chain of intermediate nodes. The following result describes the shortest path from node E and F to all other nodes.

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

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

It has been observed that it takes maximum time to reach node D from node $E$and time is $4\,ns$. There are 2 intermediate nodes. If any other path is taken to node D from node E which is $E \rightarrow B \rightarrow C \rightarrow D$, the time needed to reach node D from E is $4.5\,ns$ and intermediate nodes to be traversed is same 2. The time needed has been reduced by,

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

the time has been reduced by 11.11%.

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

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

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

From the result, it has been observed that it takes maximum time to reach node B from node H which is $3.5\,ns$. The intermediate nodes which has to be crossed for node H to node B is 1 . If any other path has been taken from node H , $H \rightarrow D \rightarrow C \rightarrow B$, the time taken to reach node B from node H will be $6\,ns$ and there were 2 intermediate nodes. Thus the time needed with our selected path is reduced by,

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

Thus we have seen that there is change in the intermediate nodes and the time taken is reduced by 41.66%.

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

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

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

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

From the results shown, it has been observed that it takes maximum time to reach node B from node I and that is 3.5 ns. The intermediate nodes which is needed to be crossed on this path is only 1. If the path chosen to reach node B from node I other than our optimized path, $I \rightarrow G \rightarrow B$, then the time taken is 4 ns and number of intermediate nodes has been same. optimized path as compared to the other arbitrary path taken is reduced by,

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

When one node is failed then what effect on data transsmission,we will see that by our manual inspection and simulation result. We assume that the node B is failed in our graph, then we again we find all the shortest path from one node to all other node.

First all the shortest path from node A and B to all other node is show in the table 5.1

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

.**Table 5.1 Shortest paths from node A and B to all other nodes**

All the shortest path from node C and D to all other node is show in the table 5.2

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

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

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

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

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

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

S.No.	Node	Path	Delay	Intermediate Nodes
1.	A	$A \xrightarrow{1} J$	1	0
2.	B	Infinite	–	–
3.	C	$C \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$C \xrightarrow{2} D \xrightarrow{2} H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	8.5	4
		$C \xrightarrow{2} D \xrightarrow{2} H \xrightarrow{1} J$	5	2
		$C \xrightarrow{2} H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	5.5	3
4.	D	$D \xrightarrow{2} H \xrightarrow{1} J$	3	1
		$D \xrightarrow{1.5} G \xrightarrow{1.5} H \xrightarrow{1} J$	4	2
		$D \xrightarrow{2} C \xrightarrow{2} H \xrightarrow{2} I \xrightarrow{1} J$	7	3
5.	E	$E \xrightarrow{1} J$	1	0
		$E \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$E \xrightarrow{2} I \xrightarrow{2} H \xrightarrow{1} J$	5	2
6.	F	Infinite	–	–
7.	G	$G \xrightarrow{1.5} H \xrightarrow{1} J$	2.5	1
		$G \xrightarrow{2} I \xrightarrow{1} J$	3	1
		$G \xrightarrow{1.5} D \xrightarrow{2} H \xrightarrow{1} J$	4.5	2
8.	H	$H \xrightarrow{1} J$	1	0
		$H \xrightarrow{2} I \xrightarrow{1.5} J$	2.5	1
		$H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{1} J$	4.5	2
		$H \xrightarrow{1.5} G \xrightarrow{2} I \xrightarrow{2} E \xrightarrow{1} J$	6.5	3
9.	I	$I \xrightarrow{1} J$	1	0
		$I \xrightarrow{2} E \xrightarrow{1} J$	3	1
		$I \xrightarrow{2} G \xrightarrow{1.5} H \xrightarrow{1} J$	4.5	2
		$I \xrightarrow{2} G \xrightarrow{1.5} D \xrightarrow{2} H \xrightarrow{1} J$	6.5	3

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

Table 5.4 Shortest paths from node G and H to all other nodes

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

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

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

when the two node are failed simultanuously than what will be the effect on our data tranmission we show it in our manual and simulation.Here we take node B and F in failuar condition

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Table 6.5 Shortest paths from node I and J to all other nodes

VI. CONCLUSION AND FUTURE SCOPE

The literature survey on has been carried out by considering the various parameters such as processing speed, scheduling algorithms, etc. The observations from the literature survey has been stated in the section 4 which clearly highlights that the performance of the network to handle real time tasks can be enhanced by employing the scheduling algorithms. Various soft computing tools can be used for optimization.

REFERENCES

[1] Yang-ping Chen, Lai-xiong Wang, and Shi-tan Huang Xi'an "A Novel Task Scheduling Algorithm for Real-Time Multiprocessor Systems" 2007 IEEE International Conference on Control and Automation Guangzhou, China.

[2] Lars Lundberg and Hakan Lennerstad Department of Systems and Software, School of Engineering, Blekinge Institute of Technology, 372 25 Ronneby, Sweden'' Slack-Based Global Multiprocessor Scheduling of Aperiodic Tasks in Parallel Embedded Real-Time Systems".

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

[4] Paulo Baltarejo Sousa , Bjorn Andersson and Eduardo Tovar ''Implementing Slot-Based Task-Splitting Multiprocessor Scheduling" IEEE 2011.

[5] operating system internals and design principal, fifth edition by William stallings

[6] K. Frazer, "Real-time Operating System Scheduling Algorithms", 1997.

[7] C. M. Krishna and K. G. Shin, "Real-Time Systems," MIT Press and McGraw-Hill Company,1997.

[8] Yi-Hsiung Chao, Shun-Shii Lin, Kwei-Jay Lin "Schedulability issues for EDZL scheduling on real-time multiprocessor systems" Received 18 November 2006; received in revised form 31 January 2008. Available online 15 March 2008 Communicated by A.A. Bertossi. vol. No. , 2008 pp.158–164.

[9] Sanjoy ,Baruah and Joel Goossens ''Rate-Monotonic Scheduling on Unifor Multiprocessors" IEEE Transaction On Computer, Vol. 52, No. 7, July2003

[10] Peng Li and Binoy Ravindran, Senior Member, IEEE On "Fast, Best-Effort Real-Time Scheduling Algorithms" IEEE Transaction On Computer, Vol. 53, No. 9, September 2004.

[11] Ming Xiong, Song Han, , Kam-Yiu Lam, and Deji Chen, Member, IEEE "Deferrable Scheduling for Maintaining Real-Time Data Freshness: Algorithms, Analysis, and Results" IEEE Transaction On Computers, Vol. 57, No. 7, July 2008.

[12] Euiseong Seo, Jinkyu Jeong, Seonyeong Park, and Joonwon LeeEnergy ''Efficient Scheduling of Real-Time Tasks on Multicore Processors" IEEE Transaction on Parallel and Distribution System Vol. 19, No. 11, November 2008.

[13] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari, Member, IEEE "Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms" IEEE Transaction On Parallel And Distributed Systems, Vol. 20, No. 4, April 2009.

[14] Enrico Bini, Thi Huyen Chaˆu Nguyen, Pascal Richard, and Sanjoy K. Baruah "A Response-Time Bound in Fixed-Priority Scheduling with Arbitrary Deadlines" IEEE Transaction On Computer, Vol. 58, No. 2, February 2009.

[15] Fengxiang Zhang and Alan Burns, Senior Member, IEEE "Schedulability Analysis for Real-Time Systems with EDF Scheduling" IEEE Transaction on Computers, Vol. 58, No. 9, September 2009.