COMPUTOLOGY: Journal of Applied Computer Science and Intelligent Technologies
Volume 1, Issue 1, Jan-Jun 2021, pp. 1-12
DOI: 10.17492/computology.v1i1.2101
https://www.journalpressindia.com/cjacsit
© 2021 Journal Press India

A Study of the Performance of Linearly Extensible Multiprocessor Architecture

*Mukul Varshney**

ABSTRACT

A number of design techniques for multiprocessing architectures have been investigated as a result of various developments in IC processing and integrating technologies. While designing massively parallel computer systems, the choice of the interconnection network's topology is one of the many crucial design concerns. And as a result, there have been numerous proposals for interconnection networks in the literature, and a ton of study has been done on the creation and evaluation of these interconnection networks. However, the issue of integrating the processing components in multiprocessing parallel architectures in order to achieve great computational efficiency has not yet been fully solved. In order to effectively manage parallelism on an interconnection network, it is necessary to maximize a number of competing performance indicators, such as reducing communication and scheduling overheads and distributing workloads evenly. In order to reduce communication cost, load balancing entails distributing work to each processor in proportion to its performance. The assignment may be completed statically at compile time or dynamically at run time. Many load balancing policies boost system performance by using more processing power, memory, or a combination of the two. The present work is centered on implementation of two existing dynamic load balancing schemes –*Sender Initiated Diffusion* (SID) and *Receiver Initiated Diffusion* (RID) to the Linearly Extensible Multiprocessor (LEM) architecture-The results achieved in the simulation are presented to evaluate the performance of LEM architecture.

Keywords: LEC; LET; SID; Load Balancing; Time; Load Imbalance Factor; Ideal Load; Linearly Extensible Triangle.

1.0 Introduction

A multiprocessor system is a collection of several separate processors that have been grouped together so that they can cooperate to tackle a given complicated task more quickly. There are essentially two different ways for using several processors: shared memory and message forwarding systems. The shared memory concept offers a global memory that is shared by all of the system's processors and has a single address space. Message-passing models, on the other hand, have several address spaces and provide every CPU access to their own local memory. There are several methods and approaches for organizing processes on a system to improve performance. The load-balancing, load-sharing, and work assignment approaches are a few of them. Each step in the job assignment approach is seen as a collection of linked [9, 12].

We refer to an increase in parallelism in the architecture as the addition of extra processors to the system. Increased parallelism will inevitably result in higher communication costs.

**Department of Computer Science & Engineering, Sharda University, Greater Noida, Uttar Pradesh, India (E-mail: Mukul_varshney02@yahoo.com)*

These overheads include message traffic density, inter-node distance, knowledge overhead, and fault tolerance. All of these rely on the network's diameter and the node's position within it. [2-7]. Many interconnection network architectures, such as the Linearly Extensible Tree (LET) network, Linearly Extensible Cube (LEC), and Linearly Extensible Triangle, have been developed in the pursuit of creating effective parallel multiprocessing systems. [2, 6].

These architectures only have 6 processors, as opposed to the hypercube or de-Bruijn architecture's 8 processors. Sender Initiated Diffusion (SID), a dynamic scheduling technique, has demonstrated that the LE [7] performs well with various designs [10]. Compared to the remaining comparable networks, LEC is functioning on par with or even better. The aforementioned designs imitate Sender initiated Diffusion (SID), [8, 9]. In this research, we simulate dynamic load balancing techniques known as SID onto several linearly extensible multiprocessor architectures with various features and schedule the incoming load on various architectures to provide user access to data in the smallest amount of time.

2.0 Related Work

(i) Different Linearly extensible Multiprocesor Interconnection network

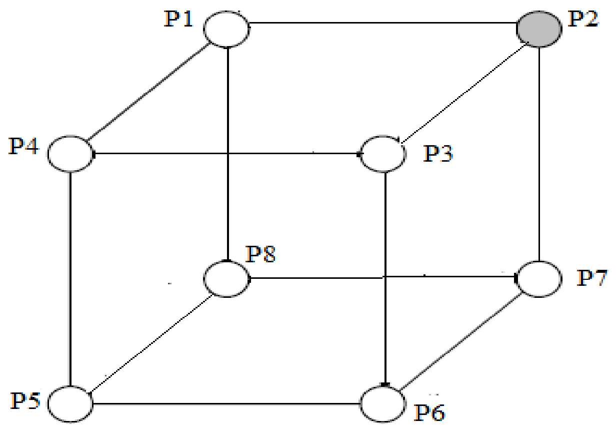
Numerous distinct linearly extensible multiprocessor architectures with various topological structures and interconnections have been developed during the past year. Numerous studies have focused on the design and connections between them. In the literature, a variety of multiprocessor architectures have been published.LET and LEC networks, ring networks, hypercubes, Debruijn networks, and so on [6,7,8].[1,3,4,7].

Most advantageous use of multiprocessor networks To store and retrieve data on the internet, a web server is utilized. A switch or router receives a web request (URL) and routes it to the web server in accordance with scheduling algorithms that minimize scheduling overhead and load balancing time. Scheduling overhead essentially addresses both processor and communication overhead. The communication costs are part of the load balancing overhead.

(ii) Design and Characteristics of Various Existing Linearly Extensible Interconnection network for Multiprocessor

A. Base Networks

(i) Hypercube



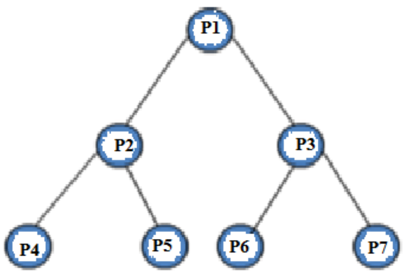
Adjacency Matrix of Interconnection

	P1	P2	P3	P4	P5	P6	P7	P8
P1	0	1	0	1	0	0	0	1
P2	1	0	1	0	0	0	1	0
P3	0	1	0	1	0	1	0	0
P4	1	0	1	0	1	0	0	0
P5	0	0	0	1	0	1	0	1
P6	0	0	1	0	1	0	1	0
P7	0	1	0	0	0	1	0	1
P8	1	0	0	0	1	0	1	0

Characterstics of Hypercube

Network	No. of Link	Node degree.	Diameter	Bisection Width
Hypercube	$NLog_2N/2$	N	Log_2N	N/2

(ii) Tree



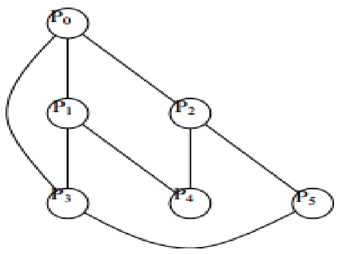
Adjacency Matrix of Interconnection

	P1	P2	P3	P4	P5	P6	P7
P1	0	1	1	0	0	0	0
P2	1	0	0	1	1	0	0
P3	1	0	0	0	0	1	1
P4	0	1	0	0	0	0	0
P5	0	1	0	0	0	0	0
P6	0	0	1	0	0	0	0
P7	0	0	1	0	0	0	0

Network	No. of Link	Node degree.	Diameter	Bisection Width
Binary Tree	N-1	3	$2(Log_2N -1)$	1

B. Linearly Extensible Interconnection

(i) Linerly Extensible Tree with 6 Nodes

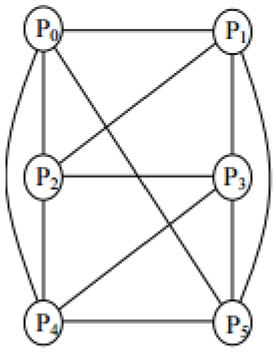


Adjacency Matrix of Interconnection

	P0	P1	P2	P3	P4	P5
P0	0	1	1	1	0	0
P1	1	0	0	1	1	0
P2	1	0	0	0	1	1
P3	1	1	0	0	0	1
P4	0	1	1	0	0	0
P5	0	0	1	1	0	0

Network	No. of Link	Node degree.	Diameter	Bisection Width
LET	N+2	4	√N	2

(ii) Linearly Extensible Cube with 6nodes

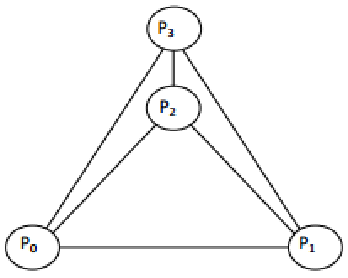


Adjacency Matrix of Interconnection

	P0	P1	P2	P3	P4	P5
P0	0	1	1	0	1	1
P1	1	0	1	1	0	1
P2	1	1	0	1	1	0
P3	0	1	1	0	1	1
P4	1	0	1	1	0	1
P5	1	1	0	1	1	0

Network	No. of Link	Node degree.	Diameter	Bisection Width
LEC	$(N/2)^2+3$	4	N	N

(iii) Linearly Extensible Triangle (LEΔ) with 5 nodes



Adjacency Matrix of Interconnection

	P0	**P1**	**P2**	**P3**
P0	0	1	1	1
P1	1	0	1	1
P2	1	1	0	1
P3	1	1	1	0

Network	**No. of Links**	**Node degree.**	**Diameter**	**Bisection Width**
LEΔ	N=∑k+1	N-1	2	N+1

3.0 Load Balancing

In the load balancing strategy, tasks are evenly distributed throughout the system nodes in order to balance burden. Because it is impossible for each processor to work for the same length of time during parallel computing, it is exceedingly difficult to divide the workload among the processors. Some processors finish their duties earlier than others, leaving them idle while the others work. Therefore, minimizing LIF, communication, and processing overhead is a component of load balancing. As a result, we need effective dynamic load balancing techniques. There are several dynamic load balancing strategies that have developed. We examine Sender Initiated Diffusion (SID), a dynamic load balancing technique, in this study. There are three performance assessment criteria for load balancing: the threshold load (also known as the ideal load), the load imbalance factor (LIF), and the load balancing duration.

SID is divided into four phases:

1. Identification of processor load (under loaded & overloaded)
2. Determination of load balancing gain
3. Load Selection method
4. Load Migration Technique

4.0 Sender Initiated Diffusion (SID) Algorithm

```
//phase 2: processor load migration profitability determination

id_load=load/n;
for(j=0;j<n;j++){
        while(processor[l]>id_load){
                        processor[j]++;
                        processor[l]--;}

If(pro>7)
                        {
                                        for(z=0;z<n;z++){
                        if(pro[s][z]==1){
                        adjancy[y]=z;
                        y++;}}
x=y;
for(z=0;z<x;z++){
for(w=0;w<n-1;w++){
path[z][w]=0;}}
```

```
// phase 3: task selection
for(y=0;y<x;y++){
        count=0;
        z=s;
        w=adjency[y];
        l=0;
        while(w!=d+1){
                if(pro[z][w]==1){
                path[y][l]=w;
                z=w;
                w=z+1;
                        delay(1);
                if(w==s)
                w=w+1;
                count++;
                l++;}
                else
                w++;}
        load[y]=count;}
```

```
//phase 4: task migration
min=c[0];
 for(z=1;z<x;z++){
if(c[z]<min){
delay(1);
min=c[z];
m=z;}}
if(min==0){
m=m-1;
min=c[m];}
                for(y=0;y<min;y++) {
                b=processor[path[m][y]]+id_load;
while(processor[path[m][y]]<b&&processor[s]>id_load){
                processor[s]--;
                processor[path[m][y]]++;
                delay(1);}
                                s=path[m][y]; }}}}}}
max=p[0];
for(j=0;j<n;j++){
if(p[j]>max)
max=p[j];}
```

5.0 Simulation Results

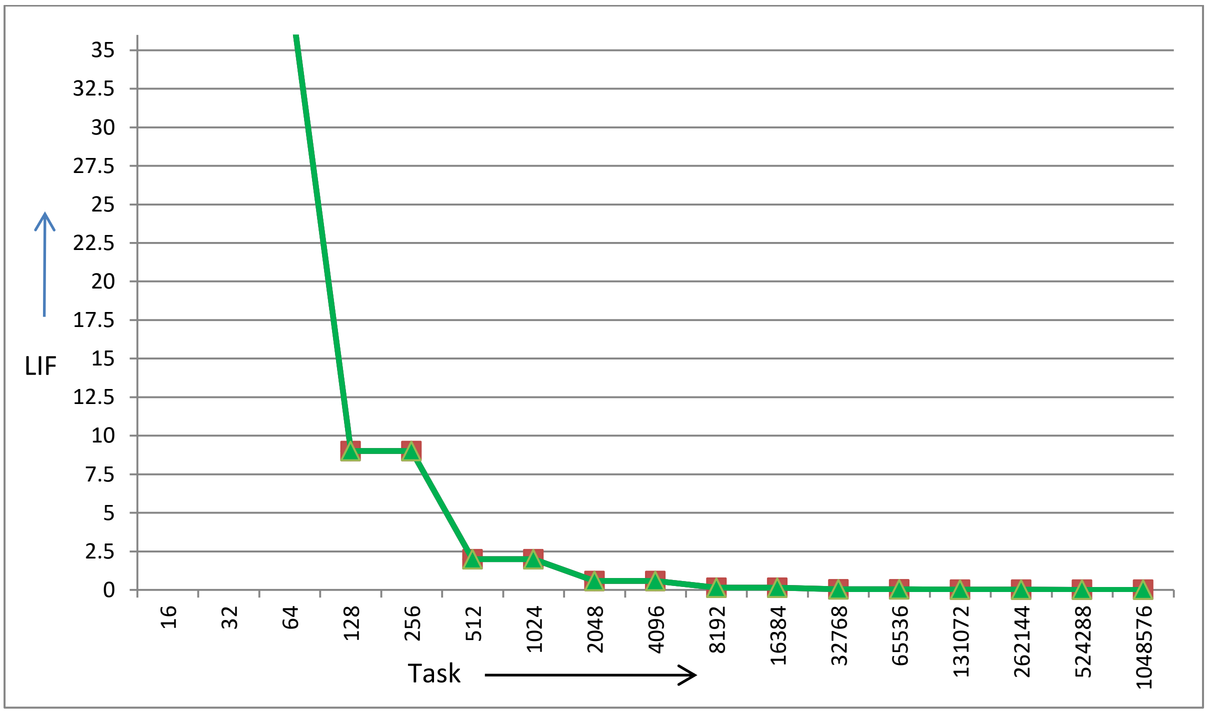
We utilize two performance parameters to assess the effectiveness of several currently existing linearly extendable interconnections: a) the load imbalance factor (LIF), and b). Balance of Loads Time

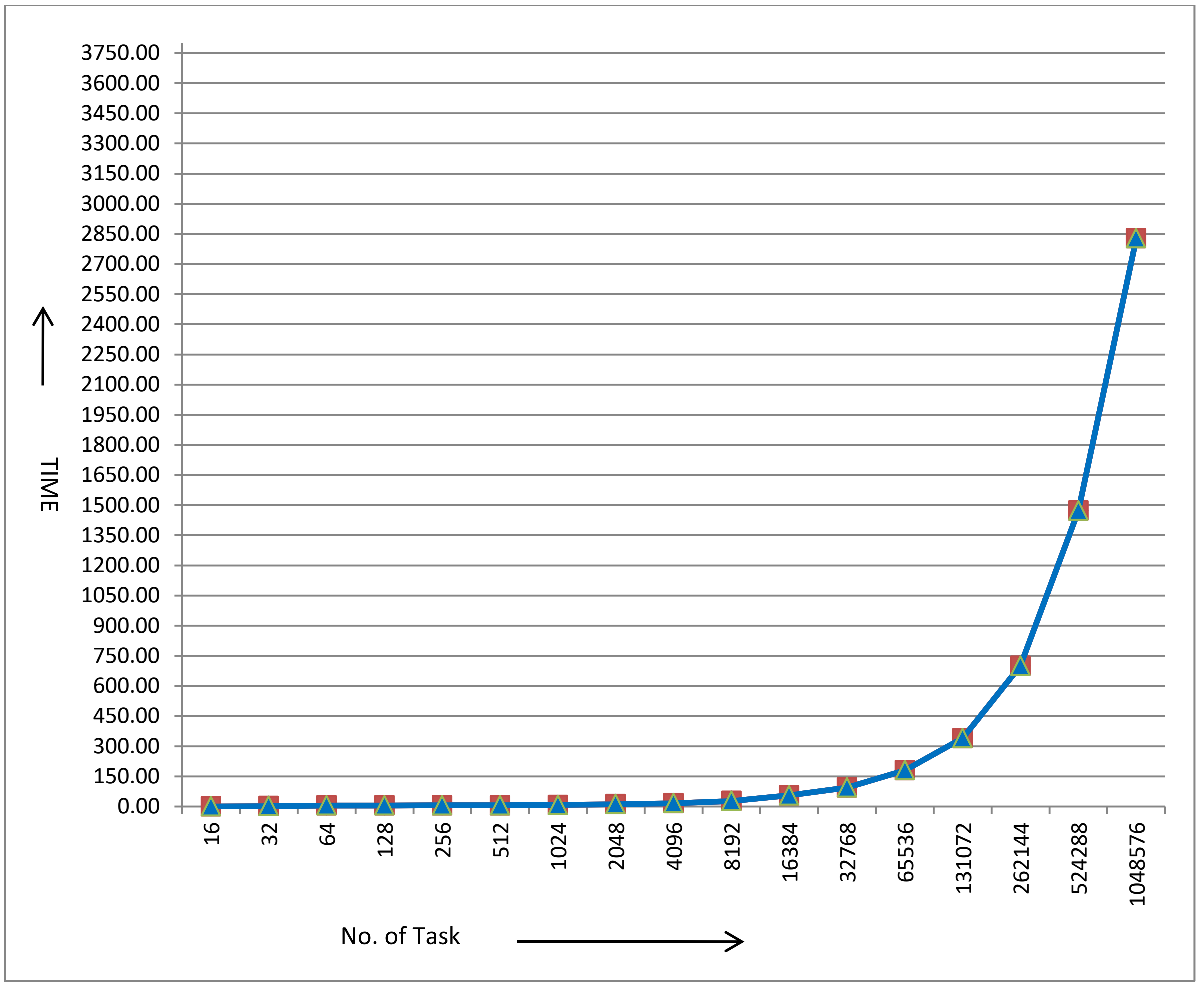
- The LIF (Load Imbalance Factor) is equal to [((max load on a processor after balancing -Ideal Load)/Ideal Load]].*100
- The load balancing time is equal to the difference between the before and after balancing times.

The table and graph below in [Fig....] indicate the time taken by various linearly expandable multi-processor architectures under various samples of load. By simulating a dynamic load balancing technique known as SID, we will compare the performance of several linearly extensible multiprocessor architectures in our work.

No of task	LIF	TIME
16	**199**	**0.00**
32	39	1.97
64	39	4.98
128	9	4.98
256	9	6.01
512	1.99	6.01
1024	1.99	6.98
2048	0.58	10.99
4096	0.58	15.96
8192	0.14	27.02
16384	0.14	54.99
32768	0.04	94.01
65536	0.04	179.99
131072	0.01	339.97
262144	0.01	700.01
524288	0	1471.96
1048576	0	2827.98

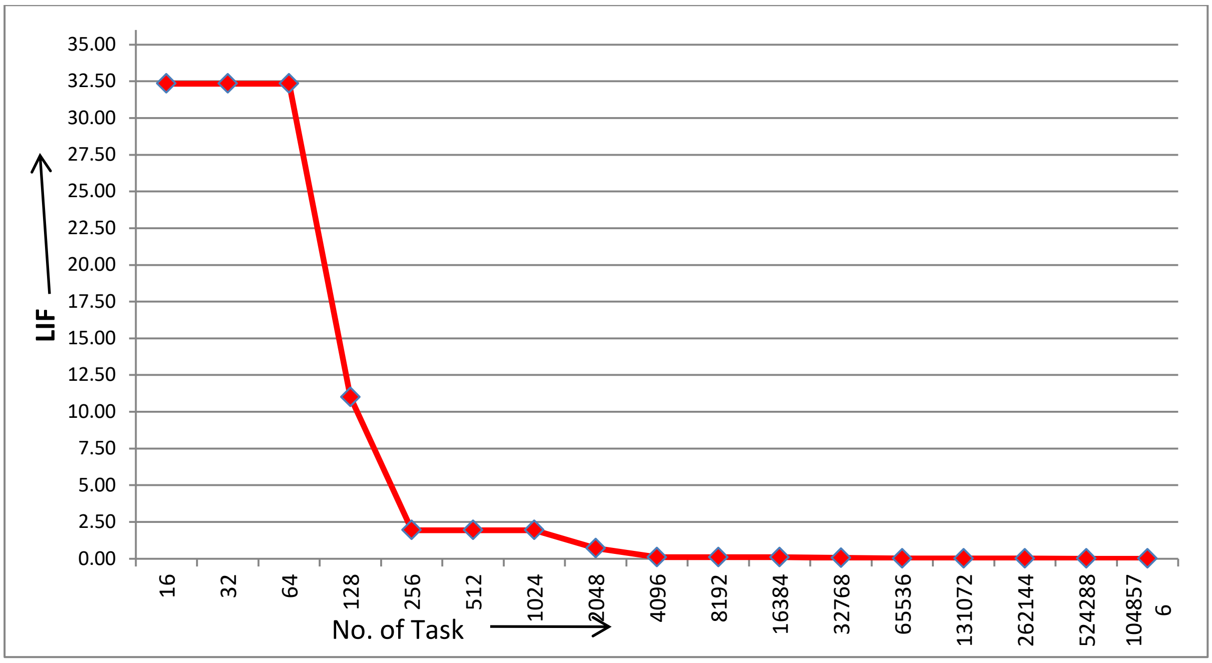
LEC LIF VS TIME

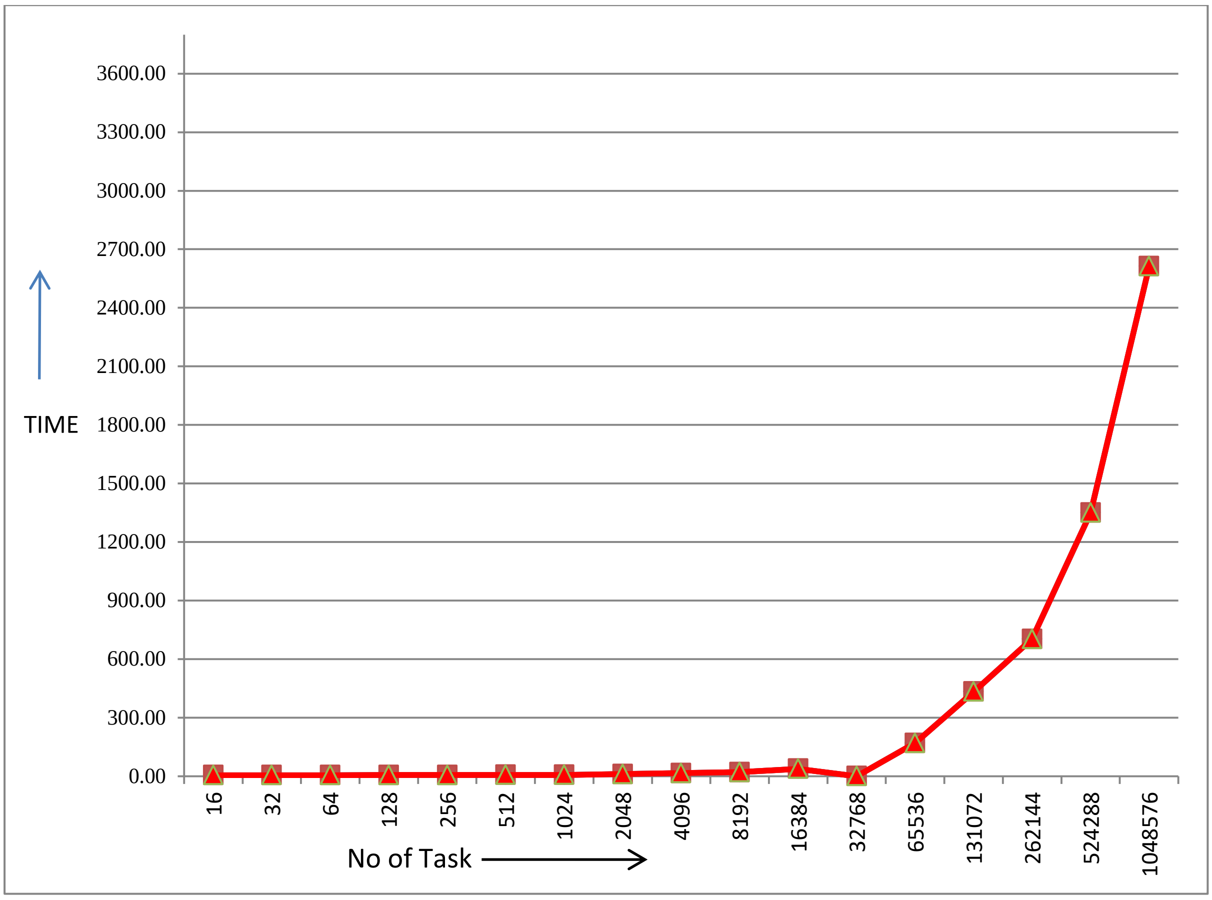




No of task	LIF	TIME
16	**32.33**	**5.00**
32	32. 33	5.01
64	32.33	5.01
128	11.00	5.98
256	1.94	5.98
512	1.94	6.99
1024	1.94	6.98
2048	0.71	10.97
4096	0.10	15.98
8192	0.10	21.01
16384	0.10	37.98
32768	0.05	76..96
65536	0.01	170.01
131072	0.01	433.01
262144	0.01	702.99
524288	0.00	1349.92
1048576	0.00	2612.95

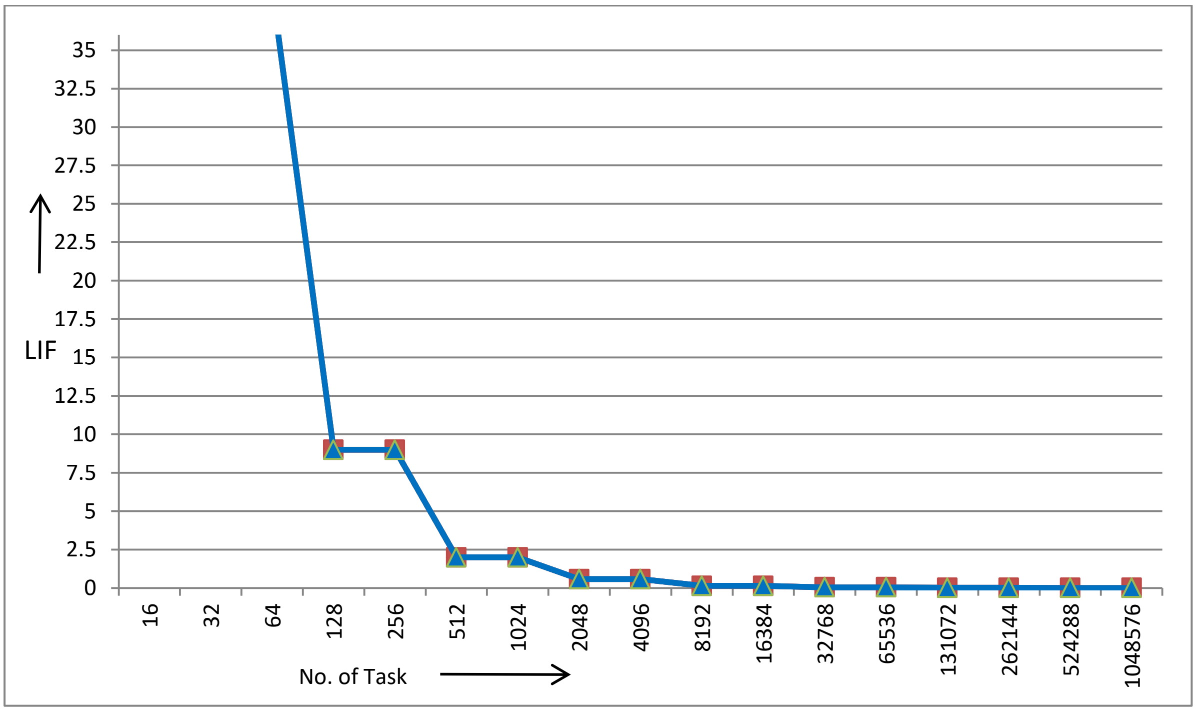
LEΔ LIF VS TIME

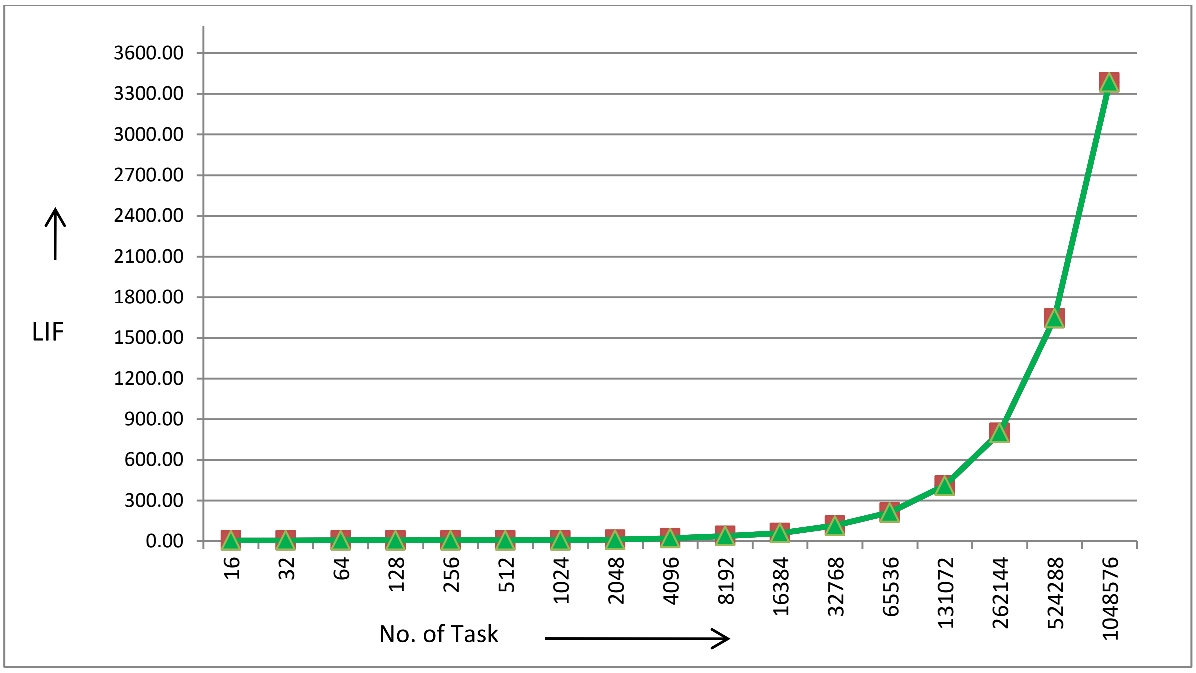




No of task	**LIF**	**TIME**
16	**199**	**4.98**
32	39	4.98
64	39	5.95
128	9	5.95
256	9	6.94
512	1.99	6.94
1024	1.99	6.98
2048	0.58	11.01
4096	0.58	21.98
8192	0.14	37.97
16384	0.14	59.93
32768	0.04	115.01
65536	0.04	211.96
131072	0.01	409.99
262144	0.01	800.02
524288	0	1644.97
1048576	0	3383.97

LET LIF VS TIME





6.0 Conclusion

By modeling Sender Initiated Diffusion (SID), a dynamic load balancing method, we have assessed the performance of several linearly expandable interconnection networks such as LEC, LET, and LE.We have described our model as a Linearly Extensible Tree (LET) network with the SID Method dynamic scheduling technique [13]. As opposed to the hypercube or de-Bruijn architecture, which has 8 processors, this design has 6 processors. It has been demonstrated that the LET is functioning well with various architectures using the dynamic scheduling approach known as SID [6]. It has also been stated that a Linear Extensible Cube (LEC) network exists [1,2,7]. The characteristics of LET and hypercube networks are combined in this LEC. The LEC is judged to be performing on par with or even better than the remaining comparable networks. An additional linear extensible

References

1. Abdus Samad & Qasim Rafiq ―A Novel architecture LEC for Network Sever‖, Ph.D, 2010
2. Rafiq M.Q., M. Ba-Ru-K & A.Samad, ― A Linearly Extensible Cube Network‖ sent for publication in IEEE, Jan. 2001
3. Rafiq M.Q., M. Ba-Ru-K & A.Samad, ― A Linearly Extensible Cube Network‖ sent for publication in IEEE, Jan. 2001
4. Abdallah Boukerram, Samira Ait Kaci Azzou, "Implementation of Load Balancing Algorithm in a Grid Computing", American Journal of Applied Sciences, 2006.
5. Manaullah, "Performance Evaluation of Multiprocessor Architectures", Ph.D. thesis, Jamia Millia Islamia, 2002.
6. Abdus Samad, "Performance Evaluation of Linearly Extensible Multiprocessor Architectures for Networking", Ph.D. thesis, Aligarh Muslim University, 2009.
7. M. Q. Rafiq, "Studies on the Performance Evaluation of a Linearly Extensible Multiprocessor Network", Ph.D thesis, Univ. of Roorkee, 1995.

8. D. Acker, S. Kulkarni, "A Dynamic Load Dispersion Algorithm for Load-Balancing in a Heterogeneous Grid System", Sarnoff Symposium IEEE, pp 1- 5, 2007.
9. A. Chhabra, G. Singh, "Qualitative Parametric Comparison of Load Balancing Algorithms in Distributed Computing Environment", 14th International Conference on Advanced Computing and Communication, IEEE, pp 58– 61, 2006.
10. Neeraj K., "Simulation Study for Performance and Prediction of Parallel Computers," BVICAM's International Journal of Information Technology, (BIJIT) , vol. 4, 2012
11. Abdus Samad, Jamshed Siddiqui "Properties and Performance of Cube-based Multiprocessor Architectures" IAJIT international journal, vol-3, 2014
12. Marc H. Willebeek-LeMair and Anthony P. Reeves "Strategies for Dynamic Load Balancing on Highly Parallel Computers" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 4, NO. 9, SEPTEMBER 1993