A Clustered Approach for Load Balancing inDistributed Systems

Mrs. Geetmala[1], Dr. Neelendra Badal[2], Dr. Shri Om Mishra[3]

[1]Assistant Professor, Department of Computer Science and Engineering, Feroze Gandhi Institute of Engineering and Technology, Raebareli, India
[2]Professor and Head, Department of Computer Science and Engineering, Kamla Nehru Institute of Technology, Sultanpur, India
[3]Assistant Professor, Department of Electronics & Communication Engineering, IET, Dr. RML Awadh University, Ayodhya, India

Abstract:*Distributed systems are increasingly becoming the dominant and rapidly expanding computational paradigm of the tomorrow. A cluster is really a form of parallel or distributed processing system that consists of a set of intertwined stand-alone machines that function together like a truly coherent computing and storage resources with a single system image (SSI) that means that perhaps the clusters are viewed as a single platform by the consumers. Global resource management, on the other hand, poses several concerns due to the sheer complexity and range of tools, as well as the need for user accountability. The possible advantages of load balancing in addressing the occasional congestion faced by some nodes when everyone else is idle or congested are widely agreed on a level of performance. This is also widely acknowledged that neither specific load balancing algorithm can adequately address evolving device characteristics and complex capacity management in a distributed ecosystem.*

To have a systematic approach and also in distributed systems, a proposed approach is created for a holistic view of element load balancing and also the qualities features of load balancing algorithms. The nomenclature has been expanded. In order for adaptive algorithms to understand the problem and manner of prefixing resilience along different components in distributed systems, they must first recognize the concerns. In addition, a proposed approach is specified. The much more effective load balancing techniques and the modeling hypotheses used in prior load balancing experiments are established through a study of related research. We consider the most appropriate load balancing algorithm and optimum metrics for parameter estimation of the algorithm as a consequence of and output of this assessment for a range of formulations of resulting goals, distributed system features, and workload balancing framework.

Keywords: **Load Balancing, Load Balancing taxonomy, Static Load Balancing, Dynamic Load Balancing, Proposed algorithm**

1. INTRODUCTION

Without clustering, the Load balancing can also be donewhen there are many numbers of independent servers that have same working setup, but other than that, are unknown of each other. Then, we can use a load balancer to forward requests to either one server or other, but one particular server does not use the other server's resources. Also, one resource does not share its current state with other resources.

A load balancer disburses workloads across various computing resources, including such slightly elevated computers or a cluster of computer. Load balancing is a strategy for

spreading processes across all nodes in a system in order to uniformly distribute workload across all nodes [1]. To ensure proper overall system performance, the load balancing algorithm attempts to fix the complete system load by straightforwardly moving workload through occupied heavily loaded nodes to available or strongly loaded nodes. Load balancing improves server efficiency, maximizes their usage, and ensuring that no single server is overburdened [2].

The load balancer is primarily described by two attributes. First, load should be assigned to the best candidate node, and then load should be migrated from a highly congested node to a light load node. The determination of load across each node and the opportunity to change computation through one node over another is essential tasks in load balancing.

Among the most complicated issues in accomplishing goals in distributed systems is load balancing, which can include several heterogeneous resources linked by one or more communications systems. It's indeed probable for some machines in these distributed systems to be highly loaded while others have been lightly loaded. This situation may cause the system to perform poorly.

2. LITERATURE REVIEW

A load balancer disburses workloads across various computing resources, including such slightly elevated computers or a cluster of computer. Load balancing is a strategy for spreading processes across all nodes in a system in order to uniformly distribute workload across all nodes [1]. To ensure proper overall system performance, the load balancing algorithm attempts to fix the complete system load by straightforwardly moving workload through occupied heavily loaded nodes to available or strongly loaded nodes. Load balancing improves server efficiency, maximizes their usage, and ensuring that no single server is overburdened [2].

The load balancer is primarily described by two attributes. First, load should be assigned to the best candidate node, and then load should be migrated from a highly congested node to a light load node. The determination of load across each node and the opportunity to change computation through one node over another is essential tasks in load balancing.

Among the most complicated issues in accomplishing goals in distributed systems is load balancing, which can include several heterogeneous resources linked by one or more communications systems. It's indeed probable for some machines in these distributed systems to be highly loaded while others have been lightly loaded. This situation may cause the system to perform poorly.

3. TAXONOMY OF LOAD BALANCING

Distributed scheduling is classified as a resource management concern in the taxonomy. The algorithms throughout this research can be categorized as global, dynamic, distributed, cooperative, suboptimal, heuristic, adaptive, and also have load balancing as a global goal that use this taxonomy.

Distributed computing, a branch of computer science, explores distributed systems. Whenever the data to still be operated on is dispersed around the network, the system is said to have been

distributed. This shows that the dataset to be worked on is spread through several computers connected by a network. Rather than centralized processing, data is effectively spread across an amount of nodes in a distributed network, resulting in faster execution. Use of such distributed systems to address computational challenges is often referred to as distributed computing. A process is converted into several activities in distributed computing, that are each resolved by one or even more computers that interact by sending message, which is known as message passing. Since the inception of digital computers, parallel processing was on the rise. The factors that encourage the study of concurrency in software and multithreading in hardware are numerous, perhaps one of the most important is a need to reduce the time consumption of large amounts of data [4].

Rather than just centralized processing, data is effectively spread across an amount of nodes in a distributed network, resulting in faster execution. Through use of distributed systems to overcome computational problems is often referred to as distributed computing. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate by exchanging messages to each other, which is actually called message passing.

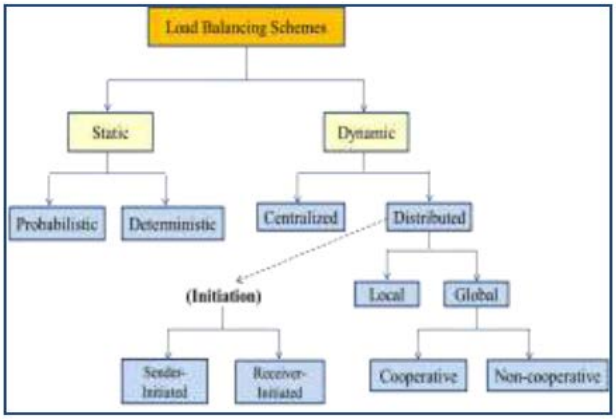

Figure 1: Load Balancing Taxonomy

In distributed computing, which task will be allocated to which processor and its execution time and response time is considered. Tasks represent loads, allocated to numerous heterogeneous nodes or processors in a network. In distributed system, load balancing is referred for allocation of tasks to different processors. Tasks allocated are independent of each other and executed according to the order they are assigned to processor and stored in its queue. Distributed system consists of distributed load balancers. Distributed load balancer consists of various types of processor, memory, and speed of the network. It allocates and balances the load among various processors for optimum resource utilization and minimum response time.

A load balancer distributes loads based on this order of service capabilities of each node, as by taking advantage of this diversity and by reducing the modular computing time. Like this, overall job execution time reduces and which further leads to quicker business decisions. Optimum load balancing algorithm leads to optimized resource utilization and throughput enhancement. It reduced response time, as well as evenly distributes the load among various nodes to avoid overloading problem. Keeping back-up, on another server, of all tasks and its

data residing on different nodes increases the reliability, as in case of one machine crash, the whole system can still survive. A load-balancing method may be categorized as static or dynamic, depending on the application's requirements and the method used [1]. As shown in Figure 1, these groups can be further subdivided into different schemes.

Static Load Balancing (SLB)

SLB method tasks are allocated to the processors on compile time and ones they are allocated, there can be no changes further at run time. As names specifies, in SLB, processes or tasks are allocated statically instead of dynamic allocation. Here, allocation of tasks occurs based on its prior information and various factors such as; mean execution time, IPC (inter-process communications), incoming time and extent of resource needed by it.

The decisions made by SLB policies are based on device statistics. They don't consider the actual state of a system into account. When another system load and number of procedures are calculated and very well defined at compilation time, static load balancing is used. The system's specifications are mostly set. The Static Load Balancer allows balancing choices based on the system's average workload. As a result, static load balancing take minimal time and is easier to implement than most certain load balancing strategies.

Dynamic Load balancing (DLB)

Whenever the system load and the quantity of procedures are expected to adjust over time Dynamic Load Balancing could be used. Within that situation, it's important to keep track of the system's load on a regular basis. DLB is method, under which allocation of tasks, to different processors, is done at run time. Dynamic policies make assessments based on the system's present state. These are much more complicated than measures that are set in stone. In DLB, at runtime, load is transferred nodes that are heavily loaded to nodes that are slightly loaded and likewise load is balanced among all available nodes and approximately at same time all processors get into idle state. It raises the workload and complicates the system [5].

The Dynamic Load Balancer makes each load balancing choice depending on the present system state. As a result, while Static Load Balancing is easier, quicker, and less costly versus Dynamic Load Balancing, it is just not sufficient for networks of varying workloads. As a result, the dynamic approach seems to be more effective for distributed networks in terms of keeping aware of current load mostly on system and migrating it accordingly.

Dynamic load balancing consists of the following steps:-

1. Initiation
2. Load selection
3. Information Exchange
4. Load balancer location

In case of reallocation of tasks there is the constant need of load monitoring system. This increases the overhead and makes the system more complicated. In DLB, at run time allocation, distribution of tasks and its reallocation increases and this results in increased overhead and less stability in comparison to static algorithm.

4. LOAD BALANCING POLICIES

Centralized vs. Distributed

In a centralized approach, all the nodes are connected to single centralized node that makes decision whereas, in distributed load balancing strategy, on certain workstations, that load balancer becomes repeated [6]. Where load information of each node is broadcasted and performs load balancing decision based on that shared information.

The issues regarding these strategies are: In centralized load balancing strategy, limited scalability and on failure of centralized load balancer bottleneck problem arises whereas, in distributed strategy the overhead and congestion across whole network increases, rapidly. Proper planning can help to solve these problems.

Sender-initiated vs. receiver-initiated

Heavily loaded nodes strive towards softly loaded nodes in a sender-initiated approach, while softly loaded nodes strive towards heavily loaded nodes in such a receiver-initiated approach. Since the likelihood of obtaining a slowly node is greater than those of locating a heavily-loaded node, the subscriber activated strategy is very well equipped for low and moderate device loads. The receiver-initiated strategy performs notably better systems because finding a heavily-loaded node is so much simpler.

Local vs. Global

All the nodes in the system are considered In global strategy that means every node has to be searched to check the lightly loaded nodes and in local strategy, Nodes are organized throughout groups of approximately equal cumulative computational capacity, allowing them to make balancing decisions locally by evaluating nodes within just a single group. Supplemental coordination and coordination between all the numerous workstations is needed for a global strategy. The advantage of a local strategy is that output data is only shared inside the community [7]. However, in situations in which the different groups demonstrate substantial performance discrepancies, the decreased coordination and synchronization amongst workstations can be a disadvantage.

Cooperative vs. non-cooperative

Distributed organizations work together to enable load-balancing decisions in a cooperative strategy. These are much more flexible than non-cooperative strategies, but they have a higher scheduling and coordination overhead [8]. Individual organizations function as distinct entities in the non-cooperative system, making scheduling informed decisions of the activities of other entities. These are unstable, but quick, and require less planning.

5. PROPOSED ALGORITHM

Within that proposed work, a clustered approach is used, with each cluster consisting of three nodes and then each cluster having a sustaining node. Also every cluster's load is stored in a queue maintained by the load balancer.This lowers the cost of infrastructure with in [3] architecture while also improving the service provided by [2] by utilizing clusters instead of

individual nodes. It determines how well a node is massively loaded or not using a threshold strategy.

The Load Balancer is divided into three parts: the Load Monitoring Server (LMS), the Load Reporting Server (LRS), and also the Decision Making Server (DMS) (DMS). The Load Monitoring Server and the Load Reporting Server conduct related calculations and gather data about both the system's load. Unless a node is overloaded, the decision-making server runs and identifies the most suitable node with which the overload should be moved.

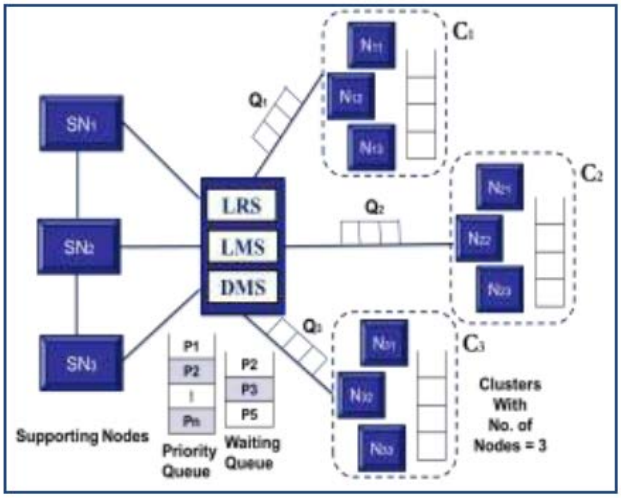

Figure 2: Contributing node within each cluster of three nodes, a suggested design for load balancing

Assumptions:

1. Every node is capable enough to maintain its priority queue and to handle tasks.
2. Factors such as mobility, battery power, processing power and memory capability do not vary rapidly with respect to time.
3. There are no byzantine faults possible in the system.
4. Nodes remain in active state and in a cluster allotted to them until they move out of range or stops working due to some technical fault.

CENTRAL_LOAD_BALANCER( )

1. Make C[i:1 to n] clusters.
2. Across each cluster N[j:1 to3], define three nodes.
3. Instead every cluster in T has a read threshold.
4. That initial load through each node is really the time it takes for the procedure to turn about.
5. Invoke the LOAD_REPORTING_SERVER().
6. That overall loading at each cluster is really the aggregate of the loads at its nodes plus the awaiting procedure turn-around period.
7. Construct a blank waiting queue as well as a preference queue with both the initial processes ordered by importance.
8. For each and every cluster SN_i[i:1 to n] , eventually create a supportive node.

9. Invoke the LOAD_MONITORING_SERVER() procedure.
10. Keep an eye mostly on load at SN;

    If (SN[i].load==0)
    then:-
    Invoke the DECISION_MAKING_SERVER(Process P)

11. Take the exit.

LOAD_REPORTING_SERVER ( )

1. At every other node N[j], accumulate load.
2. In the Load Queue Q[j], archive the load of Node N[j].
3. Go back to Q.

LOAD_MONITORING_SERVER (Process P)

1. Make Overload equal to Overload + P.time.
2. Transfer Procedure P to the back of the queue.
3. Overload Returns

DECISION_MAKING_SERVER (WQ, P)

1. For i=1 to n, reiterate stage [2]
2. If (SN[i].load==0), then:

Set SN[i].load := P.time

Set SN[i].process:= P

Set SN.pri = P2.pri

3. Set Process P1i:=iPop (WQ)
4. If (P1.prii>iSN[i].pri)ithen:

    i. Interrupt (SN[i].process)
    ii. Push (SN[i].process) to WQ
    iii. Set SN[i].proc := P1
    iv. Set SN[i].load := P1.time
    v. Set SN.pri := P2.pri

5. Else PushiP1 toiW.
6. Return

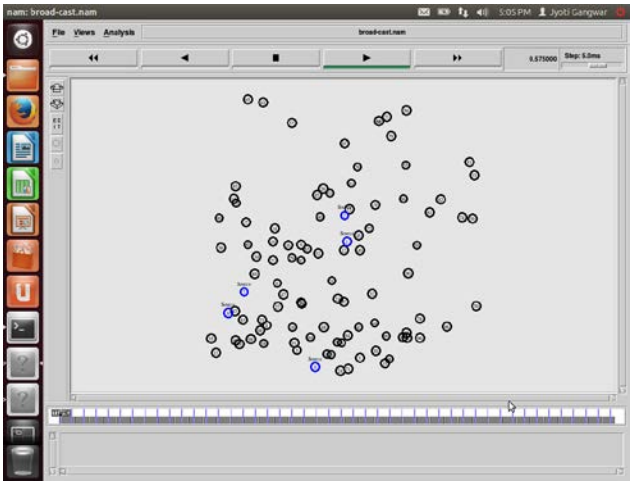

Figure 3: Blue nodes are with load

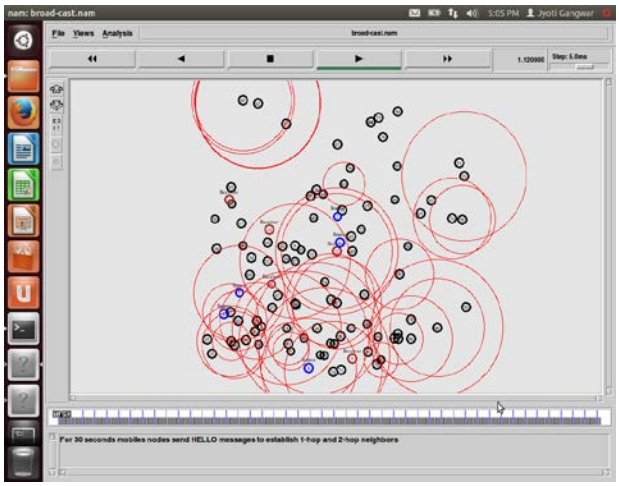

Figure 4: Clustering of nodes for load distribution

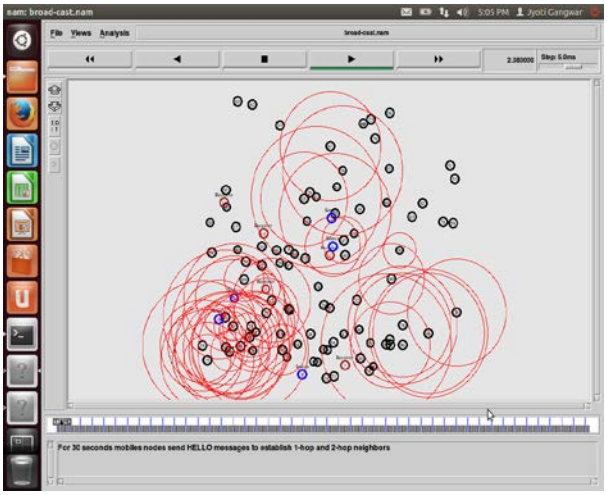

Figure 5: maximum numbers of nodes are covered for load distribution

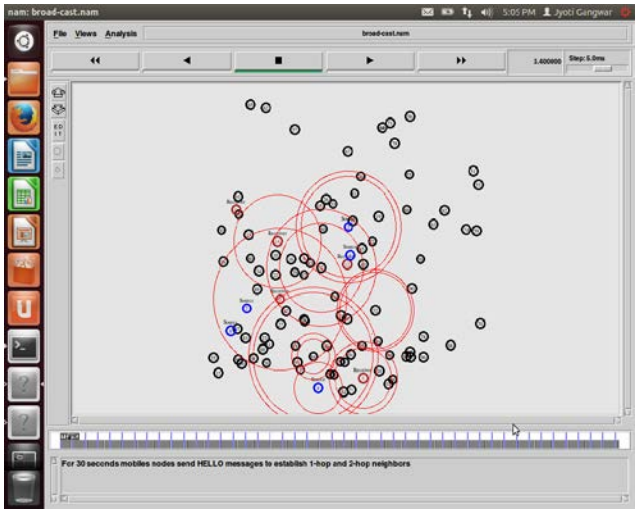

Figure 6: Static load distribution by clustering

6. ALGORITHM MODEL

Weight Model

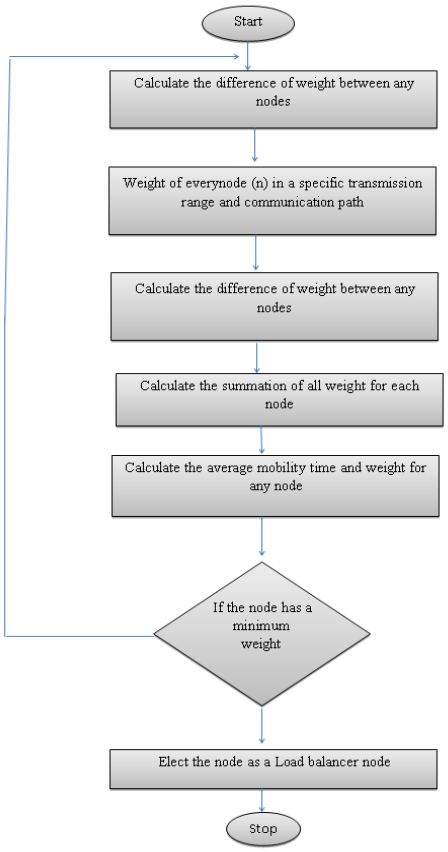

Figure 7: Weight Algorithm Flow Chart

Proposed Model

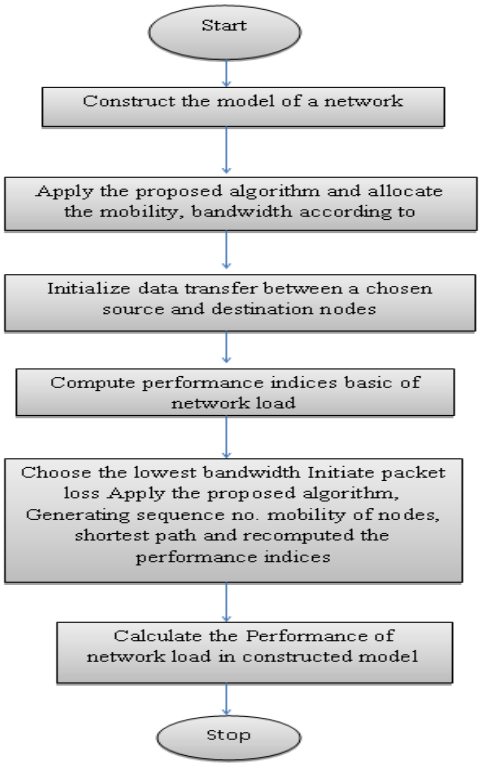

Figure 8: Proposed Algorithm Flow Chart

7. RESULTS

In this paper, the performance evaluation Model for Average Throughput,Average Packet Delivery Ratio (PDR), End-to-End Delay, Total Packet Dropped, Normalized Routing Load (Routing Overhead Optimized). We compare the model performance in Weight Algorithm, Priority Algorithm, and Proposed Algorithm. Any of the most relevant performance indicators may be assessed.

1) Average End-to-End Delay –The period it takes packets to travel through the network on average. This really is the duration in seconds from when the sender produces the packet before it is sent to the recipient application layer. As a result, it includes all network delays, including buffer Queue, transmission, and routing protocol operations and MAC datastreams exchange.

2) Average Throughput–Throughput is calculated as the ratio of the total number of data sent by a sender to just the time required around for recipient to collect the very last packet. It is measured in kilobits per second (kbps). In quite a MANET, throughput requires frequent topology changes, poor message communication, restricted bandwidth, and inadequate capacity. Networks with a high average throughput are attractive.

(Number of available packets obtained * Packet size * 8) /Simulation Period

3) Packet Delivery Ratio (PDR) –The proportion of packets sent through the recipient to packets sent from the sender. The highest possible throughput that only the network could attain is represented by this value. Inside a network, a higher avg packet delivery ratio is required.

Obtained packets/Created packets * 100 = Packet Delivery Ratio

4) Total Packet Dropped – Whenever one or even more transmitted packets fail to deliver their intended destination.

Data Packet Transmitted - data Packet Received = Packet Drop Ratio.

5) Normalized Routing Load (Normalized Routing Overhead)–This is really the total amount of routing packets generated every data packet in kilo bits. This same overall amount of routing packets transmitted (including forwarded routing packets) is calculated by dividing the number of data packets obtained to arrive at this figure.

Routing packets/received packets = NRL

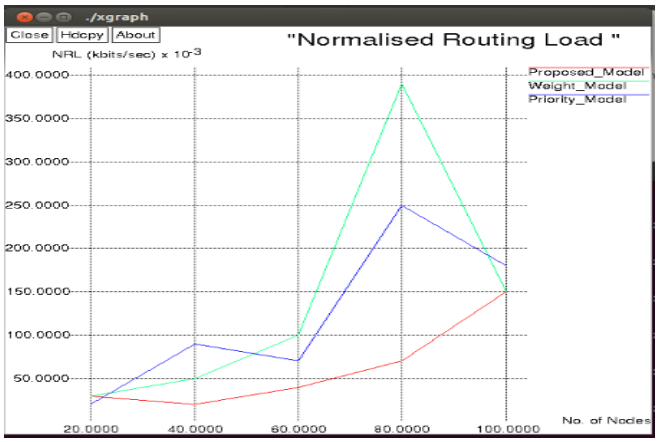

Figure 9: Normalized Routing Load (NRL) Comparison Graph

Normalized Routing Load calculation, then number of nodes connected in a network as varying with no. of Nodes shown in Figure 5 through which comparison graphs of Weight algorithm, Priority algorithm and proposed algorithm is obtained.

Table 1: Normalized Routing Load (NRL) Comparative Analysis of Result

Algorithm	No. Of Nodes				
	20	40	60	80	100
Weight Algorithm	0.03	0.05	0.10	0.39	0.15
Priority Algorithm	0.02	0.09	0.07	0.25	0.18
Proposed Algorithm	0.03	0.02	0.04	0.07	0.15

These tabulation formations have shown in table 1, that different no. of nodes values define as comparison graph values of three algorithms.

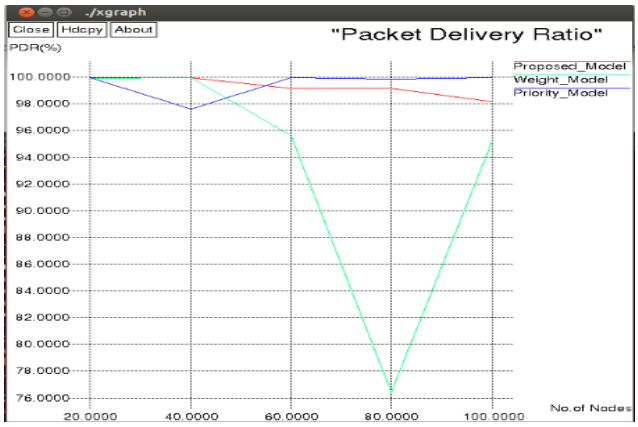

Figure 10: Packet Delivery Ratio (PDR) Graph

Packet delivery ratio measure, then the nodes is connected in a network as varying with no. of nodes shown in Fig, through which comparison graphs of Weight algorithm, Priority algorithm and proposed algorithm is obtained.

Table 2: Packet Delivery Ratio (PDR) Comparative Analysis of Result

Algorithm	No. Of Nodes				
	20	40	60	80	100
Weight Algorithm	99.93	99.94	95.62	76.41	95.23
Priority Algorithm	99.97	97.60	99.96	99.88	99.98
Proposed Algorithm	99.96	99.97	99.15	99.15	98.19

These tabulation formations have shown in table 2, that different no. of nodes values define as comparison graph values of three algorithms.

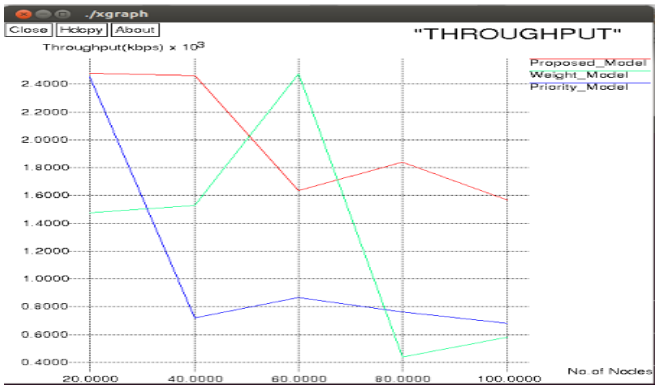

Figure 11: Throughput Comparison Graph

Average Throughput calculation, then nodes is connected in a network as varying with no. of Nodes shown in Fig., through which comparison graphs of comparison graphs of Weight algorithm, Priority algorithm and proposed algorithm is obtained.

Table 3: Throughput Comparative Analysis of Result

Algorithm	No. Of Nodes				
	20	40	60	80	100
Weight Algorithm	1473.7	1529.3	2476.6	436.27	579.3
Priority Algorithm	2453.81	720.79	867.57	762.78	678.7
Proposed Algorithm	2480.9	2460.1	1635.2	1839.2	1567.4

These tabulation formations have shown in table 3, that different no. of nodes values define as comparison graph values of three algorithms.

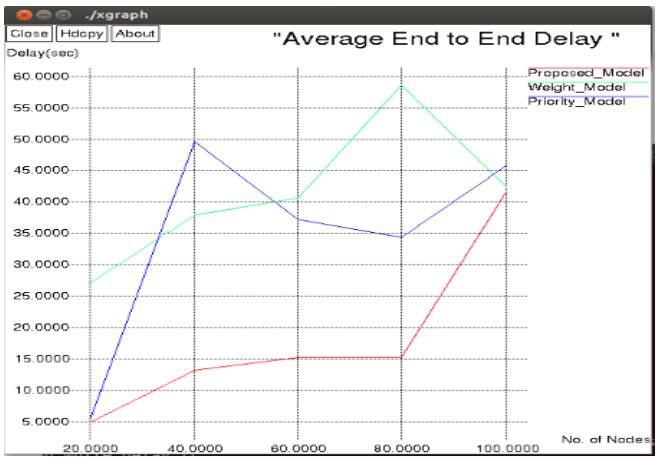

Figure 12: Average End to End Delay Comparison Graph

The performance of compare algorithm with respect to Average End-to-End Delay calculation connected in a network as varying with no. of Nodes shown in Fig., through which the comparison graphs of Weight algorithm, Priority algorithm and Proposed algorithm is obtained.

Table 4: Average End to End Comparative Analysis of Result

Algorithm	No. Of Nodes				
	20	40	60	80	100
Weight Algorithm	27.03	37.94	40.67	58.64	42.48
Priority Algorithm	5.33	49.63	37.21	34.37	45.77
Proposed Algorithm	4.84	13.21	15.20	15.24	41.52

These tabulation formations have shown in table 4, that different no. of nodes values define as

comparison graph values of three algorithms.

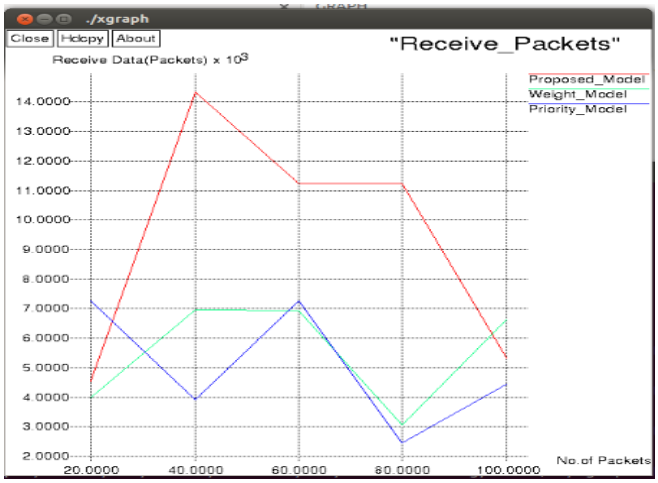

Figure 13: Receive Packets Comparison Graph

Table 5: Receive Packets Comparative Analysis of Result

Algorithm	No. Of Nodes				
	20	**40**	**60**	**80**	**100**
Weight Algorithm	4001	6960	6943	3057	6630
Priority Algorithm	7259	3906	7259	2446	4434
Proposed Algorithm	4538	14348	11239	11237	5329

These tabulation formations have shown in table 5, that different no. of nodes values define as comparison graph values of three algorithms.

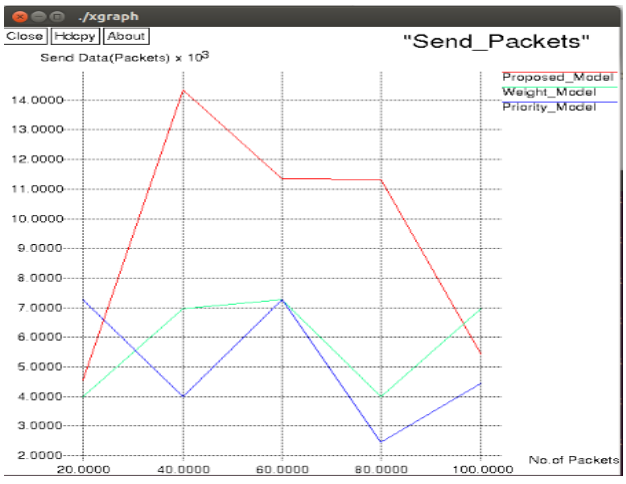

Figure 14: Send Packets Comparison Graph

Table 6: Send Packets Comparative Analysis of Result

Algorithm	No. Of Nodes				
	20	**40**	**60**	**80**	**100**
Weight Algorithm	4004	6964	7261	4001	6962
Priority Algorithm	7261	4002	7262	2449	4435
Proposed Algorithm	4540	14353	11335	11333	5427

These tabulation formations have shown in table 6, that different no. of nodes values define as comparison graph values of three algorithms.

8. CONCLUSION

This chapter provides the summary of the research work done in this thesis; first the conclusion has been made from this study and then the suggestions for the future research and discussed.

We have proposed clustered approach for distributed load balancing. The proposed architecture and algorithm are well-suited for distributed systems, as they make sure that neither process is starved for resources and no process is overburdened. The architecture discussed here works for a cluster with 3 nodes. Decreased network speed, semi-distributed architecture, dynamic approach to load balancing, and reduced cost and complexity are all benefits of the prototype architecture over the referenced designs.The proposed work ensures that no task suffers from starvation as well as, no task get overwhelmed. It operates for clusters with a ‘p’ total number of nodes of ‘n'. To make things right for heterogeneous networks, a dynamic scheduling scheme and other features have been used.The proposed packaged load balancing technique includes clusters of nodes capable of processing different tasks. Load balancer consisting of LRS, LMS and LBL balance the loads in the clusters. Initially, the Supporting Node, S_{Ni} has also been allotted the low priority load which increases the overall resource utilization by minimizing the initial ideal time of S_{Ni}.

REFERENCES

1. *Jean Ghanem, “Implementation of Load Balancing Policies in Distributed Systems”, The University of New Mexico Albuquerque, New Mexico, June 2004*
2. *Ahmad Dala¨ah, “A Dynamic Sliding Load Balancing Strategy in Distributed Systems”, International Journal of Information Technology, Vol 3, No.2, April 2006*
3. *Hao Jiang, Luo, Feng, Tang, Yin, “DALB: A Dynamic Application-sensitive Load Balancing Algorithm”, International Conference on Computer Science and Service System, 2012*
4. *Abhijit A. Rajguru, S.S. Apte, “A Comparative Performance Analysis Of Load Balancing Algorithms In Distributed System Using Qualitative Parameters”, International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-1, Issue- 3, August 2012.*
5. *P.Mohammadpour, M.Sharif, and A.Paikan, \A self training algorithm for load balancing," in Fourth*

International Conference on Networked Computing and Advanced Information Management, 2008.

6. *Mayuri A. Mehta, Devesh J. Jinwala, "Analysis of Significant Components for Designing an Effective Dynamic Load Balancing Algorithm in Distributed Systems", Third International Conference on Intelligent Systems Modeling and Simulation 2012 IEEE*
7. *Wenzheng Li, Hongyanshi, "Dynamic Load Balancing Algorithm Based On FCFS", Fourth International Conference on Innovative Computing, Information and Control 2009 IEEE*
8. *Saurabh Gupta, Avinash Kumar Pal, "A Comparison on Network on Chip using Simulation Tool NS2", International Journal of Recent Trends in Science and Engineering (IJRTSE), Vol 1, March 2021.*