Proposed Semi-Distributed Framework to Enhance Web Performance

Payal Beniwal
Research Scholar /MMICT&BM
Maharishi Markandeshwar University
Mullana, Ambala

Atul Garg
Associate Professor/MMICT&BM
Maharishi Markandeshwar University
Mullana, Ambala

ABSTRACT
Server load balancing is a technique of circulating client requests across a group of servers. Static and Dynamic load balancing methods are used to distribute the workload equitably over every node of the system. Dynamic load balancing is adaptive in nature and performs load distribution at run time, which makes it more suitable for systems where workload is unpredictable, as compared to static load balancing methods. Two approaches namely- Completely Distributed and a proposed approach Semi- Distributed are tested against two applications – database and ftp. Efficiency of models is tested using various parameters of FTP, IP, and TCP. Simulation results show that semi-distributed method provides better system efficiency and performance.

Keywords
Completely-Distributed, Semi-Distributed, OPNET, FTP, IP, TCP.

1. INTRODUCTION
To increase the efficiency of network multiple servers are deployed connected to each other to form a server farm. Client request is transferred to one of the back end server in the server farm. It replies to client request without the client ever knowing about the internal distribution [22]. The process of distributing the client request among servers is performed by load balancing technique. Load balancing improves performance by transferring requests from heavily loaded to lightly loaded nodes. Load balancing is described as a technique of dividing and circulating tasks to all nodes of the system so that more jobs can be served and the system can perform efficiently [3]. It also provides the features of bottleneck removal and system failover [16]. A load balancing method should be general and transparent to the applications, also it should provide minimum overhead to the system [19]. In general Load Balancing mechanism is divided in two categories: Static and Dynamic. Static Load Balancing is performed by considering predetermined behavior of the system. Whereas Dynamic Load Balancing considerers current state of the system [15]. In Dynamic Load Balancing work load is distributed among nodes at run time. Dynamic approach adaptive in nature i.e. it can be modified as the state of system changes. Applications where workload is unpredictable or change during execution dynamic methods are implemented to achieve better performance [15]. Dynamic load balancing can be achieved in three forms: Centralized, Completely-Distributed and Semi-Distributed.

In Centralized approach load balancing is performed by one single node and all other nodes interact with this central node. In Completely-Distributed approach the responsibility of load balancing is shared by all nodes either in co-operative form or non co-operative form. Best suited in environment where each node is given chance to act alone and lesser interaction with others [15]. There is no central node so failure of one node does not shut down the whole system. Also the condition of bottleneck does not arise in this form. The major drawback of this approach is the overhead caused by the large amount of communication present between the nodes of the system. This disadvantage sometimes leads to the delay in response to the client request and loss of the data in between communication of nodes.

Semi- Distributed on the contrary is combination of both Centralized and Completely Distributed approaches [3]. In this approach all nodes are divided in a number of clusters. Each cluster has a central node interacting with its members and performing the load balancing activity. This central node is also connected to the central node of other clusters to achieve resource sharing and load distributing. Best suited in environment where there is large number of nodes in the system. The communication overhead is more than centralized approach but is much lesser than completely distributed approach.

This paper presents the comparison of the Completely-Distributed and Semi-Distributed approaches of load balancing. Both approaches have equal number of client and server nodes. Two applications – Database and ftp are created to test the models. Both models are simulated and compared on global level and individual statistic level. Parameters used for comparison are- FTP download response time, FTP upload response time, traffic dropped, delay (TCP). The section 2 describes the proposed model. Simulation outputs are shown in section 3 and comparative results are in section 4. Conclusion is presented in section 4.

2. PROPOSED MODEL
Distributed approach offered various advantages such as minimum storage, maximum resource utilization and continuous availability of system. For large systems consisting of thousands of nodes the centralized and completely-distributed approaches suffer from major drawbacks listed below [3]:

- In completely distributed approach the random arrival of load does not provide accurate state of the whole system. Load balancing decisions made by nodes are probability based which sometimes lead to poor performance of the system.
- In completely-distributed approach communication delay can cause a situation where a request keeps on migrated from one node to another without being executed.
- With the increase in number on nodes in system the communication traffic will also increase. This may result in increase in response time and hence will degrade the efficiency of system.
- The increase in number of nodes in system will result in control overhead and can further lead to poor load balancing decisions.

- As the number nodes increase the storage capacity for maintain their information will also increase in completely-distributed system.
- In case of centralized approach the central node can become the point of bottleneck and hence will decrease the throughput.
- In centralized approach the failure of central node can stop the working of whole system.
- With the increase in size of system the overhead of the load balancing algorithm can delay the execution of the task.

The proposed Semi-Distributed system approach avoids the major drawbacks of bottleneck and over communication. In Semi-Distributed policy nodes are divided in equal clusters. Each cluster adopts a centralized approach where central nodes take charge of load balancing within the cluster. Clusters together adopt a distributed approach and exchange information with each other to achieve global Load Balancing. In the proposed approach central node of each cluster performs the following functions:

- Assigns task to individual nodes of the cluster.
- Maintains the load status of all nodes within the clusters and other neighboring clusters.
- Transferring of load to other clusters if needed.

Figure-1 illustrates the logical view of the proposed approach. Request sent by the client passes through the communication network and reaches to the central node of the cluster. The central node after checking its state information forwards request to its subordinate nodes. In case the cluster does not have the required resources for execution of the request it communicates with its neighboring clusters and passes on the client request to them.

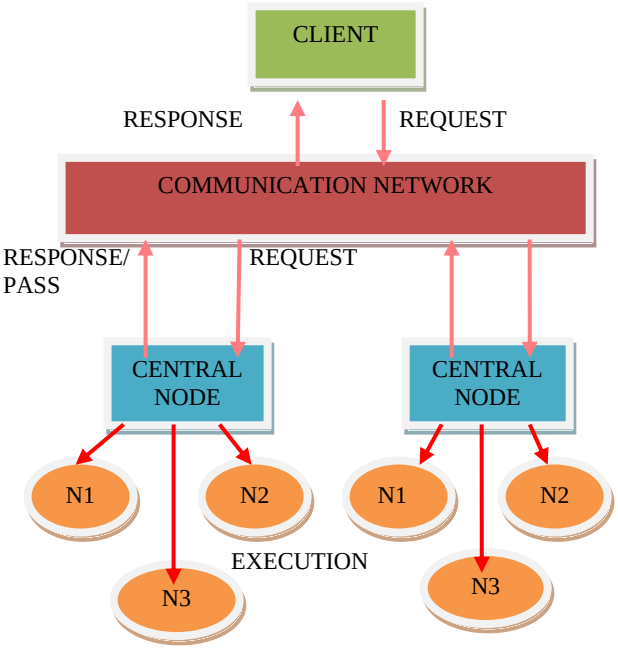

Figure-1 Logical view of Semi-Distributed Approach

Working algorithm of proposed Semi-Distributed approach is given in Figure-2 below.

Step 1: Client sent request

Step 2: Request accepted by process distribution module

Step 3: Forwarded to cluster

Step 4: Central node receives request

Step 5: Status of subordinate node validated

Step 6: If subordinate node is underloaded,

go to step 8

Else, repeat step 5

Else If, go to step 7

Step 7: Transfer load to neighboring cluster,

go to step 4

Step 8: Execute the request

Step 9: Respond back to client

Figure-2 Algorithm of Proposed Semi-Distributed Approach

Client sent the request through internet services. The request is firstly received by the process distribution module. This module acts as an interface between client and server. The distribution module maintains the state information table of all servers on the network. After verification of the current status of the clusters, the client request is forwarded to the central node of the cluster. The central node also maintains the state information table of all its subordinate nodes. After checking the load status of subordinate nodes, central node passes on the client request to a suitable node for execution. In case the central node finds out that all its subordinate nodes are in overloaded condition than, the central node transfer the request to another neighboring cluster node.

In the proposed approach two levels of communication is present- firstly between central node and its subordinate nodes, secondly between central nodes of neighboring clusters.

The proposed approach reduces the communication overhead and avoids the condition of congestion in network. Flowchart of the proposed approach is given in Figure-3.

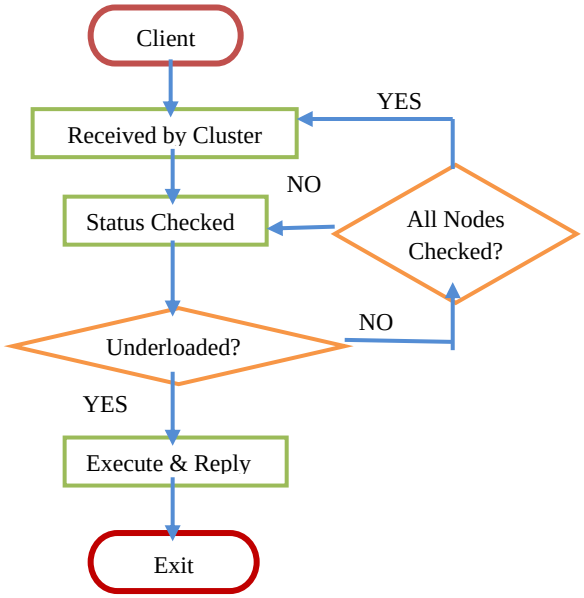

Figure-3 Flowchart of Semi-Distributed Approach

To further check the advantages of semi-distributed approach over completely distributed approach, two models are designed using OPNET Riverbed Modeler Academic Edition (version-17.5.A PL6). The OPNET Modeler is used to design protocols, devices and network behaviors by using the special purpose

modeling functions. It provides different levels of modeling depending on the necessities and requirements of the simulation. It consists of step by step GUI (graphical based interface) based guide to establish an overall environment called as project. Within a project different Approaches can be developed and their performance can be analyzed using simulation. It's easy to use environment gives it the preference from other such modelers.

In the modeler two Approaches are created- Completely-Distributed and Semi-Distributed. Both Approaches have same list of parameters and their corresponding values are also same. The list and the values are shown in table 1 below.

Table 1: List of Parameters

PARAMETER	VALUE
Scale	Office
Size	100m X 100m
Client Node	26units (eth4_slip4_multihomed_client_node)
Server Node	09units (IBM_p650_6m2_1450_8CPU_node)
Central Node	3C_SSII_1100_3300_4s_ae52_e48_ge3 switch
Load Balancer	Fd_server_e24_fe2

Using the above parameters the two Approaches are created. Figure-4 shows the Completely-Distributed scenario and Figure-5 shows the Semi-Distributed scenario respectively.

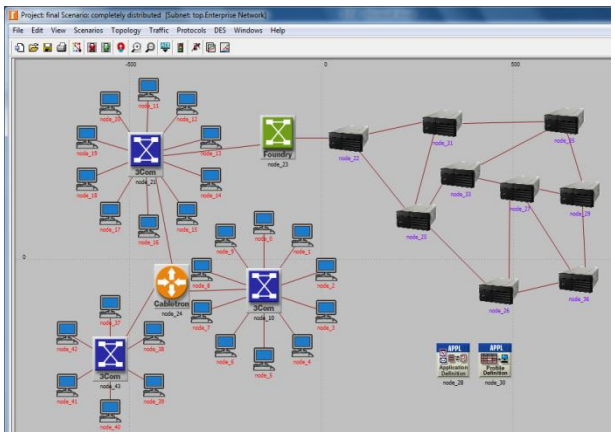

Figure-4 Completely-Distributed Scenario

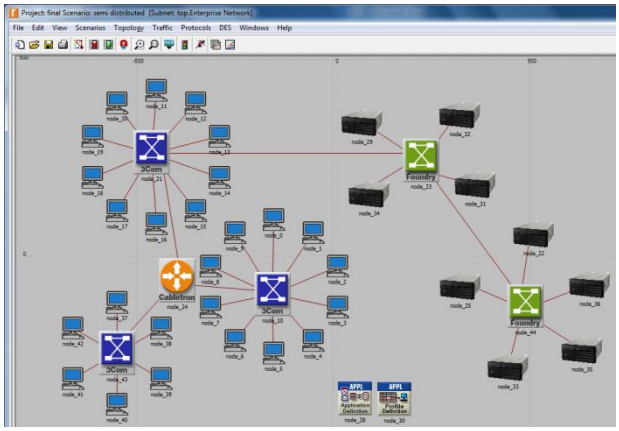

Figure-5 Semi-Distributed Scenario

To measure the efficiency of both the Approaches two applications namely – Database and ftp are created. Both applications have property of heavy load browsing. Both the applications are executed simultaneously on the both models. Using the application configuration command the attributes of both applications are set .The list of attributes and their corresponding set values set are given in table 2.

Table 2: Application Configuration

ATTRIBUTE	VALUE
Operation mode	Simultaneous
Start Time (sec)	Uniform(100,110)
Duration (sec)	End of Simulation
Inter Repetition Time (sec)	Constant(300)
Number of Repetition	Unlimited
Repetition pattern	Serial

The Approaches are compared to each other on the basis of download response time (sec) of an application, upload response time (sec) of an application, traffic dropped by the application during the communication between the client and server node and lastly the TCP delay in network. The simulation results are shown in next section.

3. SIMULATION

The proposed approaches are simulated for 20 minutes each and with 150 values per statistics. Four graphs (Figure 6 to Figure 9) show the result of both approaches in combination. Average of result is taken to show the difference in values for both approaches. Red line in figures denotes the semi-distributed approach and blue line represents the completely-distributed approach.

3.1 FTP Download Response Time

It is the time elapsed between sending a request and receiving the response packet. Every response packet sent from server to an FTP application is included in this statistics. Figure-6 shown below is the graph showing the statistics of FTP download response time (sec) of both Approaches After running the simulator for 19 min in both cases the response time for distributed came out to be 0.160 sec and that of semi-distributed is 0.195 sec.

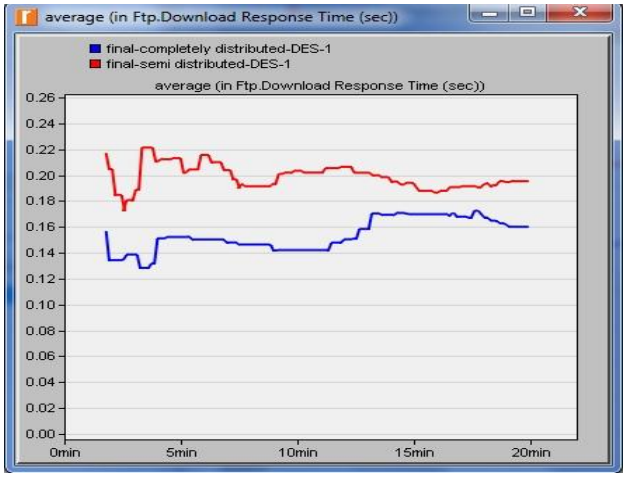

Figure -6 FTP Download Response Time

3.2 FTP Upload response Time

It is defined as the time elapsed between sending a file and receiving the response. The response time for responses sent from any server to an FTP application is included in this statistic. Simulation results show that Semi distributed approach results in better upload response time as compared to completely distributed approach. After simulating the two approaches for 20 min the average FTP upload response time of semi-distributed is 0.286 secs and that of completely distributed is 0.290 secs. Figure-7 presents the comparison of upload response time (sec) of both Approaches

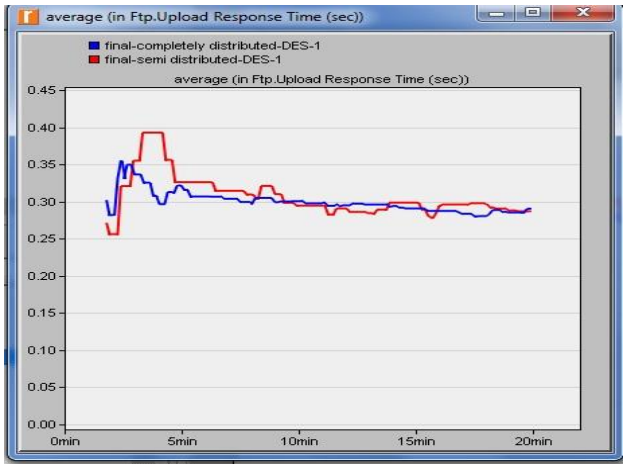

Figure- 7 FTP Upload Response Time

3.3 IP Traffic Dropped

Traffic dropped (packets/sec) is defined as the number of IP datagram's dropped by all nodes in network across all the IP interfaces. The reasons of dropping can be any one of the following:

- Insufficient space in central processors queue.
- Insufficient space in processors buffer.
- Maximum number of hops exceeded by an IP datagram.

Average traffic drop down rate for completely distributed is 2.950 secs and that for semi-distributed is 2.918 secs. Figure-8 shows the simulation result graph of both the approaches showing average traffic dropped during the communication between client and server nodes.

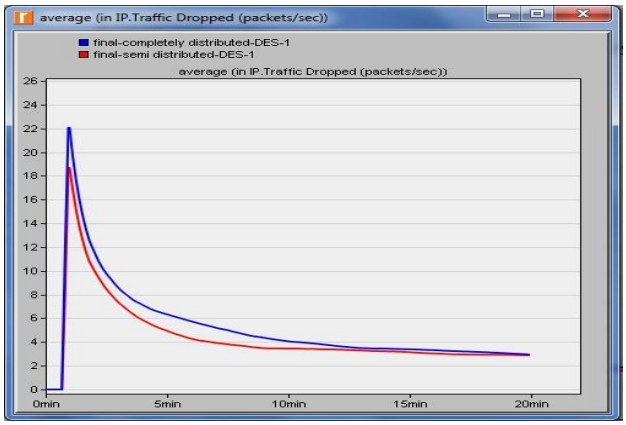

Figure-8 IP Traffic Dropped

3.4 TCP Delay

It is computed form the time an application request is dispatched from source TCP layer to the time it is collected by the TCP layer of destination node. When simulated for 20 min the average delay rate of distributed is 0.026 secs whereas that of semi-distributed is 0.022 secs. Lesser will be the delay faster will be the packet delivery which results in client satisfaction. Figure-9 below shows the graph of TCP delay. It is defined as the delay of packets received by TCP layers in the complete network for all connections. .

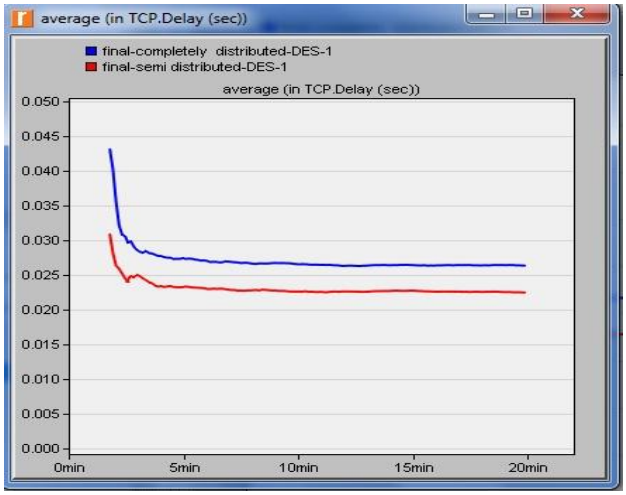

Figure-9 TCP Delay

4. RESULTS

Table 3 shows the statistical values obtained during the simulation of two approaches. Figure 10 shows the comparative chart of values obtained during simulation of both approaches.

Table 3: Simulation Values of Semi-Distributed and Completely Distributed Approaches

Parameters	Semi-Distributed	Completely Distributed
Traffic Dropped	2.918 packets/sec	2.950 packets/sec
TCP Delay	0.022 sec	0.026 sec
FTP Download Response Time	0.195 sec	0.160 sec
FTP Upload Response Time	0.286 sec	0.290 sec

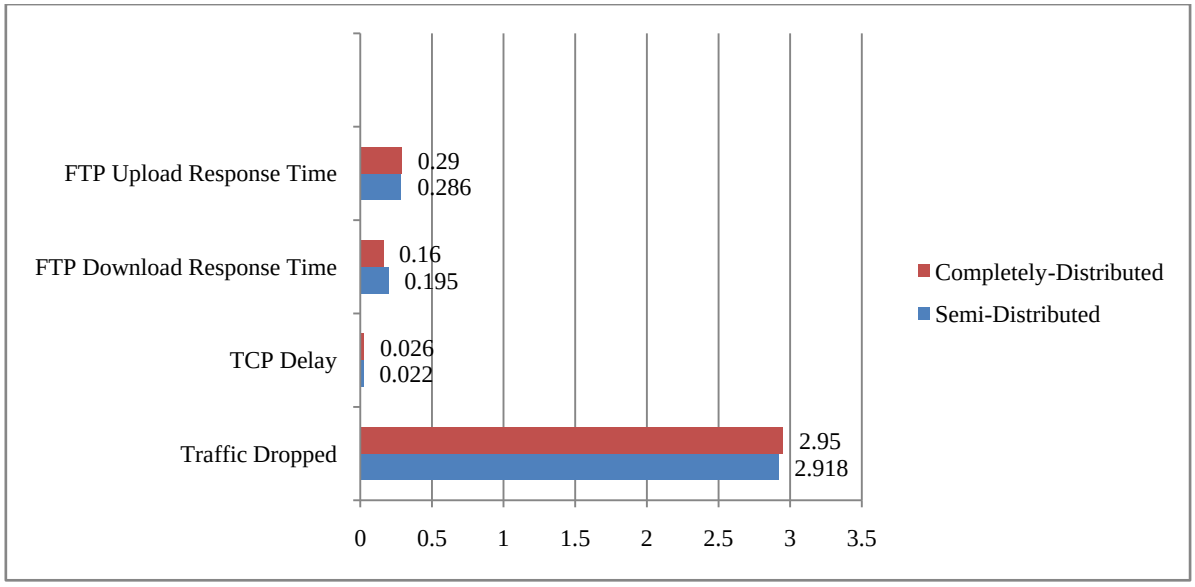

Figure-10 Comparison Chart of Semi-Distributed and Completely Distributed Approaches

5. CONCLUSION

In this paper two approaches of dynamic load balancing i.e. Completely-Distributed and Semi-Distributed are modeled and simulated in OPNET Riverbed Modeler. Both models are compared to each other on FTP, IP, TCP parameters. The graph in Figure-10 show the simulation results of proposed and traditional approach. The proposed Semi-Distributed approach gives better performance in FTP upload response time and in FTP download response time completely distributed shows better results. TCP delay in Semi-Distributed approach is less as compared to completely distributed, which results in better user satisfaction. Due to congestion problem completely distributed approach drops more data packets as compared to proposed semi-distributed approach. Further, it can be concluded that proposed approach giver better performance as compared to traditional approach. In future the proposed semi-distributed approach can be modified in order to get better download response time. The hardware implementation of proposed approach can be studied further.

6. REFERENCES

[1] Abubakar, Haroon Rashid and Usman, "Evaluation of Load Balancing Strategies", National Conference on Emerging Technologies, 2004.

[2] Ali M.Alakeel, "A Fuzzy Dynamic Load Balancing Algorithm for Homogeneous Distributed Systems", World Academy of Science, Engineering and Technology, vol.6, 2012.

[3] Ali M.Alakeel, "A Guide to Dynamic Load balancing in Distributed systems", IJCSNS, vol.10 No.2, 2012.

[4] Asser and Charles, "Design, Implementation, and Performance of a Load Balancer for SIP Server Clusters", IEEE/ACM transactions on networking, June 2012.

[5] Atul and Anil, "Portable Extended Cache Memory to Reduce Web Traffic", International Journal of Engineering Science and Technology, Vol.2 No.9 2010.

[6] Atul and Dimple, "A Comparison and Analysis of Various Extended Techniques of Query Optimization", IJICT vol.3 No.3, 2012.

[7] B. Narendran, Sampath Rangarajan, and Shalini Yajnik, "Data distribution algorithms for load balanced fault-tolerant Web access", In Proc. 16th IEEE Sympos. on Reliable Distributed Systems, pp 97–106, 1997.

[8] Bryhni, E. Klovning and O. Kure, "A Comparison of Load Balancing Techniques for Scalable Web Servers", IEEE Network pp 58-63, July/August 2000.

[9] Cardellini, Colajanni, M., and Yu, "Dynamic load balancing on web-server systems", IEEE Internet Computing vol.3 No.3 pp 28–39,1999

[10] Daniel and Anthony, "Algorithmic Mechanism Design for Load balancing in Distributed Systems", IEEE Trans. on Systems, Man and Cybernetics", vol.34, No.1, 2004.

[11] Dimple Juneja and Atul Gar, "Collective Intelligence based Framework for Load Balancing of Web Servers", IJICT, vol. 3 No.1, 2012.

[12] Eager, D.L, E.Lazowska and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems", IEEE Trans. Software Eng. vol.12 pp 662-673, 1986.

[13] F. Arlitt and C. L. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", IEEE Trans. Networking vol.5 No. 5 pp 631-645, 1997.

[14] Gaochao Xu, Junjie Pang, and Xiaodong Fu, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", TSINGHUA SCIENCE AND TECHNOLOGY, vol.18, no.1, February 2013.

[15] H.C. Lin and C.S. Raghavendra, "A Dynamic Load-Balancing Policy with a Central Job Dispatcher (LBC)", IEEE Transaction on Software Engineering vol.18 No.2 pp148-158, 1992.

[16] K.Salah, P.Calyam and M.I.Buhari , "Assessing Readiness of IP Networks to Support Desktop Videoconferencing Using OPNET", Journal of Network and Computer Applications, November 2008.

[17] Manju Sharma and Manoj, "Comparative Investigation on Throughput and Client Response Time for a Switched and Routed Wireless LAN based on OPNET", Proceedings of

National Conference on Emerging Trends in Computing and Communication (ETCC-07), pp 436-440, 2007.

[18] Manju Sharma, Manoj Kumar and Ajay K.Sharma, "HTTP and FTP Statistics for Wireless and Wire-line Network with and without Load Balance based on OPNET", IJISS, vol.5, no.1, p 112-125 , 2008.

[19] Mayank & Atul, "Comparative Survey of Load Balancing Algorithms in Cloud Computing Environment", IJDCC, vol.1, No.2, 2013.

[20] Mohsen & Hossein Delda, "Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants", Informatica 32, pp 327–335, 2008.

[21] Payal and Atul, "A Comparative study of Static and Dynamic Load Balancing Algorithms", International journal of Advance Research in Computer Science and Management Studies, Vol.2 No.12, Dec 2014.

[22] Ritika and Harjot, "Load Balancing Techniques using Mobile Agents", IPASJ-IIJCS, vol.2, no.12, 2014.

[23] Santosh Kumar and Vikram Singh, "Load Balancing in Distributed Systems", IJRREST, vol.1, no.2, September 2012.

[24] Suriya and Prashanth, "Review of load balancing in cloud computing", IJCS, vol.10, no.1, Jan 2013.

[25] . Lan and T.YU, "A Dynamic Central Scheduler Load Balancing Mechanism", Proc. of 14th IEEE Conf. on Computers and Communications, 1995.