Performance Comparison of Server Load Distribution with FTP and HTTP

Yogesh Chauhan
Assistant Professor
HCTM Technical Campus, Kaithal

Shilpa Chauhan
Research Scholar
University Institute of Engg & Tech, Kurukshetra

ABSTRACT
A Web server's constraint on the number of pages it can serve simultaneously is largely because of two factors: the bandwidth available and the Web server itself. When a website is popular and if there is only one web server responding to all the incoming HTTP requests for a website it may be possible that the capacity of the web server may not be able to handle high volumes of incoming traffic. The increase in traffic and connections to the website can lead to a point where the upgrading of server hardware will no longer be cost effective. Thus, more servers need to be added to distribute the load among the group of servers. The load distribution among these servers is known as load balancing. In this paper we are analyzing the performance of HTTP network for optimum load balancing using OPNET.

Keywords
Opnet, CPU Utilization, IP, Cluster

1. INTRODUCTION

Server Load Balancing (SLB), Fig.1, is defined as a process and technology that distributes site traffic among several servers using a network-based device. This device intercepts traffic destined for a site and redirects that traffic to various servers. [6] The load-balancing process is completely transparent to the end user. There are often dozens or even hundreds of servers operating behind a single URL. The functions of a load balancer are as:

1. It intercepts network-based traffic (such as web traffic) destined for a site and splits the traffic into individual requests and also decides which servers receive individual requests.
2. It watches all the available servers and ensures that they are responding to traffic. If they are not responding they are taken out of process.
3. It provides redundancy by employing more than one unit in a fail-over scenario.
4. It also offers content-aware distribution, by doing things such as reading URLs, intercepting cookies, and XML parsing.

Load balancing applies to all types of servers (application server, database server), however this paper is about FTP and HTTP server only.

Load balancing is a critical issue in parallel and distributed systems to ensure fast processing and good utilization. Load balancing involves IP Spraying when multiple web servers are present in a server group; the HTTP traffic needs to be evenly distributed among the servers. In this process, these servers must appear as one web server to the web client, for example an internet browser. The equipment used for IP spraying is also called the 'load dispatcher' or 'network dispatcher' or simply, the 'load balancer'. In this case, the IP sprayer intercepts each HTTP request, and redirects them to a server in the server cluster. Web server's performance is determined by the underlying hardware resources available to it. This limit is higher when the content delivered is static like images or text, but considerably lower when dealing with dynamic content. Load balancing involves spreading the load among multiple machines, or sometimes even among multiple sites, thereby increasing the resources available.

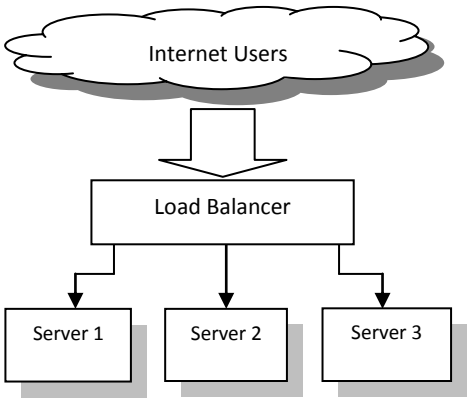

Fig 1 SLB simplified

Load balancing in its crudest form would, for example, involve placing all HTML files on one host, all images on another and all CGI scripts on the third. Real-life load balancing, however, involves carefully examining access patterns of various files on the website and keeping identical copies of the same Web server and distributing the load amongst them.

2. RELATED WORK

Qin Zheng [1] gives details on load distribution over computers in a group that leads to the minimum response time or computational cost. Many researchers work on how to reduce the computational cost in servers and how to improve the resource utilization in load balancers. For this a framework of weighted alpha rules[2] that can improve the global resource utilization and load balancing, which translates into a smaller blocking rate of MS arrivals without extra resources, while the aggregate throughput remains roughly the same or improved around the hotspots is presented. Increasing utilization of cluster web servers through effective and fair load balancing for heterogeneous as well as homogeneous network [8] is a crucial task specifically when it comes to advent of dynamic contents.

The exponential demands for high performance web servers led to use of cluster-based web servers especially for multimedia applications [7]. The algorithm to select dynamically servers from a class and assigns the request to a server is proposed by Saeed Sharifiana[3]. Also mobile agent based framework[4]

called Mobile Agent based Load balancing (MALD) that uses mobile agents technology to implement scalable load balancing on distributed web servers. The web servers can dispatch mobile agents to collect system-wide load information and accomplish load redistribution on all servers. Beside this A random selection algorithm [5] is proposed in which concurrent requests occur, a server will be selected randomly for clients from the optional servers which accord with anycast condition. The method not only ensures the QoS of anycast server but also provides clients with an optimal server. The algorithm has been proved to be feasible and efficient by simulation experiment. Experiments on minimizing use of computer hardware, software failures and mitigating recourse limitations by cloud computing [9], [10] are the hot area of research now days.

3. LOAD-BALANCING TECHNIQUES

Load balancing can be done through hardware- or software-based techniques. DNS load balancing [6], involves maintaining identical copies of the site on physically separate servers. The DNS entry for the site is then set to return multiple IP addresses, each corresponding to the different copies of the site. The DNS server then returns a different IP address for each request it receives, cycling through the multiple IP addresses. This method gives you a very basic implementation of load balancing. However, since DNS entries are cached by clients and other DNS servers, a client continues to use the same copy during a session. This can be a serious drawback, as heavy website users may get the particular IP address that is cached on their client or DNS server, while less-frequent users get another. So, heavy users could experience a performance slowdown, even though the server's resources may be available in abundance.

Another load-balancing technique involves mapping the site name to a single IP address, which belongs to a machine that is set up to intercept HTTP requests and distribute them among multiple copies of the Web server. This can be done using both hardware and software. Hardware solutions, even though expensive, are preferred for their stability. This method is preferred over the DNS approach, as better load balancing can be achieved. Also, these load balancers can see if a particular machine is down, and accordingly divert the traffic to another address dynamically. This is in contrast to the DNS method, where a client is stuck with the address of the dead machine, until it can request a new one.

Another technique, reverse proxying, involves setting up a reverse proxy, that receives requests from the clients, proxies them to the Web server and caches the response onto itself on its way back to the client. This means that the proxy server can provide static content from its cache itself, when the request is repeated [6]. This in turn ensures that the server itself can focus its energies on delivering dynamic content. Dynamic content cannot generally be cached, as it is generated real time. Reverse proxying can be used in conjunction with the simple load- balancing techniques discussed earlier, static and dynamic contents can be split across different servers and reverse proxying used for the static content Web server only.

Firewall Load Balancing [6] (FWLB), Fig. 2, has been developed to overcome some of the limitations of firewall technologies. Most firewalls are CPU-based, such as a SPARC machine or an x86-based machine.

Because of the processor limitations involved, the amount of throughput a firewall can handle is often limited. Processor speed, packet size, configuration, and several other metrics are all determining factors for what a firewall can do, but generally, they tend to max out at around 70 to 80 Mbps (Megabits per second) of throughput. Like SLB, FWLB allows for the implementation of several firewalls sharing the load in a manner similar to SLB. Because of the nature of the traffic, however, the configuration and technology are different. Figure 2 shows a common FWLB configuration.

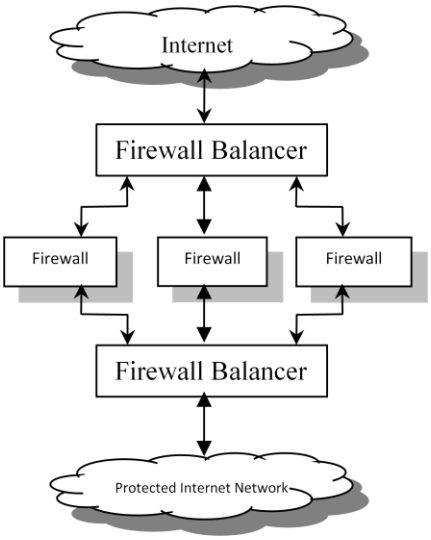

Fig 2 A common FWLB configuration

4. SIMULATION WORK

Studies can be done to determine the best load balancing policy to use for a particular. We use the image browsing as the internet application and determine the traffic received on the work stations with four different load balancing configurations using Load Balancer with 3000 users using the application. The following statistics are collected at the load balancer when load balancing is enabled. Statistics are collected per application and per server. Traffic Sent (in Bytes/sec or Packet/sec). The amount of traffic sent to a particular application server. Traffic Received (in Bytes/sec or Packet/Sec) The amount of traffic received from a particular application server. The following statistics are collected at each of the servers. CPU load (%), the percentage load on server CPU. The results shown on the CPU Utilization (%) are collected from the individual server who serves the workstations for One Hour.

4.1 No Load Balancing Configuration

Six client server are made, out of three are http servers and other three are ftp servers. Client1 is addressed to internet_server (server 1) as destination server. (Likewise all six clients are addressed to internet_server (server 1) as destination server) Client1 has supported the Web_User Profile. (Likewise all six clients are using the same profile).

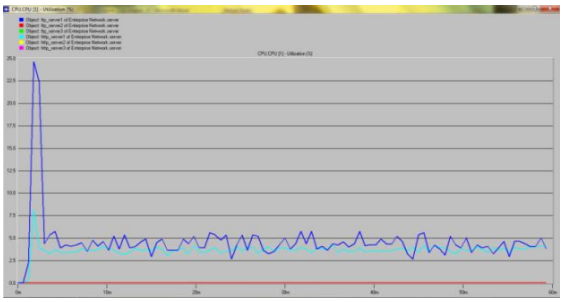

Fig 3 FTP Server (1-3) & HTTP Server (1-3) CPU Utilization (%) under No Load Configuration

4.2 Random Balancing Scenario

Load balancer randomly chooses from one of the three application servers according to the specified weight.

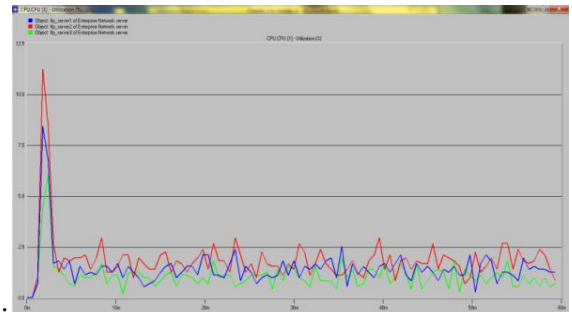

Fig 4 FTP Server (1-3) CPU Utilization (%) in random configuration.

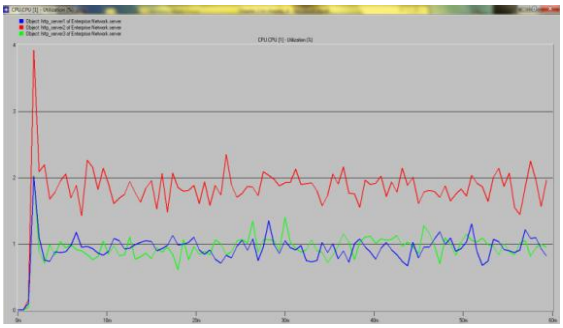

Fig. 5 HTTP Server (1-3) CPU Utilization (%) in random configuration.

4.3 Round Robin balancing configuration

This scenario uses the Round Robin fashion in which the load balancer chooses each application server in turn depending on the server weight. Fig.6 shows the CPU utilization of FTP servers, and of HTTP servers in Fig.7.

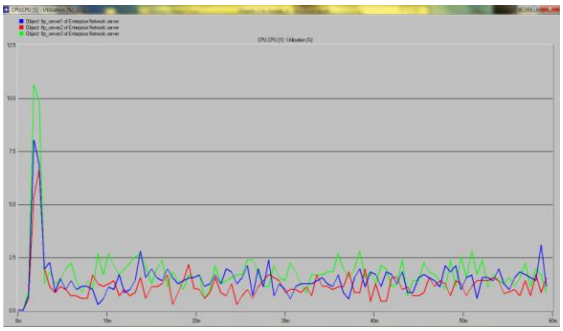

Fig. 6 FTP Server (1-3) CPU Utilization (%) in Round Robin configuration.

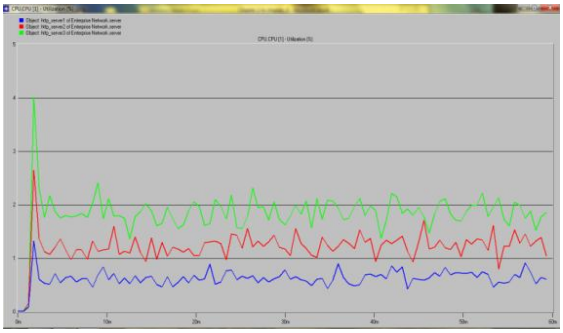

Fig. 7 HTTP Server (1-3) CPU Utilization (%) in Round Robin configuration.

4.4 Server Load balancing configuration

The fourth scenario uses the Server Load fashion in which the load balancer chooses the server with the lowest load at the time when a request is made. In this scenario, the load balancer is connected to the server through an Ethernet hub instead of individual links.

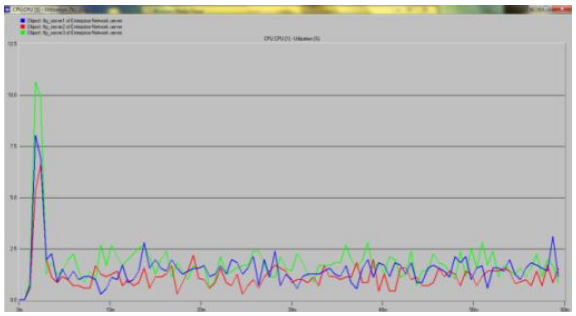

Fig. 8 FTP server (1-3) CPU Utilization (%) in Server Load Configuration

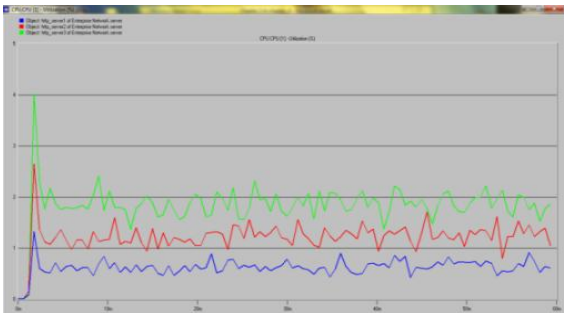

Fig. 9 HTTP Server (1-3) CPU Utilization (%) in Server Load Configuration

4.5 Number of connection Configuration for Load Balancing

In Number of Connection configuration scenario the load balancer tracks the number of open connections it has with each server. When a new request is received, it chooses the server with the least number of connections.

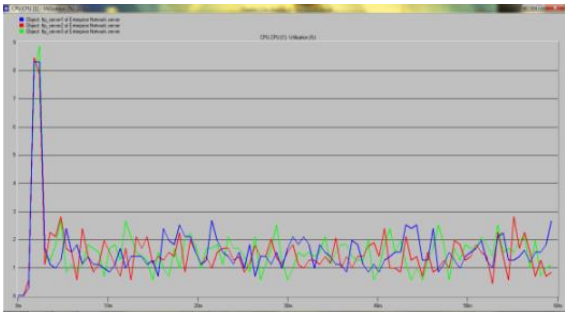

Fig. 10 FTP Server (1-3) CPU Utilization (%) in No. of Connection Load Configuration

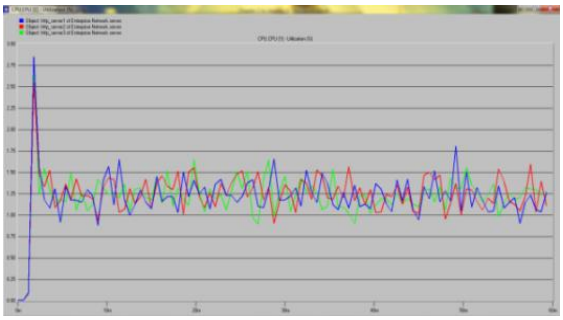

Fig. 11 HTTP Server (1-3) CPU Utilization (%) in No. of Connection Load Configuration

4.6 Server's performance in different Load configuration

Here each performance shows its performance individually in different Load Balancing configuration. So it is seen here that the Performance is looking sharply vary in different configuration.

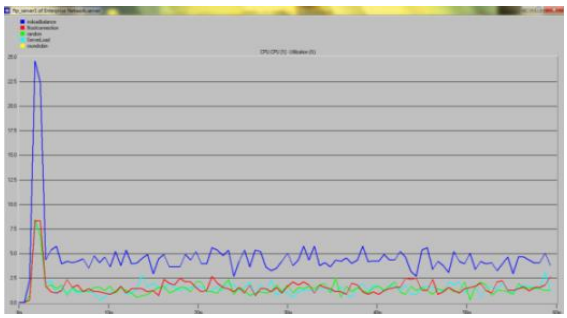

Fig. 10 Performance of FTP Server 1 in four load balancing configurations with no load balancing configuration

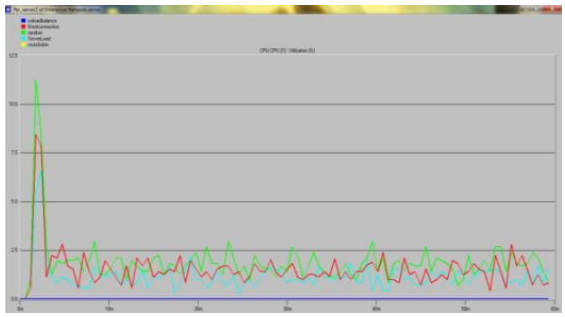

Fig. 11 Performance of FTP Server 2 in four load balancing configurations with no load balancing configuration

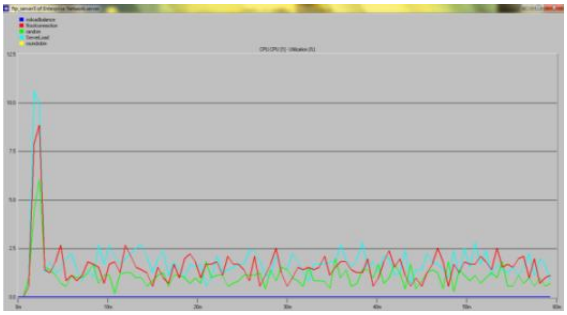

Fig. 12 Performance of FTP Server 3 in four load balancing configurations with no load balancing configuration

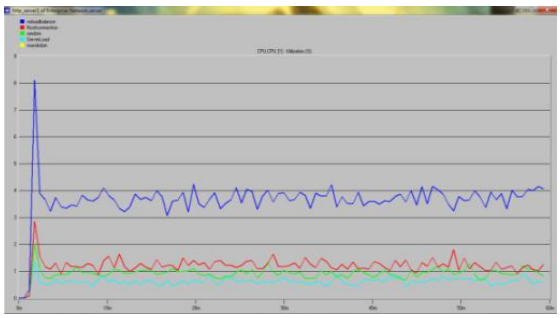

Fig. 13 Performance of HTTP Server 1 in four load balancing configurations with no load balancing configuration

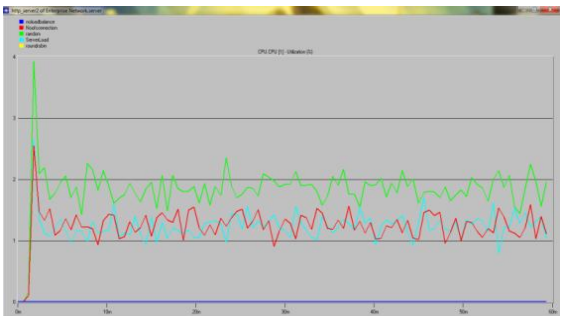

Fig. 14 Performance of HTTP Server 2 in four load balancing configurations with no load balancing configuration

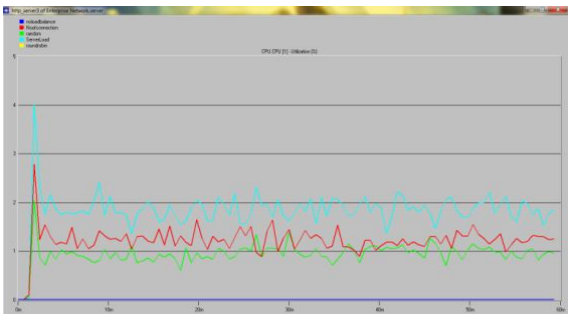

Fig. 15 Performance of HTTP Server 3 in four load balancing configurations with no load balancing configuration

So we see that in the all the load balancing configuration load balancer equally divide the load on the servers either it is

http server or ftp server. Also the there is some uneven distribution of the load in http application. The ftp application shows the somewhat equal distribution of load among the ftp servers. In this project we use only 3 servers for the ftp and3 servers for the http application, but in reality we have hundreds, thousands of servers for handling the applications.

5. CONCLUSION

The statistics which we are obtained are given below. The entire four scenarios have shown the CPU Utilization for the purpose for performance analysis. Table I shows the CPU Utilization for the single server. It is shown in the table that the server 1 is busy during the whole simulation period or during the whole busy working hours. In this case server1 is getting the whole Load.

Table 2 shows the CPU utilization for random configuration. In table 3, round robin configuration and table 4, server load configuration, data shows that server 3 is at shoot for this configuration. Table 5 shows that as no. of connections increases the ftp server CPU usage is increases. Thus it can be concluded that the effect of different load balancing configuration is very less for small network. But as the number of connections increases, FTP server CPU utilization is also increases.

Table 1 Top Objects Report in CPU Utilization (%) for No Load Balance Configuration

Rank	Object Name	Minimum	Average	Maximum
1	http_server1	0.000028	3.64	8.1
2	http_server2	0.000028	0.00	0.00
3	http_server3	0.000028	0.00	0.00
4	ftp_server1	0.000028	4.61	24.6
5	ftp_server2	0.000028	0.00	0.00
6	ftp_server3	0.000028	0.00	0.00

Table 2 Top Objects Report in CPU Utilization (%) for Random Configuration

Rank	Object Name	Minimun	Average	Maximum
1	http_server1	0.000028	0.92	2.0
2	http_server2	0.000028	1.83	3.9
3	http_server3	0.000028	0.94	2.0
4	ftp_server1	0.000028	1.46	8.4
5	ftp_server2	0.000028	1.88	11.2
6	ftp_server3	0.000028	1.09	6.0

Table 3 Top Objects Report in CPU Utilization (%) for Round Robin Configuration

Rank	Object Name	Minimum	Average	Maximum
1	http_server1	0.000028	0.62	1.3
2	http_server2	0.000028	1.21	2.6
3	http_server3	0.000028	1.83	4.0
4	ftp_server1	0.000028	1.51	8.0
5	ftp_server2	0.000028	1.16	6.6
6	ftp_server3	0.000028	1.83	10.6

Table 4 Top Objects Report in CPU Utilization (%) for Server Load Configuration

Rank	Object Name	Minimum	Average	Maximum
1	http_server1	0.000028	0.62	1.3
2	http_server2	0.000028	1.21	2.6
3	http_server3	0.000028	1.83	4.0
4	ftp_server1	0.000028	1.51	8.0
5	ftp_server2	0.000028	1.16	6.6
6	ftp_server3	0.000028	1.83	10.6

Table 5 Top Objects Report in CPU Utilization (%) for No. of Connection Configuration

Rank	Object Name	Minimum	Average	Maximum
1	http_server1	0.000028	1.21	2.85
2	http_server2	0.000028	1.24	2.55
3	http_server3	0.000028	1.22	2.79
4	ftp_server1	0.000028	1.60	8.31
5	ftp_server2	0.000028	1.53	8.45
6	ftp_server3	0.000028	1.59	8.87

6. REFERENCES

[1]. Zheng, Q. Chen-Khong Tham and Bharadwaj, V. J. 2008. Dynamic Load Balancing and Pricing in Grid Computing with Communication Delay, Grid Computing, 6:239–253 DOI 10.1007/s10723-007-9093-5

[2]. Aimin Sang, Madihian, M. , Richard D. Gitlin, 2004. Coordinated Load Balancing, Handoff/Cellsite Selection, and Scheduling in Multicell Packet Data Systems, MobiCom 2004, Philadelphia, Pennsylvania, USA.

[3]. Sharifiana, S., Motamedia, S. A.and Akbarib, M. A. 2010. A predictive and probabilistic load-balancing algorithm for cluster-based web Servers, Appl. Soft Comput. J. (2010)

[4]. Jiannong Cao, A., Sun, Y., Wang, X. and Das,S. K.2003. Scalable load balancing on distributed web servers using mobile agents, Parallel Distrib. Comput. 63 (2003) 996–1005

[5]. Zhou, Z., Xu, G., Deng，C. and Jiang, J. 2009. A Random Selection Algorithm Implementing Load Balance for Anycast on Application-layer, Proceedings of the 2009 International Symposium on Web Information Systems and Applications (WISA'09), May 22-24, 2009, pp. 444-4486.

[6]. Bourke, T., " Server Load Balancing", Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, 2001.

[7]. Guo,J. and Bhuyan,L. N. 2006. Load Balancing in a Cluster-Based Web Server for Multimedia Applications, IEEE Transactions On Parallel And Distributed Systems, Vol. 17, No. 11, November 2006.

[8]. Rommel, C. G.1991. The Probability of Load Balancing Success in a Homogeneous Network, IEEE Transactions On Software Engineering, Vol. 17, No. 9, September 1991.

[9]. Chaczko, Z., Mahadevan, V., Aslanzadeh, S. and Mcdermid, C. 2011.Availability and Load Balancing in Cloud Computing, International Conference on Computer and Software Modeling, Singapore, Vol. 14, 2011.

[10]. Mishra, R. and Jaiswal, A. 2012. Ant colony Optimization: A Solution of Load balancing in Cloud, International Journal Of Web & Semantic Technology (IJWEST) Vol.3, No.2, April 2012