SENRA Academic Publishers, Burnaby, British Columbia
Vol. 3, No. 1, pp. 691-700, 2009
ISSN: 1715-9997

Canadian Journal of pure&applied sciences

A JAVA TCP SERVER LOAD BALANCER: ANALYSIS AND COMPARISON OF ITS LOAD BALANCING ALGORITHMS

*Majdi Al-qdah[1] and Mohd Khir Bin Abu Yan[2]
[1]Faculty of Computer Science, University of New Brunswick, Canada
[2]Faculty of Information Technology, Multimedia University, Malaysia

ABSTRACT

In this study, a TCP server load balancer (SLB) programmed in Java is proposed as an affordable alternative to commercial or open source server load balancers for small companies by having all basic server load balancer features in order to maximize the usage of small companies financial, human and hardware resources. The features include load balancing algorithms namely Round Robin, Random, Least Connection and Hash IP Address. The Java TCP Server Load Balancer employs Rules which consist of the service, Virtual IP Address model, load balancing algorithm, and keep-alive. The Java TCP Server Load Balancer features are real server Health Checking and Graphical User Interface (GUI) for ease of configuration and administration. Subsequently the algorithms included in the balancer are studied and analyzed to compare their performance in the Linux Operating System environment with network traffic on different applications in order to find the most optimal use of the load balancing algorithms in the Java TCP Server Load Balancer. Analysis and comparison of the load balancing algorithms are conducted in experiments involving a number of test cases with clients, the Java Server Load Balancer, and real servers hosting HTTP and FTP applications. The most important conclusion from the experiments is that the performance of the two services tested namely HTTP and FTP is not actually directly influenced by the load balancing algorithm.

Keywords: Java, TCP, server, load balancer, balancing, algorithms.

INTRODUCTION

In the early years of websites, it was perfectly normal to have only one web server to serve million of requests from clients. This is due to the fact that most of the websites at that time have only static contents such as HTML and a few images (Ampornaramveth and Sanguanpong, 2002; Bourke, 2001). A server with average resource capacity was able to process these requests within acceptable response time to the clients (Casalicchio and Colajanni, 2001). As time went by, contents of websites have gradually shifted from static to dynamic types or in the case of FTP sites the downloaded file size is getting bigger. More websites are becoming interactive and more business driven websites, FTP sites and email service (Webmail) turn up as businesses see the opportunity of expanding their customer base through the Internet to gain more profit (DeRienzo, 2007). This has lead to websites having more dynamic contents due to data encryption, database applications, business logic, contents reformatting and others while FTP sites add more space for files and email service gets more subscribers (Chatterjee *et al*., 2005; Feng *et al*., 2000). Unfortunately, this causes these servers to process more data and receive more connections from clients. By using only one server, the service response time to clients starts to increase and in the Internet world even a few seconds increase can result in companies losing customers (Ho *et al*., 2004; Hong *et al*., 2006). The first reaction to this setback is to upgrade the server hardware such as CPU and RAM. This approach has had its own problems since there is a limit to hardware capacity of any server and its network components (Extremenetworks, 2007). Network engineers worked hard to find a viable solution to this problem and eventually came up with a clever solution. This solution is server load balancing. A Server load balancer is able to make a group of servers behind a server load balancer appearing to clients as a single server and is capable of distributing clients' requests statically or dynamically to the actual servers (real servers) that provide the services to the clients (Viswanathan, 2001).

*Corresponding author email: majdi.qdah@gmail.com

Load Balancing Algorithms

A server load balancer typically works by load balancing traffic to the real servers based on load balancing algorithms (Lu and Lee, 2005; Min *et al*., 1999). There are various load balancing algorithms used, with the most common ones to be Round Robin, Random, Least Connection, Source IP Address Hash (Hash IP), URL hash, and Cookie. There are also custom or proprietary load balancing algorithms which were developed by commercial server load balancer vendors or were developed for research. The algorithms are usually the variants of the common load balancing algorithms such as Weighted Round Robin or Weighted Least Connection. The purpose of these types of algorithms is usually to improve the performance of load distribution to the real servers by getting as much information as possible about the real servers or clients' states so that the server load balancer is able to determine the optimal traffic distribution.

MATERIALS AND METHODS

The Java TCP SLB Software Operation

The Java TCP server load balancer is a software application written in Java that accepts TCP connections from clients and then distributes these connections to real servers which serve the clients' requests for specific TCP based services such as HTTP or FTP. The server load balancer is placed between the clients and the real servers in the network connections path. The client connections to the real servers are distributed using load balancing algorithms such as Round Robin or Least Connection. The load balancing configuration is done only on the Java TCP Server Load Balancer itself while the real servers provide the services to the clients. The Java TCP Server Load Balancer offers service availability discovery through Health Checking feature. The Java TCP Server Load Balancer application is administered and configured by using a Java GUI accessible from the host where the Java TCP Server Load Balancer is running.

Load Balancing Algorithms Comparison

The purpose of the comparison is to find the most optimal use of load balancing algorithm in the Java TCP Server Load Balancer. This is performed with simulated traffic from clients to real servers. In the Round Robin algorithm the traffic is sent to the real servers in ordered sequence and repeated in a loop. Each of the real servers receives equal number of connections from clients regardless of its capacity or load. The Random algorithm main use is to distribute connections randomly to any of the real servers. In the Least Connection algorithm, traffic is distributed to the server that has the least connections and the real server handles connections equally with other available servers. The Hash IP Address algorithm causes the subsequent connections from the same client IP address to connect to the real server where the first connection from that IP address is connected.

The Design

Network Architecture

As figure 1 show, the Java TCP Server Load Balancer runs on a host which has at least two network interface cards (NIC). One NIC is used for outside network connection such as Wide Area Network (WAN) and the Internet. The other NIC is used for the internal network where the real servers are located or inside a Local Area Network (LAN). Figure 2 shows the interface of the Load Balancing Algorithm.

In using the SLB, below is an example of a step-by-step how-to based on specific requirement below:

Service name:	http
Service port:	80
IP Address:	0.0.0.0
Terminate on Disable:	No
Half Close:	Yes
Connection TimeOut:	2 seconds
Connection Failure Limit:	5
Real servers:	realsever1.fit6z.com
	realsever2.fit6z.com
	realsever3.fit6z.com
	realsever4.fit6z.com
Algorithm:	Round Robin
KeepAlive:	HTTP

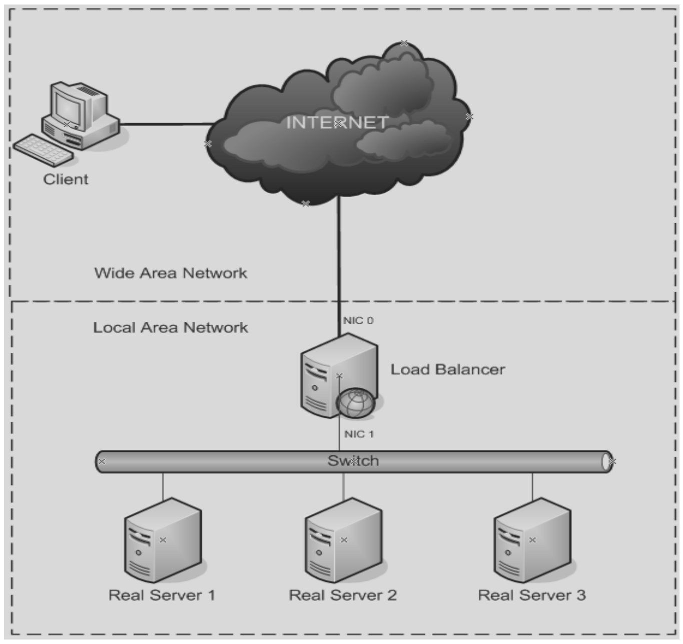
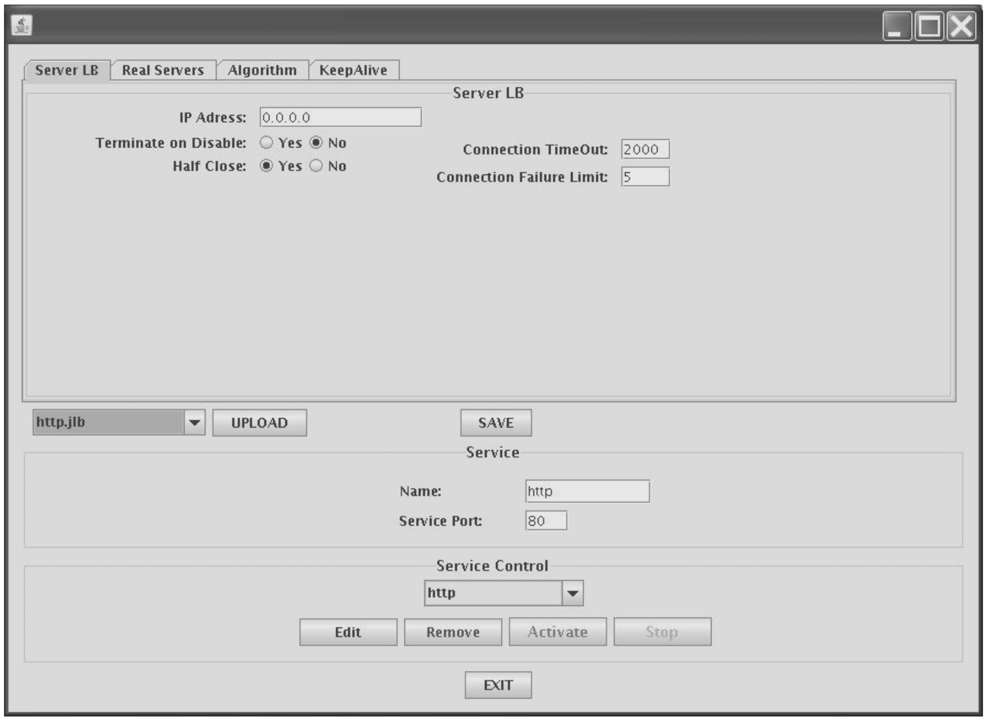


Fig. 1. Overall network architecture.

Frequency:	60 seconds
TimeOut:	5 seconds
Doc Path:	/index.html
Response Code:	200
Doc Text:	TARGET

In the experiments performed, the HTTP and FTP services were used for testing. Ten different clients with different private IP addresses were configured to simulate the traffic requesting for the services. Automated scripts on each client were configured to request the services above at thirty minutes interval. Each client had a script to measure the services' response time while the Java TCP Server Load Balancer and the real servers have a monitoring program to measure used resources: CPU and Memory load.



Server LB

IP Adress: 0.0.0.0

Terminate on Disable: Yes [✓ No]

Half Close: [✓ Yes] No

Connection TimeOut: 2000

Connection Failure Limit: 5

Service

Name: http

Service Port: 80

Service Control

Fig. 2. Server LB tab.

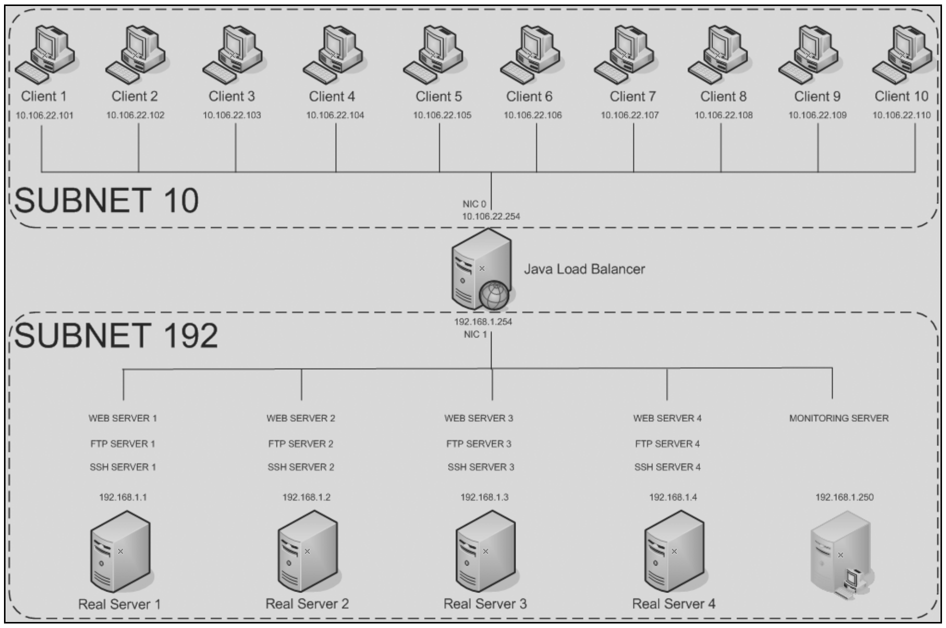


Fig. 3. Experiment network diagram.

Comparison Metrics

Service Response Time

This is the most important metric as the performance of the service is very dependent on this metric. This metric show how long the client takes to complete access to the service. This metric has a definition that varies slightly from service to service. In HTTP, it is defined as the time it takes for a client to completely browse and download pages from a website with pre-configured path/links using Iceweasel add-on iMacros; while in FTP, it is defined as the time it takes for a client to complete a single FTP session to the FTP server by authenticating itself and then downloading 10 files with different sizes until the connection is closed.

Real Server and Java SLB CPU and Memory load

CPU and Memory load is defined as the total percentage of CPU and free Memory on the real server and the Java TCP SLB (RADirect, 2007). The CPU and Memory load on the Server Load Balancer is a good indication of how well is the performance of the individual algorithm running on SLB since the SLB software is the most active application running during the experiment in the SLB. However the CPU and Memory load on the real servers are not good indicators of the individual algorithm performance because the CPU and Memory load is the result of the process of accessing and running the application such as web server or FTP server. It is not the load from running the SLB software. However in order to see the performance of the algorithm, the distribution of the load among the servers was used. This can be achieved by calculating the standard deviation of the data gathered from the real servers CPU and Memory load. This is where CPU and Memory load on the real servers can be used, that is to measure the distribution of the CPU and memory load. The smaller the standard deviation the more even the distribution of the load among the real servers, which implies a better algorithm.

The experiment objective was to compare four load balancing algorithms which were employed by the Java TCP Server Load Balancer. In order to simulate real life environment, ten clients, four real servers and one monitoring station were required for the experiment apart from the most essential host running the server load balancer as illustrated in figure 3.

RESULTS AND DISCUSSION

Each client recorded the individual response time from each service test. One file was created for every service on each client using the script on the client. This file contained the response time from each algorithm test performed when the SLB was running Linux Debian 4.0.

HTTP

The average response time from each load balancing algorithm test was computed for each client for the HTTP service with SLB running on Linux Debian and was recorded in table 1.

The average of the values for each algorithm was then computed to get the average for specific algorithm response time using all the values from the clients. Based on the results from table 1, the algorithms' response times are close to each other, which imply that the performance of the HTTP service is generally the same among the real servers no matter which algorithm was used. Figure 4(a-d) shows the CPU load on the SLB against running each algorithm for thirty minutes.

These results show that the distribution process of the load on the SLB for HTTP service was very light. It was less than 10 percent of CPU usage on average for any algorithm. This indicates that the SLB does not require high CPU power to process clients' HTTP requests. Possible reasons for this event is the size of files i.e. data that was transferred from real servers to clients which was

Table 1. HTTP Service Response Time.

	Service Average Response Time (seconds)			
	Round Robin	Hash IP	Least Conn.	Random
Client 1	239.56	244.68	248.38	249.84
Client 2	291.12	125.94	299.85	303.45
Client 3	209.52	212.47	216.86	219.69
Client 4	224.54	227.84	233.24	236.28
Client 5	194.78	197.26	199.53	202.45
Client 6	219.41	225.44	225.11	231.00
Client 7	209.29	207.81	210.15	211.14
Client 8	115.05	115.47	116.82	118.54
Client 9	255.09	258.66	261.22	263.04
Client 10	165.18	165.93	167.24	174.11
Algorithm Average Response Time (seconds)	212.35	198.15	217.84	220.95

not very large. Thus the transfer of the static HTML files and images do not require high CPU usage. The total size of the website on the real server was 152 MB but the majority of individual websites files were small.

Figure 5(a-d) shows the Memory load on the SLB against running the algorithm for thirty minutes and a table 2 shows the maximum Memory usage in each thirty minute test.

The above results show that generally the SLB Memory usage gets higher as the test progresses. The difference is in the pattern of Memory usage. The pattern for Round Robin algorithm is especially a little bit different than those of the other three algorithms. With Round Robin, the usage fluctuation is significant; it is relatively high and occurs quite early compared to other algorithms. Looking at the Maximum Memory Usage table, Round Robin has the lowest usage of Memory. This means that the size of fluctuation of usage of the Round Robin algorithm actually is similar to the other algorithms because the graphs of the other three algorithms have a larger maximum value of y-axis so it appears that the fluctuation of usage is smaller. The difference with Round Robin is that the fluctuation of usage starts early. These results also show that for HTTP service, Round Robin is the best algorithm in terms of Memory usage on the SLB.

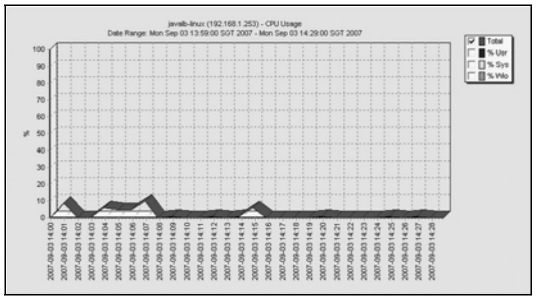


Fig. 4(a). Round Robin.

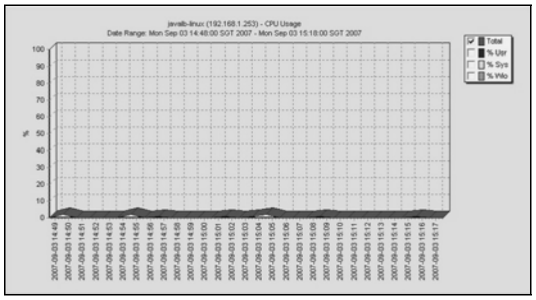


Fig. 4(b). Hash IP Address.

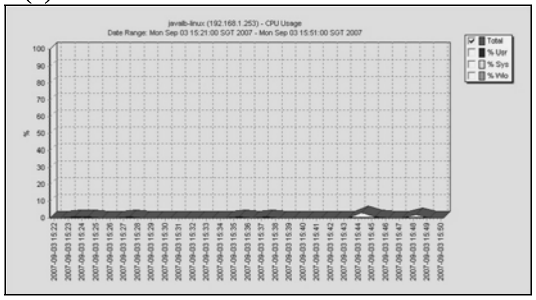


Fig. 4(c). Least Connection.

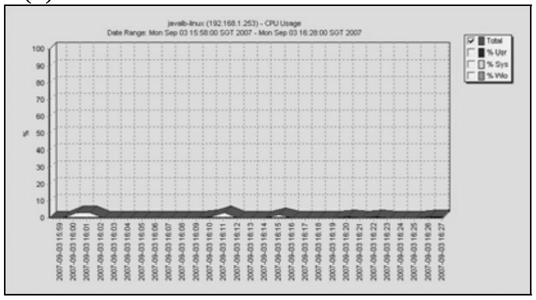


Fig. 4(d). Random.

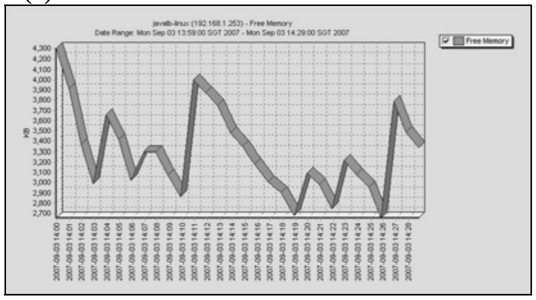


Fig. 5(a). Round Robin.

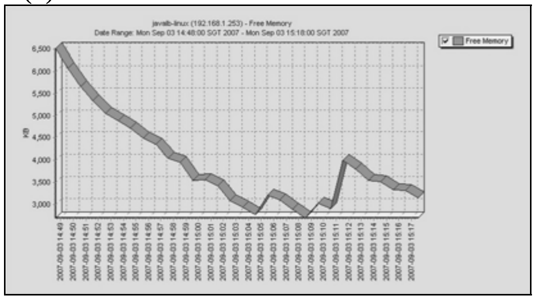


Fig. 5(b). Hash IP Address.

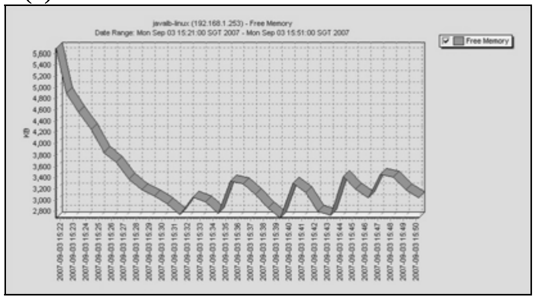


Fig. 5(c). Least Connection.

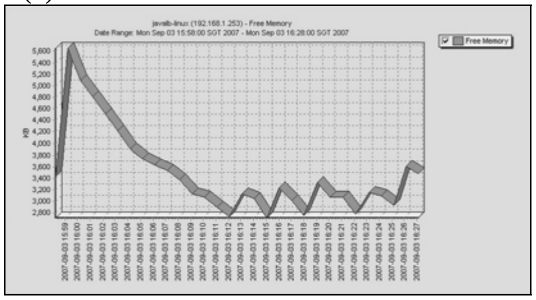


Fig. 5(d). Random.

Table 2. SLB maximum memory usage.

Java SLB	Round Robin	Hash IP	Least Conn.	Random
Maximum Memory Usage (kiloBytes)	1600	3800	2800	2800

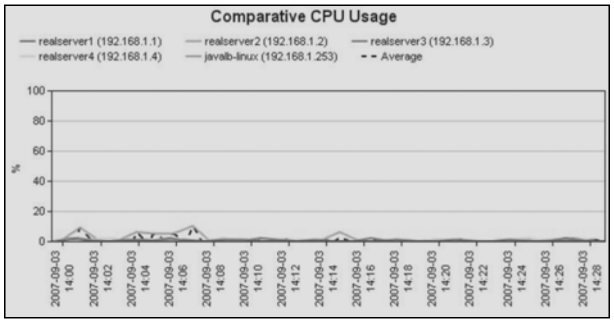


Fig. 6(a). Round Robin.

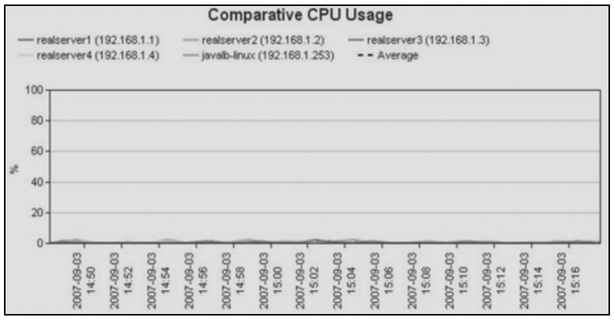


Fig. 6(b). Hash IP Address.

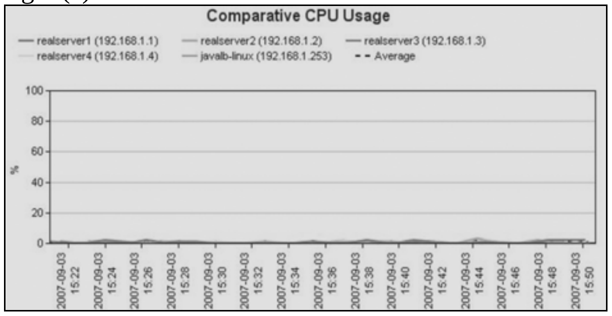


Fig. 6(c). Least Connection.

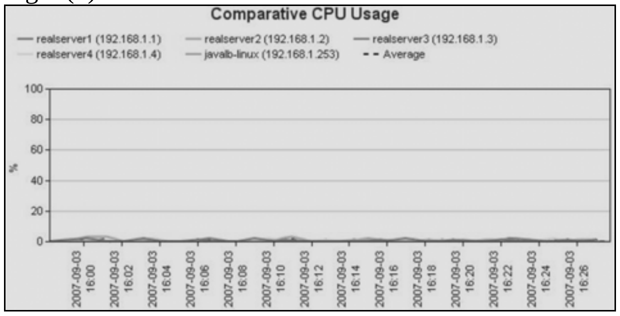


Fig. 6(d). Random.

Figure 6 (a-d) shows the CPU load on the real servers against running each of the algorithms for thirty minutes.

Based on the above results, the CPU usage on the real servers is very low for the same reasons mentioned previously i.e. the HTML files and images are small and also the HTTP server processes the client static files requests requiring only low CPU power. There is also no significant difference in performance among the algorithm. As mentioned under the "performance metric", it is not the CPU or Memory usage that indicates algorithm performance, it is the distribution of the load that is important. From these results since the load on each real servers are close to each other, it means that the load is distributed evenly among the real servers.

Figures 7(a-d) to 10(a-d) show the Memory load on the real servers against running each one of the algorithms for thirty minutes and tables that show the maximum Memory usage in each thirty minute test. Tables 3 to 6 show the maximum memory usage for each one of the algorithms.

The results show that in all test cases the free Memory was decreasing as the tests progressed. This was expected and the only difference was just the rate of usage towards the end of the test and the maximum usage. There was no significant event that needed attention based on these results.

FTP

The average of response times from each load balancing algorithm test was calculated for each client for FTP service with SLB running on Linux Debian and recorded in the table 7.

Based on the results from the table, the algorithm response times are quite close to each other which imply that the performance of the FTP service is generally the same among the real servers no matter which algorithm is used. This results show that the distribution process of the load on the SLB for FTP service requires more than 30 percent of CPU usage on average for any algorithm. This indicates that the SLB requires significant CPU power to process client request to transfer files. Possible reason for this event is the size of files i.e. data that is transferred from real servers to clients is bigger than that of in HTTP service. Thus the transfer of ten files with multiple sizes requires significant CPU usage. Also, the results show that generally the SLB Memory usage gets higher as the test progresses except for Hash IP Address and Least Connection algorithms. In Hash IP Address test, the rate of Memory usage decreases towards the end of the test (increase of the graph means increase of free Memory) while with Least Connection the Memory usage drops very rapidly after about 20 minutes. A possible explanation for this outcome is that the FTP session finishes early compared to other algorithms. The Least

Round Robin

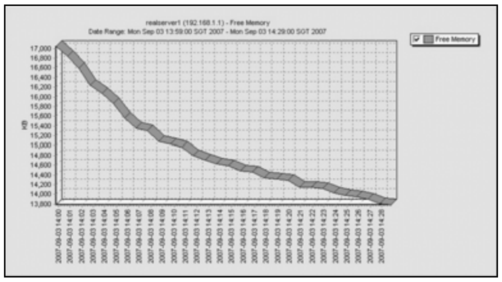


Fig. 7(a). Real Server 1

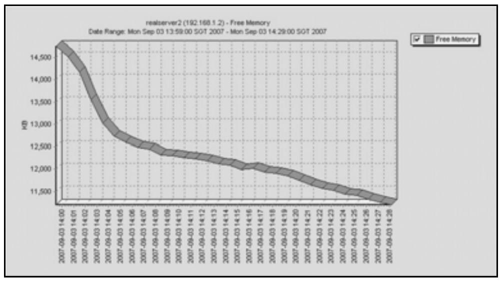


Fig. 7(b). Real Server 2

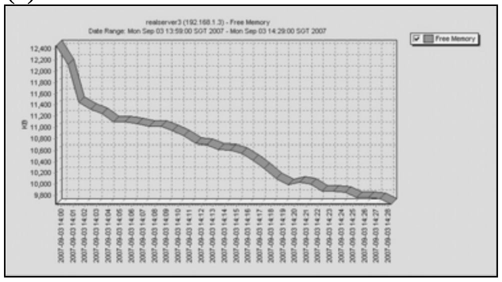


Fig. 7(c). Real Server 3.

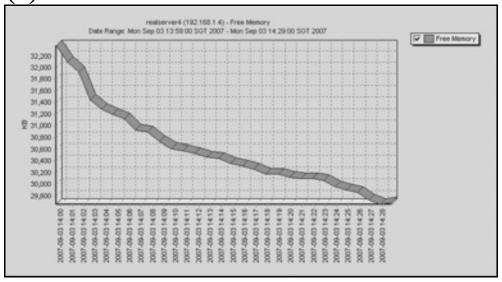


Fig. 7(d). Real Server 4.

Table 3. Maximum memory usage with Round Robin.

	real server 1	real server 2	real server 3	real server 4
Maximum Memory Usage (kiloBytes)	3200	3500	2800	2700

Hash IP Address

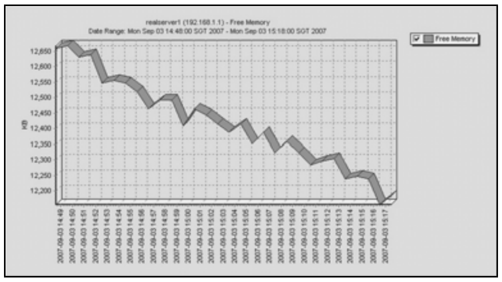


Fig. 8(a). Real Server 1.

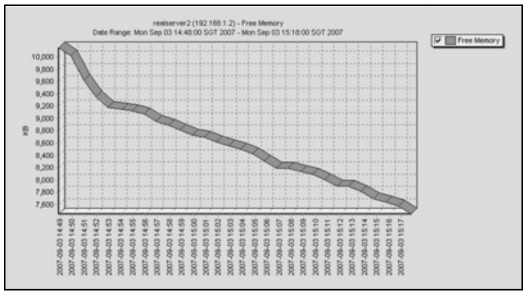


Fig. 8(b). Real Server 2.

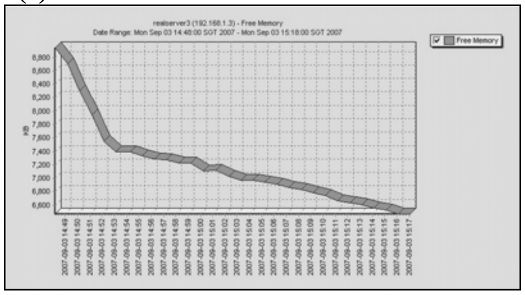


Fig. 8(c). Real Server 3.

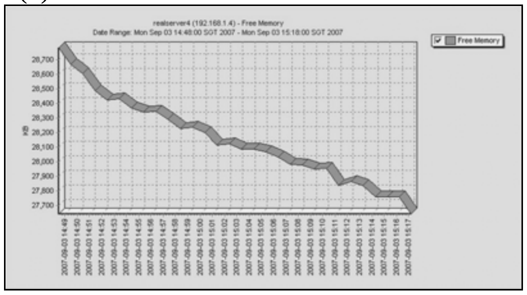


Fig. 8(d). Real Server 4.

Table 4. Maximum memory usage with Hash IP Address.

	real server 1	real server 2	real server 3	real server 4
Maximum Memory Usage (kiloBytes)	500	2600	2400	1100

Connection algorithm had the lowest usage of Memory. This value does not take into account the Memory usage drop because the values after the drop are not likely contributed by the FTP session. It is the maximum usage from the start of the test until the Memory usage starts to drop.

Least Connection

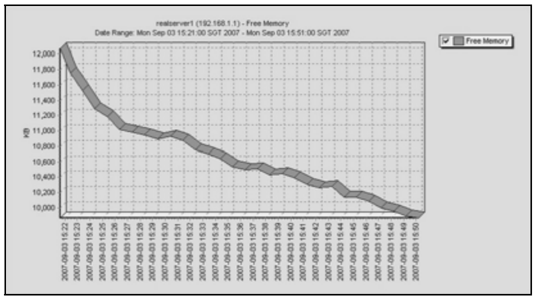


Fig. 9(a). Real Server 1.

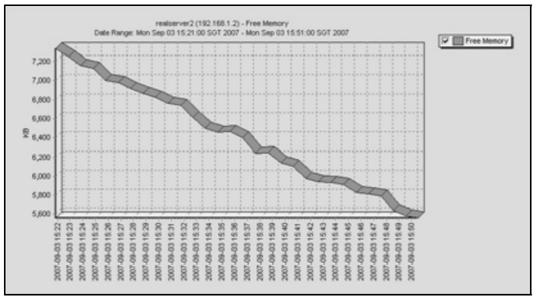


Fig. 9(b). Real Server 2.

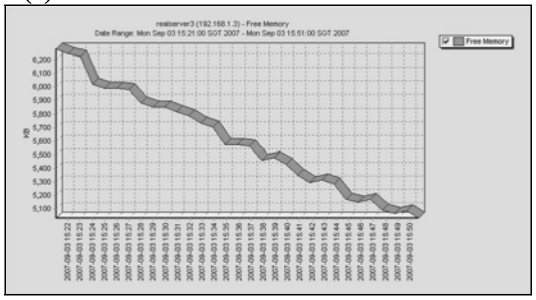


Fig. 9(c). Real Server 3.

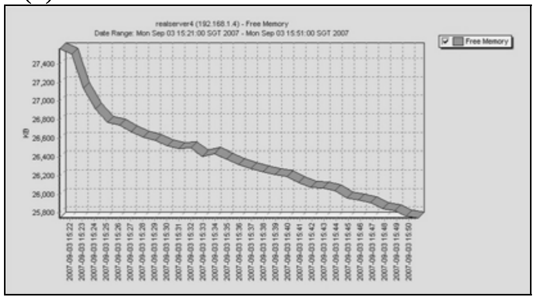


Fig. 9(d). Real Server 4.

Table 5. Maximum memory usage with Least Connection.

	real server 1	real server 2	real server 3	real server 4
Maximum Memory Usage (kiloBytes)	2200	1700	1200	2800

Random

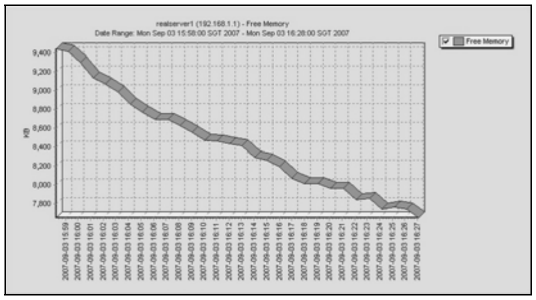


Fig. 10(a). Real Server 1.

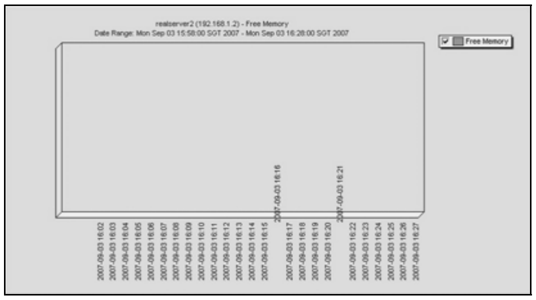


Fig. 10(b). Real Server 2.

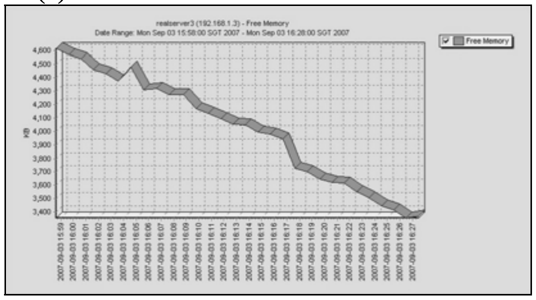


Fig. 10(c). Real Server 3.

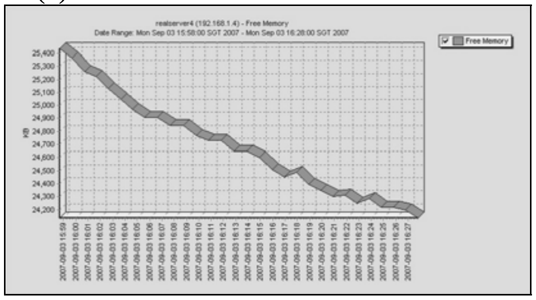


Fig. 10(d). Real Server 4.

Table 6. Maximum memory usage with Random.

	real server 1	real server 2	real server 3	real server 4
Maximum Memory Usage (kiloBytes)	1800	(N.A.)	1250	1300

In addition, the results show that for FTP session, the real servers have used a significant CPU power. There are two important characteristics that can be concluded: the first one is the closeness between the graph lines representing the CPU usage load and the second one is the fluctuation of individual line. The first characteristic represents the

Table 7. FTP Service Response Time.

	Service Average Response Time (seconds)			
	Round Robin	Hash IP	Least Conn.	Random
Client 1	300.11	338.26	319.27	336.49
Client 2	322.34	333.09	319.21	327.22
Client 3	321.91	302.50	340.92	344.02
Client 4	317.49	305.56	320.78	326.14
Client 5	332.97	356.43	326.97	328.49
Client 6	323.98	305.55	317.91	334.76
Client 7	388.45	371.68	346.08	345.18
Client 8	327.37	367.00	321.29	341.00
Client 9	326.45	330.74	337.63	339.72
Client 10	332.62	299.82	336.57	320.08
Algorithm Average Response Time (seconds)	329.37	331.06	328.66	334.31

Table 8. Algorithms Rank with Linux.

APPL	HOST	METRIC	ALGOIRTHMS' RANK				STD. DEV
			1st	2nd	3rd	4th	1st Algo
HTTP	Client	Response Time	HIP	RR	LC	RD	N.A.
	SLB	CPU	No significant difference				
		Memory	RR	LC	RD	HIP	N.A.
	Real Servers	CPU	LC	HIP	RD	RR	
		Memory	RD	RR	LC	HIP	304.13
FTP	Client	Response Time	LC	RR	HIP	RD	N.A.
	SLB	CPU	No significant difference				
		Memory	LC	HIP	RR	RD	N.A.
	Real Servers	CPU	LC	RR	RD	HIP	
		Memory	LC	HIP	RR	RD	221.27

load distribution among the real servers. Closer graph lines mean a more even distribution of load. The second characteristic represents the change of load on that particular real server. It can be observed that each graph shows both characteristics but differ in magnitude. The graph that has the closest lines with each other is the Least Connection algorithm graph and the least is the Random algorithm graph. This means Least Connection algorithm distributes load the most even among the algorithms and Random algorithm is the least efficient in distributing the load evenly based on the CPU usage. It is also observed that the Hash IP Address graph lines are the least fluctuating and the most fluctuating is Random algorithm. This is expected since Hash IP algorithm keeps the same source IP address into the same real server. This causes the CPU to process the same load from the same client which results in least fluctuating CPU usage. The real server results show that in all test cases the free Memory was decreasing as the tests progressed. This is expected and the only difference is just the rate of usage towards the end of the test and the maximum usage. The Memory usage is not directly contributed by the algorithm; instead it is a direct result from the processes running on the real servers themselves. There is no significant event that needs attention based on these results.

Load Balancing Algorithms Rank

Based on the previous tables that summarized the results, it is possible to rank the algorithms in terms of the efficiency of the algorithms in distributing requests to specific application. The algorithm rank is shown in table below:
(N.A. means Not Applicable)

Legend:	RR	=	Round Robin
	HIP	=	Hash IP Address
	LC	=	Least Connection
	RD	=	Random

CONCLUSIONS

The Java TCP Server has been developed with big improvement to an existing load balancing algorithms. The most important of improvements is the addition of two more algorithms namely Least Connection and Random algorithms. In the experiment performed, three performance metrics have been used to determine the best algorithm for each of the HTTP and FTP services. The performance metrics are service response time, CPU load and Memory load of the Java TCP Server Load Balancer and the real servers. The most important conclusion from the experiment is that the performance of the services tested namely HTTP and FTP is not actually directly influenced by the load balancing algorithm. Also, the experiments have concluded that each of the Java TCP Server Load Balancer algorithms distributes the load to real servers exactly the way it was suppose to do. This is proven by real servers CPU and Memory load usage data and graphs, specifically the standard deviation value. Smaller standard deviation means a more even load distribution.

REFERENCES

Ampornaramveth, N. and Sanguanpong, S. 2002. Optimization of Cluster Web Server Scheduling from Site Access Statistics. [Online]. Available: http://anres.cpe.ku.ac.th/pub/WebCluster-anscse2002.pdf [2007, 02 April].

Bourke, T. 2001. Server Load Balancing. O'Reilly and Associates, Inc., California, USA.

Casalicchio, E. and Colajanni, M. 2001. A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services. [Online]. Available: http://www10.org/cdrom/papers/434/colajanni_html.html #Sec:class [2007, 01 April].

Chatterjee, D., Tari, Z. and Zomaya, A. 2005. A Task-Based Adaptive TTL Approach forWeb Server Load Balancing. Proceedings of the 10th IEEE Symposium on Computers and Communications. 877-884.

DeRienzo, F. 2007. Choosing a Hardware Load-Balancing Device. Macromedia, Inc. [Online]. Available:

http://www.adobe.com/devnet/server_archive/articles/choosing_hardware_lbdevice.html [2007,05 April].

Extremenetworks. 2007. TechBrief Extreme Networks Server Load Balancing [Online]. Available: http://apps.extremenetworks.com/libraries/whitepapers/technology/Server_load_balancing.pdf [2007, 02 April].

Feng, Y., Li, D., Wu, H. and Zhang, Yi. 2000. A Dynamic Load Balancing Algorithm Based on Distributed Database System. The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. 2:949-952.

Ho, LK., Sit, HY., Ho, KS., Leong, HV. and Luk, RWP. 2004. Improving Web Server Performance by a Clustering-Based Dynamic Load Balancing Algorithm. Proceedings of the 18th International Conference on Advanced Information Networking and Application. 2: 232-235.

Hong, YS., NO, JH. and Kim, SY. 2006. DNS-Based Load Balancing in Distributed Web-server Systems. Proceedings of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Second International Workshop on Collaborative Computing, Integration, and Assurance. pp4.

Lu, YC. and Lee, LT. 2005. On implementation of an efficient multi-purpose load balancing server. [Online]. Available:http://www.hpc.csie.thu.edu.tw/cthpc2005/pdf/II_4.pdf [2007, 01 April].

Min, D., Choi, E., Lee, D. and Park, B. 1999. A load balancing algorithm for a distributed multimedia game server architecture. 1999 IEEE International Conference on Multimedia Computing and Systems. 2:882.

RADirect. 2007. Server Load Balancing with Radware's WSD. [Online] Available: http://www.rad-direct.com/Application-server-load-balancing.htm [2007, 03 April].

Viswanathan, V. 2001. Load Balancing Web Applications. In O'reilly On Java.com [Online]. Available:http://www.onjava.com/pub/a/onjava/2001/09/26/load.html [2007, 02 April].