A RELATIVE STUDY OF LOAD BALANCING AND MANAGEMENT TECHNIQUES IN A DISTRIBUTED SYSTEM: THE IUB CASE STUDY

Dost Muhammad Khan[1], Najia Saher [2], Faisal Shahzad[3], Nawaz Mohamudally[4]
[1, 2, 3] Department of Computer Science & IT, The Islamia University of Bahawalpur, Pakistan
E-mail: khan.dostkhan@iub.edu.pk, {najiasaher, faisalsd}@gmail.com
[4] School of Innovative Technologies and Engineering (SITE), University of Technology, MAURITIUS, alimohamudally@utm.intnet.mu

ABSTRACT: *Nowadays, an appropriate distribution and management of load across the various systems in distributed environment is indispensable due to the heavy load of users' requests particularly on the main server. The problem of congestion and slow processing of user requests can be solved by using a suitable Load-Balancer which helps the user to get faster and consistent response time by directing the traffic to the least loaded and most responsive system. In this paper we discuss various load balancing and management techniques that are commonly used and furthermore, we make a relative study of these load balancing techniques after deploying them in a distributed system of the IUB.*

Key-words: Cluster, Load-Balancer, Round Robin, EquiLoad, SITE-A, Weighted Round Robin

1. INTRODUCTION

Load balancing is an even distribution of the load amongst all serving entities in a distributed environment. Load balancing and management is a process of grouping the servers participate in the same service to do the same work. The main purpose of load balancing and management is to increase availability, improve throughput, reliability, maintain stability, optimize resource utilization and provide fault tolerant capability. As the number of servers grows, the risk of a failure increases and such failures must be handled carefully. The ability to maintain unaffected service during any number of simultaneous failures is termed as high availability [16-18]. Load balancing is very essential in distributed systems to improve the quality of service by managing loads that change over time. The incoming requests demand the even distribution among the available systems in order to avoid resource bottlenecks and the full utilization of available resources. Load balancing also provides horizontal scaling e.g., adding computing resources in order to address increased loads [19]. The main purpose of load balancing and management in a distributed system is to transfer the work submitted by users to a lightly loaded member server instead of a heavily loaded member server. Improved Performance, Equality of Job, Fault Tolerance, Modifiability and System's Stability are some of the main objectives of load balancing and management.

The wide spread of networks has imposed new needs that required new paradigms and new technologies. There are several network technologies available which support user-level communication between processing a shared-memory. The client-server architectures are commonly used in distributed environment due to optimization, modularly, no wastage of resources, reliability, availability and provides graphical user interface aid. The ever growing amount of data that are stored in distributed form over networks of heterogeneous and autonomous sources poses several problems such as network bandwidth, communication, autonomy preservation, scalability, data buffering and privacy protection. The client-server computing is an environment that satisfies the business needs by appropriately allocating the application processing between the client and the server processors. The client requests services from the server; the server processes the request and returns the result to the client [1-2]. Figure 1 depicts a client-server model.

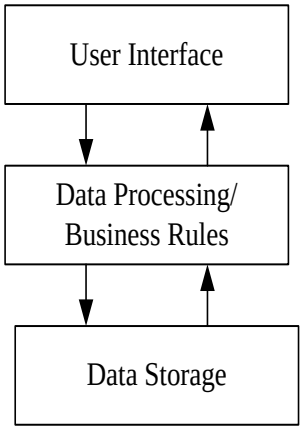

Figure 1. A Client-server Model

One of the primary advantages of client-server architecture is that as data storage needs grow without affecting clients the way data is stored can be changed. The middle layer of system is commonly referred to as the application server can thus concentrate on centralizing business rule processing. The client-server model is based on the idea that one computer specializing in information presentation displays the data stored and processed on a remote machine. A multi-user application is a slight variation on the typical client-server application. The only difference is that information passes from one client through the server to other clients. On a typical client-server application, information flows only from the client to the server and then back. In an ideal environment, the server side of the application handles all common processing and the client side handles user-specific processing [1-2, 12-15]. In client-server based networking environment where the main server remains under stress due to the heavy load of users' requests therefore, a load-balancer is essential.

The rest of the paper is organized as follows: Section 2 reviews the load balancing and management techniques, Section 3 is about deploying these techniques over a distributed system of the IUB which is the Methodology of the paper. In Section 4 we discuss the results and dicussion and finally Section 5 presents the conclusion.

2. Load Balancing and Management Techniques

The techniques or algorithms are further divided into homogeneous and heterogeneous servers. The problem of congestion and slow user-request processing speeds can be solved by using a single large powerful server. This solution soon fails because of the enormous network traffic. The second solution is replicating the server information over many geographically separated independent servers called "mirrored-server" architecture. This will solve the problem of congestion but with a number of disadvantages including huge loss of network and computer resources and lack of control on the request distribution by server system. A promising and efficient approach is the development of distributed architecture where the user-requests can be routed among several server nodes. This solution of distributed servers being managed under a single system provides us with improved throughput performance. Thus a server system with ease of manageability, greater availability and scalability of the servers is attained. This will increase user satisfaction because the user get faster, more consistent response time, directing traffic to the least loaded and most responsive servers and also prevent servers from getting overloaded.

Figure 2 depicts a simple load-balancer in a client-server architecture where the requests of a client for the services terminate at Load-Balancer which in turn forwards the requests to the servers based on various load balancing algorithms and mechanisms.

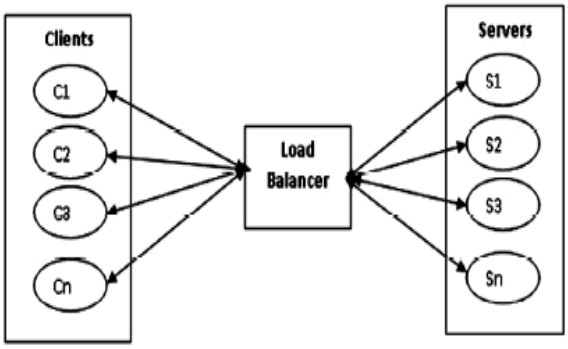

Figure 2. A Load-Balancer in a Client-Server Model [16-17]

Load balancing and management techniques are classified into two types namely Static Load Balancing and Dynamic Load Balancing. The main purpose of these techniques is to improve performance by redistributing the workload among available server nodes [20-21]. A comparison of static and dynamic load balancing techniques is drawn using different measures in [16-17].

2.1. Round Robin Technique

Assignment of jobs in a Round Robin to each of the member servers is suitable for homogeneous servers with same processing capabilities and size of jobs (data to be transferred) are almost same from each client. In such situations, there is no need of introducing processing overheads in selecting the appropriate server for redirection of data transfer request. IP addresses of the member servers are stored in a dynamic array (vector) and addresses are fetched on turn basis from zero index position to the last index of the vector. This is a cyclic process. The algorithm is given below:

Step 1: Initialize pointer to zero: index = 0
Step 2: Loop till Index = size of the vector – 1
Step 3: Get IP address from the location pointed by Index
Step 4: Increment Index by 1
Step 5: If Index = size of the vector – 1, re-initialize pointer with zero
Step 6: Loop end

We test this algorithm on four member servers connected to the cluster manger and the IP addresses of the servers are stored in a vector at the cluster manger. On first request, request is routed to server with IP address at index = 0 (192.168.0.1) and index is incremented. On second request, request is routed to server with IP address at index = 1 (192.168.0.2) and index is incremented. On third request, request is routed to server with IP address at index = 2 (192.168.0.3) and index is incremented. On fourth request, request is routed to server with IP address at index = 3 (192.168.0.4) and index is re-initialized to zero because it reaches the end of the vector. This is illustrated in Table 1.

Table 1. Vector Table

IP Address	Index number
192.168.0.1	0
192.168.0.2	1
192.168.0.3	2
192.168.0.4	3

The above cycle is repeated for the next four requests and so on. Meanwhile the size of the vector can change as more member servers can get connected or any of the connected servers may get disconnected [3-5].

2.2. Weighted Round Robin Technique

This scheme is suitable for heterogeneous servers (cluster-members) where processing capabilities of machines are already known and the size of jobs submitted from the clients are same because load is assigned to each member according to its capability. Weighted Round Robin Fashion Technique algorithm is given below:

The weight factors of all the member servers are calculated and stored in a vector. For each request, IP address of the server with highest weight factor is selected, weight factor is decremented by 1 and the same process is repeated for next requests. When the weight factors of all member servers become zero, the process of calculation of weight factors is repeated. This is done after every 10 requests because meanwhile there could be change in the number of connected servers.

IP addresses and strengths of member servers are stored in two different vectors in a manner that corresponding positions in both the vectors indicate the information about a server. A third vector is used to store the calculated weight factors of member servers. Weight factor determines the number of requests to be routed to the member server out of every 10 requests. Weight factor of certain member server is calculated using this formula: Weight factor = total strength of member servers/sum of strengths of all servers*10 (the figure is rounded off to a whole number). There are four member servers connected to the cluster manger and the IP

addresses and strengths of the servers are stored in two vectors. IP addresses of the connected servers are stored in a vector table shown in Table 2.

Table 2. Vector Table

IP Address	Index number
192.168.0.1	0
192.168.0.2	1
192.168.0.3	2
192.168.0.4	3

Strengths of the connected servers are stored in a vector shown below in Table 3.

Table 3. Strength Table

Strength	Index number
8	0
10	1
4	2
2	3

Calculated weight factors of the connected servers are stored in a vector. This is calculated after every 10 requests. The results are shown in Table 4.

Table 4. Weight Table

Weight Factor	Index number
8 / 24 *10 = 3	0
10 / 24 *10 = 4	1
4 / 24 * 10 = 2	2
2 / 24 * 10 = 1	3

According to this weight factor table, the request has value '3' is routed to server 192.168.0.1, value '4' to 192.168.0.2, value '2' to 192.168.0.3 and the value '1' is routed to 192.168.0.4.

For the first request, IP address of server with maximum weight factor (index = 1, IP = 192.168.0.2) is chosen and weight factor of server is decremented. The contents of the weight factor vector after first request is shown in Table 5.

Table 5. Weight Table

Weight Factor	Index number
3	0
3	1
2	2
1	3

For the second request, IP address of the server with maximum weight factor (at index = 0, IP = 192.168.0.1) is chosen and weight factor of the server is decremented. Table 6 shows the weight table after second request.

Table 6. Weight Table

Weight Factor	Index number
2	0
3	1
2	2
1	3

Similarly, all the requests are chosen respectively, the final contents of the weight table are shown in Table 7.

Table 7. Weight Table

Weight Factor	Index number
0	0
0	1
0	2
0	3

Now weight factors are recalculated and the same cycle discussed able is repeated [3-5].

2.3. EquiLoad Technique

Load balancing using EquiLoad ensures that each member server can take equal load from the LoadBalancer. If using EquiLoad then each member server can get equal size of request from the main server.

Assuming that the number of back-end servers is, EquiLoad policy requires partitioning the possible request sizes into N intervals, [(s0 0; s1), (s1; s2),……, (sN1; sN1)], so that server 1 is responsible for satisfying request of size between si1 and si. In practice the size corresponding to an incoming request might not be available to the front-end dispatcher but this problem can be solved using a two-stage allocation policy. First the dispatcher assigns each incoming request very quickly to one of N back-end servers using simple policy such as Round- Robin which is even easier to implement. When server 1 receives a request from dispatcher it looks up to size s and if si1 s<si it will put the request in its queue otherwise it will reallocates it to the server j satisfying sj1 s<sj (any server i receives from another server is instead en-queued immediately since it is guaranteed to be in the correct size range). Letting the back-end servers reallocate requests among themselves is very sensible, since the size of information is certainly available to them.

Assume there are four member servers connected to the cluster manger and IP addresses of the servers are stored in a vector. This is explained by using Round Robin fashion of load assignment and illustrated in Table 8.

Table 8. Vector Table

IP Address	Index number
192.168.0.1	0
192.168.0.2	1
192.168.0.3	2
192.168.0.4	3

There are ten requests arrive from client. Now main server will use EquiLoad policy and then assign a equal number of request to each member server. First request will be routed to server with IP address at index=0 (192.168.0.1) and IP address is returned to the client. Second request is routed to sever with IP address at index=1 (192.168.0.2) and IP address is return to the client. Third request will be routed to server with IP address at index=2 (192.168.0.3). Now this process will continue and then each member server can find equal request size from main server [3-5].

2.4. SITA-E Technique

Size Interval Task Assignment with Equal Load is called SITA-E. The SITA-E algorithm is based on the observation: if task size variability were very small (c2 < 1) FCFS would outperform PS for a single queue. Therefore, SITA-E's goal is to reduce the variability of tasks arriving at each host. It achieves this by partitioning tasks among hosts, according to their sizes. Surprisingly this method is even able to

compensate for high variability of a heavy-tailed distribution. SITA-E has additional advantage that it has a static policy and therefore has a simple implementation. In this policy when a request arrives its size will be determined and only specific member server is assigned to the client. SITA-E relies on the assumption that the distribution of the size of incoming requests is known and further this distribution has mean M. In SITA-E each host only accepts tasks whose size falls within a specified size interval where this size range is chosen such that each host receives equal work in expectation. Specially let F(x) = PfX denote the cumulative distribution function of request sizes and F(x) = PfX = xg the corresponding density function. Let 'k' denote the smallest possible request size, 'p' denote the largest possible request size and 'h' be the number of hosts. Assume there are four member servers connected to the cluster manager and the IP addresses of the servers are stored in a vector. This is explained by using Round Robin fashion of load assignment, illustrated in Table 9.

Table 9. Vector Table

IP Address	Index number
192.168.0.1	0
192.168.0.2	1
192.168.0.3	2
192.168.0.4	3

Strengths of the connected servers are stored in a vector shown in Table 10.

Table 10. Strength Table

Strength	Index number
8	0
10	1
4	2
2	3

Calculated weight factors of the connected servers are stored in vector shown in Table 11. This is calculated after every 10 requests.

Table 11. Weight Table

Weight Factor	Index number
8 / 24 *10 = 3	0
10 / 24 *10 = 4	1
4 / 24 * 10 = 2	2
2 / 24 * 10 = 1	3

In this policy each member server is assigned a strength so if the job size is less then 1024 kb it will be routed to index =3 and IP=192.168.0.4.If the higher size job which is more then 2 or 3 Mb then higher strength server is assigned to the client request i.e. index=1 and IP=192.168.0.2. Each server can utilize its power on the large processing request [3-5].

3. METHODOLOGY

The Islamia University of Bahawalpur (IUB) has almost ten thousand users that utilize the facilities of networking and Internet for their research and academic purposes. As more users are switching over to Internet and networking day by day, the problem of congestion and slow user-request processing speeds due to heavy loads of users and traffic occurs in the network. So, there is indeed a need of some type of technique that will make the processing of server faster and easier for the users. The load management in client-server system is capable of transferring the load of request from main server to the member servers or clients [22]. A cluster approach is used, a cluster server system consist of four independent servers that works together [6-7]. Figure 3 depicts a clustered-based distributed system of the IUB.

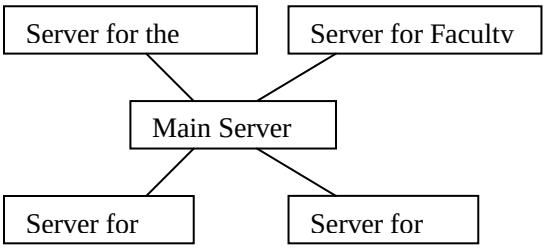

Figure 3. A Clustered-based Distributed System of the IUB

The servers have been designed with the multithreading capabilities which will process the data transfer request from the multiple clients simultaneously. Socket communication has been used for the communication between client and server. The Load Management System is equipped with the techniques or algorithms that can be used according to the requirements [6-7]. The implemented architecture with multithreading capabilities of the server is shown in Figure 4.

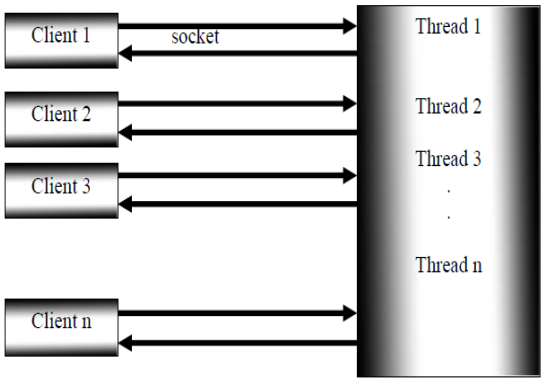

Figure 4. A Server running 'n' Threads

Initially, server opens Server Socket and dedicates a thread to listen to requests. Then the client initiates request for connection to server and consequently, server opens dedicated socket for the client. The Communication Handler object starts a separate dedicated thread at server side for the client and client starts its thread to communicate with the thread at server side. The system has three main components, namely, Cluster Manager Module: running on a machine with powerful hardware and is responsible for managing the activities of member servers in the cluster like redirection of load to the appropriate member server. Member Server Module: software component the instance running on multiple machines and this component actually serves the data transfer request after being redirected from cluster manager and Client Module: software component which requests the data transfer [8-11].

Figure 5 shows the sequence diagram an interaction between Server Helper and Server.

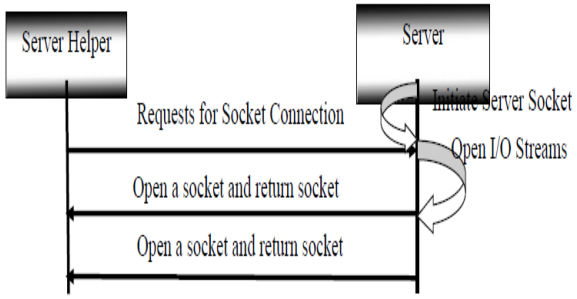

Figure 5. A Sequence diagram of Server Helper and Server

In the first step, a Server Helper initiates a request for the socket connection and after establishing the connection a socket is returned i.e. server opens Server Socket and dedicates a thread to listen the requests. Then the client initiates request for connection to server and consequently, server opens dedicated socket for the client. In this way the server tickles the requests of the clients in a client-server based distributed system [8-11]. The activity diagram of the system is shown in Figure 6.

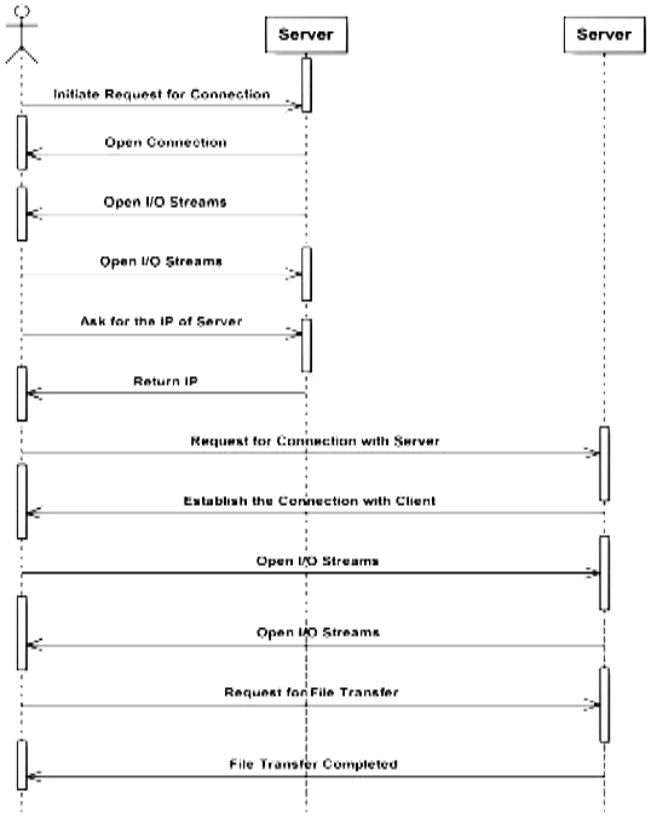

Figure 6. An Activity Diagram of the System

The explanation of Figure 6 is: a user requests for the connection with server, the server opens the connection through I/O Streams and finally the user asks for the IP of the server, the server also returns its IP. Similarly, the client requests for the connection with the server after establishing the connection through I/O Streams. Finally the client request for the File Transfer and after completing the request the server sends the completion message to the client [8-11].

4. RESULTS AND DISCUSSION

We test the above discussed load balancing and management techniques on a cluster-based distributed system of The Islamia University of Bahawalpur, Pakistan. The techniques or algorithms are further divided into homogeneous and heterogeneous servers. The problem of congestion and slow user-request processing speeds can be solved by using a single large powerful server. This solution soon fails because of the enormous network traffic. The second solution is replicating the server information over many geographically separated independent servers called "mirrored-server" architecture. This will solve the problem of congestion but with a number of disadvantages including huge loss of network and computer resources and lack of control on the request distribution by server system. A promising and efficient approach is the development of distributed architecture where the user-requests can be routed among several server nodes. This solution of distributed servers being managed under a single system provides us with improved throughput performance. Thus a server system with ease of manageability, greater availability and scalability of the servers is attained. This will increase user satisfaction because the user get faster, more consistent response time, directing traffic to the least loaded and most responsive servers and also prevent servers from getting overloaded. Preferred users and mission critical application traffic can be given higher priority by the LoadBalancer. Servers and the network resources can be allocated for high priority users and applications with the bandwidth management feature. Mission critical application and user accessing these applications will get consistently good performance. The Round Robin technique is simple and very predictable. This approach uses the cyclic process. All the member servers using this technique are suitable for homogeneous environment with same processing capabilities. The weakness of this approach is there is some chance of convoying, i.e. when one server is significantly slower than the others. It also has no knowledge about load of the back-end server. The EquiLoad approach is the best for load balancing with its nature. It assigns the equal load to each member server so that all the member servers will have the equal size of jobs. It is the best for homogeneous environment. Weighted Round Robin technique is easy to implement and it has the awareness of the different capabilities of the servers. It is much suitable for the heterogeneous environment. The drawback of this technique is that the weight is manually assigned by the administrator and also the ungraceful degradation in case of overload. SITA-E technique is easy to implement. It has an additional advantage that it has a static policy and therefore has a simple implementation. In this policy when a request arrives its size will be determined and only specific member server is assigned to the client. It is much suitable for the heterogeneous environment. The drawback of this technique is that the weight is manually assigned by the administrator and also the ungraceful degradation in case of overload.

Table 12 summaries the comparison of all these techniques.

Table 12. A Comparison of Load Balancing & Management Techniques

Property/Type	Homogeneous		Heterogeneous	
	Round Robin	EquiLoad	Weighted Round Robin	SITE-A
Processing Capabilities	Cluster of servers have same processing capabilities	Cluster of servers have same processing capabilities	Cluster of servers have same processing capabilities	Cluster of servers have same processing capabilities
Size of Request	No check	Checks the size	No check	No check
Resource Consumption	Low	Low	High	High
Weight Allocation	No	No	Yes	Yes

The result derived from the above table is, 'EquiLoad Technique' is appropriate for homogeneous environment and 'SITE-A Technique' is suitable for heterogeneous situation. Furthermore, these four techniques of load balancing and management are tested on the job load, multiple of '10' and their processing time (in nanoseconds) shown in Table 13.

Table 13. No. of Jobs and Processing Time

No. of Jobs	Round Robin	EquiLoad	Weighted Round Robin	SITE-A
10	20.2	15.5	25.7	23.4
20	30.3	25.3	35.8	33.0
30	43.5	31.2	46.6	43.4
40	55.7	40.1	58.4	48.5

A graph is drawn between the number of jobs, a multiple batch of '10', and the processing time in 'ns', as shown in Figure 7.

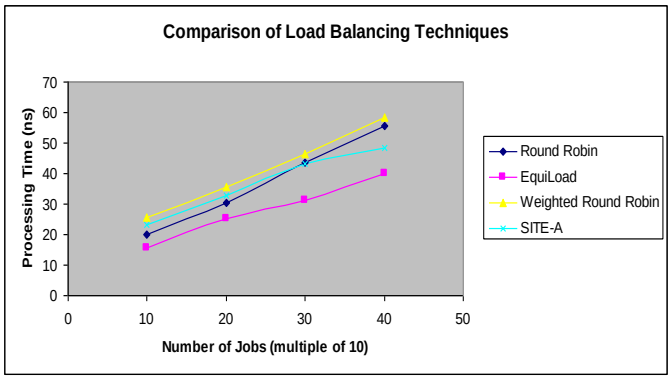

Figure 7. A Graph between Number of jobs and processing time (ns)

Figure 7 shows a graph between Number of Jobs (multiple of 10) and Processing Time (ns) for all load balancing techniques discussed in this paper. The graph shows that 'Weighted Round Robin' takes more processing time then the other techniques. In case of the 'SITE-A' its processing time is almost equal to 'Weighted Round Robin' for the first three jobs and for the last job it takes less time. In case of the 'Round Robin' its processing time is almost less than 'Weighted Round Robin' and 'SITE-A' balancing techniques for the first three jobs but it is equal to 'Weighted Round Robin' for the last job. The 'EquiLoad' load balancing technique takes less processing time for balancing and managing the load as compared to the other techniques discussed in this paper. The graph shows that the 'EquiLoad' technique performs better than the other techniques, therefore, in our case the 'EquiLoad' load balancing technique is selected.

5. CONCLUSION

In this paper we present the most commonly used load balancing and management techniques. There are advantages of each technique on the other hand limitations are also there. All techniques are deployed and found they are equally good for tackling the problem of congestion and overloading of the main server. In this scenario both homogeneous and heterogeneous environments are used. However, in homogeneous environment the EquiLoad provides good results. In heterogeneous environment the best technique is Weighted Round Robin where processing capabilities of machines are already known and the size of jobs submitted from the clients are the same because load is assigned to each member according to its capability. This will increase user satisfaction because the user get faster, more consistent response time, directing traffic to the least loaded and most responsive servers and also prevent servers from getting overloaded. Preferred users and mission critical application traffic can be given higher priority by the Load-Balancer. The servers and the network resources can be allocated for high priority users and applications with the bandwidth management feature. The mission critical application and user accessing these applications will get consistently good performance. We also draw a comparison of these techniques which reveals that the EduiLoad technique performs better than the other techniques, therefore, we propose the EquiLoad technique for the distributed system of the IUB. We conclude this paper that load balancing and management in a distributed system where traffic and load is heavy, is the essential.

Future Work

The number of users in the university network are increasing; the university is planning to implement a heterogeneous network. The campuses of the university are situated at different locations and the two campuses are far away from the main service provider of the network, this is an example of n-tier client-server distributed system, the use of intelligent mobile agents will be further benefited for the load management and balancing. The intelligent mobile agents are very commonly used in distributed network systems given that they are not cumbersome for the network traffic. Moreover, they overcome network latency, operate in heterogeneous environment and possess fault-tolerant behavior.

REFERENCES

[1] R. Buck-Emden, J. Galimow (30 Aug 1996). Client-server Technology SAP R/3 System (Hardcover) Publisher: Addison Wesley; Subsequent edition ISBN-13: 978-0201403503.

[2] Elbert B, Martyna B (1994). Client-server Computing Architecture, Applications and Distributed Systems Management**,** Publishers, Boston * London, ISBN 0-89006-691-4.

[3]Wikipedia (Encyclopedia), Network_Load_Balancing_Services,

http://en.wikipedia.org/wiki/Network_Load_Balancing _Services, November 20, 2008.

[4] Wikipedia (Encyclopedia),Weighted_round_robin, http://en.wikipedia.org/wiki/Weighted_round_robin, November 20, 2008.

[5] Neosmart.net (Website), http://neosmart.net/blog/2008/weighted-round-robin-dns-solutions/, November 22, 2008.

[6] ACM (Website), http://portal.acm.org/ October 15, 2008.

[7] Kennedy Clark, Kevin Hamilton (1999). CISCO LAN SWITCHING by CISCO PRESS. ISBN 1-57870-094-9.

[8] H Richard, Thayer H. (2004), Software Engineering Project Management. Second Edition, IEEE Computer Society Order Number BP08000, ISBN 9812-53-095-9.

[9] N Ashok, Kamthane N. (2004), Object-Oriented Programming, ISBN 81-7808-772-3.

[10] Pressman Roger S. (2008), Software Engineering: A Practitioner's Approach, 5th Edition, McGraw-Hill**,** ISBN 0073655783.

[11] Szyperski C. (2004), Component Software beyond Object-Oriented Programming. 2nd Edition, ISBN 81-297-0400-5.

[12] Khan, Dost Muhammad., Mohamudally, Nawaz., "From Mainframe to Cloud Computing: A Study of Programming Paradigms with the Evolution of Client-Server Architecture", Journal of Computing, Volume 4 Issue 12, 2011, pp.: 21-27.

[13] Khan, Dost Muhammad., Saher, Najia., et al, "The Human Resource Development (HRD) at the Higher Education and Research Institutions of Pakistan: The IUB Case Study", Journal of Computing, Volume 4 Issue 3, 2012, pp.: 120-124.

[14] Khan, Dost Muhammad., Saher, Najia., et al, "The Integration of Networking and Computerization towards e-Education and e-Learning at the Higher Education and Research Institutions of Pakistan", International Journal of Computer Science Issues (IJCSI), Volume 9, Issue 2, 2012 pp.: 546-551.

[15] Khan, Dost Muhammad., Mohamudally, Nawaz., "The Adaptability of Conventional Data Mining Algorithms through Intelligent Mobile Agents in Modern Distributed Systems", IJCSI International Journal of Computer Science Issues (IJCSI), Vol. 9, Issue 1, No 1, January 2012, pp.: 38-46.

[16] Ardhendu Mandal and Subhas Chandra Pal, "An Empirical Study and Analysis of the Dynamic Load Balancing Techniques Used in Parallel Computing Systems", Proceedings of ICCS-2010, 19-20 Nov, 2010.

[17] P. Beaulah Soundarabai*1, Sandhya Rani A.1, Ritesh Kumar Sahai1, Thriveni J.2, K.R. Venugopal2 and L.M. Patnaik, "COMPARATIVE STUDY ON LOAD BALANCING TECHNIQUES IN DISTRIBUTED SYSTEMS", International Journal of Information Technology and Knowledge Management, December 2012, Volume 6, No. 1, pp. 53-60.

[18] Raman a Kumar K., Mahesh V. Ghatage, "Load Balancing of Services with Server Initiated Connections", ICPWC, 2005.

[19] Branko Radojevic, Mario Žagar, "Analysis of Issues with Load Balancing Algorithms in Hosted Cloud Environments", Opatija, Croatia, MIPRO 2011, May 23-27, 2011.

[20] Hisao Kameda, El-Zoghdy Said Fathy, Inhwan Ryu, Jie Li, "A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe - Personal Computer Network Model", Proceedings of IEEE Conference on Dedsion and Control Sydney, Australia December, 2000.

[21] J. Li, C. Kim and Y. Zhang, "Optimal Load Balancing in Distributed Computer Systems", Springer, 1997.

[22] Saher, Najia., Khan, Dost Muhammad., et al, "The Prospects of ERP Systems on Quality of Education and Research in Higher Education and Research Institutions of Pakistan", JOURNAL OF COMPUTING, Volume 4 Issue 3, 2012, pp.: 115-119.