International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3475-3482

Database Replication Algorithm Performance in High Speed Networks Under Load Balancing

Rekh Nath Singh[1], Raghuraj Singh[2]

[1]*Research Scholar, A. P. J. Abdul Kalam Technical University , Lucknow, India.*

[2]*Director, K.N.I.T., Sultanpur, Sultanpur, India.*

Abstract

Database replication process maintains objects of database like tables, in a number of databases that build a system of distributed database. The database replication requirements are increasing day by day due to the more use of internet. To meet such requirements priorities of request can be used. In this work two classes of requests are considered i.e., Low Priority and High Priority. Low priority requests are not so important and they can be delayed or drooped over high priority requests. For example, downloading a song is low priority requests, while information send by defense applications are high priority requests. The request loss rate can be further reduced using load balancing conditions where some of the contending requests sent to some other nodes, and they reach their destination using some alternative paths. In view of above aspects performance evaluation of M-PDDRA (Modified Pre-bringing Based Dynamic Data Replication Algorithm) is done using computer simulation.

Keywords: Database repliation; database; priorities; load balancing etc.

INTRODUCTION

We could define the process of Replication of copying and keeping up the objects of database like tables, in a number of databases that build a system of distributed database [1]. We notice and make storing of these variations that are put into one site prior to sending and are being applied at all the remote positions. This process makes use of technique of distributed database in order to make sharing of data between numerous sites, however we can conclude that a replicated database and a distributed database are different. If we talk about a distributed database, we can find the data at numerous positions, still a specific table is available at just one position. We are going to mention few typical causes for making use of replication: This technique gives rapid, local access to shared data due to the fact that it maintains activity over a number of sites. A few users can have the authority to access one server however other may have the benefit of accessing various servers, hence diminishing the load at each server. In addition, the replication site with the least expense of access could be source from where users can access data. Generally, this is the geographically nearest site to them [2]. However, in the distributed database servers can be located anywhere across the globe. Nowadays, backbone network runs on fiber optic cable, with these cables servers are connected using O/E or E/O conversions as required.

The request arriving on these servers may have priorities, and if request cannot be served, then it will be dropped. To save the dropping of requests, buffering is performed at the servers. But if buffer overflows, then it is most likely that low priority requests will be dropped. To minimize this loss network load balancing scheme can be applied. This paper, investigate the performance of the algorithm, under low and high priority of requests along-with load balancing conditions at various server.

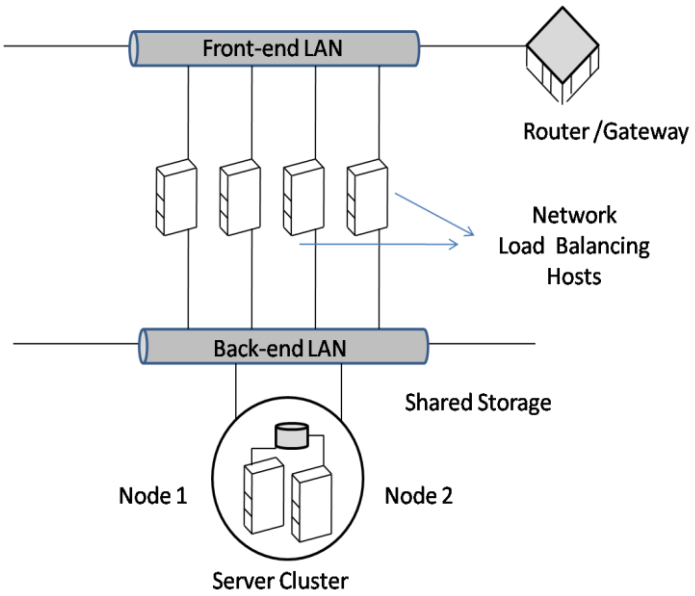

Figure 1: Schematic of a four-hosts cluster a single virtual server to handle network traffic.

We get great accessibility and versatility to enterprise-wide TCP/IP services by Network Load Balancing. These services may be streaming media, proxy, Web, Terminal Services, Virtual Private Networking (VPN) services. IP traffic is distributed to numerous copies of a TCP/IP service by Network Load Balancing like a Web server, all going on a host inside the cluster. Network Load Balancing straightforwardly segments the requests of the client among the hosts and gives the clients a chance to get to the cluster by making use of at least one or more "virtual" IP addresses. If we consider the client's perspective, the client find cluster to

be a solitary server that gives responses to these requests made by clients. With the enhancement in the enterprise traffic, network administrators could just connect an extra server into the cluster.

For instance, as shown in the Figure 1, the clustered hosts operator with one another with the purpose to serve traffic of the network from the Internet. A copy of an IP-based service like Internet Information Services 5.0 (IIS) is run by each server, and networking workload is distributed by Network Load Balancing among them. This pace up ordinary processing in the manner that client of Internet can observe speedier turnaround in regards to their requests. For included framework accessibility, the application at the back-end (suppose a database) may work on a two-node cluster going on Cluster service.

In comparison to the other software solutions, Network Load Balancing gives better results. For example, round robin DNS (RRDNS), makes the distribution of workload among a number of different servers however it is not able to give a mechanism for the availability of server. In the event of a server failure, RRDNS, not in the way like Network Load Balancing, will carry on to transfer it work till the failure is observed by a network administrator and eliminates the particular server from the DNS address list. This is in turn, brings in service interruption for clients. We have some benefits of Network Load Balancing over some other options for load balancing—on the basis of both hardware- and software—that present single failure points or execution hindrances by making use of a centralized dispatcher. Since Network Load Balancing got no restrictive hardware necessities, we can use any proper computer. This gives noteworthy cost reserve funds when contrasted with exclusive equipment load balancing solutions .

RELATED WORKS

A PDDRA (Pre-bringing Based Dynamic Data Replication Algorithm) is exhibited in [7]. The principle thought is to pre-get a few information utilizing the heuristic algorithm prior to the real replication begin to lessen latency. In earlier research, adjustments in PDDRA (M-PDDRA) are recommended to make the further reduction in latency. In yadav et. al. modified the PDDRA scheme and also establish connections among RS (regional servers) for sharing information, this allow local searching of the required information [8-11]. For more detail Please allude to [7] for further details. The fundamental purposes of the algorithm are outlined as below:

1. We consider the internet cloud in M-PDDRA technique as master node due to the fact that there is availability of data in the internet for the replication (Figure 2)

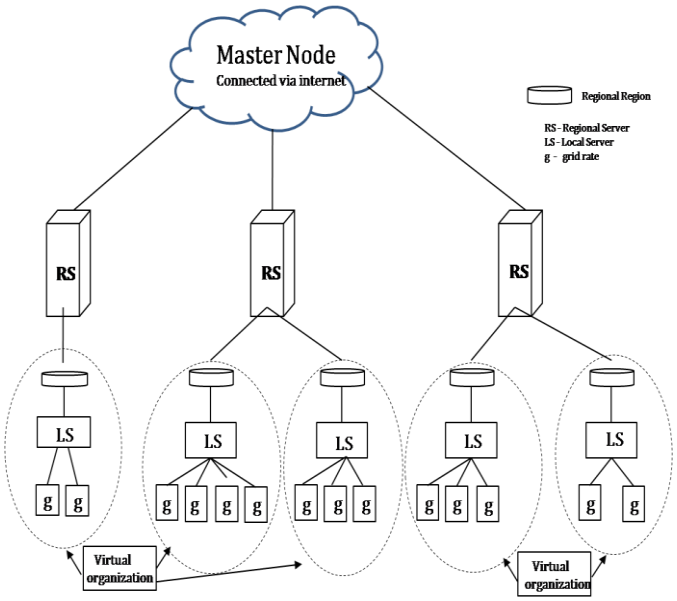

Figure 2: Schematic of the PDDRA scheme

2. In the case a node develops any replication request then it will get looked for in local network through edge node, and further a simultaneous request will be transmitted to the global network.

3. It is possible that we may not have the availability of data at any local node or we have a large waiting time is too large, due to the reason that simultaneous request is transmitted to both to a local node as well as a master node, in the event of master node access is in queue for suppose time t_q then we can make the local search for time $t_s < t_q$. The above discussed simultaneous requests to both global and local network will make the reduction in latency as compared to the initial request send to local network and after that to global network.

SIMULATION AND RESULTS

We carry out the simulation in MATLAB. The simulator is based on a random event generator and popularly termed as Monte Carlo simulation. In the simulation random traffic model is considered. This model is not complex; and even then it gives decent insight about the replication process. This model considers that the request can be originated from any of one the client with probability ρ and each generated request is equally likely to be served by any of the N servers with probability $1/N$. Therefore, probability that l requests arrive for a specific server in any time slot is [12]

$$\Pr(l) = \frac{N!}{l!(N-l)!}\left(\frac{\rho}{N}\right)^{l}\left(1-\frac{\rho}{N}\right)^{N-l} \quad \text{for } 0 \le l \le N, \quad (1)$$

Let Q_1, Q_2, ……., Q_q denote the ratio type-1,type-2,……, type-q requests to the total number of requests; $\sum_{i=1}^{q} Q_i = 1$.where q is the priority types (1 is the highest, q is the lowest).

Probability that n_1 type-1, n_2 type-2**, ...** n_q type-q requests arrive at the server in same time slot can be formulated as:

$$b_{n_1,.....,n_q} = \Pr(\sum_{K}(Q_1)^{n_1}...(Q_P)^{n_q} \times \frac{(\sum_{l} n_l)!}{\prod_{l}(n_l!)}) \qquad (2)$$

The requests will only be generated at the slot boundary only. Most of the systems have load in range of 0.4 to 0.8. Systems having load 1 will always be saturated and in general it is impractical. If requests is generated then it will randomly assigned a server from the available servers which can serve the request. However, if more than one server can serve the request than server selection is done randomly. However, if load balancing scheme is employed, then request will be assign a server with lesser request to serve. Again if same number of requests is left to be served then any one of the available server is randomly assigned. This paper adds one more parameter on the request generation, i.e., the priority. In this mechanism each generated request carries priority. We considered two type of priority; high and low. High priority requests are served first over low priority requests. If request is generated on the given load then high priority requests are serve first over low priority requests.

If arriving request can be served instantly, then it will be placed in the buffer and later on it will be retrieved from the buffer and served. The number of requests that can be buffered will depends on the buffer capacity of the server. If arriving request cannot be served at the server then it will be drooped and a negative acknowledgement (Server in not found/ please try again) is send back to the sender and sender again re-generate request after a few more time slots. To avoid loss of larger number of requests a hard load balancing scheme which restrict the number of request that can be send to particular server, while other leftover requests are send to other servers is employed. Requests are filled in the buffer using rules defined under:

A. Rules for filling Buffer

1. For each arriving request first buffer is checked, if buffer is empty, then request will be served instantaneously.
2. If buffer is not empty, then priorities of the buffered will be checked and one high priority request leaves the buffer in FIFO manner, and incoming request will be buffered using rule 5
3. If in the buffer only low priority requests are stored and arriving request also has low priority then it will be buffered using rule 2.
4. If in the buffer only low priority requests are stored and arriving request has high priority then it will be served.
5. The number of requests in the buffer should be lesser or equal to buffer capacity.
6. In above scheme low priority request may remain in the buffer for very long duration, to avoid this after a fix time slots a low priority request leave the buffer. This time slot is chosen randomly depending on buffer capacity.
7. To avoid overflow of buffer a hard load balancing scheme is employed at each server which restricts the number of requests that need to be served by particular server.

B. Results and Discussions

Request Loss probability: It could be defined as the volume of data that cannot flow via a network, or else we can define it as the fraction of the generated requests which are not served by any one of the server.

Network Load: We can define network load as the measure of data (traffic) is flowing through the network.

In the simulation two types of request requests low and high is considered. In figures 3 and 4 legends TRL, HPR and LPR are stand for total request loss, high priority request loss and low priority request loss respectively. The numbers of clients/servers (N) are considered to be 4.

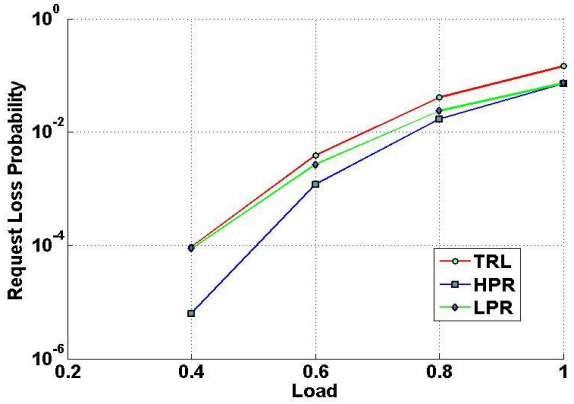

Figure 3: Request loss probability vs. load for Low priority 0.2 under buffer 4

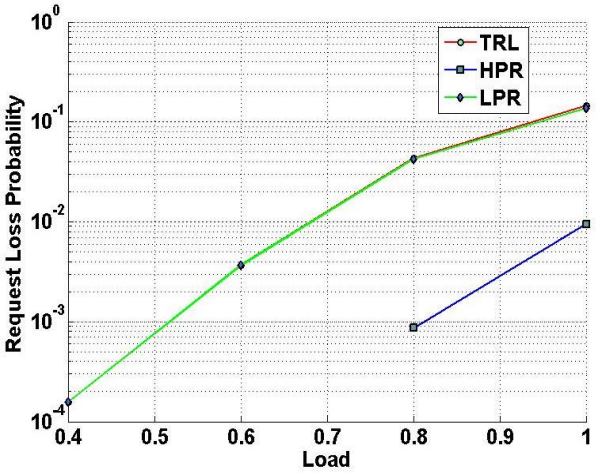

Figure 4: Request loss probability vs. load for Low priority 0.6 under buffer 4

Figure 3 shows the request loss probability vs. Load. In our work we have not shown throughput vs. load plot because low request loss rate. Moreover throughput is equal to 1- request loss probability. Therefore, both the graph can be used as they lead to same conclusions. In the request generated 4 clients are considered and servers are also considered to be 4. The performance low priority requests is shown with diamond marker, for high priority requests is shown by square marker while total request loss which include the loss of both high and low priority requests is shown with circle marker. Out of the total generated requests 20% are of low priority while left over 80% are high priority requests. At the load of 0.4, the request loss probability for high priority requests is 6×10^{-6}, for low priority requests it is 9×10^{-5} which is nearly equal to the total loss it is evident from the figure that the request loss rate of high priority requests is much less than that of low priority requests.

Figure 4 shows the request loss probability vs. Load. Out of the total generated requests 60% are of low priority while rest 40% are high priority requests. Considering the request loss probability at the load of 0.8, for HPR is 8.6×10^{-4}, for LPR and TRL is 4.27×10^{-2}. Here it is clear from the figure that till load 0.8, HPR loss is zero and only low priority requests are lost. Comparing figures 3 and 4, it is clear that low priority packets are lost first, and as their proportion in total requests increases, their loss also increases. However, the performance of HPR improves significantly.

Load Balancing

The load on a particular node can be reduced by deflecting the some of the arriving packets. The number of packets arriving for a particular output can be expressed as

$$E[l]=\sum_{l=0}^{N} l\,\frac{N!}{l!(N-l)!}\left(\frac{\rho}{N}\right)^{l}\left(1-\frac{\rho}{N}\right)^{N-l} . \qquad (3)$$

Now, '*g*' is the fraction of packets that are deflected that effective load is

$$\rho_e=\frac{\sum_{l=0}^{N}(1-g)l\,\frac{N!}{l!(N-l)!}\left(\frac{\rho}{N}\right)^{l}\left(1-\frac{\rho}{N}\right)^{N-l}}{N}. \qquad (4)$$

In core nodes once packets arrive then decision regarding deflection is performed, therefore above equation can be simplified to

$$\rho_e=(1-g)\rho \qquad (5)$$

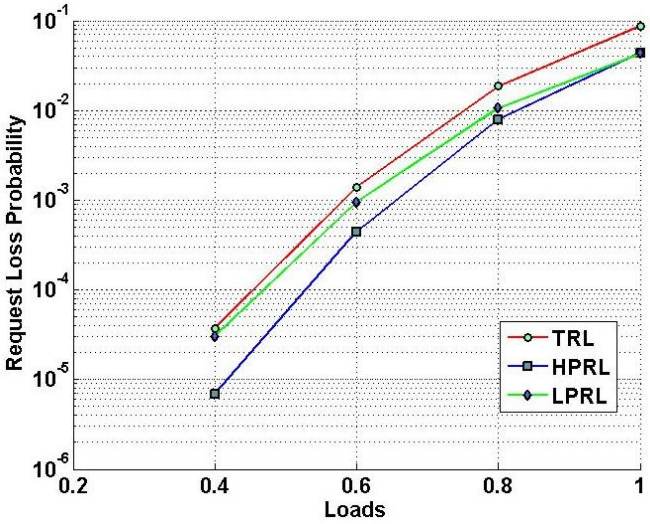

Figure 5: N = 4, B = 4, Low priority 0.2, load balancing factor 0.1

Figure 5 shows the request loss probability vs. Load. Out of the total generated requests 20% are of low priority while rest 80% are high priority requests and out of generated requests 10% requests follows some alternative path to reach to the server. Comparing the results at the load of 0.8, the request loss probability (HPRL) for high priority requests is 1.26×10^{-3}, for low priority (LPRL) requests it is 2.4×10^{-3} while the total loss (TRL) is 3.66×10^{-3}.

Figure 6 shows the request loss probability vs. Load. Out of the total generated requests 25% are of low priority while rest 75% are high priority requests and out of generated requests 25% requests are directed towards the output through some alternative path. Initially below 0.6 load, a significant difference is observed between high and low priority requests loss. Comparing the results at the load of 0.8, the request loss probability for high priority requests is 8×10^{-3}, for low priority requests it is 1.1×10^{-2} while the total loss is 1.86×10^{-2}.

Figure 7 shows the request loss probability vs. Load. Out of the total generated requests 50% are of low priority while rest 50% are high priority requests and out of generated requests 25% requests are directed towards the output through some alternative routes. Comparing the results at the load of 0.8, the request loss probability for high priority requests is 2.43×10^{-4}, for low priority requests it is 4.85×10^{-5}while the total loss is 9.7×10^{-5}. In the figure 3.7 at the load of 0.8, the request loss probability for high priority requests is 3.4×10^{-3} for low priority requests it is 3.66×10^{-3}. Thus it is evident from the figures load balancing reduces the request loss probability.

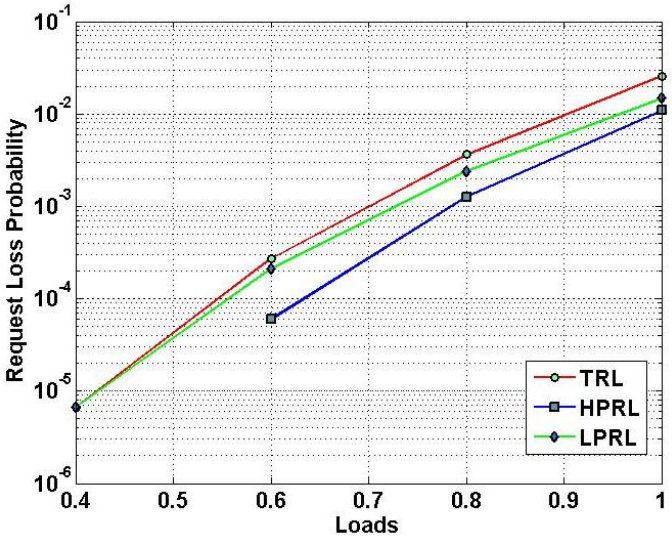

Figure 6: N = 4, B = 4, Low priority 0.25, load balancing factor 0.25

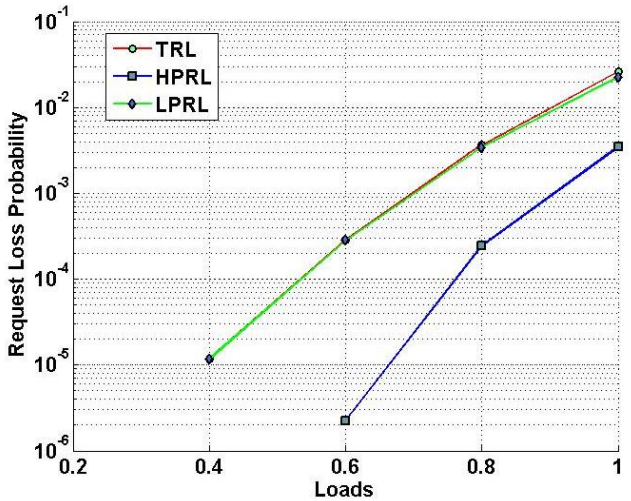

Figure 7: N = 4, B = 4, Low priority 0.25, load balancing factor 0.5

It is clear from Figs 5 to 7 that as the load balancing factor increases the request loss probability decreases. However, due to the high priority of some requests they get upper hand over other requests, therefore improvement for high priority requests is more.

As the number of low and high priorities requests affects the total loss, therefore for clear observation various types of requests losses are obtained for this Monte Carlo simulation is performed for 500000 iterations while keeping load to a fixed value of 0.6 for different proportion of high priority packets. The obtained results are shown in Figure 8. Here, for higher proportion of high priority requests, in total loss both high and low priority request contributes. While for moderate value of high priority requests in total loss major contribution is due to low priority requests. For lower proportion of high priority requests, in total loss is due to the low priority requests only.

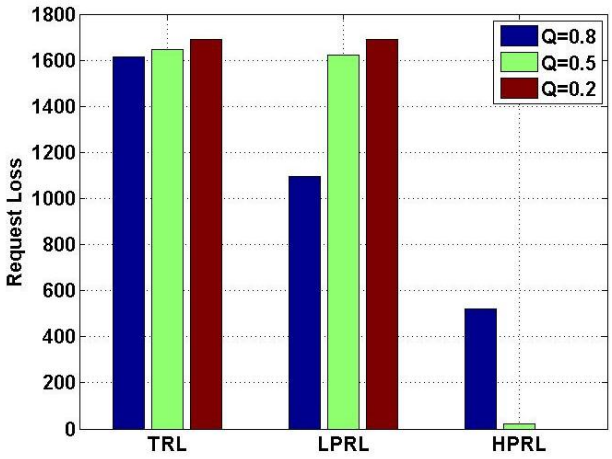

Figure 8: Bar graph for request loss for different proportion of high priority requests

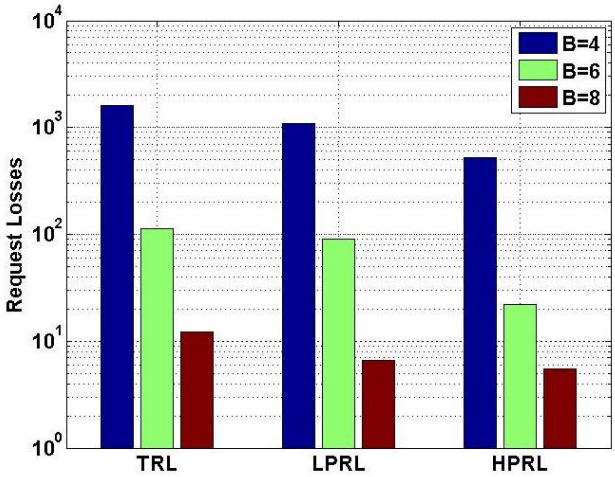

Figure 9: Bar graph for request loss for different buffering conditions

As discussed above, different proportions of high and low priorities can change the proportion of the loss of different types of requests, but over-all loss cannot be reduced. The over-all loss can be reduced by using more buffers as shown in Figure 9. In this figure proportion of low priority request is taken to be 0.2.By increasing the buffer form 4 to 6 and then from 6 to 8, the request loss reduces by a factor of more than 10. However, the quality of service is maintained, and loss of high priority requests is lowest and by increasing buffer and using load balancing can be brig down to a negligibly small value.

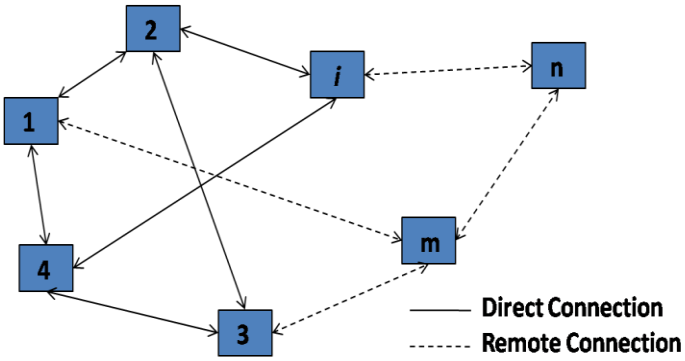

Figure 10: Schematic of *n* nodes network

Load Balancing

On a particular node '*i*' the arriving load is the sum of the partial load arriving from various links (Figure 10) and can be written as

$\rho = \sum_{j=1}^{m} \rho_j$, under the condition $0 \le \rho \le 1$ and $m < n$ where m is the number of nodes directly connected to node *i* and *n* is the total number of nodes in the network.

$$\rho_{eff} = \sum_{j=1}^{m} \rho_j - g_i \sum_{j=1}^{m} \rho_j \qquad (6)$$

where g_i denotes the fraction of load which is being deflected. In addition to this other *m* nodes which are directly connected to node *i* can also deflect their data to node *i*. Therefore effective load should be written as

$$\rho_{eff} = \sum_{j=1}^{m} \rho_j - g_i \sum_{j=1}^{m} \rho_j + \sum_{j=1}^{m} g_j \rho_j p$$

If links are chosen uniformly then we have,

$$\rho_{eff} = \sum_{j=1}^{m} \rho_j - g_i \sum_{j=1}^{m} \rho_j + \sum_{j=1}^{m} \frac{g_j \rho_j}{w_j}. \qquad (7)$$

Where, w_j denotes the number of input/outgoing links to a particular node '*j*'.

The load balancing is effective when

$$g_i \sum_{j=1}^{m} \rho_j - \sum_{j=1}^{m} \frac{g_j \rho_j}{w_j} > 0$$

Therefore,

$$g_i > \frac{\sum_{j=1}^{m} \frac{g_j \rho_j}{w_j}}{\sum_{j=1}^{m} \rho_j}. \qquad (6)$$

Eqn 6, provides the minimum value of fraction of load that needs to be deflected on node '*i*' for load balancing to be effective. While ρ_j denotes the load arriving form link '*j*' towards node '*i*' and it considered to be random between 0 and 1. The simulation results for two networks are detailed in Figure 10. In high speed optical networks, the numbers of core nodes are less than 20, while in current electronic networks number of core nodes can grow up-to to millions on nodes. In two networks 10 and 100 nodes are considered which can be considered as representation of optical and electronic networks. The results for 10 nodes network is shown in Figure 10, while for a general electronic networks of 100 nodes is shown in Figure 11. As shown in figure 10, the minimum value of load deflection factor is high with lesser number of nodes and as the number of nodes increases the value of load deflection factor reduces. For example in a 4 node network minimum value for load deflection is 0.125 while for 10 nodes it becomes 0.05. in case of large node network, load deflection factor reduces to zero if number of nodes are greater than 80. Therefore results presented in figures 3-9 are applicable for large node networks. However, with lesser number of nodes some corrections need to be done in packet loss results.

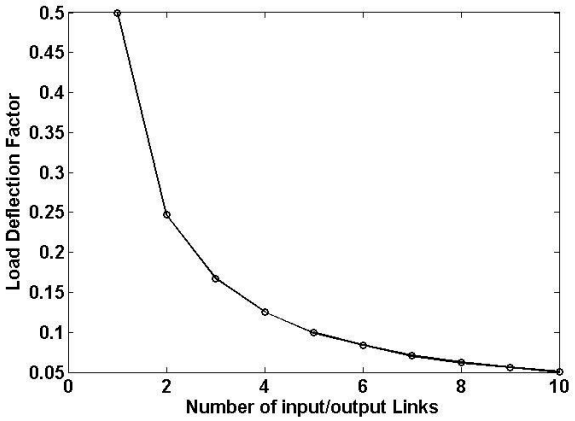

Figure 11: Load balancing factor vs. number of input/output links (10).

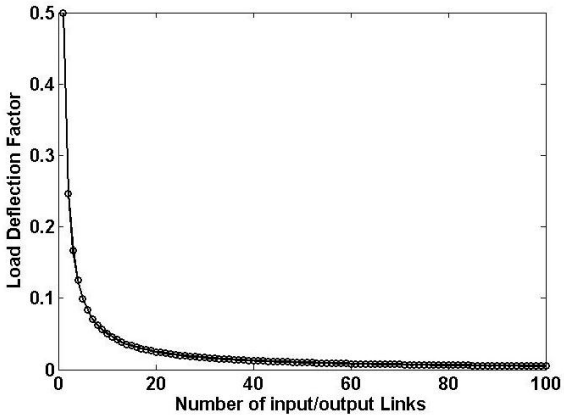

Figure 12: Load balancing factor vs. number of input/output links (100).

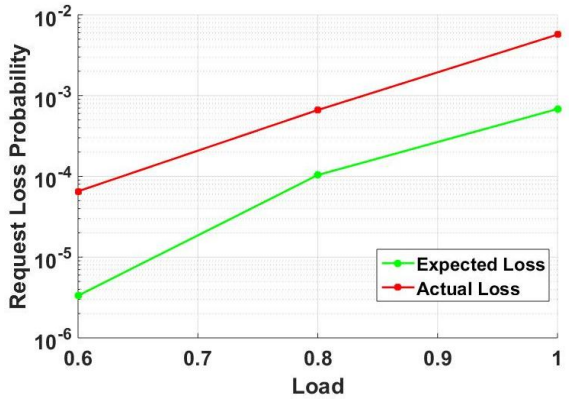

Figure 13: Request loss probability vs. load under load balancing factor (0.5) expected and actual loss.

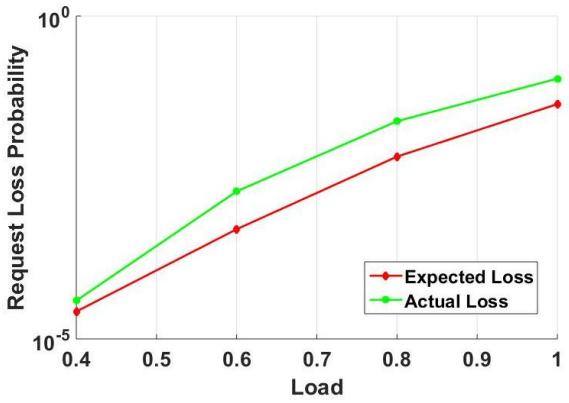

Figure 14: Request loss probability vs. load under load balancing factor (0.2) expected and actual loss.

In figure 13, request loss probability vs. load is plotted, under load balancing factor of 0.5, here expected loss is much lesser in comparison to actual loss, and obtained difference is significant. Here, expected loss is obtained using eqn. 6, while actual loss is obtained using eqn. 7. At the load of 0.6, the expected request loss probability is 3.3×10^{-6} while actual loss probability is 6.5×10^{-5}. In figure 14, request loss probability vs. load is plotted, under load balancing factor of 0.2, here again expected loss is much lesser in comparison to actual loss, and obtained difference is less significant. At the load of 0.6, the expected request loss probability is 5×10^{-4} while actual loss probability is 1.9×10^{-3}.

CONCLUSIONS

In this paper, performance evaluation of per-fetching replication algorithm is done. The performance evaluation is done under prioritized traffic while considering load balancing scheme. In the results it has been found that, using buffering, high priority requests can be served with nearly 100 percent efficiency. However, at higher loads (>0.8) for low priority requests throughput is slightly lesser. To keep loss of high and low priority requests to a very low level, we have also adopted a hard load balancing mechanism, which reduces the load on a particular server. Further, it is shown that in high speed networks, load balancing mechanism is affected by both outgoing and incoming traffics, and load balancing at particular node is also affected by load balancing of other nodes.

REFERENCES

[1] B. Kemme and G. Alonso, "Database replication: a tale of research across communities," *Proceedings of the International Conference on VLDB Endowment*. Switzerland), Vol. 3, No. 1, pp.5-12, 2010.

[2] A. Yair, D. Claudiu, M.A. Michal, S. Jonathan and T. Ciprian, "Practical wide-area database replication. Technical report, Johns Hopkins University,"2002.

[3] Y. Chen, D. Berry and P. Dantressangle, "Transaction based grid database replication," *Proceedings of UK e-Science. Edinburgh,* U.K. pp. 166-173, 2007.

[4] A. Correia, L. Rodrigues, N. Carvalho, R. Vilaça, R. Oliveira and S. Guedes, "GORDA: An open architecture for database replication" *Proceedings of Sixth International Symposium on Network Computing and Applications.* Boston, USA, pp. 287-290, 2007.

[5] S. Goel, R. Buyya, "Data replication strategies in wide area distributed systems," *Enterprise service computing: from concept to deployment,* pp. 211-241, 2006.

[6] A. Thomson, T. Diamond, S. C. Weng, K. Ren, P. Shao and D. J. Abadi, "Calvin: fast distributed transactions for partitioned database," *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Scottsdale, Arizona, USA, pp. 1-12, 2012.

[7] N. Saadat and A.M. Rahmani, "PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids," *Springer: Future Generation Computer Systems,* Vol.28, pp. 666-68, 2012.

[8] S.K. Yadav, G. Singh and D. S. Yadav, "Mathematical framework for a novel database replication algorithm," *International journal of ModernEducation & Computer Science*, Vol.5, No. 9, pp.1-10, 2013.

[9] S.K. Yadav, G. Singh and D. S. Yadav, "Analysis of database replication algorithm in local and global networks," *International journal of Computer Applications*, Vol.84, No. 6, pp.48-54, 2013.

[10] S.K. Yadav, G. Singh and D. S. Yadav, "Throughput and delay analysis of database replication algorithm," *International journal of ModernEducation & Computer Science*, Vol.5, No. 12, pp.47-53, 2013.

[11] S.K. Yadav, G. Singh and D. S. Yadav, "Analysis of a database replication algorithm under load sharing in networks,," Journal of engineering Science and

Technology (JESTEC),, Vol.11, No. 2, pp.193-211, 2016.

[12] R. J. Mishra, and A. Jain, "Performance of Data Replication Algorithm in Local and Global Networks under Different Buffering Conditions," *International journal of ModernEducation & Computer Science*, Vol.7, No. 9, pp.34-41, 2015.