ISSN: 0976-3104
SPECIAL ISSUE
Kumar et al.

THE IIOAB JOURNAL

ARTICLE OPEN ACCESS

FAULT AWARE LOAD BALANCING ALGORITHM FOR CONTENT DELIVERY NETWORK

Prakash kumar[1*], Krishna Gopal[1], JP Gupta[2]
[1]*Department of Computer Science Engineering & IT, JIIT, Noida, INDIA*
[2]*Hydrocarbons Education and Research Society, New Delhi, INDIA*

ABSTRACT

With the increasing use of data sharing, traffic over the internet has increased significantly. There is a need to effectively to manage the load over the servers and maintain the overall system performance with better Quality of Service (QoS). To maintain better QoS, Content Delivery Network (CDN) is used. CDNs offer services that improve network performance in terms of utilizing the bandwidth, improving accessibility, maintaining correctness through content replication and reducing load on servers. The limitation of existing CDN load balancing algorithms is that it considers servers and the systems as non faulty which increases the probability of request been allocated at faulty server. With the increase in number of requests, the failure probability increases due to long waiting queue which increases the network load and processing time. To overcome this problem, a fault aware load balancing algorithm for CDNs is proposed that improves the QoS and reliability of the system. In this paper, the effect of network failure on QoS and reliability of the system is studied in the presence of high request rate and network traffic. Performance of existing load balancing algorithm is investigated and compared in faulty environment. Moreover, the performance of the proposed algorithm is compared with reported techniques. The experimental results demonstrate that proposed algorithm provides better robustness and resilience to fault without affecting the QoS. Further, a dynamic fault model is proposed and implemented which takes care of changing failure probability with load and provided better result as compared to static fault models.

Received on: 18th-June-2015
Revised on: 20th-July-2015
Accepted on: 03rd– August-2015
Published on: 4th-Jan-2016

KEY WORDS

CDN; QoS; Reliability; Load balancing; Fault rate; Network load; System load; resiliency

***Corresponding author: Email:** kprakash91@yahoo.com, **Tel:** +91-9810292083

INTRODUCTION

Recently, the network traffic is exploding with rapid development of internet. Therefore to provide uninterrupted services to users and maintain QoS, CDN is required. CDN is a popular solution to balance the load over a distributed system which acts as a single system for users. It is among one of the best methods to cope up with the increasing demand and an effective solution to support the load of fast growing web applications by adopting a distributed network of servers.

CDN has been widely accepted as a method for circulating large amount of content to the users by making several redundant copies of content on multiple servers. CDN can solve even high congestion issues occurred due to unexpectedly high request rate from clients. There are many issues and parameters which restricts the performance of CDN such as issue of load balancing, cost, request traffic, response time. Many proposals [1-3] have been proposed to balance load based on Cost, Response time and load on server [4 - 6]. CDN has also been designed on the basis of Energy consumption and data transfer rate [3, 7, 9]. These proposals take into consideration energy consumed by the server and data transfer rate in the server. So the primary issue that persists in CDN is load balancing of request. In this paper, we have proposed a scalable and reliable architecture for CDN along with fault aware load balancing algorithm. Although many existing approaches address the issue of load balancing in CDN but they do not take into consideration failures at servers which increased with increase in load. The proposed algorithm takes into consideration both load and failure over a server and scalability of CDNs. To summarize, the proposed algorithm tried to solve the problem of scalability and load balancing in CDN and to overcome the drawbacks of existing techniques.

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

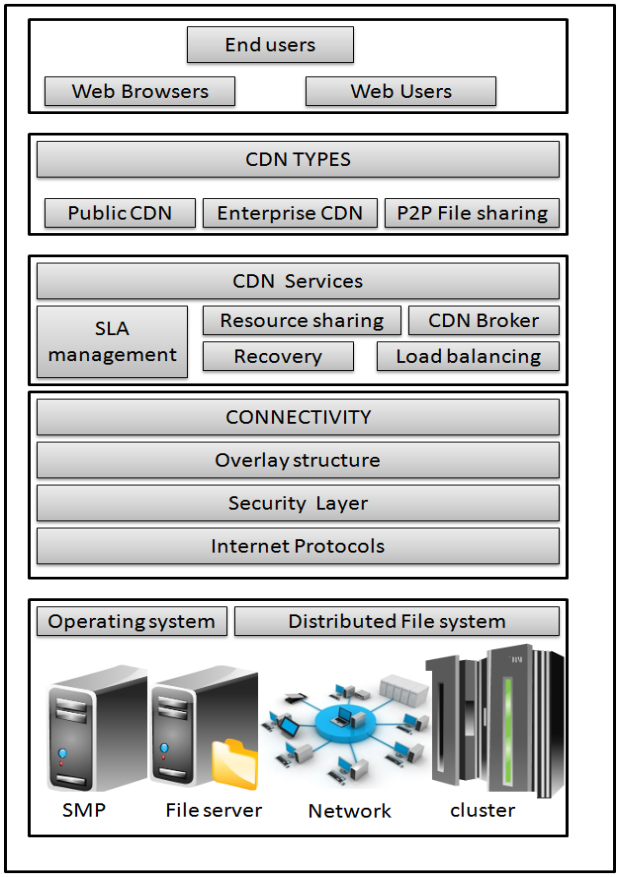


Fig: 1. Layered Architecture

Figure- 1 shows the layered architecture of CDN system with application layer at the top with client end, Then the type of CDN services offered. Third layer is management layer which is responsible for resource allocation security and load balancing and all other lower layers are standard network protocols and network connectivity. Servers are the last entity which remain connected using the networks.

This paper is divided into 5 sections: Introduction, Related Work, Proposed Model, Experimental result and Conclusion. Section II discusses the survey of proposed model for load balancing and fault in CDN and their drawbacks. In section III, We present proposed architecture and load balancing algorithm. Section IV presents Simulation environment and experiment results. Section V finally summarizes the findings and future prospective of the proposed architecture.

RELATED WORK

This section focus on the survey of existing load balancing algorithm for CDNs and cloud computing. These algorithm aims to reduce the load over the overall system and maintain the overall utilization of system. Cardellini et al.[16] proposed a survey on load balancing algorithms and classified them into two categories static and dynamic load balancing algorithms. This classification over load balancing algorithm helps in better understanding of difference between static and adaptive load balancing algorithms.

Dahlin et al.[17] proposed an least-loaded (LL) load balancing algorithm beast example for dynamic load balancing. In this requests are distributed to a server which is least loaded in term of queue length. Here requests are distributed to lead loaded server until it is completely saturated. To overcome this response time based algorithm was proposed to overcome the drawback of LL. Carter et al.[18] proposed a response time based load balancing algorithm which diverts the load to the server with fastest response time.

Manfredi et al.[19] proposed an load balancing taking care of load over the system and the capability of server to process the request. In this proposed algorithm each server is assumed to have a fixed queue size and if the queue

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

length increases load balancing is initiated and a least loaded server with empty queue is selected to balance the load.

Javadi et al. [20] discoursed load balancing which takes care of hardware fault based on Byzantine fault, i.e. an error in the system may lead to subsequent failure in the system. On the other hand, software failure, which covers request because of resource unavailability or high queue length also leads to request failure. In distributed system, failure can be correlated with a workload using spatial and temporal correlation between workload type and intensity of failure at different servers in short interval of time. Spatial correlation refers to multiple failures occurring on different servers in short interval of time. Temporal correlation means skewness of the failure spread over time. Where correlation between failure is the time between two consecutive failure .Let Ts(Fi), Ts(Fj) be the start time of failure i , j. Temporal correlation can be calculated as:

$$Li\ j = || Fi + Fj || = | Ts(Fi) - Ts(Fj) | \quad (1)$$

$$Ct\ (L) = 1 - \alpha \frac{L}{\theta} + \beta \left(\frac{L}{\theta}\right)^3 \quad (2)$$

Where Θ is an adjustable time scale parameter for determining the temporal correlation between two failure events, and α and β are positive constants where

$$\alpha = \beta + 1 \quad (3)$$

A hybrid approach based on random and LL was proposed by Mitzenmacher et al..[15] two random choice algorithm (2RC). In this 2 servers are randomly chosen and least loaded among those is selected. This approach is beneficial is there are a large number of servers and random choice algorithm help to provide an equal probability of a server been selected.

Papagianni en al. [1] proposed similar load balancing algorithms based on cost in which a hierarchical framework is proposed which is further evaluated towards an efficient and scalable content distribution over a multi provider networked cloud environment, where inter and intra cloud communication resources are simultaneously considered along with traditional cloud computing resources. The performance of this proposed framework is accessed via simulation and modeling, while appropriate metrics are defined to associate with and reflecting the interests of different key players.

Maki en al. [3] proposed a periodic combined-content distribution mechanism to increase the gain in traffic localization. This Proposed mechanism automatically optimizes the distribution period by using how long we can expect the previous downloaded combined-content to localize traffic. A pictorial view of this model in shown in figure 2 below.

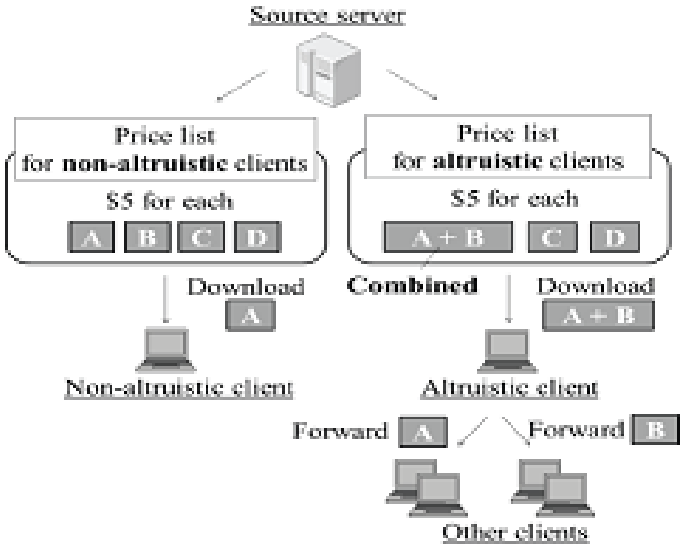


Fig: 2. Cost based Distribution

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

www.iioab.org

THE IIOAB JOURNAL

Mathew en al. [7] provided a new dimension to CDN proposed an energy aware load balancing algorithm which is an optimal offline and online algorithm and can be used to extract energy savings both at the level of local load balancing at the data center and global load balancing across data centers.

Mahajan en al.[19] proposed and affinity based round robin load balancing algorithm for cloud to balance the requests in cloud infrastructure . This algorithm takes into consideration the load over the data center and then if the datacenter is over loaded initiates load balancing in round robin fashion. Kansal en al.[25] have discussed an survey on various existing load balancing algorithm in distributed environment. They have identified various parameters used to de

However all the existing techniques considers the system non faulty and do not take into consideration the reliability due to request failure and failure in system. So to overcome all these issues a fault and reliability aware distributed load balancing is proposed to overcome the request failures due to faults.

RELATED BACKGROUND

In this section we have proposed a fault model for CDN. Fluid models are being proposed for TCP flow control and in many MANET routing protocols [12-14]. CDN proposed framework consists of servers with independent queues of self-determining queue length and service rate. Our proposal uses a fluid model for dynamic queue and real time behavior of the system. We have assumed a CDN with 'n' number of servers with service queue and high rate of request traffic over the system, which cannot be fulfilled by single system and the system remains in critical situation. In such situation load balancing plays an important role to resolve the critical condition of servers by diverting the request to the server with lower request rate and empty queue which can full fill the requests. For a server with a fluid flow model we need to introduce few notations.

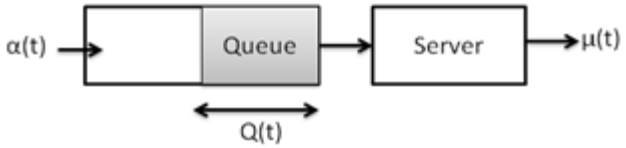


Fig: 3. Fluid model

Qi(t) : Queue length of server 'i' at time t.
αi (t) :Request arrival rate of server 'i'
μi (t) : Service rate of the server 'i'

In the fluid model can defined by

$$\frac{dQ_i(t)}{dt} = Q'_i(t) = \alpha_i(t) - \mu_i(t) \quad (4)$$

For i=1 2….N

Where Q' (t) are the extra requests to be fulfilled .In a fluid model with an increase in request arrival rate αi (t) over a node or server queue size increases if service rate is less than the request arrival rate and server cannot handle the requests. As a result of which requests in the queue have to wait for a long time. Also with increase in queue length, load over the servers increases resulting in an increase in response time and computation time. On the other hand, all this result in higher probability of request failure. To provide best QoS (Quality of service) we need to maintain the relation between the average incoming rate and average service rate.

$$\bar{\alpha} \leq \bar{\mu} \quad (5)$$

Average incoming rate = $\bar{\alpha}$

Average service rate = $\bar{\mu}$

www.iioab.webs.com

THE IIOAB JOURNAL
ISSN: 0976-3104

Next parameter we need to find is the Probability of failure of request in server over a time t. We have assumed that request rate αi (t) is distributed randomly over the time which follows a Poisson distribution.

$$p(x,\lambda) = \begin{cases} \frac{\Lambda^x e^{-\Lambda}}{!\,x} & ,x \geq 0 \\ 0 & ,x < 0 \end{cases} \quad (6)$$

Where e is the natural logarithm and k is the possible number of occurrences of the event (positive integer values). X (The number of events in a given interval), λ (mean number of events per interval) is a positive number representing the expected number of occurrences within a specified interval. For example, if 6 requests arrive every 10 minutes, then for 1 hour λ will be 36. The Poisson distribution models the occurrence of an event without knowing the total number of possible occurrences. We have used the Poisson distribution for calculating fault rates and reliability of a system. So probability distribution for failure in a system can be given by

$$f(x,\lambda) = \frac{\Lambda^x e^{-\Lambda}}{!\,x} \qquad ,x \geq 0 \quad (7)$$

Equation 7 shows the failure probability distribution over a time t, x (number of failures), ʎ (failure rate) .Where F (T) is defined as the probability of failure over the time t. To define failure in a system over a time t and t+ Δ T is given as:

$$F(t \leq T \leq t + \Delta T \mid T > t) = \frac{\exp(-\lambda t) - \exp(\lambda(t+\Delta T))}{\exp(-\lambda t)} = 1 - \exp(-\lambda \Delta t) \quad (8)$$

$$F(t) = 1 - \exp(-\lambda t) \quad ,\text{for interval } [0,t]$$

The Reliability of a system can be defined in term of many parameters such as durability, failure and QoS over a time t .In general probability, reliability can be defined as the probability of an item to perform a required function under stated conditions for a specified period of time. In terms of failure reliability of a system can be defined as resistance to failure of an item over time. The Probability that a system is reliable over a time t can be given as:

$$R(t) = 1 - F(t) \quad (9)$$

$$= 1 - 1 - \exp(-\lambda t)$$
$$= \exp(-\lambda t)$$

For interval [0, t]

And for time interval [t, t+ Δ T] reliability R (t) is given as

$$R(t) = \exp(-\lambda t) \quad (10)$$

PROPOSED ALGORITHM

Proposed load balancing is an improved algorithm over existing algorithm discoursed in section 2. Proposed algorithm takes into fault over a server over a period of time t as discoursed in section 3. Therefore the factors on which our algorithm is based on are Network Load, Fault Rate, Queue length, and Response Time. These parameters can be defined as:

System Load: The percentage of serve request queue filled.
Fault rate: Number of faults over a period of time.
Queue Size: Maximum size of request queue length a server can maintain and fulfill.
Response time: Time taken to start fulfilling a request.
Network load: Total bandwidth of server out of total under utilization.

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

Since each server is assumed to have its service rate, request rate, response time, queue size and failure rate which changes dynamically with time. For load balancing we need to find overloaded server or as we say the hot spots. A Server is said to be overloaded if:

QSize (i): queue size of server i;
Qi(t) : queue length of server i at time 't' from equation 4
ΔQi : Extra Queue to be balanced

Queue Size (i) < Qi(t) (11)

The Proposed algorithm is divided into three phases:
a) Initialization
b) Load balancing
c) Updating

a. Initialization

In this phase fitness value for a server is initialized with default values of all the parameters discoursed. All the parameters are checked and updated periodically. Initially fault rate and network load are zero, where as Queue size and response time are based on the server properties. Based on these values, fitness values are calculated. When a new server is introduced in CDN it is initialized with default values and fitness value is calculated and updated with equal intervals of time. Initial parameters are defined as:

Fault_Ini : Initial fault rate.
QSize (i) : Initial Queue length based on server.
Resp_Ini : Initial response time based on server.
N_load_Ini : Initial network load.
S_load_Ini :Initial system load.

a. Load balancing

In this phase when the original server queue is full and no more requests can be queued, in order to save request from waiting in queue of original server and fail due to deadline because they cannot be processed. So to overcome this replica of the data being requested is made on another server to balance the request load over the original server. To balance load we require finding a server which can fulfill the request with highest fitness value and same quality of service as promised by original server. Here we can classify the servers into two categories as a hot spot and a cold spot.

Hot spots are those servers which are overloaded with requests and have most of the MIPS and network bandwidth utilized and long request waiting queue. Cold spots are those servers which have low request rate and can accommodate more requests .In other words servers with low MIPS and network bandwidth under utilization i.e. load network and processing load. Load balancing is required to stop server becoming hotspot and find a cold spot to balance the request load. Whenever a server is found over loaded based on equation 11 we need to find the server which can fulfill extra request defined as :

$$\Delta Qi = Qi(t) - QSize(i) \quad (12)$$

ΔQi is the extra queue size to be balanced on server *i* where $i \in \{1,2,3 \ldots . n\}$. If ΔQi is positive we will call load balancing function. To balance the load we need to find a server with empty queue length and highest fitness value from a list of all such servers maintained. This list used by load balancing algorithm along with other parameters to find the best server over which request can be diverted.

Whenever load balancing is called we need to find a best fit server based on following parameters.

1) Fault rate: It is directly propositional to the load on the server that can be network load which leads to network failure and system load which leads to system failure which is due to high request rate, increasing the queue length. If the size of the queue is too large beyond the processing rate, the requests waiting time increases which lead to request failure. On other hand system load also increases the probability of system

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

failure in the form of hard disk and machine failure. All the above discoursed reason leads to degradation in QoS (quality of service) provided by the server.

$$\lambda(t) = F(N\_Load, S\_load) \tag{13}$$

$\lambda(t)$: fault over a time t.

Above equation defines that fault rate at a particular instance of time is functionally and directly proportional to system load and network load.

λ: fault rate

$$\lambda = \sum \text{total number of fault / per hour;} \tag{14}$$

2) Response time: This can be defined as the time taken to start processing a request, i.e. the difference between the time request was submitted and the time server started processing the request. It is directly propositional to system load. As the CPU utilization of server increases response time increases. So the server which needs to be selected should have least average response time and can complete the request in least time.
Resp : Response time

3) Queue length: Every server has a fixed request queue length, which can be fulfilled without request failure. So we need to select a server for load balancing which have a sufficient largest free queue size to accommodate new requests without failure.

To balance the load we need to take all the above parameters into consideration and calculate a fitness value for each server over which load can be balanced to provide better QoS and increase the reliability of the overall system by balancing the load and reducing failures.

The Fitness value for a server can be determined as:
Fval (s): Fitness value of server s

$$Fval(s) = \left(\alpha 1 * \frac{1}{\lambda}\right) + \left(\alpha 2 * \frac{1}{Resp}\right) + \left(\alpha 3 * \frac{1}{N\_load}\right) + \left(\alpha 4 * \frac{1}{S\_Load}\right) + free\,(Queue\_Length) \tag{15}$$

$$\alpha 1 + \alpha 2 + \alpha 3 + \alpha 4 = 1$$

$$\alpha 3 = \alpha 4$$
$$\alpha 1 > \alpha 2 > \alpha 3 > \alpha 4$$

$$server\_id = \max(fval1, fval2\ldots, fvaln); \tag{16}$$

Load balancing is divided into 3 steps 1) Find the list of all the servers which have empty queue grater then ΔQi is created. 2) From the created list find a server for load balancing with highest fitness value but at the same time the new server should have less or equal fault rate than the searching server to provide same or higher quality of service as promised by the original server, i.e. least fault rate, lease network load, least system load, and largest free queue length. 3) Transfer the set of extra requests to the selected server. This approach helps in maintaining skewness and increase reliability and decrease fault rate.

b. Fitness updating

This phase includes updating the value of current network load, system load, fault rate, queue length of server. This phase is repeated after an equal interval of time to get the updated current status of the servers. Initially, all the parameters are initialized with default values in which fault rate λ (t) is initially zero, network load in also

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

zero and system load is also taken as zero. The Queue length of a server in always initially zero because there is no request made to that server.

ʎ (t)_Initial = 0 \\ Initial fault rate

N_load_ini = 0 \\Initial network load.

S_load_ini = 0 \\Initial system load.

Q_len_ini = 0 \\Initial queue length

Res_Ini = Not zero \\Initial server response time

For calculating new fitness value we need to find changes in the parameters. Let Si be the server, ʎ (t)_new, N_load_new, S_load_new, Q_len_new , Res_new are new fault rate over a time 't' ,new network load, system load, queue length and response time correspondingly. Let new fitness value be fvalt_new (Si) of server i.

$$ʎ\,(t) = F\left(N_{Load_{new}}, S_{load_{new}}\right)$$

$$Fval_{new}\,(s) = \left(\alpha 1 * \frac{1}{ʎ_{new}}\right) + \left(\alpha 2 * \frac{1}{Resp_{new}}\right) + \left(\alpha 3 * \frac{1}{N\_Load_{new}}\right) + \left(\alpha 4 * \frac{1}{S\_Load_{new}}\right) + free(Queue\_Length) \quad (17)$$

server_id = min (fval1new, fval2new…., fvalnnew) (18)

$$\alpha 1 + \alpha 2 + \alpha 3 + \alpha 4 = 1$$

$$\alpha 3 = \alpha 4$$
$$\alpha 1 > \alpha 2 > \alpha 3 > \alpha 4$$

Whenever a fitness value is upgraded next request is always diverted to server with largest fitness value based on updated fitness values give in equation 17.

Load balancing Algorithm

```
1: Initialize servers
2: Start sending requests
3: push request in queue.
4: if (queue length > server queue length)
5:        s = find_server()        // find server with empty queue and highest fitness value
6:        if (( s !=  searching server ) & (fault rate < searching server))
7:           Migrate request  =>"s"
8:        else
9:         keep searching free server.
10: else
11: pop request from queue process it
```

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com

THE IIOAB JOURNAL
ISSN: 0976-3104

Update fitness value algorithm

1: Find updated values of parameters
2: Find network load
3: Find system load
4: Find fitness value using equation 14
5: update the new fitness value

In this section we have described the performance of proposed fault aware load balancing algorithm with round robin (RR), random (Rand), least loaded (LL), two random choice (2RC) and queue length based load balancing (QLBLB) algorithms. In this for simulation GridSim API [10] is used. GridSim API basically supports scheduling and load balancing in parallel and distributed environment. Load balancing, fault in server and server request queue feature of GridSim are used to simulate CDN.

Initially GridSim do not support failure in servers. In this implementation we have introduced fault aware scheduling in GridSim to study the performance of CDN in the fault aware environment. In simulation to create a network architecture as one shown in figure 1 we have used Brite file. Brite file helps to define network properties and the interconnection between the nodes which are the servers in our case. To compare performance based on the number of faults occurring by using each of the previous algorithms and proposed algorithm. We have considered 3 servers S1, S2, S3 each of them having their independent failure rate λ (t), request arrival rate, processing rate and queue length. Table 1 shows the specification of each server.

Table: 1. Servers Parameters

Server Name	Queue length	Fault rate	Service rate
Server1	20	0.143	7
Server2	50	0.125	8
Server3	50	0.5	2

Queue length defines that after the specific queue is full extra requests will be balanced using proposed algorithm, to save the request to fail due to large waiting time. **Table– 2** and **figure– 4** shows the number of requests failed when the algorithms are tested for 60,100, 200,500,600 and 700 requests count with all algorithms. Workload traces are achieved from load traces of DAS-2multi-cluster system obtained from the Parallel Workload Archive are used to generate requests [24].

Table: 2.Request Failure Count

Algorithm	Request count					
	60	100	200	500	600	700
Proposed	8	15	28	85	102	121
QLBLB	13	23	43	110	131	167
RAND	12	25	36	91	109	132
2RC	10	19	36	98	117	142
LL	10	23	39	102	124	151
RR	10	18	37	93	112	131

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

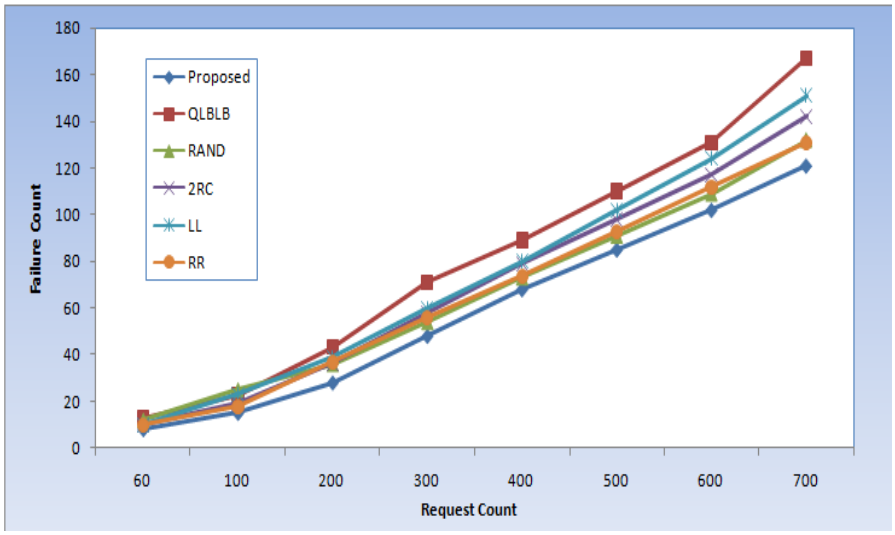


Fig: 4. Failure count of proposed algorithm against other algorithms

To compare performance based on probability of failure occurring by using each of the previous algorithms and proposed algorithm using scenario given in **Table– 1**.

Table: 3. Failure probability

Algorithm	Request count							
	60	100	200	300	400	500	600	700
Proposed	0.133	0.15	0.14	0.16	0.17	0.17	0.17	0.172857
QLBLB	0.216	0.23	0.23	0.233	0.224	0.22	0.218333	0.238571
RAND	0.75	0.25	0.18	0.18	0.182	0.18	0.181667	0.188571
2RC	0.166	0.19	0.18	0.199	0.197	0.2	0.195	0.202857
LL	0.166	0.23	0.195	0.2	0.2	0.204	0.206667	0.215714
RR	0.166	0.18	0.185	0.1866	0.185	0.186	0.186667	0.187143

Table– 3 and figure–5 show the probability of request failure when each of the algorithm is tested over 60,100,200,300,400,500,600 and 700 requests. **Table– 3** shows that with increase in request failure probability increases for QLBLB, 2RC and LL. On the other hand the probability of failure is stable for RR and Rand but grater then proposed algorithm. This shows that the proposed algorithm proves to have a lower failure count and failure probability as compared to other algorithms.

Table: 4. Reliability

Algorithm	Request count					
	60	100	200	500	600	700
Proposed	0.867	0.85	0.86	0.83	0.83	0.827
QLBLB	0.784	0.77	0.77	0.78	0.781	0.761
RAND	0.25	0.75	0.82	0.82	0.818	0.811
2RC	0.834	0.81	0.82	0.8	0.805	0.797
LL	0.834	0.77	0.805	0.796	0.793	0.784
RR	0.834	0.82	0.815	0.814	0.813	0.81

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

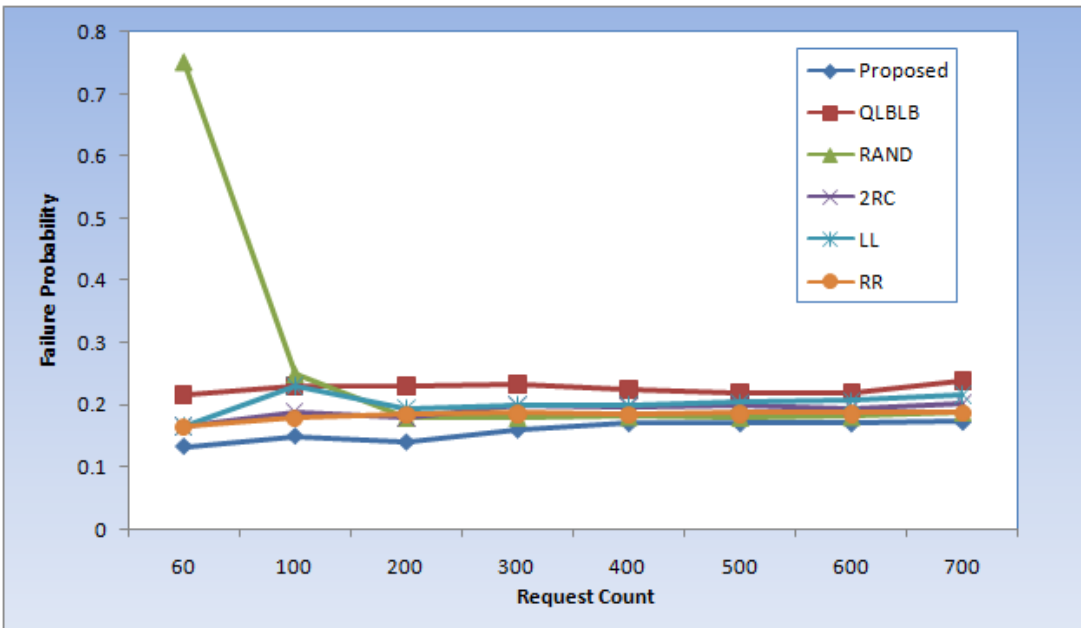


Fig: 5. Failure probability of proposed algorithm against other algorithms

The Proposed algorithm can also be compared with other algorithm based on one more parameter, i.e. reliability which is defined in equation 9. Reliability defines the algorithm to be more dependent and probability that the request will be completed. So, higher the reliability lowers the chance of request failure.

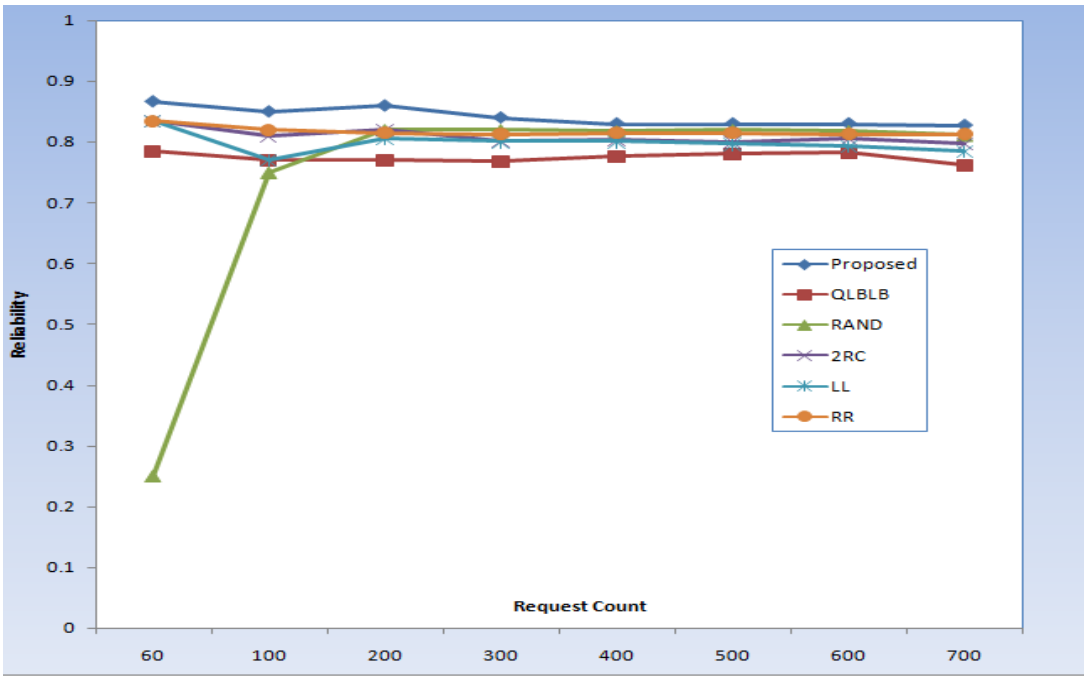


Fig: 6. Reliability of proposed algorithm against other algorithms

Table: 5.Completed request count

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

Algorithms	Request count							
	60	100	200	300	400	500	600	700
Proposed	52	85	172	252	332	415	498	579
QLBLB	47	77	157	229	311	390	469	533
RAND	48	75	164	246	327	409	491	568
2RC	50	81	164	242	321	402	483	558
LL	50	77	161	240	320	398	476	549
RR	50	82	163	244	326	407	488	569

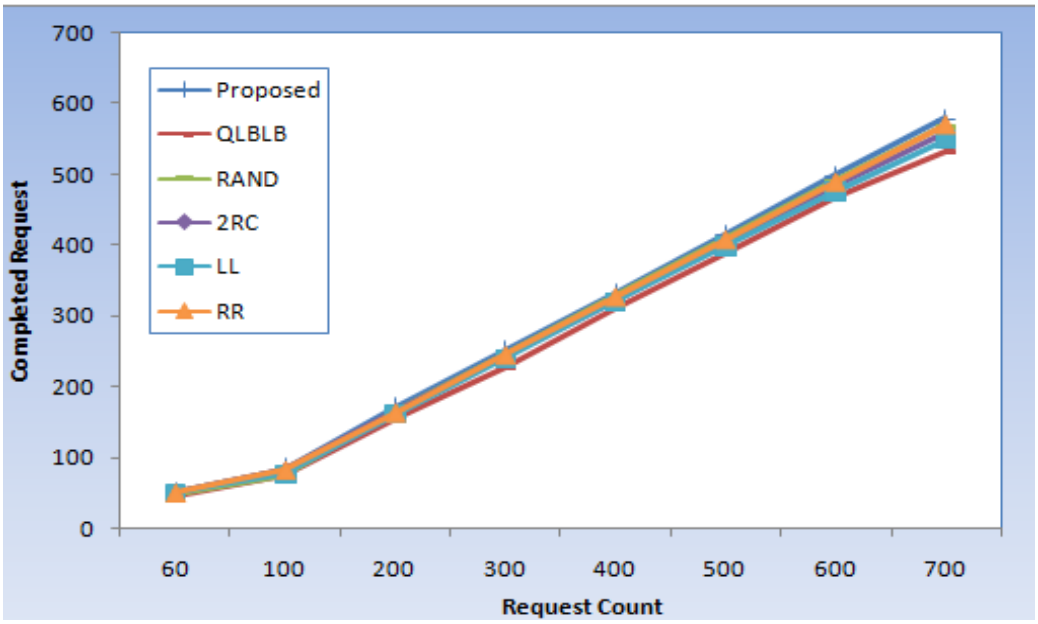


Fig: 7. Completed request count of proposed algorithm against other algorithms

Table– 4 and Figure– 6 shows the increase in reliability using proposed algorithm and improvement over other algorithm. Other advantages of the proposed algorithm over other algorithms that can be derived from **Table– 3 and Table–4** is that the algorithm which has higher reliability has shown to have high request failure, on the other hand proposed algorithm have a lower request failure and higher reliability.

Table– 5 and **Figure– 7** shows the improvement in Count of completed request using proposed algorithm over other proposed algorithm I faulty environment.

$$\text{Average Queue length} = \frac{\sum_{i=0}^{n} \text{Max\_len}_i}{n} \quad (20)$$

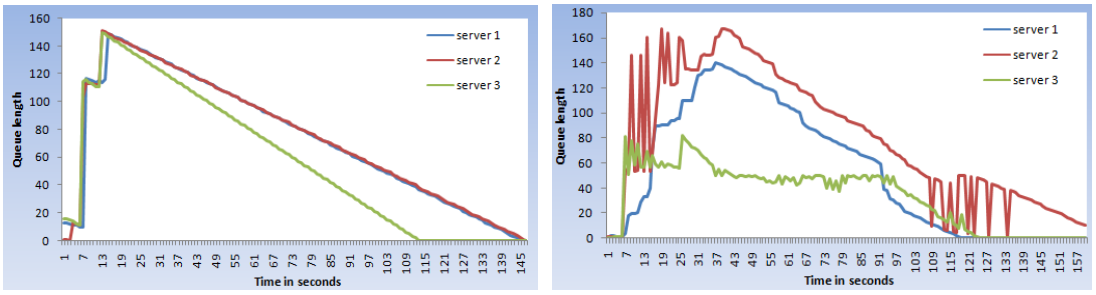
N= number of servers.

Max_length_i =Maximum queue length of server i

Proposed can also be compared based on the maximum queue length, because higher the queue size more the request waiting time. This increases the probability of request to fail over the period of time. So by comparing the maximum queue length achieved by each algorithm we can find the best algorithm.

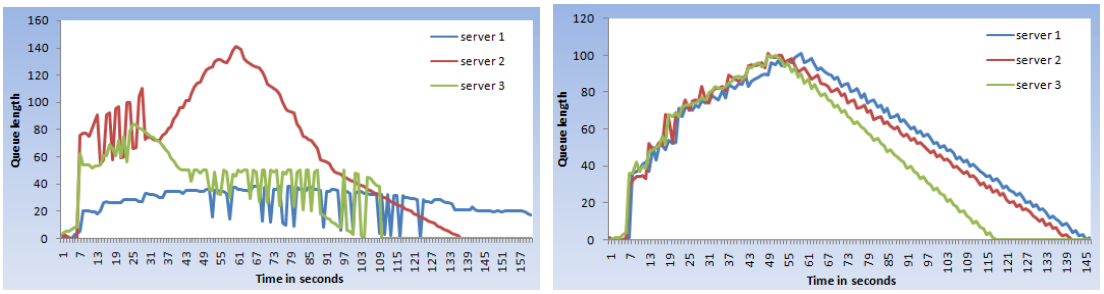
www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com

THE IIOAB JOURNAL
ISSN: 0976-3104



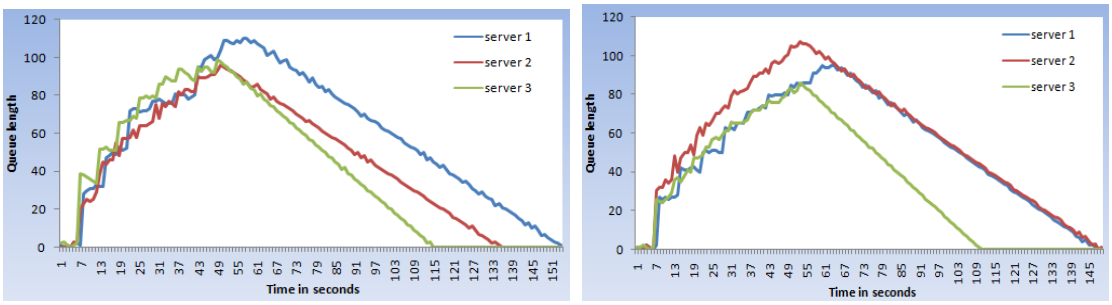
(a) 2RC (b) QLBLB

Fig: 8. Queue length of 2RC and QLBLB algorithm



(a) Fault **(b) LL**

Fig: 9. Queue length of Proposed Fault and LL algorithm



(a) RAND **(b) RR**

Fig: 10. Queue length of RABD and RR algorithm

Table: 6.Average Queue Length

Algorithms	RR	LL	RAND	2RC	QLBLB	
Average Queue length	96	101	101	150	129	87
Failure count	97	95	96	94	97	93

Figure– 8, 9 and 10 shows the behavior for queue length due to proposed and all other algorithms. An observation that comes out from above figures is that the algorithm which lower average queue length but has a higher fault rate. **Figure– 8 (a)** of 2RC algorithm has 150 average queue length and so on for other algorithm as shown in table 5 correspondingly. **Table– 6** clearly shows that an algorithm which has a lower average queue length, but has a higher failure count like RR algorithm, but proposed algorithm prove to have better performance in term of average queue length and failure count at the same time compare to other algorithm. Output for **table–**

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

www.iioab.org

THE IIOAB JOURNAL

6 is tested in the scenario shown in **table–1** with 3 servers and corresponding failure rate, service rate and queue length. The service rate for server 1is 7 requests can be processed at the same time, similarly 6 requests for server 2 and server 3.

Table: 7. Throughput

Algorithms	Request count							
	60	100	200	300	400	500	600	700
Proposed	86.66667	85	86	84	83	83	83	82.71429
QLBLB	78.33333	77	78.5	76.333333	77.75	78	78.16667	76.14286
RAND	80	75	82	82	81.75	81.8	81.83333	81.14286
2RC	83.33333	81	82	80.666667	80.25	80.4	80.5	79.71429
LL	83.33333	77	80.5	80	80	79.6	79.33333	78.42857
RR	83.33333	82	81.5	81.333333	81.5	81.4	81.33333	81.28571

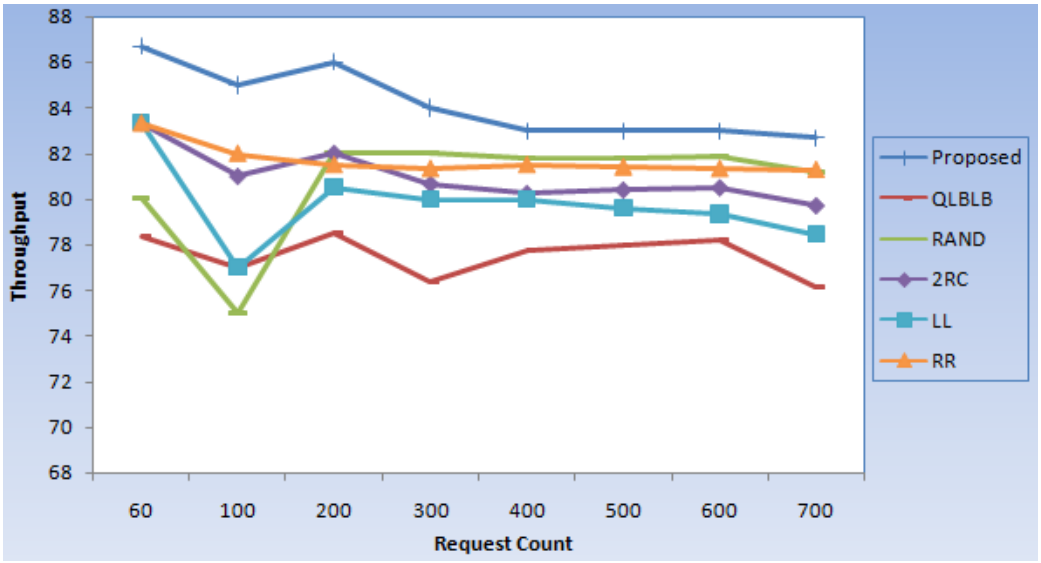


Fig: 11. Throughput Comparison of proposed algorithm against other algorithms

Table– 7 compares the throughput of proposed algorithm and other proposed algorithms for 60, 100, 200, 300, 500, 700 request over the servers. **Figure– 11** compares the throughput of proposed algorithm graphically and shows the improvement of proposed algorithm over other algorithms.

Taking into consideration all the performance parameters we can suggest that fault and reliability based proposed algorithm prove to have better performance and QoS over other algorithms.

CONCLUSION

In this paper, different types of Load balancing algorithm have been discussed with their drawbacks in CDN. To overcome the drawbacks, an efficient fault aware load balancing algorithm is proposed which performs better than other existing load balancing algorithms proposed for CDN in the fault aware environment. For future work, this algorithm may be compared with other proposals and study may be done for further improvements in the QoS.

www.iioab.webs.com

COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

CONFLICT OF INTEREST

Authors declare no conflict of interest.

ACKNOWLEDGEMENT

None

FINANCIAL DISCLOSURE

No financial support was received to carry out this project.

REFERENCES

[1] Akyildiz Papagianni, Chrysa, Aris Leivadeas, and Symeon Papavassiliou.[2013] A cloud-oriented content delivery network paradigm: modeling and assessment. Dependable and Secure Computing, *IEEE Transactions* 10(5): 287−300.

[2] Leong, Derek, Tracey Ho, and Rebecca Cathey.[2009] Optimal content delivery with network coding." In Information Sciences and Systems, CISS 2009. 43rd Annual Conference on 2009, IEEE, 414−419.

[3] Maki, Naoya, Ryoichi Shinkuma, Tatsuya Mori, Noriaki Kamiyama, and Ryoichi Kawahara. [2013] A periodic combined-content distribution mechanism in peer-assisted content delivery networks. In ITU Kaleidoscope: Building Sustainable Communities (K-2013), *IEEE* 2013 Proceedings of, 1−8.

[4] Jiang, Xueying, Shiyao Li, and Yang Yang.[2013] Research of load balance algorithm based on resource status for streaming media transmission network. In Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on, *IEEE*, 503−507.

[5] Ling Li, Ma Xiaozhen, and Huang Yulan.[2013] CDN cloud: A novel scheme for combining CDN and cloud computing. In Measurement, Information and Control (ICMIC), 2013 International Conference on, *IEEE*, 1:.687−690.

[6] Kim TaeYeon, and HoYoung Song.[2012] Hierarchical Load Balancing for Distributed Content Delivery Network. In *Advanced Communication Technology (ICACT)*, 2012 14th International Conference on, *IEEE*.810−813.

[7] Mathew Vimal, Ramesh K Sitaraman, Prashant Shenoy.[2012] Energy-aware load balancing in content delivery networks. In INFOCOM, 2012 Proceedings *IEEE*, 954−962.

[8] Maki Naoya, Takayuki Nishio, Ryoichi Shinkuma, et al.[2013] Expected traffic reduction by content-oriented incentive in peer-assisted content delivery networks. In Information Networking (ICOIN), 2013 International Conference on, *IEEE*, 450−455.

[9] Manfredi Sabato, Francesco Oliviero, and S Pietro Romano.[2012] Optimised balancing algorithm for content delivery networks. *IET communications* 6(7):733−739.

[10] Buyya Rajkumar, and Manzur Murshed. [2002] Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience* 14(13−15):1175−1220.

[11] Manfredi Sabato, Francesco Oliviero, and Simon Pietro Romano. [2013]A distributed control law for load balancing in content delivery networks. *IEEE/ACM Transactions on Networking (TON)* 21(1): 55−68.

[12] Misra Vishal, Wei-Bo Gong, and Don Towsley.[2000] Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In ACM SIGCOMM Computer Communication Review, 30(4):151−160.

[13] Hollot Christopher V, Vishal Misra, Donald Towsley, and Weibo Gong.[2002]Analysis and design of controllers for AQM routers supporting TCP flows. Automatic Control, *IEEE Transaction* 47(6): 945−959.

[14]]V.Misra, W Gong, W boGong, and DTowsley,[2000] Fluid-based anal-ysis of a network of AQ Mrouter ssupporting TCP flows with an applicationtored, *Proc.ACM SIG COMM,*:151–160.

[15] Mitzenmacher, Michael.[2001] The power of two choices in randomized load balancing. Parallel and Distributed Systems, *IEEE* Transactions 12(10): 1094−1104.

[16] Cardellini Valeria, Emiliano Casalicchio, Michele Colajanni, and Philip S Yu. [2002]The state of the art in locally distributed Web-server systems. *ACM Computing Surveys* (CSUR) 34(2): 263−311.

[17] Dahlin, Michael.[2000] Interpreting stale load information. Parallel and Distributed Systems, *IEEE Transactions* ,11(10):1033−1047.

[18] Carter Robert L, Mark E Crovella.[1997] Server selection using dynamic path characterization in wide-area networks. In INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., *Proceedings IEEE*, 3: 1014−1021.

[19] Mahajan Komal, Ansuyia Makroo, and Deepak Dahiya. [2013] Round Robin with server affinity: a VM load balancing algorithm for cloud based infrastructure. *Journal of information processing* system,s 9(3): 379−394.

[20] Javadi, Bahman, Jemal Abawajy, and Rajkumar Buyya.[2012] Failure-aware resource provisioning for hybrid Cloud infrastructure.*Journal of parallel and distributed computing*, 72(10): 1318−1331.

[21] Fu, Song, and Cheng-Zhong Xu. [2010] Quantifying event correlations for proactive failure management in networked computing systems. *Journal of parallel and distributed computing*, 70(11):1100−1109.

[22] Gallet Matthieu, Nezih Yigitbasi, Bahman Javadi, Derrick Kondo, Alexandru Iosup, and Dick Epema.[2010] A model for space-correlated failures in large-scale distributed systems. In Euro-Par 2010-Parallel Processing, 88−100. *Springer Berlin Heidelberg*

[23] Yigitbasi Nezih, Matthieu Gallet, Derrick Kondo, Alexandru Iosup, Dick Epema.[2010] Analysis and modeling of time-correlated failures in large-scale distributed systems. In Grid Computing (GRID), 2010 11th *IEEE/ACM International Conference*, 65−72.

[24] ParallelWorkloadArchive. http://www.cs.huji.ac.il/labs/parallel/workload/.

[25] Kansal, Nidhi Jain, and Inderveer Chana.[2012] Existing load balancing techniques in cloud computing: a systematic review. *Journal of Information Systems and Communication* ,3(1): 87−91.

www.iioab.org
THE IIOAB JOURNAL
www.iioab.webs.com
COMPUTER SCIENCE

THE IIOAB JOURNAL
ISSN: 0976-3104

www.iioab.org

THE IIOAB JOURNAL

www.iioab.webs.com

ABOUT AUTHORS

Prakash Kumar *has received his B. Tech. in Electronics and Communication Engineering, and M. Tech in Computer Science and Technology from University of Roorkee (Now Indian Institute of Technology, Roorkee), India. He is currently an Assistant Professor (Senior Grade) in Jaypee Institute of Information Technology,(Deemed University), Noida, India. His area of interest is in Computer Networks and Communications, Distributed Computing, Cloud Computing and Virtualization. He is currently pursuing his Ph D in the field of virtualization of resources viz. Systems and network resources. His main focus is on Trust, Reliability and Fault Tolerant networks and systems for distributed and cloud environments.*

Prof. Krishna Gopal *is currently Dean Academic and Research at JIIT, Noida, India since 2011. He is Ph. D. from REC KurukshetraKurukshetra University Kurukshetra, India. He is having 45 years of teaching and R&D experience. He received his Bachelor, Master and PhD in Electronics engineering from IIT, Madras, RECKurukshetra in 1966, 1972, 1979 respectively. He published more than 100 papers in different journals, conferences, patents etc. He is member of various professional bodies like: Life Member System Society of India, Indian Society for Technical Education, senior member of IEEE etc. His area of interest is Reliability and Fault Tolerant Networks and communication Systems.*

Prof. J P Gupta *has received his Ph D degree from University of Westminster, UK. He is currently the Director Emeritus (QA) at Hydrocarbons Education and Research Society, New Delhi, India. He is an academician having more than 35 years experience including Professor at IIT Roorkee, India, Vice Chancellor at JIIT Noida,India, Galgotia University, and Sharda University, India. He is author of more than 70 research papers published in International journals and conferences. He has received Commonwealth Fellowship and many more awards and memberships to his credit.*