***I.J. Mathematical Sciences and Computing*, 2015, 4, 1-10**
Published Online November 2015 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2015.04.01

***Available online at* http://www.mecs-press.net/ijmsc**

Efficient Optimization of Edge Server Selection Technique in Content Delivery Network

Debabrata Sarddar[1], Enakshmi Nandi[2]

[1]*Assistant Professor, Department of Computer Science & Engineering, Kalyani, Nadia, 741235, India*
[2]*Ph.D. Student, Department of Computer Science & Engineering, University of Kalyani, 741235, Kalyani, India*

Abstract

Cloud Computing provides the infrastructure as a "*Cloud*" from which businesses and users are permit to access applications from anywhere in the world on demand. Thus, the computing world is rapidly transforming towards developing software for millions to consume as a service, rather than to run on their individual computers. But many users could not satisfy on cloud services completely due to their uncovering security purpose for handling large numbers of data. Even the network becomes uncontrollable, when large numbers of user's request to the server create network congestion and data losses vigorously. Content Delivery Network OR CDN is an eminent solution of this problem. Our objective is to create optimized method for edge selection technique in Content Delivery Network to deliver and direct the user request to the nearest edge server and establish the connection between them and transfer the respective content

Index Terms: CDN, Haversine Distance, and Floyd's Algorithm, Least recently used

1. Introduction

A content distribution network is a large, cumulative collection of network element spanning the internet, where contents are replicated over different mirrored web server. It helps to deliver the content to end users transparently, improving the performance through maximize the bandwidth and improve the accessibility and served improved quality of service or QoS through overcoming all bindings. If the CDN server is closer to users then the content deliver to them with earlier. CDN not only provides high availability and good performance but also provides high security from DOS attacks through their large distributed server infrastructure to absorb the attack traffic. Here the replica of end user's content are distributed among different

* Corresponding author. Tel.:
E-mail address: dsarddar1@gmail.com, pamelaroychowdhurikalyani@gmail.com

edge servers using content replica placement algorithm. In this paper we want to optimize the searching technique of edge server using Floyd's algorithm and web data collector concept over content delivery network, so that the searching process of edge servers of content delivery network will be very fast.

1.1. Related Works

S.Saroiu, R.J.Dunn, H.M.Levy and so on are the important name in the field of CDN. They examined on the basis of four content delivery system such as web traffic, Content Delivery Network of the Akamai and Kazza,Gnutella peer-to-peer file sharing traffic[1].Recently Rajkumar Buyya, Al-Mukaddim works on detail taxonomy of Content Delivery Network with respect to four impact factors such as request-routing procedure, content replication mechanisms, cache management ,load balancing methodology [2]. Phooi Yee Lau,Sungkwon Park research on the basis of relationship between the placement strategy and throughput among different edge servers [3].

2. Proposed Work

In this paper our aim is to create the respective distance from different users to host server which is connect to web data collector which is adhere to different edge servers via content delivery network. We want to optimize the edge server selection process by creating haversine distance and then calculate shortest path distance through Floyd's algorithm and reduce the load over content delivery network and deliver the data as soon as possible. At first that the earth is divided into six regions: North and South America , Europe ,Asia, Africa, Australia as shown in the figure below(R0,R1,R2,R3,R4,R5,R6) . At first users send request for a definite content to host server, if they can't find then request is terminate the request to web data collector. Such as an Australian's users send the request to host server if they can't find the required content then terminate the request to web data collector. Host server connected with web data collector, where an indexed table found. The replica of all contents lies in host server and this indexed table. The replica of contents lies in this indexed table according to shortest distance and low latency basis. This table holds the relevant information of the connected edge servers and the associated databases of the web-sites for which the request is to be handled. To optimize the load over CDN and proper process of edge server selection has approached if the content does not match with the host server content then automatically request will be transfer to CDN through web data collector. In web data collector contains the indexed table, where contents contain in the edge server according to shortest distance edge server basis. This shortest distance edge server works on Floyd's algorithm basis.

2.1. Find the Haversine Distance for Finding the Distance of Different Edge Server from Client

The haversine formula is used for calculating distance between sender and receiver on the earth evaluated from their given longitudes and latitudes [7].

The following algorithm is used for calculating distance between sender and receiver.

```
1. R ⟵ 6371 // Radius of Earth in k.m
2. $\Phi1$ ⟵  lat1.toRadians ()// Latitude of Sender
3. $\Phi2$ ⟵  lat2.toRadians()// Latitude of receiver
4. $\lambda1$ ⟵ lon1.toRadians()// Longitude of sender
5. $\lambda2$ ⟵ lon2.toRadians()// Longitude of receiver
6. $\Delta\Phi$ ⟵   (lat2-lat1).toRadians()
7. $\Delta\lambda$ ⟵   (lon2-lon1).toRadians()
```

```
8. a = Math.sin(ΔΦ/2) * Math.sin(ΔΦ/2) + Math.cos(Φ1)* Math.cos(Φ2) * Math.sin(Δλ/2) * Math.sin(Δλ/2)
9.c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a))
10.d = R * c // Haversine distance between sender and receiver over the earth surface
```

2.2. After Finding the Distance from Client to Different Edge Server We Use Floyd's Algorithm for Finding Shortest Distance Edge Server is as Follows

When solving practical transportation problems, it is often necessary to calculate the shortest path between all pairs of nodes in a transportation network. If the network has n nodes, then the Dijkstra algorithm must be applied n times, taking a different node each time as the starting node. This is a lengthy process. Hence, Dijkstra's algorithm is rarely used to determine the shortest path between all pairs of nodes; instead Floyd's algorithm is used. The algorithm works by updating two matrices, namely D_k and Q_k, n times for a n - node network. The matrix D_k, in any iteration k, gives the value of the shortest distance (time) between all pairs of nodes (i, j) as obtained till the k^{th} iteration. The matrix Q_k has q_{ij}^{k} as its elements. The value of q_{ij}^{k} gives the immediate predecessor node from node i to node j on the shortest path as determined by the k^{th} iteration. D_o a Q_o give the starting matrices and D_n and Q_n give the final matrices for an n-node system. The first task is to determine D_o and Q_o. D_o is taken up first. The element d_{ij} of matrix D_o are defined as follows:

If a link (branch) exists between nodes i and j ,the length of the shortest path between these nodes equals length $l\ (i, j)$ of branch (i, j) which connects them. Should there be several branches between nodes i and node j, the length of the shortest path d_{ij}^{0} must equal the length of the shortest branch, i.e.: [8]

$$d_{ij}^{0} = \min\,[l_1(i,j), l_2(i,j), \ldots\ldots l_m(I,j)$$

where, m is the number of branches between node i and node j.

It is clear that $d_{ij}^{0} = 0$ when $i = j$. In the case when there is no direct link between node i and node j, we have no information at the beginning concerning the length of the shortest path between these two nodes so we treat them as though they were infinitely far from each other, that is, $d_{ij}^{0} = \infty$

Elements q_{0j}^{0} of the predecessor matrix Q_o are defined as follows:

First, we assume that $q_{0j}^{0} = i$, for $i \neq j$, i.e. that for every pair of nodes (i, j) for the immediate predecessor of node j on the shortest path leading from node i to node j is actually node i. After defining $D_o$and Q_o the following steps are used repeatedly to determine D_n and Q_n.

Step 1 : Let $k = 1$

Step 2 : We calculate elements d_{ij}^{k} of the shortest path length matrix found after the k-th passage through algorithm D_k using the following equation:

$$d_{ij}^{k} = \min|\, d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1} \,|$$

Step 3: Elements q_{ij}^{k} of predecessor matrix Q_k found after the *k-th* passage through the algorithm are calculated as follows:

$$q_{ij}^{k} = < \frac{q_{kj}^{k-1},\ for, d_{ij}^{k} \neq d_{ij}^{k-1}}{q_{ij}^{k-1}, Otherwise}$$

Step 4: If $k = n$, the algorithm is finished. If $k < n$, increase k by 1, i.e. $K = k+1$ and return to step 2.Let us now look at the algorithm in a little more detail. In step 2, each time we go through the algorithms we are checking as to whether a shorter path exists between node i and j other than the path we already know about which was established during one of the earlier passages through the algorithm. If we establish that $d_{ij}^{k} \neq d_{ij}^{k-1}$,i.e. if we establish during the k *-th* passage through the algorithm that the length of the shortest path d_{ij}^{k} between nodes i and j is less than the length of the shortest path d_{ij}^{k-1} known previous to the k*-th* passage, we have to change the immediate predecessor node to node j. Since the length of the new shortest path is:

$$d_{ij}^{k} = d_{ik}^{k-1} + d_{kj}^{k-1}$$

it is clear that in this case node k is the new immediate predecessor node to j, and therefore: $q_{ij}^{k} = q_{kj}^{k-1}$

This is actually done in the third algorithmic step. It is also clear that the immediate predecessor node to node j does not change if, at the end of step 2, we have established that no other new, shorter path exists. This means that:

$$q_{ij}^{k} = q_{kj}^{k-1} \text{ for } d_{ij}^{k} = d_{ij}^{k-1}$$

When we go through the algorithm n times (n is the number of nodes in the transportation network), elements d_{ij}^{n} of final matrix D_n will constitute the shortest path going from node i to node j

2.3. Least Recently Used Algorithm

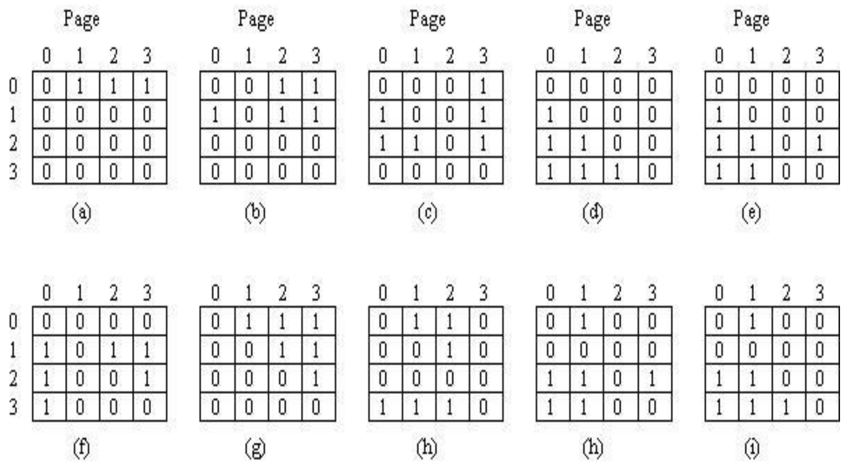
Although we cannot implement an optimal algorithm by evicting the page that will not be used for the longest time in the future, we can approximate the algorithm by keeping track of when a page was last used. If a page has recently been used then it is likely that it will be used again in the near future. Conversely, a page that has not been used for some time is unlikely to be used again in the near future. Therefore, if we evict the page that has not been used for the longest amount of time we can implement a least recently used (LRU) algorithm. Whilst this algorithm can be implemented (unlike the optimal algorithm) it is not cheap. Ideally, we need to maintain a linked list of pages which are sorted in the order in which they have been used. To maintain such a list is prohibitively expensive (even in hardware) as deleting and moving list elements is a time consuming process. Sorting a list is also expensive. However, there are ways that LRU can be implemented in hardware. One way is as follows. The hardware is equipped with a counter (typically 64 bits). After each instruction the counter is incremented. In addition, each page table entry has a field large enough to accommodate the counter. Every time the page is referenced the value from the counter is copied to the page table field. When a page fault occurs the operating system inspects all the page table entries and selects the page with the lowest counter. This is the page that is evicted as it has not been referenced for the longest time. Another hardware implementation of the LRU algorithm is given below.[9]

If we have n page table entries a matrix of n x n bits, initially all zero, is maintained. When a page frame, k, is referenced then all the bits of the k row are set to one and all the bits of the k column are set to zero. At any time the row with the lowest ***binary*** value is the row that is the least recently used (where row number = page frame number). The next lowest entries are the next recently used; and so on.

If we have four pages frames and access them as follows

0 1 2 3 2 1 0 3 2 3

it leads to the algorithm operating as follows. It might be worth working through it and calculating the binary value of each row to see which page frame would be evicted.



Page

	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

(a)

Page

	0	1	2	3
0	0	0	1	1
1	1	0	1	1
2	0	0	0	0
3	0	0	0	0

(b)

Page

	0	1	2	3
0	0	0	0	1
1	1	0	0	1
2	1	1	0	1
3	0	0	0	0

(c)

Page

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

(d)

Page

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	1
3	1	1	0	0

(e)

	0	1	2	3
0	0	0	0	0
1	1	0	1	1
2	1	0	0	1
3	1	0	0	0

(f)

	0	1	2	3
0	0	1	1	1
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0

(g)

	0	1	2	3
0	0	1	1	0
1	0	0	1	0
2	0	0	0	0
3	1	1	1	0

(h)

	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	1	1	0	1
3	1	1	0	0

(h)

	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	1	1	0	0
3	1	1	1	0

(i)

2.4. *Web Data Collector*

Web data collector maintains the following track record

i. Web data collector connected to all edge servers.
ii. Web data collector maintains an indexed table based on shortest distance edge server from user which contains replica of contents of web service.
iii. This indexed table maintains all records of parameters and arranged according to shortest distance edge server and high throughput basis.
iv. Web data collector maintains another table based on least recently used edge server which contains information about least recent edge servers table along with parameter like workload, network throughput, average waiting time, storage capacity, channel capacity and so on. According to these features all the least recently used edge servers in this second table are arranged and updated periodically after performing the request response operation of delivery content from respective edge server.
v. Web data collector provides secured file sharing services between edge server and system manager.
vi. When the request come from host server to Web data collector then it checks to its indexed list and most frequently used edge server list that which server has that content that match the request content. If match found then that respective IP address of that edge server send to user and the corresponding edge server then directly connected to user for delivering the content. If match not found then it checks the next server from table. If in case it is found that the requested content will contain in two nearest edge servers in Web data collector then Web data collector select the edge server which gives better performance according to checking the scheduling criteria from the table.

2.5. *Calculate the Latency from Sender to Receiver*

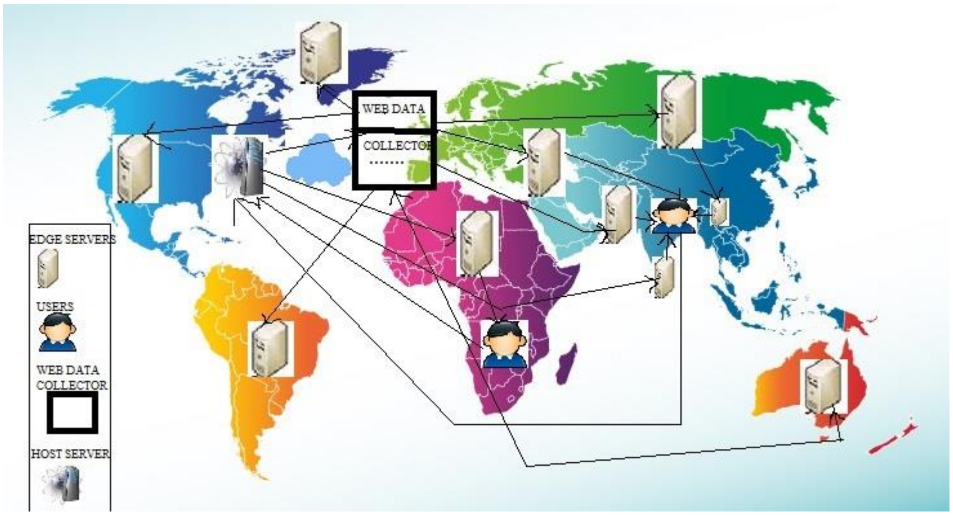
Latency is an expression of how much time it required for a packet of data to get from one to another point.

- Latency = Propagation delay +Queuing delay +Serialization delay
- Propagation delay =haversine distance /speed of light
- Queuing delay = queue depth (bits) / link data rate (bits per second)
- Serialization delay =Packet size in bit/Transmission rate in bits per second [7].

2.6. Algorithm for Selecting Edge Server for a Particular CDN

1. Let U= {$u_{1,}u_2$, u_3,u_4…….u_n} be the set of users from six geographic locations.
2. And E= {e1, e2, e3…en} be the set of edge servers and web data collector situated centrally and connected to both host server and CDN edge servers.
3. When a client from set U sends a request to a specific region to host server, Then host server check the requested content to its contain servers, if match found then stop process. Otherwise host server terminates the request to web data collector.
4. Web data collector first identifies the client's location i.e. longitude, latitude and IP address and then it looks up indexed table to find that which nearest edge sever have the replica of requested content and then check the next least significant used edge server table to get the proper edge server with its scheduling criteria accordingly and also check which type of content is to be served; it sends the request to the proper edge servers through registered CDN.
    a. If it finds the IP address and associated geo-coordinates with edge parametric in these tables then it immediately directed the associated edge server with the corresponding delivery content to the respective client.
    b. Else go to step 6;
5. Now the request comes from a new client –
    a. Calculate the shortest distance between the user and edge servers of the CDN using floyd'smethod, and store the shortest distance edge server with its IP address to indexed table accordingly.
6. Now choose first shortest edge server and check it to least significantly used edge server list for its respective parametric values.
    a. If match found then For each temporary edge server ,i.e. temp_server
        i. If(getNetworkthroughput(edge_server)<getNetworkThroughput(temp_serve)) then selected_edge_server=edge_server ;
        ii. Else selected_edge_server=temp_server;

    End for
7. After the selection of the edge server, the parameters of the selected_edge_server and the client are written to the least significant used server list.
8. Now the original request of the client is forwarded to the selected_edge_server and then selected_edge_server transmit the definite contents immediately.
9. Now the connection between the selected_edge_server and the client is established.
10. If match not found go to first indexed table and search for requested content to all edge server.
11. If user again sends a request for either some dynamic content or for static content then
    a. If request is for dynamic content then:
        i. selected_edge_server acts as the proxy and fetch the dynamic content from the host web server and sends it to the client.
    b. If request is for content then :
        i. If content is available then transmit the static content immediately.
        ii. Else if it is not available in selected_edge_server then it will search for content by sending a search query to its siblings and if found redirects it to the client.
        iii. Else if it is not available within the CDN, then a brought back request is sent to the host server and the static content is fetched, simultaneously content is distributed over the CDN with the help of RMS.
12. For each request repeat step 5 to step 11
13. End



2.7. Practical Example with Result Analysis

- As we see in the above figure the respective client stay in R5 region, i.e. Africa. She sends request to her host server in Africa for respective content. She found the respective content to her respective server. She found that content then no need for CDN.
- The client from R4 region i.e. Asia (Kolkata) ,send request for mail service for sending 1400 bytes data using the mail server to host server but no one match with requested content ,then host server terminate the request to web data collector[7].
- Web data collector then find shortest distance edge server from the indexed table by using Floyd's algorithm and finding the definite mail service from two respective tables with low latency value and minimal response time from corresponding list and directed the particular edge server in Content Delivery Network with the Kolkata's client.
- CDN provider stores the details information of all edge servers through web data collector. As shown in the table1. web data collector calculates the mean waiting time using multi level queuing formula and latency of every edge servers from previous formula.

Table 1. Consider the name of places of edge server of particular CDN

Name of the region	Name of the place of edge server
1. Asia	Colombo
	New Delhi
2. Europe	London
3. North America	Mexico
	Washington (Host Server)
4. South America	Buenos Aires
5. Africa	Cape Town
6. Australia	Canberra

Table 2. Mean Waiting Time of each edge servers using multilevel queue

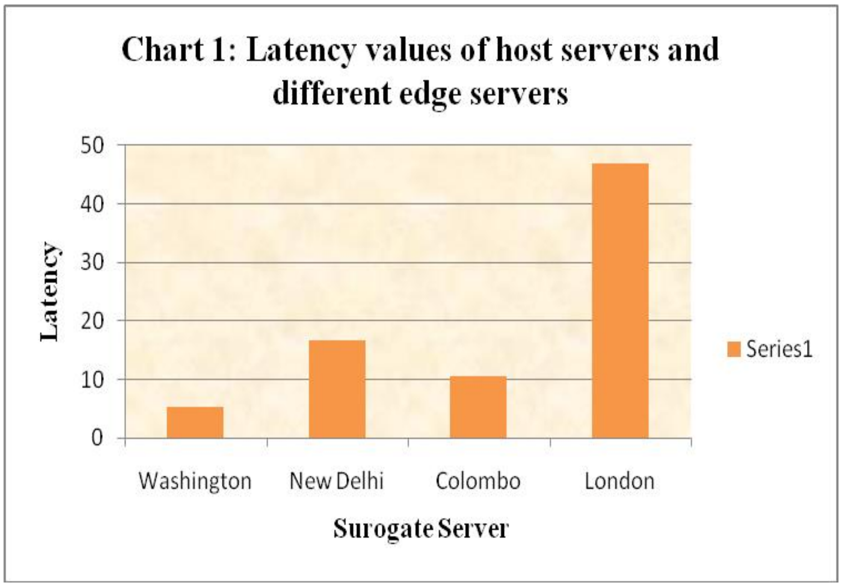
Host Server	**Arrival rate**	**Service rate**	**Mean Waiting Time using Multilevel Queue**
Washington	19	30	0.000056
Edge Server	**Arrival rate**	**Service rate**	**Mean Waiting Time using Multilevel Queue**
London	18	30	0.000036
New Delhi	115	118	0.000015
Colombo	112	116	0.000025

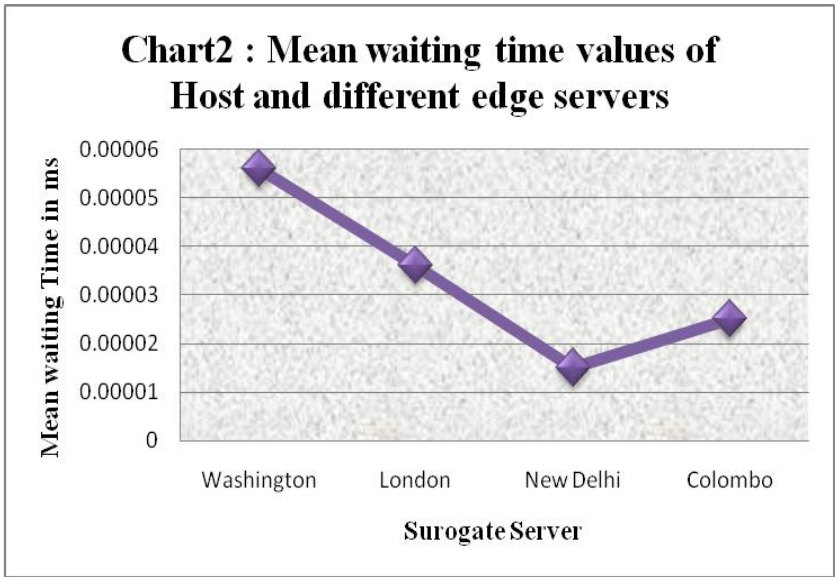
Table 3. Shortest Edge Server distance using Floyd's algorithm and bandwidth of host server and each edge server

Host Server	**Shortest Distance edge server using Floyd's algorithm**	**Bandwidth**
Washington	13840 (Distance from host server)	10 Gbps
Edge Server	**Shortest Distance edge server using Floyd's algorithm**	**Bandwidth**
New Delhi	1131	1Mbps
Colombo	1741	350Mbps
London	6212	100Mbps

Table 4. Latency of Host Server and each Edge Server using throughput formula in multilevel queuing level

Operating Host Server	Latency (ms)
Washington	5.321
Edge Server	**Latency (ms)**
New Delhi	16.5732
Colombo	10.5347
London	46.8315





2.8. Result Analysis

By analyzing above result, we see that the client of Asia (Kolkata) send request to his host server at Siliguri, but he can't find the respective content what he wants. So the request automatically terminates from Siliguri to Washington, where web data collector presents which is connected to CDN.CDN check the required content to its table. Web data collector first calculate the shortest distance edge server from Kolkata using Floyd's algorithm where the required content founds from its contained indexed and least significant used edge server tables. These tables track all records about all edge server as we see these tables calculates that Colombo is that shortest distance place where that respective content found and latency value is high mean waiting time is less comparative to next shortest distance edge server of client from Kolkata. If the required contents found in Siliguri then there is no need for CDN in that case load for edge servers in CDN will be reduce. Here latency calculation has done by multilevel queuing algorithm and other parametric values of each edge server through this method.

3. Conclusion and Future Scope

In this paper we want to optimize the edge server selection process using web data collector concept over CDN. We have done that work with efficiently. Our simulation analysis shows that result that CDN becomes an efficient way of delivering both static and dynamic contents faster to client with the help of web data collector. Web data collector scale down the workload over CDN and help to improve the network performance with efficiently. But if we use pipelining concept and another flooding algorithm for obtaining more optimize route for searching edge servers and handling more request in parallel over CDN then the performance will be very impressive in future.

Acknowledgements

We would like to thank our HOD Dr. kalyani Mali in department of Computer Science and Engineering in University of Kalyani for purpose of laboratory and university infrastructure

References

[1] Saroiu, Stefan, et al. "An analysis of internet content delivery systems." *ACM SIGOPS Operating Systems Review* 36.SI (2002): 315-32.

[2] Pathan, Al-Mukaddim Khan, and Rajkumar Buyya. "A taxonomy and survey of content delivery networks." Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report (2007).

[3] Wong, Khai Hsiang. Using surrogate servers for content delivery network infrastructure with guaranteed *QoS*. Diss. UTAR, 201.

[4] file:///C:/Users/User/Desktop/ph.d%20documents/Dijsktra%27s%20algorithm.htm.

[5] Books of Operating System Concepts: International Student Version Paperback – 20 Apr 2009 by Silberschatz (Author), Galvin (Author), Gagne (Author).

[6] https://www.excentis.com/blog/measuring-throughput-effect-used-tcp-settings.

[7] Sarddar, Debabrata, Sandip Roy, and Rajesh Bose. "Queueing Based Edge Server Selection in Content Delivery Network Using Haversine Distance.".

[8] http://nptel.ac.in/courses/105104098/TransportationII/mod13/16slide.htm,Module XIII: Demand analysis.

[9] http://www.cs.nott.ac.uk/~gxk/courses/g53ops/Memory%20Management/MM14-LRU.html.

Authors' Profiles

Debabrata Sarddar is an Assistant Professor at the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, India. He completed his PhD from Jadavpur University. He did his M. Tech in Computer Science & Engineering from DAVV, Indore in 2006, and his B.E in Computer Science & Engineering from NIT, Durgapur in 2001. He has published more than 75 research papers in different journals and conferences. His research interests include Wireless and Mobile Systems and WSN, and Cloud computing.

Enakshmi Nandi received her M.Tech in VLSI and Microelectronics from Techno India, Salt Lake, West Bengal and B.Tech in Electronics and Communication Engineering from JIS College Of Engineering, West Bengal under West Bengal University of Technology, West Bengal, India. At present, she is Research scholar in Computer Science and Engineering from University of Kalyani. Her research interests include Cloud Computing, Mobile Communication system, Device and Nanotechnology.

How to cite this paper: Debabrata Sarddar, Enakshmi Nandi,"Efficient Optimization of Edge Server Selection Technique in Content Delivery Network", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.1, No.4, pp.1-10, 2015.DOI: 10.5815/ijmsc.2015.04.01