PROJECT REPORT

on

Load Balancing Algorithm for Content Delivery Network

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

COMPUTER SCIENCE & ENGINEERING

under the Supervision of

PUNIT GUPTA

By

Anushree Gupta

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled "Load Balancing Algorithm for Content Delivery Network ", submitted by Anushree Gupta in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: 15/05/2015 **Punit Gupta**

Acknowledgement

I am highly indebted to Mr. Punit Gupta for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in implementing the project.

I would like to express my gratitude towards my parents & members of Jaypee University of Information Technology for their kind co-operation and encouragement which helped me in this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

Date: 15/05/2015 Anushree Gupta

*

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Content Delivery Networks (CDNs) have evolved to overcome the inherent limitations of the Internet in terms of user perceived Quality of Service (QoS) when accessing Web content. A CDN replicates content from the origin server to cache servers, scattered over the globe, in order to deliver content to end-users in a reliable and timely manner from nearby optimal surrogates. Content distribution on the Internet has received considerable research attention. It combines development of high-end computing technologies with high performance networking infrastructure and distributed replica management techniques.

Over the last decade, considerable research efforts and momentum have been directed into this sphere, both from the academia and the commercial developers. It could undoubtedly be considered as one of the top emerging technologies that will have a major impact on the quality of science and society over the next 20 years. Having said that a glimpse of the technological trends and future directions in this domain would be helpful to position researchers and practitioners at the forefront of the field.

A critical component of CDN architecture is the request routing mechanism. It allows to direct users' requests for content to the appropriate server based on a specified set of parameters. The proximity principle,by means of which a request is always served by the server that is closest to the client, can sometimes fail. Indeed, the routing process associated with a request might take into account several parameters (like traffic load, bandwidth, and servers' computational capabilities) in order to provide the best performance in terms of time of service, delay, etc. Furthermore, an effective request routing mechanism should be able to face temporary, and potentially localized, high request rates (the so-called flash crowds) in order to avoid affecting the quality of service perceived by other users.

CHAPTER 1

1.1 EVOLUTION OF CDN OVER INTERNET

Over the last decades, users have witnessed the growth and maturity of the Internet. As a consequence, there has been an enormous growth in network traffic, driven by rapid acceptance of broadband access, along with increases in system complexity and content richness. The over-evolving nature of the Internet brings new challenges in managing and delivering content to users. As an example, popular Web services often suffer congestion and bottleneck due to the large demands made on their services. A sudden spike in Web content requests may cause heavy workload on particular Web server(s), and as a result a hot spot can be generated. Coping with such unexpected demand causes significant strain on a Web server. Eventually the Web servers are totally overwhelmed with the sudden increase in traffic, and the website holding the content becomes temporarily unavailable.

Content providers view the Web as a vehicle to bring rich content to their users. A decrease in service quality, along with high access delays mainly caused by long download times, leaves the users in frustration. Companies earn significant financial incentives from Web-based e-business. Hence, they are concerned to improve the service quality experienced by the users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructures supporting these technologies form a new type of network, which is often referred to as content network.

Several content networks attempt to address the performance problem through using different mechanisms to improve the Quality of Service (QoS). One approach is to modify the traditional Web architecture by improving the Web server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not flexible. Moreover, small enhancements are not possible and at some point, the complete server system might have to be replaced. Caching proxy deployment by an ISP can be beneficial for the narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. Caching proxies may also be equipped with technologies to detect a server failure and maximize efficient use of caching proxy resources. Users often configure their browsers to send their Web request through these caches rather than sending directly to origin servers. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. Thus, the caches contain most popular content viewed by all the users of the caching proxies. A provider may also deploy different levels of local, regional, international caches at geographically distributed locations. Such arrangement is referred to as hierarchical caching. This may provide additional performance improvements and bandwidth savings.

A more scalable solution is the establishment of server farms. It is a type of content network that has been in widespread use for several years. A server farm is comprised of multiple Web servers, each of them sharing the burden of answering requests for the same website. It also makes use of a Layer 4-7 switch, Web switch or content switch that examines content request and dispatches them among the group of servers. A server farm can also be constructed with surrogates instead of a switch. This approach is more flexible and shows better scalability. Moreover, it provides the inherent benefit of fault tolerance. Deployment and growth of server farms progresses with the upgrade of network links that connects the Web sites to the Internet.

Although server farms and hierarchical caching through caching proxies are useful techniques to address the Internet Web performance problem, they have limitations. In the first case, since servers are deployed near the origin server, they do little to improve the network performance due to network congestion. Caching proxies may be beneficial in this case. But they cache objects based on client demands. This may force the content

providers with a popular content source to invest in large server farms, load balancing, and high bandwidth connections to keep up with the demand. To address these limitations, another type of content network has been deployed in late 1990s. This is termed as Content Distribution Network or Content Delivery Network, which is a system of computers networked together across the Internet to cooperate transparently for delivering content to end-users.

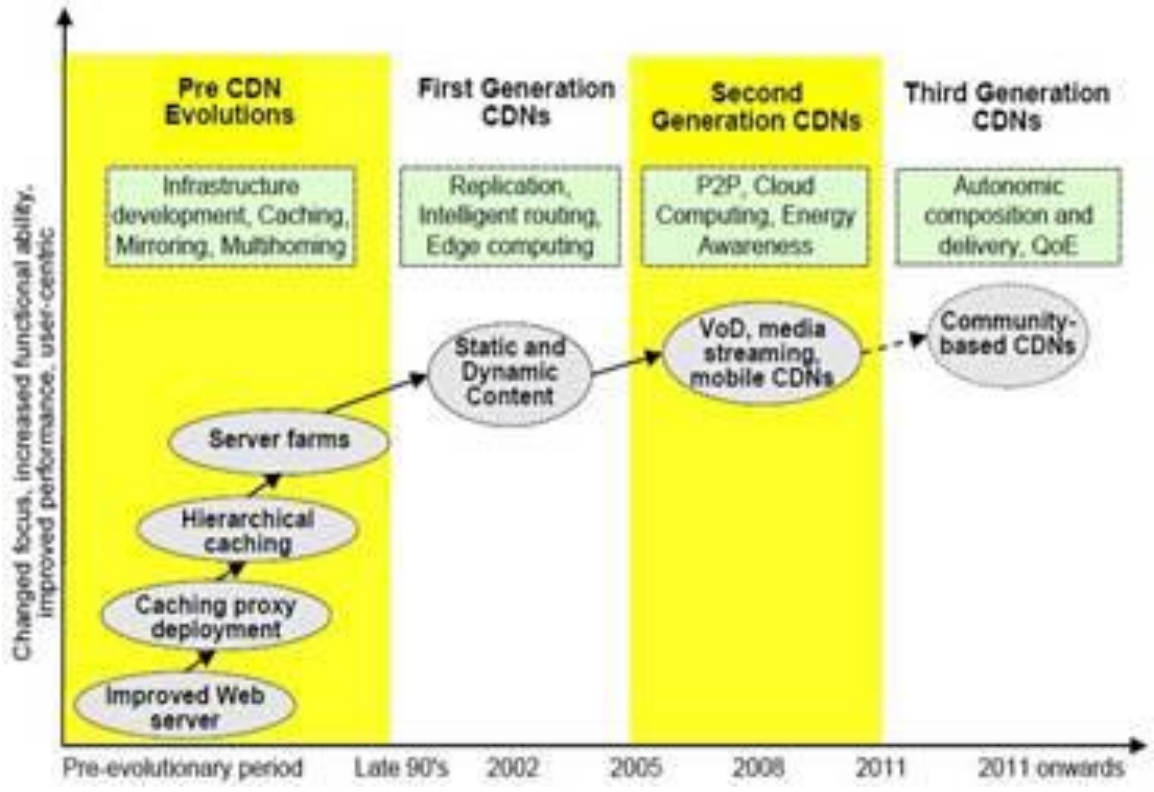


FIGURE 1: EVOLUTION OF CDN

With the introduction of CDN, content providers started putting their Web sites on a CDN. Soon they realized its usefulness through receiving increased reliability and scalability without the need to maintain expensive infrastructure. Hence, several initiatives kicked off for developing infrastructure for CDNs. As a consequence, Akamai Technologies evolved out of an MIT research effort aimed at solving the flash crowd problem. Within a couple of years, several companies became specialists in providing fast and reliable delivery of content, and CDNs became a huge market for generating large revenues. The flash crowd events like the 9/11 incident in USA [98], resulted in serious caching problems for some site. This influenced the CDN providers to invest more in CDN infrastructure development, since CDNs provide desired level of protection to Websites against flash crowds. First generation CDNs mostly focused on static or Dynamic Web documents. On the other hand, for second generation of CDNs the focus has shifted to Video-on-Demand (VoD), audio and video streaming. But they are still in research phase and have not reached to the market yet.

With the booming of the CDN business, several standardization activities also emerged since vendors started organizing themselves. The Internet Engineering Task Force (IETF) as a official body took several initiatives through releasing RFCs (Request For Comments) . Other than IETF, several other organizations such as Broadband Services Forum (BSF) , ICAP forum , Internet Streaming Media Alliance [took initiatives to develop standards for delivering broadband content, streaming rich media content – video, audio, and associated data – over the Internet. In the same breath, by 2002, large-scale ISPs started building their own CDN functionality, providing customized services. In 2004, more than 3000 companies were found to use CDNs, spending more than $20 million monthly. A market analysis shows that CDN providers have doubled their earnings from streaming media delivery in 2004 compared to 2003. In 2005, CDN revenue for both streaming video and Internet radio was estimated to grow at 40%. A recent marketing research shows that combined commercial market value for streaming audio, video, streaming audio and video advertising, download media and entertainment was estimated at between $385 million to $452 million in 2005. Considering this trend, the market was forecasted to reach $2 billion in fouryear (2002-2006) total revenue in 2006, with music, sports, and entertainment subscription and download revenue for the leading content categories. However, the latest report from AccuStream iMedia Research reveals that since 2002, the CDN market has invested $1.65 billion to deliver streaming media (excluding storage, hosting, applications layering), and the commercial market value in 2006 would make up 36% of the $1.65 billion four-year total in media and entertainment, including content, streaming advertising, movie and music downloads and User Generated Video (UGV) distribution . A detailed report on CDN market opportunities, strategies, and forecasts for the period 2004-2009, in relation to streaming media delivery can be found in .

With the proliferation of the Internet, popular Web services often suffer congestion and bottlenecks due to large demands made on their services. Such a scenario may cause unmanageable levels of traffic flow, resulting in many requests being lost. Replicating the same content or services over several mirrored Web servers strategically placed at various locations is a method commonly used by service providers to improve performance and scalability. The user is redirected to the nearest server and this approach helps to reduce network impact on the response time of the user requests.

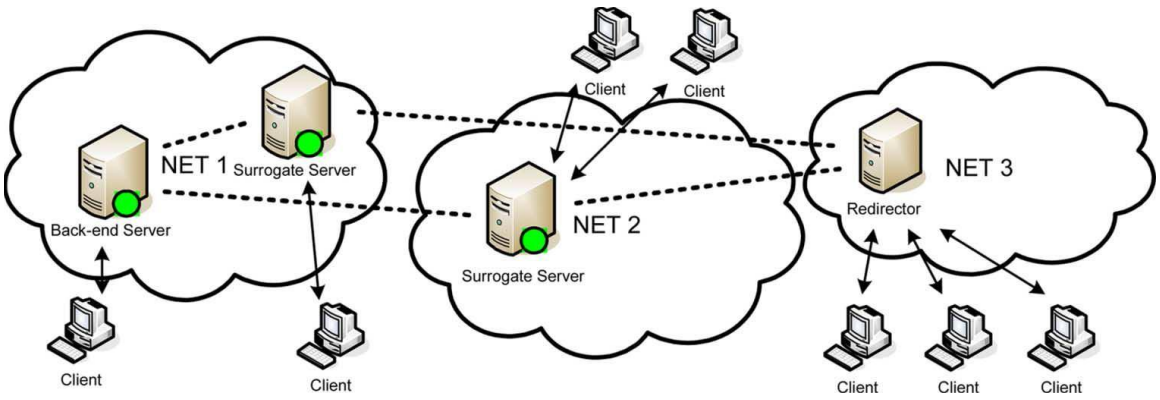


FIGURE 2: CONTENT DELIVERY NETWORK

Content Delivery Networks provide improved network performance by maximizing bandwidth, improving accessibility and maintaining correctness through content replication. They offer fast and reliable applications and services by distributing content to cache or edge servers located close to users. A CDN has some combination of content-delivery, request-routing, distribution and accounting infrastructure. The content-delivery infrastructure consists of a set of edge servers (also called surrogates) that deliver copies of content to end-users. The request-routing infrastructure is responsible to directing client request to appropriate edge servers. It also interacts with the distribution infrastructure to keep an up-to-date view of the content stored in the CDN caches. The distribution infrastructure moves content from the origin server to the CDN edge servers and ensures consistency of content in the caches. The accounting infrastructure maintains logs of client accesses and records the usage of the CDN servers. This information is used for traffic reporting and usage-based billing. In practice, CDNs typically host static content including images, video, media clips, advertisements, and other embedded objects for dynamic Web content. Typical customers of a CDN are media and Internet advertisement companies, data centers, Internet Service Providers (ISPs), online music

A CDN is a collection of network elements arranged for more effective delivery of content to end-users . Collaboration among distributed CDN components can occur over nodes in both homogeneous and heterogeneous environments. CDNs can take various

forms and structures. They can be centralized, hierarchical infrastructure under certain administrative control, or completely decentralized systems. There can also be various forms of inter networking and control sharing among different CDN entities. The typical functionality of a CDN includes:

1. Request redirection and content delivery services to direct a request to the closest suitable surrogate server using mechanisms to bypass congestion, thus overcoming flash crowds or SlashDot effects.
2. Content outsourcing and distribution services to replicate and/or cache content to distributed surrogate servers on behalf of the origin server.
3. Content negotiation services to meet specific needs of each individual user (or group of users).
4. Management services to manage the network components, to handle accounting, and to monitor and report on content usage.

A CDN provides better performance through caching or replicating content over some mirrored Web servers (i.e. *surrogate servers*) strategically placed at various locations in order to deal with the sudden spike in Web content requests, which is often termed as *flash crowd* or *SlashDot effect* . The users are redirected to the surrogate server nearest to them. This approach helps to reduce network impact on the response time of user requests. In the context of CDNs, *content* refers to any digital data resources and it consists of two main parts: the *encoded media* and *metadata* . The encoded media includes static, dynamic and continuous media data (e.g. audio, video, documents, images and Web pages). Metadata is the content description that allows identification, discovery, and management of multimedia data, and also facilitates the interpretation of multimedia data. Content can be pre-recorded or retrieved from live sources; it can be persistent or transient data within the system. CDNs can be seen as a new virtual overlay to the Open Systems Interconnection (OSI) basic reference model. This layer provides overlay network services relying on application layer protocols such as HTTP or RTSP for transport .

Figure 3 provides a high-level view of the request-routing in a CDN environment. The interaction flows are: (1) the client requests content from the content provider by specifying its URL in the Web browser. Client's request is directed to its origin server; (2) when origin server receives a request, it makes a decision to provide only the basic content (e.g. index page of the Web site) that can be served from its origin server; (3) to serve the high bandwidth demanding and frequently asked content (e.g. embedded objects – fresh content, navigation bar, banner ads etc.), content provider's origin server redirects client's request to the CDN provider; (4) using the proprietary selection algorithm, the CDN provider selects the replica server which is 'closest' to the client, in order to serve the requested embedded objects; (5) selected replica server gets the embedded objects from the origin server, serves the client requests and caches it for subsequent request servicing.

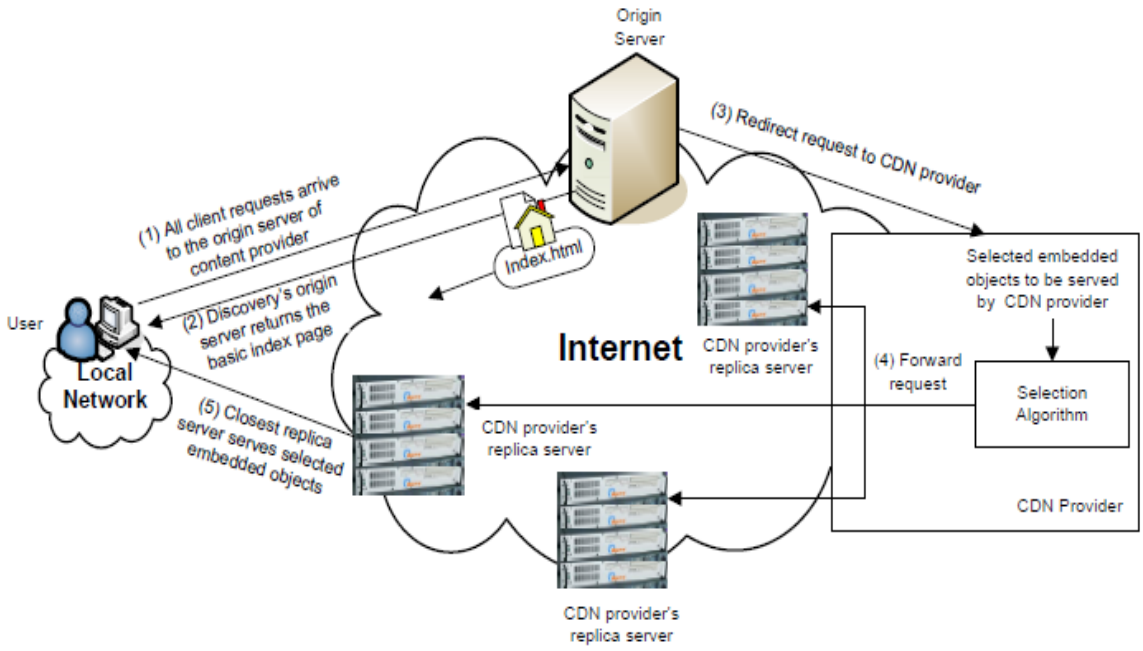


FIGURE 3: REUQEST ROUTING IN CDN ENVIRONMENT

The three key components of CDN architecture are:

1. Content Provider
2. CDN Provider
3. End User

1. **Content provider** or customer is the one who delegates the URI name space of the Web objects to be distributed. The origin server of the content provider holds those objects.
2. **CDN Provider** is a proprietary organization or company that provides infrastructure facilities to content providers in order to deliver content in a timely and reliable manner.
3. **End Users** or clients are the entities who access content from the content provider's website.

Figure 1 shows a typical content delivery environment where the replicated Web server clusters are located at the edge of the network to which the end-users are connected. A content provider (i.e. customer) can sign up with a CDN provider for service and have its content placed on the content servers. The content is replicated either on-demand when users request for it, or it can be replicated beforehand, by pushing the content to the surrogate servers. A user is served with the content from the nearby replicated Web server. Thus, the user ends up unknowingly communicating with a replicated CDN server close to it and retrieves files from that server. CDN providers ensure the fast delivery of any digital content. They host third-party content including static content (e.g. static

HTML pages, images, documents, software patches), streaming media (e.g. audio, real time video), User Generated Videos (UGV), and varying content services (e.g. directory service, e-commerce service, file transfer service). The sources of content include large enterprises, Web service providers, media companies and news broadcasters. The end-users can interact with the CDN by specifying the content/service request through cell phone, smart phone/PDA, laptop and desktop. Figure 2 depicts the different content/services served by a CDN provider to end-users.

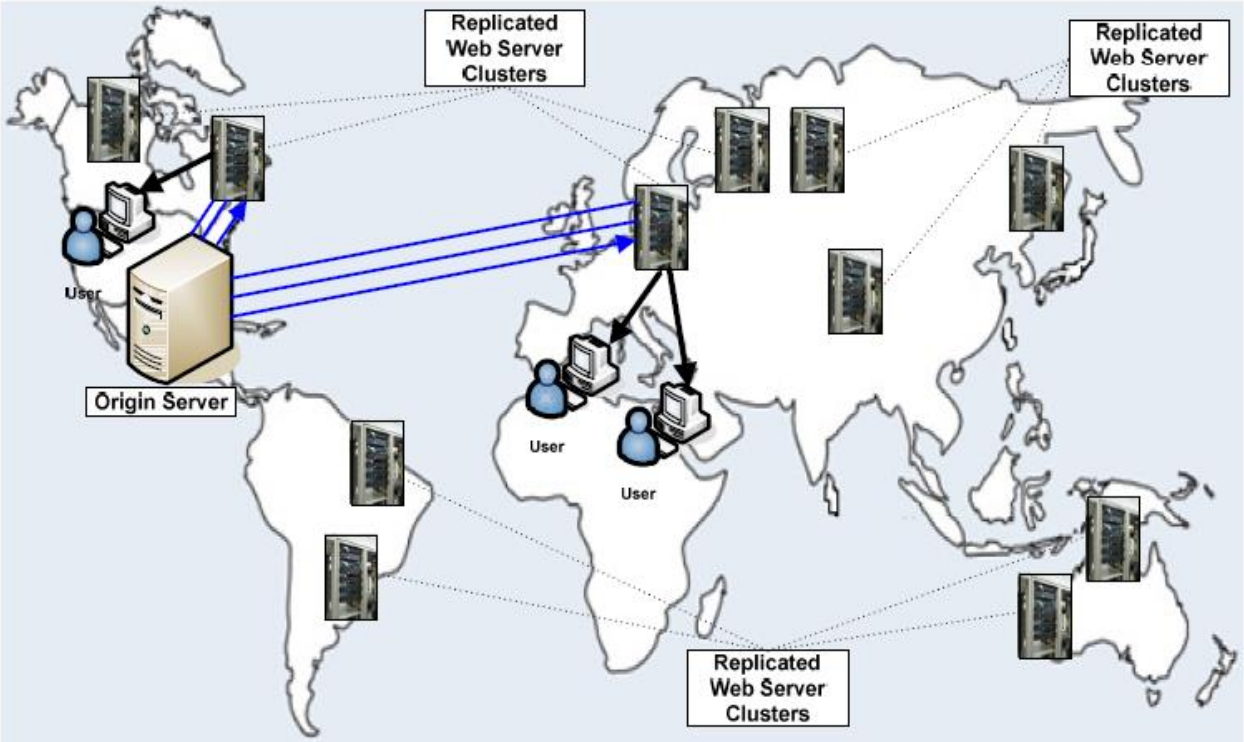


FIGURE 4: ARCHITECTURE OF CDN

1.3 EXISTING CDNS

In this section, we provide a state-of-the-art survey of the existing CDNs. Many commercial CDNs (e.g. Akamai, Adero, Digital Island, Mirror Image, Inktomi, Limelight Networks etc.) as well as academic CDNs (e.g. Coral, Codeen, Globule etc.) are present in the content distribution space. Table 1 shows a list of these CDNs and a brief summary of each of them. Most or all of the operational CDNs are developed by commercial companies which are subject to consolidation over time due to acquisition and/or mergers. Hence, in the survey, we focus on studying those CDNs that have been in stable operation for a significant period of time. In this context, it is worth mentioning that many CDN-specific information such as fees charged by CDNs, existing customers of CDNs are ignored since they are highly likely to change quickly over time. Therefore, the information provided in this section is expected to be stable and up-to-date. We provide a brief discussion on the pricing policies used for CDN services. As mentioned earlier, a CDN charges its customers according to the content delivered (i.e. traffic) to the end-

users by its surrogate servers. There are technical and business challenges in pricing CDN services. However, most of the commercial CDNs do not reveal any information about the strategies used by them to charge their services. In the CDN research community, pricing of CDN services is a relatively new and unexplored area. The use of analytical models to address the optimal prices of CDN services is presented in. It shows that CDN pricing polices should consider providing volume discount to content providers. Such an approach is consistent with current industry practices. It concludes that the recent trends such as decreasing bandwidth cost will have impact on CDN pricing policies and the price of CDN services will decline, while the content delivery process on a Web site will accelerate. In another work Hosanagar et al. demonstrates the importance of realizing an optimal pricing strategy for CDN services under varying traffic patterns, the adoption driver of CDN services, and the drivers of profitability within CDN services. It also reveals that the pure usage based pricing strategy used by most commercial CDNs is suboptimal in cases with varying level of traffic burstiness. Therefore, it concludes that a percentile-based pricing strategy can be used in this case that allows
volume discount for content providers with high mean traffic and also additional charges for content providers with highly bursty traffic.

FIGURE 5: CONTENT/SERVICES PROVIDED BY A CDN

1. Accellion	For Delivery
2. AppStream	Provides on demand software distribution and on demand software license management tools
3. CoDeen	For caching of content and redirection of http requests
4. Coral	Provides content replication
5. Edge Stream	For disrupted video streaming
6. Globix	Provides internet infrastructure and network services
7. Lime light Networks	On demand, disrupted games, and downloads
8. Netli	Provides business quality internet services

TABLE 1: EXISTING CDN SYSTEMS

CHAPTER 2

2.1.1 LITERATURE REVIEW ONE

2.1.1.1 PAPER TITLE

- A Distributed Control Law For Load Balancing In Content Delivery Networks

2.1.1.2 SUMMARY

- This paper deals with the challenging issue of defining and implementing an effective law for load balancing in Content Delivery Networks. They Proposed a novel distributed and time-continuous load balancing algorithm. They say that processing of arriving requests is not continuous over time.

- The algorithm is first introduced in its time-continuous formulation and then put in a discrete version specifically conceived for its actual implementation and deployment in an operational scenario.

- Suggested removal of local queue instability conditions through redistribution of potential excess traffic to the set of neighbours of the congested server.

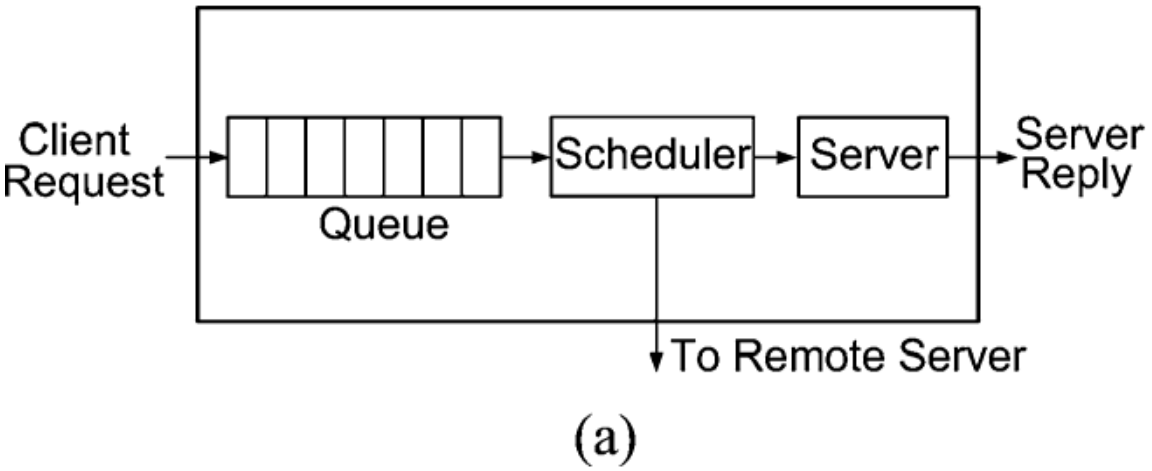


FIGURE 6 : LOAD BALANCING STRATEFY

2.1.2 LITERATURE REVIEW TWO

2.1.2.1 PAPER TITLE

- Optimized Balancing Algorithm For Content Delivery Networks

2.1.2.2 SUMMARY

- In this study the authors present the `fictitiously starred optimised balancing` (FSOB), a novel algorithm for load balancing in a content delivery network (CDN) scenario. FSOB exploits the multiple redirection mechanism of the HTTP protocol to optimally redistribute clients requests among the servers which build up the CDN.

- Load redistribution is aimed at equalizing the level of occupancy of the server queues and is achieved through the periodical exchange of information computed locally at each node. The algorithm initially makes a fictitious assumption about the local topology of the network, as it is seen by each single server node, which looks at itself as the centre (i.e. the master) of a star made up of all of its neighbors (i.e. the slaves). Load redistribution is performed by the master which, if needed, appropriately redirects incoming requests to its slaves.

2.1.3 LITERATURE REVIEW THREE

2.1.3.1PAPER TITLE

- An Efficient Distributed Control law for Effective Load Balancing in Content Delivery Network

2.1.3.2 SUMMARY

- In this paper, authors face the challenging issue of defining and implementing an effective law for load balancing in Content Delivery Networks.
- They first design a suitable load-balancing law that assures equilibrium of the queues in a balanced CDN by using a fluid flow model for the network of servers. Then they present a new mechanism for redirecting incoming client requests to the most appropriate server, thus balancing the overall system requests load.
- The mechanism leverages local balancing in order to achieve global balancing. This is carried out through a periodic interaction among the system nodes.
- This result is then leveraged in order to devise a novel distributed and time-continuous algorithm for load balancing, which is also reformulated in a time-discrete version.

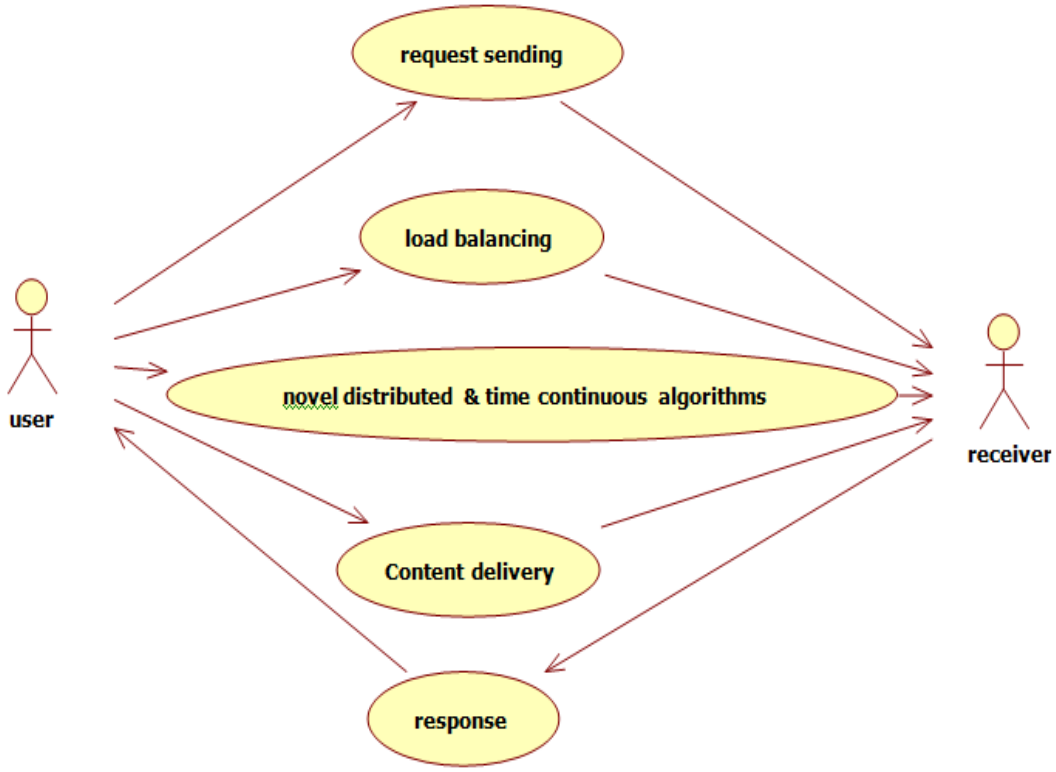


FIGURE 7: USE CASE DIAGRAM

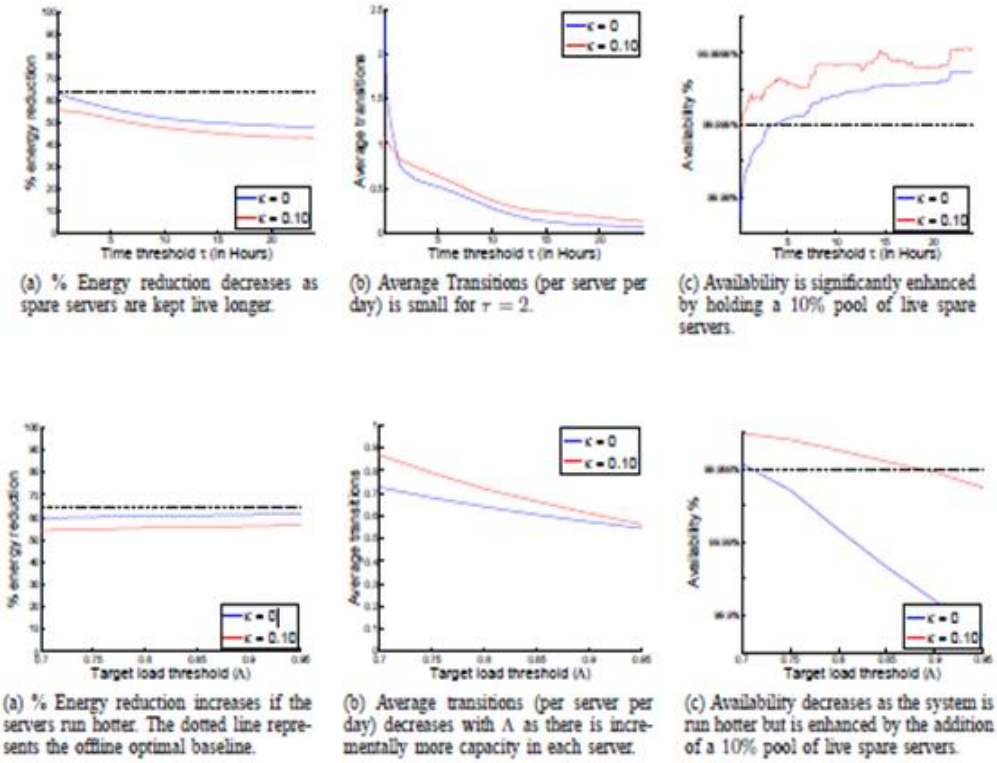
2.1.4 LITERATURE REVIEW FOUR

2.1.4.1 PAPER TITLE

- Energy-Aware Load Balancing in Content Delivery Networks

2.1.4.2 SUMMARY

- Internet-scale distributed systems such as content delivery networks (CDNs) operate hundreds of thousands of servers deployed in thousands of data center locations around the globe. Since the energy costs of operating such a large IT infrastructure are a significant fraction of the total operating costs, the authors argue for redesigning CDNs to incorporate energy optimizations as a first-order principle.



(a) % Energy reduction decreases as spare servers are kept live longer.

(b) Average Transitions (per server per day) is small for $\tau = 2$.

(c) Availability is significantly enhanced by holding a 10% pool of live spare servers.

(a) % Energy reduction increases if the servers run hotter. The dotted line represents the offline optimal baseline.

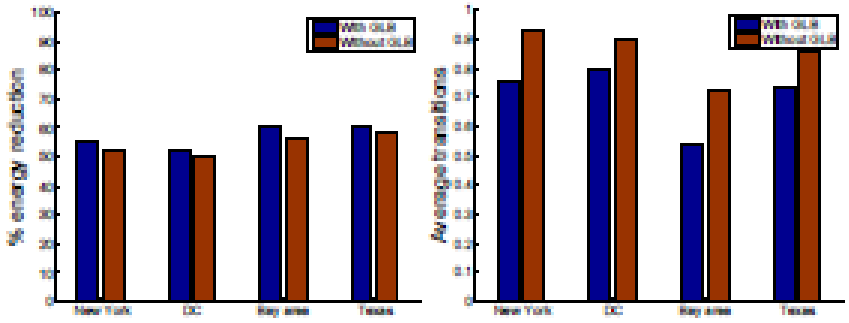
(b) Average transitions (per server per day) decreases with Λ as there is incrementally more capacity in each server.

(c) Availability decreases as the system is run hotter but is enhanced by the addition of a 10% pool of live spare servers.

FIGURE 8 : ANALYSIS OF THE VARIATION OF THREE METRICS, AS DISCUSSED IN THE PAPER

- They have proposed techniques to turn off CDN servers during periods of low load while seeking to balance three key design goals: maximize energy reduction, minimize the impact on client-perceived service availability (SLAs), and limit the frequency of on-off server transitions to reduce wear-and-tear and its impact on hardware reliability.

- They propose an optimal offline algorithm and an online algorithm to extract energy savings both at the level of local load balancing within a data center and global load balancing across data centers. We evaluate our algorithms using real production workload traces from a large commercial CDN.



(a) % Energy reduction (b) Average Transitions (per server per day)

FIGURE 9: RESULTS OF THE PROPOSED ALGORITHM

2.1.4 LITERATURE REVIEW FIVE

2.1.5.1 PAPER TITLE

- Efficient and Distributed Control Mechanism for load Handling in Content Distributed Network

2.1.5.2 SUMMARY

- A Content distribution network is an effective solution to the emerging Web applications. Unfortunately it also faces a higher risk of degradation in overall performance of entire distributed network when high number of request arrives from client flash crowd. In this research paper the authors propose an efficient control law for handle the load on individual servers by using efficient request routing mechanism which can handle worst case scenario in the existing system.

- This paper discusses the importance of handling load using cooperative control law of surrogate servers. The simulation results show that the proposed system can successfully handle the all draw back that can be faced in existing system.

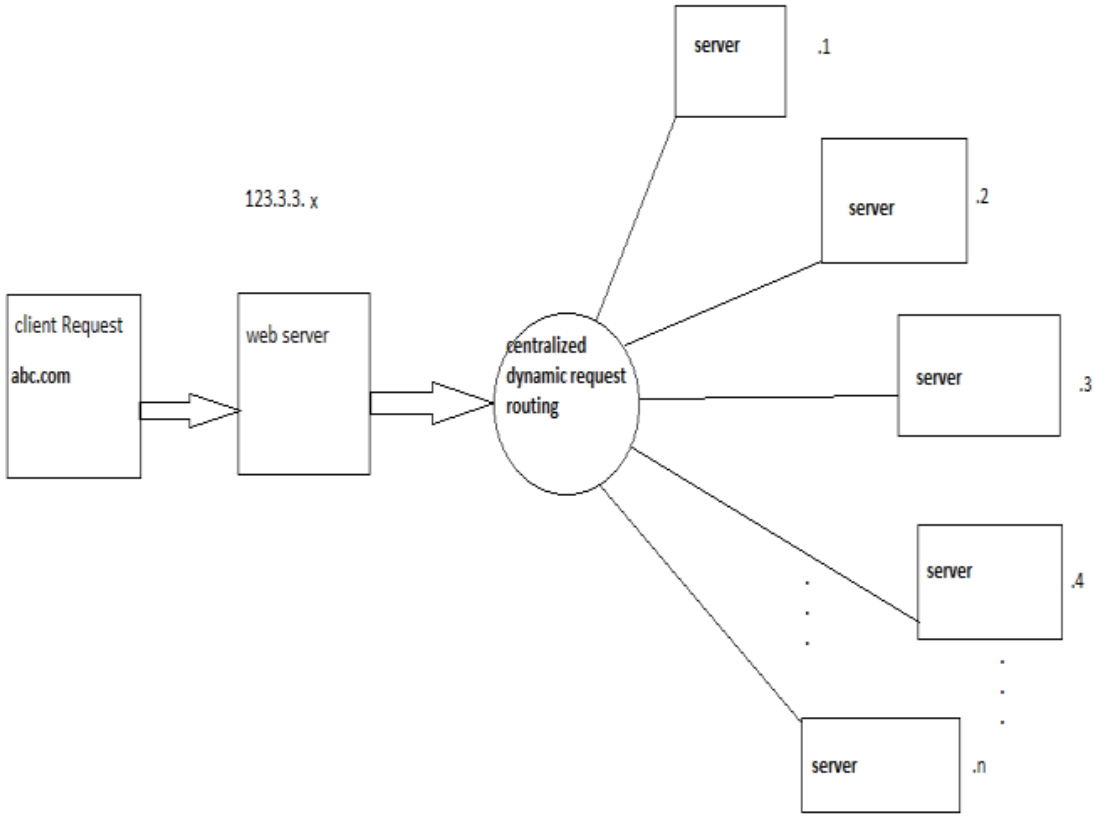


FIGURE 10 : DEPICTING A DYNAMIC APPROACH TO LOAD BALANCING

2.1.4 LITERATURE REVIEW SIX

2.1.6 PAPER TITLE

- Implementation of Effective Law for Load Balancing

2.1.6.1 SUMMARY

- This paper discusses about the implementation of an effective law for load balancing Content Delivery Networks (CDN). In this method we will only have the hardware implementation costs that will be giving the output of the program without software implementations costs.

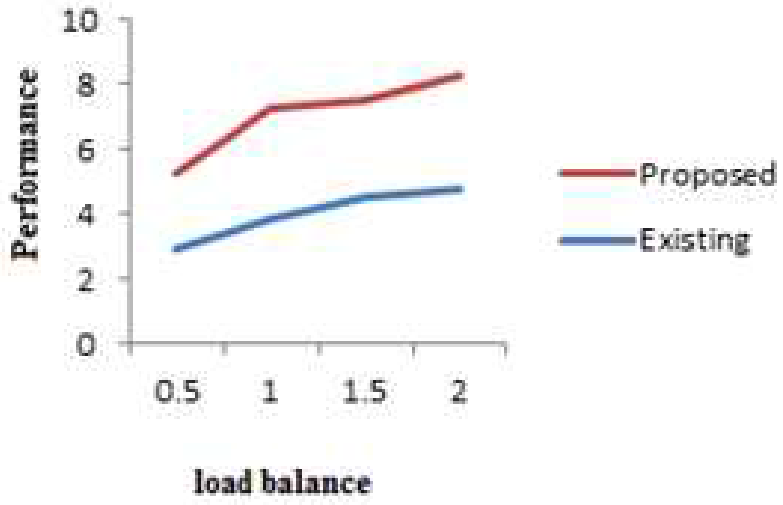


FIGURE 11: COMPARISON GRAPH

- So for these requirements, the paper introduces the hardware driven flow slicing along with software driven load balancer which also reduce implementation costs when compared to the existing system method.

2.1.7 LITERATURE REVIEW SEVEN

2.1.7 PAPER TITLE

- A Scalable Approach for Effective Content Delivery Using Enhanced Distributed Load Balancing Mechanism

2.1.7.1 SUMMARY

- In the paper, the authors base their proposal on a formal study of a CDN system, carried out through the exploitation of a fluid flow model characterization of the network of servers.

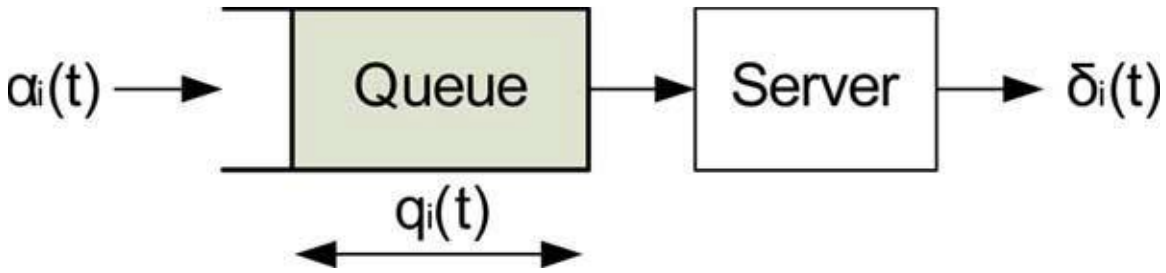


FIGURE 12 : FLUID QUEUE MODEL

- They have derived and prove a lemma about the network queues equilibrium. This result is then lever- aged in order to devise a novel distributed and time-continuous algorithm for load balancing, which is also reformulated in a time-discrete version. The discrete formulation of the proposed balancing law is eventually discussed in terms of its actual implementation in a real-world scenario.

2.1.4 LITERATURE REVIEW EIGHT

2.1.8 PAPER TITLE

- Applying Load Balancing: A Dynamic Approach

2.1.8.1 SUMMARY

- In this paper, the authors have proposed a Dynamic approach to the load balancing algorithm. The paper discusses the implementation of application load balancer by using Dynamic load balancing method. load balancing is performed by integrating more than two physical servers with logical load balancing.

- Dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they use current or recent load information when making distribution decisions. As a result, dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. However, this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits

- The paper investigates on the comparative behavior of load balancing with different parameters, and concludes that dynamic load balancing is more reliable.

2.1.4 LITERATURE REVIEW NINE

2.1.9 PAPER TITLE

- Network delay-aware load balancing in selfish and cooperative distributed systems.

2.1.9.1 SUMMARY

- In this paper the authors consider a geographically distributed request processing system composed of various organizations and their servers connected by the Internet. The latency a user observes is a sum of communication delays and the time needed to handle the request on a server. The handling time depends on the server congestion, i.e. the total number of requests a server must handle. The paper focuses on the analyis of the problem of balancing the load in a network of servers in order to minimize the total observed latency.

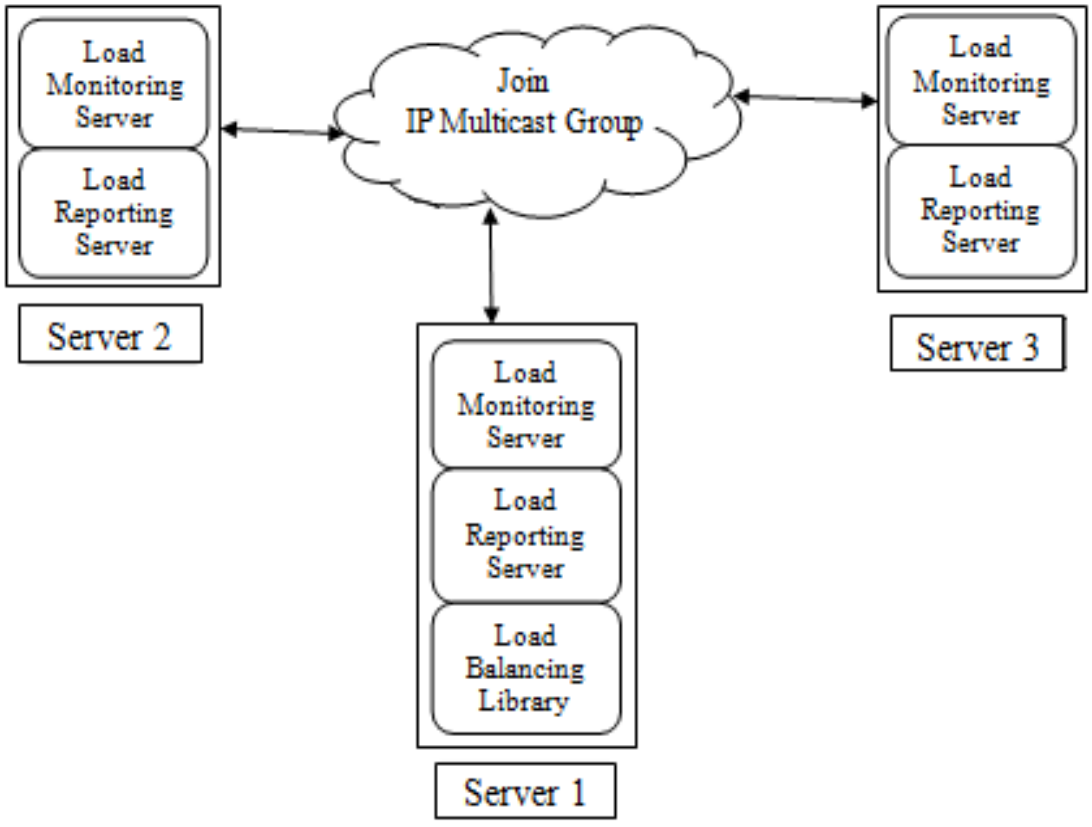


FIGURE 13: DEPLOYMENT SCENARIO

2.2 COMPARATIVE STUDY

The following section contains the comparative study of the literature reviews mentioned in the above section. In order to summarize the important aspects of the literature reviews, the following table is presented. Proposal enumerates the topic of concern in the research paper. The heading improvements elaborates on the suggestions made in the research papers.

PAPER ID	PAPER TOPIC	PROPOSAL	IMPROVEMENT
06176282	A Distributed Control Law For Load Balancing In Content Delivery Networks	A novel distributed and time-continuous load balancing algorithm through derivation and proof of network queue equilibrium lemma.	Removal of local queue instability conditions through redistribution of potential excess traffic to the set of neighbors of the congested server
12783915	Optimized Balancing Algorithm For Content Delivery Networks	Fictitiously starred optimized balancing(FSOB).	Optimizing the load redistribution system, by utilizing the multiple redirection mechanism of the HTTP protocol.
EL3425142521	An Efficient Distributed Control law for Effective Load Balancing in Content Delivery Network	A novel distributed and time continuous algorithm for load balancing, reformulated in a time discrete version	Removal of local queue instability conditions through redistribution of potential excess traffic to the set of neighbours of the congested server
1109.5641	Energy-Aware Load Balancing in Content Delivery Networks	1. Techniques to turn off CDN servers during period of low load.	Improved upon reducing the energy consumption by more than 55% while ensuring high level of availability that meets

		2. An optimal online algorithm and an online algorithm to extract energy savings both at the level of local load balancing within a data centre and global load balancing across data centres	customer's service availability requirements.
IJARCCE9A	Efficient and Distributed Control Mechanism for load Handling in Content Distributed Network	Efficient control law for handling the load on individual servers.	Overall performance in terms of availability, response time and queue length handling
3	Implementation of Efective Law for Load Balancing	Hardware driven flow slicing along with software driven load balancer	Implementation cost
400_1688	A Scalable Approach for Effective Content Delivery Using Enhanced Distributed Load Balancing Mechanism	A novel distributed and time continuous algorithm for load balancing	Removal of local queue instability, improved response time
contentDeliver y_hcw2013	Network delay-aware load balancing in selfish and cooperative distributed systems	Distributed algorithm iteratively balancing the load	Efficiency
V2I600231	International Journal of Advanced Research in	Implementation of application load	Efficiency and performance

	Computer Science and Software Engineering	balancer by using dynamic load balancing method	

TABLE 2: COMPARATIVE STUDY

CHAPTER 3

3.1 PROBLEM STATEMENT

3.1.1 EXISTING LOAD BALANCING ALGORITHMS

3.1.1.1 RANDOM BALANCING MECHANISM

- Traffic is directed arbitrarily to any server in your farm. In a random Scheduling, the requests are assigned to any server picked randomly among the group of servers.
- Pros: Random Scheduling load balancing algorithm is simple to implement.
- Cons: It can lead to overloading of one server while under-utilization of others.

3.1.1.2 ROUND ROBIN ALGORITHM

- Robin Scheduling Algorithm is that the IP sprayer assigns the requests to a list of the servers on a rotating basis. For the subsequent requests, the IP sprayer follows the circular order to redirect the request. Once a server is assigned a request, the server is moved to the end of the list. This keeps the servers equally assigned.
- Pros: Better than random allocation because the requests are equally divided among the available servers in an orderly fashion.
- Cons: Not enough for load balancing based on processing overhead required and if the server specifications are not identical to each other in the server group.

3.1.1.3 LEAST LOADED ALGORITHM

- Least Loaded is a well-known dynamic strategy for load balancing. It assigns the incoming client request to the currently least loaded server.
- Pros: Adopted in several commercial solutions.
- Cons: It tends to rapidly saturate the least loaded server until a new message is propagated.

3.2 ISSUES OF EXISTING LOAD BALANCING ALGORITHMS

Response Time

- Response time is the time it takes for the server to cater to the requests of the user. Performance of a CDN is typically characterized by the response time perceived by the end-users.
- Although there are many other factors to be taken into consideration for building an effective load balancing algorithm, the scale is tipped in favor of the response time being the most coveted one. Slow response time is the single greatest contributor to customers' abandoning Web sites and processes.

Local Queue Instability

- In a Content Delivery Network, each server has its own queue, for maintain the requests according to a predetermined order. The stability of the network depends on the stability of the queues. Condition for stability is a function of length of the queue.

Data Transfer Rate

- Data transfer rate is the rate at which data is transferred from the server to the user and vice versa. The faster the data transfer rate, better would be the experience of the user. The CDNs strive to improve upon this rate, through improving upon the response time.

Implementation Cost

- This is another important factor in rendering the load balancing algorithms. As we know to more and more servers are being deployed to make the process faster, the implementation cost keeps on increasing with each new server.

3.3 PROPOSED MODEL

To overcome the drawbacks of the existing algorithms, and to take into account various issues listed in the previous section, we propose a load balancing algorithm.

Our aim is to design a deadline based approach for a load balancing algorithm which would take into account various issues and present a solution for these issues with the help of comparative study of the existing algorithms. The algorithm is based on the following parameters

1. Request Rate
2. Process Rate
3. Queue length
4. Deadline
5. Network Delay

Request Rate: The rate at which requests are to be received. The server receiving more requests in unit amount of time would have high request rate, where as server receiving less requests in unit amount of time would have less request rate. The server with the least request rate would be preferred over the server with comparatively more request rate, while redirection of the requests.

Process Rate: Process rate is the ability of the server to process as many requests as possible in unit amount of time. The higher the process rate, the better. While redirection, the server with high process rate would be preferred over the one with less request rate.

Queue Length: Each server maintains a queue, for the requests to be queue in, while the server is handling some other requests. If the queue length of the server is full, the request must be redirected to another server, whose queue has space to accommodate new requests.

Deadline: The deadline of the request would also be taken into consideration while determining the load balancing algorithm. If the request's deadline is about to expire, it would be given top most priority, and the queue would reshuffle itself, to place it at the front position.

Network Delay: The delay between the two servers should also be taken into account while designing an efficient load balancing algorithm. The lesser the delay, the better.

3.4 FLOW DIAGRAM

Given below is the flow diagram of the load balancing algorithm

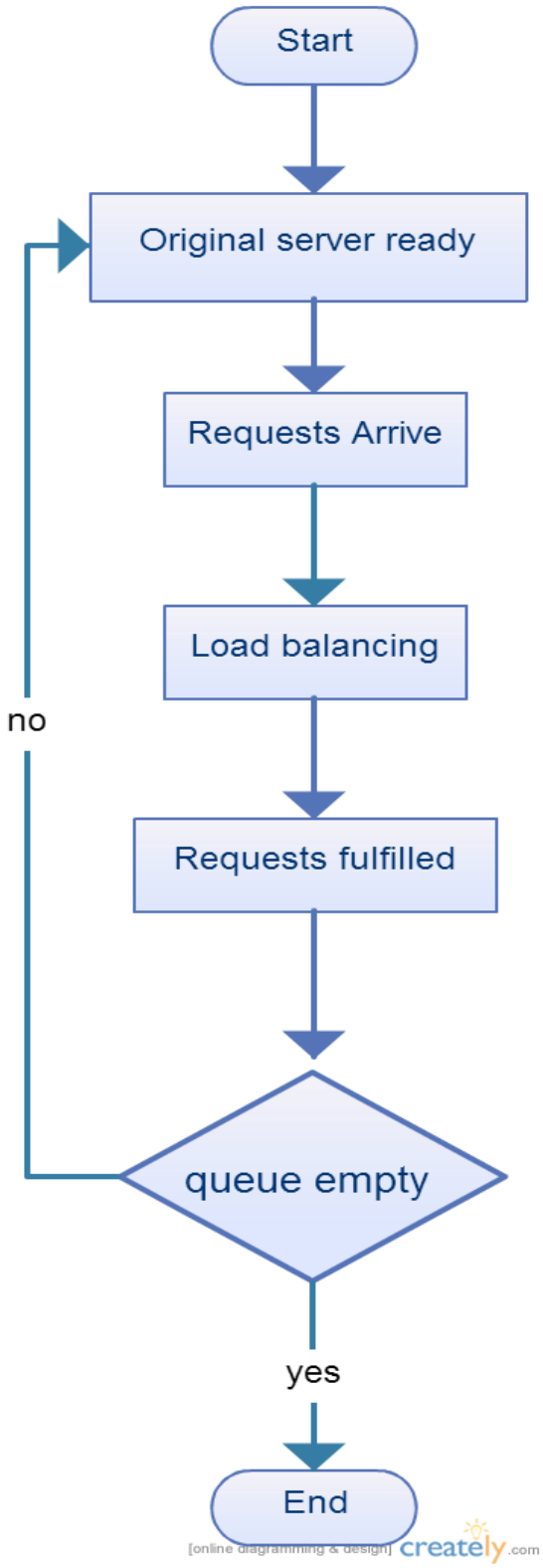


FIGURE 14: FLOW DIAGRAM

Description of flow diagram :

The flow diagram aims to show the work flow of the load balancing algorithm. It is explained with the help of following steps.

Step 1. First of all we check if servers are ready to receive the requests.

Step 2: When the servers are ready the requests start to arrive.

Step 3: After the arrival of the requests, the requests are queued in for load balancing.

Step 4: Once the requests have been dealt with, they are diverted to their respective servers, which are then responsible for processing those requests

Step 5: After all the requests in the queue of the serve have been fulfilled, the server becomes ready to receive more requests.

Step 5: The whole process again is repeated in a loop.

Step 6: In case there are no requests to be fulfilled, we can halt the load balancing for the time.

3.5 SYSTEM ARCHITECTURE

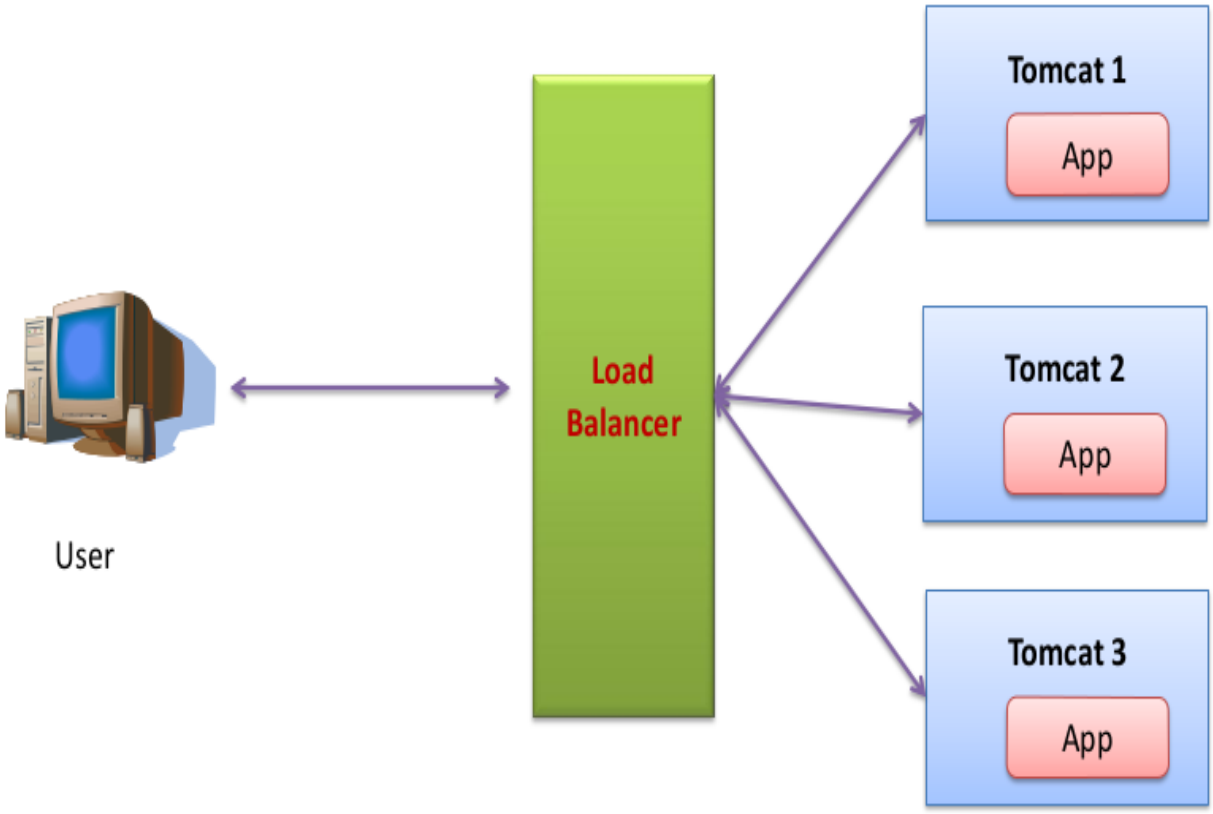


FIGURE 15: SYSTEM ARCHITECTURE

3.6 PSEUDO CODE

The following section gives the pseudo code, consisting of two functions, schedule, and load balance.

Description :

The function FindServer() is depicted in thefollowing figure, from line 1 to 12.

This function finds the first available server, to push the requests into the queue of the server. First of all it checks whether the queue of the server is empty or not. On finding the queue empty, it will put the request in the server's queue. In case it encounters that the queue of the server is full, then it would move on to the next server, and again in the similar fashion , as stated above.

```
1.  find server()  // To find the available server
2.  {
3.      if (queue length == full)  // check to see whether the queue is full
4.      {
5.          findanotherserver(); // if yes, find another server
6.          loadbalance();
7.      }
8.      else
9.      {
10.         schedule(); //if not, put the request in the queue
11.     }
12. }
```

FIGURE 16: PSEUDO CODE

Description :

The Function LoadBalance() is given in the figure, shown below, from line 16 to 30.

This functions primarily aims to balance the load of the requests among the servers. After the server with empty queue is selected from the findserver() function, this function springs into action. It again checks for the queue to empty. That again has two cases:

1. When the queue is empty, then it increments the queue counter, i.e. top, and inserts the request in the respective serve's queue.

2. In case the server's queue is not empty, the request would be waitlisted.

```
13. function loadbalance() // to balance the load among the servers
14. {
15.     findserver() // to find the first available server
16.     {
17.             if(queue==empty) // check whether the queue is empty
18.             {
19.                     top++;  // if yes, increment the counter
20.                     insert_request(); // insert the request
21.             }
22.             else  // if not,
23.             {
24.                     top++; // increment without inserting
25.             }
26.     }
27. }
```

FIGURE 17: PSEUDO CODE

CHAPTER 4

4.1 IMPLEMENTATION

4.1.1 SPECIFICATIONS OF THE SYSTEM USED

- Operating System: Windows 8
- RAM : 16GB
- Processor: i7
- IDE: Netbeans 7.4, Java sdk 1.7.9
- Glassfish

Description:

Here we have generated 20 requests, with queue length of the each server being constant, the requests are scheduled on the first come first serve basis, in the respective server's queue.

```
run:
How many requests to be generated?
20
Generating requests...
request generated for server 4
Request 0 inserted in the queue
request generated for server 4
Request 1 inserted in the queue
request generated for server 2
Request 2 inserted in the queue
request generated for server 3
Request 3 inserted in the queue
request generated for server 3
Request 4 inserted in the queue
request generated for server 1
Request 5 inserted in the queue
request generated for server 4
Request 6 inserted in the queue
request generated for server 3
Request 7 inserted in the queue
request generated for server 2
Request 8 inserted in the queue
request generated for server 1
Request 9 inserted in the queue
request generated for server 3
Request 10 inserted in the queue
request generated for server 2
```

FIGURE 18: SCREENSHOT 1

Description: When the queue of the server is full, the rest of the requests are waitlisted, and the message "Server busy, request in waiting state " is displayed.

```
Request 11 inserted in the queue
request generated for server 3
Request 12 inserted in the queue
request generated for server 3
Request 13 inserted in the queue
request generated for server 3
Server busy, request in waiting state
request generated for server 1
Request 15 inserted in the queue
request generated for server 3
Server busy, request in waiting state
request generated for server 3
Server busy, request in waiting state
request generated for server 4
Request 18 inserted in the queue
request generated for server 3
Server busy, request in waiting state
BUILD SUCCESSFUL (total time: 7 seconds)
```

FIGURE 19: SCREENSHOT 2

4.1.2 EMULATION

Description: Screen shot of the emulation home page. The user sends the request by clicking on the submit button. The request is directed to one of the servers.

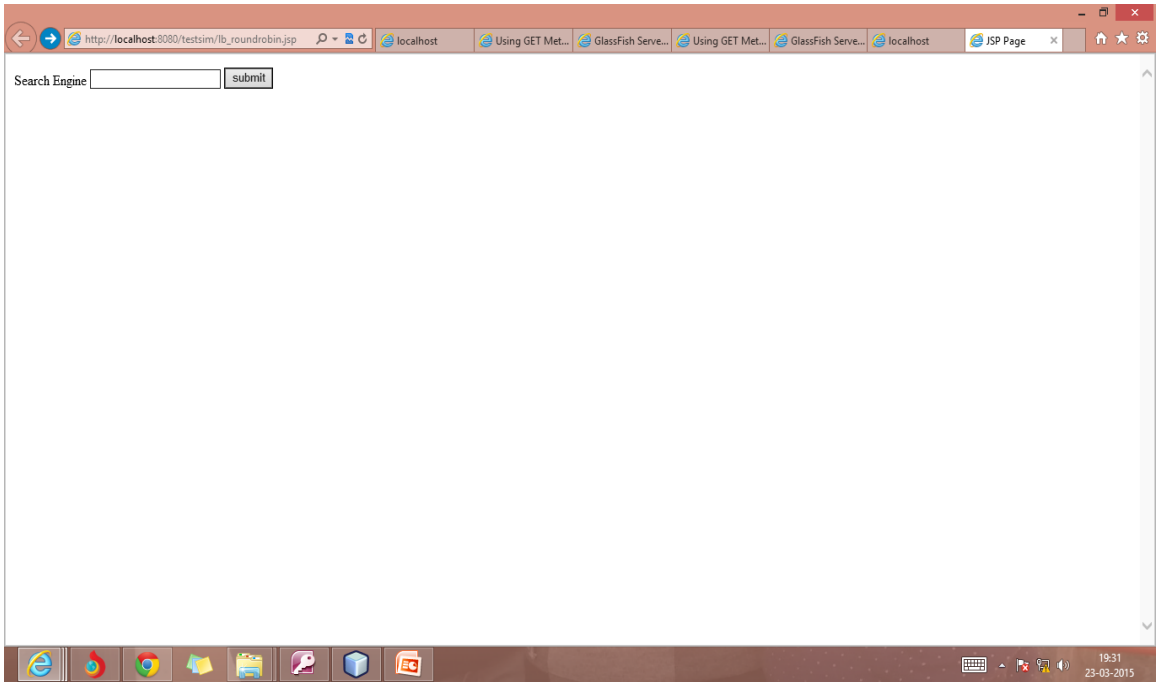


FIGURE 20: HOME PAGE

Description: Here the request is directed to server 1 by the load balancer. The request's processing time and Id is displayed.

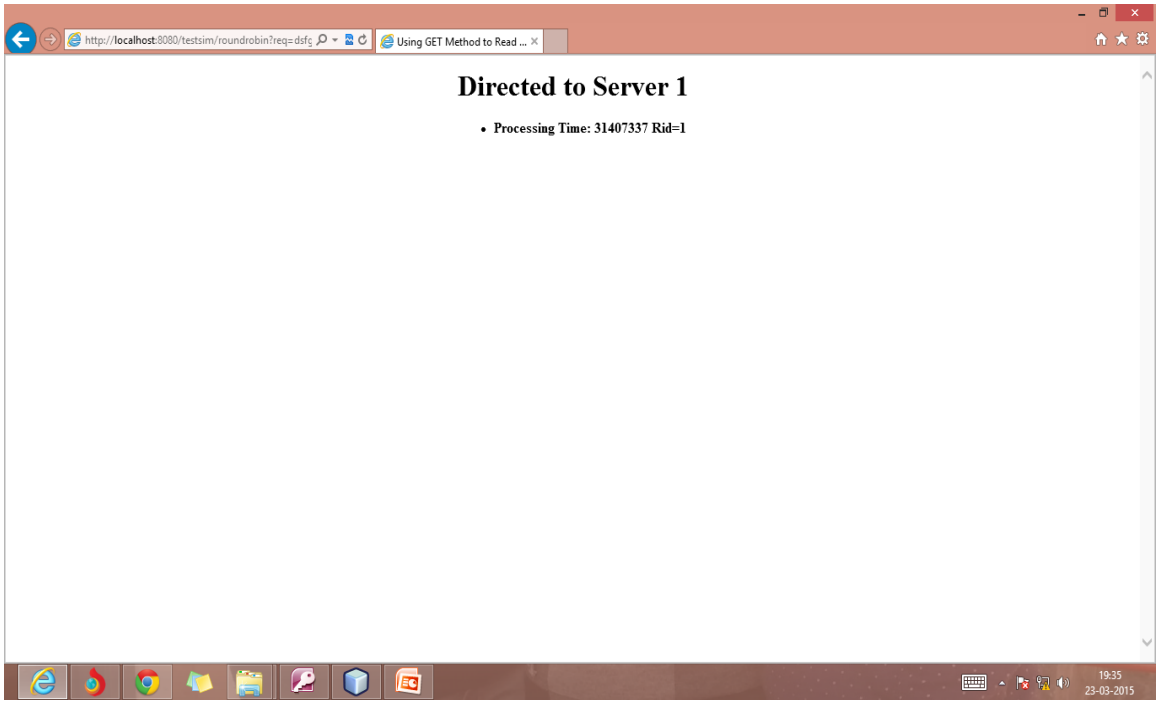


FIGURE 21: : DIRECTED TO SERVER 1

Description : The following is a screenshot of the request being diverted to the server 2.

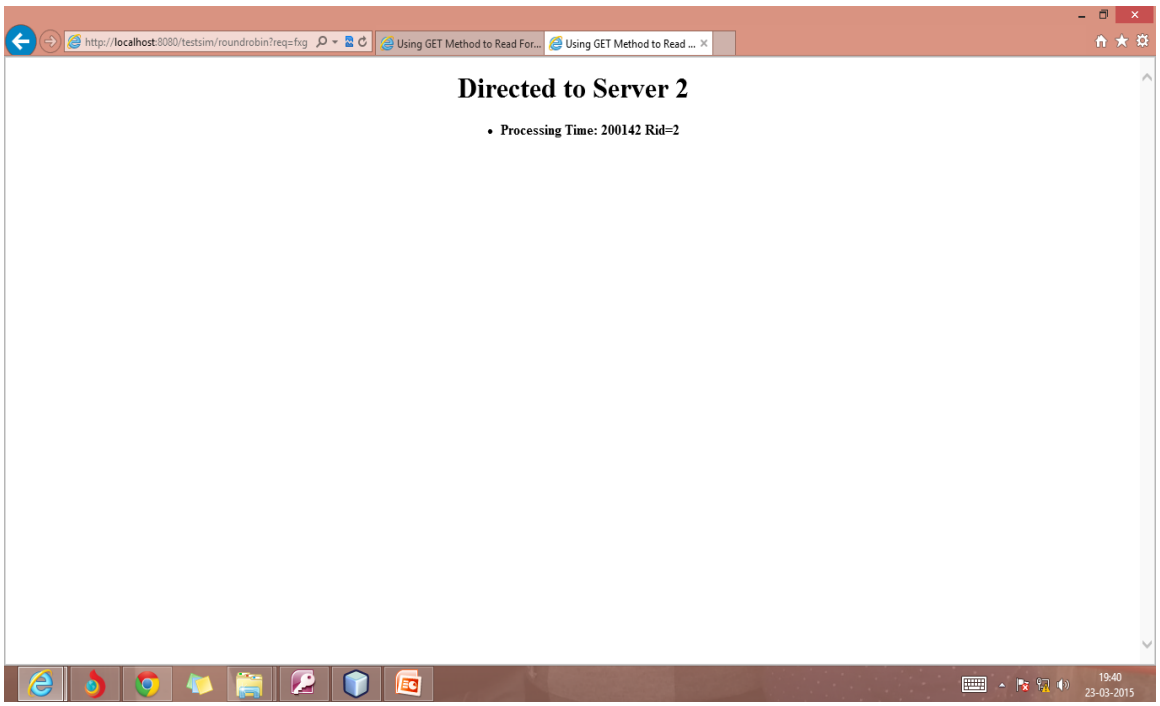


FIGURE 22: DIRECTED TO SERVER 2

Description: The following is a screenshot of request being diverted to the server 3.

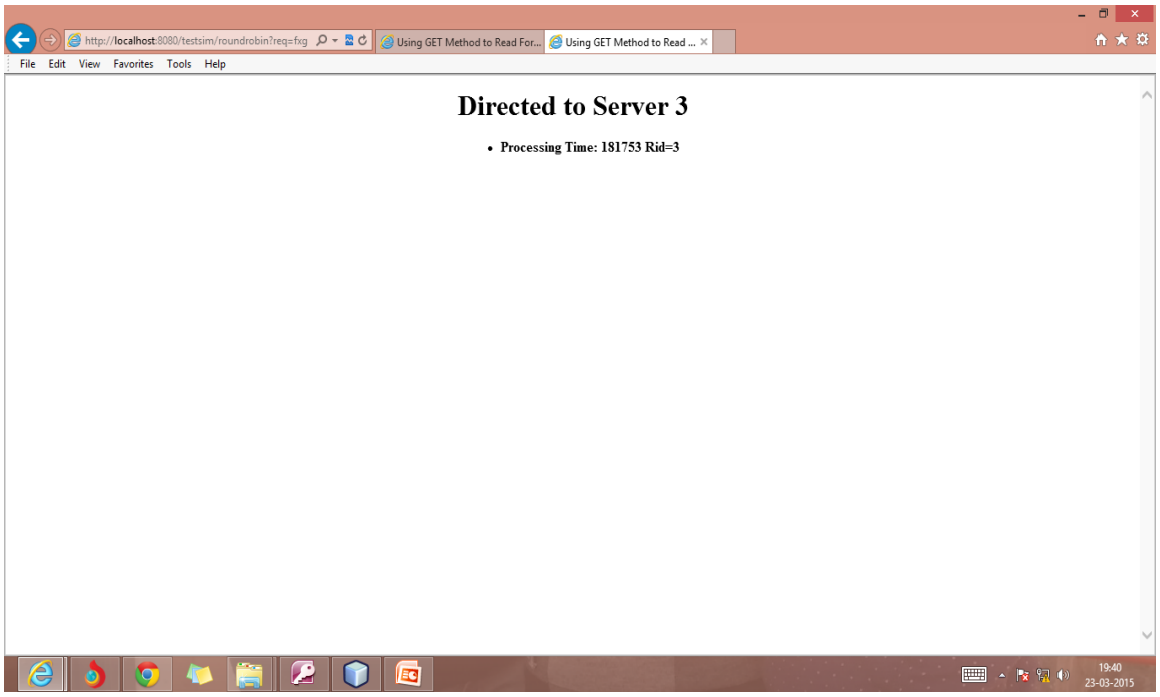


FIGURE 23: DIRECTED TO SERVER 3

Description: The following figures shows the log report of 100 requests being generated and their processing time by each server.

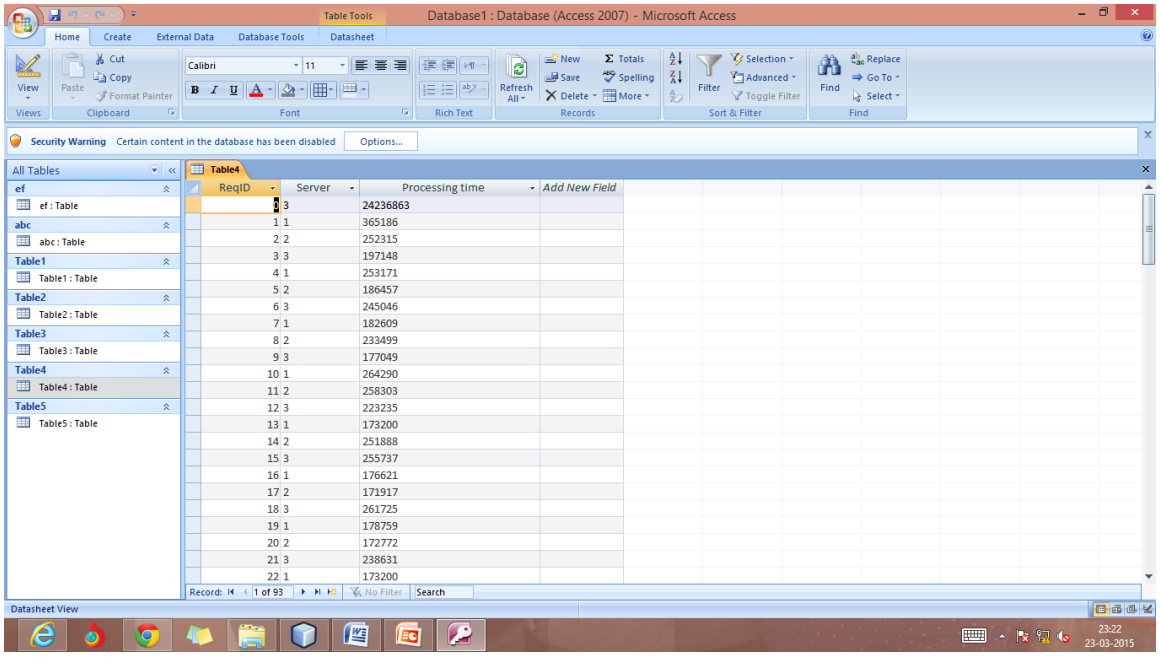


FIGURE 24: DATABASE RECORD 1

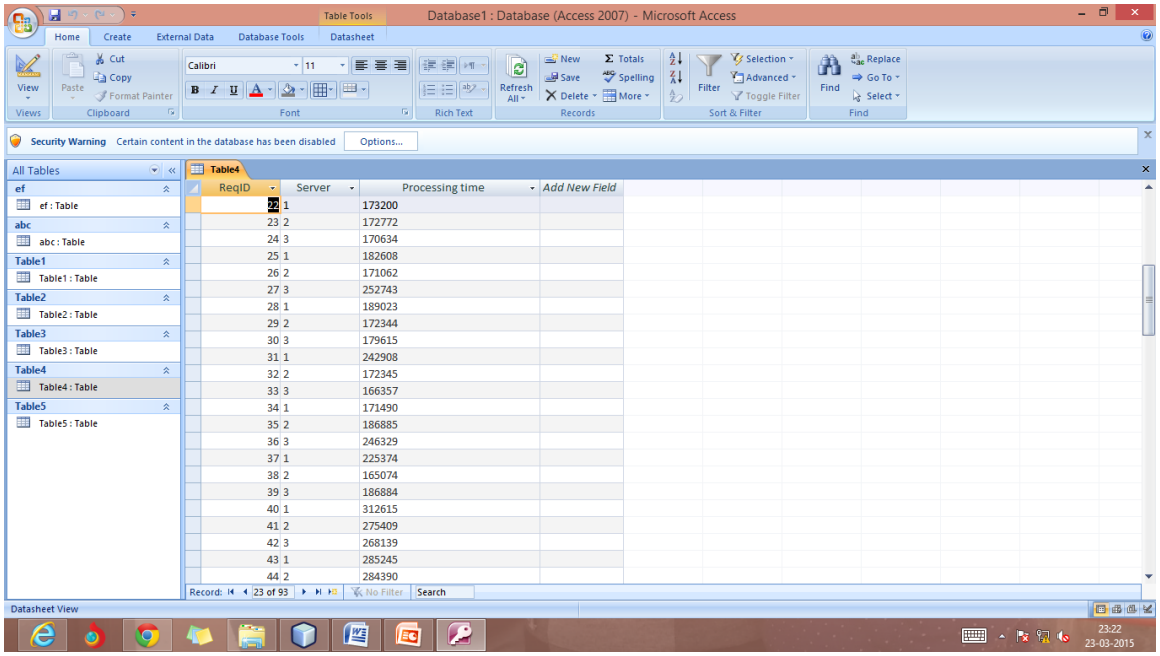


FIGURE 25 : DATABASE RECORD 2

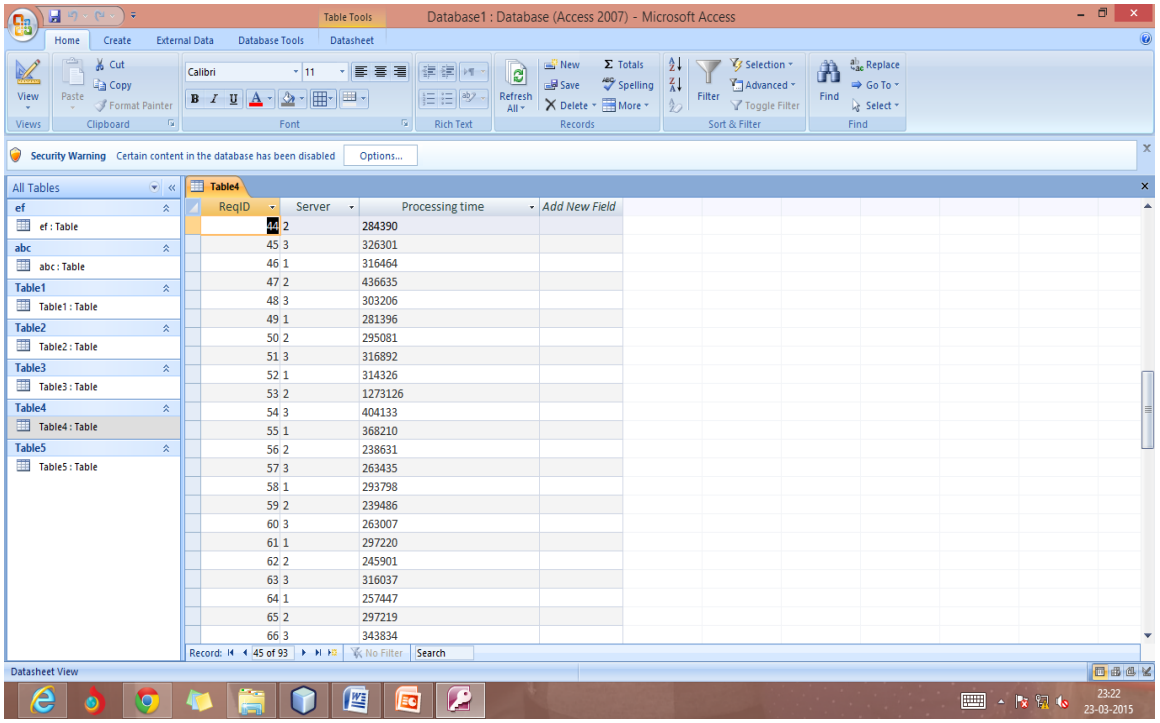


FIGURE 26: DATABASE RECORD 3

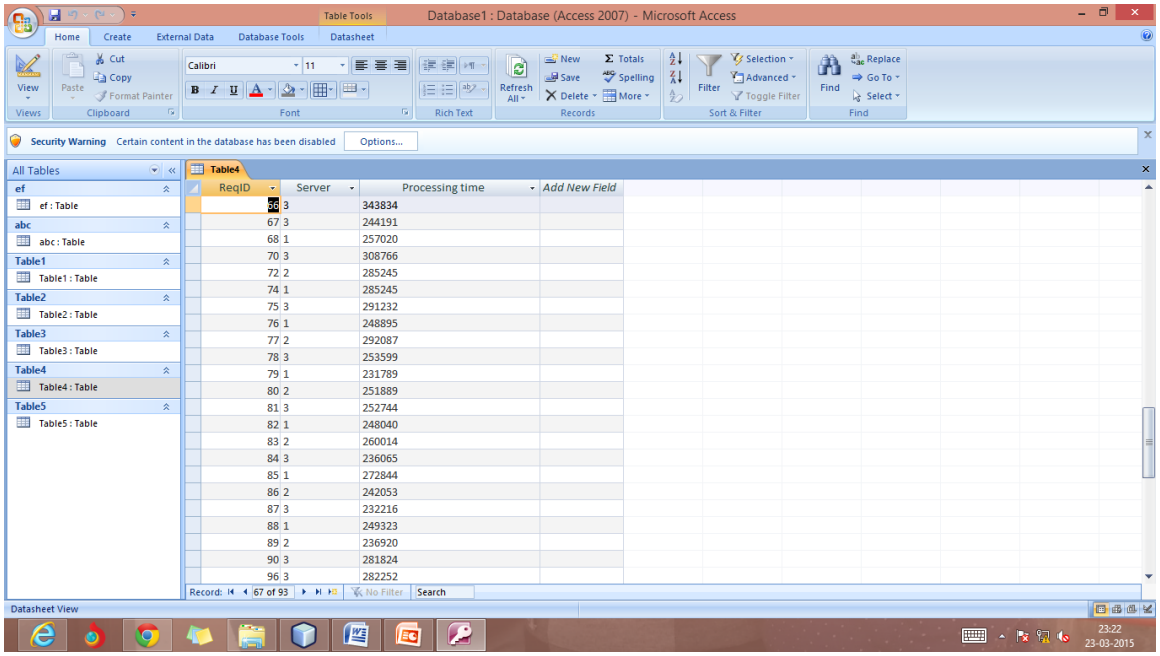


FIGURE 27: DATABASE RECORD 4

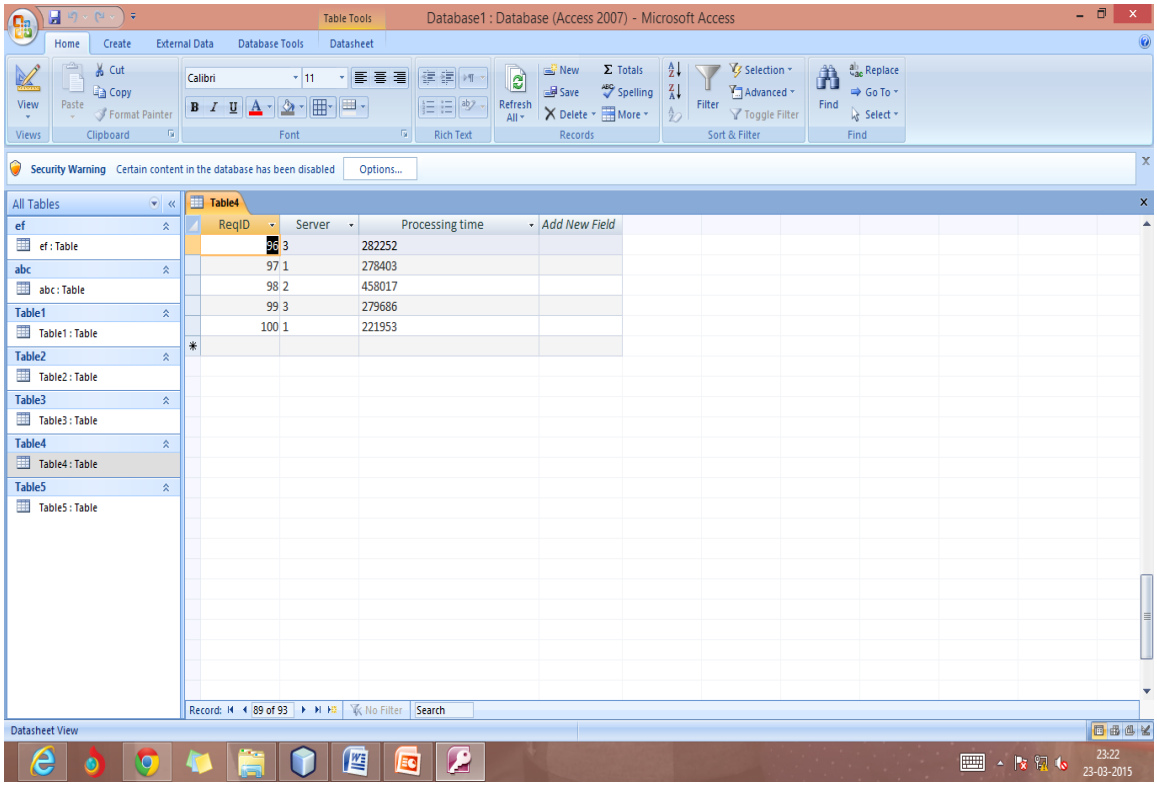


FIGURE 28: DATABASE RECORD 5

4.1.3 SIMULATION

4.1.3.1 FIRST SCENARIO

Here we are assuming three servers for experiment purpose. The constraints are:

- The processing time for 3 servers is as follows:
  Server1 : 1000 ms
  Server2 : 2000 ms
  Server3 : 3000 ms
- Requests generated for all the servers is 20.
- ID the following figure denotes the unique request ID for the particular request.
- Each server has also been assigned a unique id, denoted by server ID in the following table.
- The processing time for each request has been recorded.

Algorithm Tested: Round Robin, i.e. The requests are distributed to the servers in a round robin fashion. First request being sent to the very first server available, the second being sent to the next, and third to one after that. After the sequential distribution of requests to -all the available servers, it comes back to the very first server.

ID	Server ID	Processing Time
0	1	1.01E+09
1	2	2.01E+09
2	3	3.01E+09
3	1	2.32E+09
4	2	4.09E+09
5	3	6.09E+09
6	1	3.4E+09

7	2	6.17E+09
8	3	9.17E+09
9	1	4.48E+09
10	2	8.23E+09
11	3	1.22E+10
12	1	5.56E+09
13	2	1.03E+10
14	3	1.53E+10
15	1	6.64E+09
16	2	1.24E+10
17	3	1.84E+10
18	1	7.72E+09
19	2	1.45E+10

TABLE 3: FIRST SCENARIO

4.1.3.2 SECOND SCENARIO

Keeping the constraints of the first scenario constant, i.e. 3 servers, with respective processing time, and 20 requests are generated.

However now we test it on different algorithm. This algorithm is based on the random approach. Here the requests are distributed to the server in a random fashion. Any request can be diverted to any server, randomly, irrespective of queue length, processing time etc.

ID	Server ID	Processing Time
0	2	2.005E+09
1	2	4.103E+09
2	3	3.005E+09
3	2	6.18E+09
4	3	6.081E+09
5	2	8.256E+09
6	2	1.033E+10
7	1	1.003E+09
8	3	9.159E+09
9	1	2.327E+09
10	2	1.241E+10
11	2	1.449E+10
12	1	3.409E+09
13	1	4.484E+09
14	1	5.564E+09
15	3	1.224E+10
16	1	6.644E+09
17	3	1.531E+10
18	2	1.657E+10
19	3	1.839E+10

TABLE 4:SECOND SCENARIO

4.1.3..3 THIRD SCENARIO

Again, keeping the constraints of the previous two scenario constant, we now come to our most efficient algorithm. Least loaded algorithm, which diverts the requests to the server based on the queue length of the server. The server having least queue length is sought, and the request is send to that server, so as to minimize the delay and increase efficiency.

The following table shows the details of the requests, server and the processing time.

ID	Server ID	Processing Time
0	1	1.006E+09
1	2	2.007E+09
2	3	3.005E+09
3	1	2.289E+09
4	2	4.09E+09
5	3	6.083E+09
6	1	3.387E+09
7	2	6.174E+09
8	3	9.163E+09
9	1	4.467E+09
10	2	8.252E+09
11	3	1.224E+10
12	1	5.544E+09
13	2	1.034E+10
14	3	1.532E+10
15	1	6.623E+09
16	2	1.241E+10
17	3	1.84E+10
18	1	7.709E+09
19	2	1.449E+10

TABLE 5: THIRD SCENARIO

Now, we compare the results of the above three algorithms, for 20 requests, on 3 servers and analyze their performance. The following graph shows the comparison based on the processing time.

Least Loaded (in nano seconds)	Round Robin (in nano seconds)	Random (in nano seconds)
1999237493	1007936399	2005070692
4092742675	2008345236	4102759649
6186337237	3011954206	3004646031
2998390739	2321348050	6179696281
998335145	4088624408	6081437259
8263795571	6090383349	8255553512
6091678245	3402917027	10332727490
2278873355	6166358613	1003135130
10364978419	9168905719	9159266830
9193254534	4480416450	2327092297
3372394360	8230883447	12410025916
12473721327	12246174209	14490273014
12286108827	5558984151	3408631765
4480847747	10308266121	4483578522
14590507543	15337937169	5563823054
15403267531	6637815288	12237281148
5589609043	12403783026	6644413986
16684163259	18413450995	15314368312
18508445946	7719400943	16568492606
6682991061	14479549168	18390276436

TABLE 6: COMPARISON

With the help of the above table, following graph was generated for 20 Requests, which were directed to 3 servers

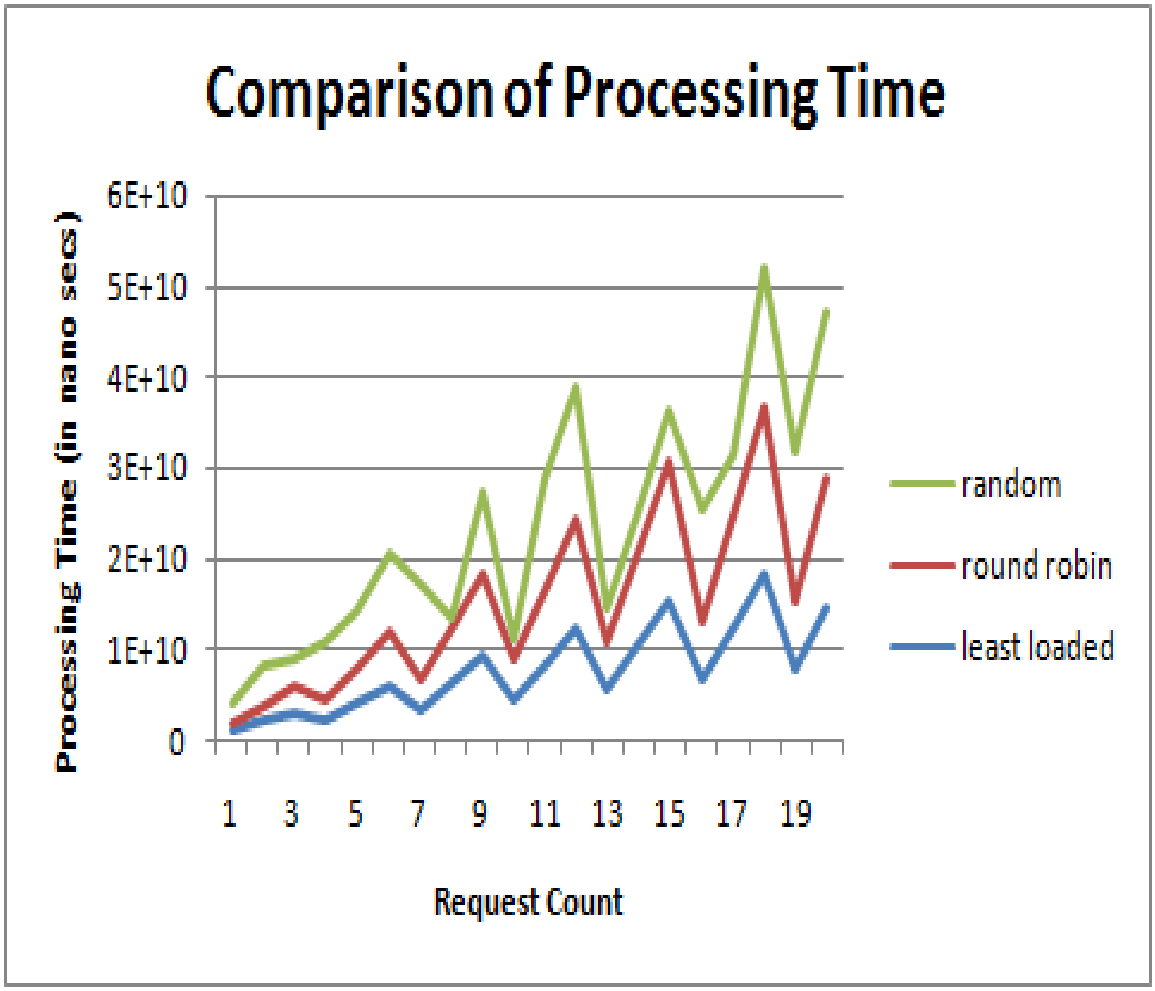


FIGURE 29: COMPARATIVE GRAPH GENERATED FOR 20 REQUESTS

CPU Utilization for 20 requests in the three cases is given by the following data

	Least Loaded	Round Robin	Random
CPU Utilization(%)	34.1861	37.7613	31.89

TABLE 7:CPU UTILIZATION OF 20 REQUESTS

Graph for CPU Utilization for 20 requests is give as

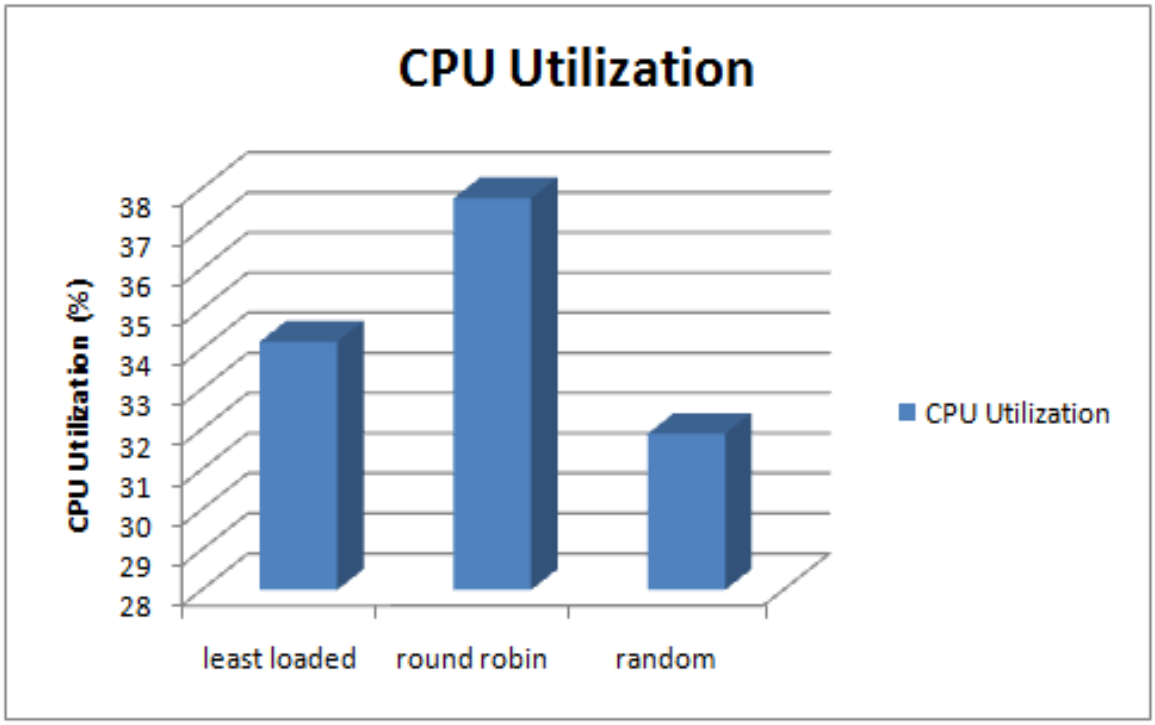


FIGURE 30: CPU UTILIZATION GRAPH

Performing the similar procedure, for 30 requests, with 3 servers having respective processing times for the three algorithms, ie.e random, round robin, and least loaded, we get the following figures:

Plotting the processing time on the y-axis, and requests ids on x

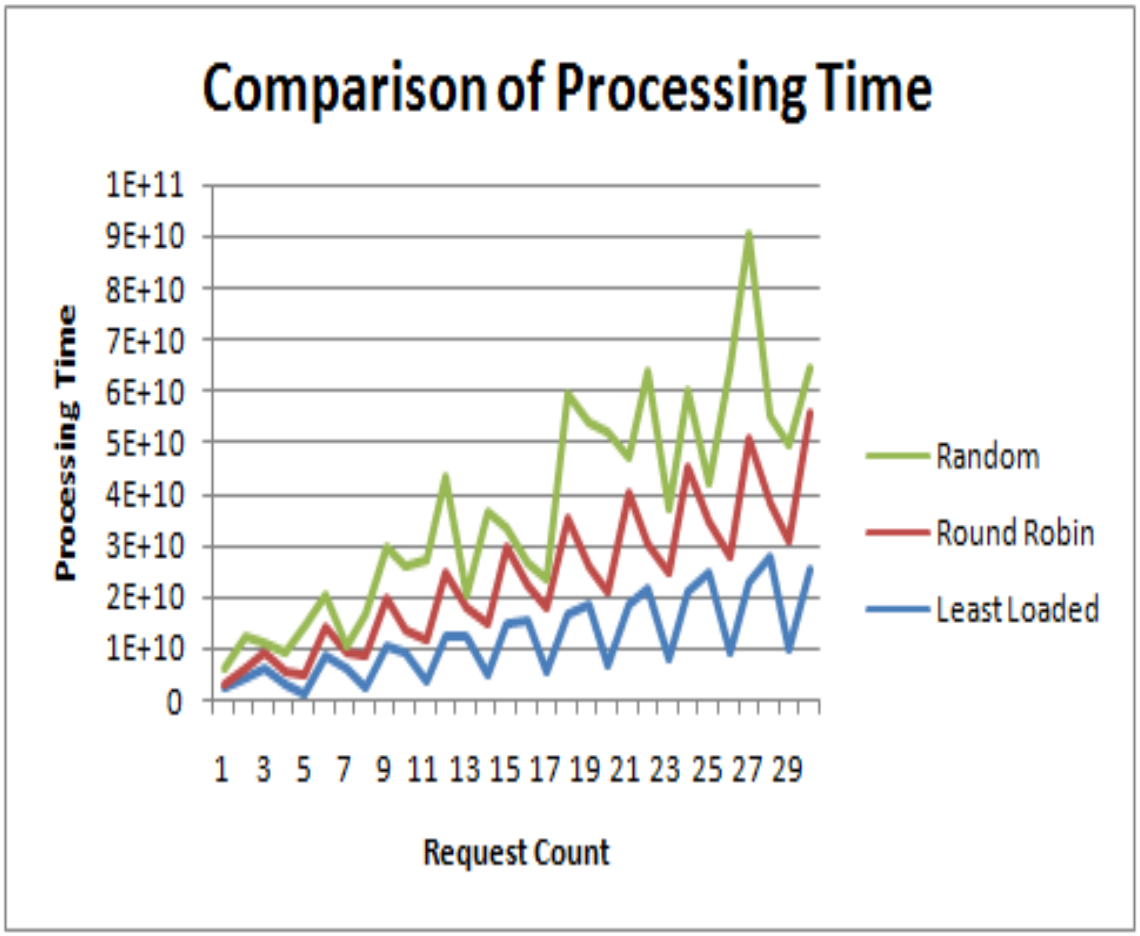


FIGURE 31:COMPARISON GRAPH FOR 30 REQUESTS

CPU Utilization for 30 requests in the three cases is given by the following data

	Least Loaded	Round Robin	Random
CPU Utilization(%)	25.3374	26.6992	28.6699

TABLE 8: CPU UTILIZATION OF 30N REQUESTS

CPU Utilization for 30 requests in the three cases is given by the following data

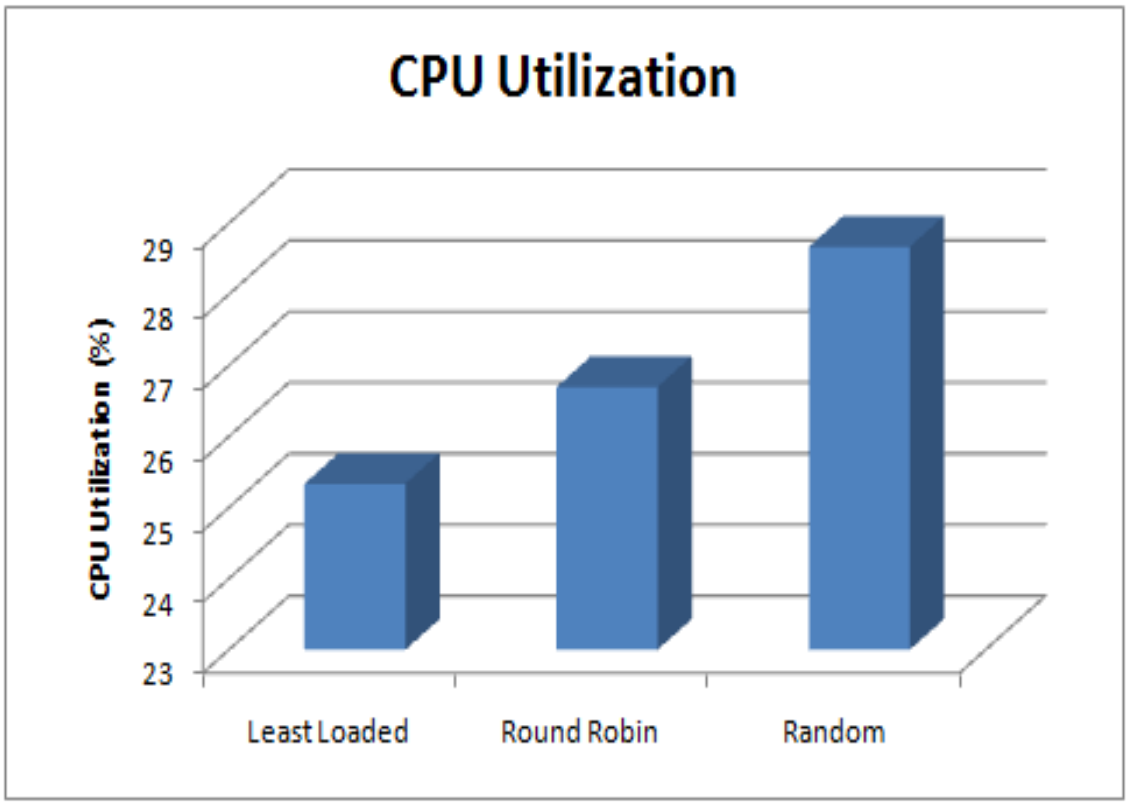


FIGURE 32:CPU UTILIZATION GRAPH

Similarly for 40 requests, we generate the following graph.

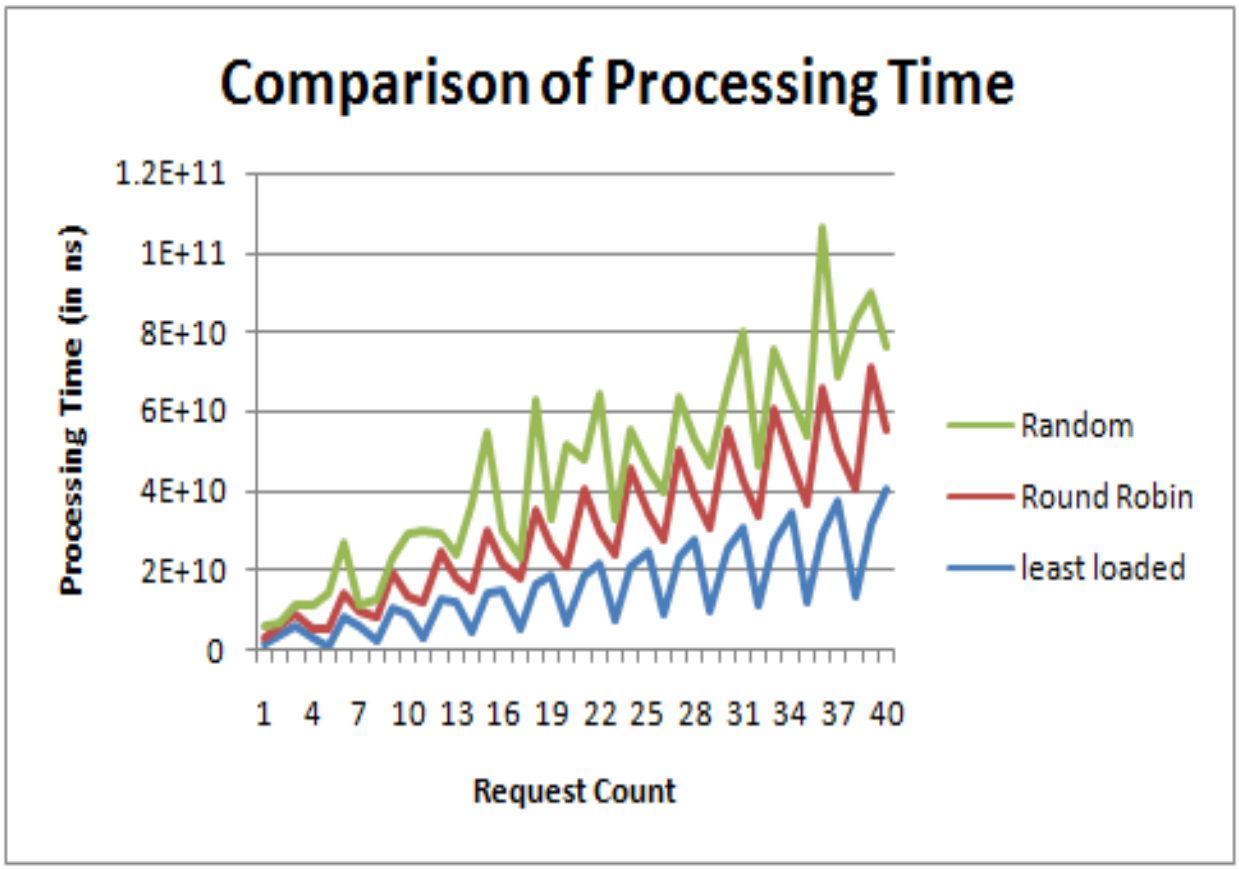


FIGURE 33:COMPARISON GRAPH FOR 40 REQUESTS

CPU Utilization for 40 requests in the three cases is given by the following data

	Least Loaded	Round Robin	Random
CPU Utilization(%)	15.9083074	20.15253913	19.30233

TABLE 9 : CPU UTILIZATION OF 40 REQUESTS

Graph for CPU Utilization for 40 Requests is give as :

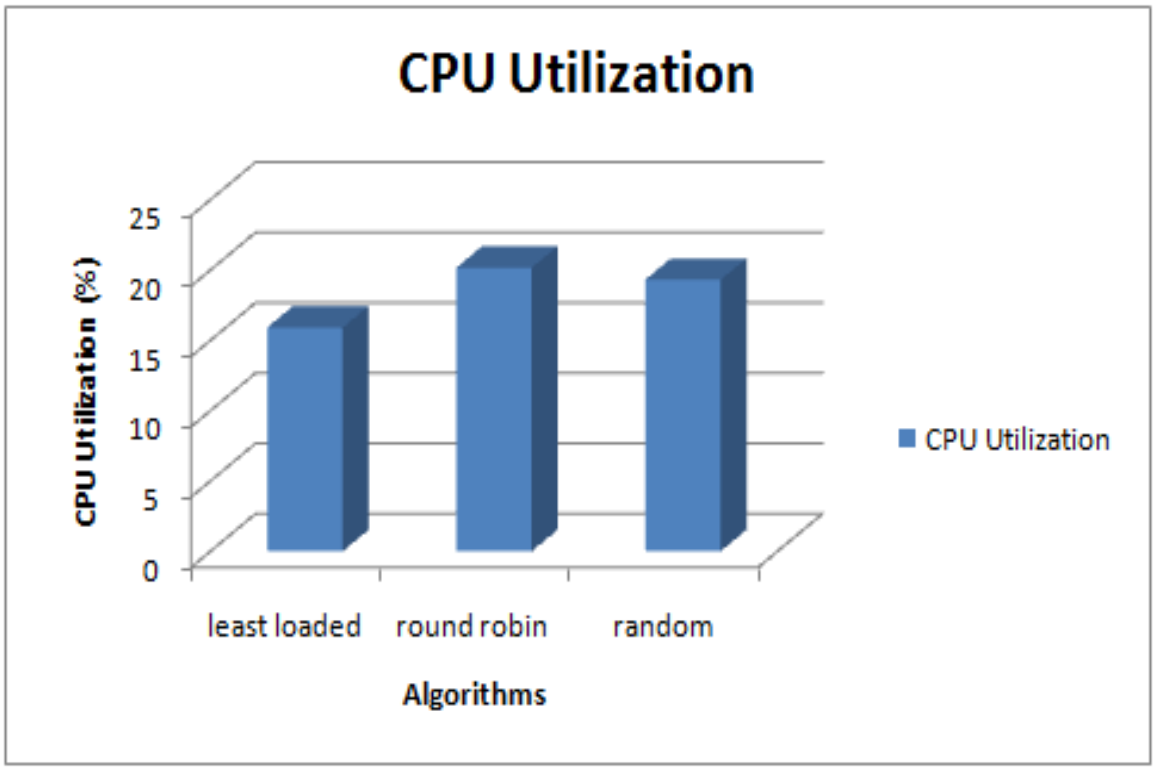


FIGURE 34:CPU UTILIZATION

Performing the same for 50 requests, we get the following graph.

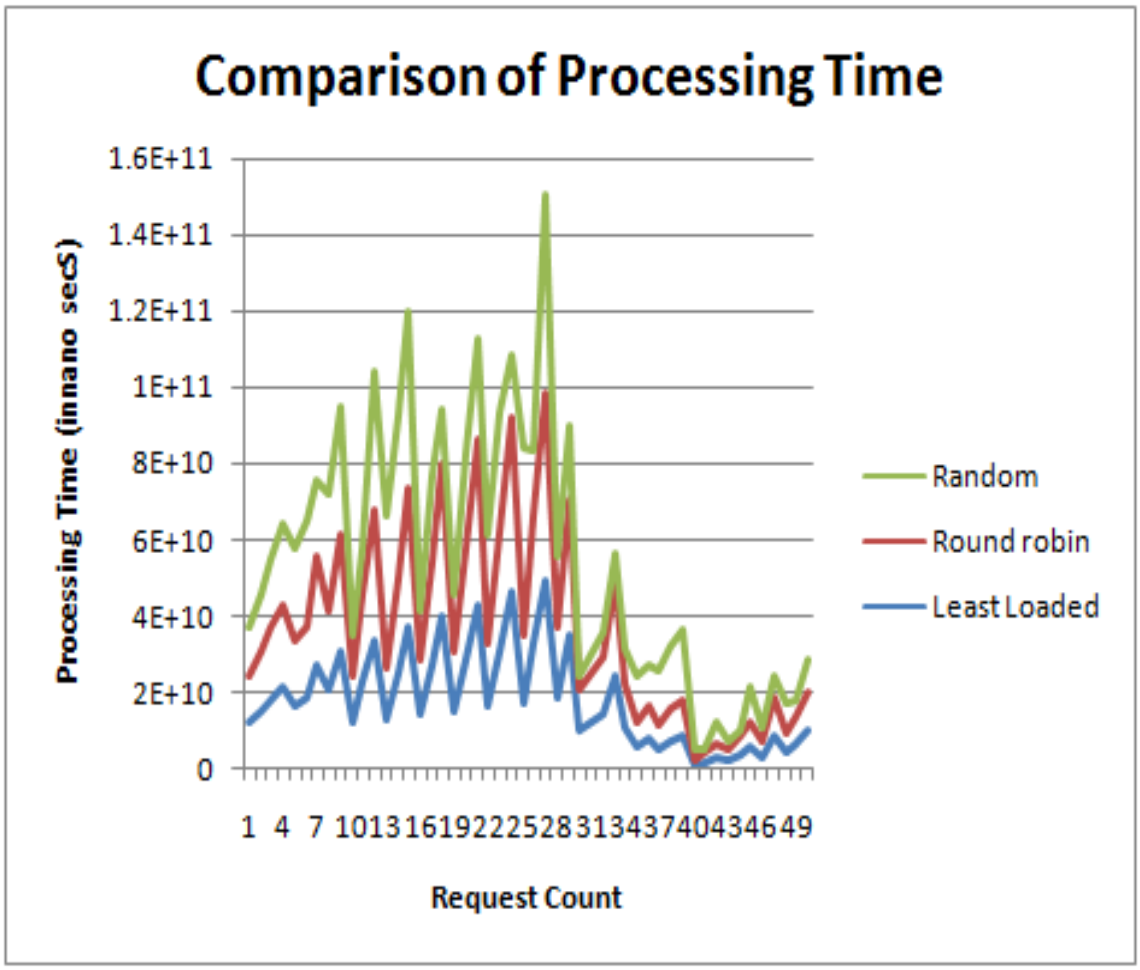


FIGURE 35:COMPARISON GRAPH FOR 50 REQUESTS

CPU Utilization for 40 requests in the three cases is given by the following data

	Least Loaded	Round Robin	Random
CPU Utilization (%)	16.6896	16.9858	14.2009

TABLE 10 : CPU UTILIZATION OF 50 REQUESTS

Graph for CPU Utilization for 50 requests is given as :

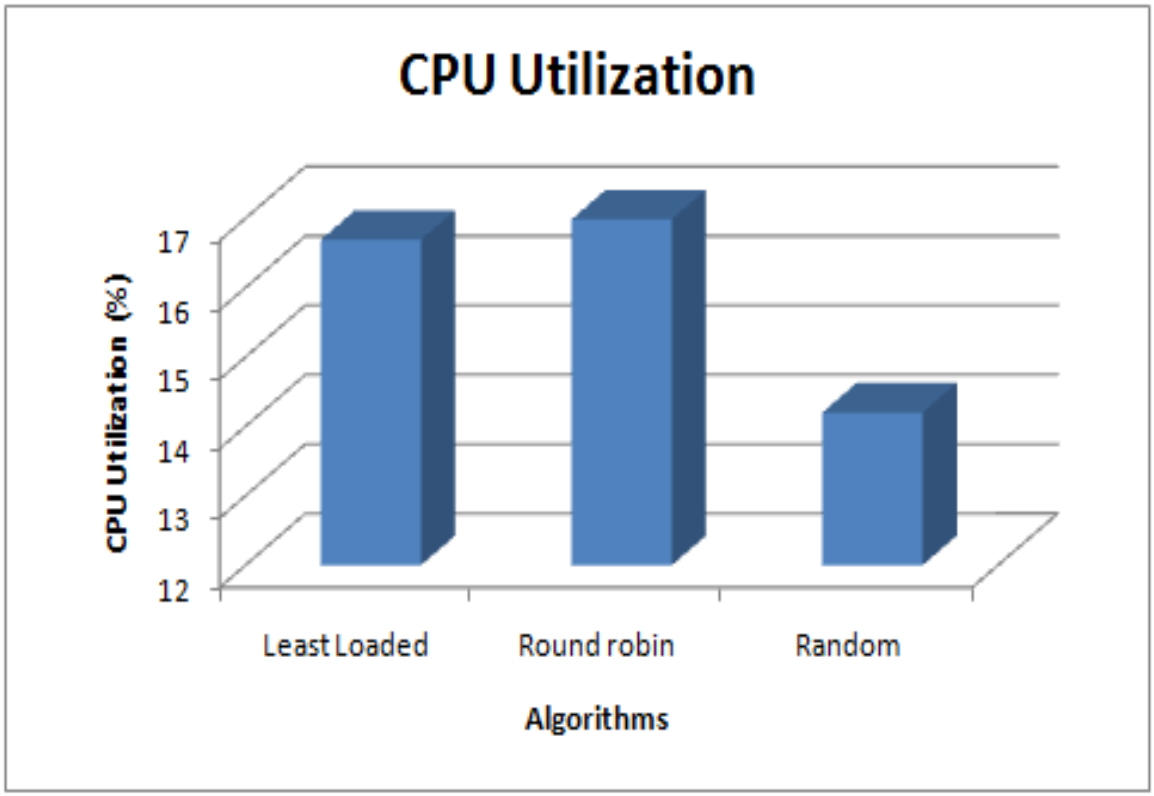


FIGURE 36:CPU UTILIZATION GRAPH

4.1.4 COMPREHENSIVE STUDY

The following section shoes the comprehensive study of the three algorithms.

The following figure depicts the CPU utilization of three algorithms, with different number of requests.

Requests -->	20	30	40	50
Least Loaded (%)	34.181	25.3374	20.12478	17.12195
Round Robin(%)	37.7613	26.6992	20.1525	16.9858
Random(%)	31.89	28.6699	19.3023	14.2009

TABLE 11 : COMPREHENSIVE CPU UTILIZATION

The following graph was generated using the above table.

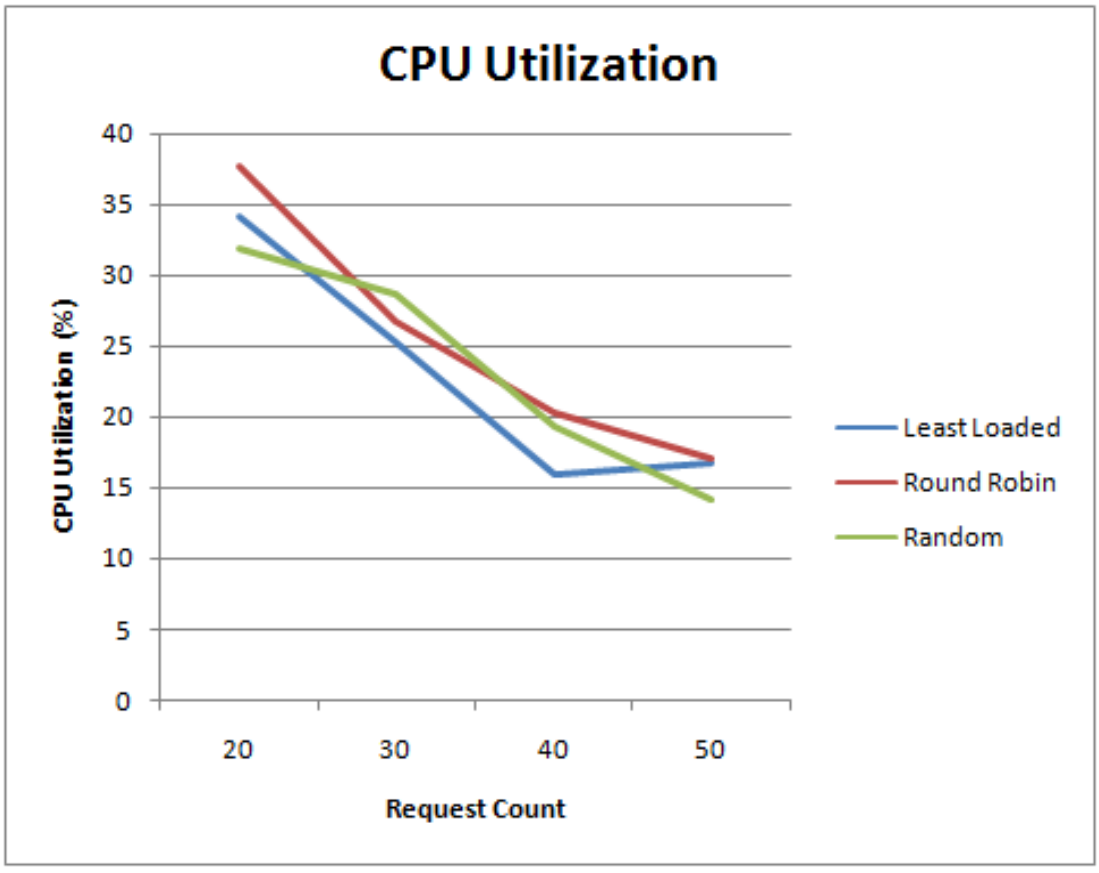


FIGURE 37: CPU UTILIZATION GRAPH

The following figure depicts the successful requests in three algorithms, with different number of requests.

Requests -->	20	30	40	50
Least Loaded	13	15	15	15
Round Robin	14	16	16	16
Random	13	16	16	15

TABLE 12 : SUCCESSFUL REQUESTS

The following graph was generated using the above table.

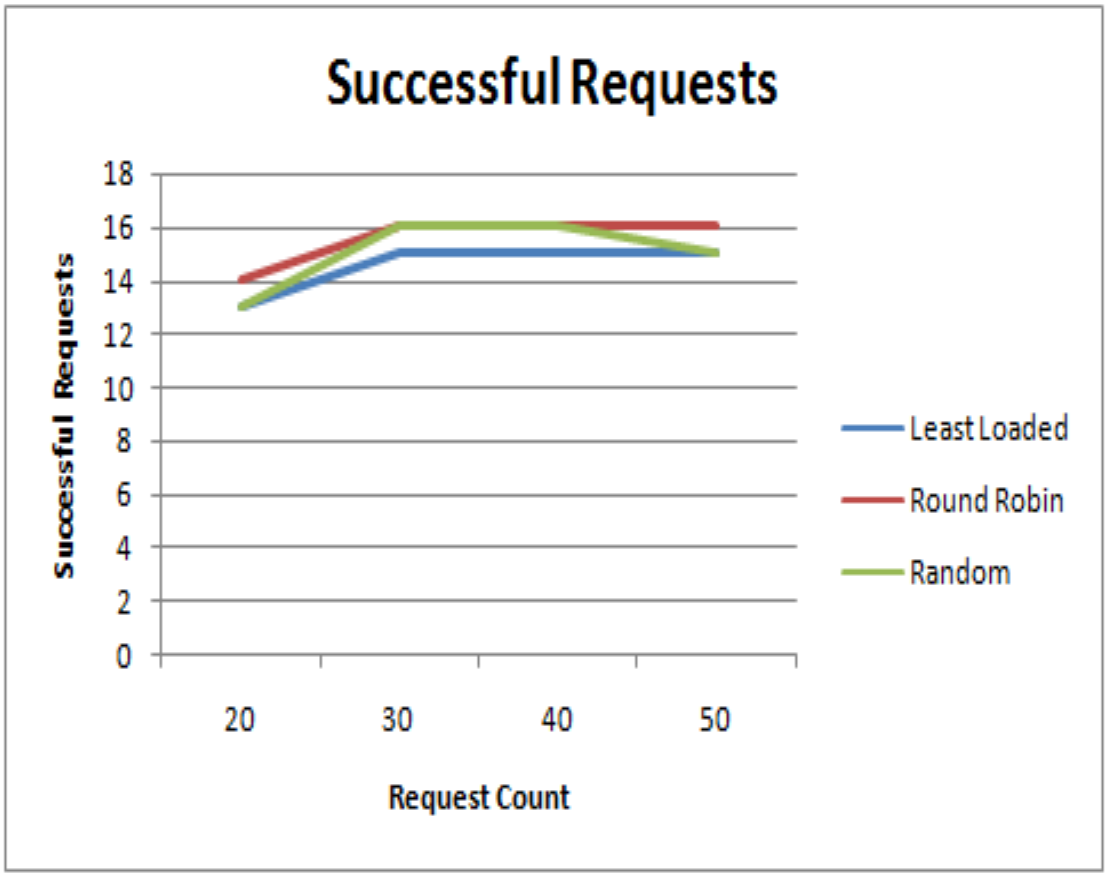


FIGURE 38 : SUCCESSFUL REQUESTS

CONCLUSION

Least loaded load balancing approach would prove to more efficient than the earlier approaches of load balancing, as it takes into account the important factor of time associated with it. Therefore, the requests which are usually lost due to expiry of their deadlines, would also be taken into consideration, as the request having the shortest deadline would be given topmost priority. This would ensure high performance and better user experience. Also the response time would be improved upon, as the factors such as queue length, network delay, request rate and process rate are taken into account. The future work would deal with proving this hypothesis, with the help of scientific data.

REFERENCES

1. Amrita, D., and A. Aruna. "A Scalable Approach for Effective Content Delivery Using Enhanced Distributed Load Balancing Mechanism."

2. Manfredi, Sabato, Francesco Oliviero, and Simon Pietro Romano. "A distributed control law for load balancing in content delivery networks." IEEE/ACM Transactions on Networking (TON) 21, no. 1 (2013): 55-68.Skowron, Piotr, and Krzysztof Rzadca. "Network delay-aware load balancing in selfish and cooperative distributed systems." In Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, pp. 7-18. IEEE, 2013.

3. Mathew, Vimal, Ramesh K. Sitaraman, and Prashant Shenoy. "Energy-aware load balancing in content delivery networks." In INFOCOM, 2012 Proceedings IEEE, pp. 954-962. IEEE, 2012.

4. Mohammed, Sabah, and Edward A. Pingoy. "Implementation of Effective Law for Load Balancing." (2011).

5. 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).

6. Roy, Sudipta, Sanjay Nag, Indra Kanta Maitra, and Samir K. Bandyopadhyay. "International Journal of Advanced Research in Computer Science and Software Engineering." International Journal 3, no. 6 (2013).

7. Sampath, K. V., and T. S. Ragavendra. "Efficient and Distributed Control Mechanism for load Handling in Content Distributed Network." *future* 3, no. 5 (2014)..

8. Skowron, Piotr, and Krzysztof Rzadca. "Network delay-aware load balancing in selfish and cooperative distributed systems." In Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, pp. 7-18. IEEE, 2013.

9. S. K. Mehertaj, K. V. Subbaiah, P. Santhi, T. Bharath Manohar, International Journal of Modern Engineering Research (IJMER) Vol. 3, Issue. 4, Jul - Aug. 2013 pp-2514-2521.

WEB REFERENCES

1. http://www.ramkitech.com/2012/10/tomcat-clustering-series-simple-load.html

2. http://www.cloudbus.org/

3. http://www.loadbalancerblog.com/blog/2013/06/load-balancing-scheduling-methods-explained

4. http://www.google.co.in/patents/US6249801

5. https://devcentral.f5.com/articles/choosing-a-load-balancing-algorithm-requires-devops-fu

6. http://docs.oracle.com/javase/tutorial/