Intelligent load balancing in SDN

Mithesh A
Student
Anna university
Chennai
mithesharun@gmail.com

Niveditha B
Student
Anna university
Chennai
nivi532@gmail.com

Rubak preyan G
Student
Anna university
Chennai
rubakkeerthi123@gmail.com

Yogeeswar S
Student
Anna university
Chennai
yogeeswar9656vm@gmail.com

***Abstract*—The conventional networks are under additional strain as a result of rising users and new technologies. To remedy the problem, traffic management technologies have been employed. This traffic management involves things like traffic load balancing and QOS-based scheduling, among other things. Load balancing is an important aspect of controlling and regulating data traffic among resources. SDN controllers have a global view of the network, which aids in the creation of load balancing techniques that are more efficient. To acquire a better outcome in terms of latency, network speed, fault tolerance, and resource usage, we added a mechanism that employs all algorithms such as round robin, random, weighted round robin, and least connection algorithms. We got mininet simulation results utilising the POX controller in this research, which resulted in greater throughput than other existing techniques.**

***Index Terms*—Load balancing, SDN, Round-Robin, POX controller, Mininet, Robin, Random, Weighted round robin.Load balancing, SDN, Round-Robin, POX controller, Mininet, Robin, Random, Weighted round robin.**

I. Introduction

The need for data is skyrocketing under the present COVID-19 pandemic situation, where schools, institutions, and offices are shuttered. Classes are being moved to the internet, and offices are operating on a work-from-home basis. Load management is the most serious issue in this circumstance. Because traditional networks have a static architecture, they have a hard time dealing with changing network needs. When there is a lot of traffic on the network, this problem becomes increasingly difficult to solve. When the size of the network grows, the complexity grows at an exponential pace. The data plane (task of computing) and the control plane are split into two planes in a typical network (job of communication). Decoupling the data and control plane is made easier with software defined networking. As a result, the network administrator uses an external SDN controller to govern their network. A particular form of controller is employed, one that has the ability to modify the network's forwarding behaviour. The use of a controller lowers costs, streamlines operations, optimises resource consumption through algorithms, and makes software upgrades easier. With POX protocol, the controller sends flow entries to the appropriate switches and thereby controls all of the switches' forwarding rules. The controller is being overburdened with routing decisions as the quantity of packets grows. An effective load balancer distributes network load equally to enhance network metrics including latency, network speed, fault tolerance, and resource usage takes care to keep the entire network's power usage to a minimum.

The technique of successfully distributing traffic across network devices is a crucial component of creating and administering a network. A good load balancer improves network factors including latency, resource usage, throughput, and fault tolerance while using the least amount of power possible. Load balancing is often handled by a dedicated server in older networks. Dynamic load balancing servers have been developed to cope with the ever-changing network needs.

Traditional load balancing techniques are not well suited for the data centre environment due to the unique topology and traffic patterns. On the one hand, typical network load balancing algorithms are built for long scheduling periods and hence fail to balance bursty traffic in data centres. Traditional static load balancing methods, on the other hand, are indifferent to congestion information, which may result in congestion in some cases.

II. Background

Application layer, Control layer, and Infrastructure (or Data) layer are the three primary layers. The SDN application is located on the application layer and is designed to meet the needs of the users. The network administrator is in charge of these SDN applications. The controller, sometimes known as the brain of the SDN architecture, is found at the control layer. By having a global perspective of the network, it will be able to control all of the SDN's activities. To connect with the Application layer, the SDN controller employs a Northbound API such as REST API. It communicates with network devices via the Southbound API. The data layer is also known as the infrastructure layer. This is the physical layer where network devices like switches, routers, and so on are installed. The duty of the switch is to gather network status and communicate it to the controller.

A. SDN's architecture

In the realm of technological industry, SDN is a well-known technocrat. The router is divided into two planes: data and control. The control plane makes the final judgement about how to run the network. The controller's function is to determine the path after the packet arrives in the network. The packet is routed in that path if the route is present. Otherwise, the controller looks for the first packet of each flow's path.

The flow table are used to route the work. The router is no longer working on a compute job. This total network routing information is controlled by the central controller via the remote end. The router updates itself whenever the routing information changes. Both central and distributed systems have their own set of benefits and drawbacks.

B. Controller

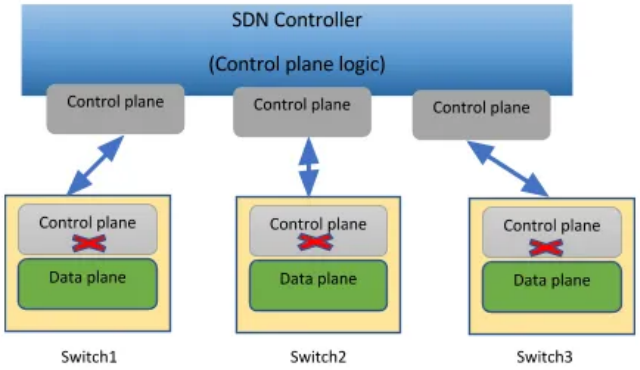

Fig. 1. Controller

The data plane is a network of interconnected forwarding systems connected by wireless radio channels or wired cables, whereas the control plane can be referred to as the network's brain since it contains control logic. Southbound Application Programming Interface is a standard set of instructions for forward devices in the southbound interface. It also specifies the mechanism for exchanging data between forward devices and control plane components. The network working framework provides northbound API (Application Programming Interface) to the application layer. This interface is reused in the building of applications. The major benefit of software defined networking (SDN), which allows for centralised control, is the separation of control and data planes. It simplifies system organisation (which is quite essential on the control plane), while network programability facilitates improvements.

C. Load balancing

Load balancing is a term used to describe the process of balancing In (SDN)software defined networking, load balancing serves as an aware routing mechanism.; it is a required element that aids availability and scalability, resulting in the shortest possible application response time. Millions of individuals are linked to the internet, resulting in increased web traffic, network congestion, and packet losses.As a result, load balancing techniques substantially boosts network efficiency to overcome this problem.

D. Load balancing applications of SDN

1) Network virtualization secures a physical network by dividing it into many components. Network virtualization enhances network performance and automation when new software components are introduced.
2) The SDN controller assigns topology discovery as a service. It maintains and gives a global perspective of the network.
3) Security enhancement refers to going above the baseline specification in terms of security skills and offers.
4) Traffic monitoring provides the necessary data to notify drivers of issues and information important to road engineers, such as vehicle count, speed, and occupancy.

III. Literature survey

In [2] Rukmini bhat and et al have compared RTT for two cases using Fat Tree topology. One with application of least connection algorithm which is dynamic scheduling algorithm because it counts the number of active connections for each server and to make it more efficient, Dijkstra algorithm is used to find the shortest path from the host to the selected server, and another without applying the above mentioned algorithm. Results shows that standard deviation for sending packets is much higher before applying algorithm, but this solution does not consider real time load on the servers.

In [3], Buhyavarapu Manasa, A.Ramesh Babu examined the feasibility of various approaches, atypical strategies for load balancing, and related studies for forecasting load balance mechanisms utilising a variety of ways. SDN's path-repetition period, like mainstream society, succeeded in adding strength and balance to the community. However, SDN researchers have discovered that the uneven distribution of traffic over various routes is a serious issue.

Kiran A Jadhav, Mohammed Moin Mulla and Narayan D in [4] have compared Round Robin algorithm in which each server node receives the request from client nodes in a circular fashion, and Random load balancing algorithm in which the request to the servers are forwarded in a random order. Various parameters such as number of servers, number of clients, and flow timeout are varied. It is evident from the results that applying the concept of Round Robin selection for load optimization reduces packet loss, increases the throughput and provides better performance in comparison to the random selection technique since random selection has large overheads. It is also clear from the results that Round Robin selection also avoids overloading of any single resource which minimizes resources consumption and congestion, leading to an effective load balance which enhances the network performance.

In [5], Yassin Abdulkarim Hamdalla Omer, Amin Babiker A. Mustafa and Ashraf G. Abdalla examines a situation in which if some servers have more CPU, RAM or other specifications, there is no way the algorithm can distribute more requests to those servers. As a result, servers with lower capacity might soon become overburdened and fail, while other servers' capacity remains idle. Also, in the traditional network, hardware comes from one vendor and the protocols are limited to use and typically the

devices are configured one by one. The load balancer design is to efficiently spread the load between the servers based on the protocol. The results showed that network performance has increased after running the load balancer. This can replace traditional dedicated load balancer hardware.

Mohammad Riyaz Belgaum, Shahrulniza Musa, Muhammad Mansoor Alam and Mazliham Mohd Su'ud in [6], examines that due to the increasing demand and the scarcity of resources, To manage incoming traffic and resources and increase network performance, the load balancing issue must be solved quickly. The function of the controller in SDN in balancing the load for improved Quality of Service is one of the most significant challenges (QoS).They also included the study of metrics and parameters which have been used to measure the performance. It would help academics to learn more about load balancing techniques in SDN while also allowing them to fill in research gaps.

In [7], R. Rajalakshmi and E. Uma examines the many methodologies accessible, as well as the numerous load balancing strategies and related studies of forecasting load balancing schemes utilising a variety of ways. This study categorises load balancing techniques and examines their merits and downsides.

Suchismita Rout, Sudhansu Shekhar Patra, Punyaban Patel and Kshira Sagar Sahoo in [8] offered a quick overview of the SDN network, its architecture, and how the SDN has effectively increased communication in the fast-paced technological environment in their article. SDN and NFV's relevance in allowing intelligent load balancing was also mentioned. Network managers may use SDN and NFV to offer automated network management and efficient resource allocation and provisioning.

Gaurav Tiwari, V Deeban Chakaravarthy and Aditi Rai in [9] examined traffic and proposed a number of solutions for dealing with it. With the RYU controller, the mininet tool is utilised. Latency has been increased while bandwidth has been reduced. Wireshark was used to analyse the network, and a graph was created. They have proposed Dijkstra's algorithm which is a load-balancing algorithm that takes advantage of the shortest path.

Tao Hu, Peng Yi, Jianhui Zhang and Julong Lan in [10] proposes Reliable and Load balance-aware Multi-controller Deployment (RLMD) strategy to address the above problems. To begin, construct a multiple-controller network model and define the RLMD settings. To find the most trustworthy controller installations, the Controller Placement Selection (CPS) approach is utilised. The Multiple Domain Partition(MDP)algorithm is now used to allocate controller switches based on node attractability and controller load balancing rate, resulting in a domain design that is acceptable.

In [11], Kavana H M , Kavya V B , Madhura B and Neha K implemented and compared the algorithm using Floodlight controller, Mininet and testing results via Wireshark Network Analysis Tool. In this paper, they have used python in mininet to convert the algorithm into code for balancing the load. The information about the host connected is found using Shortest Path First concept. So according to the code, to balance the load it should choose that link which has the least cost. The total link cost for all possible routes is found. After getting the current transmission rate, the best path is chosen.

IV. Proposed work

The volume of internet traffic has expanded dramatically as a result of technical advancements. As a result, the intelligence, efficiency, and dependability of the underlying network must all be increased. Traditional network architecture has a major problem with load management. Traditional networks have a hard time responding with changing network demands because of their rigid architecture. This problem is tough to solve in instances when there is a lot of network traffic. The complexity of a network increases at an exponential rate as its size expands. Existing networks were not designed to endure such rapid expansion, resulting in their incapacity to deal with rapid development. In a normal network, the data plane, which is responsible for computation, and the control plane are kept distinct. The number of routers increases in direct proportion to the network's size. Route calculation consumes more time than data transport in a big network. Software Defined Networking (SDN) frees routers from computational chores by separating the data plane and control plane. The control operations of all routers are managed by the external controller, which governs the whole network and determines global routing choices. It also has software that enables it to make network management decisions. The technique of appropriately dispersing traffic across network devices, also known as load balancing, is a crucial component of planning and administering a network. SDN also adds programmability to the network by allowing the configuration and operation of the network to be specified and adjusted via programming. A good load balancer increases network characteristics such as latency, resource utilisation, throughput, and fault tolerance while using as little power as feasible. In older networks, load balancing is frequently performed by a dedicated server.

In this paper, we examine and compare several load balancing approaches on SDN, as well as the benefits of using numerous controllers with various combinations of algorithms. We did the following in order to conduct a complete experimental study:

1) We highlight the most important aspects of employing remote controllers (POX) and Software Defined Networks.(SDN).
2) Review of existing methods in terms of single-controller load balancing techniques.
3) A list of performance indicators for comparing and evaluating algorithms across several controllers.

4) Implementing load balancing by using each technique in numerous controllers and analyzing results.
5) Finally, amalgamation of the above-mentioned techniques to create a more efficient load-distribution solution in SDN.

The working technique was first implemented using a single controller to gain knowledge of the controller structure. Connections are appropriately secured. The remote controller is activated and requests are sent through mininet. Each request will be sent through the switch to the controller. In response to the host, the controller applies the load balancer's algorithm properly and routes the requests as required. But a more effective solution is required since a single controller is not sufficient to manage the requests submitted in real-world scenarios.

This paper's topology includes four controllers, as well as the essential switches, servers, and hosts, all of which are interconnected. The single controller has just one switch, but the implemented topology has the same number of servers and hosts. To determine the effectiveness of the aforesaid structures, a detailed comparison is made. Several tests are carried out on both of these topologies in order to determine the more efficient solution. The following test are used to assess efficiency:

1) Concurrency Level
2) File size
3) Request Sent
4) Throughput

The results are obtained and reviewed independently for both topologies. To acquire our observations and results, the efficiencies based on the above-mentioned criteria are also studied and combined. Because no one SDN controller will be utilised in real-time settings owing to high load and demand, the tests are conducted on a slightly modified version of the implemented topology in order to get findings that are as near and accurate to a real-life scenario as feasible. Different load balancing algorithms are assigned to each of the four controllers. It consists of a single load balancing switch that is linked to four additional switches, as well as sixteen host systems. The results of each of the five tests are collected for this modified form of load balancing, and the findings are tallied to arrive at a conclusion on its effectiveness.

There are many long-established algorithms often used for the purpose of load balancing in SDN. Each of the standardized algorithms have been used to try to obtain productive results. But these algorithms perform efficiently only under specific circumstances. In real-life scenarios, this is not feasible. The networking system in practical applications needs to yield coherent and appropriate results under all kinds of situations that may arise. In order to suit rising needs of technology, a hybrid algorithm has been proposed in this paper:

A. Random Algorithm

Input : Request, Set of available servers.
Output : Request allocation to servers.
Begin
1) Flowtime = 20-80 seconds
2) Get list of servers
3) Whenever a new client request comes in, pick a random server from the list of servers using: chosen-server = random.choice(SV-HOSTS)
4) Assign the request to a random chosen server

End

B. Round Robin Algorithm

Input: Request, Set of available servers.
Output: Efficient request allocation to servers.
Begin
1) Flowtime = 20-80 seconds
2) Get list of servers
3) Whenever a new client request comes in Pick a server from the list of servers using self.last-server-idx=(self.last-server-idx+1)
   chosen-server = sv-hosts[self.last-server-idx]
4) Request is forwarded to each server in a cyclic fashion

End

C. Least Connection Algorithm

Input : Request, Set of available servers..
Output : Request allocation to servers.
Begin
1) Flowtime = 20-80 seconds
2) Get list of servers
3) Get number of active connections of respective servers.
4) Whenever a new client request comes in, pick a server with least number of active connections from the list of servers.
5) After choosing a server number of connection is incremented.

End

D. Weighted Round Robin Algorithm

Input: Request, weight, Set of available servers.
Output: Efficient request allocation to servers.
Begin
1) Flowtime = 20-80 seconds
2) Get list of servers
3) Whenever a new client request comes in Pick a server from the list of servers using round robin strategy but to follow the weights,i.e. Server with higher weight get higher number of request at a time.
4) Request is forwarded to each server in a cyclic fashion

End

V. Implementation

To run the algorithms, a network simulator and remote controllers are vital. Mininet is used in this paper to create a virtual networking system on a single system while POX is connected and used as the remote controller.

A. Mininet

It's a network emulator that creates a virtual host, switch, controller, and connection system. Mininet's switches support OF for highly adjustable custom steering and SDN technologies, and they run normal Linux arrange programming. Mininet is widely used because it is simple to set up, supports custom topologies and bundle sending, runs genuine Linux projects, runs on PCs, servers, and virtual machines, has sharing and recreating capabilities, is easy to use, and is in an open source and dynamic development state.

The number of hosts in our custom topology is 16, the number of switches is 5, and the number of controllers is four. We developed the topology in such a manner that each controller uses a different algorithm.

1) C0 - Random algorithm
2) C1 - Round robin algorithm
3) C2 - Least correction algorithm
4) C3 - Weighted round robin algorithm

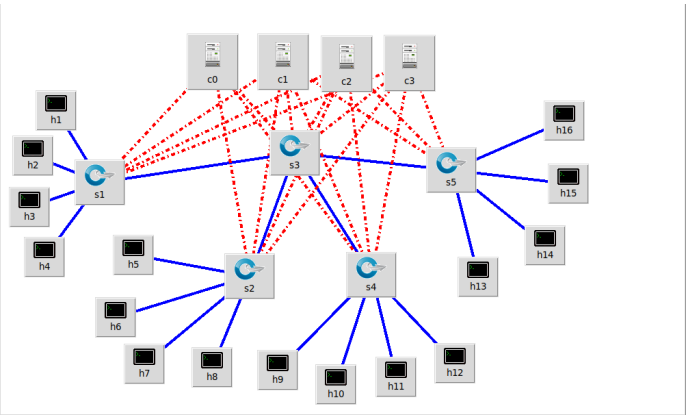

Fig. 2. Mininet

As can be seen, the switch c1 has four hosts. The hosts h1,h2,h3 function as servers and respond to the requests made by the host h4. Every switch in the system follows this pattern. The first three hosts will function as servers, while the fourth and final host will serve as hosts.

---

sudo mn –topo single(Type of topology),8(number of hosts) –mac –arp –controller=remote
xterm h1 h2 h3 h4 h5. . . .(nodes).

B. Pox

The POX controller is in charge of establishing communication with SDN switches using the OpenFlow or OVSDB protocols. With the aid of the Python programming language, developers may utilise POX to create an SDN controller. Python is a widely used programming language for constructing and designing software defined networks and network applications.

/pox.py log.level –DEBUG rand –ip=10.0.1.1 –servers=server1,server2,...(enter as many nodes as you want).

Here the word 'rand' represents the algorithm used. This should be changed to round or least to attain the results for Round robin and Least count algorithm respectively .

VI. Results and Discussion

The results of simulating the suggested load balancing approaches in SDN using the Mininet emulator are discussed in this section. The suggested technique's performance is compared using the following QoS parameters: average throughput and average total time: Average Throughput: This is the number of data packets that successfully pass across the communication channel in a given amount of time. Average Total Time : The time it takes to complete all of the requests from the host.

To show that the usage of multiple controllers is more efficient, a few parameters(concurrency level, file size and number of requests) are compared.

Time Vs. Concurrency Level

	10s	20s	30s
Four Controllers	2.094	1.248	0.860
Single Controller	2.231	2.405	1.036

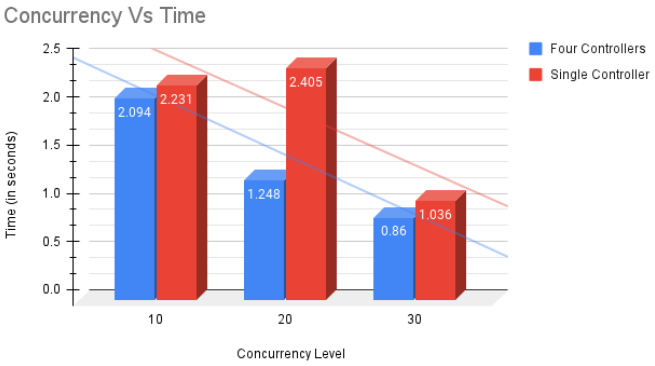

Fig. 3. Time Vs. Concurrency Level

From the graph in Fig 5, it is clear that the time taken to complete 100 requests decreases when concurrency level increases. Also the time taken by multiple controllers is lesser when compared to the time taken by single controller. This is due to the obvious reason that load is distributed more evenly.

Time Vs. Number of request

	100s	200s	300s
Four Controllers	2.94	4.11	5.156
Single Controller	1.913	4.998	7.488

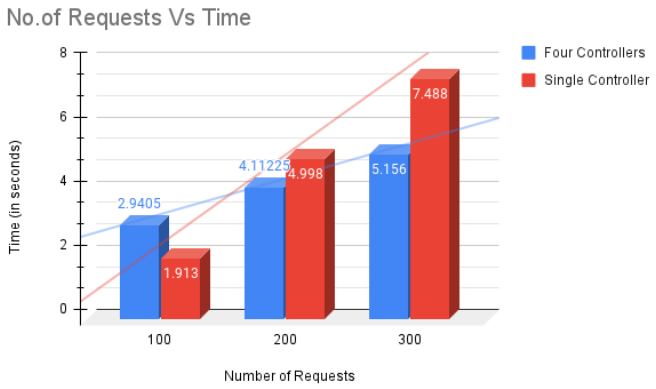

Fig. 4. Time Vs. Number of request

In the above Fig 6, we can see that the total time taken to complete all the request increases with increase in number of requests, but the topology with multiple controllers takes lesser total time when compared to single controller in all the tested situations, hence giving a higher throughput.

Time Vs. File size

	100kb	500kb	100mb
Four Controllers	1.381	1.377	29.64
Single Controller	2.171	3.343	41.56

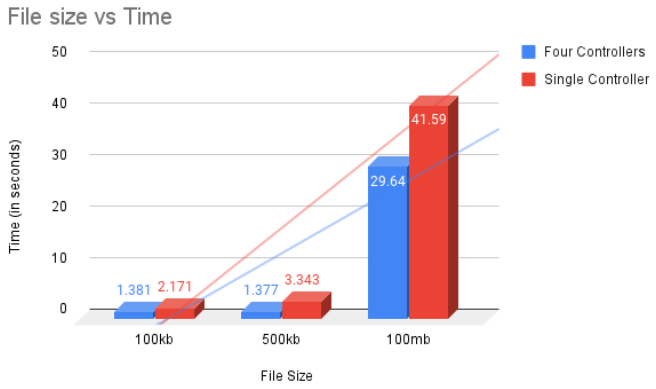

Fig. 5. Time Vs. File size

On analysing the above Fig 7, although total time taken to complete all the request increases with the increase in file size, time taken by topology with multiple controllers is less when compared to topology with single controller. Inferring from all of the above cases, we can say that having multiple controllers is more productive when compared to having a single controller as the load gets distributed faster and more evenly. To evaluate the performance of a network with multiple controllers, various conditions are switched to see under which circumstances the system works at peak efficiency. The topology's capability when using different algorithms are also explored.

In order to evaluate the performance of our proposal, we have created a scenario that includes a rise in the number of requests from each host, which is equal to an increase in the number of users in a real-world situation. As a result, 100, 200, 300, and 400 requests are made to the server from each host in order to compare the performance of our proposed algorithm to that of other methods.

Average total Time Vs. Number of Request

	100s	200s	300s	400s
Implemented	2.9405	4.11225	5.156	13.65
Random	3.336	5.91	10.31	17.911
Round robin	2.951	14.32	16.033	15.168
Least count	2.1855	4.857	6.837	17.572
Weighted RR	2.142	4.216	7.286	19.258

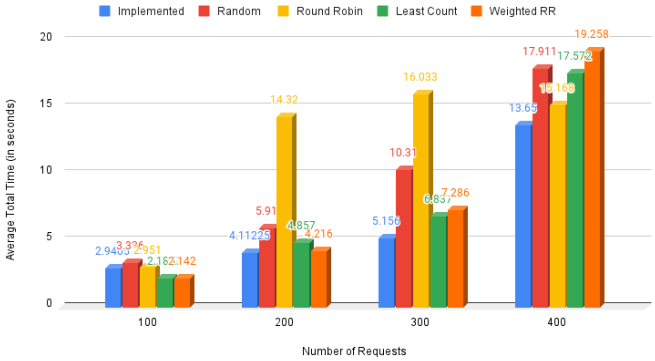

Fig. 6. Average total time Vs. Number of request

From the graph in Fig 8, it is shown that there is a subsequent increase in average total time of proposed algorithm when there is an increase in number of requests, but that is less than all the other implementations where all the controller runs the same algorithms.The load on each of the routers are reduced and the response time is faster.

Average Throughput Vs. Number of Request

	100	200	300	400
Hybrid	32.61	51.66	56.66	29.34
Random	30.51	33.87	29.11	22.35
Round robin	34.07	14.49	14.56	26.43
Least count	45.88	51.95	44.075	22.77
Weighted RR	46.847	47.57	40.66	20.78

From the below graph that is in the Fig 9 , it is clear that there is a decrease in average throughput with increase in number of requests. When comparing the overall result, the

average throughput of the proposed work is greater than all others in cases of high requests.

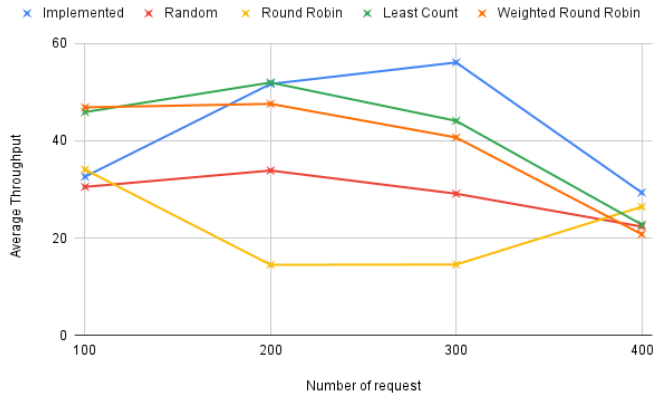

Fig. 7. Average Throughput Vs. Number of requests

VII. Conclusion

SDN is an open and flexible paradigm that helps in the regulation of the network switches on a very large-scale network. Centralization of the forwarding information base allows optimum routes to be calculated deterministically for each flow end-to-end across the topology. SDN has gained tremendous momentum and as it is advancing towards an era of modern data networks, it requires a highly extensible and intelligent Load Balance Management. This paper outlines the design of SDN and assesses the ramifications of using various algorithms utilized by single or multiple controllers.

Two points are evident from the results, first, performance of the network with single controller is poor when compared to network with multiple controllers, and secondly, throughput of network with multiple controllers running different algorithms at a time is better with increase in the number of requests, when comparing this with each controller running on the same algorithm.

Selection of SDN controller, load balancing mechanisms etc. are purely dependent upon the type of the application. The algorithmic combinations and topologies used in this paper cannot be equated fully to real time load on the servers. That is, it'll not consider traffic volume, uneven loads, and requests with large file size. So, the necessary software requirements and networking factors must be pre-determined according to the usage so that the SDN can effectively respond to the requirements of real-life applications and optimize the employment of the network without compromising on service quality.

Acknowledgment

We thank Prof.Dr.Varalakshmi.P, for providing essential input for improving the quality of this article. Her zeal for research will have a big influence on our future projects.

References

[1] Varalakshmi Perumal and Sankari Subbiah "Network Resource Provisioning in Cloud Data Center Across Manifold SDN Controllers," 2016.
[2] Rukmini Bhat B, Sneha N S, Keerthana Bhat, Chaithra C Kamath and Chaitha Naik, "Improving the Efficiency of Software Defined Network through Load Balancing Algorithms," 2021.
[3] Buhyavarapu Manasa and A.Ramesh Babu, "A Research on load balancing on software defined networks," 2021.
[4] Kiran A Jadhav, Mohammed Moin Mulla and Narayan D, "An Efficient Load Balancing Mechanism in SDN," 2020.
[5] Yassin Abdulkarim Hamdalla Omer, Amin Babiker A. Mustafa and Ashraf G. Abdalla "Performance Analysis of Round Robin Load Balancing in SDN,",2020.
[6] Mohammad Riyaz Belgaum, Shahrulniza Musa, Muhammad Mansoor Alam and Mazliham Mohd Su'ud "Systematic Review of Load Balancing Techniques in SDN," 2020.
[7] R. Rajalakshmi and E. Uma"A Survey On Load Balancing in Software Defined Networks," 2020.
[8] Suchismita Rout, Sudhansu Shekhar Patra, Punyaban Patel and Kshira Sagar Sahoo "Intelligent Load Balancing Techniques in Software Defined Networks: A Survey," 2020.
[9] Gaurav Tiwari, V Deeban Chakaravarthy and Aditi Rai, "Dynamic Load Balancing In Software Defined Networking,", 2019.
[10] Tao Hu, Peng Yi, Jianhui Zhang and Julong Lan, "Reliable and load balance-aware multi-controller deployment in SDN,", 2018.
[11] Kavana H M , Kavya V B , Madhura B and Neha K, "Load Balancing using SDN Methodology,", 2018.
[12] Subbiah Sankari, Perumal Varalakshmi2 and Boopathi Divya, "Network Traffic Analysis of Cloud Data Centre,", 2015.
[13] R. Rajalakshmi and E. Uma, "A Survey On Load Balancing in Software Defined Networks," 2020.
[14] Sankari Subbiah and Varalakshmi Perumal "Energy-Aware Network Resource Allocation in SDN,",2016.
[15] Buhyavarapu Manasa and A.Ramesh Babu "A Research on load balancing on software defined networks,",2021.
[16] Sikandar Ejaz, Zeshan Iqbal, Peer Azmat Shah, Bilal Haider Bukhari, Armughan Ali and Farhan Aadil "Traffic Load Balancing Using Software Defined Networking (SDN) Controller as Virtualized Network Function,",2019.