Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 10, October-2022

Controller Load Balancing in Software-Defined Networking

Genetu Y. Basena
Department of Computer Science and Systems Engineering, Andhra University
Vishakhapatnam, Andhra Pradesh, India

Prof. P V G D Prasad Reddy
Department of Computer Science and Systems Engineering, Andhra University
Vishakhapatnam, Andhra Pradesh, India

***Abstract*- In cloud environment, handling users' service requests and providing the requested resources fairly is a decisive challenging. The Load balancing is important to fairly allocate service requests to unloaded server dynamically maintain uniform load distribution in server farms. In conventional IP networks, maintaining a load balancing is an obstinate and unadaptable task due to lack of global topology view by the forwarding devices. However, SDN provides centralized decision making for any topological changes in minimum time fractions. To address the aforementioned challenge, we proposed a new server side load balancing strategy that allows efficient and effective server management scheme for the SDN OpenFlow networks. Experimental results conducted on Ryu controller and Mininet emulator depicted increased performance compared to the existing mechanisms.**

Keywords – SDN, Control plane, OpenFlow, Load Balancing, Ryu, Mininet.

I. INTRODUCTION

The rapid growth of Internet and the services hosted on it, the computing platforms which are dynamically expandable and virtualized such as Servers, data storages, software, and networking are available online has greatly simplified the innovation and research activities in the field of IT [1]. The cloud provides server storage and computing resources online to use on on-demand bases for the end users. Users acquire those services from the distributed servers located on different data centers. The Computing system hardware and software resources, mainly the computing power and data storage are obtainable on-demand, without the end users local administration and management. Users can keep their data and files on clouds rather than saving them on hard drives for effective access from anywhere connected to the Internet. The cloud predominantly divided into two layers, the frontend which provides interface to interact their stored data using software applications. The backend contains software and hardware components such as computing powers, central servers, database and storages. In some cases middleware software interconnects the frontend and backend for unified access.

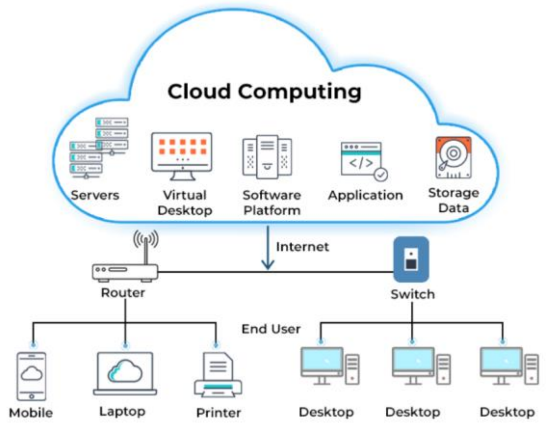

Figure 1. Cloud computing architecture

1.1 Software Defined Network

SDN is an innovative model that simplifies the network management and control by providing the low level functional abstractions [2]. It allows the network administrators to quickly respond to any change evolving changes due to topological or policy requirements [3]. In SDN, unlike the traditional networking, performance and functionality changes are managed and controlled through software programs residing in controllers rather than individual devices being configured by the administrator. The SDN architecture has two main planes. The control plan which is accountable for data transmitted over the entire network. It has all the needed software logics programed inside the server known as the controller. The data plane is mainly responsible for data forwarding. It contains network devices such as switches and routers. The two planes interact through an OpenFlow which is a standard protocol implemented for SDN systems communication [4].

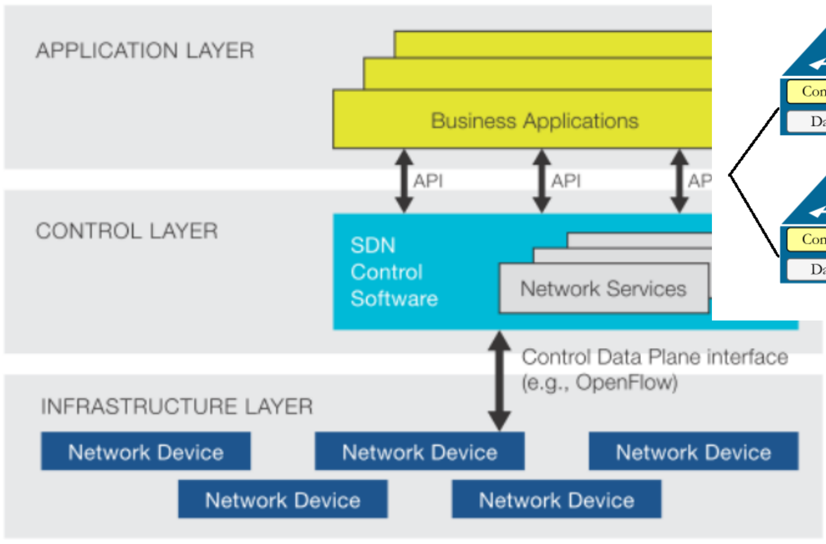

Figure 2. The SDN architecture

SDN architecture is shown in Fig. 2. (a) Shows a layered architecture having the forwarding plan, the control plane, and the application plane. Network application layer comprises the programs such as load balancer, firewall, and security apps and so on. The forwarding plane involves virtual and physical devices which run traffic forwarding rules implemented using the OpenFlow protocol. The control layer consists of centralizing control plain used for communication with below layer of sdn known as infrastructure layer using OpenFlow protocol which uses southbound API.

Table 1. Difference between traditional network and SDN network

Conventional IP networking	Software Defined Networking
Statically configured per devices and strict networks, not convenient for custom applications.	Easily programmable and adaptive networks during application development as well as on the production times whenever business requirements are changed.
Operates using proprietary protocols	Operates through APIs, customizable as per users need
Hardware devices	Virtual devices customized by users via open software tools.
Decisions made per device bases through the distributed control plane	Logically centralized control plane

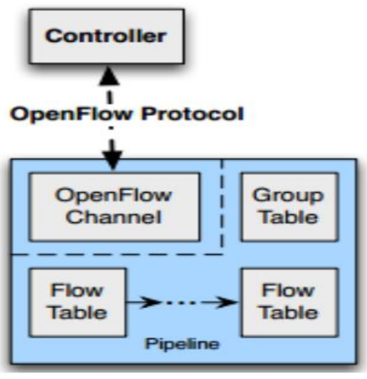

Figure 3. OpenFlow architecture

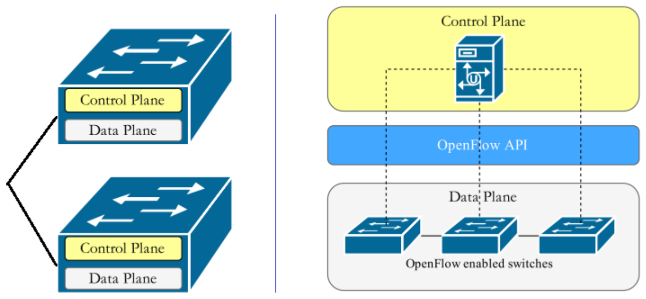

Figure 4. Conventional Vs. SDN networks

The remainder of the paper is arranged as follows. Related literature works are presented in section II. The suggested Load balancing model is detailed in section III. Section IV will present the experimental results. Finally section V presents the concluding remarks.

II. RELATED WORKS

This section briefly discusses the latest research works conducted in the area of SDN based Load balancing.

Guoyan Li et. al in [5], proposed a particle swarm optimizations approach using network queuing model in order to analyze the propagation delay. To implement this method, they used Breadth First algorithm to search best paths to implement load balancing. Their study formulated the Intra and Inter domain communication overhead cost estimation approach. Authors in [6] have presented a dynamic load balancer optimization approach using a swarm optimization algorithm (SOA). Their study shown better approach optimized the number of switches to be connected with a controller for better load balancing. Results also showed minimized latency and deployment costs. Xai Xue et al. [7] presented an approach known Ant-Colony Optimization technique augmented with the Genetic Algorithm (GA) to implement an optimized Load balancer. Their study addressed network resource overloading problem and obtained improved path selection method for routing purposes. S. Kaur et al. [8] presented a server load handling mechanism that implemented load balancer using the Round Robin (RR) approach in POX controller and OpenFlow vSwitches. They obtained better performance results compared with randomized Load balancing algorithm. Tkachava et al. [9] discussed a Centralization management and control of traffic distribution among multiple network paths having better throughput utilization. Their approach was tested on small sized network and provided better load distribution on data forwarding devices. Tiago Oliveira et al. [10] presented a dynamic load balancing strategy that allocates video traffic request in multi-servers system. They achieved a flexible approach to deliver high capacity quality of service needs required in telecommunication industries to maintaining users' quality of experience to address the QoS issues. Sushma Sathyanarayana et al. [11] proposed an Ant-colony optimization combined with dynamic server load balancing approach to choose the best path which has smaller delay and better throughput. Kannan Govindarajan et al. [12] presented a novel load balancing technique implemented using a particle swarm optimizations approach. The deployed technique revealed minimum average response time performance. Yao Chung Chang et al. [13] presented bacteria inspired network algorithm combined with genetic algorithm to send large traffic volume using paths with larger bandwidth.

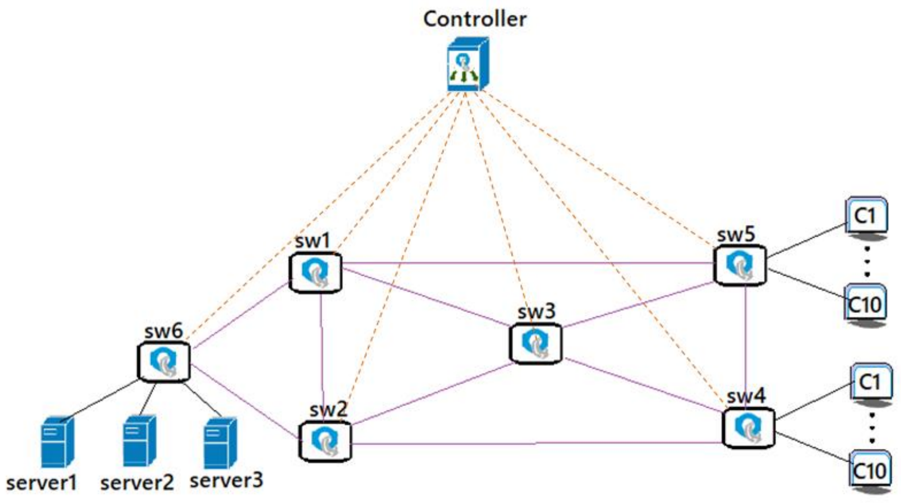

Figure 5. Proposed systems architecture

II. PROPOSED MODEL

This section describes the essential components of our network load balancing application design which is followed by load adjustment mechanism. Usually, when a new packet arrives to a switch port, It checks out its flow table entries with the address information inside the packet header. Incase if a match fails, then the switch sends a PACKET_IN message to the SDN controller expecting routing information. Then, the controller forwards PACKET_OUT message to the switches to install the flow information. Otherwise, the switch forwards the packet to next hop. Similarly, all switches connected in the SDN network will follow the above procedure till the packet reaches its final destination.

In this study, we designed an SDN networks as undirected Graph G which is consists of a controller C={C1}, M switches S={S1, S2, S3,…, SM}, K servers D={D1, D2, D3, …, DK} and N hosts H={H1, H2, H3, …, HN}.

The Servers provide the same services for clients' request.

The presented system model suggests an improved version of load balancing strategy that can be used in the SDN controller and the openFlow switches. The switches store traffic information in their flow table, the controller analyses the network statistics and the information is constantly updated for load adjustments.

3.1 Measurring Server load

In our design, we calculate the network latency at equal time interval, T. Let's assume A_x and B_y to designate the x_{th} switch and the yt_h time interval respectively. We assume T_{arrive} shows the PACKET_IN message time of arrival and T_{reply} indicates time interval until we get PACKET_OUT message from the controller. Hence, latency is defined as,

$$T_{responce} = T_{reply} - T_{arrive} \qquad (1)$$

3.2 Measuring system Threshold

To find the overloaded server and load imbalance existence on the network, a constant threshold value is important. When the server load is higher than the threshold value it is highly possible for network unbalance due to longer time needed to settle the massive network traffic.

3.3. Proposed algorithm

Algorithm 1: Proposed Algorithm to select least loaded server.

```
Input:    Set of flows through switches F={f_1, f_2, f_3,..., f_n}
          Set of links connecting servers L= {l_1, l_2, l_3, ..., l_m}
          Set of connected servers S={s_1, s_2, s_3, ..., s_n}
Output: an Optimal server S_min is selected for the pool.
```

```
begin:
while packet (i) is generated by clients do
          foreach flow $f_i$ Ɛ F do
          if flow size ($f_i$) > Thershold Ts
                    foreach S Ɛ $S_n$ do
                    if L(s) < $L_{min}$ then
                      $L_{min}$ = L(s)
                      $S_{min}$ = S
                    end
          else
          Select server using WRR scheme    //least    loaded
server (Smin) using weighted RR
      end
end
```

IV. SYSTEM IMPLIMENTATION

In this study, we implement a new load balancing algorithm in SDN network topology shown in Figure 5. above and study performances in terms of server response time and throughput. The load balancing experimentation is conducted on the software and hardware specified in Table 1 and Table 2 respectively. The algorithm and network topology were scripted using Python programming language that is Python3. The Mininet emulator is connected with a Ryu controller at port number 6633.

Experimental Setup

The subsequent software and hardware components are used to perform this experiment. In Table 1 provides the list of the software with description.

Table 2. Software specifications

Software	Version
Ubuntu	20.04
Ryu controller	4.32
Mininet	2.3.0d6
Open vSwitches	2.13.0
iperf	2.0.13
Virtual Box	6.1
Wireshark	3..2.3
RESTer	4.1.1

Table 3. Hardware specifications

Hardware	Version/capacity
CPU/Processor	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Memory (RAM)	8 GB
Hard Disk	1 TB

V. RESULT AND DISCUSSIONS

This section discusses the results obtained by simulating the load balancing algorithms in SDN. We used Ryu as controller and the Mininet emulator. The performance of the load balancing algorithms is compared using the two performance measurement parameters, the average response time and throughput. which are defined as follows.

Throughput: Is a performance measurement criteria that measures the amount of packets transmitted successfully in any given period of time. Average Response Time: Is a performance measurement parameter which calculates the total time taken to respond for clients’ request during a given time period divided by the responses made by server.

The HTTP protocol server Load testing tool is utilized to test the load balancing algorithms performance. The Throughput and Response time data were collected from the tool and tabulated as shown in the Table 3.

Table 3. Comparison of Average response time and Throughput for different Load balancing algorithms

Load balancing Algorithms	Performance Metrics	
	Average response time (sec.)	Throughput (MB/sec)
Proposed algorithm	1.15	0.389
Weighted Round Robin (RR)	1.41	0.364
Least connection	2.12	0.340
Random	2.48	0.350

Figure 6 shows results obtained to analyze the average response time collected by varying the number of clients for each Load balancing algorithms in the experiment. It is perceived that as the number of clients increases, the average response time becomes higher in every algorithm. Though the extent of the variation differs per each algorithm. The performance of our proposed algorithm shows minimum average response time result compared to the others.

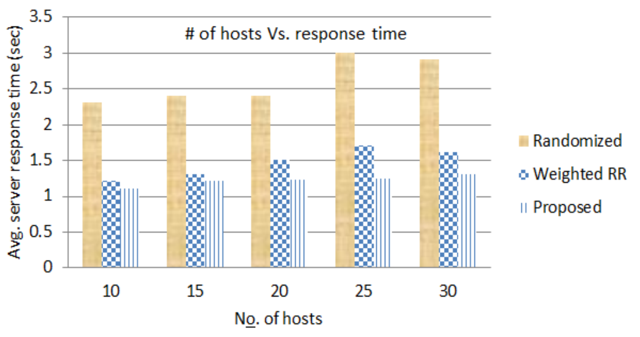

Figure 6. Comparison of response time for different Load balancing Algorithms by varying the size of clients.

Figure 7. depicts the throughput performance of the three load balancing algorithms evaluated by varying the number of clients generating the HTTP GET request using curl command in Linux system. The proposed algorithm performs better than Randomized load balancing algorithm. However, as networked clients size increases the throughput slightly decreases. The weighted Round Robin algorithm shown better throughput value compared to the three Load balancing mechanisms.

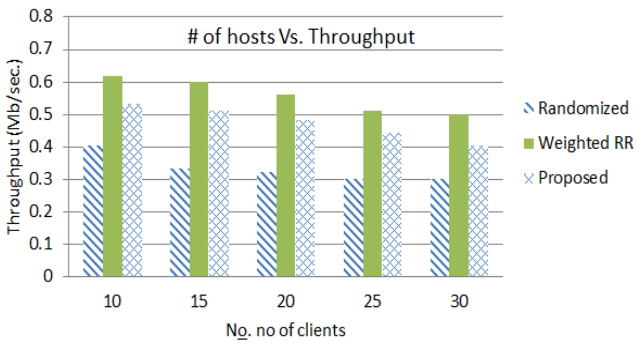

Figure 7. Comparison of Throughput for different Load balancing Algorithms by varying the size of clients.

V. CONCLUSION

Dealing with SDN based load distribution issues among the available servers is easy and efficient compared to the conventional load balancers. The experimental study conducted to measure the throughput and average response time performances of Load balancing algorithms by changing the number of clients requesting services on the network. In this venture, our proposed load balancing scheme shown a minimum response time compared to the other load balancing algorithms. We also observed the Weighted Round Robing scheme has higher throughput than the other algorithms. The main limitation of the experimentation is that it is not conducted using hardware setups. We used Ryu controller and Mininet emulator as the testing tools. The proposed algorithm delivered smooth traffic flow between clients and servers without impairments.

REFERENCES

[1] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., Al-Jaroodi, J.: A survey of load balancing in cloud computing: challenges and algorithms, pp. 137–142 (2012)

[2] Zhang, H., Guo, X.: SDN-based load balancing strategy for server cluster. In: 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems (2014)

[3] Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using openflow: a survey. IEEE Commun. Surv. Tutor. 16(1), 493–512 (2014)

[4] J. Ali and B. H. Roh, "An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping," IEEE Access, vol. 8, pp. 88990–89006, 2020.

[5] Li G, Wang X, Zhang Z. SDN – based load balancing scheme for multicontroller deployment". IEEE Access 2019

[6] Ateya AA, Muthanm A, Vybornova A, et al. Chaotic Salp swarm algorithm for SDN multi-controller networks. Eng Sci Technol. 2019.

[7] Xue X, Kim KT. Dynamic load balancing of software – defined networking based on genetic Ant colony optimization. Sensors. 2019.

[8] Kaur S, Singh J, Kumar K. Round Robin based load balancing in softwaredefined networking. 2nd international conference on computing for sustainable global development IEEE 2015

[9] Tkachova O, Chinaobi U, Yahya Ar. A load balancing algorithm for SDN. Sch J Eng Technol. 2016.

[10] Oliveira T, Sargento S. QoE – based load balancing of OTT video content in SDN networks. Symposium on computers and communications (ISCC). IEEE; 2019.

[11] Sathyanarayava S, Moh M. Joint route – server load balancing in software defined networks using ant colony optimization. IEEE. 2016.

[12] Govindarajan K, Kumar VS. An intelligent load balancer for software defined networking (SDN) based cloud infrastructure. IEEE; 2017.

[13] Yao CC, Wei X-C, Jin WJ. Bacteria inspired communication mechanism based on software defined network. 27th Wireless and optical communications conference (WOCC 2018). IEEE 2018