**Load Balancing in Software Defined Networking**

Chaithra Naik, Keethana Bhat, Chaitra C Kamath, Rukmini B

Shri MadhwaVadiraja Institute of Technology & Management,Bantakal-574115,Udupi Dist,Karnataka

**ABSTRACT**: **In this current era, usage of the internet is increasing drastically. Digitization has led to high network traffic which makes the overall management of network highly complex and expensive as traditional networks are non programmable. As a solution for these issues in traditional networks, Software Defined Network (SDN) has been introduced. SDN decouples the data plane and control plane there by making the network programmable. SDN allows network administrators to manage the network services by separating the control plane which is called as the brain of the network by data plane where packet forwarding is done. Load balancing in SDN is done to ensure effective management of resources as per client’s request. Some of the load balancing parameters are throughput, transaction rate, response time and the algorithm used. In this paper, the need for load balancing in SDN is discussed and for load balancing we have used the least connection algorithm with Dijkstra's algorithm.**

**INDEX TERMS**: *Software Defined Network (SDN), load balancing algorithms, openflow, controllers.*

**I. INTRODUCTION**

Software-defined networks are emerging technologies that are dynamic, cost-effective, and customizable, making them ideal for the dynamic nature of today's applications [1]. It is a network mechanism that allows you to control and program networks using certain software applications. Its origins can be traced to a research collaboration between the University of California and Sternford in Berkeley in 2008 [2,3]. Nowadays technologies like cloud computing and Big Data are becoming popular. But with these, the maintenance of computer networks is also becoming more complex and expensive. Recently, to address this issue, a Software Defined Network (SDN) has been proposed, which simplifies network management [2]. The basic idea of ​​SDN is to control and separate data. Control planes in a software-defined network have a global view of the entire network and are called the brain of the network [4]. The control plane has centralized or distributed controllers to control the connected forwarding elements in the data layer, so that we may have a single point of control. The data plane consists of many forwarding elements like switch,router that are grouped together in some topological order. These devices forward the data packet according to the decision made by the controller. The protocol used for the communication between control plane and infrastructure plane is “OpenFlow”.

Due to the ever-growing avalanche of data, one of the issues to consider is the load balancing problem between multiple resources.Necessary steps have to be taken to balance the load, to increase the efficiency of network resources which in turn improves the QoS (Quality of Service). There are many load balancing algorithms which perform the task of delivering data traffic efficiently among multiple resources [2]. The main purpose of load balancing is to forward incoming client requests and distribute load to various servers using optimized algorithms implemented in Load Balancer (which is implemented in controller). Traditional load balancers are very expensive and flexible hardware. In Software Defined Networking (SDN) it is possible for programmers to design their own strategy of load balancing, which makes load balancers flexible and programmable.

The paper is organized as follows. In section I, A. briefly describes the architecture of Software defined network. Section II contains the related work on various controllers and algorithms of SDN and section III has the information about the system architecture and design. Section IV provides the implementation details like the tools used in the experimentation with the network topology and the algorithms. Lastly section V shows the result of the experiment and section VI concludes the paper.

**A. SDN Architecture**

Figure 1 depicts the architecture of SDN. Software Defined Networks mainly have three layers namely : Application layer, Control layer and Infrastructure (or Data) layer.

****

Figure 1.SDN Architecture

Application layer contains the SDN application which is designed to satisfy the users requirement. These SDN applications are operated by the network administrator. The control layer contains the controller which is called the brain of the SDN architecture. It will control all the operations of the SDN by having the global view of the network. SDN controller uses the Northbound API such as REST API to communicate with the Application layer. It uses Southbound API to communicate with network devices. Infrastructure layer is also known as the data layer. This is a physical layer, where the network devices, such as switches, routers, etc., are placed. The switch's job is to collect the status of the network and send it to the controller. Openflow protocol is one of the most widely used protocols in software defined networks. Openflow protocols are used for the communication between the controller and network devices [5].

**II RELATED WORK**

*C. Fancy et al.* in [6] have compared traditional networks with SDN networks. The SDN networks have shown great advantages in many aspects, but if the load distribution is uneven in the SDN networks, it will greatly affect the performance of the network. The three load balancing algorithms taken into account for analysis purpose are least connection, Round Robin and Weighted Round Robin algorithm. If we observe the throughput analysis, it shows that round robin algorithm could only provide only the least number of transactions. Also, it is evident that the least connection algorithm provides the highest throughput when compared to the Weighted Round Robin and Round Robin algorithms.

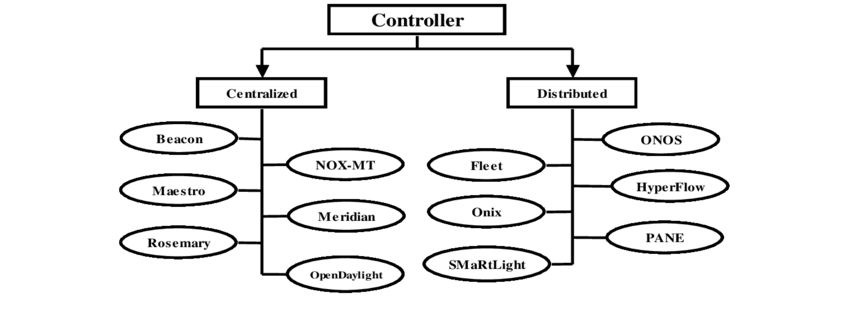
*M. Paliwal et al.* in [7] have compared many controllers of two categories that are centralized and distributed. Below Figure 2 gives the various controllers available.The authors have compared the performance of many controllers based on two parameters throughput and latency. Controller's throughput performance is also evaluated in both centralized and distributed form.

Figure 2: Controller classification [7]

From the survey authors have concluded that the centralized controller provides simplified architecture, efficient handling of request messages. But in this case the network is not scalable and also if the controller fails then the entire network will fail. Distributed controllers perform well at scalability issues and give maximum throughput with high availability but they require proper message exchange procedure in the cluster. Selection of controllers depends upon the size, requirement of the organization.

*S.P Wilson et al.* in [8] have explained how web services are facing difficulty. Now the web services are accessed by many of the users and maintaining them is a real problem. They have also introduced the concept of dynamic clustering but managing the traffic in that is difficult. The load balancing algorithms will give an efficient way to manage the network traffic. Here they have explained the SDN architecture with the static and dynamic load balancing algorithms and also how these algorithms will perform in the SDN. The authors have proposed a load balancing algorithm with the SDN architecture. The algorithm is called the DServe-LB algorithm. Dynamic server load balancing algorithm has been applied in the SDN architecture with the openflow switch, sFlow protocol and POX as a controller. This paper mainly shows the comparison between three algorithms. Those are a random algorithm, Round Robin algorithm and the DServe-LB algorithm. It is proved that the DServe-LB algorithm will choose the best server for the incoming request in less time than the other load balancing algorithm.

*Hailong Zhang et al.* in [9] have explained how open flow works in SDN and also the comparison between the Round Robin and least connection algorithm. The Round Robin algorithm does not consider the actual status of the server connected to the network.Since the Round Robin algorithm allocates the load in circular manner, it does not make full utilization of the resources. So it is predicted that it may not give the real load balancing effect than the least connection algorithm. The experiment is done using a Linux system with Floodlight controller and Mininet software tool to show the network infrastructure. Here the three experiments have been conducted to prove that the least-connection algorithm method is better than the Round-Robin Algorithm.

*Waleed S et al.* in [10] have compared two least cost finding algorithms Dijkstra’s Algorithm and Bellman Ford Algorithm. It has been shown that the Dijkstra Algorithm gives the least path for the chosen server with the least number of iterations than Bellman Ford Algorithm. Results show that the technique of Dijkstra is more powerful and produces a depletion within the processing time for the route calculation in case of any failure occurrence. The comparison between algorithms in SDN architecture provides the information of failure of taking the path to the resource and recovery solution to it.

**III SYSTEM ARCHITECTURE AND DESIGN**

To address the load balancing issues in software defined networking we are proposing a method which is derived from the existing load balancing methods. Figure 3 shows the architecture of load balancing module. Load balancer accepts the requests from various clients/switches using service IP of the controller and assigns them to the servers.

****

Figure 3: Load balancing Architecture [11]

Figure 4 presents basic processes when a packet is transmitted from a client to a server.

1. A client sends packets with a real IP address A to a server with an IP address B.

Figure 4: The operation of SDN (controller-switch)

1. When the first packet arrives to an SDN-aware device,it is immediately checked for a match in the flow table.
2. Control is immediately forwarded to an SDN controller if there is no match for the entry in the flow table.
3. The controller updates the data tables that contain the match field and action to be performed.
4. Then it forms an open-flow instruction and forwards it to a router.

**IV IMPLEMENTATION DETAILS**

The load balancing algorithm will distribute the load of the network among the available resources and ensures the effective interaction with the user. A selected load balancing algorithm should have useful and suitable properties for the network which will enhance the network ability. Those properties of the algorithm should give higher throughput, higher response time, must have fault tolerance, scalability, high performance, effective resource utilization, and the low overhead.

There are many load balancing algorithms in use such as round robin algorithm, weighted least connection algorithm, weighted round robin algorithm and least connection algorithm etc. In [9] three experiments have been conducted to prove that the least-connection algorithm method is better than the Round-Robin Algorithm. So in our research we used the least connection algorithm to select the best server to allocate the load. The least connection algorithm assigns the new connection to that server with least number of active connections. This is one of the dynamic scheduling algorithms, because it needs to count the number of connections for each server dynamically to estimate its load. The load balancer records the connection number of each server, increases the connection number of a server when a new connection is dispatched to it, and decreases the connection number of a server when a connection finishes or timeouts.

To make the load balancing more efficient we used Dijkstra’s algorithm to find the shortest path from client to the selected server.This algorithm will find the shortest path to the host. When we get a host name from the load balancing algorithm, we apply Dijkstra's algorithm to find the shortest path to reach that host so that it can lead to least connection time and less packet loss.

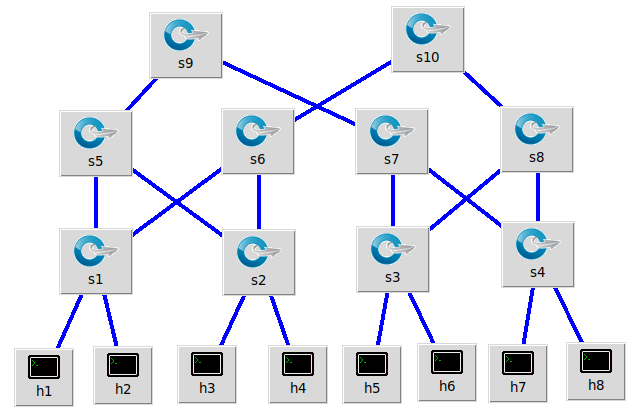
**A. EXPERIMENTAL SETUP**

**Mininet**

Mininet is a network emulator which creates a virtualized network and that network resembles the real hardware system. This emulator operates on a linux and it runs the collection of virtualized hosts, switches and controllers in a single system [8].

We have used mininet because creating a simple network takes only a fraction of seconds, so it means that running or debugging in the mininet is very quick. The simple or complex topology can be created using mininet and those network topology will be the replica of the real hardware system and the switches created in the mininet are programmed by openflow protocol. This program can be directly transferred to the hardware system for packet forwarding.

The Figure 5 shows the network topology created using miniedit. The miniedit is the additional feature provided by the mininet where the network can be created by drag and drop. The load balancing is applied for the shown figure. In our work we have taken h1, h2, h3 as the server and the rest of the hosts will behave as the clients. The network also has the OpenFlow10 switches with a controller, where it will have the global view of the network.

 Figure 5: Network Topology

**POX Controller**

POX is a python based open source development platform for software defined network controller. It is a tool for research in SDN.

We may require many packages for the controller to work. POX comes with a stock of packages bundled with it and it allows fast development and prototyping. So we used POX as our controller.

****Figure 5: methodology of the load balancing in software defined network

Figure 5 shows the methodology of load balancing in a software defined network which has mainly four phases in it.

1. Initially create virtual network topology using mininet.
2. Start the POX controller in another terminal.
3. Converting some hosts to simple HTTP servers.
4. Check whether all the servers are alive by sending ARP packets to all the servers.

|  |
| --- |
| Algorithm 1 : Least Connection Algorithm |
| 1. Let S = {S0, S1, ..., Sn-1} be the set of ‘n’ Servers 2. W( Si ) is the weight of server Si for i = 0,2,3,......n-1 3. C( Si ) is the current connection number of server Si , 4. for each server Sm where m = 0,1,2,.....,n-1 5. if ( W( Sm ) > 0 ) 6. for (i = m+1; i < n; i++) 7. if ( W( Si ) <= 0) 8. continue; 9. if ( C( Si ) < C( Sm )) 10. m = i; 11. end for 12. return Sm 13. end if 14. end for 15. return NULL |

1. When a request packet is received from client use Algorithm 1 to find out the server with the least number of active connections.
2. Use Dijkstra’s algorithm as in Algorithm 2 to find the least cost path from client to selected server.

|  |
| --- |
| Algorithm 2 : Dijkstra(Graph, source) |
| create vertex set Q  for each vertex V in Graph:  dist[V] ← INFINITY  prev[V] ← NULL  add V to Q  dist[source] ← 0  while vertex set Q is not empty:  u ← vertex in Q with min dist[u]  remove u from Q  for each neighbor v of u: // only v that are still in Q  alt ← dist[u] + length(u, v)  if alt < dist[v]:  dist[v] ← alt  prev[v] ← u  return dist[], prev[] |

1. Push the flow rules to the flow table of switches which are included in the least cost path.

**V RESULT**

We have created the fat tree topology using the mininet emulator as shown in Figure 6. Network has 8 hosts in which three hosts(h1, h2, h3) acts as HTTP server and remaining as clients. When a request is sent from client h8 using service IP the best server with the best path will be allocated as shown in Figure 7.

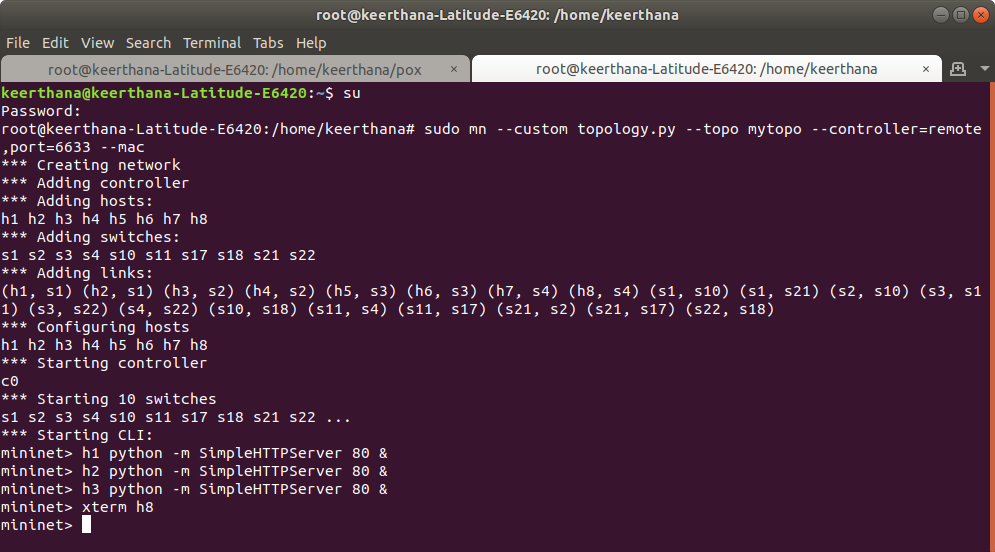
****

Figure 6 : Creating Network Topology

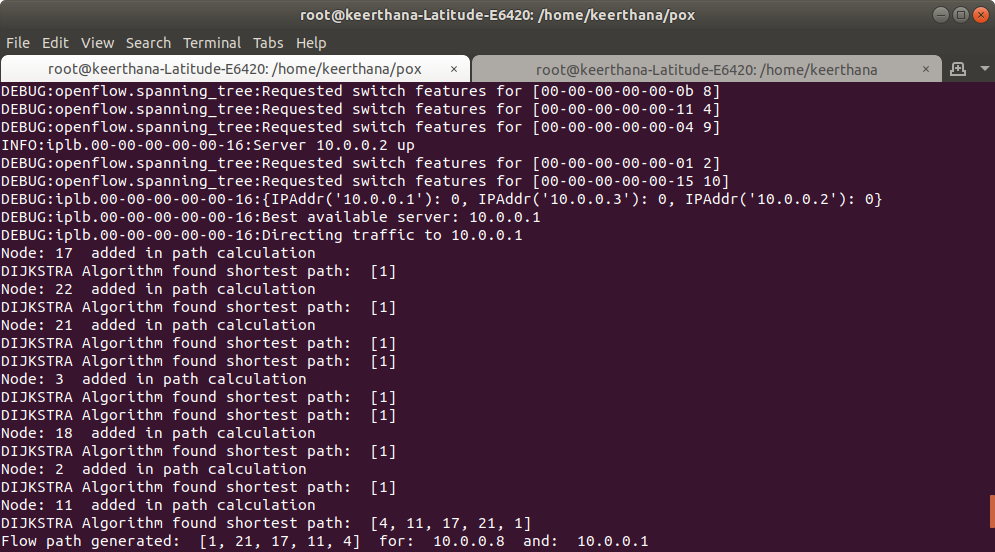
****

Figure 7: Sending request to the server with least connection through shortest path

**VI CONCLUSION**

In this digitization era, it is difficult to control huge network traffic by conventional networks. Handling traditional networks is expensive and complex. So we can say that in the future Software Defined Network will be the problem solving toolset. In this paper we have proposed how load balancing in Software Defined Network is done by using Least Connection algorithm and Dijkstra’s algorithm to find shortest path from client to the selected server with the POX controller.

**REFERENCES**

[1] S. Raghul, T. Subashri and K. R. Vimal, "Literature survey on traffic-based server load balancing using SDN and open flow," 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, 2017, pp. 1-6. doi: 10.1109/ICSCN.2017.8085416

[2] A. A. Neghabi, N. Jafari Navimipour, M. Hosseinzadeh and A. Rezaee, "Load Balancing Mechanisms in the Software Defined Networks: A Systematic and Comprehensive Review of the Literature," in *IEEE Access*, vol. 6, pp. 14159-14178, 2018. doi: 10.1109/ACCESS.2018.2805842

[3] S Bhandarkar, KA Khan “Load Balancing in Software-defined Network (SDN) Based on Traffic Volume” Adv Comput Sci Inf Technol (ACSIT), 2015 online- http://www.krishisanskriti.org/ACSIT.html

[4] Subramanian, R. & Manoranjitham, T. ”Dynamic scheduling for traffic management and load balancing using SDN” in IJCTA, 9(2), pp. 1259-1265. 9. 919-925, 2016.

[5] Wenfeng Xia, Yonggang Wen, Senior Member, IEEE, Chuan Heng Foh, Senior Member, IEEE, Dusit Niyato, Member, IEEE, and Haiyong Xie, Member, IEEE, “A Survey on Software-Deﬁned Networking” , IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 1, FIRST QUARTER 2015 .

[6] C. Fancy, M. Pushpalatha, Pushpa, “ Experimentation of Traditional Load Balancing Algorithms in Software Defined Network”, ISSN: 2277-3878, Volume-8, Issue-1S4, June 2019 [Online]. Available: ijrte.org/wp-content/uploads/papers/v8i1s4/A10970681S419.pdf

[7] M. Paliwal, D. Shrimankar and O. Tembhurne, "Controllers in SDN: A Review Report," in *IEEE Access*, vol. 6, pp. 36256-36270, 2018.doi: 10.1109/ACCESS.2018.2846236

[8] Wilson Prakash S , Deepalakshmi P, “ DServ‐LB: Dynamic server load balancing algorithm”, John Wiley & Sons, Ltd 2018 , DOI: 10.1002/dac.3840.

[9] Hailong Zhang and Xiao Guo, "SDN-based load balancing strategy for server cluster," *2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems*, Shenzhen, 2014, pp. 662-667. doi: 10.1109/CCIS.2014.7175817

[10] Waleed, S., Faizan, M., Iqbal, M., & Anis, M. I. (2017). “Demonstration of single link failure recovery using Bellman Ford and Dijkstra algorithm in SDN”. 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT). doi:10.1109/icieect.2017.7916533

[11] Anam Ikram, Subhan Arif, Nafees Ayub, Waqas Arif, “Load Balancing In Software Defined Networking (SDN) ” ,MAGNT Research Report (ISSN. 1444-8939) ,Vol.5(1). PP. 298-305, 2018 doi:1444-8939.2018/5-1/MRR.33.