Traffic Based Load Balancing in Software Defined Networking

Harkirat Kaur
Department Of Computer Science and Engg.
NorthWest Group of Institutions, Dhudike, Moga, Punjab
harkirat.gill92@gmail.com

Navjot Jyoti
Department Of Computer Science and Engg.
NorthWest Group of Institutions, Dhudike, Moga, Punjab
navjotkaurdahien@gmail.com

***Abstract*—This Single server is unable to handle clients requests with the fast growth of internet users and applications. Our network should be able to cope with large volume of traffic without introducing unnecessary delay. One solution to this problem is load balancing for network scalability and service availability. Load Balancer is a device that distribute the traffic among number of servers. But traditional vendor specific Load Balancer is a device that is much more expensive. Internet user can not change the functionality of that device means traditional Load Balancer are non programmable. In the network single Load Balancer become a single point of failure or sometimes in large network it become bottleneck. Software defined networking is a promising solution to all these disadvantages. In Software Defined Networking architecture, control and data plane are to be separated. We can convert a dumb openflow switch into Load Balancer by writing a program or application and run application at control plane. In this paper we written a traffic based Load Balancer program and run on controller named POX. With this program we are able to convert the dumb switch into Load Balancer. Mininet emulation tool is used for experiment evaluation. SDN based Load Balancer is dynamic, less expensive and programmable as compared to traditional Load Balancer.**

Keywords- Traffic based load balancing, Mininet emulation tool, OpenFlow protocol, Software Defined Networking (SDN), POX. Python.

I. Introduction

Social network, social web sites are the online services that has a high volume of traffic so these networks have to able to handle large volume of traffic per day. Web server is unable to handle such a large amount of data so administrator have to face many problems regarding web server capacity. There are two solutions are available. First, we can increase the capacity of single web server. Second solution is that we have multiple web server that working collectively as a single web server. By this method services are replicated on all servers and we can increase reliability. These number of servers are called server pool or cluster of servers. In this technology, incoming requests are forwarded among number of servers in transparent way [1].

Load balancing scheme must be adopted to properly utilize the resources of a number of servers. The chosen scheme has a Load Balancer entity doing the load balancing and purpose of algorithm is select the server which handle the client request. In dispatcher-based load balancing, dispatcher or Load Balancer setup at the ingress port of the network that receive the inputs from a client and distributing the client requests among number of servers. Load Balancer is a front end of server. The scheduling policy is also run at Load Balancer to select a server where client request is forwarded. [2] The scheduling policy has a important role in load balancing architecture. If the scheduling policy select a correct server then we can increase throughput of cluster and decrease response time.

Our distributed architecture allows that we can easily scale the size of network. It is necessary when we have to add new servers in the clusters to cope the services of clients. Another advantages of adding new servers in the network is availability. If one server fail, another server handle the traffic of that server.

In our paper, We used POX controller to implement Traffic Based Load Balancing strategy. Various tasks are performed in this paper are as follow.

- Created "Traffic Based Load Balancing Strategy".
- Mininet Emulation Tool is used to test this algorithm.
- Openload load testing tool is used to measure the performance parameters.
- We compared our "Traffic Based Load Balancing" using SDN with previously implemented round robin strategy. The measured parameters are Transactions per second, Total number of request, average response time.

In this paper, total VI sections are present. Second section describes what is SDN and Openflow protocol, Third section explaining the work that has already done. Fourth section discuss about load balancing algorithm, Fifth section elaborate about experiment topology and evaluation of experiment with the help of graph and Fifth section mentioned what is conclusion of paper and future work that we have done.

II. SOFTWARE DEFINED NETWORKING ARCHITECTURE

IT departments have unable to get a best performance form network because traditional network devices such as routers, switches, Load Balancer, firewalls have limited functionality. But the demands of network users are increasing day by day. They have to configure each and every device individually, sometimes the total number of devices can be thousand. IT professionals have also developed large number of protocols to increase the performance of network. To setup and configure of these protocols and devices are very tedious task. For example, after adding and removing the device from the network we have to configure switches, Access control list (ACL), routers, firewalls. It is very complex, long, error prone and tedious task [3].

Forwarding, control and management are three planes in network devices. The main task of data plane is forwarding the data. The main task of SDN controller or control plane is controlling the forwarding plane. Controller defines how should forwarding plane have to behave. This is done using the management plane by imposing the flow rules into the forwarding plane. In these devices, data plane, control plane and management plane are tightly bundled as shown in Fig. 1. If we want to adding new protocol, it must be deployed in every network device. So adding new features in these devices are very difficult task. The complexity and cost of network is increase. The management of this network is also very tedious task.

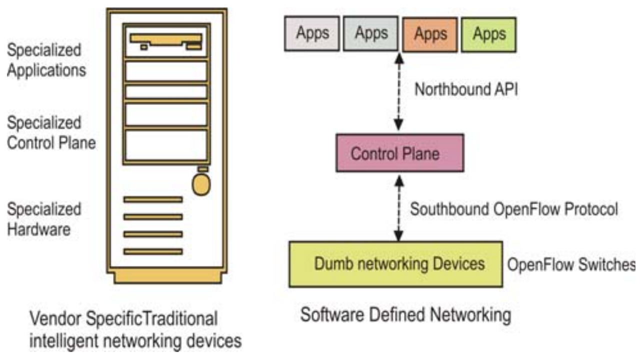
Specialized Applications
Specialized Control Plane
Specialized Hardware
Vendor SpecificTraditional intelligent networking devices
Apps
Apps
Apps
Apps
Northbound API
Control Plane
Southbound OpenFlow Protocol
Dumb networking Devices
OpenFlow Switches
Software Defined Networking

Figure 1. Traditional network vs software defined networking

Software defined networking is a new technology in networks. The main purpose of SDN is to centralize the control plane called openflow controller. We can view or control the whole network from the central place by centralizing the control plane at common place as shown in Fig. 2. In this approach by writing the applications on top of control plane, we can manage the whole network. The configuration is also very easy due to separation of data plane and control plane [4]. We can easily change the network by writing the applications.

Open Flow protocol is used to implement SDN (software defined networking). It is a standard protocol that is used between forwarding and control plane. The messages exchange between control and forwarding plane are defined by openflow protocol [5].

Openflow enabled or native switch, secure channel, Openflow controller are three components of Openflow architecture as shown in Fig. 3. For securely communication between dumb switch and controller, secure channel is used. Each switch contains one or more number of tables. Number of flow rules are present in the table of switch. Each flow rule consists of matching parameter, actions and counters. These parameters define how packets will be handled. Ingress port, Source IP, Destination IP, Source mac, Destination mac are the matching parameters used for matching purpose. Actions specify how packet will be handled means what to do with the packet that are coming from source [6]. Counters are used to count the total packets in the request and total number of flows.

When packet coming from client reach at switch, packet matched against the flow entries of switch. Appropriate action is performed, if all matching parameters are matched. If match is not found, then action is performed according to table-miss entry such as dropping the packet or forward the packet to the controller.

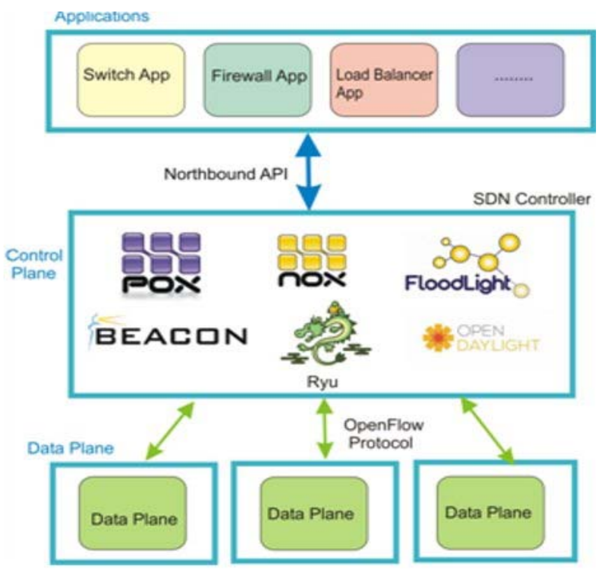

Figure 2. Software defined networking architecture

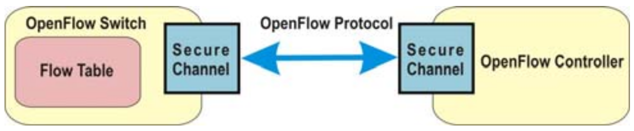

Figure 3. Openflow protocol

Background And Related Work

Before Dispatcher based load balancing implemented by traditional vendor specific network devices. These devices are also provided additional functionalities such as security features and caching. But these devices have many limitations. First, dedicated hardware is very expensive. The configuration of each and every device is also very difficult task for network administrator. The customizability is also not possible because rigid policy is used [7]. SDN Load Balancer remove all these limitations by converting a dumb openflow switch into powerful Load Balancer by simply writing a program on top of control plane.

Uppal and Brandon [8] implemented three algorithms such as random, round-robin and load based algorithm. In random algorithm forward the request to random chosen server from a pool of servers. Main limitation of this strategy is that load is not equal at each server. In round-robin algorithm, forward the request to server in round robin fashion. One after another server is chosen from a pool of servers. Main limitation of this strategy is that it does not take into account server load or capacity. In load based strategy, forward the request to server having lowest load. Lowest load defined by the more number of pending requests.

Kaur et al. [9] also developed round robin load balancing strategy. Main limitation of this paper is that it does not considered server actual load.

The main limitation of all these approaches is unnecessary latency because first request message is pass through the Load Balancer then reply from the server also pass through the Load Balancer. But our algorithm use direct routing from server to client in return message.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

III. Load Balancing Algorithms

To properly utilize the server resources, load balancing is a mechanism that divide the total traffic from clients among a number of servers. By using this technology we can increase throughput, minimize response time and minimize overload on each server. In a dispatcher based strategy there is a pool of servers that are available to the client as a single resource with IP address 10.0.0.254. This is a virtual IP of dispatcher that act as a central point between clients and pool of servers. When clients requests are come then dispatcher control all the routing decision of requests. There are two approaches for communication between clients and servers 1) if client want to access some online service then client send a request at ip address. The client request reaches at dispatcher (Load Balancer). Then Load Balancer send client request to server machine choosen by policy. Server send a reply back to the Load Balancer and at last Load Balancer forward that reply to the client machine. In this case 4 steps are involved as shown in Fig. 4.

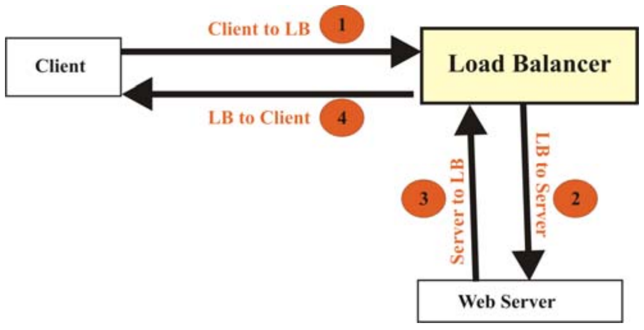

Figure 4. Load balancer architecture

Mostly, in both forward and backward process the Load Balancer is involved that cause extra time to execute the request. In our approach, the Load Balancer is not participated in backward process. In this server machine directly send the reply message to the client machine and eliminate the delay and extra time. This means in return traffic direct routing is used from server to client. In our new approach only 3 steps are involved as shown in Fig.5.

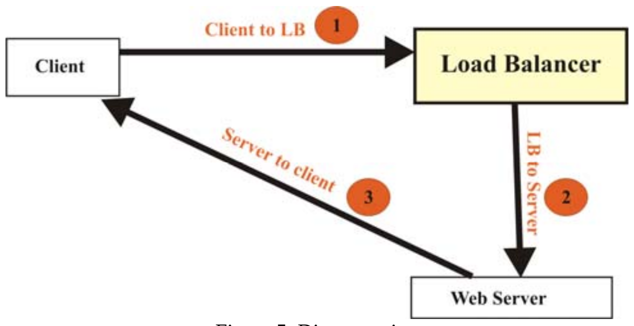

Figure 5. Direct routing

Now we explained the algorithm which defined how Load Balancer select a particular server from a pool of servers. In this approach, the server having less number of packets is chosen. In this strategy after every 2 second, controller send a "flow _statistics_request" to the switch. OpenFlow switch send a reply to the controller. Then "flow_statistics_received" event generated at controller. The event handler in load balancing application count total number of packets handled by each server. After counting, the server with the lowest number of packet is chosen by the Load Balancer machine. Then Load Balancer send a packet to the chosen server.

IV. Experiment Setup And Evaluation

The topology that we use for our experiment consist of 1 POX controller [10], 1 OpenFlow switch, 4 hosts. The host having IP address 10.0.0.4 act as a client, hosts having IP addresses 10.0.0.1, 10.0.0.2, 10.0.0.3 act as servers. The topology was created in Mininet emulation tool [11, 12]. We implemented web server on hosts such as h1 (IP is 10.0.0.1), h2 (IP is 10.0.0.2) and h3 (IP is 10.0.0.3). We installed "openload" testing tool on host h4 having IP address 10.0.0.4. It is a open source tool. We run a traffic based load balancer on SDN controller. Then our Openflow switch act as a load balancer. All the servers are connected with the Openflow switch as shown in Fig 6.

Client will send a request to the IP address of Load Balancer is called virtual ip. In our case virtual ip is 10.0.0.254. Load Balancer will forward the request to server that is chosen with any load balancing policy such as "Round Robin", "Traffic Based Load Balancing". Then server send a reply to the client directly or send the reply again to Load Balancer. Then Load Balancer send a reply to the client machine. In our case when client send a request at virtual ip then Load Balancer send that request to the server having lowest number of packets. Then server send a reply to the client directly. This means in return traffic Load Balancer is not participated in our strategy.

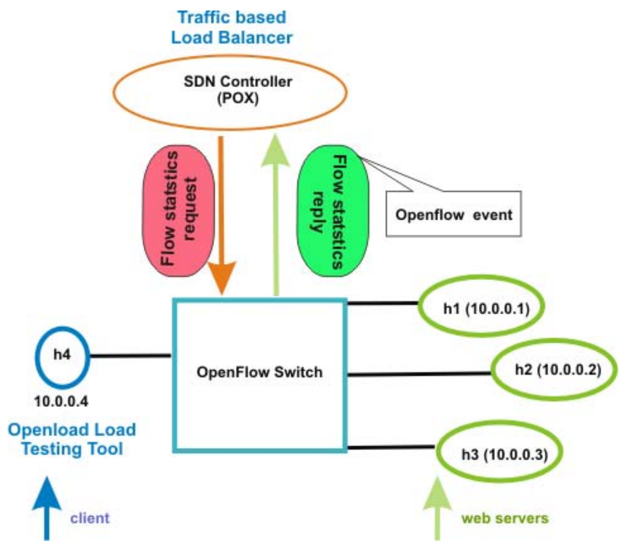

Figure 6. Network topology

We compared our “Traffic based Load Balancing” with “Round Robin load balancing” strategy. We compared the strategies on the basis of response time (RT), transactions per seconds (TPS), total number of requests. The tool to measure these parameters is “openload load testing tool”.

Average response time of servers shown in Fig 7. In a graph y-axis shows average response time, x-axis shows number of concurrent clients. We can clearly observed from graph that average response time (RT) is better in our “Traffic based load balancing” than “round robin strategy”.

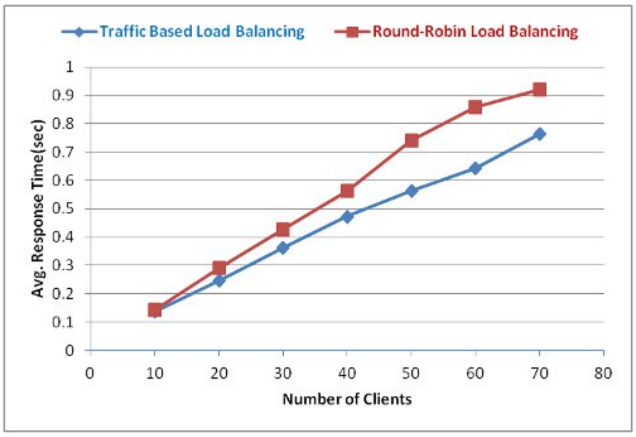

Figure 7. Average response time in “traffic based load balancing” and 'round robin load balancing”

Transaction per second of servers shown in Fig 8. In a graph y-axis shows transaction per second, x-axis

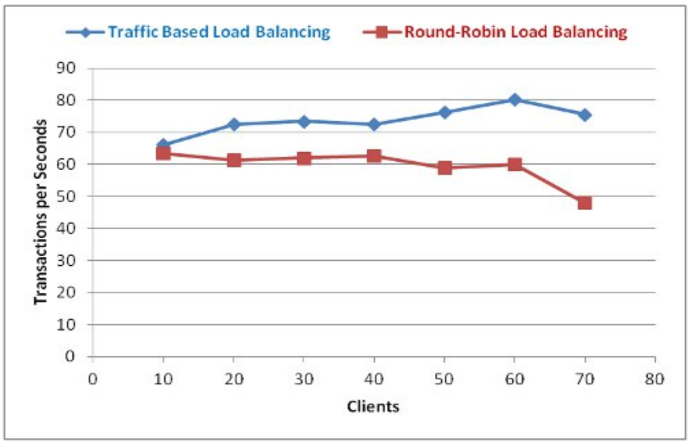

Fig. 8 Transaction per second in traffic based load balancing and round robin load balancing

shows number of concurrent clients. We can clearly observed from graph that transaction per second (TPS) is better in our “Traffic based load balancing “than “ round robin strategy”.

Total number of requests handled by both strategies are shown in Fig. 9. We can clearly observed from graph that our algorithm handle more number of requests than round robin algorithm.

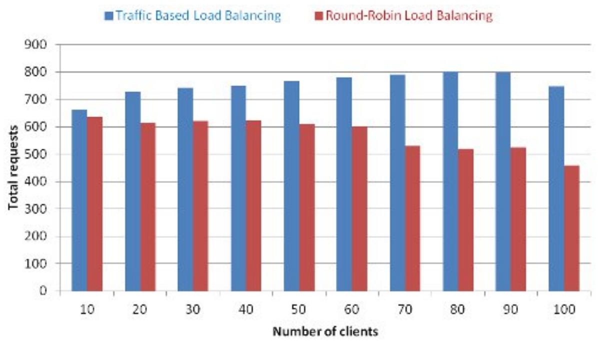

Figure 9. Total number of requests handled by “traffic based load balancing” and “round robin load balancing”

V. Conclusion

Dedicated hardware that is used in traditional Load Balancer is very expensive and inflexible. SDN based Load Balancer remove many limitations of traditional Load Balancer. SDN Load Balancer is very cheaper as compared to traditional Load Balancer because in SDN with programming facility of control plane we can convert a OpenFlow switch (dumb device) into powerful Load Balancer. Another advantages of such Load Balancer is flexibility means we can change the functionality as per requirements. Main purpose of this paper is to build a load balancing application based on traffic. We also verify that it is very easy to create new applications in SDN network. We created Traffic based Load Balancer that use direct routing approach. We compared our algorithm with Round Robin algorithm. At the last we conclude that our Load Balancer is better in performance parameter as compared to Round Robin strategy. Single point of failure is limitation of our paper because we have use only one controller. So in future we can use more controller instead of one. In this case failure of one controller, the functionality of that controller is handled by another controller.

References

[1] Ghaffarinejad, Ashkan, and Violet R. Syrotiuk. "Load balancing in a campus network using software defined networking." In Research and Educational Experiment Workshop (GREE), 2014 Third GENI, pp. 75-76. IEEE, 2014.

[2] Zhou, Yuanhao, Li Ruan, Limin Xiao, and Rui Liu. "A method for load balancing based on software defined network." Advanced Science and Technology Letters 45 (2014): 43-48.

[3] Nunes, Bruno Astuto A., Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of software-defined networking: Past, present, and future of programmable networks." IEEE Communications Surveys & Tutorials 16, no. 3 (2014): 1617-1634.

[4] Xia, Wenfeng, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. "A survey on software-defined networking." IEEE Communications Surveys & Tutorials 17, no. 1 (2015): 27-51.

[5] Ramos, Fernando MV, Diego Kreutz, and Paulo Verissimo. "Software-defined networks: On the road to the softwarization of networking." Cutter IT journal (2015).

[6] Braun, Wolfgang, and Michael Menth. "Software-defined networking using OpenFlow: Protocols, applications and architectural design choices." Future Internet 6, no. 2 (2014): 302-336.

[7] Ghaffarinejad, Ashkan. "Comparing a commercial and an SDN-based load balancer in a campus network." PhD diss., Arizona State University, 2015.

[8] Uppal, Hardeep, and Dane Brandon. "OpenFlow based load balancing." University of Washington. CSE561: Networking. Project Report, Spring (2010).

[9] Kaur et al. "Round-robin based load balancing in software defined networking." In Computing for Sustainable Global Develop. (INDIACom), 2015 2nd Int. Conf. , pp. 2136-2139. IEEE, 2015.

[10] Kaur, Sukhveer, Japinder Singh, and Navtej Singh Ghumman. "Network programmability using POX controller." In ICCCS International Conference on Communication, Computing & Systems, IEEE, no. s 134, p. 138. 2014.

[11] de Oliveira, Rogério Leão Santos, Ailton Akira Shinoda, Christiane Marie Schweitzer, and Ligia Rodrigues Prete. "Using mininet for emulation and prototyping software-defined networks." In Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on, pp. 1-6. IEEE, 2014.

[12] Wang, Shie-Yuan. "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet." In Computers and Communication (ISCC), 2014 IEEE Symposium on, pp. 1-6. IEEE, 2014.