Round-Robin Based Load Balancing in Software Defined Networking

Sukhveer Kaur
SBS State Technical Campus
Ferozepur, India
Email Id: bhullarsukh96@gmail.com

Japinder Singh
SBS State Technical Campus
Ferozepur, India
Email Id: japitaneja@gmail.com

Krishan Kumar
SBS State Technical Campus
Ferozepur, India
Email Id: k.salujasbs@gmail.com

Navtej Singh Ghumman
SBS State Technical Campus
Ferozepur, India
Email Id: navtejghumman@yahoo.com

Abstract – These days our networks have to handle large amount of traffic, serve thousands of clients. It is very difficult for a single server to handle such huge load. The solution is to use multiple servers with load balancer acting as a front end. The clients will send the requests to the load balancer. The load balancer will forward the client requests to different servers depending upon load balancing strategy. Load balancer use dedicated hardware. That hardware is expensive and inflexible. Currently available load balancers contain few algorithms that can be used. Network administrators can not create their own algorithms since traditional load balancer are vendor locked, non programmable. On the other hand SDN load balancers are programmable and allow you to design and implement your own load balancing strategy. Other advantages of SDN load balancer is we do not need dedicated hardware. The dumb silicon device can be converted to a powerful load balancer by using SDN controllers. In this paper we are implementing and comparing Round-Robin load balancing strategy with already implemented random strategy using an OpenFlow switch connected to a POX controller.

Keywords – Software Defined Networking, OpenFlow, Mininet, Load Balancing, POX.

NOMENCLATURE

Software Defined Networking (SDN), Python OpenFlow controller (POX). Transactions Per second (TPS), Response Time (RT), Hyper Text Transfer Protocol (HTTP)

I. INTRODUCTION

Software Defined Networking (SDN) has the potential to enable innovation in the evolution of computer networks. It is based on the principle of separating the control and data planes. In traditional networks, switches, routers, firewalls, load balancers are specialized vendor specific hardware devices having tightly coupled data and control plane. The functionality of these devices can not be changed dynamically [1,2]. In SDN, data planes are cheap commodity silicon devices. Decoupled Control plane will use OpenFlow protocol to communicate with data plane. On top of control plane, network applications such as switching, routing, firewall and load balancer can be implemented by using combination of flow table entries (Fig. 1).

Controller independently sets up every flow in per flow based routing. There is one entry per flow in flow table. Exact match is performed for flow entries. It is a requirement for fine grain control. One flow entry can covers a large group of flows in

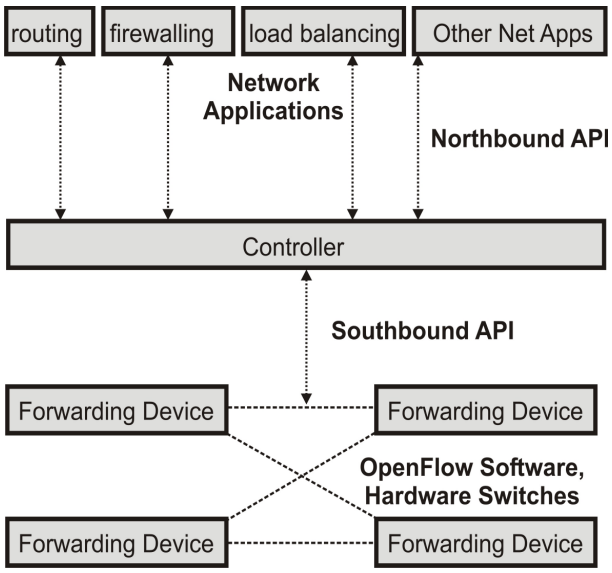

Fig. 1 SDN Architecture

aggregated flow routing. Each category of flows contains one flow entry. It uses wildcard flow entries for aggregated flow based routing

The OpenFlow specification [3,4] describes the information exchange between the two planes. In this architecture, an OpenFlow contains a flow table consisting of flow table entries. A flow entry is made up of fields on which incoming packets are matched, and action to applied upon a match. If there is no match, the packet is forwarded to a controller, which runs a program to handle the packet, and decides whether to insert, delete, or update flow entries in the flow table for subsequent packets matching the same fields. Statistics are collected on packet which may be used to make decisions.

Currently networking traffic load is very heavy and growing very rapidly. Server overload and network congestion are major problems. Online services such as web sites, search engines and social networking sites require multiple servers to offer enhanced capacity and high reliability. Load balancer is used to distribute requests to multiple servers. Traditional load balancers are implemented on dedicated hardware, very costly, non-programmable. Our load balancing implementation does not require separate load balancer device and allows for flexibility of network topology. Our solution can be scaled as the number of switches and servers increases, while handling client requests at line rates.

Load balancing is a method used for spreading requests across multiple services. This is done using highly expensive vendor specific hardware devices. Load balancer helps in enhancing performance of networks by properly utilizing available resources. The purpose is to minimize response time, latency and increasing throughput. They distribute traffic based on random, round-robin, current server loads, and location based request strategies. Our Contribution in this paper is

- Creating Round-Robin Load balancing code.
- Tested our load balancing application using Mininet Emulator
- Comparing Round Robin with Random Strategies Based on different load using OpenLoad and measuring Response Time and Transaction per Seconds.

The outline of our paper is as follow. Section II contains load balancing related work. Section III contains load balancing architecture. Section IV describes implementation details. Section V covers Experimental Evaluation & Section VI contains conclusion and future work.

II. RELATED WORK

Load Balancing is a technique used to distribute large number of requests across multiple devices. These load balancer devices are very expensive, specialized and vendor-specific devices. Load balancer increases network performance by properly using the available resources and helps in improving response time and transactions per second.

Richard et al. [5] discusses about load balancing strategy that distribute traffic based on to load balancing weights assigned. The limitation of this paper is that it introduces large overhead. Marc et al. [6] discusses about multiple load balancing where one load balancer is used for balancing web servers while other controller is used for balancing e-mail servers. Our work involves implementing round-robin load balancing strategy & comparing it with random based load balancing strategy.

III. LOAD BALANCING ARCHITECTURE

The load balancing architecture consists of OpenFlow switch network with a POX [7,13] controller and multiple servers connected to the ports of the OpenFlow switch. Each server is assigned static IP address and the POX controller maintains a list of live servers that are connected to the OpenFlow switch. Web service is running on each server on a well known port 80 (Fig 2).

Controller contains a virtual address. All requests from the clients are sent to the virtual IP address. When the client sends a request packet to the virtual IP, OpenFlow switch uses the

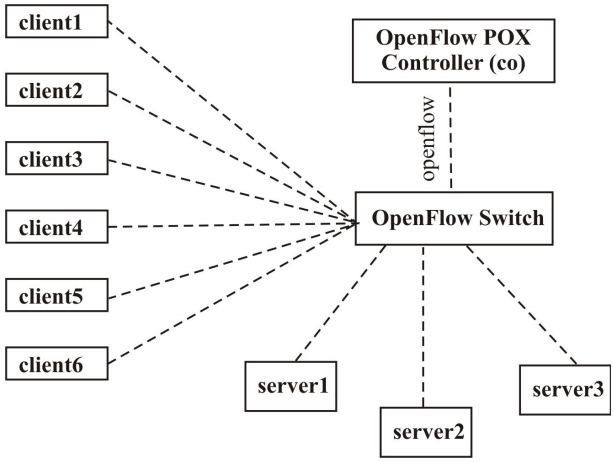

Fig. 2 Load Balancing Architecture

information contained in packet header and compare it with flow entries in switch, if the client packet header information matches up with the flow entry, then switch modifies the destination virtual ip address to the address of one of the servers based on load balancing strategy and forward the packet to that server, If packet does not matches with any flow entry, the switch will forward this packet to the POX Controller. The Controller will decide how to deal with the packet. The Controller adds flow entries to the switch through using OpenFlow [8]. The various load balancing strategies are:

A. Random Strategy:

The Load Balancer randomly selects a server from a list of servers [9].

B. Round Robin Strategy:

In this, the requests are equally distributed to the servers. The requests are assigned on round robin base [10].

C. Load Based Strategy:

The load balancer selects the server with minimum load on receiving new request.

IV. IMPLEMENTATION

In order to implement the functionality of the load balancer, the following softwares were used:

A. VMPlayer:

It is used for running virtual machines.

B. Mininet:

Mininet emulator is used for creating complex networks on the limited resources of a Virtual Machine[11,12].

C. POX:

POX Controller is Python based open source Software Defined Networking (SDN) Controller. This controller is used for rapid development of new networking applications.

D. Openload:

Open source Tool for load testing of web applications. This tool is used for measuring Transactions per seconds (TPS) and average response time (RT). TPS is numbers of requests completed in a second while average response time is the time taken by web server to respond to user's request.

Our Mininet topology consists of 1 OpenFlow switch, 12 hosts and 1 Pox controller. Web server was implemented on hosts h1, h2 and h3. h4 to h12 hosts were used as testing hosts. The testing was performed using open source tool called "Openload". Our load balancing implementation uses OpenFlow switch along with POX controller and three server machines connected to the ports of the OpenFlow switch as shown in the Fig. 3.

Our Load Balancer contains two types of addresses namely service IP and server IP's. Client will send the requests to the service IP. Service IP is the address given to load balancer. Load Balancer will redirect the request to the server depending upon load balancing strategy. It could be random, round-robin or load based. In our case, we are comparing random and round-robin load balancing strategies by measuring transactions per seconds (TPS), Response Time (RT). In load balancing, we are actually load balancing service IP. The other important point in implementing load balancer is to perform check whether the server is alive or not. We implement this by sending ARP probes to servers.

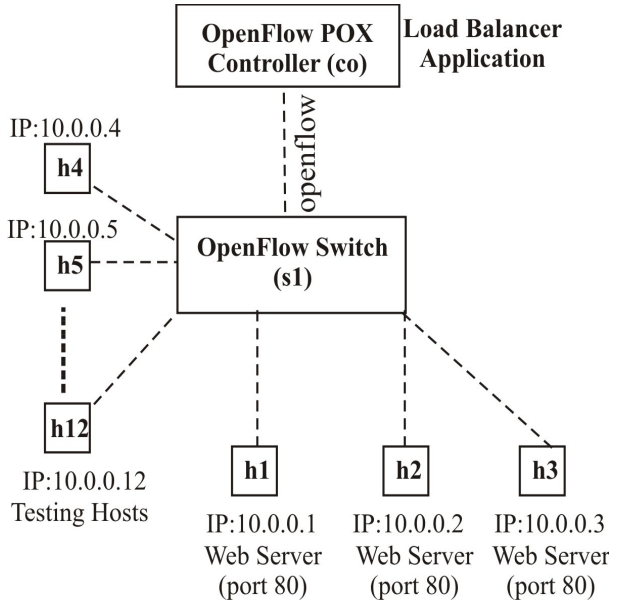

Fig. 3 Network Topology

V. EXPERIMENTAL EVALUTION

We compared the results of two load balancing strategies on the basis of total transactions per second and average response time of the web server. Hosts h4 to h12 were used as testing hosts. The tool used was "Openload". In our case, we took the readings by sending different type of load from hosts h4 to h12. As shown in Fig. 4 and Fig. 5. Transactions per Second and Response Time are better in case of round-robin strategy. The other notable feature of the results is that round-robin was

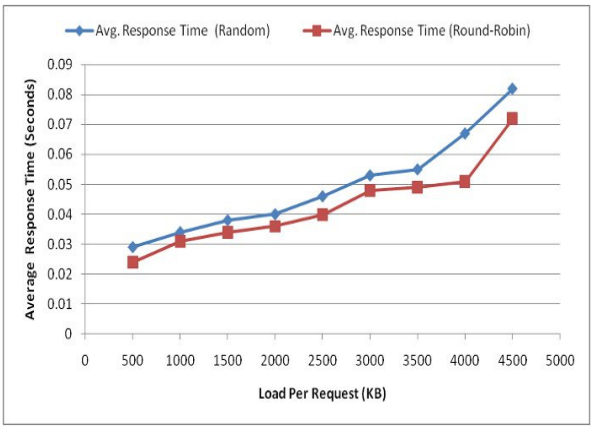

Fig. 4 Average Response Time in Random and Round-Robin Strategy

uniformly distributing the load while there was difference in requests send in case of random strategy.

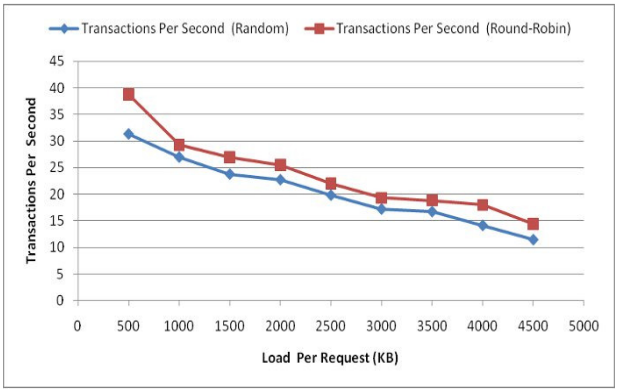

Fig. 5 Transactions Per Second in Random and Round-Robin Strategy

VI. CONCLUSION AND FUTURE SCOPE

In this paper, we present a strategy for load balancing based on SDN. SDN load balancer solves many problems of traditional load balancers. We implemented load balancer using round robin strategy and compared it with already implemented random load balancer strategy. Results show that round robin strategy is better than random strategy. The other positive point about round-robin is that it is actually distributing the load uniformly while random is not. Limitation of our work is that we have not testing code on real hardware which might generate different results. Other Limitation is that code was tested using POX Controller. We did not use other controllers. SDN application's performance is also dependent upon the performance of controller. This could be area for future research. One Major drawback in both round robin & random is that they do not take into consideration the current load on servers. Future works can involve load balancing based on server load using flow statistics.

ACKNOWLEDGEMENT

We thanks Mr. Vipin Gupta of U-Net Solutions, Moga for his valuable contribution.

REFERENCES

[1] Nunes, B., Marc Mendonca, X. Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of software-defined networking: Past, present, and future of programmable networks." (2014): 1-18.

[2] Feamster, Nick, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks." *ACM SIGCOMM Computer Communication Review* 44, no. 2 (2014): 87-98.

[3] Lara, Adrian, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey." (2013): 1-20.

[4] Suzuki, Kazuya, Kentaro Sonoda, Nobuyuki Tomizawa, Yutaka Yakuwa, Terutaka Uchida, Yuta Higuchi, Toshio Tonouchi, and Hideyuki Shimonishi. "A Survey on OpenFlow Technologies." *IEICE Transactions on Communications* 97, no. 2 (2014): 375-386.

[5] Wang, Richard, Dana Butnariu, and Jennifer Rexford. "OpenFlow-based server load balancing gone wild." (2011).

[6] Koerner, Marc, and Odej Kao. "Multiple service load-balancing with OpenFlow." In *High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on*, pp. 210-214. IEEE, 2012.

[7] Kaur, Sukhveer, Japinder Singh, and Navtej Singh Ghumman. "Network Programmability Using POX Controller."

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

[9] Shang, Zhihao, Wenbo Chen, Qiang Ma, and Bin Wu. "Design and implementation of server cluster dynamic load balancing based on OpenFlow." In *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*, pp. 691-697. IEEE, 2013.

[10] 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.

[11] Kaur, Karamjeet, Japinder Singh, and Navtej Singh Ghumman. "Mininet as Software Defined Networking Testing Platform."

[12] Lantz, Bob, Brandon Heller, and Nick McKeown. "A network in a laptop: rapid prototyping for software-defined networks." In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19. ACM, 2010.

[13] POX at https://openflow.stanford.edu/display/ONL/POX+Wiki