ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

LDDWRR: Least Delay Dynamic Weighted Round-Robin Load Balancing in Software Defined Networking

Manamrit Singh Sroya
Computer Science and Engg.
North West Group of Institutions
Moga, Punjab, India

Vikramjit Singh
Computer Science and Engg.
North West Group of Institutions
Moga, Punjab, India

Abstract— In traditional network devices, tight coupling of data plane with the control plane so large amount of traffic on the web is not correctly handled by the Traditional network devices. The emerging strategy that decoupled the data plane from control plane is called Software Defined Networking (SDN). Control plane of these devices act like a brain or master. It is also called Software Defined networking controller or Openflow controller. Data plane is called slave that behave according to the instructions given by the master or control plane. To convert a data plane into powerful network devices we have to create Firewall, Load balancer, Intrusion Detection system applications and then run that applications at management plane that is on top of control plane. In such a manner we can convert data plane into powerful network devices according to the application. The application that we have created on the top of control plane is load balancer application that convert the dumb data plane into load balancer. Load balancer architecture consists of number of servers and clients which distributed the client traffic among number of servers based on particular strategy. There are number of strategies are available such as Round Robin, random policy. Each load balancing policy has some advantages and disadvantages. We created "least delay dynamic weighted round robin (LDDWRR)" strategy in this paper and run on the top of SDN controller. Then we evaluate the result of our load balancing strategy by comparing with the round robin strategy. Mininet is used for the experiment and controller that is used as control plane is called POX controller.

Keywords— OpenFlow, Load Balancer, POX, Mininet, Software Defined Networking.

I. Introduction

Internet is overloaded with large amount of traffic in these days. The web sites and other online services have to manage large volume of traffic per day. The administrators of web sites have to handle large number of problems because the capacity of web server is not sufficient. To handle the large number of requests, we can use large number of servers instead of one. Then all these multiple servers act as a single server collectively. Online Services should be copied into all servers. Incoming requests from the multiple clients are distributed among the servers by Load Balancer. Load balancer in traditional network are basically use dedicated or specific hardware so these type of load balancers are very costly. Second disadvantage of such devices are that these are non programmable means vender specific. Vendor write the code and there is a long delay to introduce new features. But in SDN you can write a one load balancer application or program in programming language and then run that application on controller or control plane. By using this, we are able to change a dumb openflow switch into high powered load balancer [1]. The language in which we write the program depend upon the controller. If we use POX or Ryu controller then write a code in python language. If we use opendaylight controller then write a code in java language.

Load balancer basically act between client and multiple server. In transparent manner, load balancer distribute the clients load among number of servers. Client does not directly interact with the server [2]. This is the responsibility of load balancer that according to the program forwards the client requests to the server. In backword process first web server send the reply to the load balancer and then load balancer forward that reply to client.

In this paper, we implement Least Delay Dynamic Weighted Round-Robin Load Balancing. POX Controller is used in this implementation. The main objectives that we fulfill in this paper are as follow:

- First created Least Delay Dynamic Weighted Round Robin Load Balancing algorithm (LDDWRR).
- Mininet emulation tool is used for test the.LDDWRR algorithm
- At last we Compared LDDWRR algorithm with previous implemented Round-Robin load balancing algorithm.
- To generate a load siege load balancing tool is used.

The outline of paper consist of many sections are as follow: section II explaining what is software defined networking, section III described about related work, section IV described experiment setup, section V contains Evaluation of experiment and conclusion of paper and future work that we can do described in section VI.

II. Software Defined Networking

Traditional network devices are non programmable so any type of modification and configuration in such devices are very difficult task and vender specific as shown in Fig. 1. Traditional network devices are hardware devices. If we want any type of modification, only vendor can do this task. So user cannot modify the device as per requirements. The time and cost for creating a small network is very high due to data plane and control plane tightly coupled. The limitations of traditional network is removed by Software defined network by separate the data plane and control plane. Due to separation of both planes allows us easier development of new applications such as firewall, load balancer. Using these applications we can convert data plane into powerful devices [3]. SDN introduces the innovation in network. Now user can change the network according to the requirements.

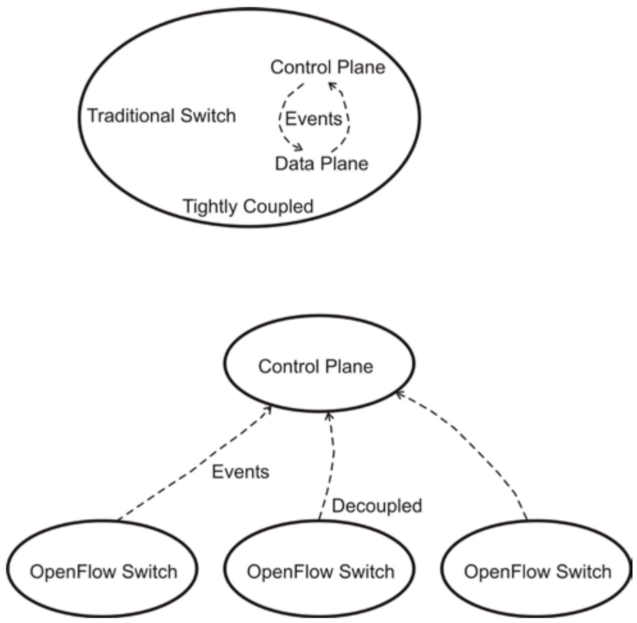

Fig. 1 Traditional v/s Software Defined Networking

There are number of advantages of SDN network over traditional network such as flexibility, innovation, dynamic, cheaper, scalability. In SDN, control plane is to be programmable.

Openflow [4] is a protocol that is used in SDN network. It is used for the purpose of communication between data and control plane. ForCES (forwarding and control element separation) [5] is another protocol that we can use but openflow is a standard protocol. In SDN architecture there are 3 components such as data plane, control plane, SDN applications as shown in Fig. 2. The Data Plane is simply dumb devices that have no functionality. In SDN terminology these devices are called Openflow switches. Openflow switches can be physical or virtual switches. Second component is control plane that is called SDN controller. There are various controllers such as NOX [6], POX [7], RYU [8], Opendaylight [9] are available. It is called master of data plane. Third component is SDN applications that are written in language. According to these applications control plane insert the rules into openflow switches.

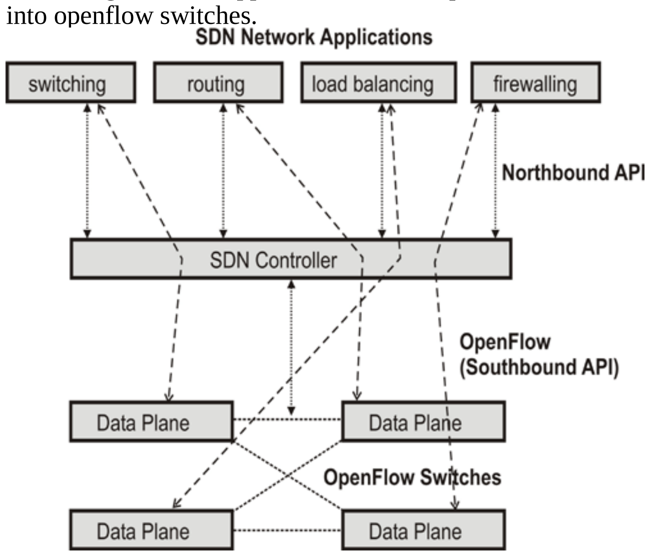

Fig. 2 SDN Architecture

The place where the control plane insert the rule is called flow table of openflow switch. There are number of flow rules are present in flow table. The flow rule described how the particular packet to be handle. Each flow rule consist of matching field and action parameter. When packet come into the switch, packet is matched with the matching field of flow rule and then action is to be performed. If match is not find, then action will depend upon the table miss entry of flow table. Because to handle table miss, table miss entry is present in the flow table. Drop the packet, send the packet to controller are the table miss entry actions. Set of messages are exchanged between controller and switch via a secure channel.

There are two approaches are available that are used by controllers to instruct to openflow switch how to handle a packet. In first method, switch has no knowledge what to do with the packet when packet come into switch. It send the packet to controller. Then according to application controller insert the flow rules into flow table [10]. Then switch Performed the action according to rules. This is called Reactive approach. In proactive method, switch performed the action on incoming packet according to proactively installed rules into flow table of switch.

III. Related Work

In today network, single server is unable to handle all client requests because amount of traffic is huge. Load balancer is used to solve these problems. The main functionality of load balancer is to distribute the client load among number of servers. Traditional load balancer are used for this purpose. But main problems with these load balancers are non-programmable, expensive hardware. Now days SDN based load balancer are used. We can convert openflow dumb device into strong load balancer by writing SDN application (such as load balancer).

Kaur et al. [11] implemented load balancing strategy that distribute the traffic in round robin fashion. Server load, link delay are the parameters that does not take into account by this strategy as shown in Fig. 3. But these parameters are necessary in reality because link load and speed are vary from one server to the other. Because it could never happen that each link have same capacity and speed.

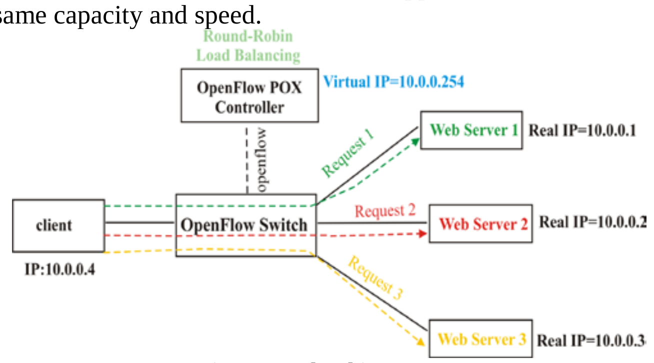

Fig. 3 Round Robin Strategy

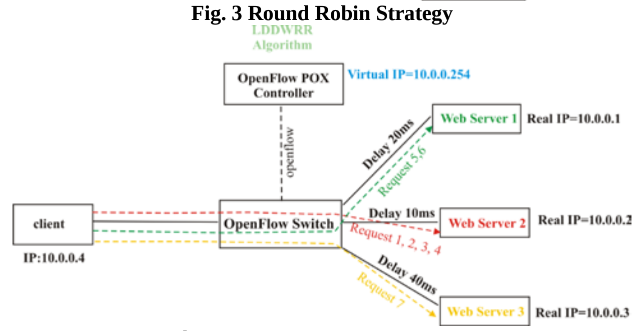

Fig. 4 LDDWRR Topology

Richard et al. [12] implemented algorithm that divides the total traffic of clients among servers according to the weight of server. Large overhead is the main limitation of this paper. Marc et al. [13] discusses about strategy in which large number load balancers are used to manage the load of different servers. For example one load balancer is used to handle or balance the load of one server such as email server, other handle the load of other server such as web server. Link delay parameter is .not consider by this algorithm.

Our load balancing strategy solved these limitations. In this paper we implemented Least Delay Dynamic Weighted Round Robin Load balancing strategy. We consider all the parameters such as link delay, link speed. According to different speed we assigned different delay to each link. The server with the highest speed link mean least delay handle more number of requests.

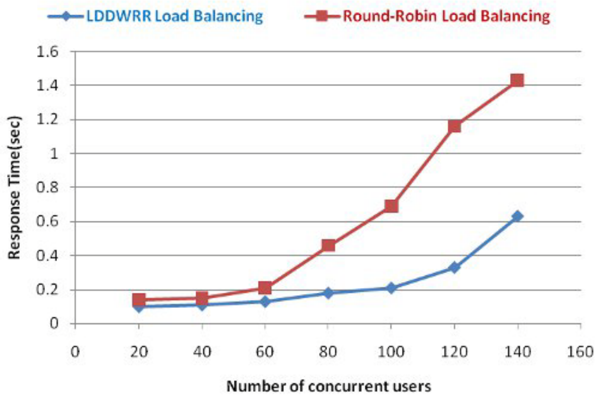

Fig. 5 Response time in LDDWRR and Round Robin Strategy

IV. Experiment Setup

Mininet tool [14] is used to create a network topology to test LDDWRR algorithm that is Least Delay Dynamic Weighted Round Robin Load Balancing algorithms or application. There are 3 servers, 1 client, POX controller, openflow switch in the topology. The host h4 having IP address 10.0.0.4 act as a client. The host h1, h2, h3 having IP addresses 10.0.0.1, 10.0.0.2, 10.0.0.3 act as a web servers. On host h4 we installed "siege" load testing tool. On hosts h1, h2, h3 we implemented web servers. There are links between servers and Openflow switch and we assigned 20,10,40 ms delay on each link. After that we assigned the dynamic weight according to the delay. The server with the least delay handle more weight (traffic) than other servers having more delay.

As shown in Fig. 4 the link delay between server h2 and Openflow switch is 10 ms, h1 and Openflow switch is 20 ms, and h3 and Openflow switch is 40 ms. According to this scenario the h2 server handle dynamic weight that is more than other servers. For example requests 1,2,3,4 are handled by h2. The next h1 server handles dynamic weight that is less than h2 server but more than h3 server. For example requests 5,6 handled by h1 server. The server having more delay than others handled least weight. For example only request 7 handled by h3 server. The weight that we assigned is dynamic according to delay. Unless all the requests are not end this process continued.

V. Experiment Evaluation

We evaluate our algorithm "Least dynamic delay weighted round-robin load balancing (LDDWRR)" with existing round-robin load balancing algorithms on the basis of Transaction Rate (trans/sec), Throughput (MB/sec) and Response time (sec). Siege tool is used to generate a load that is send to the servers. The total load generated by the siege is equal to the no of concurrent users multiplied with number of requests by each user. For example we have a 20 concurrent user each send a 5 requests then total load or number of requests are equal to 100. With the increase of concurrent users the total no of requests are also increased.

The response time of servers shown in Fig. 5. The concurrent users are shown at x axis. The Response Time (sec) shown at y axis. We can clearly observed from the graph that response time is better in our LDDWRR algorithm than already existing Round Robin algorithms. The file that is downloaded from server is index file that is very small file.

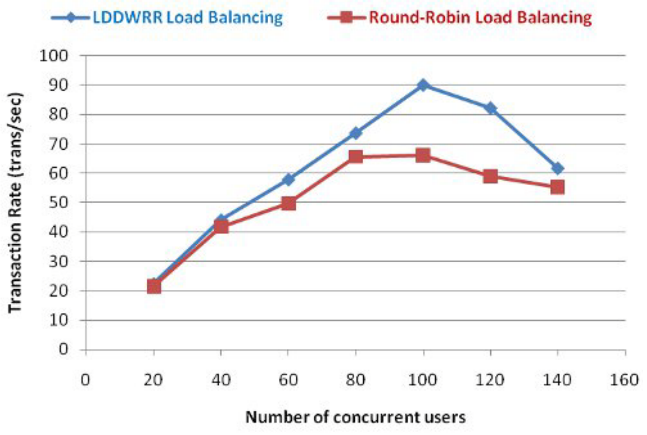

Fig. 6 Transaction Rate in LDDWRR and Round Robin Strategy

The transaction rate of servers shown in Fig. 6. The concurrent users are shown at x-axis. The Transaction rate (trans/sec) shown at y-axis. We can clearly observed from the graph that transaction rate is better in our LDDWRR algorithm than already existing Round Robin algorithms.

The can not correctly measure the throughput with this file because the size is very small. Therefore We have created a large file of 100 KB to measure the performance of throughput. As shown in Fig. 7 throughput is also better in our LDDWRR algorithm than the round robin algorithm.

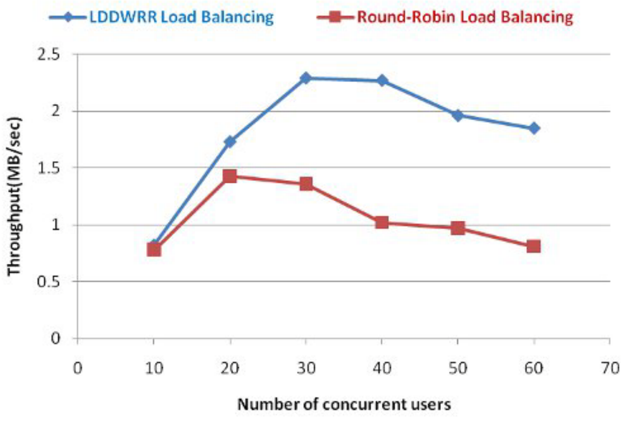

Fig. 7 Throughput in LDDWRR and Round Robin Strategy

Other parameter such as Response time (sec), Transaction rate (trans/sec) also measured with large file. As shown in Fig. 8 response time is still better in our LDDWRR algorithm than round-robin algorithm.

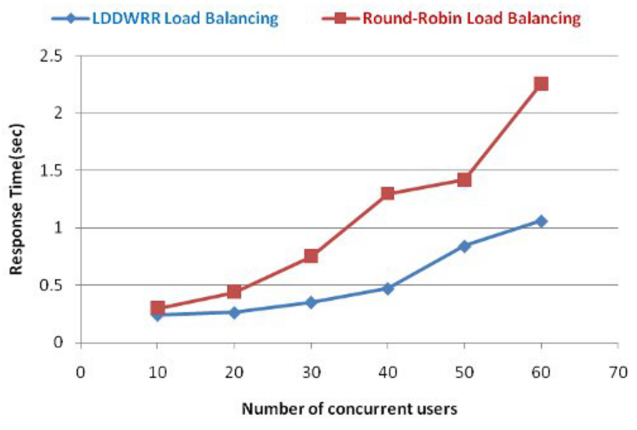

Fig. 8 Response time in LDDWRR and Round Robin Strategy with 100 kb file

As shown in Fig. 9 the transaction rate is also better in our algorithm than round-robin algorithm.

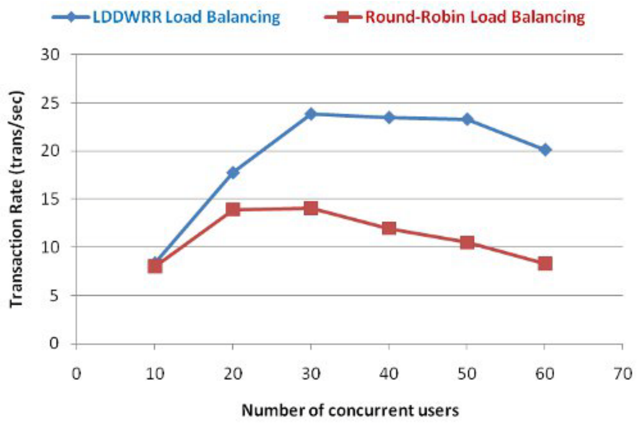

Fig. 9 Transaction Rate in LDDWRR and Round Robin Strategy with 100 kb file

VI. Conclusion

The conclusion of our paper was that we successfully implemented the LDDWRR algorithm using POX controller. We also conclude that our LDDWRR algorithm is better in all performance parameters such as Response time (sec), Throughput(MB/sec), Transaction Rate(trans/sec) than round robin algorithm. The hardware based load balancer is very expensive but our software based load balancer is cheaper. The modification or deployment of new behavior is very easy in SDN based load balancer.

In this paper we dynamically assigned a load to the servers according to the delay. But in our paper, we are using only single controller. There could be a single point of failure. In future work we can use multiple controllers.

VII. References

[1] Kaur, Sukhveer, and Japinder Singh. "Implementation of Server Load Balancing in Software Defined Networking." In Information Systems Design and Intelligent Applications, pp. 147-157. Springer India, 2016.

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

[3] 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.

[4] Akyildiz, Ian F., Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. "A roadmap for traffic engineering in SDN-OpenFlow networks." Computer Networks 71 (2014): 1-30.

[5] 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.

[6] Shah, Syed Abdullah, Jannet Faiz, Maham Farooq, Aamir Shafi, and Syed Akbar Mehdi. "An architectural evaluation of SDN controllers." In Communications (ICC), 2013 IEEE International Conference on, pp. 3504-3508. IEEE, 2013.

[7] Prete, Ligia Rodrigues, Christiane Marie Schweitzer, Ailton Akira Shinoda, and Rogerio Leao Santos de Oliveira. "Simulation in an SDN network scenario using the POX Controller." In Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on, pp. 1-6. IEEE, 2014.

[8] Khondoker, Rahamatullah, Adel Zaalouk, Ronald Marx, and Kpatcha Bayarou. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers." In Computer Applications and Information Systems (WCCAIS), 2014 World Congress on, pp. 1-7. IEEE, 2014.

[9] Medved, Jan, Robert Varga, Anton Tkacik, and Ken Gray. "Opendaylight: Towards a model-driven sdn controller architecture." In A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on, pp. 1-6. IEEE, 2014.

[10] Fernandez, Marcial P. "Comparing openflow controller paradigms scalability: Reactive and proactive." In Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on, pp. 1009-1016. IEEE, 2013.

[11] Kaur, Sukhveer, Krishan Kumar, Japinder Singh, and Navtej Singh Ghumman. "Round-robin based load balancing in Software Defined Networking." In Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on, pp. 2136-2139. IEEE, 2015.

[12] Wang, Richard, Dana Butnariu, and Jennifer Rexford. "OpenFlow-Based Server Load Balancing Gone Wild." Hot-ICE 11 (2011): 12-12.

[13] 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.

[14] Keti, Faris, and Shavan Askar. "Emulation of Software Defined Networks Using Mininet in Different Simulation Environments." In Intelligent Systems, Modelling and Simulation (ISMS), 2015 6th International Conference on, pp. 205-210. IEEE, 2015.