Advances in Computer Science and Information Technology (ACSIT)
Print ISSN : 2393-9907; Online ISSN : 2393-9915; Volume 2, Number 7; April – June, 2015 pp 72 – 76
Krishi Sanskriti Publications
http://www.krishisanskriti.org/ACSIT.html

Load Balancing in Software-defined Network (SDN) Based on Traffic Volume

Smriti Bhandarkar[1], Kotla Amjath Khan[2]

[1]M. Tech. Student
School Of Computer Engineering
KIIT University, Bhubaneswar, Odisha
8889064205, smriti.bhandarkar@gmail.com
[2]Assistant Professor
School Of Computer Engineering
KIIT University,Bhubaneswar, Odisha
7504006924, kotla.amjath@gmail.com

Abstract: *Software-defined networking (SDN) is a new emerging technology in the field of computer networks which is evolved from the work done at Stanford University and Berkeley University around 2008. It allows network administrator to manage the network services through the abstraction of lower level functionality, which is done by separating the control layer (brain of network) from the data layer (forwarding the packets). The centralized architecture of SDN is the bottle neck for scalable and dynamic nature of SDN. The growth in today's organizational network like cloud computing technology for data center and big data, virtualization etc increases the traffic on the link which may cause collision of packets or data loss. The routing algorithms developed till now and the splitting of control layer to divide the load among various controllers is not suitable for solving this problem. The key limitations are statically configured forwarding plane and uneven load balancing among the controllers in the network. The dynamic load balancer is an approach which dynamically shifts the load to the other shortest path when it is greater than the bandwidth of the link. By experimental analysis, we conclude that it gives better results in terms of responses/sec and efficiency as compared with the existing round-robin load balancing algorithm.*

Keywords: *software-defined network(SDN), data plane, control plane, centralized control plane, decentralized control plane, dynamic load balancer, round-robin load balancer, floodlight, mininet, cbench*

1. INTRODUCTION

Software defined network[1-3] is a new emerging technology in the field of networking in which programs written in high-level languages like C, java, ruby, Perl etc for control plane by the network administrator is used to control the behavior of whole network. Software defined network deals with splitting of infrastructure layer from control layer which enhances the programming capability, flexibility, malleability and manageability of the network. In spite of having lots of benefits over traditional network scalability of SDN is a big issue i.e. centralized nature of control plane is not friendly with the growing organizational network.

As software-defined network is developed to manage large networks like WAN, cloud computing technologies like data center, big data or virtual network technologies etc. Growth in today's network leads to large amount of traffic on the link due to which performance and efficiency of the network degrades by collision and information loss. So, there is a need of such an efficient algorithm which is able to handle or manage large amount of load.

2. RELATED WORK

PALETTE [4-5] follows the two ideas of DIFANE. The network administrator has the authority to specify the policies which defines how the switches can forward, drop, modify and measure the traffic. It is an efficient solution which keeps the traffic in the data plane and forwards the packets through intermediate switches having necessary rules and the controller partition rules over the switches.

R-SDN [6] has a vertically distributed control plane. Number of network/forwarding devices on each layer increases according to the Fibonacci series as the idea keep in mind that series increase like branches of a tree (spanning tree) with no loop. They manages the network by using Fibonacci heap ordered tree for load balancing and routing. The algorithm is solvable in polynomial time and gives less response time as compared to the traditional network.

Flow Slice (FS) [7] was used to divide each traffic flow into several flow slices and balance the load through various paths in a network. The paper claimed that if the setting of a slicing threshold was 1 to 4 milliseconds, the FS strategy could obtain nearly optimal performance. Based on the measurement, the paper presented various slice thresholds with other variables, such as Flow-Slice packet count, Flow-

Slice size, and Flow-Slice number, to find the impact. Finally, the paper measured delay, packet loss rate, and out-of-order packet value to determine the performance of the FS scheme.

In paper [8], author configuring a mesh Ethernet network using SDN topology showing L2 essential/necessary features e.g. creation of spanning tree is still missing in the SDN creation. They focus on typical computing centers of cloud where both loop free network topologies and their energy efficiency. GreenMST is the proposed prototype fulfills the basic requirements of loop free L2 network topology which is suitable or fit for various production and experimental network production. This prototype avoids the drawback of traditional non-openflow solutions like STP protocol by providing network applications to specify the metric dynamically, which is used by the controller for preparing the spanning tree. It reduces the energy consumption by switching off inactive ports. The future work focuses on providing the solutions on current prototype i.e. introducing the cache with the list of deactivate interfaces.

In SDN, distributed controllers [9] have been proposed to solve the scalability issues and reliability issues of network control plane. There is a limitation of distributed controllers, the mapping of switch and controller is configured statically due to which the load distributed among various controllers is not even. To solve this problem, this architecture is proposed inwhich the pool of controllers is shrink and grow dynamically according to the traffic on the link and load on each link is dynamically shifted across all controllers.

In CORONET [10], the scalability of SDN is improved as compared to standard approach of SDN. VLAN reduces the number of packets forwarding rules and packet forwarding. It only specifies logical paths rather than physical paths. There are various openflow applications exists which directly control the packets path, these applications can be rewritten using CORONET architecture. In future work, author plan to evaluate the generality of CORONET to support common SDN applications and build a general framework which allows seamless integration with any SDN application.

Control plane and reliability of controller :- It only supports fault tolerance for any failure in data plane. In CORONET, they tries to provide a complete solution for fault tolerance which recovers multiple failure in SDN based domain.
For faults in controller, they compare two approaches:
1. An approach based on a distributed hash table inspired by Onix.
2. A traditional software failover solution based on heart beat detection.

For the control plane, they check feasibility using traditional distributed mechanisms of the network like spanning tree protocol, and compare with a approach in which the controller reconfigures the control plane when faults are detected.

In paper [11], they introduces a new algorithm focuses on the issue of load balancing and strategies of routing in SDN. Although there are various algorithms present on this issue but they are not suitable for the large flow distributed network because they don't consider load collision on the middle of the transmission of packets. They proposed an efficient algorithm for path switching to balance the uneven load exists on the network. The experimental results show that this algorithm gives better performance than the other one.
In paper [12], they assumed that openflow switch in SDN deals with multi-protocol packet header in various packets like Ethernet, HTTP, and SIP etc. They discussed architecture and requirements of a openflow switch to work with multi-protocol packet header. For this, more intelligent and programmable switch function is required. Therefore, they developed architecture by combining active network technology and SDN. It's done with virtual CPU and memory called packet processor and a user program is loaded in it which is invoked packet by packet. All packet processing shouldn't do by this this user program. Several system calls, library functions and utility functions are provided by this platform to the user program. A component named transfer engine are programmed to support lower layer protocols. Since, the user program handles various protocols. They prepare various transfer engines according to the platform. The controller has the responsibility to send the user program to the openflow switch and notifies what kind of transfer engine is invoked for that program. They proposed various load balancing servers (proxy) for HTTP using this platform. By this, it is proved that one can apply SDN architecture to higher layer protocol also.

3. PROBLEM DEFINITION

Using round robin load balancer method incoming packets or requests are distributed across the number of available switches in the network i.e. switch cluster. One can choose this load balancer, if all the switches present in the cluster have the same capability and handle equal amount amount of load. By considering this constrain, the round robin load balancing method is a simplest and effective method for load distribution. If the switches with equal capabilities are used for round robin load balancing this means that less capable switch gets flooded with the number of packets or requests even though it is not able to process that much number of requests.

The limitations of round robin load balancer are as follows -
1. Statically configured
2. Uneven load balancing

The floodlight controller also has a limitation that Topology module set cost 1 for all the links present in the topology. The Topology Service discovers the topologies according to the information on the link. And best path can be computed by calculating the minimum number of hops between the switches.

4. IMPLEMENTATION

Load balancing means a efficient and smart congestion aware routing protocol in SDN. It is an essential constraint for the network based on SDN environment to improve the scalability and availability which leads to achieve maximum number of packets handled by the controller in minimal time for any application.
Dynamic load balancer is a module integrated with the floodlight controller to set different custom costs on each link dynamically cached by topology module. By using this module in the floodlight controller, new custom costs are setup on the link on both the directions.

Topology Module : - Used to extract and setting information of topology.

DBalancer : - Contains the main function and used to get dependencies of modules and for loading modules.

IDBalancer : - It is an interface used to provide service getlinkcost () and setlinkcost ().

IDListener : - It is also an interface used to change the cost of the link.

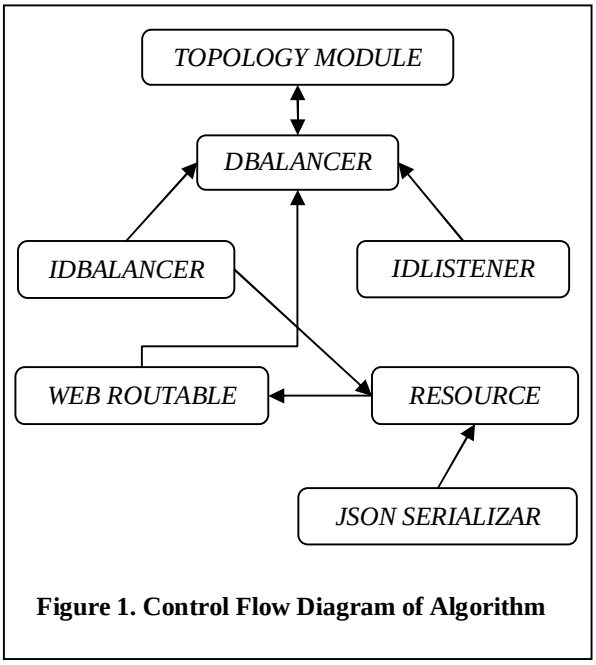

Figure 1. Control Flow Diagram of Algorithm

Web Routable : - Getting the link cost with supporting information.

Resource : - Managing the resources and return the link cost.

JSON Serializer : - It extracts source and destination address, source and destination port information of the link i.e. handles the link information.

5. EXPERIMENTAL RESULTS

The dynamic load balancer has been implemented using floodlight-0.90 with openflow protocol-1.0.0 on a system with Intel(R) Pentium(R) Dual CPU T3200, 2.00GHz, 3.00GB RAM and Ubuntu 13.10 OS. Mininet is used to create topology consisting of about 60,000 networking devices i.e. 150 openflow switches and 400 hosts attached per switch. Number of flows per second and number of responses given by the controller per second can be calculated by Cbench in both modes i.e. throughput mode and latency mode.

Cbench simulates a configurable number of OpenFlow switches, each sending a stream of Packet In messages to the controller undergoing the test. Cbench is intended to test hub/learning switch behavior, and can vary the source MAC address in the Packet In messages by configurable range to stress the controller's table lookup code. In the following benchmarks Cbench is configured to run 20 tests per controller combination, each lasting 10000ms. Each test's total responses received are averaged to produce responses-per-second result.

Cbench operates in one of two modes, throughput or latency. In throughput mode, each of 150 emulated switches constantly sends as many Packet In messages as possible to the controller, ensuring that the TCP buffer is always full, and that the controller always has work to be done.

In Cbench's latency mode, each emulated switch sends a single packet to the controller, waits until it receives either a Packet Out and/or Flow Mod message, and then repeats the process as quickly as possible. This test has been configured with a single emulated switch, and because only one message is in flight at any given time, the total number of responses received at the end of the time period can be used to compute the average time it took the controller to process each.

Table 1. Throughput Mode - Number Of Responses/sec

	MAX	MIN	AVG	STD. DIV.
DYNAMIC LB	40244.75	32707.33	39088.66	1659.11
ROUND-ROBIN LB	4020.92	3906.56	3989.97	35.06

Table 2. latency Mode - Number Of Responses/sec

	MAX	MIN	AVG	STD. DIV.
DYNAMIC LB	15109.46	13928.99	14762.27	286.43
ROUND-ROBIN LB	8557.81	7924.60	8394.84	158.74

In both the tables 1 and 2 shows that by using dynamic load balancer controller can send more number of packets/responses as compared to round robin load balancer. In throughput mode average number responses/sec = 39088.66 and in latency mode it is 14762.27.

The graph in figure 2 illustrated the "Number of Flows Handled by the Controller per Second Using Dynamic Load Balancer and Round Robin Load Balancer in Throughput Mode", which is bounded by the maximum value of 40 flows/sec and the minimum value of 17.25 flows/sec.

The graph in figure 3 illustrated the "Number of Flows Handled by the Controller per Second Using Dynamic Load Balancer and Round Robin Load Balancer in Latency Mode", which is bounded by the maximum value of 14.75 flows/sec and the minimum value of 12.15 flows/sec. The first observation is minimum for both the figures 2 and 3 because at the initial stage all the switches are not fully configured.

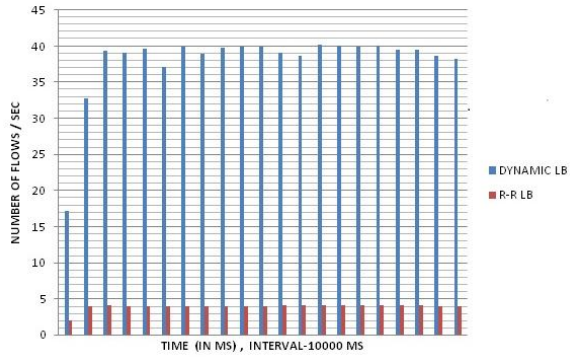

Figure 2. Number Of Flows Handled By The Controller Per Second Using Dynamic Load Balancer And Round Robin Load Balancer In Throughput Mode

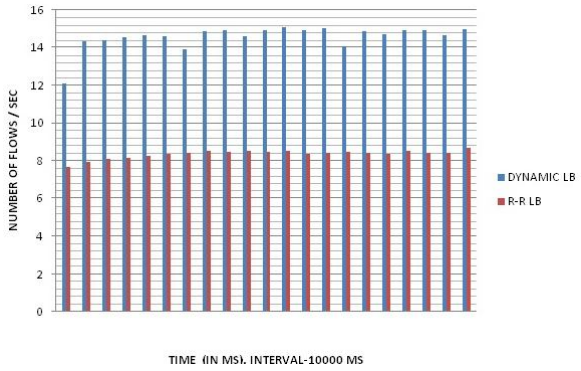

Figure 3. Number Of Flows Handled By The Controller Per Second Using Dynamic Load Balancer And Round Robin Load Balancer In Latency Mode

6. CONCLUSION AND FUTURE SCOPE

SDN is logically centralized networking paradigm but due to the high load on the control plane, capability of the network decreases. For this, the load can be divided among various controllers. The uneven load balancing among the controllers is the challenging issue for the scalability and dynamic nature of the SDN.

The dynamic load balancer is an algorithm used for load shifting from the best calculated path to reduce the collision and information loss, when the load on the link will be greater than the bandwidth of the link. The experimental results, proves that it can handle more packets and having greater efficiency than round-robin load balancer in both the modes.

As software-defined network is developed to manage large networks like WAN, cloud computing technologies like data center, big data etc. Growth in today's network leads to large amount of traffic on the link due to which performance and efficiency of the network degrades. The algorithm is not analyzed over such a big network. One can check the performance on these networks and update the algorithm according to the experimental results.

7. REFERENCES

[1] Yosr Jarraya, Taous Madi, and Mourad Debbabi, "A Survey and a Layered Taxonomy of Software-Defined Networking", published in Communications Surveys and Tutorials, IEEE Issue: 99 , 2014.

[2] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti,"A survey of software defined networking: Past, present, and future of programmable networks,"Communications Surveys Tutorials, IEEE,no. 99, pp. 1–18, 2014.

[3] F. Hu, Q. Hao, and K. Bao, "A survey on software defined networking(SDN) and openflow: From concept to implementation,"IEEE Commun. Surveys Tuts., no. 99, pp. 1–1, 2014.

[4] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," Technion, Tech. Rep. TR12-05, 2012.

[5] Yossi Kanizo, David Hay, and Isaac Keslassy. Palette: Distributing tables in software-defined networks. In INFOCOM, pages 545–549,2013.

[6] DAI, Wei, Guochu SHOU, Yihong HU, and Zhigang GUO. "R-SDN: A RECUSIVE APPROACH FOR SCALING SDN."

[7] Shi, Lei, Bin Liu, Changhua Sun, Zhengyu Yin, Laxmi Bhuyan, and H. Jonathan Chao. "Load-balancing multipath switching system with flow slice." *Computers, IEEE Transactions on* 61, no. 3 (2012): 350-365.

[8] Prete, Luca, Fabio Farina, Mauro Campanella, and Andrea Biancini. "Energy efficient minimum spanning tree in OpenFlow networks." In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pp. 36-41. IEEE, 2012.

[9] Dixit, Advait, Fang Hao, Sarit Mukherjee, T. V. Lakshman, and Ramana Kompella. "Towards an elastic distributed SDN controller." In *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 7-12. ACM, 2013.

[10] Kim, Hyojoon, J. R. Santos, Y. Turner, M. Schlansker, J. Tourrilhes, and Nick Feamster. "Coronet: Fault tolerance for

software defined networks." In *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pp. 1-2. IEEE, 2012.
[11] Long, Hui, Yao Shen, Minyi Guo, and Feilong Tang. "LABERIO: Dynamic load-balanced routing in OpenFlow-enabled networks." In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pp. 290-297. IEEE, 2013.
[12] Hata, Hiroaki. "A study of requirements for sdn switch platform." In *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, pp. 79-84. IEEE, 2013.