A Comparative Performance Analysis for Static and Dynamic Load Balancing Techniques in Software Defined Network Environment

Haeeder Munther Noman1, 2, Mahdi Nsaif Jasim3
1 College of Information Technology, University of Babylon, Iraq
2 Technical Instructors Training Institute, Middle Technical University, Baghdad, Iraq
3 University of Information Technology and Communications, Baghdad, Iraq
1hydermu79@gmail.com, 3mahdinsaif@uoitc.edu.iq

Abstract. Traditional load balancer suffers from inflexibility, high cost and the difficulty to manage the network. Software-Defined Network (SDN) gives a promising solution for traditional load balancer limitations through enabling programmability and centralized control over the network, which offers inexpensive and scalable solutions. This paper investigates the impact of increasing the workload requests from 0 up to 180 requests per second (req/sec) in order to explore average network throughput under the deployment of static as well as dynamic load balancing algorithms by the POX controller. The study was based on the utilization of HTTPperf based on the fact that it provides a flexible facility for the generation of various HTTP workloads as well for the measurement of server performance. Our experiments revealed that as the number of requests increases the throughput increases as well. Dynamic least bandwidth-based load balance scheme has shown a remarkable improvement in terms of average network throughput up to 8%, 3.3% and 2.56 %, as compared with static load balancing schemes like random, round-robin and weighted round-robin. However, Dynamic least bandwidth recorded a slight ineffective improvement less than 1 % was recorded when a comparison was carried out dynamic least connections, this directed us to the fact that their performance was almost the same. Based on what dynamic least bandwidth scheme has achieved in terms of average network throughput, more efforts should be driven by the researchers on algorithm development like introducing static weight for each server as well as the application of the algorithm on several POX controllers to avoid a single point of failure.

1. Introduction
The quick development of Internet technology makes the server cluster of large Internet service providers (ISP's) more challenging. With the growth of users as well as network bandwidth, servers need to handle a large number of access requests during a short period. If the server cannot timely process user access request which leads to the extension of user’s waiting time, then it will greatly reduce the Quality of Service (QoS), circumstances like these make servers become the new bottleneck in-network and force researchers to start the study of how to improve servers performance.to achieve that, enterprises adopt several measures, such as improvement of CPU’s processing speed, the increment of server’s cache capacity, deployment of the high-speed disk array, as well as the construction of the server cluster[1]. Simply upgrading the hardware system will not only cause the idle criteria of existing resources but as the business continues to expand, companies will face the same challenging situation. Through the establishment of the server cluster, companies can be able to forward access request to a pool of servers with the aim to improve the performance of the server to a certain extent. However, such
a solution also comes up with a new problem, that when the server cluster receives an access request, which server will have the chance to respond. As the control system cannot reasonably assign access requests, then within such conditions load balancing situation will probably occur. Based on the above, researchers proposed load balancing strategies [2]. The classical load balancing techniques use expensive hardware devices and cannot achieve precise control of the traffic load; such limitations make the traditional load balancing technology not suitable for large-scale applications [3]. Further improvement relies on a single load balancer located in front of the server cluster to reasonably direct the load among several servers with many goals to achieve such as the full use of the server resources which leads to achieving the concept of load balance among the pool of servers, reducing access requests response time and improving system throughput and fault tolerance. The emergence of Software Defined Networking [4] (SDN) offered the opportunity for the managers to possess a traffic management technology that included the advantage of low cost and flexible form of operation. The main advantage was allowing network administrators to manage network services through the abstraction of lower-level functionality through the separation of the control plane and the forwarding plane of traditional network architecture. To achieve the objective for data forwarding control, the controller functioning in the control plane holds the ability to manage the switches’ flow table [5]. SDN comes up with two essential interfaces, the northbound interface, which allows any lower-level component to communicate with a higher-level component, hence it represents the communication interface between the application layer and the control layer in the architecture of SDN. North-bound application program interface can support for network functions like loop avoidance, security, routing, load balancing, computation and many other networks functions. the other interface is the southbound interface acting as the link between controller and forwarding devices. OpenFlow protocol is the most suitable southbound interface until now. OpenFlow first proposed in [6] as a way to enable researchers to conduct experiments in production networks

2. Related work
The overload problem for web servers, different loads balancing algorithms have been proposed, but still an open challenge for researchers. Some of the loads balancing algorithms were proposed based on a single load balancing parameter that was not enough to select the best server to process the requests, as it can only satisfy specific requirements of the users [7]. [8] Proposed that SDN offers a cost-effective and flexible approach in the implementation of the load balancer, the utilization of the SDN paradigm for server cluster load balancing reduced the cost and offered deployment flexibility that carried on the degradation time to deploy, the automation and empowerment of network development with no-vendor related information. The key behind the use of SDN load balancer was that it didn’t require any separate hardware as well as the ability to address more issues as compared with conventional load balancers. [9] stated that conventional load balancers were inflexible to change or to modify since conventional load balancers are locked by vendors and because of their non-programmable design. Hence, network administrators weren’t able to build their algorithms on their own, SDN solved the issue and made load balance device versatile and programmable. Typically, the load balance scheme is categorized into two; Static and Dynamic. The static scheme [10] distributed the load without taking into account the capability of nodes, such as RAM, server processor bandwidth and ties. The static scheme came with many advantages such as the appropriateness for homogenous servers, the less overhead and the easiness to implement. But, in general not flexible and incapable of considering dynamic attributes changes. For example, if one server received a huge number of tasks, after a certain time another task will be sent to the same server regardless of the capacity of the server or size of the task. dynamic scheme [11] distributes the load according to the current status of the network nodes. It ensures that the load-balance system tests the load ability of the server and the links at run time. Nonetheless, the dynamic scheme neglects the user request type and size and can only use one algorithm for all the different services. In fact, the static load balancing is mainly used for a small-sized network [12]. Its performance is better compared with that of dynamic load balancing. However, when large area web servers are adopted where requests are generated dynamically, the performance of the static load balancing could be
worsened. Therefore, the Dynamic load balancing is used for worldwide networks as it could process the requests based on the existing load of the servers.

3. Problem Statement
Being able to answer the common question, which load balancing technique could achieve the best results in terms of essential metric related to performance such as web servers’ average throughput in order to accomplish preeminent server performance and scalability. The main objective of this study is to perform a gap analysis review on the approaches/techniques of the SDN load balancers. Results and observations from the simulations are analysed using HTTPerf network testing tool to evaluate the performance of the static as well as dynamic load balancing to select the optimal performing load balancing algorithm.

4. Proposed Scheme
Various parameters have been utilized for the simulation proposed to run in SDN-Based environment as illustrated in Table 1. and utilized to build the SDN network topology shown in figure 1.

| Parameter                      | Value                  |
|-------------------------------|------------------------|
| 1- Simulation tool            | HTTPerf                |
| 2- SDN Controller Platform    | POX                    |
| 3- SDN Virtual Switch         | OpenFlow Switch        |
| 4- Southbound Protocol        | OpenFlow Version 1.0   |
| 5- Total no. of Connections   | 100 Connection         |
| 6- Max no. of requests per    | 180 (req/sec)          |
|    second in each Connection  |                        |
| 7- Min no. of requests per    | 20 (req/sec)           |
|    second in each Connection  |                        |
| 8- No. of HTTP Web Servers    | HTTP Server 1 - 10.0.0.1 |
|    (3)                         | HTTP Server 2 - 10.0.0.2 |
| 9- No. of HTTP Client         | HTTP Server 3 - 10.0.0.3 |
|    (3)                         | Client 4 - 10.0.0.4    |
| 10- SDN topology type         | Single                 |

Figure 1. SDN-Based Network Topology
5. Experimental Setup and Evaluation
The key design of the proposed load balancing system architecture consists of a single POX controller, single OpenFlow switch, and load balancing modules invoked into the POX controller. The POX Controller represents the upgraded version of the NOX controller with the ability to provide several benefits such as reusable components, path selection, load balancing and topology discovery. Moreover, the POX controller has been considered as the default running controller when deployed through the Mininet software. OpenFlow switch mainly consists of a flow table which in turn contains the flow entries with corresponding actions, the process of OpenFlow switch initially started when it receives a packet after that a comparison was carried out between the header of the packet with the flow entries inside the flow table. In case there was a match with one of the flow entries then this results in applying the corresponding action of this flow entries, otherwise the packet is forwarded to the POX controller to inform the switch how to deal with this packet. Httperf tool is utilized for measuring web server performance. It provides a flexible facility for generating various HTTP workloads and for measuring server performance. The focus of HTTPerf is not on implementing one particular benchmark but on providing a robust, high-performance tool that facilitates the construction of both micro- and macro-level benchmarks. The three distinguishing characteristics of HTTPerf includes the ability to generate and sustain server overload, the support for the HTTP/1.1 and SSL protocols, and the extensibility to new workload generators and performance measurements. Finally, Mininet makes it possible to create a realistic virtual machine where real kernel, switch, and application code were being able to run on a single machine (virtual machine, cloud or native) during seconds with the use of a single command. However, Mininet is a remarkable strategy to significantly experiment, share and develop through OpenFlow and software-defined networks (SDN).

All the tests are carried out through the utilization of POX controller which is selected due to free open-source utility, that in fact, facilitated the addition and removal of invoked modules and gave the experiments more convenience. Port 6633, is considered as the default connection port for establishing a connection between OpenFlow switch and POX controller. POX controller includes a list of IP addresses assigned to each server statically. Mininet emulator carried out the creation of the virtual network topology, which is made up from three hosts acting as servers (h1, h2, h3 with IP's 10.0.0.1, 10.0.0.2, 10.0.0.3) with the same configuration to provide the same web services to the clients. The other three hosts act as clients (h4, h5, h6 with the IP's 10.0.0.4, 10.0.0.5, 10.0.0.6) attempting to access the server through the use of virtual IP address which represented the service IP. If a particular client requested the virtual IP (service IP), then the request would be forwarded via the OpenFlow switch to the POX controller to make the necessary computations and select the appropriate server to handle the request.

The total number of connections was set to 100 per second, and the request rate started from 0 up to 180 (req/sec) with a gradual increase of 20 (req/sec). The interval time between each sample was set to one second to ensure that the responses were received by the users. The user’s requests that are sent from different clients included only the HTTP service.

6. Results
Results were categorized into three sections: section A, section B and section C.

6.1 Section A:
was based on the implementation and comparison of static load balancing schemes (random, round-robin and weighted round-robin) in terms of average network throughput, according to Figure 2, results revealed that weighted round-robin technique has not recorded a noticeable improvement when compared with round-robin technique. However, there was an impressive enhancement when compared with random technique up to 6%. 

All the tests are carried out through the utilization of POX controller which is selected due to free open-source utility, that in fact, facilitated the addition and removal of invoked modules and gave the experiments more convenience. Port 6633, is considered as the default connection port for establishing a connection between OpenFlow switch and POX controller. POX controller includes a list of IP addresses assigned to each server statically. Mininet emulator carried out the creation of the virtual network topology, which is made up from three hosts acting as servers (h1, h2, h3 with IP's 10.0.0.1, 10.0.0.2, 10.0.0.3) with the same configuration to provide the same web services to the clients. The other three hosts act as clients (h4, h5, h6 with the IP's 10.0.0.4, 10.0.0.5, 10.0.0.6) attempting to access the server through the use of virtual IP address which represented the service IP. If a particular client requested the virtual IP (service IP), then the request would be forwarded via the OpenFlow switch to the POX controller to make the necessary computations and select the appropriate server to handle the request.

The total number of connections was set to 100 per second, and the request rate started from 0 up to 180 (req/sec) with a gradual increase of 20 (req/sec). The interval time between each sample was set to one second to ensure that the responses were received by the users. The user’s requests that are sent from different clients included only the HTTP service.

6. Results
Results were categorized into three sections: section A, section B and section C.

6.1 Section A:
was based on the implementation and comparison of static load balancing schemes (random, round-robin and weighted round-robin) in terms of average network throughput, according to Figure 2, results revealed that weighted round-robin technique has not recorded a noticeable improvement when compared with round-robin technique. However, there was an impressive enhancement when compared with random technique up to 6%. 

Figure 2. average network throughput through the utilization of a static load balancing scheme

Figure 3. average network throughput through the utilization of dynamic load balancing schemes.

Figure 4. average network throughput through the utilization of static as well as dynamic load balancing schemes

6.2 Section B:
no discrimination regarding the performance was recorded (less than 1%) for Dynamic load balancing schemes like least connection-based and least bandwidth-based as was illustrated from Figure 3.

6.3 Section C:
This was the core section of our work, due to the fact that it was based on the overall performance comparison in terms of average network throughput intended for all static as well as dynamic load balancing schemes. Referring to Figure 4, results demonstrated that dynamic schemes recorded a remarkable enhancement when compared with static schemes. While dynamic least connection based verified an enhancement up to 7%, 2.2% and 2% compared with static random, round-robin and...
weighted round-robin, dynamic least bandwidth-based achieved further enhancement up to 8%, 3.3% and 2.56% when compared with the same static load balancing schemes.

7. Conclusion
The results were presented in tabular and graphical form to make easy for understanding and to carry out a comparison between static (random, round-robin, weighted round-robin) and dynamic (least connections based, least bandwidth-based) load balancing schemes. The evaluation was based on the average network throughput considered as the best indicator to investigate the overall performance.

Server utilization was achieved through the ability to direct the traffic to a pool of backend servers by utilizing static as well as dynamic server load balancing algorithms. Through the observation of evaluation results, we can clearly say that the least bandwidth load balancing scheme better handled the network traffic and balanced the load among the pool servers more efficiently.

8. References
[1] Prakash M S and Deepalakshmi P 2019 “DServ-LB: Dynamic server load balancing algorithm,” Int. Journal of Communication. Systems 32
[2] Li L and Xu Q 2017 “Load balancing researches in SDN: A survey,” 7th IEEE Int.Conf. on Electronics Information and Emergency Communication (ICEIEC), pp. 403–408.
[3] Caulfield A M, Chung E S, M., Papamichael M K, Burger D C and Alkalay S 26-Dec-2019, “Hardware implemented load balancing.” Google Patents
[4] Mendiola A, Astorga J, Jacob E, Higuero M 2016 “A survey on the contributions of software-defined networking to traffic engineering,” IEEE Commun. Survey 19. pp. 918–953
[5] Kavana H M, Kavya V B, Madhura B, and Kamat N 2018 “Load balancing using SDN methodology,” Int. Journal of Engineering Research and Technology 7, pp. 206-208.
[6] Bholebawa I Z, Jha R K, and Dalal U D 2016 “Performance analysis of proposed OpenFlow-based network architecture using mininet,” Wireless Personal Communications 86 PP 943-958
[7] Lebiednik B, Mangal A, and Tiwari N 2016 “A survey and evaluation of data center network topologies,” arXiv preprint arXiv:1605.01701
[8] Mithbavkar D, Joshi H, Kotak H, Gajjar D, and Perigo L 2016 “Round robin load balancer using software defined networking (SDN),” Capstone Team Res. Proj 5, pp 1–9.
[9] Arahunashi A K, Vaidya G G, Neethu S, and Reddy K V 2018 “Implementation of Server Load Balancing Techniques Using Software-Defined Networking,” 3rd Inte. Conf. on Computational Systems and Information Technology for Sustainable Solutions (CSITSS) pp 87–90.
[10] Chen-Xiao C and Ya-Bin X 2016 “Research on load balance method in SDN,” Int. Journal of Grid and Distributed Computing 9. PP. 25-36
[11] Wang Y, Lin Y and Chang G 2019 “SDN-based dynamic multipath forwarding for inter-data center networking,” Int. Journal of Communication Systems 32.
[12] Raghul S, Subashi T, and Vimal K R 2017 “Literature survey on traffic-based server load balancing using SDN and open flow,” Fourth International Conference on Signal Processing, Communication and Networking (ICSCN) pp 1–6.