Load Balancing using queue length in SDN based switches

Methaq Khamees Faraj
Department of Computer Engineering
University of Technology, Baghdad, Iraq

Dr. Ahmed Al-Saadi
Department of Computer Engineering
University of Technology, Baghdad, Iraq

Dr. Riyadh Jabar Albahadili
Department of Computer Engineering
University of Technology, Baghdad, Iraq

Abstract- Traditional traffic management techniques suffer from lacking global view about networks due to the decentralized nature of network infrastructure. Moreover, traffic on networks fluctuates from time to time, which could result in overloading some devices while underutilizing other devices. A new paradigm in network architecture is Software Defined Networking (SDN), which provides a centralized network management platform and enables network engineers to develop customized programs. The centralization of SDN provides an opportunity of monitoring Quality of Service (QoS) parameters for the underlying network switches. This paper proposes an SDN based QoS aware load balancing system by monitoring queue length on SDN based switches in a certain time interval. The results of the proposed algorithm are evaluated with three of the most widely cited schemes. The results indicate significant improvement in system throughput.

Keywords – Software Defined Networking, Load Balancing, Quality of Service, POX

I. INTRODUCTION

The internet has become crucial and essential in our daily life. The number of internet users is predicted to reach up to 4.8 billion by the beginning of 2022 and an increase of traffic by 91% as Visual Networking Index (VNI) by Cisco predicted, Figure1 [1]. Servers overloading and network congestion have become a real problem that network engineers need to address [2]. Traditional networks devices required additional messages and control data to collect QoS that could create extra traffic and define new packet types. The centralized control in SDN provides the opportunity to obtain the QoS parameter for underlying switches without the need to add new control packets [3]. SDN has changed networking infrastructure via transforming from distributed control hardware fashion towards centralized control software, hence services that allow more flexible management of the network resources by decoupling control plane from the data plane [4]. The controller in SDN acts as a supervisor on network behavior through a different application used to forward flows in the data plane [1]. The queue length on switches represents the load on each path, as the data flow rate increases a bottleneck can be created that results in network congestion [5].

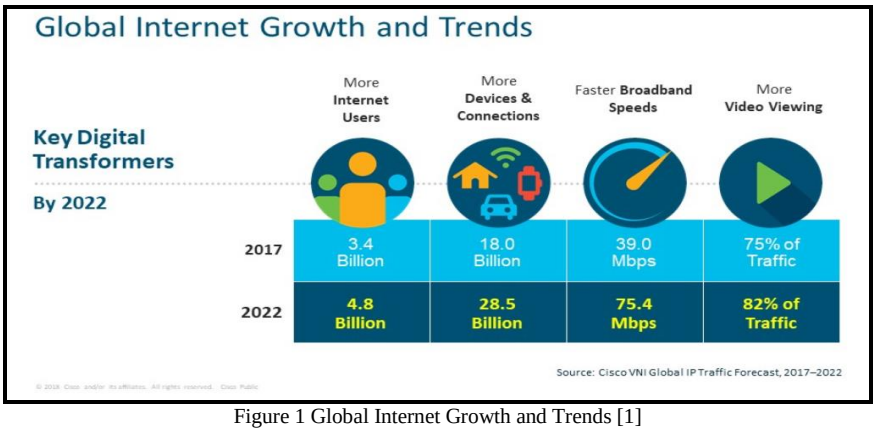

Figure 1 Global Internet Growth and Trends [1]

One approach to reduce the performance degradation is the re-routing of the flows in order to reduce congestion on crammed paths or queues [6]. Load balancing is an important technique that emerged in the mid-1990 when network traffic rapidly increased [7]. It has been used to improve the performance and scalability of the network systems [8]. The traditional network load balancer has some limitations such as using one load balancing algorithm for all the services and applications simultaneously, any reconfiguration in the traditional load balancer devices will require network shutdown, also it is very expensive and closed by the vendor [9]. On the other hand, SDN was developed to overcome the shortcoming of the long-established network structure by using the control layer that facilitates the downloading of various applications. SDN Balancer is an application that provides scalability and availability which leads to high utilization of network resources. Using suitable load balancing methods, the network performance can be kept as intended [10].

This paper proposes two approaches to utilize the queue length on SDN switches to direct the traffic to the least congested port. The first one, the controller collects the instance queue length and directs the traffic to the port with less packet on the queue at that time; the term instance queue load balancing (IQLB) is used to represent this approach throughout the paper. The second approach is QoS aware traffic management (QTM) which selects a switch port with less congested queue based on the load of the queue based on the load of the queue over a time period. In order to evaluate the suggested algorithms, the results are compared with a number of the most cited algorithms used in the load balancing algorithms. The results show that the proposed algorithm is more efficient in different loads of TCP and UDP traffic.

The remaining parts of the paper were arranged as follows: section 2 highlights the related work; section 3 explains the proposed approaches; section 4 discusses performance evaluation; section 5 consists of the conclusion and future work.

II. RELATED WORK

Load balancing becomes an important task that improves system performance and maintains stability [11]. SDN provides a good opportunity to utilize the programmable capabilities to develop applications that could redirect traffic to a server based on some policies [12]. Thus, it becomes necessary to develop an algorithm that can optimize system performance by balancing the workload among servers. One of the simplest algorithms is random load balancing that randomly redirects traffic between servers without considering any QoS parameters [13]. The main limitation of this strategy is that the load is uneven among servers then one server may be overloaded while others are idle. Another load balancing algorithm that utilizes SDN is Round Robin [2,14-18], which distributes traffic evenly across servers. This algorithm will be useful in load balancing applications as well as in other contexts where scheduling is required. However, the outcoming result is not very efficient, because round load balancers suppose that all servers and transmission links are similar which is not the case in realistic scenarios. Another approach which is used widely in data center networks known as SDN-based Equal-cost multi-path (SDN-Based ECMP) algorithm [19-21], this algorithm distributes data flows across available paths using flow hashing methods. Hash calculation is based on the header fields of the ingress packets. This approach is flow-based load balancing and does not take QoS in its consideration; if a lot of packets have the same header fields then it has the same hash value and will be forwarded along the same path without considering the queue length of switches. This results in network congestion and packet loss in case of overloading the network and creating an unbalanced utilization in the network resources. Another approach is Least bandwidth strategy [22] which distributes load dynamically and reroutes traffics to a server with the minimum network traffic. It collects the load on each server and decides according to the collected information;

therefore, it's more efficient but more difficult to implement as the load information for each server is not sent to the controller without any additional control messages. One more issue with the previous algorithm is not considering the quality of transmission link for each server. One more SDN-based load balancing is multipath QoS solution (HiQoS) [23]; This approach provides a QoS-guarantee solution by using different queues for each traffic type and each queue has different priority. However, this suggested system may have some struggles in guaranteeing the quality of the transmission for a specified traffic. Packet loss may occur when some type of traffic with the same queue priority needs to be transmitted simultaneously. Some research in literature utilizes queues in SDN based networks to analyze the resource utilization for the controller and the underlying switches [24], but it does not employ the queue length parameter in load balancing traffic management. Beacon QoS queue load balancing (BeaQoS) [25] is another approach that utilizes queue length. This algorithm uses statistical messages generated via Beacon controller to estimate the data rate of a multi-queue interface in multi-core processors. It introduces many strategies for management and load balancing. various dedicated methods for assigning flows to specific queues are employed while the deadline method is used to re-route flows to queue with less congestion if it arrived at deadline time. Also, this algorithm uses another load balancing strategy called min-load; it assigns upcoming traffic to less congested queue. while Multiway load balance method collects all flows in a specific queue and rearranges flows in descending order based on computed estimated Rates then assigns each flow to a suitable queue.

This paper handles the limitation in previous methods through utilizing the queue length to predict load on the server and also to reflect the quality of the link. Lower bandwidth link or bad link quality may induce more retransmission or more latency that increase the length of the packets in the queue and create more congestion and packet loss.

III. PROPOSED ALGORITHM

Queue overload is one of the main reasons of network transmission loss. The high number of packets in the queue means congestion on the network. Thus, one important approach is to adjust flow rate by using the queue length as a parameter to estimate the best path to forward traffic. This paper employs two approaches to monitor the queue length on switches that forward traffic to multiple servers and reroutes packets before buffer overflow. The first one is instance queue load balancing (IQLB) which simply selects the switch port with the least number of packets on the queue at the time there are packets ready for transmission. The second part suggested a queue monitoring algorithm called QoS aware traffic management (QTM), which considers the current queue length in addition to the previous load to consider more precise estimation of the load on the switch port.

3.1 Instance queue load balancing (IQLB):

In IQLB a threshold value of nodes buffer is determined as 80% from max queue length (this value has been selected empirically). After that, when there are some packets ready to transmit, the SDN controller obtains the number of packets on each port of the switch that perform load balancing algorithm. This value is stored in *Qlen* for each server S_i. Traffic are distributed based on a minimum number of backlogged packets on the switch port that forward traffic to the available servers connected each port using Eq (1)

$$Min\_Q = Min_1^n\ Qlen(Si)\ \&\&\ Qlen(Si) < Thr \quad (1)$$

Where S is a pool of available servers for the client to use and the total number of servers are n. Min_1^n is the function of selecting the switch port with minimum backlogged packets in the queue and i is the index of the available server where $1 \leq i < n$. Thr is the threshold value which equals to 80% from max queue length. Min_Q is the switch port with the lowest number of packets on the queue. Figure 2 illustrate IQLB approach, where Pox controller and switches initiate connectivity through OpenFlow protocol. The controller receives information about the existence of active switches via the Link Layer Discovery Protocol (LLDP) packets in the initiate handshake. The controller sends LLDP packets to switches by using the OpenFlow Packet-Out message. Each Packet-Out message contains an action that orders the switches to redirect the packet to all ports except the incoming one. The rule that illustrates any received LLDP packets are to be sent to the controller via an OpenFlow Packet-In message is pre-configured in switches. Then, the switches send the LLDP packet to the controller via Packet-In message. Through this scenario, the Pox controller discovered the topology. After the Pox controller discovers the topology, the initialization variable will be set and deploy the monitoring network traffic process. When traffic is initiated, the monitor network traffic calculates the number of backlog packets in each queue port and selects a path based on minimum queue length. If queues have the same length, then traffic will be forwarded randomly.

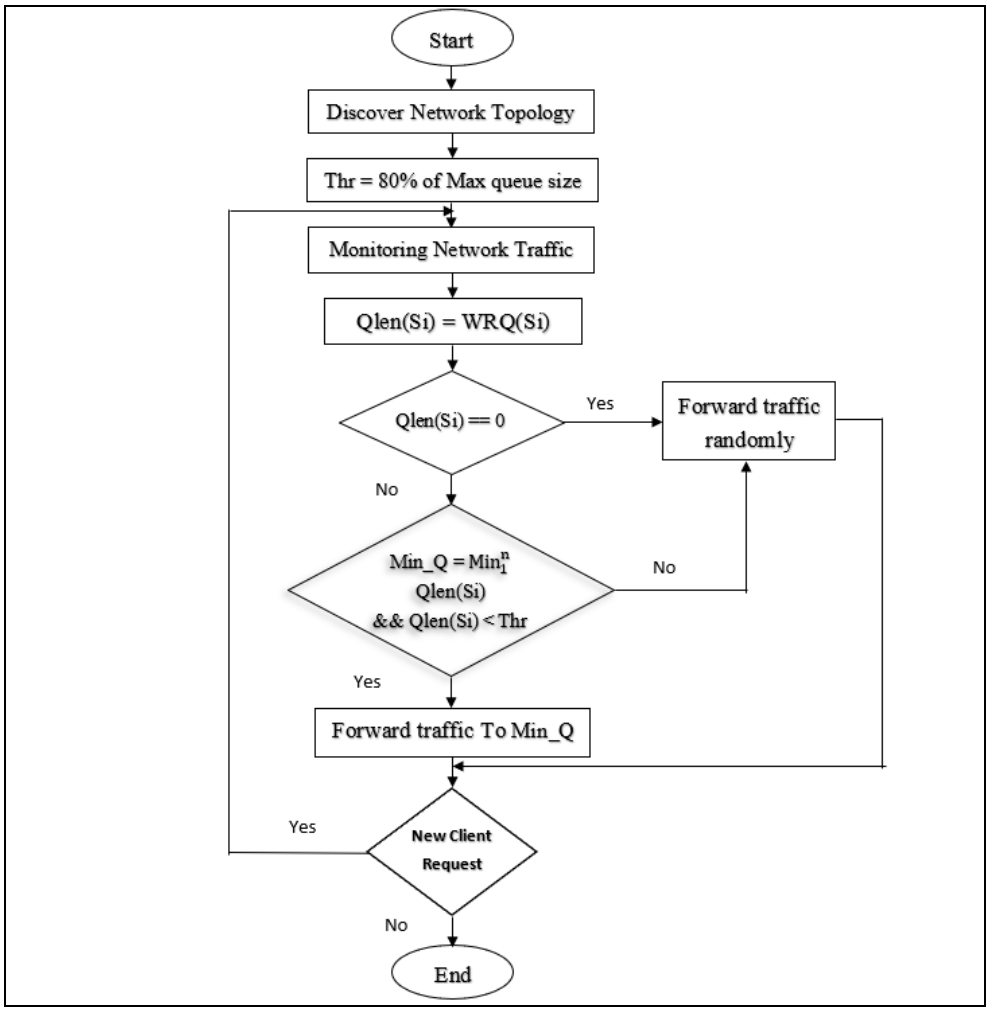

Figure 2. LQLB Flowchart

3.2 QoS aware traffic management (QTM)::

The IQLB approach has a limitation in the burst traffic because the traffic queue initialized as empty then rapidly filled and rapidly emptied. Thus, the instance queue length may not reflect the real load on each port of the switch. To address this limitation, the second approach (QTM) calculated the average queue length based on the equation (2) from [26,27] to reflect the load on the server.

$$\text{AvgQln}(ti)=(1-w) * \text{AvgQln}(ti-1) + \text{Qln}(ti)*w \quad (2)$$

Where AvgQln (ti) is the average queue length, ti is the current time and ti-1 represents the previous time, Qln(ti) is the queue length at time ti and w is the queue length weight ($0 \leq w \leq 1$, w = 0.5 is empirically selected). The flowchart of QTM illustrated in Figure 3. Topology discovered by the Pox controller is the same as explained in the IQLB flowchart. Pox Initializes variables and sets a timer. Monitor network traffic module deploy and average queue length is calculated for each port periodically (each one second is used in this work) as shown in Eq (2). Traffic will be forwarded to the server based on minimum average queue length.

IV. PERFOMANCE EVALUATION

This paper evaluated the performance of the proposed algorithms (IQLB & QTM) using Mininet [28] emulator to create network topology and POX [29] as SDN controller. The two approaches are compared with Random, Round-robin, Least bandwidth consuming and Least packet load in terms of throughput.

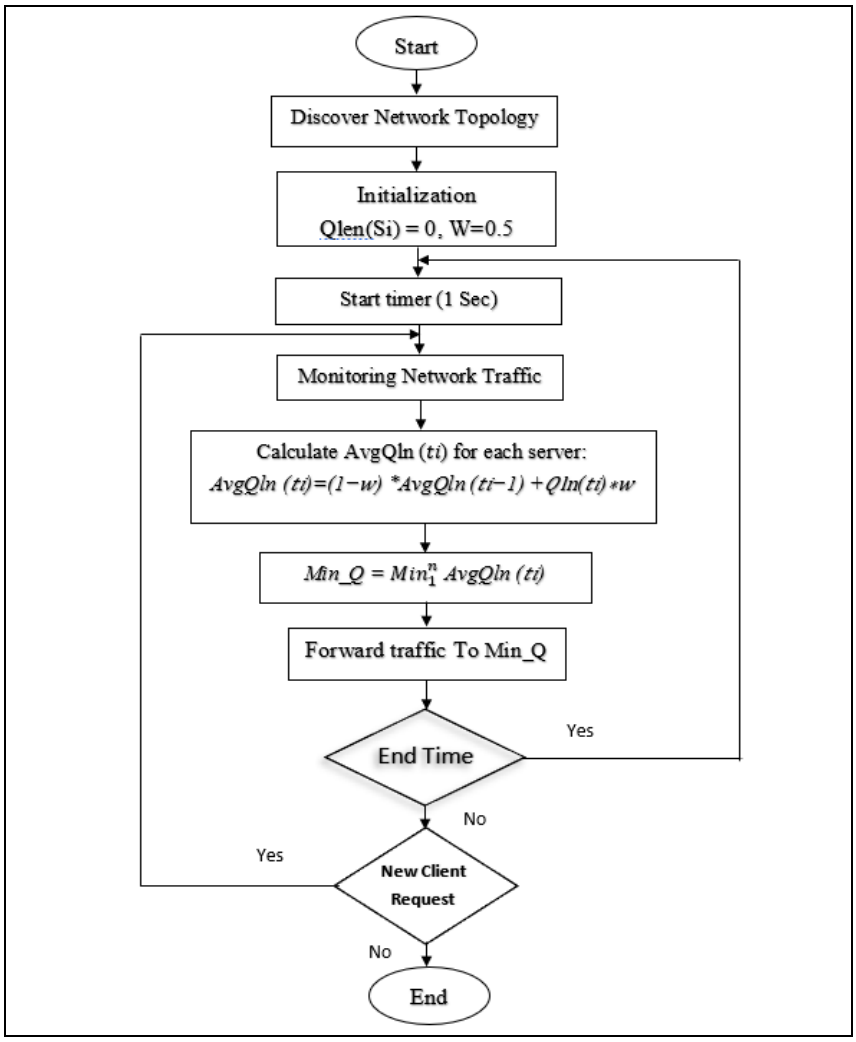

Figure 2. QTM Flowchart

4.1 Simulation Setup:

Mininet is used in conducting research in SDN and OpenFlow [30] protocol in order to evaluate the proposed algorithm. Mininet emulation is installed on version 6.0.14 Oracle Virtual box [31]. In this paper, the emulated network employs OpenFlow protocol using remote pox controller on port: 6633, two servers and a different number of clients implemented in three scenarios (4,10,20) to test the performance of the proposed method in diverse load. Iperf [32] is a network testing tool utilized to generate and test flows in different scenarios.

4.2 Evaluation and validating:

The proposed algorithms in this paper are evaluated and compared with some of popular cited benchmarks like random, round robin, least bandwidth. The results are compared in terms of throughput under various amounts of loads for each algorithm. The final results are analyzed using an analysis of variance (ANOVA) which is one of most frequently statistical methods used to verify the compared algorithms are statistically different using F > Fcrit.

Where F statistic is a variance ratio between intergroup and intragroup, Fcrit is the value extracted from analysis variance table and P is significant probability value. The approval value for P is less than 0.05. The null hypothesis at the significance level 0.05 will be rejected If F is greater than Fcrit. The information extracted from ANOVA means the obtained results are varying and at least one group differs. This means it does not provide information about which group causes the difference. Therefore, this paper utilizes Fisher's least significant difference (LSD) to show whether the proposed algorithm (QTM) is the one that causes differences. LSD value is calculated using Eq. (3) [27].

$$LSD_{A,B} = t_{0.05/2DFW}\sqrt{MSW\left(\frac{1}{nA}+\frac{1}{nB}\right)} \quad (3)$$

Where t is extracted from T-distribution with probability level 0.05, $\sqrt{MSW\left(\frac{1}{nA}+\frac{1}{nB}\right)}$ is a standard error of difference between two means and n is a number of samples [33]. Table 1 and 2 show the ANOVA and LSD results of IQLB for each scenario respectively while table 3 and 4 show the ANOVA and LSD results of QTM.

Table -1 IQLB ANOVA Result

Number of transmission nodes	**ANOVA Test**		
	F	**F_{crit}**	**P < 0.05**
4Hosts_TCP	*8.217392*	*2.410058*	*3.22E-06*
4Hosts_UDP	*11.21746*	*2.410058*	*2.4E-08*
10Hosts_TCP	*5.382161*	*2.402248*	*0.000339*
10Hosts_UDP	*28.96997*	*2.394533*	*3.94E-21*
20Hosts_TCP	*33.30041*	*2.389948*	*1.3E-24*
20Hosts_UDP	*23.75252*	*2.389948*	*5.47E-18*

Table -2 QTM ANOVA Result

Number of transmission nodes	**ANOVA Test**		
	F	**Fcrit**	**P < 0.05**
4Hosts_TCP	*27.48244*	*2.246015*	*1.21E-22*
4Hosts_UDP	*40.19975*	*2.246015*	*3.89E-31*
10Hosts_TCP	*15.4954*	*2.239486*	*8.65E-14*
10Hosts_UDP	*49.39689*	*2.233031*	*3.8E-41*
20Hosts_TCP	*43.99202*	*2.229193*	*1.28E-38*
20Hosts_UDP	*76.10665*	*2.229193*	*1.32E-61*

Table -3 IQLB LSD Result

Number of transmission nodes	**Throughput Average for the Networks (Mbps)**					**LSD**
	IQLB	Random	RR	LBW	SLPLB	
4Hosts_TCP	48.8563	*39.48375*	*40.0605*	*40.523*	*39.714*	*7.68871*
4Hosts_UDP	39.3938	28.48938	31.3711	30.757	27.709	7.57939
10Hosts_TCP	28.1589	23.32617	24.8648	25.771	24.804	2.32017
10Hosts_UDP	23.6137	14.842	15.4849	16.061	17.968	1.72199
20Hosts_TCP	20.5922	14.26855	15.1245	17.689	16.519	0.72148
20Hosts_UDP	14.2425	9.6503	11.1283	13.34	11.537	0.55084

Table -4 QTM LSD Result

Number of transmission nodes	**Throughput Average for the Networks (Mbps)**						**LSD**
	QTM	IQLB	Random	RR	LBW	SLPLB	
4Hosts_TCP	57.5208	48.8563	39.4838	40.0604	40.523	39.714	7.7928
4Hosts_UDP	50.9083	39.3938	28.4894	31.3711	30.752	27.71	7.81148
10Hosts_TCP	31.7776	28.1589	23.32617	24.8648	25.771	24.804	2.37322
10Hosts_UDP	25.5054	23.6137	14.842	15.4849	16.061	17.968	1.63748
20Hosts_TCP	21.8866	20.5922	14.26855	15.1245	17.689	16.52	0.81594
20Hosts_UDP	16.642	13.2425	9.6503	11.5363	13.3	11.128	0.51790

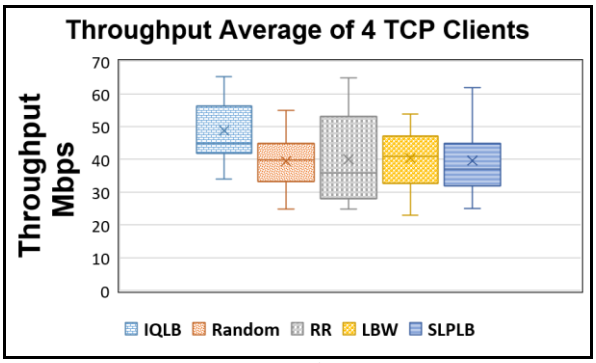

Figure 4 Throughput Average of 4 TCP IQLB Clients

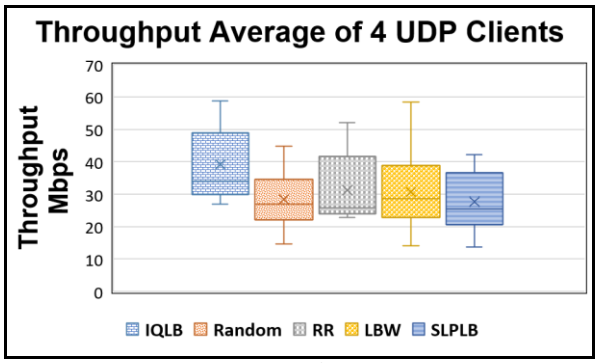

Figure 5 Throughput Average of 4 UDP IQLB Clients

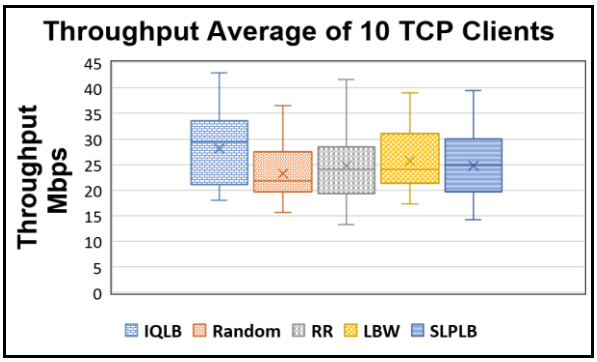

Figure 6 Throughput Average of 10 TCP IQLB Clients

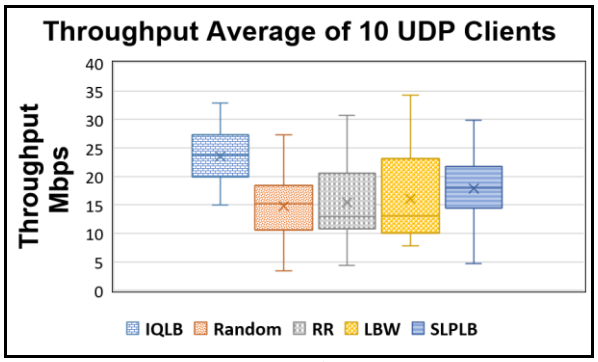

Figure 7 Throughput Average of 10 UDP IQLB Clients

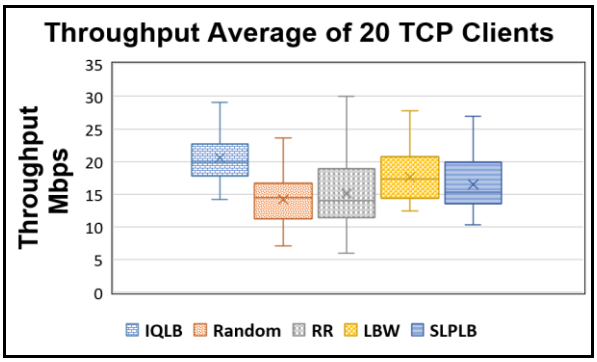

Figure 8 Throughput Average of 20 TCP IQLB Clients

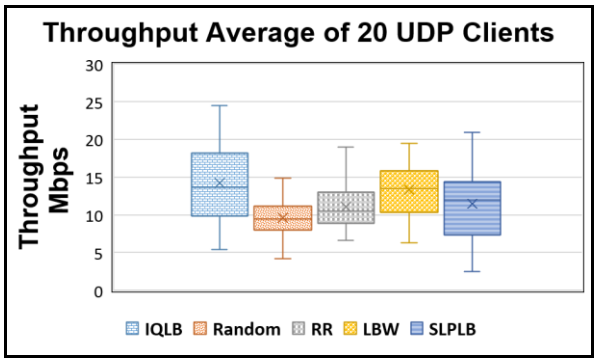

Figure 9 Throughput Average of 20 UDP IQLB Clients

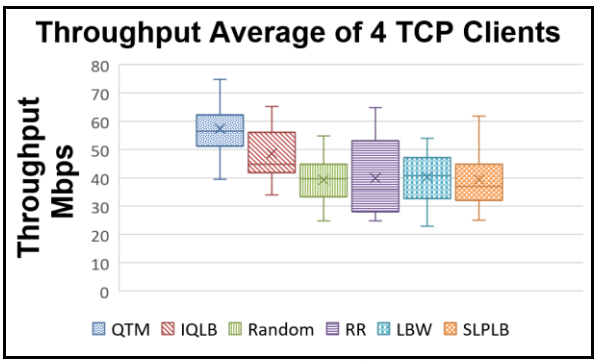

Figure 10 Throughput Average of 4 TCP QTM Clients

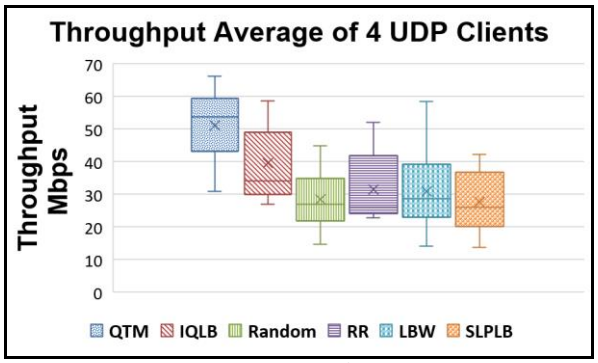

Figure 11 Throughput Average of 4 TCP QTM Clients

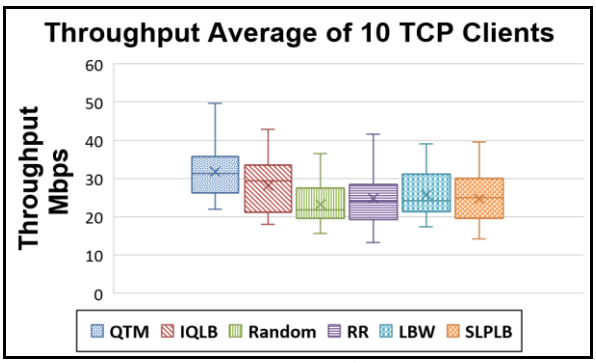

Figure 12 Throughput Average of 10 TCP QTM Clients

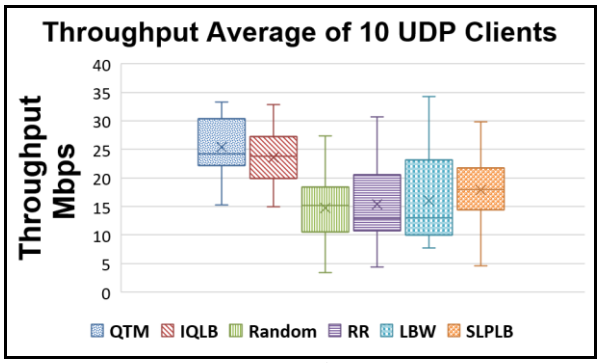

Figure 13 Throughput Average of 10 TCP QTM Clients

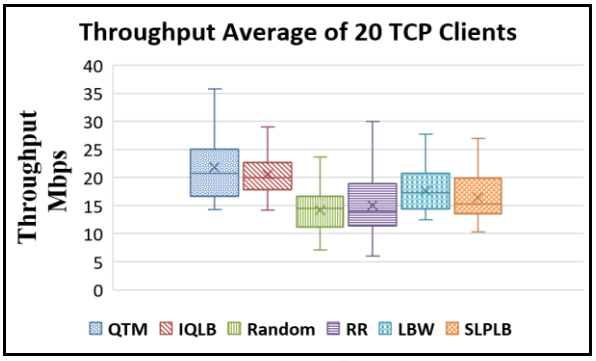

Figure 14 Throughput Average of 20 TCP QTM Clients

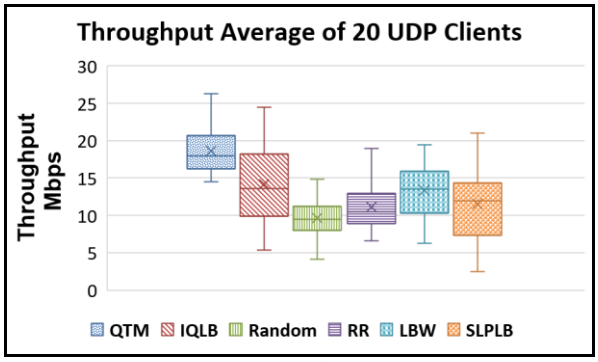

Figure 15 Throughput Average of 20 TCP QTM Clients

This paper performed extensive simulation tests. In each scenario for each algorithm, a number of various throughput results are generated. The mean value of these results is calculated for each algorithm (QTMavg, IQLBavg, Randomavg, RRavg and LBWavg). Any mean difference greater or equal LSD value considered significant. QTM significantly enhances throughput of network traffic as compared with the rest of the benchmark and IQLB. Firstly, LSD for IQLB avg is compared with all of the benchmark algorithms to validate that the proposed algorithm is causing the statistical difference as ANOVA test shows. Figures 4-9 show the results of the first approach (IQLB) of the proposed algorithm compared with the benchmark, while figures 10-15 show the second approach (QTM) results. QTM performance is significantly higher than other benchmarks because the difference between the average results are greater than the LSD. Each approach in the proposed algorithm is represented by a Box and Whisker graph. Box and Whisker plot (Box plot) is a useful graph tool in interpreting the distribution of data and understanding information that we work on. The Box in the Box plot is divided by the median to show the average throughputs above and below the median. The highest and lowest values of the results are represented by the upper and lower whiskers. For example, figure 15 illustrates the QTM throughput results between 21 and 16.5 Mbps while the benchmark algorithms achieve throughput between 18 and 10 Mbps (with an increase of 22.5%). Furthermore, QTM outperforms all benchmarks with different load on the network.

V.CONCLUSION AND FUTURE WORK

This paper introduces a new Software defined networking based load balancing algorithm that addresses the problem of network congestion and server overloading by monitoring the queue length on switches ports. The queue length is employed using two approaches, the first one utilizes the instance queue length and forward traffic to the port with the least load at the time of transmission. The second approach monitors the incoming packet on each switch port to estimate the load based on queue length over a time period. The suggested approaches outperform the benchmark algorithms in different network traffic types and load. The queue length in this paper is employed to provide load balancing among servers connected to switch ports. One future direction for this work is to utilize queue length with other QoS parameters to reduce congestion on switches and selecting the best routing path. Moreover, this work could be employed to implement a test bed with a multi-controller system instead of a single controller to avoid single point failure in the network.

REFERENCES

[1] Global Internet Growth and trends, Accessed on FEB, 2020, [online] available: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1955935

[2] Z. Shang, W. Chen, Q. Ma, and B. Wu, "Design and implementation of server cluster dynamic load balancing based on OpenFlow," in Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), Japan, pp. 691–697, 2013.

[3] A. Morreale, M. Anderson," Software Defined Networking: design and deployment," 1st ed. New York: CRC Press,2015, p28.

[4] P. Manzanares-Lopez, J. Muñoz-Gea, J. Malgosa-Sanahuja, A. Cruz "a virtualized infrastructure to offer network mapping functionality in SDN networks," International Journal of Communication Systems - Wiley Online Library, Vol.32, issue 10, 2019

[5] M. Marchese, "QoS Over Heterogeneous Networks," U.K.: Wiley, p. 101, 2007.

[6] C. Wilson, H. Ballani, T. Karagiannis, A. Rowtron," Better never than late: meeting deadlines in datacenter networks," in: Proceedings of the ACM SIGCOMM 2011 Conference, in: SIGCOMM, Toronto, Ontario, Canada, pp. 50–61, 2011.

[7] A. Osman, "Service based load balance mechanism using Software-Defined Networks," Thesis of Doctor of Philosophy, University of Malaya, pp. 32-33,2017

[8] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN Control: Survey, Taxonomy and Challenges," IEEE Communications Surveys & Tutorials, vol. 20, pp. 333-354, Dec. 2017.

[9] U. Zakia, H. Yedder, "Dynamic Load Balancing in SDN-Based Data Center Networks," 18th IEEE annual information technology, electronics and mobile communication conference (IEMCON), Vancouver, BC, Canada, pp. 242-247, 2017.

[10] V. Chakravarthy, B. Amutha, "A novel software-defined networking approach for load balancing in data center networks," International Journal of Communication Systems - Wiley Online Library, 2019, [online] doi:10.1002/dac.4213

[11] S. Rekha, C. Kalaiselvi, "Review of Scheduling Methodologies of Virtual Machines (VMs) In Heterogeneous Cloud Computing," International Journal of Scientific & Technology Research, Vol. 8, ISSUE 09, 2019.

[12] D. Devi, V. Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Non preemptive Dependent Tasks," Sci. World Journal, Vol. 2016, pp. 1-14, 2016.

[13] W. Prakash, "DServ - LB: Dynamic server load balancing algorithm," International Journal of Communication Systems - Wiley Online Library, Vol. 32, issue 1, pp. 1–11,2019

[14] N. Ganesh, S. Ranjani, "Dynamic Load Balancing using Software Defined Networks," in: International Conference on Current Trends in Advanced Computing (ICCTAC-2015), Bengaluru, India, 2015.

[15] A. Ghaffarinejad, R. Syrotiuk" Load Balancing in a Campus Network using Software Defined Networking," in 2014 IEEE Third GENI Research and Educational Experiment Workshop, Atlanta, GA, USA,2014.

[16] S. Kaur, K. Kumar, J. Singh, and N. Ghumman, "Round-robin based load balancing in Software Defined Networking," in 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi (INDIA), pp. 2136–2139, 2015.

[17] M. Koerner and O. Kao, "Multiple service load-balancing with OpenFlow," in 2012 IEEE 13th International Conference on High Performance Switching and Routing, Belgrade, Serbia, pp. 210–214, 2012.

[18] G.Tiwari, V. Chakaravarthy, and A. Rai, "Dynamic load balancing in software defined networking," International journal engineering advanced technology, pp. 2706–2712, Vol. 8, Issue. 5, 2019.

[19] Z. Cao, E. Zegura, "Performance of hashing-based schemes for Internet load balancing," INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications, Israel, pp.332 – 341 Vol. 1, 2000.

[20] F. Akyilid, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," IEEE Network, pp. 52-58, Vol. 30, issue 3, 2016.

[21] H. Zhang, X. Guo, J. Yan, B. Liu, and Q. Shuai, "SDN-based ECMP algorithm for data center networks," 2014 IEEE Computers, Communications and IT Applications Conference, Beijing, China, pp. 13-18, 2014.

[22] L. Padilha and D. Batista, "Effectiveness of Implementing Load Balancing via SDN," in ETRI Journal. pp 197-206, Vol. 41, 2019.

[23] J. Yan, H. Zhang, Q. Shuai, B. Liu, "HiQoS: An SDN-based multipath QoS solution," China Communications, pp.123-133, Vol. 12, issue:5, 2015.

[24] A. Jassar, "An analysis of QoS in SDN-based network by queuing model," Telecommunications and Radio Engineering Journal, pp. 297–308, Vol. 77, no. 4, 2018.

[25] L. Boero, M. Cello, C.Garibtto, M. Marchese, M. Mongelli, "BeaQoS:Load balancing and deadline management of queues in an OpenFlow switch," Computer Networks journal, pp. 161-170, Vol. 106, 2016.

[26] X. Ao, S. Jiang and L. Tang," Traffic-aware active link rate adaptation via power control for multi-hop multi-rate 802.11 networks", IEEE 12th International Conference on Communication Technology, Nanjing, China, pp. 1255-1259, 2010.

[27] A. Al-Saadi, R. Setchi, Y. Hicks, and S. M. Allen, "Multi-rate medium access protocol based on reinforcement learning," 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, pp. 2875–2880, 2014.

[28] Mininet: An Instant Virtual Network on your Laptop (or other PC) – Mininet, Accessed Feb 2020, [online] available http://mininet.org/

[29] "Installing POX — POX Manual Current documentation," Accessed on 14 Feb 2020, [online] Available https://noxrepo.github.io/pox-doc/html/

[30] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, pp. 69–74, Vol. 38, Issue: 2, 2008.

[31] https://www.virtualbox.org/ , Accessed on 14 Dec 2019.

[32] C. Partsendidis, "what is iperf and how is it used", Accessed FEB, 2020, [online] available https://searchnetworking.techtarget.com/answer/What-is-iPerf-and-how-is- it-used

[33] R. Lyman, T. Longecker, "An introduction to statistical methods and data analysis," 6th ed, USA, Thomson Learning Academic Resource Center, pp.463,2010.