Load balancing for Software Defined Network using Machine learning

Aashish kumar[a] and Darpan Anand[b]

[A] Research Scholar, Department of C.S.E., Chandigarh University Mohali, India
[b]Associate Professor, Department of C.S.E., Chandigarh University Mohali, India.

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: Software-Defined Networking is one of the most revolutionary and prominent technology in the field of networking. It solves the problem that our traditional network faces. Still it can face a problem of bottleneck and can be overloaded. To overcome this issue, various researcher has it given various works but they are based on two or three-parameter to perform load balancing and also they are static or dynamic. We have proposed an intelligent technique that forwards the packet i.e. TCP/UDP packet traffic based on several parameters (based on 12 parameters discussed in the latter part of this section). Based on these parameters, we have applied the trained machine using KMeans [1] and DBSCAN [2] clustering algorithm and also determine the optimal number of clusters. We have tested it on the huge number of packet that are 5000, 10000, 20000, 50000, 100000, 10000000.We have also compared there results of the KMeans and DBSCAN algorithm and also discussed researchers view

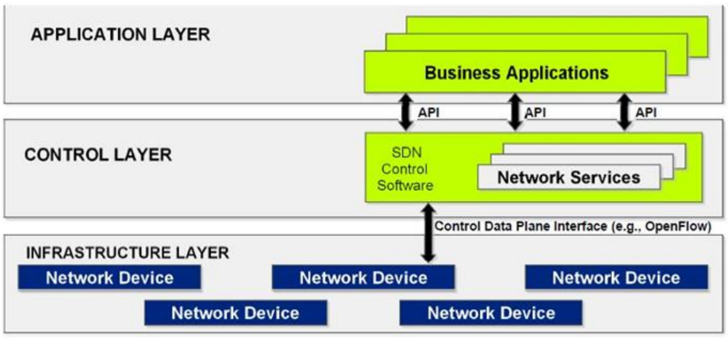
Keywords: Software defined Network, Load Balancing, Communication Network, and Network Performance.

1. Introduction

Software Defined Network [3]solves many problems that our traditional network is facing. But as we know that a huge amount of data is generated in our era. It might possible that the controller in SDN becomes overloaded.To overcome this issue, many researchers have given their view to solve problem of controller overloading. But they have taken two or three-parameter to balance the load in SDN.So there is need of a machine learning technique which can take various parameter and solve the problem of load balancing.These issues can be solved by machine learning techniques. So, we are going to discuss SDN, Load balancing and clustering in machine learning [4] in a later section of this paper.

1.1. Software Defined Network

Traditional systems have tight coupling between the control plane [5] andinformation plane [6] yet this prompts the issue of dynamic IP allocation, change in routing, bandwidth management end to end reachability, etc. SDN solves these problems and separates the control layer and information layer.So,we can define SDN as a network in which the control layer(may be located on different geographical locations) is physically separated from the information layer and a logically centralized regulator governs the routing devices.Main layers of SDN arcitecture(as shown in Fig1) are



– Data Layer:It contains switching devices like router, switches etc. It is responsible to process and forward the packets as per the rules de- fined in the forwarding table.
– Control Layer :It is also known as NETWORK BRAIN.It is responsible for routing the data. Some of the function this layer are System Configuration, routing table information's exchange and management
– Management plane:It is used to access and provides in management of our network equipment.Still, SDN has the issue of standard protocol that can exchange between informationplane and control plane. In 2008, the issue is tackled by the Open Flow protocol [7] which is famous southbound API in the SDN and maintained by the Open Flow Networking Foundation(ONF) [8]. Some of the components of Open flow are:-
– Flow table

– Port
– Messages
Whatever, we have discussed is about SDN but SDN controller can be overloaded by traffic. So we will discuss load balancing, load balancer in SDN in later section.

1.2 Load Balancing

It is a process used to spread the load between servers or any other computing equipment. The purpose of load balancing is to maximize resource utilization, maximum throughput, reduce response time, avoid overload and avoid crashing. It also used to avoid from failover. Customarily load balancer comes programming introduced on equipment. Along these lines, it was merchant explicit and costly. The product load balancer runs on a virtual machine or hardware.

1.2.1 Load balancing Types

– Transport layer load balancer [9]: In this method, load balancer utilizes data , for example, IP address of source and objective and ports characterized at data header of the information.
– Application Layer load balancer [10It circulates the heap to the servers dependent on application layer conventions, for example, HTTP, COOKIES or information.
In our proposed system, we relate with layer 4 load balancer.

1.2.2 Load balancing in SDN

There are two ways to deal with balance the heap in SDN as a centralized and distributed methodology. In a central methodology, there is a super-regulator that adjusts the heap between different regulators (ordinary controllers). The issue with this methodology is that if the super-regulator comes down, the entire organization gets down and other issues are adaptability and accessibility issues. The super-regulator methodology was settled by a distributed methodology. In the distributed methodology, there are a few regulators that balance the heap between them. Yet, the issue is that correspondence is overhead. Presently, we will examine different work done by researchers.

1.3 Tabular Related Work

This section shows the previous work done by the researchers.It include advantages and disadvantages,and the parameter they have used for load balancing.
The tabular format is shown as Table 1

Sr no	**Year**	**Author Name**	**Algorithm Used**	**Advantages**	**Disadvantages**	**Basis of Descision**
1	2016	Sufiev et al [11]	Dynamic Cluster	Reduced latency Super controller does not depend on Regular Controller	Single point of failure Scalability is- sues	
2	2014	Chou et al [12]	Genetic Based Load Balancing	Avoid Bottleneck Save cost	High Complexity and Computation time Scalability issues	Arithmetic average for the coefficient of variation metric
3	2017	Hu et al [13]	Switch-Migration based decision making	Low Response time	Not tested in large scale wireless network	Real time controller load information .
4	2015	Wang et al [14]	Based on Distributed Architecture	Efficient adjustment of traffic flow Reduced consumption overhead Solve reliability problem	Communication overhead	ICMP Packet testing
5	2014	Zhang et al [15]	Hybrid routing	Achieve near optimal load	Latency is not considered.	

				balancing Increase throughput Reduced TCAM		
6	2016	He et al] [16	Swarm Optimizatio n	Decrease latency Increased QoS	Security issues Energy Consumption	
7	2016	Yong et al [17]	Load balancing Technique Based on SDN	High throughput	Scalability issues Availability issues Latency is not considered	Hased based
8	2017	Zhong et al [18]	Load Balancing based on Server Response Time	Easy to implement Low Response time	Low availability Low scalability System bottleneck	Low availability Low scalability System bottleneck
9	1017	Rangi setti ˜and Tamm a [19]	QoS Aware Load Balancing algorithm(Q ALB	Improved GBR satisfaction Better QoS data rates	Jitter and Delay is not considered Scalability issues	Loads of neighbour cells. QoS profiles of UEs. Throughput
10	2018	Filali et al [20]	Optimizatio n algorithm	Minimize latency between SDN controller and Switches	Jitter and Throughput is not considered	Response Time. Resource utilization.
11	2017	Shang et al [21]	Service-Oriented Load Balancing Mechanism	improve the average link bandwidth utilization rate up to 79% with smaller link load jitter and average link delay	Overhead of maintaining mean flow request deviation table	Average link bandwidth utilization Link load jitters Average link delay
12	2016	Seung and Kwon [22]	Centralized based Load balanced based on Genetic Algorithm		Single point of failure Communicati on overhead	CPU utilization Packet messages
13	2020	Rupan i et al [23]	Backpropag ation Artificial neural network	It is scalable, link failure and node failure	Highly Complexity	
14	2016	Chen and Xu [24]	Backpropag ation Artificial neural network	Achieve 19.3% network latency Decrease at most.	Complex	Bandwidth utilization ratio Packet loss rate Transmissio n latency

						Transmissio n hops
15	2017	Ying et al [25]	Two tier architecture	Hardware independent Improves WiFi s load balancing degree by 34 to 41% An improvement of 28 to 36% in WiFi s re association Time.	Not consider traffic patterns of the associated devices, user priorities and QoS constraints	Re association requests
16	2018	Kavan a et al [26]	Load balacing algorithm	it requires least hardware		
17	2018	Chen et al [27]	Traffic-aware load balancing scheme	Reduce service response time up to 50%	Not included IoT security in the proposed scheme	
18	2019	Aly et al [28]	Controller adaptive Load balancing	Throughput increased to 12% Response time increased to 9%	Single point of failure Low availability	No of requests on controller

2. Proposed Methodology

We have used Software Jupyter notebook to implement our methodology. Our main is to propose a technique for the Software-Defined network which considers various parameters.For this, we have taken a dataset from Kaggle.com of Universidad Del Cauca Popayan, Colombia. The proposed methodology diagram is shown in Fig 2. In our proposed methodology, when clients sends request for particular service to the server. it first sends to the Software Load balancer(the al-gorithm is implemented here) and the first algorithm will obtained the flow statis-tics (IP addresses, ports,inter-arrival times, etc) using CIC Flowmeter [30]. Based on these features, it calculates the cluster value of request and it will send to the appropriate servers. We will discuss each thing in the upcoming sections.

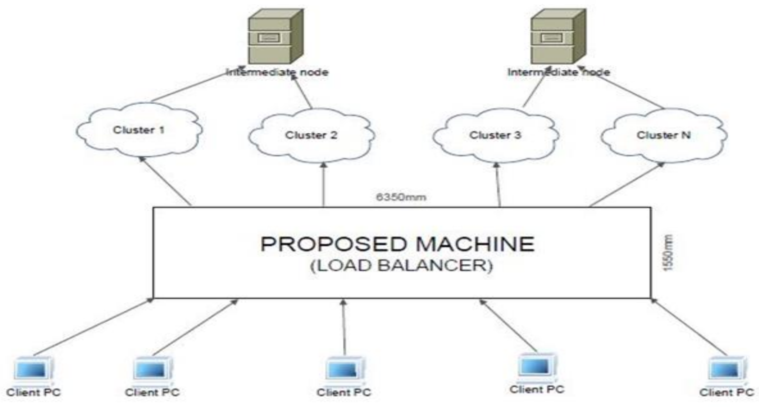


Fig.2. Proposed Work Architecture

2.1 Flowchart of Proposed algorithm

In our proposed methodology, when clients send requests for particular service to the server, it first sends to the Software Load balancer (an algorithm is imple-mented here) and the first algorithm will obtained statistics using CIC Flowmeter. Based on these features, it calculates the cluster value of request using KMeans and also checks the threshold of respective server(No of request it can handle). If it is less than the threshold, it will send to the intermediate nodes. This is how it will work.Let us discuss about the dataset in the next section.Load balancing for Software Defined Network using Machine learning 9

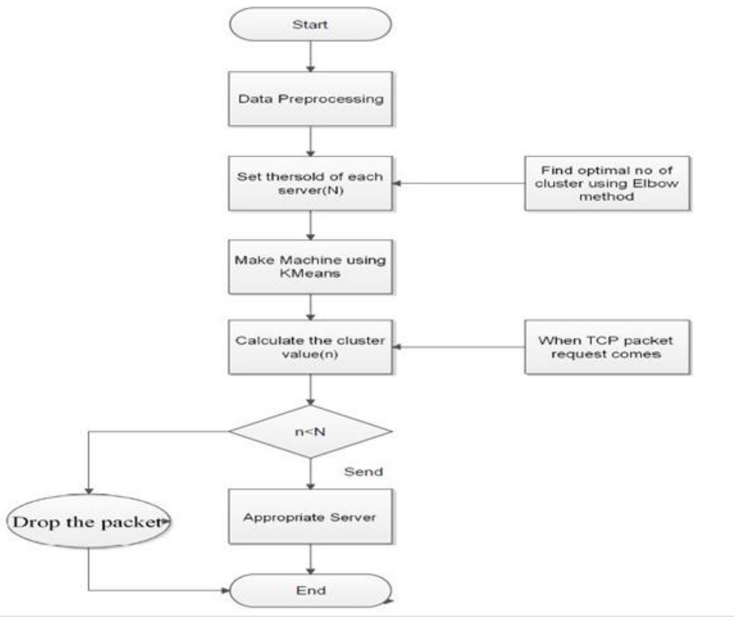

Start
Data Preprocessing
Set thersold of each server(N)
Find optimal no of cluster using Elbow method
Make Machine using KMeans
Calculate the cluster value(n)
When TCP packet request comes
n<N
Send
Drop the packet
Appropriate Server
End

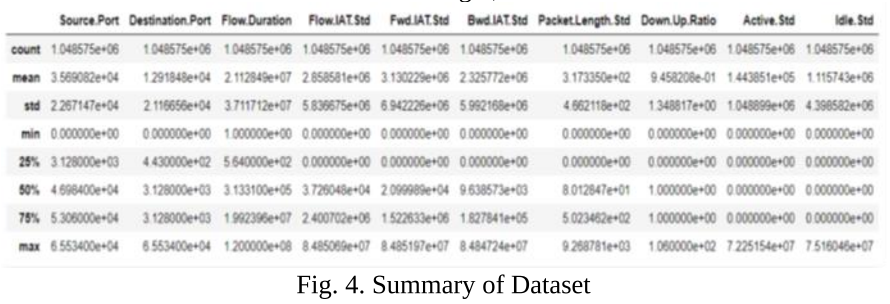
Fig. 3. Flow Chart of Proposed Work

2.1.1 Data Set

The dataset is taken from the Kaggle website, basically it was captured from the Universidad Del Cauca, Popayan, Colombia United States of America. It is the morning and evening of 2017. A total of 3577296 packets were captured,out of these 100000 packets are used to train our machine. This format has 87 features but we have taken 12 features to balance the load which are described below:

– Source.IP: The source IPV4 address of the client
– Destination.IP: The source IPV4 address of the destination.
– Source.Port: The source port number.
– Destination.Port:The destination port number.
– Flow.Duration: Total Flow duration(millisecond(ms))
– Flow.IAT.Std: Standard deviation of the inter-arrival time of the packets.
– Fwd.IAT.Std: Standard deviation of inter-arrival time from source to destina-tion.
– Bwd.IAT.Std: Standard deviation of inter-arrival time from destination to source.
– Packet.Length.Std: standard deviation of the length of the packets of the packet in both ways.
– Down.Up.Ratio: Download and transfer proportion.
– Active.Std: Standard deviation time of the packet before getting inactive
– Idle.Std: Standard deviation time of the packet before getting active.

Basic Statistics of the dataset is summarized below: In Fig 4, we can see there



	Source.Port	Destination.Port	Flow.Duration	Flow.IAT.Std	Fwd.IAT.Std	Bwd.IAT.Std	Packet.Length.Std	Down.Up.Ratio	Active.Std	Idle.Std
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	3.569082e+04	1.291848e+04	2.112849e+07	2.858581e+06	3.130229e+06	2.325772e+06	3.173350e+02	9.458208e-01	1.443851e+05	1.115743e+06
std	2.267147e+04	2.116656e+04	3.711712e+07	5.836675e+06	6.942226e+06	5.992168e+06	4.662118e+02	1.348817e+00	1.048899e+06	4.398582e+06
min	0.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	3.128000e+03	4.430000e+02	5.640000e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	4.698400e+04	3.128000e+03	3.133100e+05	3.726048e+04	2.099989e+04	9.638573e+03	8.012847e+01	1.000000e+00	0.000000e+00	0.000000e+00
75%	5.306000e+04	3.128000e+03	1.992396e+07	2.400702e+06	1.522633e+06	1.827841e+05	5.023462e+02	1.000000e+00	0.000000e+00	0.000000e+00
max	6.553400e+04	6.553400e+04	1.200000e+08	8.485069e+07	8.485197e+07	8.484724e+07	9.268781e+03	1.060000e+02	7.225154e+07	7.516046e+07

Fig. 4. Summary of Dataset

8 rows, which describe details about dataset. These parameters are described below as:

– count: return number of elements in particular column.
– mean: return averages of the particular column.
– std : return standard deviation of particular column.
– min : return minimum value in particular column.
– 25,50,75 : return percentile value of all numeric values in a column.
– max : return maximum value in particular column.

All above statistics are useful to perform data preprocessing.In the next section, we will discuss Data preprocessing.

2.1.2 Data Preprocessing

As we know that Data preprocessing is the technique to process the raw data so that it can be used in an efficient way. Steps involved in data preprocessing are:

– Data Cleaning

– Data Transformation
– Data Reduction

In our methodology, we first check the missing data, noisy data, as well as data type of each of the parameters. These are performed by pandas module, Seaborn module of python language. Whatever, we get the result, we performed hashing on these datasets by using apply map() method. To normalize data, so that it can lie between 0 and 1.We apply min-max scaling, Min-max scalar works by using the following formula:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1)$$

where x is the particular datapoint,max(x) is maximum value in the data-points,min(x) is minimum value in the datapoints. This can be achieved by sklearn.preprocessing.MinMaxScaler class. Now we have data ready for train-inpurposes.Next step is to train the machine using KMeans as discussed in next section.

2.1.3 Training Model

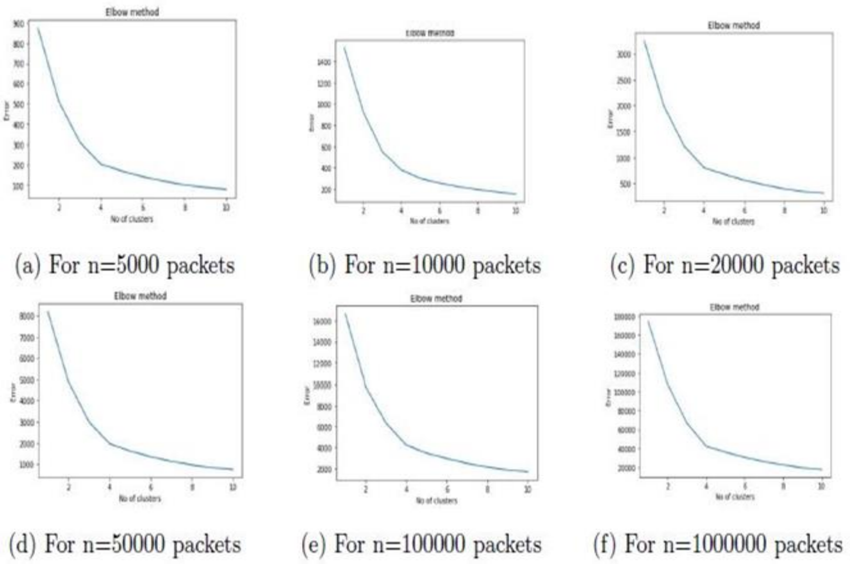
In this module, we use KMeans and DBSCAN to train or to perform clustering based on flow statistics(12 parameters) of our dataset. KMeans, DBSCAN. Both use a parameter called Euclidian distance. Greater the Euclidean distance, the lower will be the similarity between the data points or vice versa. In n-dimensional space, Euclidean distance between two data points can be calculated as:

$$dis(x, y) = \sqrt{\sum (x_i - y_i)^2)} \quad (2)$$

where xi, yiare the ith data point KMeans algorithm aims at minimizing an objec-tive function known as Square error function given by:

$$J(v) = \Sigma\Sigma(|x_i - v_j|^2) \quad (3)$$

One of the major problem KMeans clustering faces that is finding the optimal-number of clusters. This can be solved by Elbow Method [32] and Sihoullte Method



(a) For n=5000 packets (b) For n=10000 packets (c) For n=20000 packets

(d) For n=50000 packets (e) For n=100000 packets (f) For n=1000000 packets

Fig. 5: Elbow method result for different number of packets

2.1.4 Elbow Method

It is quite possibly the most popular strategies to locate the ideal number of groups.It plots the value of the sum of squared error (SSE) [33] Vs values of k. The main aim is to select small SSE after that SSE will tend to decrease towards zero as the number of k increases. In the Elbow method, KMeans will runs for the entire dataset for a range of values of k. For each k, the SSE will be calculated.

$$SSE_1 = \Sigma dist(x_i - c_i)^2 \qquad (4)$$

$$SSE_2 = \Sigma dist(x_i - c_i)^2 \qquad (5)$$

$$SSE_n = \Sigma dist(x_i - c_i)^2 \qquad (6)$$

where xi is the i th datapoint and ci is the ith centroid. In our algorithm, we have result of Elbow method for different number of packets.

2.2 Silhouette Method

It is utilized to quantify how close every data point in a group to other neighbour-ing bunches. Its worth extents from - 1 to +1. An estimation of +1 shows that the ex-ample is excessively far away from its neighbouring group and excessively near the allocated bunch. So also, an estimation of - 1 demonstrates that the fact of the matter is nearer to its neighbouring bunch than its allocate group. Suppose ith is the data point, whereas a(i),b(i) is the mean distance between the point i and cluster (A),(B) respectively.

Thus, the silhouette s(i) can be expressed as

$$S(i) = \frac{(b(i) - a(i))}{\max(b(i) - a(i))} \qquad (7)$$

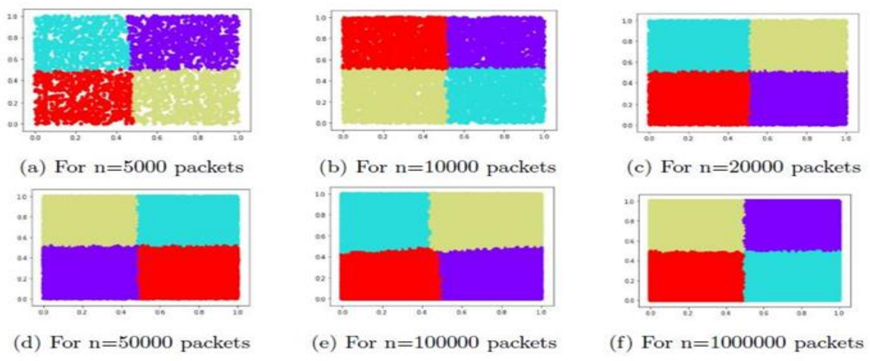


Fig. 6: KMeans Figure t ested on different number of packets

KMeans unsupervised clustering algorithm clusters the data points into spherical shape whereas DBSCAN is suitable for non-convex clusters.It is also used to iden-tify the outliers or noise We will give a short description of DBSCAN in the next section.

2.3 DBSCAN

Density-based spatial clustering of Application with noise(DBSCAN) is suitable for dataset having nonconvex clusters and having outliers. DBSCAN clustering technique requires two parameters:

1. eps
2. minPoints: It can be define as minimum number of points required to form dense region.

Now, we will discuss results obtained in next section.

3. Experimental Result

We have tested it on several number of parameter that are 5000,10000,20000,50000,100000,1000000 packets(described below) The fol-lowing graph shows that KMeans perform better than DBSCAN .Load balancing for Software Defined Network using Machine learning 13

4. Conclusion

For the proposed methodology, we conclude that KMeans perform better than DBSCAN and it can be proposed as Machine learning techniques for Software Defined Network but there are limitations to this approach. Basically it take care of a total of 12 parameters which may make it a better machine-learning algo-rithm and also take care about the forward and backward transmission parame-ter which make it intelligent and optimized methodology. In the coming years, we would like to test on SDN scenario and measure the various load balancing pa-rameters.

5. Future Work

The Proposed work need to be implemented on real world scenario such as on Software Defined Network Applications. It need to be tested on more numbers of packets to measure various parameters such as Throughput,Response time and latency etc.The proposed work needs to check time complexity and more general software need to capture TCP/IP packets which contains the more features

References

A. Chauhan, “Systems and methods for providing load balancing as a service,” Oct. 162018. US Patent 10,104,166.

A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui, “Sdn controller as-signment and load balancing with minimum quota of processing capacity,” in 2018 IEEE International Conference on Communications (ICC), pp. 1–6, IEEE, 2018.

A. K. Rangisetti, B. R. Tamma, et al., “Qos aware load balance in software de-fined lte networks,” Computer Communications, vol. 97, pp. 52–71, 2017.

A. Lashkari, Y. Zang, G. Owhuo, M. Mamun, and G. Gil, “Cic flow meter,” 2019.

A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algo-rithm,”Pattern recognition, vol. 36, no. 2, pp. 4461.2003.

B. G¨orkemli, S. Tatlıcıo˘glu, A. M. Tekalp, S. Civanlar, and E. Lokman, “Dy-namic control plane for sdn at scale,” IEEE Journal on Selected Areas in Com-munications,vol. 36, no. 12, pp. 2688–2701, 2018.

C. Chen-Xiao and X. Ya-Bin, “Research on load balance method in sdn,” In-ternational Journal of Grid and Distributed Computing, vol. 9, no. 1, pp. 25–36, 2016.

D. Michie, D. J. Spiegelhalter, C. Taylor, et al., “Machine learning,” Neural and Statistical Classification, vol. 13, no. 1994, pp. 1–298, 1994.

E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X.Xu,dbscan,” ACM Trans-actions on Database Systems(TODS), vol. 42, no. 3, pp. 1–21, 2017.

F. Hu, Q. Hao, and K. Bao, “A survey on software-defined nework and open-flow: From concept to implementation,” IEEE Communications Surveys & Tuto-rials, vol. 16, no. 4,pp. 2181–2206, 2014.

G. Shang, L. Mao, and W. Gong, “Service-aware adaptive link load balancing mechanism for software-defined networking,” Future Generation Computer Sys-tems, vol. 81,pp. 452–464, 2018.

H. Xylomenos and G. C. Polyzos, “Tcp and udp performance over a wireless lan,” in IEEE INFOCOM’99. Conference on Computer Communications. Pro-ceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Com-munications Societies.The Future is Now (Cat. No. 99CH36320), vol. 2, pp. 439–446, IEEE, 1999.

I. Farhad, H. Lee, and A. Nakao, “Data plane programmability in sdn,” in 2014 IEEE 22nd International Conference on Network Protocols, pp. 583–588, IEEE, 2014.

1. H. Kathi, S. Srinivasan, and P. Singhal, “Data traffic load balancing based on application layer messages,” July 13 2006. US Patent App. 11/031,184.
2. H. Kavana, V. Kavya, B. Madhura, and N. Kamat, “Load balancing using sdn methodology,”International Journal of Engineering Research and Technolo-gy, vol. 7, no. 5,pp. 206–208, 2018.
3. H. Sufiev and Y. Haddad, “A dynamic load balancing architecture for sdn,” in 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE),pp. 1–3, IEEE, 2016.
4. H. Zhong, Q. Lin, J. Cui, R. Shi, and L. Liu, “An efficient sdn load balancing scheme based on variance analysis for massive mobile users,” Mobile Infor-mation Systemsvol. 2015, 2015.

J. Peng and Y. Xia, “A cutting algorithm for the minimum sum-of-squared er-ror clustering,”in Proceedings of the 2005 SIAM International Conference on Da-ta Mining, pp. 150–160, SIAM, 2005.

K. Zhang, K. Xi, M. Luo, and H. J. Chao, “Load balancing for multiple traffic matrices using sdn hybrid routing,” in 2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR), pp. 44–49, IEEE, 2014.

L. Kirkpatrick, “Software-defined networking,” Communications of the ACM, vol. 56,no. 9, pp. 16–19, 2013.

5. K.-Y. Wang, S.-J. Kao, and M.-T. Kao, “An efficient load adjustment for bal-ancing multiple controllers in reliable sdn systems,” in 2018 IEEE International Conference on Applied System Invention (ICASI), pp. 593–596, IEEE, 2018.
6. L.-D. Chou, Y.-T. Yang, Y.-M. Hong, J.-K. Hu, and B. Jean, “A genetic-based load balancing algorithm in openflow network,” in Advanced technologies, em-bedded and multimedia for human-centric computing, pp. 411–417, Springer, 2014.
7. P. Bholowalia and A. Kumar, “Ebk-means: A clustering technique based on elbow method and k-means in wsn,” International Journal of Computer Applica-tions, vol. 105,no. 9, 2014.
8. S. Bhalla, P. Kwan, M. Bedekar, R. Phalnikar, and S. Sirsikar, Proceeding of International Conference on Computational Science and Applications: ICCSA 2019. Springer Nature, 2020.
9. S.-B. Kang and G.-I. Kwon, “Load balancing of software-defined network controller using genetic algorithm,” Contemporary Engineering Sciences, vol. 9, no. 18, pp. 881–888, 2016.
10. W. Braun and M. Menth, “Software-defined networking using openflow: Proto-cols, applications and architectural design choices,” Future Internet, vol. 6, no. 2, pp. 302–336, 2014.
11. W. H. F. Aly, “Controller adaptive load balancing for sdn networks,” in 2019 EleventhInternational Conference on Ubiquitous and Future Networks (ICUFN), pp. 514–519,IEEE, 2019.

12. X. He, Z. Ren, C. Shi, and J. Fang, “A novel load balancing strategy of soft-ware-defined cloud/fog networking in the internet of vehicles,” China Communi-cations, vol. 13, no. Supplement2, pp. 140–149, 2016.
13. Y. Li and M. Chen, “Software-defined network function virtualization: A sur-vey,” IEEE Access, vol. 3, pp. 2542–2553, 2015.
14. Y. Wang, X. Tao, Q. He, and Y. Kuang, “A dynamic load balancing method of cloud centerbased on sdn,” China Communications, vol. 13, no. 2, pp. 130–137, 2016.
15. Y. Zhou, M. Zhu, L. Xiao, L. Ruan, W. Duan, D. Li, R. Liu, and M. Zhu, “A load balancing strategy of sdn controller based on distributed decision,” in 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 851–856, IEEE, 2014.
16. Y.-D. Lin, C. C. Wang, Y.-J. Lu, Y.-C. Lai, and H.-C. Yang, “Two-tier dynamic load balancing in sdn-enabled wi-fi networks,” Wireless Networks, vol. 24, no. 8, pp. 2811–2823, 2018.
17. Y.-J. Chen, L.-C. Wang, M.-C. Chen, P.-M. Huang, and P.-J. Chung, “Sdn-enabled traffic-aware load balancing for m2m networks,” IEEE Internet of Things Journal,vol. 5, no. 3, pp. 1797–1806, 2018.