ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20230323000475
Original scientific paper

Development of Bandwidth Optimization and Limiter Software for Network Efficiency in Software-Defined Networks

Abdülkadir ÇAKIR, Enes AÇIKGÖZOĞLU*

Abstract: Many devices have been connected to each other and a wide platform has been formed with the development of internet technologies. The continuous expansion of this platform has revealed requirements such as single point management, accessibility, bandwidth management and efficient use of the network. Considering that software-defined networks are systematically managed by software, it is predicted that they will meet the determined network requirements more easily. In this study, software was developed that limits network traffic on a client basis by optimizing the bandwidth of clients in software-defined networks. In the proposed study, a unique dataset was created by taking the last year's data from the university network for bandwidth optimization. In order to determine the optimum client-based bandwidth, the dataset is clustered with the K-means algorithm. The instant data coming from the live network is transferred to the software as θ_{client} and the cluster to be transferred is calculated. Web-based limitation software performs network traffic limitation by including the clients in the optimum cluster according to the cluster information coming from the dataset instantaneously. A virtual network was designed for the implementation of the web-based software and tests were carried out on this network. Efficient use of the network is aimed by allocating bandwidth according to clusters created especially in multi-user, heavy-traffic networks. In addition, client-based DDoS attack detection is also carried out thanks to the network data collected instantly.

Keywords: intrusion clustering; intrusion detection; K-means; SDN

1 INTRODUCTION

Thanks to the developments in information technologies, internet technology has created a large digital platform where every device is connected to each other. This platform has revealed control and management requirements such as accessibility, smart management, and bandwidth management. Traditionally, the manual configuration of Internet Protocol networks (IP-Internet Protocol) and devices used today are difficult and complex processes [1]. Performance delays may occur during the manual configuration process. In addition, there is a risk of human error during configuration. When predetermined rules need to be updated, programming the working network and reconfiguring the network depending on the dynamic changes that may occur on the existing network require a series of difficult operations that negatively affect the time cost.

The data produced by IT assets are stored according to a certain writing standard and these data are stored in different file extensions. These loggings of operating systems, network entities, and firewalls are critical records that must be kept for the security of the system. Data generated in the operating system or network is needed, especially in the development of software-defined network solutions. In order to reveal artificial intelligence-supported solutions in possible anomaly detection or configuration processes, data should be included in a single dataset and data-cleaning processes should be applied [2].

A Software Defined Network (SDN) can be defined as a software-managed network programmatically. The SDN network is configured using the Open-Flow protocol with software written for various purposes such as performance improvement, troubleshooting, etc. on the network. Slow, expensive, and limited change in traditional networks prevent investment and network modifying of many organizations [3, 4].

Programming switchgear has become a challenge as requirements grow with ever-growing network sizes. Setting up individual network switches manually is time-consuming in large networks and businesses running multiple virtual systems. Since SDN is an approach that provides software changes on the network, monitoring and controlling the network, it facilitates the management and configuration of the network. Compared to traditional networks, the data and control plane of the switching device are separated in the SDN network [5].

In SDN networks, the control and data planes are separate from each other and the decision power is given to the SDN controller in network packet transmission. This is the reason why switching devices behave like a dumb device. This centralized control in SDN networks makes the network vulnerable to attacks such as Denial of Service (DoS), spoofing, and overrunning. Such attacks reduce SDN performance by disabling different units such as switching devices and network controllers [6, 7].

In this study, a new model proposal for clustering home network efficiency is presented in an SDN network based on historical network traffic collected over the network. In the study, backbone switching device of Isparta University of Applied Sciences Department of Information Processing was used to obtain the data.

Clustering was performed in the K-means algorithm by analyzing the data, and the IP addresses in the clusters obtained in the last step were assigned to different bandwidths with pre-defined queues to ensure effective use of the network. While collecting data, DDoS attacks were detected on the network at the same time, and the network was protected against attacks. As a result, with this SDN-based approach developed, the bandwidth of the networks was adjusted and the network was protected against attacks. The original contributions of the proposed study model and its contributions to the literature are given below.

- Use of a large study-specific dataset.
- Ensuring network efficiency in enterprises with dense and high bandwidth.
- Detection of DDoS attacks in networks with servers for the enterprise.

2 EXAMINATION OF METHODS

2.1 Software Defined Network-SDN

Software Defined Networks (SDN) technology provides ways to solve these problems. While it is necessary to program each network device (router-router, switch-switching device) separately in generally used networks, with the management software used in software-defined networks, the devices in the network are programmed from a single point and the devices are automatically configured according to the determined rules. Managing the network from a single point offers a flexible environment that increases the traceability of the network, facilitates the detection of errors that may occur on the network, and simplifies network management.

Fig. 1 shows the working diagrams of the traditional network structure and the software-based network structure.

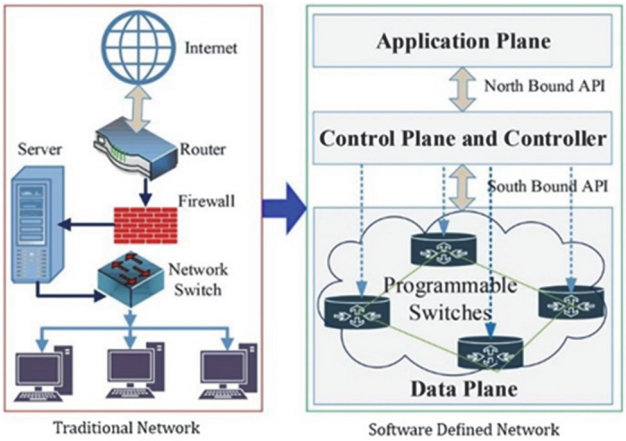

Figure 1 Traditional network and software defined shape network structures [8]

Software Defined Networks (SDN) offers the opportunity to easily manage all network infrastructures in the system together, thanks to its control panel and user interface that can be managed from a single point. In this way, the operations desired to be performed on the system can be performed quickly and easily. New technology applications planned to be created in network systems will be profitable in terms of service performance and resource usage [9, 10].

It has been experimentally shown that conventional techniques do not provide correct network behavior in case of failure. This is because these techniques only address part of the problem. Ignoring the switch state can result in inconsistencies, potentially resulting in severe network anomalies. This leads to the need to design fault-tolerant SDN solutions [11].

Software Defined Networks emerged as a challenge to the limitations of traditional network architectures. Its main advantages can be listed as programmability, centralized network view, and separate operation of the data plane and control plane. With these features, the implementation of QoS in various network applications attracts the attention of developers and researchers [12, 13].

2.2 ONOS Software Defined Network Controller

The Open Network Operating System (ONOS) is an operating system designed to help network service providers create carrier-level software-defined networks designed for high scalability, availability, and performance. While specifically designed to meet the needs of service providers, ONOS can also function as a software-defined network (SDN) control plane for enterprise campus local area networks (LAN) and data center networks.

The computer on which the ONOS software-defined network controller will be installed must have at least a 2-core processor, 2 GB of RAM, 10 GB of hard disk, 1 network controller and a Linux operating system. It is recommended not to run ONOS services with root users on Linux operating system. For this reason, it is recommended to create a user named "SDN" for the ONOS application. Since ONOS is a java-based application, java and its plugins must be installed before starting the installation [14].

SDN control software used today has been examined in terms of programming language, interface support, documentation, modularity, platform support, southern part API, and northern part API criteria and shown in Tab. 1 comparatively.

Table 1 Feature-based comparison of popular open source SDN control software [15]

SDN Software	ONOS	OPEN DAYLIGHT	NOX	RYU
Programming Language	Java	Java	C++	Python
GUI	Web	Web	Python	Python
Documentation	Good	Very Good	Poor	Fair
Modularity	High	High	Low	Low
Platform Support	Linux, Mac OS, Windows	Linux, Mac OS, Windows	Linux	Linux
SouthboundAPI	OF 1.0, 1.3, Netconf	OF 1.0, 1.3, 1.4, NETCONF/YANG, OVSDB, PCEP, BGP/LS, LISP, SNMP	OF 1.0	OF 1.0, 1.2, 1.3, 1.4, NETCONF, OFCONFIG
NorthboundAPI	Rest API	Rest API	Rest API	Rest API

2.3 Mininet Network Simulation Application

Mininet is a network emulator that can network with virtual computers, virtual switches, virtual controllers, and virtual connections. Mininet allows us to build experimental networking on a computer such as research, network development, prototyping, learning, debugging, and testing [16].

The benefits of the Mininet application can be listed as follows [17]:

- Having tools to create complex network topologies without establishing a physical network structure,
- Having networking tools to develop OpenFlow applications,
- Having a command line interface to run tests on the network,

- It has a simple Python API for creating and managing networks.

In order to install the Mininet virtual networking application, firstly, after downloading the mininet project to the local repository via GitHub, it is necessary to select the desired version and start the installation. Installation is done with the "install.sh-a" command in the downloaded mininet project [16].

The steps and the python code equivalents used to create a sample network with pseudocode on Mininet are given in Tab. 2.

Table 2 A sample network created with Pseudocode and Python equivalents

Create an empty topology	def build (net)
Add s1 to s6 switching devices in topology	s1 = net.addSwitch('s1')
Add hosts h1 to h6 to topology and set IP addresses	h1 = net.addHost('h1', cls=Host, ip='10.0.0.1')
Connect each host with a switching device	net.addLink(h1, s1)
Interconnection switching devices	net.addLink(s1, s2)
Add SDN controller to topology	c0=net.addController(name='c0', ,ip='127.0.0.1' ,port=6633)
Run topology	net.build()

2.4 SFLOW Protocol

NetFlow is a traffic monitoring technology developed by Darren and Barry Bruins at Cisco in 1996. It defines how a router exports information and statistics of routed sockets. As a de facto industry standard, it is a built-in feature of most routers and switches from Cisco, Juniper, and other vendors. There are several versions of NetFlow from v1 to v9. v5 and v9 are the most used versions [18].

Packet sampling (sFlow) of traffic flow has a long history before NetFlow was developed. Sflow is a technology supported by manufacturers such as Alcatel, Extreme, Force10, HP, and Hitachi, which uses simple random sampling and produces switches and routers that include the sFlow tool. sFlow is software that combines interface counters and flow samples into sFlow datagrams and sends these data to sFlow collectors via UPD [18].

sFlow-RT is an application with InMon's asynchronous analytics technology, providing real-time visibility into Software-Defined Networking (SDN) and DevOps stacks, and introducing new classes of performance-sensitive applications such as load balancing, DDoS mitigation, and workload placement [19].

3 DESIGN OF ATTACK DETECTION AND CLUSTERING SYSTEM

This SDN-based approach, which works to improve the resource consumption of network entities by limiting user speeds in complex networks according to their average usage, consists of four stages. The first of these stages was to create a virtual network in the mininet environment and communicate with the ONOS software-defined network controller for attack detection. In the second step, data were collected from both the created network and a real network device with the slow protocol, possible DDoS attacks were detected and the data were examined. The traffic data collected in the third stage were analyzed with the help of the K-means algorithm and divided into groups. In the fourth and last step, the IP addresses in the separated groups were assigned to the predefined Qos on the network device and bandwidth selection was made.

3.1 Creating the Dataset for Clustering

The data to be used by the K-Means clustering for speed limitation were collected with Sflow fromIsparta University of Applied Sciences network. In Fig. 2, the screen output of the interface used in the data collection process with Sflow is given. More than 7.500.000 records in total are stored in PostgreSQL after data cleaning.

```
2022-09-28 22:57:32.843874 10.251.251.154 50.7.24.18 53603 443 6 74
2022-09-28 22:57:32.846247 10.4.7.212 212.154.111.81 49772 443 17 98
2022-09-28 22:57:32.849323 10.2.0.10 10.3.23.97 0 0 47 1356
2022-09-28 22:57:33.844029 185.180.14.15 10.146.0.180 54980 49571 6 1522
2022-09-28 22:57:33.847389 185.255.92.77 10.251.251.138 25564 59197 6 85
2022-09-28 22:57:35.843927 10.2.33.92 10.2.0.10 0 0 47 144
2022-09-28 22:57:35.847377 52.85.5.43 10.141.1.100 443 53890 6 1502
2022-09-28 22:57:36.844237 161.9.213.69 88.234.211.160 1194 52046 17 1472
2022-09-28 22:57:36.847647 161.9.212.220 188.3.117.149 443 26421 6 1474
2022-09-28 22:57:38.373167 10.2.0.10 10.3.23.97 0 0 47 1356
2022-09-28 22:57:38.376661 161.9.212.150 95.9.136.252 443 53154 6 1462
2022-09-28 22:57:40.844879 161.9.212.150 185.191.171.1 443 57844 6 1522
```

Figure 2 Data collection screen output with Sflow

3.2 Designing the Virtual Network

In our study, a campus network example was designed in the mininet virtual network emulator. The view of the designed network structure on the ONOS software-based network controller is given in Fig. 3. In the figure, the user computers represent the hosts used to perform the DDoS attack, and the switch groups represent the switching devices over which the data is collected. ONOS software-based network controller manufacturer tests indicate that it works smoothly up to 3500 switching devices with a single controller. After this value, it is stated that Open Flow channels become unstable [14]. For the designed campus network example, 10 switching devices were created.

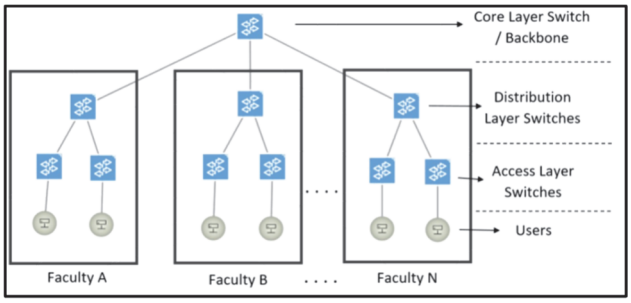

Figure 3 ONOS view of the virtual network created in the Mininet emulator

"ovs-vsctl" command from Openflow protocol commands is used for sflow protocol setting of switching devices in the created network environment. With the help of this command, it is ensured that the traffic passing through the switching devices is directed to the sflow-rt application.

There are some parameters used when configuring sflow in switching devices. With the "target" parameter, the IP address of the sflow aggregator server is entered. With the "header" parameter, the header size of the sflow package is determined. The "sampling" parameter is the ratio of the number of packets arriving at a port to the number of samples received from these packets. Port speed should be taken into account when selecting the sflow

sample rate. Sampling selection ranges according to port speed are given in Tab. 3 [20]. The "polling" parameter determines how often sflow packets will be sent to the sflow collector.

Table 3 Sflow sampling selection ranges

Link Speed	Sampling Rate
10 Mb/s	1-200
100 Mb/s	1-500
1 Gb/s	1-1000
10 Gb/s	1-2000

"target=onos-ip-address" header = 128 sampling = 400 polling = 30" parameters and values were entered to the switching devices used in the study.

Data from Sflow configured network switching devices were transferred to the database with a generated python script. Meanwhile, DDoS attacks that may occur with the DDoS prevention application of the Sflow-rt software were detected and information about stopping the traffic was sent to the ONOS SDB controller. Source IP, destination IP, packet size information from the data received with the written script were recorded in the database with time data. With the Sflow protocol, the header information given in Tab. 4 can be obtained.

Table 4 Extracted header list for TCP, UDP and ICMP packets [21]

TCP	UDP	ICMP
Src-IP	Src-IP	Src-IP
Dst-IP	Dst-IP	Dst-IP
Src-Port	Src-Port	ICMP-code
Dst-Port	Dst-Port	ICMP-Type
Protocol-Type	Protocol-Type	protocol-Type
Packet-Length	Packet-Length	Packet-Length
TTL	TTL	TTL
SYN-ACK	Length	DF Flag
Window	Checksum	Timestamps

The collected data were processed and counted how many times each destination IP address was used as the destination IP, and during this process, the packet sizes for the relevant destination IP address were collected and recorded. An example image of the recorded data is given in Table 5.While data were being collected with the Sflow protocol, DDoS attacks that may occur on the network were also detected and the attacker IP addresses were automatically blocked for a specified period of time. IP addresses whose block has expired were allowed to generate traffic again.

Table 5 Example view of the data

Date	DstIP	Count	Total Packet Size
02/09/2022	13.10	1677	2496428
02/09/2022	172.21	1610	2041244
02/09/2022	212.15	1517	171661
02/09/2022	13.10	1077	1595790
02/09/2022	88.234	610	886912
02/09/2022	159.14	448	43267
02/09/2022	212.15	405	42172
02/09/2022	88.234	393	479908
02/09/2022	193.14	382	33989
02/09/2022	34.247	359	25356

3.3 Clustering Data with K-Means Algorithm

K-means algorithm is one of the most known and used methods among clustering methods [22]. Clustering algorithms are useful tools for clustering and analysis of network traffic usage, data mining, compression, probability density estimation [23].

With K-means, it will recursively assign data points to one of its determined clusters, depending on how close the point is to the cluster center. With the K-means algorithm, it is aimed to determine the number of K cluster centroids and data points classified as clusters.

Assuming we have $x_1, x_2, x_3, \ldots, x_n$ data points and K required number of clusters, basically the procedure is followed as follows.

- The first centers from the dataset are randomly chosen as K points or the first K points.
- The Euclidean distance of each point in the dataset with the determined K cluster centers is found.
- Each data point is assigned to its nearest center point using the distance found in the previous step.
- The new center of gravity is found by averaging the points in each cluster group.
- Reassignment to the group is repeated until the centers do not change or by finding the distance for a fixed number of iterations.

The relationship between the two values in the dataset is calculated over the Euclidean distance and the distance between the two points is calculated as shown in Eq. (1).

$$d(p,q)=\sqrt{(q_1-p_1)^2+(q_2-p_2)^2} \tag{1}$$

If $p = (p_1, p_2)$ and $q = (q_1, q_2)$ the distance is given as:

[Pyhton Code]

```
def euclidean_distance(point1, point2):
return math.sqrt((point1[0]-point2[0])^2 + (point1[1]-point2[1])^2)
```

The number of clusters to be created in the K-means algorithm is given to the algorithm as a parameter. How many clusters the data should be divided into was determined using the elbow method. In Fig. 4, elbow method graphs created according to daily, weekly and monthly data are given.

Based on the daily, weekly and monthly graphs created by the elbow method, it was determined that the most suitable number of clusters for the k-means algorithm was '3'. After determining the number of clusters, the process of assigning each data to the nearest cluster is started. If each cluster center is denoted by ci, each x data point is assigned to a cluster based on Eq. (2). Here dist() is the Euclidean distance.

$$\text{argmindist}(c_i, x)^2 c_i \in c \tag{2}$$

Eq. (3) is applied to find the new center of the clustered data from the point group. S_i is the set of all points assigned to the I set.

$$c_i = \frac{1}{S_i} \sum_{x_i \in S_i} e_i \tag{3}$$

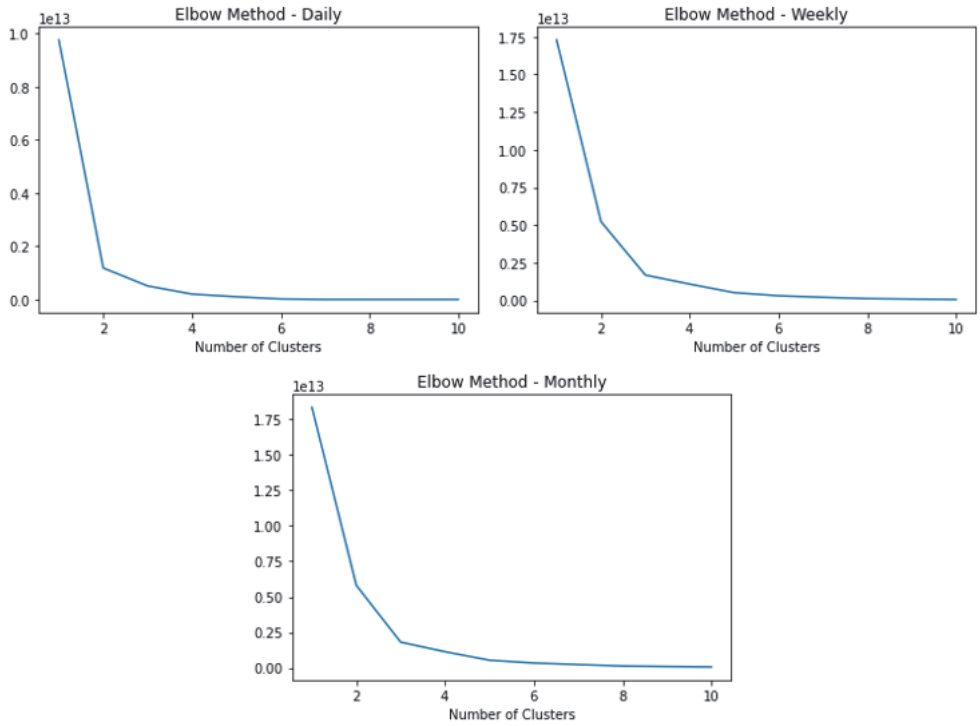

Figure 4 Elbow method graphs

The collected and separated data were divided into 3 different clusters with the K-means algorithm. Clusters created with the K-means algorithm are given in Fig. 5.

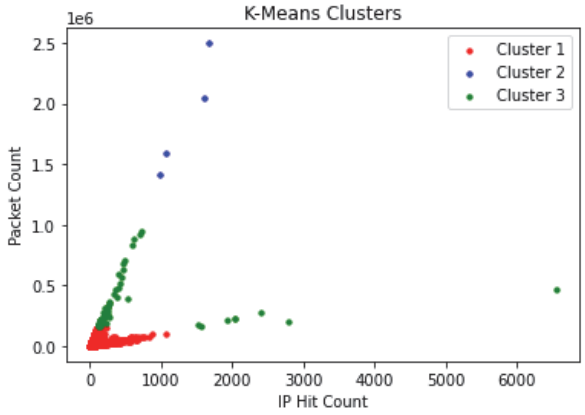

Figure 5 K-Means Clusters

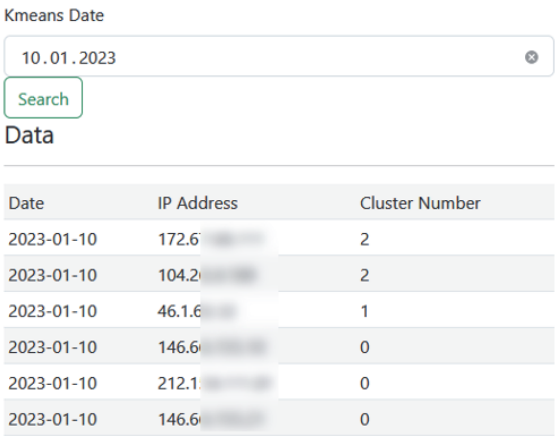

Figure 6 Data search interface created with PHP software language

Clustering is not done only on historical fixed data. After the data collected from the network switching devices during the day are recorded in the database, the traffic data of the last month are automatically retrieved with the script files created and re-clustered and stored in the database. A web page was created using php software language to see the cluster data of the past days. An example of a searched table is given in the interface created in Fig. 6.

Clustering is done automatically with K-means. However, the network administrator may want to give priority to some IP addresses on a planned basis on certain days in the system at his own request. In order to facilitate the work of the network administrator, an interface has been designed that makes it possible to enter records that will be active on the date entered into the system. The interface used to add data is given in Fig. 7.

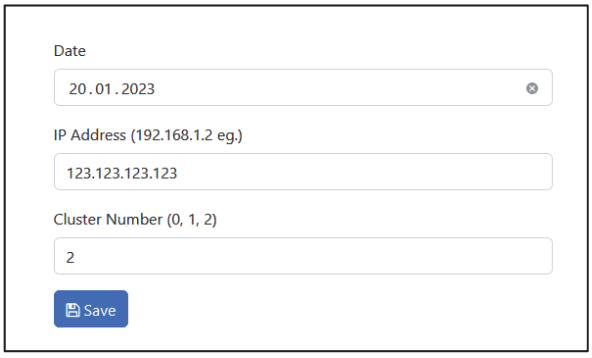

Figure 7 Data insertion interface

3.4 Processing of Clustered Data in Onos

IP addresses in clusters divided by K-means are assigned to pre-created Qos queues in network switching devices, taking into account the end link speed. The user port speed of the generally used switching devices is taken as 1 Gbps. The first queue bandwidth is 500 Mbps, the second queue bandwidth is 350 Mbps, and the third queue

bandwidth is 150 Mbps. The bandwidth controller required for the queues to work is limited as in Eq. (4). Here, *qdata* is the data controller for limiting, and *q*150, *q*300, *q*500 are the queues created.

$$\sum_{i0}^{ik} fx = \begin{cases} q_{500}, qdata \geq 500 \\ q_{300}, qdata \geq 150 \ \& qdata < 500\, mbps \\ q_{150}, qdata \leq 150\, mbps \end{cases} \quad (4)$$

For the IP address cluster with the highest total packet size and hit count among the data on which the K-Means algorithm is applied, a flow record is entered to the ONOS controller to assign these traffics to the first queue with 500 Mbps bandwidth. IP addresses in the cluster with the lowest total packet size and hit count are assigned to the queue with 150 Mbps bandwidth. For the remaining cluster, a queue with 300 Mbps bandwidth isused.

OnosSDN controller defaults to 'forward' module priority value is "10". The priority value for the entered flow records is selected to be more than "10". Fig. 8 shows examples of logged flow records for each queue. These records are automatically entered into the system by being cleaned and recalculated every night with scripts written in python programming language.

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	0	50000	0	IPV4_SRC:34.24	imm[QUEUE{queueId=345, port=3}], cleared:false	enes
Added	0	0	50000	0	IPV4_SRC:13.1	imm[QUEUE{queueId=123, port=3}], cleared:false	enes
Added	0	0	50000	0	IPV4_SRC:172.2	imm[QUEUE{queueId=234, port=3}], cleared:false	enes

Figure 8 Example of Onos controller flow records

It is possible to monitor the flow records of the devices connected to it in the Onossdn controller. In the study, an interface was created to see the Flow records without connecting to the Onos controller. In this way, data can be tracked from a single point without logging into more than one application. In Fig. 9, the interface designed to see the flow records of the Onos controller is shown.

ONOS Flows

LastSeen	Device	Packets	Bytes	Destination IP	Queue Name
2023-01-27 16:20:55	of:0000000000000001	0	0	195.87.	123
2023-01-27 16:20:55	of:0000000000000001	0	0	172.21	123
2023-01-27 16:20:55	of:0000000000000001	0	0	216.58.	345
2023-01-27 16:20:55	of:0000000000000001	0	0	50.7.23	123
2023-01-27 16:20:55	of:0000000000000001	0	0	195.17	123
2023-01-27 16:20:55	of:0000000000000001	0	0	46.20.3	123
2023-01-27 16:20:55	of:0000000000000001	0	0	138.19	123
2023-01-27 16:20:55	of:0000000000000001	0	0	142.25	123
2023-01-27 16:20:55	of:0000000000000001	0	0	8.238.1	123

Figure 9 Flow records of Onos controller

4 RESEARCH FINDINGS

Speed tests were carried out with the iperf application on the virtual network created on mininet to test the applied flow records and queues. In the tests, queues with bandwidths of *q*500 for the h1 host computer, *q*350 for the *h*2 host computer, and *q*150 for the *h*3 host computer were selected.

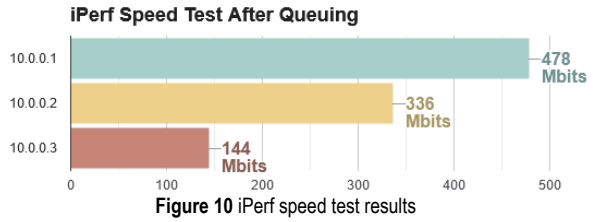

Figure 10 iPerf speed test results

In Fig. 10, the iperf speed test results from the *h*4 computer to the *h*3, *h*2, and h1 host computers are given, respectively.

According to the speed test results, the queue selection process was performed according to the flow records entered into the ONOS SDN controller, and the bandwidth selection process was carried out correctly depending on the different queues for the IP addresses of different clusters.

In order to test the DDoS attacks that may occur on the network, the intruder IP addresses are blocked with the codes added to the sflow application by continuously sending packets to the 5001 port with iperf on the virtual network. With the Sflow application, different threshold values can be set against different DDoS attacks. In this study, 2 Mbit/s is set as the threshold value for TCP reflection attacks. Packets exceeding this threshold value are automatically blocked. In Fig. 11, a screenshot of a blocked traffic and the flow record on the Onos controller are given.

FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
50000	0	IN_PORT:1, IPV4_SRC:10.0.0.1/32, TCP_DST:5001	imm[NOACTION], cleared:false	ddos

```
2022-09-29T18:15:41+03:00 INFO: blocking 10.0.0.1,5001
2022-09-29T18:15:55+03:00 INFO: unblocking 10.0.0.1,5001
```

Figure 11 A screenshot of DDoS attack blocked traffic

It is a difficult process for the network administrator to constantly monitor the sflow screen. For this reason, a web interface has been designed in order to see the IP addresses that have carried out previous DDoS attacks. A section of the designed web interface is given in Fig. 12.

DDoS Protect List (Blocked IPs and Ports)

Date Time	Blocked IP	Blocked Port
2023-01-24 12:52:31	66.	5001
2023-01-24 12:50:05	25.	55018
2023-01-24 12:16:27	194.	59362

Figure 12 DDoS web interface

5 CONCLUSION

The rapid development of internet technologies and the use of the internet in all areas have revealed the need for increased bandwidth. Software-defined networks are currently a research area used for network management. The bandwidth needs of users can be adjusted on network devices with Qos.

In this study, internet traffic created by users on the software-defined network was collected and a possible DDoS attack was detected. Different clusters were created by processing the collected traffic data with the K-means algorithm. IP addresses in the created clusters were assigned to different Qos defined in network switching devices via Onos software-defined network controller. In this way, users were provided with higher bandwidth to connect to the most frequently visited IP addresses on the internet. While performing all these operations, instead of accessing and processing the applications used separately, tasks were defined for regular operations, and interfaces were created for tracking, examining, and viewing data. In this way, the system administrator was able to monitor the internet network from a single point. In future work, we will focus on the classification of attacks detected in the network system where the proposed model is installed and the application of attack-specific defence methods. In addition to this goal, it is aimed to achieve higher successful results by using the hierarchical clustering algorithm and KMeans algorithm together.

6 REFERENCES

[1] Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). A Survey on Software-Defined Networking. *IEEE Communications Surveys and Tutorials*, *17*(1), 27-51. https://doi.org/10.1109/COMST.2014.2330903

[2] Süzen, A. A. (2021). Developing a multi-level intrusion detection system using hybrid-DBN. *Journal of Ambient Intelligence and Humanized Computing*, *12*(2), 1913-1923. https://doi.org/10.1007/S12652-020-02271-W/TABLES/8

[3] Singh Rana, D., Kumar Chamoli, S., & Ashish Dhondiyal, S. (2019). Software Defined Networking (SDN) Challenges, issues and Solution Network Security View project Sleeping Mode MODLEACH Protocol for WSN View project Software Defined Networking (SDN) Challenges, issues and Solution. *International Journal of Computer Sciences and Engineering Open Access Research Paper*, *7*. https://doi.org/10.26438/ijcse/v7i1.884889

[4] Čisar, P., Erlenvajn, D., &Maravić Čisar, S. (2018). Implementation of Software-Defined Networks Using Open-Source Environment. *Tehnički Vjesnik*, *25*(Supplement 1), 222-230. https://doi.org/10.17559/TV-20160928094756

[5] Hikmat Haji, S., M Zeebaree, S. R., Yousif Ameen, S., Shukur, H., Haji, S. H., Hasan Saeed, R., Ameen, S. Y., Shukur, H. M., Omar, N., MSadeeq, M. A., Salih Ageed, Z., Mahmood Ibrahim, I., & Maseeh Yasin, H. (2021). Comparison of Software Defined Networking with Traditional Networking. *Asian Journal of Research in Computer Science*, *9*(2), 1-18. https://doi.org/10.9734/AJRCOS/2021/v9i230216

[6] Imran, M., Durad, M. H., Khan, F. A., & Derhab, A. (2019). Toward an optimal solution against Denial of Service attacks in Software Defined Networks. *Future Generation Computer Systems*, *92*, 444-453. https://doi.org/10.1016/J.FUTURE.2018.09.022

[7] Süzen, A. A., Şimşek, M. A., Kayaalp, K., & Gürfidan, R. (2019). The Attack Methodology to Wireless Domains of Things in Industry 4.0. Nevşehir Bilimve Teknoloji Dergisi, 8, 143-151. https://doi.org/10.17100/NEVBILTEK.557886

[8] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for Network Intrusion Detection in Software Defined Networking. *Proceedings - 2016 International Conference on Wireless Networks and Mobile Communications, WINCOM 2016: Green Communications and Networking*, 258-263. https://doi.org/10.1109/WINCOM.2016.7777224

[9] Cicioğlu, M. & Çalhan, A. (2017). Yazılım Tanımlı Ağlar - YTA. *Karaelmas Science and Engineering Journal*, *7*(2), 684-695.

[10] Fosić, I. & Žagar, D. (2021). Security Features in a Hybrid Software-Defined Network. *Tehnički Vjesnik*, *28*(4), 1371-1379. https://doi.org/10.17559/TV-20201208122622

[11] Mantas, A. & Ramos, F. M. V. (2019). Rama: Controller Fault Tolerance in Software-Defined Networking Made Practical. https://doi.org/10.48550/arxiv.1902.01669

[12] Karakus, M. & Durresi, A. (2017). Quality of Service (QoS) in Software Defined Networking (SDN): A survey. *Journal of Network and Computer Applications*, *80*, 200-218. https://doi.org/10.1016/J.JNCA.2016.12.019

[13] Krile, S., Žagar, D., & Martinović, G. (2009). Better bandwidth utilization of multiple link capacities with mutual traffic correlation. *Tehnički Vjesnik*, *16*(4), 11-18.

[14] Onos. (2020). Onos Project. https://wiki.onosproject.org3

[15] Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016). SDN controllers: A comparative study. *Proceedings of the 18th Mediterranean Electrotechnical Conference: Intelligent and Efficient Technologies and Services for the Citizen, MELECON 2016.* https://doi.org/10.1109/MELCON.2016.7495430

[16] Mininet. (2020). Mininet. http://mininet.org

[17] Hesselbach Serra, X. (2019). Implementacióbàsicai proves de funcionament de la plataforma ONOS.

[18] Li, B., Springer, J., Bebis, G., & Hadi Gunes, M. (2013). A survey of network flow applications. *Journal of Network and Computer Applications*, *36*(2), 567-581. https://doi.org/10.1016/J.JNCA.2012.12.020

[19] SFlow-RT. (2021). sFlow-RT. https://sflow-rt.com/

[20] sFlow. (2009). sFlow sampling rates. https://blog.sflow.com/2009/06/sampling-rates.html

[21] Ujjan, R. M. A., Pervez, Z., Dahal, K., Bashir, A. K., Mumtaz, R., & González, J. (2020). Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN. *Future Generation Computer Systems*, *111*, 763-779. https://doi.org/10.1016/J.FUTURE.2019.10.015

[22] Sinaga, K. P. & Yang, M. S. (2020). Unsupervised K-means clustering algorithm. *IEEE Access*, *8*, 80716-80727. https://doi.org/10.1109/ACCESS.2020.2988796

[23] Hamerly, G. & Elkan, C. (2003). Learning the k in k-means. *Advances in Neural Information Processing Systems*, *16*.

Contact information:

Abdülkadir ÇAKIR, Professor
Isparta University of Applied Sciences, Faculty of Technology,
Electrical and Electronics Engineering, Turkey
E-mail: abdulkadircakir@isparta.edu.tr

Enes AÇIKGÖZOĞLU, Doctorate
(Corresponding author)
Isparta University of Applied Sciences, Keçiborlu Vocational School,
Computer Technologies, Turkey
E-mail: enesacikgozoglu@isparta.edu.tr