RESEARCH ARTICLE OPEN ACCESS

Providing a Dynamic Model to Achieve Quality of Service in Software Defined Networking

Afraa Mohammad [1], Ahmad Saker Ahmad [2]
Department of System and Computer Networks Engineering Tishreen University
Latakia – Syria

ABSTRACT
Software Defined Networking (SDN) is considered one of the most common techniques in the world of networks. It is characterized by flexibility and ease in network management and the ability of continuous development because it separates the control layer from the data layer. The controller is the most important component in SDN. It is located in the control layer, and it is responsible for the programming and management of the network. One of the important issues that must be worked on in the world of networks is the Quality of Service (QoS), which ensures the best performance in data routing. QoS specially handles some data and routes it in the best path. Providing the necessary bandwidth for data transmission plays the biggest role in achieving the requirements of QoS. It guarantees that no congestion or loss of some packets occurs. In this paper, we introduce a dynamic mechanism that classifies packets into several classes according to priorities and then searches for the best path that provides the necessary bandwidth and low packet loss rate. This mechanism ensures QoS requirements. In achieving this mechanism, we relied on the concepts of differential services and traffic engineering. We implemented it using the RYU controller. The results showed that the proposal provides an effective mechanism in achieving QoS in SDN networks.

Keywords: — Software Defined Networking, Quality of Service, Priority, Traffic Engineering, RYU controller.

I. INTRODUCTION

Software Defined Networking (SDN) architecture consists of three layers (application layer, control layer, and data layer), through this architecture the control layer is separated from the data layer, unlike traditional networks. This separation facilitates the ability to expand the network and add new elements with high flexibility. The controller is the most important component of SDN. It is located in the control layer. It has a global view of all network topologies. Through it, we can program the applications that we want. The controller is responsible for all routing decisions in the network, and the function of infrastructure devices is limited to implementing the decisions of the controller. It communicates with the data layer through the OpenFlow protocol, which organizes the communication between the controller and data layer devices [1]. SDN has a high flexibility in developing the network and achieving the best performance, including achieving QoS, which is considered an important research field. The importance of QoS stands out with the increased use of multimedia applications that are sensitive to the requirements of QoS.

QoS organizes and controls the bandwidth to determine which flows must pass the network first. To achieve QoS, the following criteria must be considered: bandwidth (BW), delay, jitter, and packet loss rate [2]. SDN has provided solutions to many problems facing networks including QoS by providing the appropriate bandwidth [3]. Bandwidth plays a primary role in achieving QoS and improves network performance in general [4].

II. THE IMPORTANCE OF RESEARCH AND ITS OBJECTIVES

The importance of the research comes through introducing a new mechanism in achieving QoS in SDN by using the concept of differential services (DiffServ) and traffic engineering (TE) to choose the most appropriate path that has the best available bandwidth and lowest packet loss rate according to a dynamic method by taking advantage of controller features and OpenFlow protocol.

This research aims to achieve QoS in the best way that improves the process of forwarding packets in the most appropriate path according to the nature of the data, the state of the network, and the available paths.

III. RESEARCH METHODOLOGY

The research was done by using Mininet simulator which is commonly used in building SDN networks, in addition to using Miniedit tool to build the network topology. We implemented the applications using the RYU controller, which is an open-source controller programmed in the Python language. We also used Iperf tool to measure the bandwidth and jitter.

IV. OPENFLOW PROTOCOL

OpenFlow is considered the most important protocol in SDN. It organizes the communication between the controller and the data layer devices. The basic idea of the operation of OpenFlow is shown in Fig.1 [5]. When a new flow reaches the switch, it is matched with the set of rules or entries in the flow table. Then this flow is handled according to the following cases:

• In the case of matching, the packet is handled according to the existing rule and then the data in the statistics column is modified.

• In the absence of a match, this packet will be forwarded completely or its header to the controller to take the appropriate decision.

• After the decision has been made, the controller sends it to the switch and updates its flow table.

• This mechanism is done by exchanging several messages between the controller and the switch.

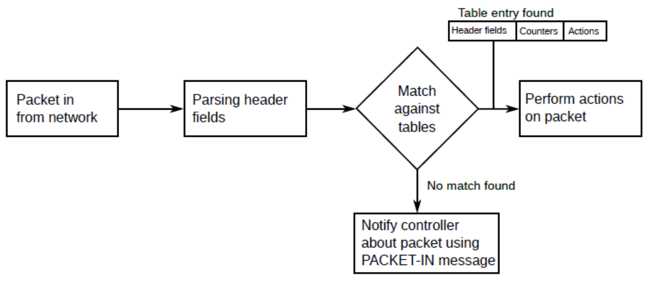

Fig. 1 OpenFlow mechanism

V. OPENFLOW MESSAGES

The communication between the controller and the switch is established by exchanging several messages, the most important are [6],[7]:

1- OFPT_HELLO: It is a welcome message when the communication between the controller and the switch is established, and it is exchanged in both directions. It includes the protocol version (OpenFlow protocol version).

2- OFPT_ECHO_REQUEST: It is exchanged in both directions to check the validity of the connection.

3- OFPT_FEATURES_REQUEST: It is sent from the controller to the switch and through it, the controller inquires about the state of the switch and its features.

4- OFPT_FEATURES_REPLY: It represents the response from the switch to the controller message (OFPT_FEATURES_REPLY).

After the communication between the switch and the controller is established, several messages are sent to organize flows and deal with them. The most important of these messages are [8],[9]:

1- OFPT_PACKET_IN: If a specific packet is received on one of the switch ports, but there is no matching rule for this flow, then the switch will send this packet to the controller to make the appropriate decision to deal with it.

2- ADD-MODIFY-DELETE: It represents (add-modify-delete) for the rules within the flow table. It is sent from the controller to the switch.

3- STATE_REQUEST: The controller sends it to the switch to know the status of the switch ports. The switch responses with the message STATE_REPLY.

4- BARRIER_REQUEST: The controller sends it to the switch to ensure that the switch has handled all requests received before this message.

5- OFPT_ERROR: It is sent by the switch in case something goes wrong so the controller knows about it.

6- SET_CONFIG: The controller sends it to the switch to confirm the expiration date of the stored rules in the flow table.

VI. QUALITY OF SERVICE IN SDN

Quality of Service (QoS) is one of the basic concepts that characterizes the performance of any technology. It shows how some data is specially handled, such as multimedia applications that are assigned a high priority in data transmission.

To achieve QoS, attention must be paid to the following criteria: delay, bandwidth, packet loss rate, and jitter.

Quality of Service is achieved through several techniques, including giving traffic a specific pattern so that the bandwidth does not exceed a predetermined value, and also includes the Resource Reservation Protocol (RSVP), Service Level Agreement (SLA), Integrated Services (IntServ) and Differentiated Services (DiffServ) [10].

Each of these techniques has its way to achieve QoS most properly, according to the state of the network.

A. QoS models

Packets are handled according to three models [11]:

1- Best Effort: In this case, packets are not classified, and they have the same priority and there is no special treatment for any type of data. This model does not achieve the basic requirements of QoS.

2- Integrated Services (IntServ): In this model, resources are reserved in advance for a specific flow and these resources remain dedicated to this flow along the path without regard to the type of packets.

3- Differentiated Services (DiffServ): It is considered one of the most important models of QoS. Flows are categorized into several classes according to what the flows need from bandwidth, delay, etc., and prioritize the classes over the other. That is, it provides a better service for some packets than the service provided for other packets. For example, multimedia applications are given high priority to other applications. In this model, we do not need to previously reserve resources.

Internet Engineering Task Force (IETF) has defined three types of QoS [12]:

- Best Effort (BE).
- Assured Forwarding (AF).
- Expedited Forwarding (EF).

B. Assured forwarding (AF)

It defines 12 classes of service. In this model, the marking of packets is done according to their classes to achieve QoS on this type. For this purpose, there is a field which is a type of service (ToS) in IP header with a length of 8 bits.

Six of these eight bits are used to define the class of service and represent the length of the differentiated service code point (DSCP) field and the remaining bits are used for coding purposes [13].

AF defines two concepts:
1- Queuing System: It includes four separate queues, each one has its resources.
2- Drop Priority: Within each of the previous four queues, there are three probabilities (low, medium, and high) related to the drop of packets. When congestion occurs, the packets with the highest drop priority will be dropped. Thus the previous two concepts define 12 classes of service.
AF values are represented as AF_{xy} where x refers to the queue and y refers to the drop priority.
For example, if there are packets with values of AF11, AF12, AF 13, that means they have the same value of x and thus all of them will go in the same queue.
While packets with values of AF 21, AF22, AF23 will go in another queue which is different from the previous queue.
Within each queue, i.e. the same x and different values for y like (AF21, AF22, AF23) that means they have different drop priority values, so each packet will be handled in a different way to avoid congestion [13].

C. *Expedited forwarding (EF)*

It provides forwarding of packets that are highly sensitive to delay, jitter, and packet loss rate, so it is especially used in transmission VOIP packets because they require less delay, less jitter, and fewer loss rates. It has one value for DSCP which is DSCP = $(101110)_2 = (46)_{10}$ [14].

VII. TRAFFIC ENGINEERING (TE)

Traffic Engineering (TE) is an important network application that provides packet measurement and management in the network. It selects the best routing paths according to the requirements of QoS. TE includes traffic measurement and traffic management.
Traffic measurement is responsible for monitoring and analyzing traffic, including[15],[16]:
- Network topology parameters: They represent the number of nodes in the network and how they are distributed, bandwidth, port statistics, and other information related to the network topology.
- Network traffic parameters: They include the number of packets that pass through the network through a specific port.
- Network performance parameters: They include delay, available bandwidth, packet loss rate, and throughput.

While traffic management includes different applications such as QoS, load balancing, energy-saving, and other applications. Depending on the state of the network, and through the information gathered from TE, and by analyzing the flow statistics, we can predict the incoming packets later and thus avoid network congestion, which improves the effectiveness of the network.
SDN provides flexibility in the implementation of TE within it due to:
- The controller and its ability to monitor the network topology continuously, and follow the changes that occur in it, through the Link Layer Discovery Protocol (LLDP) [17].
- In general, there are two types of packets in SDN:
  - Data traffic packets: They represent the exchanged data between switches.
  - Control traffic packets: They represent the exchanged data between the controller and switch through the OpenFlow protocol.

VIII. DEFAULT ROUTING ALGORITHM IN SDN

The default routing algorithm in SDN is the shortest path routing algorithm (Dijkstra). The path with the least number of hops between nodes is chosen [18]. This algorithm is not concerned with other requirements for choosing a path, such as the available bandwidth between the nodes, the delay that may occur in the network, and whether the chosen path is busy or not. So we compare our proposal with a default algorithm to measure its effectiveness in achieving QoS.

IX. MECHANISM OF THE PROPOSAL

Our proposal depends on implementing of DiffServ and TE to select the best path that guarantees QoS requirements. The steps of our mechanism are:
1- We achieve the concept of differential services by creating queues and assigning them to each class of packets to be transferred in the network. This is done by defining classes and giving each class a different priority from the other classes. When a packet arrives at the controller (the switch sends it to the controller), it detects the class of these packets and selects the packet that has a high priority in transmission, and places it in the appropriate queue.
Through this step, we will be able to choose the packets that must be sent first because they have a high priority over the rest of the packets, and put each packet from a queue to be ready for transmission.
2- Traffic engineering is applied through monitoring the network and analyzing the flows. It is considered one of the important steps that help in choosing the best path to forward packets. It is necessary to collect the statistics for each of the switch ports which are used in transmission and reception, in addition to the statistics of each flow that is sent in the network.
To do this, we implement a special application to monitor the network using the RYU controller. This application must also check the state of connection between switches in the network, and this is done through the use of a special event to monitor the state of switches.
Switch monitoring application is implemented to obtain statistics periodically in the network, ensuring that changes that may occur in the network are monitored.
The monitoring application sends OFPFLOWStatsRequest to the switches to get the statistics of the flows, and the response arrives according to the OFPFLOWStatsReply message, and to get the port statistics, OFPPORTStatsRequest is sent, and the response arrives through OFPPORTStatsReply message. By implementing this application, we get all of these stats:
(rx_pkts: number of received packets, tx_pkts: number of transmitted packets, rx_Bytes: number of received bytes, tx_Bytes: number of transmitted bytes, tx_error: transmission

errors, rx_error: errors of receive). We will use these stats to calculate the available bandwidth and packet loss rate.

3- After selecting the packet to be sent and within any queue, it is necessary to specify which path achieves QoS, and for this, we take the following steps:

a) Each link in the network has a predefined bandwidth (BW). When a packet is sent on this path and by monitoring the network, we can know the load on this link through:

load= rx_pkts + tx_pkts (1)

That means the load on the link is equal to the number of received and transmitted packets through this link.

b) We define the available bandwidth on this link which is (BW2) through:

BW2= default BW - load (2)

c) After calculating the available bandwidth for each link, we decide to choose the most appropriate path that achieves the best consumption for the bandwidth and provides the transmission of packets without congestion or delay. This is done by comparing the available bandwidth with the packet requirements of the needed bandwidth for transmission. The available bandwidth must be equal or greater to the required bandwidth.

d) To make the final decision about transmission, we must know the packet loss rate on this path between the source and destination which is calculated as:

Packet Loss Rate(S,D)= (tx_pkts - rs_pkts)/tx_pkt (3)

Where S refers to source and D refers to destination. Through (3), we can choose the path that achieves the lowest packet loss rate. From steps (c) and (d), we make the decision to choose the most appropriate path for transmission, which has the best available bandwidth and the lowest packet loss rate.

e) If this path does not provide the requirements, a new path must be chosen, so the statistics are recalculated again, and then each of the equations (1,2,3) is recalculated until we find the best path.

4- This mechanism is implemented in the controller according to the following steps:

a) When a packet reaches the switch and there is no rule for it, it is forwarded to the controller by using the OFPT_PACKET_IN message.

b) The controller checks the packet and determines the class by the existing DSCP value, i.e. the packet classification and puts it in the appropriate queue (which was previously defined).

c) Depending on the chosen path according to both the best available bandwidth and the lowest packet loss rate, and taking into account the type of service to, the proposed algorithm selects the best path to forward the packets

d)The controller sends a FLOW_MODIFY or FLOW_ADD message to modify or add a new row to the flow table for each switch along the chosen path that includes the flow properties and the required action.

e) Packets are sent using a PACKET_OUT message to the switch to be forwarded in the network.

f) The proposal is compared to the default routing mechanism in SDN which is the shortest path routing algorithm (Dijkstra).

g) This proposal should be implemented with every new packet to be forwarded in the network.

X. Experiment and Results

We used Mininet simulator, and we built a network using the Miniedit tool. Our network consists of RYU controller, 4 switches (S1, S2, S3, S4), the bandwidth for each link between the switches is 25 Mbps, and 10 hosts (h1, h2, h3, h4, h5, h6, h7, h8, h9, h10) which are distributed according to the topology as shown in Fig. 2.

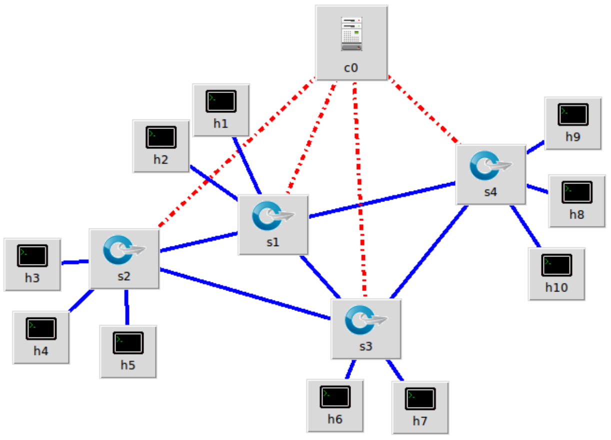

Fig. 2 Network topology

We also used iperf tool to measure bandwidth and jitter. To evaluate our proposal, we want to transfer 3 different classes of packets (EF, AF, BE) from h3 to h9 according to the parameters as shown in Table I.

TABLE I
SCENARIO PARAMETERS

Period of transmission	Size	Class	DSCP value	Source	Destination
10 sec	24 Mbps	EF	46	h3	h9
10 sec	5 Mbps	AF	26	h3	h9
10 sec	1 Mbps	BE	0	h3	h9

We used three classes of packets: The first class is expedited forwarding (EF) which is the most sensitive to delay and requires a specific bandwidth. It has a value (DSCP= 46). We want to transfer 24Mbps of this class. The second class is assured forwarding (AF) whose data is also sensitive to delay and bandwidth, and it has a value (DSCP=26). We want to transfer 5 Mbps of this class. The last class is (BE) which is best effort model and does not require any special treatment. It has a value (DSCP= 0). We want to transfer 1Mbps of this class.

According to our proposal, we must first define three queues for the three classes of packets according to the specifications as shown in Table II. For each queue, we define its ID, as well as the minimum and maximum rates, and we assign each class to a queue.

TABLE II
QUEUE SPECIFICATIONS

Queue ID	Class	Max rate	Min rate
Queue 1	EF	25 Mbps	23 Mbps
Queue 2	AF	25 Mbps	5 Mbps
Queue 3	BE	25 Mbps	1 Mbps

A. Measuring bandwidth for EF packets

We want to transfer 24 Mbps of EF packets according to the scenario parameters as shown in Table I. We obtained the results as shown in Fig. 3.

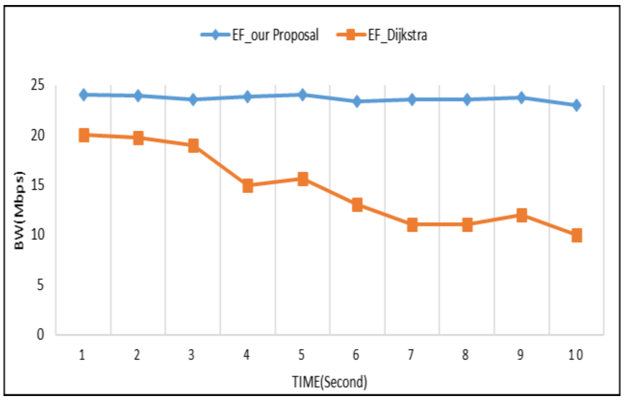

Fig. 3 Bandwidth consumption for EF packets

Fig. 3 shows that the value of the bandwidth consumption for EF packets with the implementation of our proposal maintained very close values of 24Mbps and did not decrease from the value of 23Mbps. That means they remained within the defined values for queue 1 as in Table II.
While with the Dijkstra algorithm it reached 20 Mbps and then it started to decrease to 10 Mbps.
The reason for this is when the marked packet with EF reaches the controller, it detects its class and places it in the appropriate queue for it which is queue1. EF packets take priority over the rest of the other packets because DSCP = 46. According to our proposal, the path (s2-s1-s4), which is suitable for transmission of the EF packet, whose size according to Table I is 24 Mbps. The same path will also be chosen by Dijkstra except that it suffers from congestion due to the presence of other packets to be transferred. This explains that the value of BW when using Dijkstra was low, and continued to decrease during the transmission period 10sec.

B. Measuring bandwidth for AF packets

We obtained the results as shown in Fig. 4 during transmission 5 Mbps of AF packets according to the scenario parameters as shown in Table I.

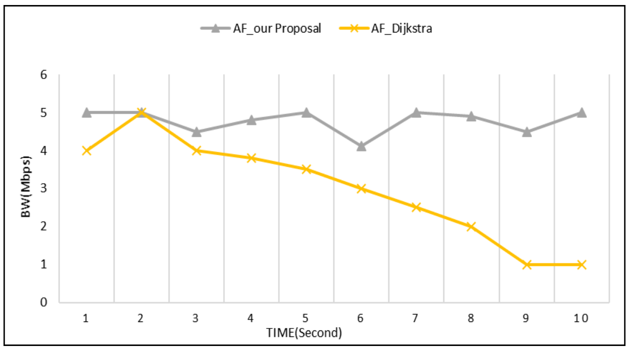

Fig. 4 Bandwidth consumption for AF packets

Fig. 4 shows that the value of the bandwidth for AF, with the implementation of our proposal, maintained very close to 5Mbps and did not decrease from this value. That means they remained within the defined values for queue 2 as in Table II. While with Dijkstra algorithm, it reached 5 Mbps, and then it started to decrease to 1 Mbps. The reason for this is that the AF packets whose DSCP= 26 have a lower priority than EF so they are next in the transmission process and are placed in queue 2. According to Dijkstra algorithm, the path (s2-s1-s4) will be chosen to transmit packets from h3 to h9, but this path suffers from congestion due to the previously transported EF packets which mean that there is congestion on this path.
When implementing our proposal, AF packets will go to another path (s2-s3-s4), because it achieves a good and sufficient bandwidth, and also the rate of discarded packets is currently equal to zero because there is no previous traffic on this path. Thus our proposal chooses (s2-s3-s4), which effectively improves bandwidth consumption.

C. Measuring bandwidth for BE packets

We obtained the results as shown in Fig. 5 during the transmission of 1 Mbps of BE packets according to the scenario parameters shown in Table I.

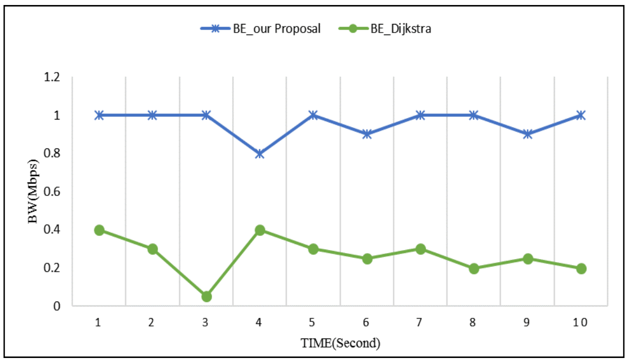

Fig. 5 Bandwidth consumption for BE packets

Fig. 5 shows that the value of the bandwidth for BE packets, with the implementation of our proposal, maintained very close of 1 Mbps and did not decrease from this value, i.e. they remained within the defined values for queue 3 as in Table II. While with Dijkstra algorithm its greatest value was 0.4 Mbps and then it started decreasing to 0.2 Mbps. The reason for this is that the BE packets are considered to be the lowest priority and are placed in the BE queue. Therefore, the controller chooses them after sending EF and AF packets.
According to Table I, the size of this packet is 1Mbps and the appropriate path must be chosen for it. We find that our proposal chooses the path (s2-s3-s4) because it guarantees sufficient bandwidth for it, and achieves a small packet loss rate according to the statistics that the switch sends to the controller via OpenFlow protocol.
Depending on Dijkstra algorithm, the path (s2-s1-s4) will be chosen to forward packets from h3 to h9, but this path suffers from congestion due to the transmission of previously AF and EF packets which means that there is congestion on this path.

D. Comparison of bandwidth consumption between the three classes (EF, AF, BE)

Fig. 6 shows the results of comparing the bandwidth consumption when transmitting packets of the three classes (EF,AF,BE). We find from Fig. 6 that the value of the consumed bandwidth for each of the classes is almost constant when using our proposal where the value of the bandwidth of the EF packets is approximately 24 Mbps during the transmission time of 10 sec, and the value of the bandwidth of the AF packets is approximately 5 Mbps during the transmission period 10 sec, the bandwidth value for BE packets is approximately 1 Mbps over 10 sec.

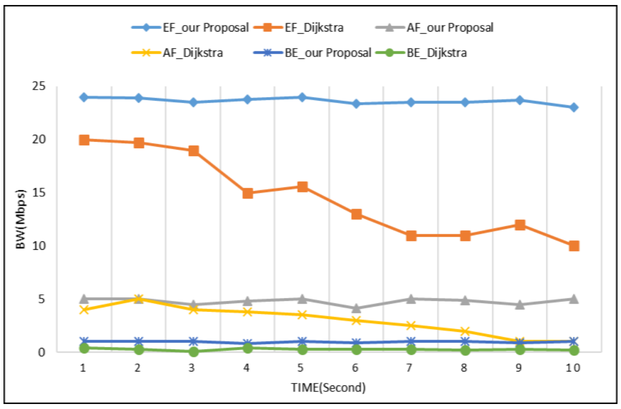

Fig. 6 Bandwidth consumption for (EF,AF,BE) packets

We notice from these results that our proposal achieved the best use of bandwidth because it finds the best routing path according to the requirements of each class, after calculating the available bandwidth and the packet loss rate for each available path. While with Dijkstra algorithm and without the use of the concept of priorities, we find that all packets will be sent in the same path and without priority in transmission, so the best use of the bandwidth is not used because it directs the three packets in the same path, which causes network congestion. Therefore, the value of the consumed bandwidth for each class will decrease with increasing transmission period as shown in Fig. 6.

E. Measuring jitter for EF packets

Fig. 7 shows the results of the jitter comparison when transmission EF packets.

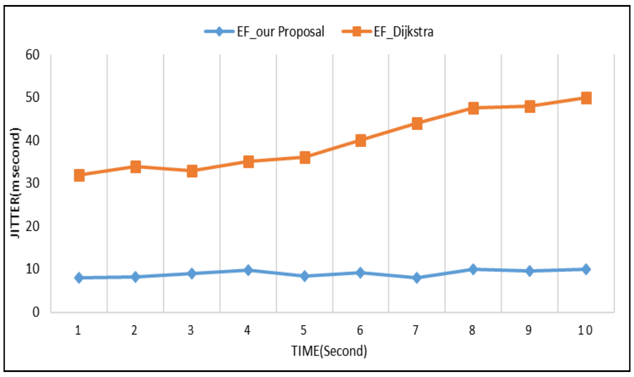

Fig. 7 Jitter for EF packets

Fig. 7 shows that jitter values are approximately constant with a value of 10 msec when transmitting EF packets with the implementation of our proposal. While with Dijkstra, it has varying values starting from 32 msec up to 50msec, and the reason is that our proposal forwards the packets in the path that ensures the appropriate bandwidth for EF packets without causing congestion on this path. Therefore, the packets arrive without a significant delay, which means the jitter value decreases. While with Dijkstra the packets of all classes go in the same path which means congestion. Therefore, packets will suffer from delay which causes an increase of jitter value. That means the effectiveness of our proposal in achieving QoS for EF packets.

F. Measuring jitter for AF packets

Fig. 8 shows the results of the jitter comparison when transmission AF packets.

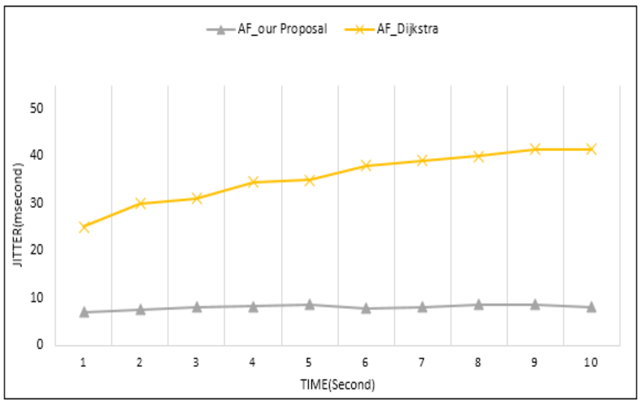

Fig. 8 Jitter for AF packets

By implementing our proposal, jitter maintained an almost constant rate of 8.5 msec when transmission AF packets. While with Dijkstra, it has different values starting from 25msec up to 41 msec. The reason is that our proposal forwards the packets in the path that ensures the appropriate bandwidth of AF packets without causing congestion on this path, after placing them in queue 2. As a result, the packets arrive without significant delay, which means a low value of jitter. While with Dijkstra the packets of all classes go in the same path causing congestion, delay and jitter.

G. *Measuring jitter for BE packets*

Fig. 9 shows the results of the jitter comparison when transmission BE packets.

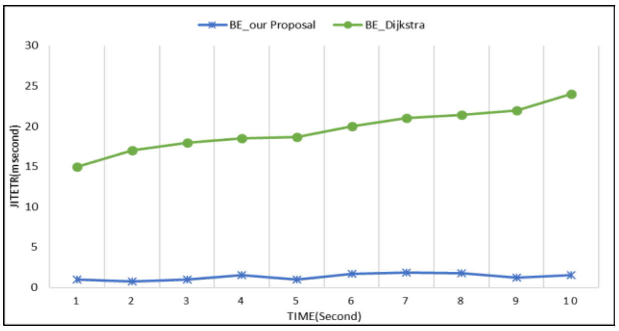

Fig. 9 Jitter for BE packets

We find from Fig. 9 that the values of jitter during the transmission of the last class of packets, which is BE, are close to 1.5 msec when implementing our proposal. While with Dijkstra it has different values starting with the 15 msec up to 24 msec and this is due to the forwarding packets in a path that ensures sufficient bandwidth for these packets without congestion on it. Therefore, there is no significant delay on this path, which means that the value of the jitter decreases. While with Dijkstra, the packets of all classes go in the same path which means congestion and delay which causes increasing in the value of the jitter.

H. *Comparing jitter values between the three classes*

We find from Fig. 10 that our proposal has achieved a low value of the jitter for the three classes compared to Dijkstra algorithm and this is because that each class has sent in the appropriate queue for it, and then choosing the best path.
While the congestion that occurs when using Dijkstra causes a delay in the arrival of the packets to the destination and thus increases the value of jitter for each class as shown in Fig. 10.

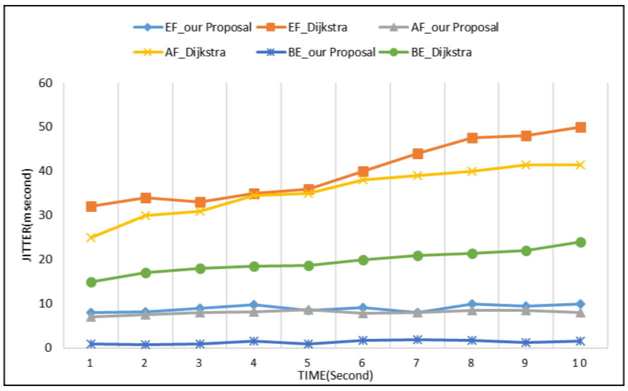

Fig. 10 Jitter values for (EF,AF,BE) packets

Our proposal introduced good values for jitter compared to Dijkstra algorithm. EF and AF packets have close values for jitter, which are low and suitable for transmission. While BE packets do not require any special treatment, so jitter is not considered important during the transmission of these packets, but our proposal gave a low value of jitter. While when using Dijkstra as shown in Fig. 10 jitter has reached high values, especially for EF, which greatly affects the transmission of these packets, specifically it requires high QoS. We find that assigning a priority to packets and placing them in the appropriate queue and then searching for the best path causes the implementation of QoS in the best way.

XI. CONCLUSION

We introduced in this paper a dynamic mechanism to achieve QoS in SDN, by applying traffic engineering in addition to classifying packets based on the concept of differentiated services (DiffServ). This mechanism selects the best path that has the best bandwidth and the lowest packet loss rate. This paper enables the best use of bandwidth in a dynamic mechanism that guarantees the pre-classified flow requirements for the bandwidth which avoids the congestion in the network and reduces the jitter.
We can implement the proposal by using another controller like FloodLight, or in a distributed SDN environment that contains more than one controller, and we can also develop this proposal by using genetic algorithms.

ACKNOWLEDGMENT

The authors wish to acknowledge the Faculty of Information Engineering at Tishreen University for their support of this research.

REFERENCES

[1] D. Rana, S. Dhondiyal, and S. Chamoli, “Software Defined Networking (SDN) Challenges, issues and Solution,” International Journal of Computer Sciences and Engineering (IJCSE), vol.7, Issue-1, Jan 2019.
[2] V. Koryachko, D. Perepelkin, M. Ivanchikova, V. Byshov, and I. Tsyganov, “Analysis of QoS Metrics in Software

Defined Networks," 6th mediterranean conference on embedded computing ,11-15 June 2017.

[3] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár, "Challenges and solution for measuring available bandwidth in software defined networks," Elsevier, 2016.

[4] M. Win, Y. Ishibashi, and K. Mya, "Available Bandwidth Based Application-aware Engineering in SDN," 9th International Workshop on Computer Science and Engineering, 2019.

[5] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," Future Internet, 2014.

[6] I. Godanj, K. Nenadić and K. Romić, "Simple Example of Software Defined Network," IEEE, 2016.

[7] F. Souad, M. MOUGHIT, and N. IDBOUFKER, "OpenFlow Controllers Performance Evaluation," International Journal of Emerging Research in Management & Technology, vol.5, Issue-5, May 2016.

[8] K. Suzuki, K. Sonoda, T. Tonouchi, and H. Shimonishi, "A Survey on OpenFlow Technologies," IEICE TRANS. COMMUN., vol. E97–B, NO.2, FEBRUARY 2014.

[9] A. Ahmad and A. Mohammad, "A Study of OpenFlow Protocol and POX Controller in Software Defined Networks(SDN) Using Mininet," Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series, vol. 41, No.1, 2019.

[10] C. Ghyar, M. Shahade, S. Bamb, and V. Mankar, "Basics of Quality of Services (QoS)," IJSRST, vol. 4, Issue 7, 2018.

[11] A. Adedayo and B. Twala, "QoS Functionality in Software Defined Network," IEEE, 2017.

[12] S. Thukral and B. Chadha, "A Survey on QoS Behavior in MPLS Networks," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, Issue 3, March 2015.

[13] W. Odom and S. Hogg, *CCNA Routing and switching ICND2 200-105 Official Cert Guide*, Cisco Press, USA, 1452, 2017.

[14] D. Aureli, A. Cianfrani, A. Diamanti, J. Vilchez, and S. Secci, "Going Beyond DiffServ in IP Traffic Classification," IEEE/IFIP Network Operations and Management Symposium (NOMS), Budapest, Hungary, April 2020.

[15] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, and C. Yang, "Traffic Engineering in Software-Defined Networking: Measurement and Management," IEEE, 2016.

[16] B. Jadhav, Z. Saquib, and S. Pawar, "ISSUES AND PARAMETERS FOR IMPROVING QoS AND PERFORMANCE IN SDN," International Journal of Advances in Electronics and Computer Science, vol. 4, Issue-7, July 2017.

[17] P. Lopez, J. Gea, F. Martinez, J. Sanahuja, and A. Cruz, "Host Discovery Solution: An Enhancement of Topology Discovery in OpenFlow based SDN Networks," 3th International Joint Conference on e-Business and Telecommunications (ICETE), 2016.

[18] A. Abdulaziz, E. Adedokun, and S. Yahya, "Improved Extended Dijkstra's Algorithm for Software Defined Networks," International Journal of Applied Information Systems (IJAIS), vol. 12, No. 8, November 2017.

[19] Satish, Karuturi S R V, and M Swamy Das. "Multi-Tier Authentication Scheme to Enhance Security in Cloud Computing." IJRAR (International Journal of Research and Analytical Reviews) 6, no. 2 (2019): 1-8, 2019.