QDFSN: QoS-enabled Dynamic and Programmable Framework for SDN

Yousef Darmani [1]*, Mehrdad Sangelaji[2]

Electrical Engineering Dept., K. N. Toosi University of Technology, Tehran, Iran.

[1]Darmani@kntu.ac.ir , [2]M.sangelaji@ee.kntu.ac.ir

*Corresponding Author

Received: 2019-10-26
Revised: 2020-09-05
Accepted: 2020-12-10

Abstract

Software Defined Network (SDN) can integrate a lot of network functions such as network resource management into a consolidated framework. TCP operates in these networks with low information traffic characteristics. As a result, it has to continuously change its congestion window size in order to handle drastic changes in the network or its traffic conditions. As a result, TCP frequently overshoots or undershoots its transmission rate, making it a native congestion control protocol. To overcome that problem, we have proposed a new QoS framework for SDN called QDFSN (QoS-enabled Dynamic and Programmable Framework for SDN) which can be effectively applied in Data Centers as well. In this, and by means of AQM (Active Queue Management), a new function for detecting the upcoming congestion situation is designed. In each node, a developed mathematical model is used to calculate the best parameters of the node adaptively, especially the service rate, to minimize the congestion in the network. This model is tested in many NS-2 scenarios, and the results are presented. The results show improvements in selected QoS parameters like throughput and delay. We conclude that QDFSN-based congestion control shortens the process of adapting TCP to network circumstances, and enhances the TCP performance.

Keywords

SDN (Software Defined Network) – QoS (Quality of Service) –Data Center

1. Introduction

The structure of all data networks including Internet has been shaped by connecting of different switches and routers via telecommunication links. These devices have been provided by a lot of vendors that produce their products with distinct hardware and software. The offered devices are physically and operationally far from each other but have only one issue in common and it is the interconnection protocol. This protocol enables the devices to send and receive data in a predefined manner. Nowadays, the most common protocol is TCP/IP.

Obviously, due to this diversity of equipment, lots of Network Management Systems (NMS) are required to manage the network duties. Each vendor has its own NMS which can only manage its related devices.

Now many issues arise when we want to transfer data from one node to another one which is passing through devices of different vendors. QoS and congestion control are two complicated of them. We should control different routers and switches in an integrated scenario, to offer a unique and predefined QoS. Moreover, we should manage each device dynamically to set its parameters to control the congestion.

To cope with this condition in the current network, we have to introduce new protocols for QoS management, and mandatory implement it in all devices, or rely on the capabilities of the TCP/IP. Actually, it is very hard to obligate all the vendors to implement a certain protocol.

In this regard, if we have a common and central control server to manage all of the network devices, a lot of facilities to implement the QoS requirements will be available. The concept of Software Defined Network would be an answer to this problem. Software Defined Network was perceived at the UC Berkeley and Stanford University in 2008. The Open Networking Foundation (ONF) [1], a non-profit industry association originated in 2011, is devoted to the elevation and adoption of SDN through open standards expansion such as OpenFlow protocol.

The main contribution in SDN is the separation of the control plane and data plane. In SDN we have a central server who manages all operation in the network, and send its messages to the elements via control plane based on the OpenFlow protocol to all nodes of campus or managed domain [2]. The data plane has only been used to forward the data packets between switches and routers of the network. This is a target that we can try to reach, but in the reality, it is impractical to think that existing networks are suddenly going to be divided into completely separate pieces to make an approach for a new world recommended by the ONF and software defined networks. It is also impractical to omit all signs of progress in the networking technology of the Internet. As an alternative, there is more likely a hybrid method whereby several portions of networks are operated by a reasonably centralized controller, while other portions would be run by the more traditional dispersed control plane. This would also infer that those two worlds would need to interwork with each other.

The TCP is commonly used as the default congestion control mechanism [3]. Measurements disclose that the TCP is 99.91% of the traffic in Microsoft data centers [4], 58% of the aggregated global Internet traffic in the world is video streaming over TCP [5], and measurements from 10 major data centers including university, enterprise, and cloud data centers show TCP as the leading congestion control protocol [6]. TCP is a developed protocol and has been widely studied over several years. Hereafter, network operators trust TCP as their congestion control mechanism to make the most of the bandwidth use of their network while keeping the network steady. Despite that, TCP functions in these networks with limited knowledge of the applied network or traffic features. As a result, it is destined to endlessly increase or decrease its congestion window size in order to perform alterations in the traffic or network circumstances. Therefore, TCP often passes or undershoots the best rate, making its behavior ineffective at times [7] [8] [9]. TCP is intended to operate in many types of networks with variant features and traffic situations. We can observe that even in SDN, the most used protocol for congestion control is TCP or its derived versions. Though, restraining TCP to a particular network and taking advantage of the confined features of that network can result in major performance improvement.

This paper introduces the use of SDN in combination with OpenFlow to coordinate the network facilities to improve the QoS. Originally, we propose a QoS management framework that allows us to manage the network flexibly. This is done using the AQM method to dynamically predict the congestion in all nodes in real-time. After that, we explain the model that is outperformed in each node in detail.

The "end-to-end" argument was considered at first in the 1980s as an essential strategy principle of the Internet. Despite huge changes in both primary technologies and also applications on the Internet, this argument is still considerably used in network structures. Nonetheless, the growth of the Internet and the move towards cloud computing also displays the restrictions of this concept. The data center environments that require the managing of the many data flows dynamically, and the variety of the applications that run on its top, have resulted in the growth of interpretation of the end-to-end argument. The end-to-end argument has a valuable effect on the design of the Internet which is convenient and simple. The simplicity and stability of the Internet's design have led to huge growth, but it makes it hard to modify and manage. Today, the demand for designing flexible and universally controlled networks and exclusively data center networks are steadily in progress.

To make the network structure flexible, SDN is a tactic where the "control plane" and the "data plane" are unconnected. In other words, SDN separates the traffic management system which makes decisions about traffic routing (the control plane) from the system beneath that transmits the traffic to the selected destination (the data plane) [10]. In the case of using two or more controllers, the complexity of this concept can be increased of course. The load balancing between these controllers is another issue that is under study and there are some newly proposed methods like distributed load balancing to do that [11]. The need for simplicity, dynamicity, and programmability in data centers is echoed in the design of some concepts related to SDN such as 4D [12], Ethane [13], Tesseract [14], and Openflow [15]. SDN suggests several solutions to separate control and data planes in enterprise networks. This separation simplifies alterations to the network control sense, enables the data and control planes to develop and scale individually, and declines the cost of the data plane portion and improves the possibility for QoS management [16]. The 4D architecture advocates decomposition of the network function into data, dissemination, discovery, and decision planes. Tesseract implements a 4D control plane and shows the merits of 4D in practice. Ethane is a distinguished example of modern work that relies on the parting of data and control planes. The OpenFlow project was initiated to provide a vendor-agnostic border with network elements to enable improvement in the control plane.

2. Related Works

In old deployments of router queue management, the role of the routers is restricted to dropping packets when a buffer becomes full (Drop-Tail). TCP sources run a congestion control protocol that infers packet loss or increases in RTT as an indication of congestion to which they reply by decreasing the transmission rate. AQM mechanisms are paired with the end-to-end congestion control methods in which the congestion is dynamically announced to the sources before the overflow in queues; either obviously by packet marking, or tacitly by dropping many packets. Random Early Detection (RED) [17], is a queue-based AQM which links the congestion announcement to queue size; alternatively, RED drops packets casually. Some other protocols have been proposed, including RED with penalty box [18] and Flow Random Early Drop [19]. These alternatives impose further operation overhead as they need to collect certain types of data, the first one monitors the unfavorable flows while the latter authorizes live connections. Another alternate of RED is Stabilized RED (SRED) [20] which alleviates the use of the router queue. SRED approximates the number of active connections and classifies misbehaving flows, but does not offer a simple router mechanism for correcting these flows. Oppositely, the "CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows" (CHOKe) [21] packet dropping pattern guesses max-min fairness for the flows that pass over a congested router. [22] Compares the TCP and AQM mechanisms, and it tries to give an idea about the best choice of the TCP and AQM couple in various types of network environments.

Explicit Congestion Notification (ECN) [23] is an AQM, initially designed to work in conjunction with RED. ECN tries to stop packet dropping by marking them using a distinct field in the IP header. ECN is an optional method that can only be used when both the receiver and the transmitter can support it. When ECN is effectively assigned, an ECN-aware router may set a bit in the IP packet header in case of dropping a packet in order to announce incoming congestion. The router which receives the packets continues to send the congestion sign to the source, which responds as though a packet is fallen. This behavior is necessary as it removes the overhead of the dropped packets. RED and other similar mechanisms are based on Active Queue Management, whereby an extra number of packets are naturally dropped by the control system in the case of congestion. The congestion is detected only when the queue length is positive. This causes jitter and delay which are not likely. On the other hand, flow-based AQMs, such as GREEN [24] [25], control congestion, and act based on the packet arrival rate. GREEN processes the packet arrival rate and compares it with a threshold level. If the estimated data arrival rate of a link, namely x, is higher than the objective link capacity, c, the rate of congestion announcement, P, will be added by x. If x is lower than c, P is decremented by x.

Besides many studies on different aspects of congestion control methods in the common TCP based networks, some researchers using the concept of SDN and its related capabilities to control the congestion. In [10] authors propose a signaling system that is designed for managing the cross-layer resources. In this framework, session control is integrated with the SDN concept to flexibly manage the services. Introducing a hands-on method, the proposed solution uses the usual and commonly deployed technologies to obtain increased benefits of the SDN. This framework imposes different control methods for the purpose of multifunctional service adaptation.

In [26] authors developed a routing strategy that is based on the SDN and is specially designed for energy saving in QoS-guaranteed backbone links of networks. With SDN virtualization, the change in the network topology can be directly detected by the network controller. In this way, the network can be managed more effectively and easily. Based on the OSPF (Open Shortest Path First) protocol, simple changes in the topology of the network can be handled in this strategy.

In [27] the authors have introduced a framework named Horizon, which predicts the congestion in a Data Center Network (DCN) by using a Markov process. Then it controls the network with the implementation of the user-centric QoS on the nodes. In [28] the authors use Deep Packet Inspection (DPI) in SDN to propose an application-aware system for traffic engineering. They show that, based on the priorities in the QoS levels, the QoS can be optimized by fragmenting the flows and classification of them in different queues.

In [29] the authors have offered a new method to control the congestion of TCP flows by checking the ACK packets by the SDN Controller and in this way tried to modify the ACK header parameters and send them back to the switches of the network. So the end TCP stack has remained unchanged.

In [30] the authors have focused on the short-lived TCP-incast traffic in Data Centers and introduced a method to decrease the unnecessarily long delays. These delays cause this kind of flows to wait for the minimum retransmission timeout (minRTO) to be elapsed.

In order to prioritize different types of traffic such as video, voice, and data, the authors of [31] propose a system, which uses a QoS based routing in SDN. So they can change the configuration of the nodes based on the QoS requirements.

In [25], a general survey has been performed on the QoS-Based routing algorithms which are particularly used in SDN. Two methods have been awarded as the best performers among many studied solutions, and these are delay cost-constrained, and Lagrange relaxation aggregated cost routing algorithms. Another related algorithm is the Network Protocol-based QoS Routing which is proposed in [32]. In [33] the authors have studied the QoS in the VOIP services in SDN. The main parameter which has been considered is Mean Opinion Score (MOS) and also packet loss. The contribution in this work is an architecture that observes the whole network and recognizes the degradation point of MOS in the network in order to compensate it.

As can be seen, in previous research, the contributions are mostly based on the RED algorithm to predict the congestion and make the proper reaction by dropping the packets of buffers. This is the same for FRED, SRED, or CHOKe with different views for prediction. All of these methods can be applied in a network which has not any central management on the parameters of the nodes. So, each node can make its own decision to handle the congestion.

Consider the concept of SDN, the situation is totally different. In this case, the condition of all routers can be observed and it is possible to make decisions to prevent or remove the congestion and satisfy the required QoS. Unfortunately, there is not completed related work in this matter. A research only uses the TCP to get the result [10]. Other works are considering the routing as the main issue and are trying to find the best routes along with considering the QoS [26, 31]. Many other studies are trying to propose various methods based on the end to end dynamic parameters [27, 28]. There are also some studies based on the concept of QoS-based routing which has been discussed in [32] and [33]. [34] Is one of the most related works which has tried to propose a dynamic platform based on the observation of the TCP parameters in the SDN.

None of the works use the AQM methods in combination with the SDN concept to achieve the required QoS. Therefore, this work integrates the AQM with central management of SDN to predict the congestion in the whole network, and use the results for tuning the nodes' parameters to get the requested QoS levels.

As a new contribution, we have proposed a new QoS framework for SDN called QDFSN (QoS-enabled Dynamic and Programmable Framework for SDN) in this paper which can be used in Data Centers as well as the wide area networks. In this framework, employing AQM (Active Queue Management), a new function has been used for predicting the upcoming congestion situation. The function is applied to each node to collaborate with the central controller. A developed mathematical model is used to calculate the optimum parameters of the network nodes, which is concentrated on the service rate. The parameters are adaptive in order to achieve the minimum congestion in the network.

The summary of the most related works can be observed in the following table.

Table I. Summary of Related Works.

No.	Related Work	Contribution and Proposed Method
1	[17]	RED
2	[19]	FRED
3	[20]	SRED
4	[21]	CHOKe
5	[23]	ECN
6	[24]	GREEN
7	[10]	Signaling System to manage cross-layer resources
8	[26]	Routing Strategy in SDN
9	[27]	Horizon with user-centric QoS
10	[28]	DPI in SDN
11	[29]	Checking the ACK Packets
12	[30]	TCP-incast Traffic
13	[31]	QoS based routing in SDN
14	[25]	QoS based routing in SDN
15	[32] , [33]	QoS based routing for VOIP in SDN

3. QDFSN

This research presents a QoS-enabled Dynamic and Programmable Framework to minimize congestion control in SDN-enabled data centers. The proposed framework gathers information about the status of the network and traffic conditions through the SDN controller of each node and uses this information to minimize the congestion in the network.

Instead of TCP end-to-end control mechanism, this work introduces QDFSN as a dynamic adaptation of TCP based on traffic conditions using the SDN concept. Using the SDN idea, QDFSN mainly focuses on interior traffic in SDN-based data centers. Actually, the SDN controller has a universal overview of the network, such as its topology plan and routing information table, and the controller can easily collect appropriate statistics namely, link utilization and traffic congestion notifications. So, QDFSN is deployable as a direct controller application in SDN. QDFSN can be organized in any traditional network by imitating a centralized controller, and gathering network statistics. Our implementation of QDFSN is based on OpenFlow, even though QDFSN should work with any other SDN protocols. A key decision is how we want QDFSN to control the TCP's action. We can either indirectly modify TCP's parameters by changing the information delivered to the nodes (for example, through ECN bits), or directly modify it by having an agent running on the node ports that can update TCP parameters by request. The first choice does not require any changes in the node but gives us slight elasticity to make the desire deviations. While our nodes are managed under integrated management control, making any change on them is not so difficult. This is accurate for example in a data center setting. Consequently, we have intended QDFSN under the supposition that we may change nodes and install an insubstantial agent that can clearly affect TCP sessions. Using the advantages of possible extensible TCP applications, we can easily adapt TCP and even present a complete new congestion control mechanism. On the other hand, we have more time to control the nodes' parameters. TCP's congestion control updates occur on a time scale of the network round-trip time (RTT). This signifies microseconds in data center atmospheres. QDFSN adjusts TCP sources on a time scale, T, which is several times slower than RTTs. The precise value of T is selected by the network administrator. In order to preserve the instability in the network, T requires being a few orders of magnitude higher than the network RTT. Instinctively, by smooth amending of TCP parameters, QDFSN gives each TCP period sufficient time to be changed to a stable state before informing its condition. Furthermore, the time period required for applying the modifications of the parameters is much higher than the RTT of the network by few orders, and so QDFSN can guarantee a very limited overhead on the SDN controller. Here is a part of NS-2 code to simulate the QDFSN.

```
$ns import _OpenFlow();
$ns import _SDN();
$ns append
_packet(data,video,emergency) append
scheduler(classifier)
set Flow(1,2,3) append queue();
$ns _SamplingRate append 1/2S;
$ns _Congestion($Queue(lentgh));
$ns set $packet(ECN(1))
$node() _congestion(flow);
$ns _Flow(trigger) CAC;
```

Under the supervision of QDFSN in the central controller, we should optimize the function which is defined for measuring the congestion in all of the nodes. On the other words, at the same time, the information is gathered from all of the nodes, and based on that the function is optimized regarding the situation of all nodes in the entire network. Then the central controller applies the calculated parameters back on the nodes to get the best optimized performance in the network. The probability function is presented in (1).

$$P_{cong} = \begin{cases} 1, & if\ TH_{max} \leq X(t) \\ f(X(t), P_{max})\ , & if\ TH_{min} \leq\ X(t) < TH_{max} \\ 0, & if\ \ X(t) < TH_{min} \end{cases}$$

(1)

P_{cong} is the probability of congestion in any node and X(t) is the parameter which is used for congestion detection and obviously is the queue length. The central controller is using this probability to sense the congestion in each node and do the optimization calculation to find the optimized parameters to overcome the congestion problem.

Using linear programming for optimizing this function we get (2). In this model, we are using the (μ_{ij}) as the parameter which is the service rate of each buffer, and try to minimize the function for queue length of buffers (q_{ij}) in the network. Actually (q_{ij}) is the queue length which is normalized by the maximum length of the buffer and has a value between 0 and 1.

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij}(\mu_{ij}) \qquad (2)$$

$$q_{ij}(\mu_{ij})\ \geq Min\ TH_{ij}\ , (\mathrm{i} = 1, \dots, \mathrm{n}\ \ \&\ \mathrm{j} = 1, \dots, \mathrm{m})\ (\mathrm{I})$$

$$q_{ij}(\mu_{ij}) \leq Max\ TH_{ij}\ , (\mathrm{i} = 1, \dots, \mathrm{n}\ \ \&\ \mathrm{j} = 1, \dots, \mathrm{m})\ (\mathrm{II})$$

$$q_{1j}(\mu_{1j}) \leq\ q_{2j}(\mu_{2j}) \leq \cdots\ \leq\ q_{nj}(\mu_{nj})\ \ , (\ \mathrm{j} = 1, \dots, \mathrm{m})$$

(III)

$$\sum_{i=1}^{n} C_{ij} =\ C_{out\ j}\ \ , (\ \mathrm{j} = 1, \dots, \mathrm{m}) \qquad (\mathrm{IV})$$

$$0\ \leq\ q_{ij}\ (\mu_{ij}) \leq 1\ , (\mathrm{i} = 1, \dots, \mathrm{n}\ \ \&\ \mathrm{j} = 1, \dots, \mathrm{m})\ (\mathrm{V})$$

$$\mu_{ij} \leq C_{ij}\ , (\mathrm{i} = 1, \dots, \mathrm{n}\ \ \&\ \mathrm{j} = 1, \dots, \mathrm{m}) \qquad (\mathrm{VI})$$

There are some conditions that should be satisfied to get the converged answer. In conditions (I) and (II) it's assumed that queue length will not exceed the upper and lower bounded limits. In condition (III) we assume that queue length of higher priority buffers should be less than the lower priority buffers. Otherwise, the packets should be transmitted firstly from the higher priority buffers to meet this condition. The main reason for this constraint is applying the class of priority to different buffers. Obviously, this constraint is a non-linear conditional constraint and adds a lot of complexity to solving of the problem. So we should relax this condition in some cases. This is a conditional constraint and we can use some solutions like big M or using objective functions to solve the model. At some conditions, the length of the queue in higher priority buffer is non zero while the traffic is not present in other buffers. In this case, maximum output capacity will be dedicated to it to handle the packets inside of high priority buffer. The total output bitrate of the buffers is equal to the capacity of each output link and this limitation is shown in the condition (IV). It relates to the length of each packet in the buffer and the rate of service for that buffer. The queue length function will be between 0 and 1 which is shown in condition (V). The maximum value of service rate in each buffer is the output rate capacity of that buffer which is shown in condition (VI).

The q_{ij} is the buffer length. μ_{ij} is the service rate in each node. Min TH_{ij} is the minimum congestion threshold indicator. Max TH_{ij} is the maximum congestion threshold indicator. C_{ij} is the maximum transmission capacity of each buffer. C_{outj} is the overall node capacity. In these equations "i" is the index of buffers and "j" is the index of the nodes' ports in the network. N is the number of buffers in each node. M is the number of nodes' ports in the network. Figure 1 shows a schematic of the central congestion control unit. In this model, the congestion in the entire network has been controlled by the central congestion control server. Each port in each node has different buffers for classifying the traffic based on the priority of each traffic class. The service rate of each buffer is controlled by the central server, and in periodic intervals, this rate has been re-calculated by the controller and will be implemented to the appropriate buffers of the port in each node. In this way, the rate of output traffic for different flows is controlled and throttled and this will prevent the congestion and packet drop in the next node. By making these arrangements the congestion probability will be reduced and this matter will reduce the number of packet drops and retransmission of packets by itself.

Again please notice that the calculations in each cycle will be made only when the congestion condition in each node is recognized by the central controller, and if, only higher priority buffer has the traffic, all of the output capacity of the link will be dedicated to that buffer e.g.

This optimization problem will be solved by solving a linear programming case by using the proper software. We have used the functional libraries of NS-2 to solve the problem and simulation of the results. Of course, it's possible to use another software like MATLAB for solving the linear programming problems. The output of this optimization is the minimized total buffer length in the network by applying the calculated optimized amounts of service rate in each buffer.

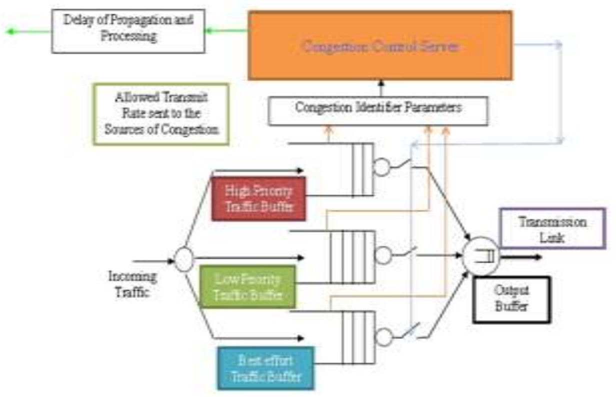
Delay of Propagation and Processing
Congestion Control Server
Allowed Transmit Rate sent to the Sources of Congestion
Congestion Identifier Parameters
High Priority Traffic Buffer
Incoming Traffic
Low Priority Traffic Buffer
Best effort Traffic Buffer
Transmission Link
Output Buffer

Fig. 1. The Schematic view of congestion control model.

The state diagram of this model has been shown in Figure 2.

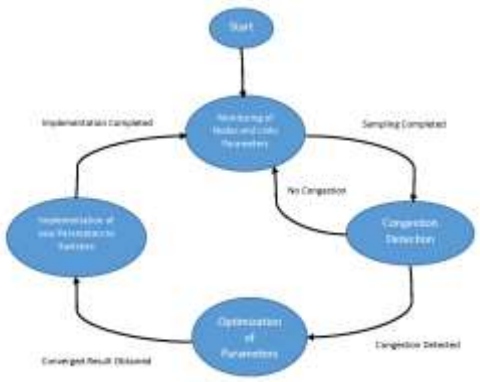

Fig. 2. State diagram of the controller in the proposed model.

As can be seen, the system starts with monitoring and gathering the parameters and congestion indexes from all of the nodes. Then the controller checks the indicators to see if there is any congestion in the network. If there is not any congestion, the algorithm repeats in a timely manner. If the system detects any congestion, it optimizes the network parameters based on the mentioned mathematical model. At the next step, the new obtained parameters which are resulted from the converged calculation are implemented on the network nodes, and the system enters the monitoring state again.

Please consider this not the complete state diagram of the entire controller, but it shows the state diagram of the congestion control module. We have considered this module as one of the internal modules in the central controller. There are some reasons to select such a structure. As it's clear, there is a considerable amount of calculations for making the congestion control model be convergent. So it needs suitable processing power which should be dedicated by the main controller. Otherwise, some other important decisions should be made by the controller and have a relation with QoS control like the routing algorithm. This relation has resulted in the introduction of some methods that perform the QoS management along with the routing at the same time. All of these facts will force us to consider the congestion control module as one of the internal modules of the central controller.

4. Results

The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency, required by many real-time and interactive traffics, and also, controlling packet loss. Moreover, it is important to make sure that providing priority for one or more flows does not affect other flows.

To evaluate the effectiveness of the proposed model for improving the QoS parameters, it is simulated using the NS-2. To evaluate the real performance of the model, three parameters named throughput, delay, and energy consumption are examined. First, we simulate the proposed model with one node and continue the simulation for up to 150 nodes in the network. A simulation set is performed when SDN is enabled. Another simulation is done with OpenTCP [34] context and another simulation with TCP protocol.

For considering the input traffic, please observe the table 2 and 3 which are showing the parameters and features of that. We have assumed that input traffic is randomly applied on the nodes for the large scale networks with 100 or 150 nodes but in more simple conditions the situation is completely explained in detail.

First, we test our method on a simple network including only two nodes. We suppose that we have three kinds of flows including High Priority, Low Priority, and Best Effort. In this simple network, we have the topology as Figure 3.

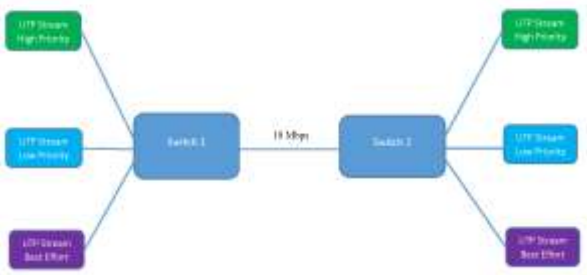

Fig. 3. Simple network with two nodes.

The simulation parameters are as table II.

Table II. Simulation parameters for a network with 2 nodes.

Flow	CBR (Kbps)	RTT (ms)	Start (Sec)	End (Sec)	Packet Length (Byte)
High Priority	5000	7.66	3	100	500
Low Priority	3000	5.4	7	95	250
Best Effort	3000	8.3	17	32	250

We compare the simulation result of these three methods as follows.

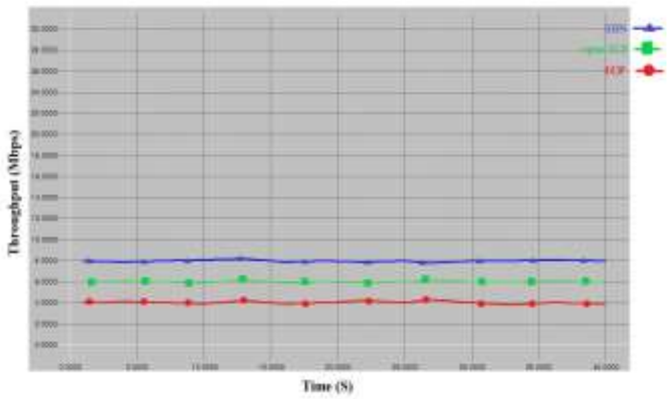

Fig. 4. Comparison of Throughput based on Mbps in a simple network with two nodes. The triangle line is related to the SDN-enabled network, the squared line represents the open-TCP network and the circled line is related to the network which is pure TCP

It can be observed that using QDFSN has given us more throughputs on the congested link. As seen in Figure 4 the overall throughput of the network with SDN implementation is 8 Mbps while the overall throughput of OpenTCP implementation is 6 Mbps and TCP implementation is 4 Mbps.
In this situation, the utilization ratio is above 80% which is more than other methods.

At a second step, we assume a more complex network with 5 nodes. The topology of this network has been selected as in Figure 5. In this case, we still have a bottleneck link but with more routes in the next hops. To have a different condition we preferred to increase the bandwidth of the links to 12 Mbps.

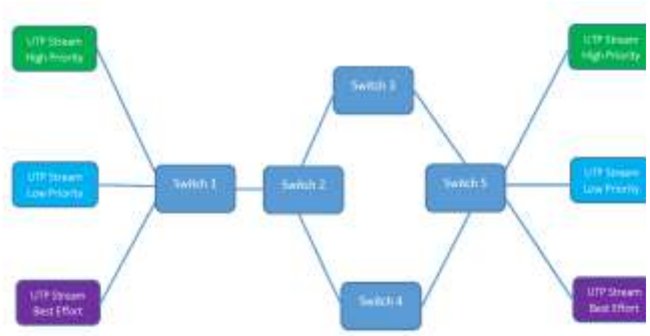

Fig. 5. Schematic view of a network with five nodes.

The simulated parameters are based on table III.

Table III. Simulation parameters for a network with 5 nodes.

Flow	CBR (Kbps)	RTT (ms)	Start (Sec)	End (Sec)	Packet Length (B)
High Priority	5000	7.66	3	100	500
Low Priority	4000	5.4	7	95	250
Best Effort	4000	8.3	17	32	250

The simulation has been done with three methods and the result has been demonstrated in Figure 6.

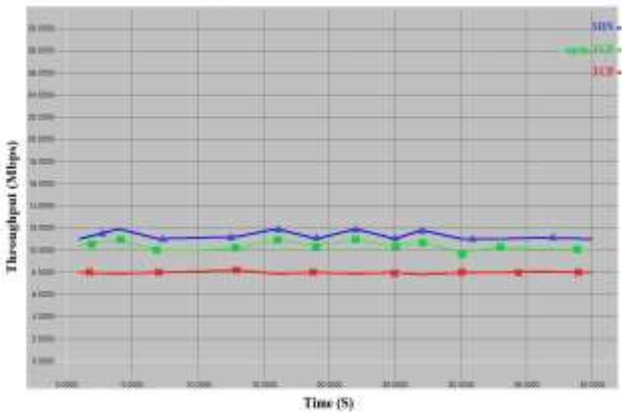

Fig. 6. Comparison of Throughput based on Mbps in a simple network with five nodes. The triangle line is related to the SDN-enabled network, the squared line represents the open-TCP network and the crossed line is related to the network which is pure TCP.

A comparison between QDFSN and other methods shows that the overall throughput is getting better. In the test, the OpenTCP implementation has negligible jitter, but overally, the performance of SDN is better than OpenTCP and TCP implementations. The results show about 12 Mbps for SDN-based network, 10 for OpenTCP implementation, and 8 Mbps for TCP. At the next step, we try to test the idea on the more complex network topologies. In this regard we built new random topology with 100, and also 150 nodes to simulate the model in a larger network.

View of these topologies is shown in Figures 7 and 8.

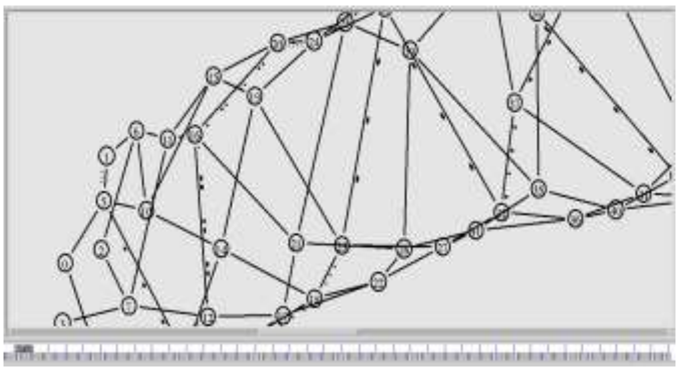

Fig. 7. NAM representation of network (Closed view - Upper bound).

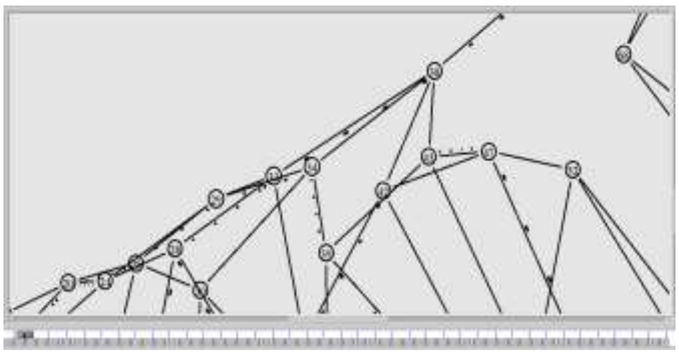

Fig. 8. Schematic view of the simulated network (Bottom bound - closed view).

At this time, we focused on other parameters such as delay and energy consumption. Please notice that energy consumption is the power that is consumed in the central controller server for processing. Actually, it will show the amount of processing that is required to solve the equations of the model with convergence and calculate the required parameters for controlling the network traffic. As the first scenario, we load the network with three flows that have different levels of priority. These flows are transmitted into the network through random nodes. We select a congested network with minimum traffic. Now we can compare many quality-related parameters including throughput, delay, and energy consumption. Please notice that the energy in the form of power can be also saved by controlling the frequency of the central processor based on the volume of traffic [35].

Like the first simulation, we choose the parameters based on table 2. The bandwidth of each link in the network is 10 Mbps and we are confident that the total amount of injected traffic is much higher than this value.

We tested this condition in different topologies and the results are evaluated for two random 100-nodes and 150-nodes network. Consider that topology of these two networks is random and there is not any similarity between them, so it is possible that each network shows a specific response to the traffic congestion, and it is because of a possible bottleneck in their connections.

Figures 9 to 11 are shown the delay, energy consumption, and throughput in a 100-nodes random network.

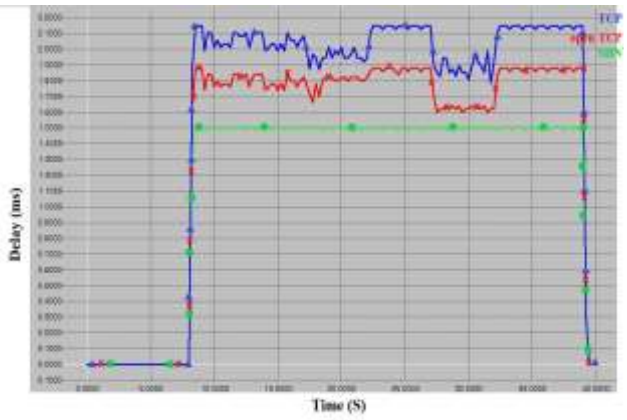

Fig. 9. Comparison of delay imposed to network with 100 nodes. The squared line is related to the SDN-enabled network, the crossed line represents the open-TCP network and the triangle liner is related to the network which is pure TCP.

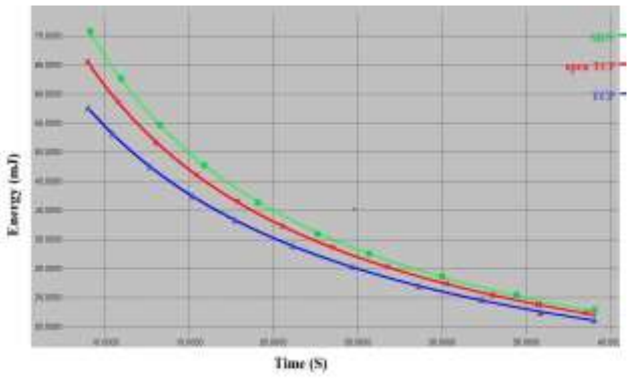

Fig. 10. Comparison of energy consumption of a network with 100 nodes. The squared line is related to the SDN-enabled network, the crossed line represents the open-TCP network and the triangle line is related to the network which is pure TCP.

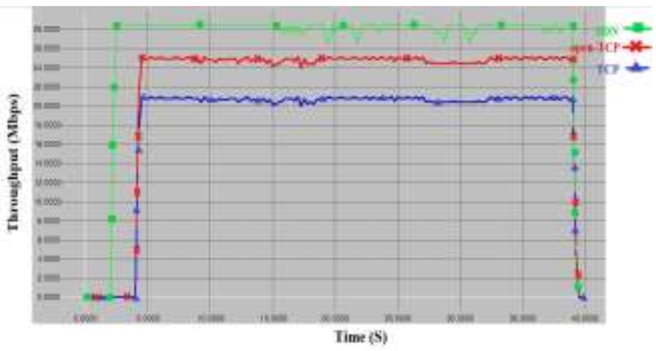

Fig. 11. Comparison of throughput gained from a network with 100 nodes. The squared line is related to the SDN-enabled network, he crossed line represents the open-TCP network and the triangle line is related to the network which is pure TCP.

To further evaluate our proposed model, we repeat the simulations with a higher number of nodes in a 150-nodes random network. This time, 150 nodes are in a simulation environment, and these results are obtained which are shown in Figures 12 to 14.

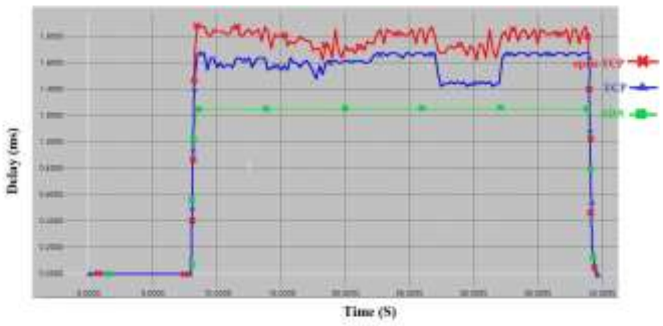

Fig. 12. Comparison of delay imposed to network with 150 nodes. The squared line is related to the SDN-enabled network, the crossed line represents the open-TCP network and the triangle line is related to the network which is pure TCP.

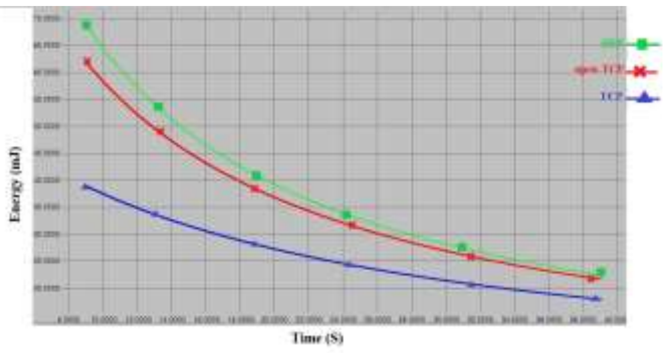

Fig. 13. Comparison of energy consumption of a network with 150 nodes. The squared line is related to the SDN-enabled network, the crossed line represents the open-TCP network and the triangle line is related to the network which is pure TCP.

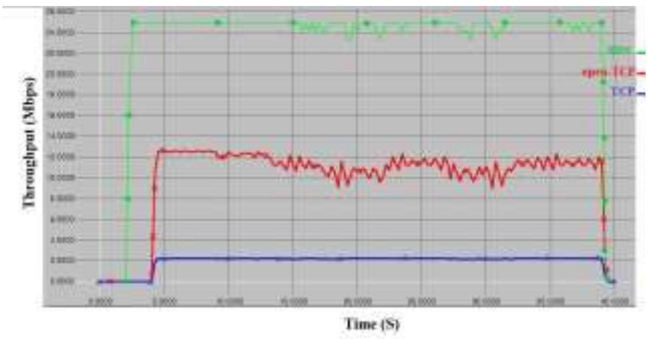

Fig. 14. Comparison of throughput from a network with 150 nodes. The squared line is related to SDN-enabled network, the crossed line represents the open-TCP and the triangle line is related to the network which is pure TCP.

As seen, TCP pacing positively mitigates the TCP concurrent flows. In a large network or data center with thousands of simultaneous flows, it is almost impossible to guarantee that the number of live flows will always be less than any threshold. Consequently, providers are unwilling to permit pacing despite its proven efficiency. Nonetheless, with the proposed method, TCP can be dynamically managed to ensure the system works at its peak performance.

Diagrams show that using the QDFSN in a network with a noticeable number of the nodes can enhance the performance of the network. The Throughput in both 100 and 150 nodes networks is higher than other methods. Consider that the numeric values of the throughput are not important because it deeply depends on the network topology and interconnection of its links. The same case applies for the delay. It is shown that the delay has been lowered for the flows using the QDFSN in comparison with other methods, and this means that flows with higher priority can pass through the network with guaranteed quality.

Although using QDFSN leads to better performance, it has its own price which is more energy consumption. As seen in the related diagrams, the energy consumption of the QDFSN method is higher than pure TCP, and also than open-TCP. For pure TCP it's clear because there is not any calculation for pure TCP in meanwhile of flow transfer and all of the decisions are made end-to-end. But in comparison with open-TCP still we have more calculations to solve our more complex model in QDFSN and it takes more energy. Of course, the difference is not considerable and will be decreased by time. Notice that at the beginning of each calculation cycle, the volume of calculations in the QDFSN is much higher than other methods, and energy consumption diagrams are showing this. By passing a reasonable period, the input traffic that may have the burst nature will be controlled between the buffers and the amount of processing will be decreased in a result.

5. Conclusion

Congestion control has been widely studied for several years. In light of the emerging popularity of centrally controlled SDN, we ask whether we can take advantage of the information available at the central controller to improve TCP. Specifically, in this paper, we examine the design and implementation of QDFSN, a dynamic and programmable TCP adaptation framework for SDN-enabled data centers. QDFSN gathers information about the status of the network and traffic conditions through the SDN controller per node and uses this data to adjust TCP. QDFSN sporadically sends updates to nodes which, in turn, inform their behavior using a simple kernel part. In this paper, we discuss the architectural design of QDFSN, as well as its implementation and simulation with the open-source discrete event simulator NS-2.

In other words, in this research, we proposed a new QoS framework for SDN called QDFSN. Compared with traditional TCP function, QDFSN-based congestion control shortens the process of adjusting TCP to network circumstances. Using the SDN concept, QDFSN can alter TCP's parameters. We take advantage of the information available in the central controller to improve the performance of TCP. In QDFSN, the network operator has complete control over the changes applied totally through the congestion control rules. Congestion control plans tell QDFSN how to adjust and which constraints to satisfy. Although this gives choice to the network operator, it runs the possibility of instability in the network. This is because the operator might define an unstable congestion control procedure or it might present mismatched congestion control policies in the whole network. Instability might likewise be caused if the operator chooses an inappropriate time scale to update TCP agents. QDFSN takes steps to guarantee the stability of the system in practice changing the default TCP parameters when it faces instability in the arrangement. A more prescribed definition of stability along with theoretical exploration and further trials remains an interesting future work. Besides, we had not any assumption about the location of the controller and the effect of the propagation delay of the command between the controller and the nodes, and it may become a challenging issue to study.

6. References

[1] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center", NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012, San Jose, CA pp. 19–19.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008. DOI: 10.1145/1355734.1355746.

[3] V. Jacobson, "Congestion avoidance and control", ACM SIGCOMM CCR, vol. 25, no. 1, pp. 157–187, 1995. DOI:10.1145/205447.205462.

[4] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)", ACM SIGCOMM, 2010, New Delhi, India pp. 63–74. DOI:10.1145/1851182.1851192.

[5] Sandvine global Internet report. https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf, Oct. 2018.

[6] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild", Internet Measurement Conference, 2010, Melbourne, Australia pp. 267–280. DOI: 10.1145/1879141.1879175.

[7] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, “Understanding TCP incast throughput collapse in datacenter networks”, WREN, 2009, Barcelona, Spain pp. 73–82. DOI:10.1145/1592681.1592693.

[8] C. Jerry, “Tuning TCP Parameters for the 21st century”, http://www6.ietf.org/mail-archive/web/tcpm/current/msg04707.html.

[9] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, “Safe and effective fine-grained TCP retransmissions for datacenter communication”, SIGCOMM, 2009, Barcelona, Spain, pp. 303–314. DOI:10.1145/1592568.1592604.

[10] W. Cerroni, M. Garbaoei, “Cross-layer resource orchestration for cloud service delivery: A seamless SDN approach”, Computer Networks, vol. 87, pp. 16-32, 2015. DOI:10.1016/j.comnet.2015.05.008.

[11] B. Ahmadi; Z. Movahedi, “Stable Distributed Load Balancing between Controllers in Software Defined Networks”, Article 2, vol. 49, Issue 1 - Serial Number 87, Spring 2019, Page 13-23.

[12] Greenberg, G. Hjalmtysson, D. A Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A clean slate 4D approach to network control and management”, ACM SIGCOMM Computer Communication Review, 2005, vol. 35, no. 5, pp. 41-54. DOI:10.1145/1096536.1096541.

[13] Software Defined Networks (SDN) talk at Structure 2010. http://www.openflowswitch. org/wp/2010/07/software-defined-networks-sdn-talk-at-structure-2010/.

[14] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, “Tesseract: A 4D network control plane”, 4th Symposium on Networked Systems Design and Implementation, 2007, Cambridge, Massachusetts, USA.

[15] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: taking control of the enterprise”, SIGCOMM '07 Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, 2007, Kyoto, Japan vol. 37, no. 4, pp. 1–12. DOI:10.1145/1282380.1282382.

[16] M. Karakus, A. Durresi, “Quality of Service (QoS) in Software Defined Networking (SDN): A survey”, Journal of Network and Computer Applications, vol. 80, pp. 200-218, 2017. DOI:10.1016/j.jnca.2016.12.019.

[17] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance”, IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397–413, 1993. DOI:10.1109/90.251892.

[18] S. Floyd and K. Fall, “Router mechanisms to support end-to-end congestion control”, Technical report, 1997.

[19] D. Lin and R. Morris, “Dynamics of random early detection”, Proceedings of the ACM SIGCOMM '97 ?Conference on Applications, technologies, architectures, and protocols for computer communication, 1997, Cannes, France,vol. 27, no.4, pp. 127–137. DOI:10.1145/263105.263154.

[20] T. Ott Lakshman, T. V. Lakshman, and L. Wong, “Sred: Stabilized red”, IEEE INFOCOM '99, Conference on Computer Communications, New York, NY, USA,1999, pp. 1346–1355. DOI: 10.1109/INFCOM.1999.752153.

[21] R. Pan, B. Prabhakar, and K. Psounis, “Choke - a stateless active queue management scheme for approximating fair bandwidth allocation”, Proceedings IEEE INFOCOM 2000, Conference on Computer Communications, 1999. DOI: 10.1109/INFCOM.2000.832269.

[22] C. A. Grazia, N. Patriciello, M. Klapez and M. Casoni, “A cross-comparison between TCP and AQM algorithms: Which is the best couple for congestion control?”, 14th International Conference on Telecommunications (ConTEL), Zagreb, 2017, pp. 75-82. DOI: 10.23919/ConTEL.2017.8000042.

[23] S. Floyd, “TCP and explicit congestion notification”, ACM Computer Communication Review, vol. 24, no. 5, pp. 10–23, 1994. DOI:10.1145/205511.205512.

[24] J. Hong, C. Joo, and S. Bahk, “Active queue management algorithm considering queue and load states”, Proceedings, 13th International Conference on Computer Communications and Networks, Chicago, IL, USA, 2007, vol. 30, pp. 886–89. DOI: 10.1109/ICCCN.2004.1401608.

[25] J. W. Guck, A. Van Bemten, M. Reisslein and W. Kellerer, “Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation”, IEEE Communications Surveys & Tutorials, vol. 99, pp. 1-1, 2017. DOI: 10.1109/COMST.2017.2749760.

[26] P. Hongyu, W. Weidong, and W. Chaowei, “QoS-guaranteed energy saving routing strategy using SDN central control for backbone networks”, The Journal of China Universities of Posts and Telecommunications , vol. 22, no. 5, pp. 92-100, 2015. DOI:10.1016/S1005-8885(15)60686-0.

[27] J. Pang, G. Xu, X. Fu, K. Zhao, “Horizon: a QoS management framework for SDN-based data center networks”, Annals of Telecommunications, vol. 72, no. 1, pp. 597–605, 2017. DOI: 10.1007/s12243-017-0579-2.

[28] S. Jeong, D. Lee, J. Hyun, J. Li and J. W. K. Hong, “Application-aware traffic engineering in software defined network”, 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2017, Seoul, Korea (South), pp. 315-318. DOI:10.1109/APNOMS.2017.8094144.

[29] A. Volkan Atli, M. Serkant Uluderya, S. Civanlar, B. Görkemli, A. Murat Tekalp, “TCP congestion avoidance for selective flows in SDN”, 26th Signal Processing and Communications Applications Conference (SIU), 2018. DOI: 10.1109/SIU.2018.8404643.

[30] A. M. Abdelmoniem, B. Bensaou, A. James Abu, “Mitigating incast-TCP congestion in data centers with SDN”, Annals of Telecommunications, April 2018, Volume 73, Issue 3–4, pp. 263–277.DOI:10.1007/s12243-017-0608-1.

[31] A. Kucminski, A. Al-Jawad, P. Shah and R. Trestian, “QoS-based routing over software defined networks”, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2017, Cagliari, pp. 1-6. DOI: 10.1109/BMSB.2017.7986239.

[32] Shakthipriya P., Bevi A.R., “Network Protocol-Based QoS Routing Using Software Defined Networking. Artificial Intelligence and Evolutionary Computations in Engineering Systems”, Springer, Singapore, pp. 363-374, 2017. DOI:10.1007/978-981-10-3174-8_32.

[33] B. Siniarski, C. Olariu, P. Perry and J. Murphy, “OpenFlow based VoIP QoE monitoring in enterprise SDN”, IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, 2017, pp. 660-663. DOI: 10.23919/INM.2017.7987354.

[34] M. Ghobadi, “TCP Adaptation Framework in Data Centers. Doctor of Philosophy, Graduate Department of Computer Science”, University of Toronto, 2013.

[35] A. Ghiasian, “Frequency scaling approach to reduce the power consumption of Openflow switches”, Tabriz Journal of Electrical Engineering, Volume 49, Issue 3 - Serial Number 89 , pp. 1273-1282 Autumn 2019.