DOI: https://doi.org/10.33103/uot.ijccce.19.4.5

Adaptive Performance Evaluation for SDN Based on the Statistical and Evolutionary Algorithms

Afrah Salman Dawood[1], Mohammed Najm Abdullah[2]
[1,2]*Department of Computer Engineering, University of Technology, Baghdad, Iraq*
afrah.salman@ymail.com, mustafamna@yahoo.com

Abstract- *Being able to send different types of data (i.e. text, audio, or video) through the network is the most important aspect of networks. Different networks have different issues and restrictions while sending data. These restrictions are basically the QoS (Quality of Service) metrics and security. The recent Software-Defined Networking (SDN) that aims to separate the control plane from the data plane can be applied where Business requirements are not responsible for the way the network is configured; instead, it is the responsibility of the high-level business policies and objectives. SDN gives preferable techniques for centralized dynamic management and control configurations. In this work, a proposed model has been estimated and discussed to promote QoS requirements in some suggested topologies. Adaptive Resource Management (ARM) and control to send different types of data through different hosts have been investigated. The intended requirements are basically the capacity and delay of traffic metrics sent through different hosts through the network. It produces a mathematical model and implementation for three proposed algorithms to enhance the quality of a sample video sent from source host to destination host by Visible Light Communication (VLC)-media player in three different topologies. These algorithms (statistical, MOGA, and PSO) have been implemented using Mininet emulator, FNSS tool, PULP, and network libraries; with two types of controllers which are Floodlight and OVS under Linux operating system and in python programming language.*

Index Terms- SDN, Performance Evaluation, ARM, QoS, MOGA, PSO, FNSS, Mininet, Floodlight Controller, OVS Controller.

I. INTRODUCTION

Many companies, industries and researches are now moving toward SDN technology in networking [1] because of the basic difficulty in traditional IP networks which is represented mainly in the complexity of management and configuration of all devices in the network. In other words, the source code of the configuration must be set up throughout all switches, routers, and other devices. Nowadays, implementing networks with SDN configuration doesn't require that the configuration setup on all devices, instead, only the controller is responsible for controlling and managing the network [2]. This controller is the brain of the network and all control-plane data pass through to other forwarding devices which are in this case only dummy devices and is responsible only for transferring data (i.e. through the data-plane). The main point in SDN is that it separates the control plane (i.e. management plane) from the data plane (i.e. forwarding plane) [3].

The environment of SDN that is also known as **Software-Defined Environment (SDE)** is mainly responsible on the management of different resources in such networks [4]. Some of these resources include sending different types of data among network hosts [5], dynamic resource management for QoS [6], enabling **High-Definition (HD)**-map-assisted cooperative driving among **Autonomous Vehicles (AVs)** to improve the navigation safety [7], ensuring security of transferred data, etc. SDN and **Network Virtualization (NVI**) are widely considered promising techniques for reducing the

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

complexity of network management in many contexts that require high QoS and the support for heterogeneous architectures [8].

Resource Management (RM) can be optimized in different methods; some of these methods are the General MOGA that is used to handle multi-objective optimization problems in different search spaces [9].

The rest of this paper is organized as follows: Section 2 shows some works that are related to the concept of this work, while section 3 discusses the basic problem definition that is to be explained and implemented in section 4. Finally, sections 5 and 6 review experimental results and conclusion, respectively.

II. RELATED WORK

RM is very essential approach in all types of networks. Researchers in reference [10] can be considered as the first attempt for strict QoS requirements and manage the exchange among different network devices. RM mechanisms, provided by obtainable SDN approaches according to OpenFlow, have been summarized in reference [11]. Reference [9] used MOGA with PSO to eliminate the distributed controller placement problem that finds out the pareto optimal solutions minimizing the switch-to-controller load imbalance for wide area SDN. A general model has been produced and discussed; this model also considers switch assignments beside to dealing with the controller placement, and explains evaluations and results. Reference [12] is a paper about highlighting the comparative analysis of nature inspired Swam Intelligence based optimization techniques according to literature analysis and the areas where these algorithms have been most successfully applied. An investigation about the problem of saving energy in hybrid IP/SDN networks [13] where traditional IP nodes are incrementally replaced by SDN ones. **Genetic Algorithms (GA)** has been proposed and evaluated through simulations over realistic network topologies to solve this problem. The paper in reference [14] proposes a novel intelligent technique that has been designed to optimize the performance of SDN. The proposed hybrid intelligent system has employed integration of intelligence-based optimization approaches with the artificial neural network. These heuristic optimization methods include GA and PSO.

III. PROBLEM DEFINITION

A network scenario has been considered in details in this section. A network $G\ (E, L)$ is a graph of a datacenter topology implemented in python programming language under PULP basis and run in Mininet as an SDN with OVS and Floodlight controller. Where, $L = \{l_1, l_2, \dots, l_n\}$ represents vertices (i.e. links available in each path in the topology) and $E = \{e_1, e_2, \dots, e_m\}$ represents edges of the proposed topology. Traffic flow description is described by a traffic matrix $T_t = (s, de)$ sent from each source $s \in E$ to each destination $de \in E$ in the network in a specific time interval, t. Edges have capacities belonging to the set $C = \{c_1, c_2, \dots, c_L\}$. The goal is to find suitable capacity for each link in the topology regarding the primary delay D_l of the link. Increasing the capacity of the link l_i results in an increased bandwidth BW of the path which in turn reduces the total delay D_t of the path between two hosts.

The issue is that when packet size is expanded, there will be more delays in packet delivery proportion. However, there must be some relationship between the amount of the capacity c_l assigned to the link $l \in L$ and the packet size. In view of this suspicion, an Integer Linear Programming (ILP) that attempts to recognize a set of capacities C assigned to vertices $l \in L$ dependent on packet size according to statistical regression approach is going to be presumed according to values of Fig. 1. Regression analysis has been made according to equation 1.

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

$$y = bx + a \quad (1)$$

From the equation above and from the measured data, regression line equation can be calculated using equation 2 and thus the capacity of every link can be estimated relating to algorithm 1:

$$c_l = 13.1237 + 0.03436 * D_l \quad (2)$$

ALGORITHM 1: STATISTICAL APPROACH PSEUDO CODE DESCRIPTION

Input: network graph composed of edges and links $\boldsymbol{G} = (\boldsymbol{E}, \boldsymbol{L})$	
Output: optimal capacity for each link in the graph	
1	Import essential libraries
2	Formulate G = (E, L) using FNSS library
3	Set basic settings for the network
4	**For all** l **in** L **do**
5	$d \leftarrow delay\ of\ l$
6	$c = f(d)$
7	$sol \leftarrow c$
8	**End for**
9	**Return** sol
10	Draw the network with nx library or web UI
11	Set video transferring properties
12	Export to Mininet topology, launch the controller, and start the network
13	Record results and evaluate the results
14	Stop the network and clear all $ethernet\ links$

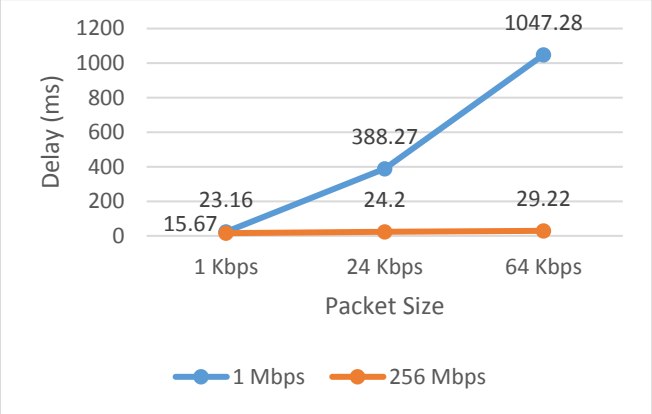


FIG. 1: MEASURED DATA FOR ESTIMATION

The statistical approach has been implemented according to Algorithm 1 above. In the second approach (GA), the population is represented by the variable P and its size is I. It is composed of a list of chromosomes represent random values of capacities$c \in C$. Each chromosome $c_p \in P$ in the generated population P represents a specific capacity and is represented by a set of genes in binary representation (for example$c_p = 11001110 = 206\ Mbps$). The following is an example of crossover process for two different chromosomes:

Chromosome 2	0	0	1	1	0	0	0	1
Chromosome 1	1	1	0	0	1	1	1	1

After performing the single-point crossover process for these chromosomes, two children will be resulted as follows:

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

Child 1	0	0	1	1	1	1	1	1
Child 2	1	1	0	0	0	0	0	1

Where child 1 is going to be the first capacity which is 63Mbps, while the second child represents a capacity of 193Mbps. Binary values are utilized to symbolize the two potential operational modes of each link, $l \in L$: $g_l = 0$ $(powered\ off)$; $g_l = 1$ $(powered\ on)$. The best solution sol_{c_p} is achieved when the network configuration is able to route the set of traffic demands, $T_t = (s, de)$and be content with both conservation and delay D_l limits. The fitness function (equation 3 bellow) is a multi-objective function (MO) of traffic demands T_t that are transferred from source to destination. It is composed of subtracting the capacity c_p from the target t multiplied by the state g_i and delay D_l of the link $l \in L$.

$$f(c_{p,}, G, T_t) = (t - c_p) g_l D_l \; ; \forall l \in L; g_l \in \{0; 1\} \tag{3}$$

The innate genetic functions of GA (selection, crossover, mutation and replacement) are repeated for p generations to response with the preferable solution sol. The crossover is based on one-point selection by dividing each parent into two halves and then performing the crossover by copying everything before this point from the first parent to the second parent. Algorithm 2 bellow describes the pseudo code of GA approach, where $f(cp, G, T_t)$ is computed according to equation 3 above.

ALGORITHM 2: GA PSEUDO CODE DESCRIPTION

Input: network graph composed of edges and links $\boldsymbol{G} = (\boldsymbol{E}, \boldsymbol{L})$	
Output: optimal capacity for each link in the graph	
1	Import essential libraries
2	Formulate G = (E, L) using FNSS library
3	Set basic settings for the network
4	$P \leftarrow Generate\ population(G, I)$
5	**For all** c_p **in** P **do**
6	$(fval_{cp}) \leftarrow f(cp, G, T_t)$
7	**End for**
8	Compute $av_{fitness}$
9	**Do{**
10	$parents \leftarrow selectParents(P, rank - selection)$
11	$children \leftarrow crossover(parents, single - point)$
12	$children \leftarrow mutation(children, non - uniform)$
13	$P \leftarrow replace(children)$
14	**For all** c_p **in** $children$ **do**
15	$(fval_{cp}, sol_{cp}) \leftarrow f(cp, G, T_t)$
16	**End for**
17	Compute $av_{fitness}$
18	$sol \leftarrow sol_{cp}$
19	p = $current\_av_{fitness}$ − $previous\_av_{fitness}$
20	**}While (**p < 10^{-6}**)**
21	**Return** sol
22	Draw the network with nx library or web UI
23	Set video transferring properties
24	Export to Mininet topology, launch the controller, and start the network
25	Record results and evaluate the results
26	Stop the network and clear all $ethernet\ links$

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

The accordance of a parent $Accordance_i$ (equation 4 below) is calculated by finding the preferable fitness value f corresponding to the global best position.

$$Accordance_i = f_t \text{ Gbest}/f_i \quad (4)$$

General PSO algorithm uses equation 5 to find current velocity of the particle according to the previous velocity and last position P^l.

$$\mathrm{V}^c = w\,\mathrm{V}^l + c_1 r_1\left(Lbest - \mathrm{P}^l\right) + c_2 r_2\left(Gbest - \mathrm{P}^l\right) \quad (5)$$

$$\mathrm{P}^c = \mathrm{V}^c + \mathrm{P}^l \quad (6)$$

$$\mathrm{V}^c = c_2 r_2\left(Gbest - \mathrm{P}^l\right) \quad (7)$$

Where $Lbest$ is the local best position of the particle, w represents inertia of a particle [15], c_1, r_1 and$c_2 r_2$ are the amount of experience to be learned from $Lbest$ and $Gbest$, respectively, r_1, r_2 are random variables in (0,1) range, and finally, P^c in equation 6 above is the current position of the particle based on its last position and velocity. Since each particle in our mutation function only runs one cycle, the w that takes the weight of last velocity and the local best position does not mean anything in our situation, the first two parts of equation 5 are dropped, and a new equation 7 has been created to calculate the velocity. Algorithm 3 illustrates the pseudo code description of the PSO approach.

ALGORITHM 3: PSO PSEUDO CODE DESCRIPTION

Input: network graph composed of edges and links $\boldsymbol{G = (E, L)}$	
Output: optimal capacity for each link in the graph	
1	Import essential libraries
2	Formulate G = (E, L) using FNSS library
3	Set basic settings for the network
4	$Generate\ swarm(G, I)$
5	**For each** $particle$ **in** $swarm$ **do**
6	$(fval_{particle}) \leftarrow f(particle, t, T_t)$
7	**End for**
8	Compute $\mathrm{av}_{\mathrm{fitness}}$
9	**Do{**
10	**For each** $particle$ **in** P **do**
11	Compute Accordance
12	Arrange $particles$ According to their Accordance
13	$v_{cp} \leftarrow velocity(Gbest, p^l)$
14	$p^c \leftarrow new\ position(v_{cp}, p^l)$
15	$particles \leftarrow replace(p^l, p^c)$
16	**For each** $particle$ **in** $swarm$ **do**
17	$(fval_{particle}) \leftarrow f(particle, t, T_t)$
18	**End for**
19	Compute $\mathrm{av}_{\mathrm{fitness}}$
20	p = $current\_\mathrm{av}_{\mathrm{fitness}} - \mathrm{previous\_av}_{\mathrm{fitness}}$
21	**}While (**$p < 10^{-6}$**)**
22	Draw the network with nx library or web UI
23	Set video transferring properties
24	Export to Mininet topology, launch the controller, and start the network
25	Record results and evaluate the results
26	Stop the network and clear all $ethernet\ links$

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

IV. Experimental Results

The proposed models and algorithms illustrated in the previous section have been implemented and tested on three different datacenter topologies and dumbbell topology with two controllers which are Floodlight and OVS and the adaptive resource management has been tested on two different video streams (their properties are illustrated in Table 1) played on VLC media player.

TABLE 1: DETAILED PARAMETERS OF TESTED VIDEO FILES

Video File Detailed Parameters		
Video 1	Video Format	MP4
	Duration	26 second
	Total File Size	2,100,396 bytes
	Resolution	320x240
Video 2	Video Format	MP4
	Duration	18 second
	Total File Size	1,804,076 bytes
	Resolution	640x360

As mentioned earlier, we have implemented different tiers of Data Center (DC) topologies infrastructure which are two and three tiers with five edges, two hosts per edge, and different number of cores (i.e. one, two, three, and five cores). It is worth mentioning that the number of aggregation switches has been considered to be a constant value of two. We have also implemented fat tree topologies with two levels, two and four levels. Finally, dumbbell topology has also been implemented with two values (four and ten) of nodes in each bell and two values (five and ten) of nodes in the path. This design implementation has been done with the FNSS library using the following commands:

```
fnss_topo = fnss.two_tier_topology(n_core=1, n_edge=5,
                                   n_hosts=2)
fnss_topo = fnss.three_tier_topology(n_core=1,
                                     n_aggregation=2,
                                     n_edge=5, n_hosts=2)
fnss_topo = fnss.fat_tree_topology(2)
fnss_topo = fnss.dumbbell_topology(4, 5)
```

The capacity of each link has been set adaptively relating to link delay and according to the proposed models explained previously in chapter three; the following command represents a function call for the already built-in solutions:

```
set_capacities_packet_delay(fnss_topo, buffer_unit='bytes',
                            capacity_unit='Mbps')
```

Now, we have built a complete FNSS scenario and it is ready to be converted to Mininet using the following command:

```
mn_topo = fnss.to_mininet(fnss_topo, relabel_nodes=True)
```

The complete topologies designed and implemented with both OVS and Floodlight controllers using the following commands, respectively:

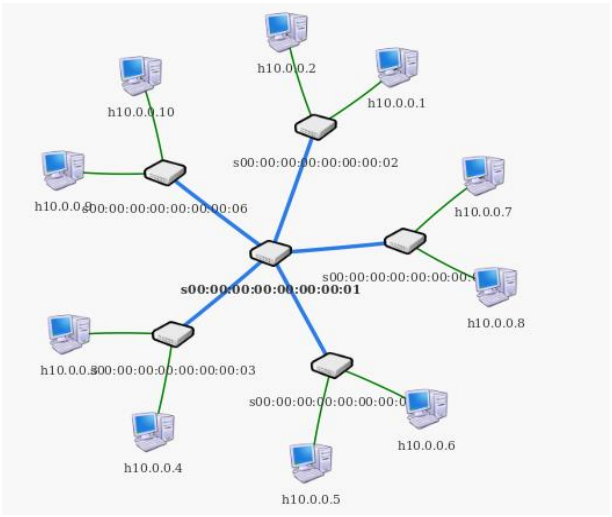
```
net = Mininet(topo=mn_topo, link=TCLink,
              controller= OVSController)
net = Mininet(topo=mn_topo, controller=RemoteController,
               link=TCLink)
fController=net.addController(name='floodlightController'
                             ,controller= RemoteController
```

Received 6 May 2019; Accepted 5 September 2019

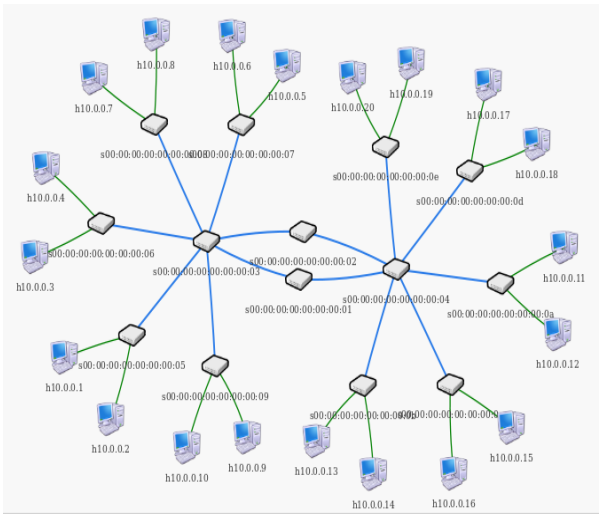
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

```
,ip='127.0.0.1',port=6653)
```

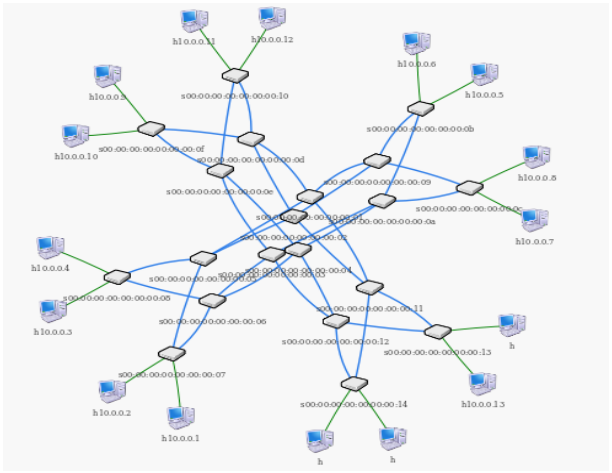
The complete SDN topologies can be now run and tested to evaluate tested resources adaptively according to proposed solutions. Selected topologies executed with OVS controller can be drawn using *networkx* library; while those executed with Floodlight controller can be exhibited in both *networkx* and web UI by the REST API facility which shows a more flexible graphs, as shown in Fig. 2. It is important to mention that according to our hardware properties, we have discovered that OVS controller can implement and run only one-core and two-level DC topologies while Floodlight controller can implement and run different number of cores and levels for the same topologies. We design and implement one, two, three, and five cores for both two and three tiers, two and four levels for fat tree DC topologies, and dumbbell topology with two values for nodes in each bell (four and ten) and for nodes in the path (five and ten).



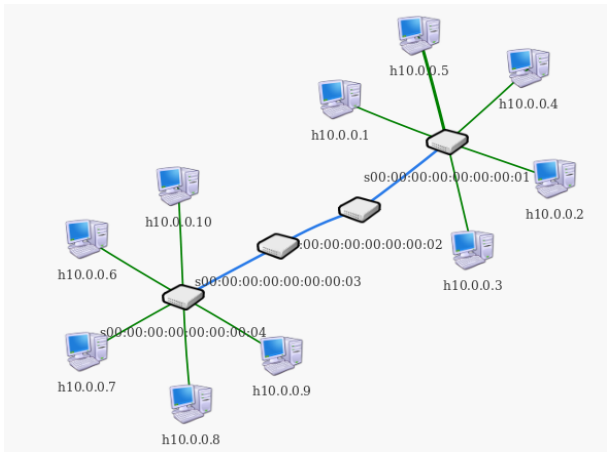
2-Tier 1-core Topology



3-Tier 2-core Topology



4-level Fat-Tree Topology



Dumbbell (5,4) Topology

FIG. 2: DIFFERENT EXAMPLES OF IMPLEMENTED TOPOLOGIES

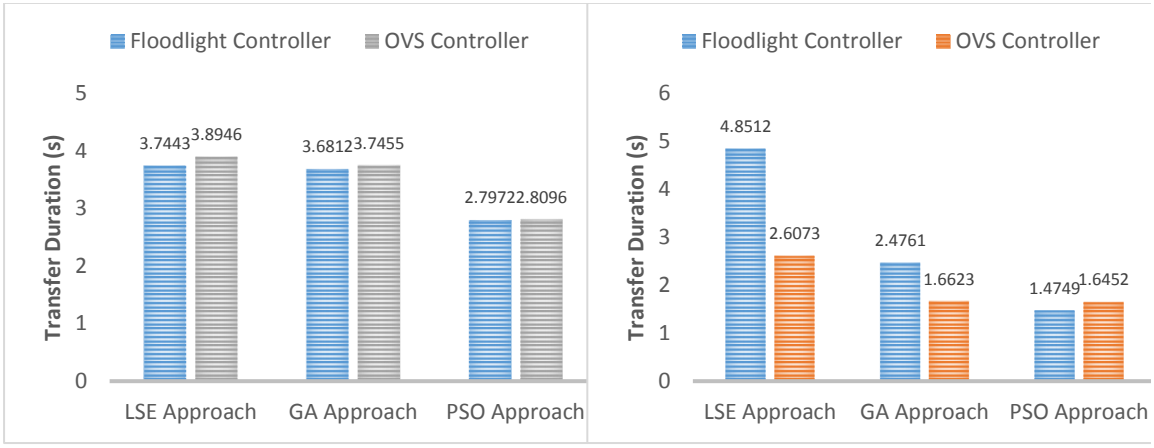
The throughput results of implemented scenarios have been improved from to 11.82Mbps-14.22Mbps and to 11.9Mbps- 13.583Mbps, to 119.2Mbps-128.96Mbps and to 110.425Mbps-114.158Mbps, and to 201.8Mbps-209.6Mbps and to 216.417Mbps-225.583Mbps in Floodlight and OVS controllers respectively and for the statistical, GA, and PSO approaches, respectively. File transfer duration (Fig. 3) between sender and receiver (measured in seconds) has been computed according to the following formula:

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

$$Transfer\ Duration = \frac{Total\ File\ Size}{Actual\ File\ Transfer\ Rate} \quad (8)$$

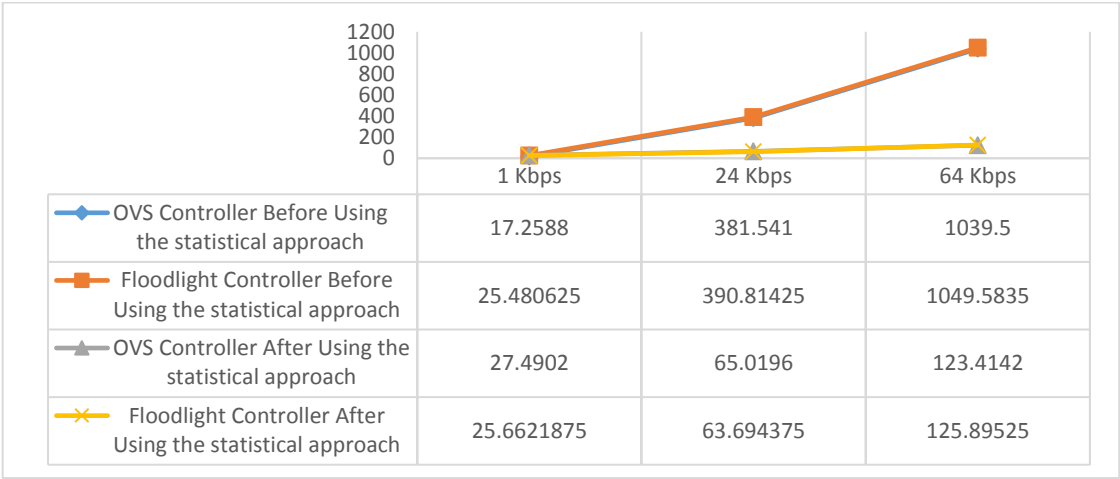
Where, total file size is the file size in bytes (referring to Table 1) and the actual file transfer rate is in bytes per second.



(a) transfer duration for the video 1 (b) transfer duration for the video 2

FIG. 3: TRANSFER DURATION RATE FOR TESTED VIDEO STREAMS

According to collected data, the average mean packet delay was reduced by 85.31% in floodlight controller and 86.46% in OVS controller in the statistical approach as shown in Fig. 4, while it was 91.42% in OVS controller and 91.58% in floodlight controller as shown in Fig. 5, and finally, it was 95.48% in OVS controller and 95.52% in floodlight controller as shown in Fig. 6. The enhancement of the proposed solutions on tested video streams is shown in Fig. 7.

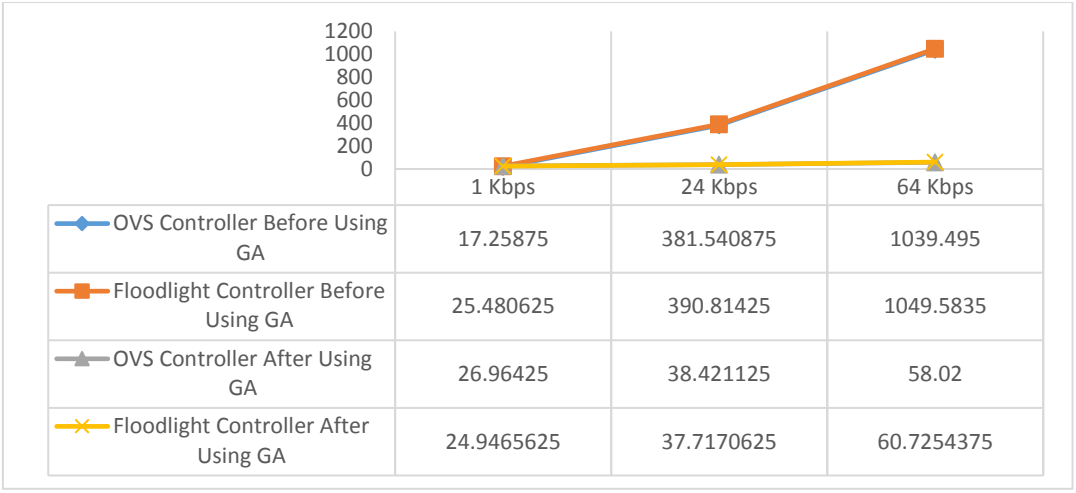


	1 Kbps	24 Kbps	64 Kbps
OVS Controller Before Using the statistical approach	17.2588	381.541	1039.5
Floodlight Controller Before Using the statistical approach	25.480625	390.81425	1049.5835
OVS Controller After Using the statistical approach	27.4902	65.0196	123.4142
Floodlight Controller After Using the statistical approach	25.6621875	63.694375	125.89525

FIG. 4: MEAN PACKET DELAY OF THE STATISTICAL APPROACH USING BOTH OVS AND FLOODLIGHT CONTROLLERS

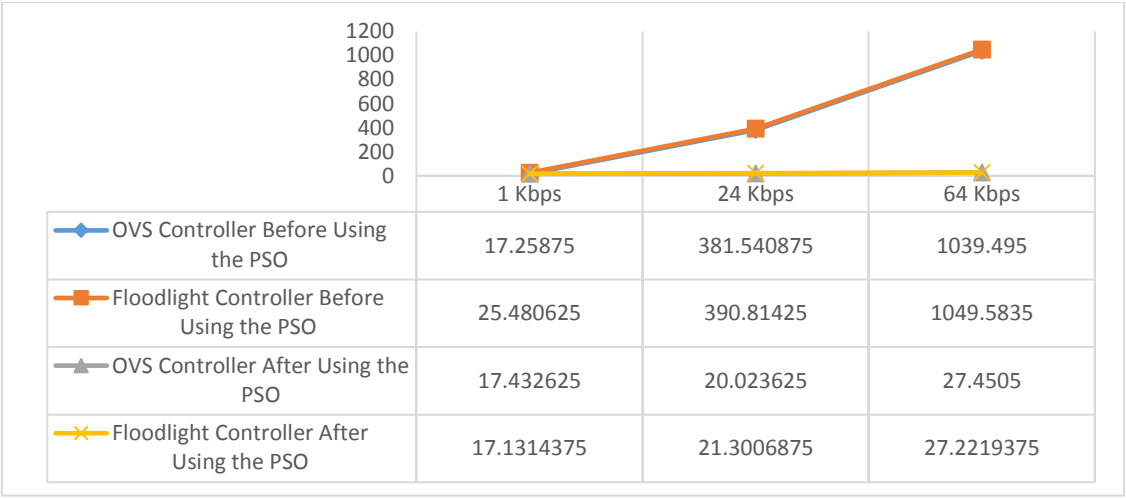
Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352



	1 Kbps	24 Kbps	64 Kbps
OVS Controller Before Using GA	17.25875	381.540875	1039.495
Floodlight Controller Before Using GA	25.480625	390.81425	1049.5835
OVS Controller After Using GA	26.96425	38.421125	58.02
Floodlight Controller After Using GA	24.9465625	37.7170625	60.7254375

FIG. 5: MEAN PACKET DELAY FOR GA USING BOTH OVS AND FLOODLIGHT CONTROLLERS



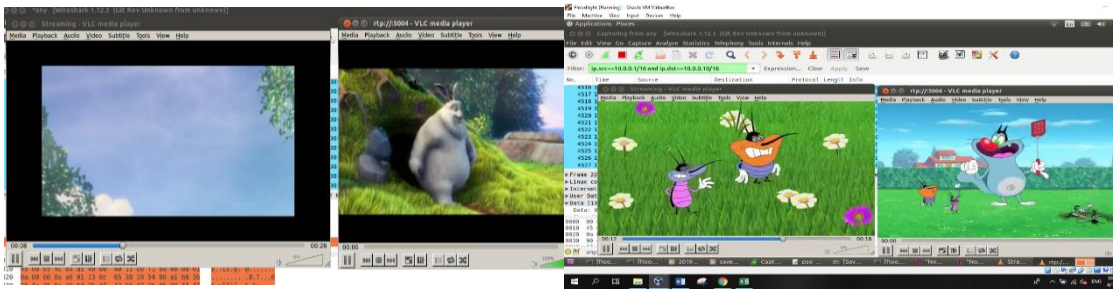
	1 Kbps	24 Kbps	64 Kbps
OVS Controller Before Using the PSO	17.25875	381.540875	1039.495
Floodlight Controller Before Using the PSO	25.480625	390.81425	1049.5835
OVS Controller After Using the PSO	17.432625	20.023625	27.4505
Floodlight Controller After Using the PSO	17.1314375	21.3006875	27.2219375

FIG. 6: MEAN PACKET DELAY FOR PSO USING BOTH OVS AND FLOODLIGHT CONTROLLERS

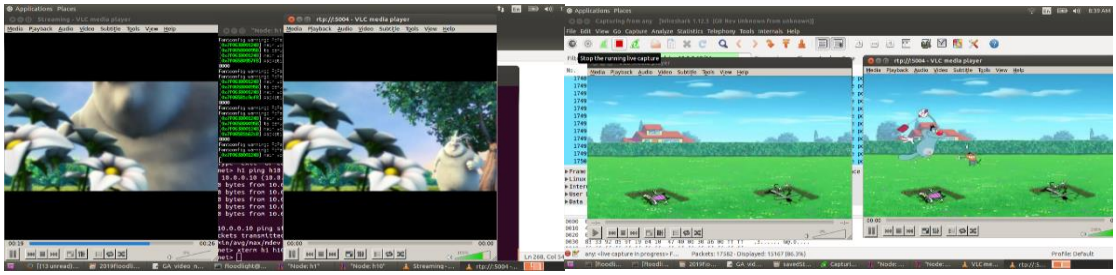
(a) Video 1 before implementing enhancement Video 2 before implementing enhancement

Received 6 May 2019; Accepted 5 September 2019

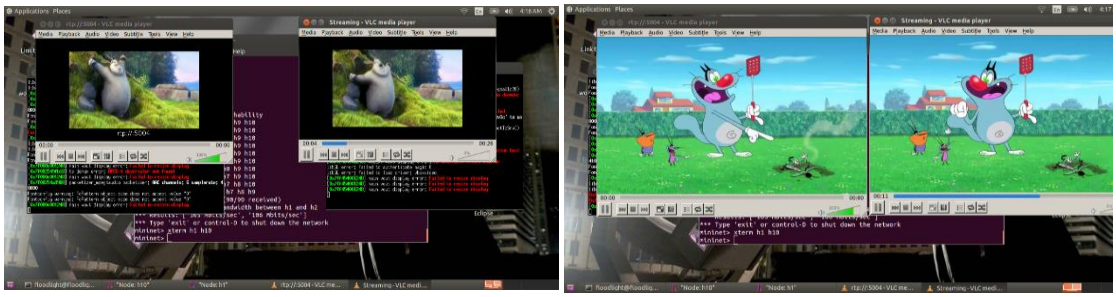
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352



(c) Video 1 after implementing the statistical approach (d) Video 2 after implementing the statistical approach



(e) Video 1 after implementing GA approach (f) Video 2 after implementing GA approach



(g) Video 1 after implementing PSO approach (h) Video 2 after implementing PSO approach

FIG. 7: PERFORMANCE EVALUATION FOR TWO TESTED VIDEO STREAMS

V. CONCLUSION

Two animation videos have been used in this research to test the performance of the proposed solutions. These videos were the Big Buck Bunny with a total file size of 2.1 Mbytes, resolution 320x240 and its length is 26 seconds, the frame rate is 15, and the same video with a total file size of 5.2 Mbytes, frame rate 25, resolution 640x480 and its length is 30 second. The other test video was a sample animation video with a total file size of 1.8 Mbytes, resolution 640x360, frame rate 25 and its length is 18 seconds. The test on the set of the chosen topologies was made by sending these videos between two hosts and capturing the performance of each approach. The VLC media player was the application used to display the tested videos on both hosts. The purpose of this research is to investigate resource management by improving QoS metrics (increasing the capacity of links and decreasing delay values). Three approaches have been proposed and implemented according to the algorithms stated in section 3. They were tested on four different topologies which are (2-tier, 3-tier, and fat-tree) DC and dumbbell topologies. Ping and Iperf commands have been used to record delay and bandwidth values and Wireshark application was used to capture packet delivery ratios and average packets per second for the streamed video between hosts. The floodlight controller has more appropriate graphics and display capabilities than that in the OVS controller. On the other hand, the performance of reducing the delay of the statistical approach for both controllers is 85.88% as

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

an average, while the performance of reducing the delay in the GA approach for both controllers is 91.5%, and lastly, it is 95.5% for PSO approach. Finally, the throughput in both controllers has been improved from 951 Kbps- 1.77 Mbps to 11.68 Mbps-13.901 Mbps in the statistical approach, and to 114.812Mbps- 121.559Mbps in the GA approach, while it has been increased to 209.108Mbps- 217.591Mbps in the PSO approach as an average. The error rates in terms of packet delivery ratio of Floodlight and OVS controllers were 6.71% and 3.42%, respectively.

REFERENCES

[1] W. Xia, Y. Wen, and C. Heng Foh, "A Survey on Software-Defined Networking", IEEE Communication Surveys & Tutorials, Vol. 17, NO. 1, First Quarter 2015.

[2] A. Salman Dawood, and M. Najm Abdullah, "A Survey and a Comparative Study on Software-Defined Networking", International Research Journal of Computer Science (IRJCS), ISSN: 2393-9842, Issue 08, Volume 3 (August 2016)

[3] M. Najm Abdullah, A. Salman Dawood, and A. Kamal Taqi, "Network Resource Management Optimization Based on Statistical Approach", International Journal of Computer Applications (0975 – 8887), Volume 177 – No.6, November 2017.

[4] M. Najm Abdullah, and A. Salman Dawood, "Novel Approach Based on Genetic Algorithm for Adaptive Resource Management in Software-Defined Networks", International Journal of Advanced Research in Computer and Communication Engineering, ISO 3297:2007 Certified, Vol. 6, Issue 1, January 2017.

[5] P. May Thet, P. Panwaree, J. Won Kim, and C. Aswakul, "Design and Functionality Test of Chunked Video Streaming Over Emulated Multi-Path OpenFlow Network", 978-1-4799-7961-5/15©2015 IEEE.

[6] C. Xu, B. Chen, and H. Qian, "Quality of Service Guaranteed Resource Management Dynamically in Software Defined Network", ©2015 Journal of Communications, doi:10.12720/jcm.10.11.843-850.

[7] H. Peng, Q. Ye, and X. Shen, "SDN-Based Resource Management for Autonomous Vehicular Networks: A Multi-Access Edge Computing Approach", arXiv:1809.08966v1, Cornell University.

[8] S. Agliano, M. Ashjaei, Moris Behnam, and L. Lo Bello, "Resource management and control in virtualized SDN networks", 2018 IEEE Conference, DOI: 10.1109/RTEST.2018.8397078.

[9] L. Liao and V. C.M Leung, Genetic Algorithms with Particle Swarm Optimization based Mutation for Distributed Controller Placement in SDNs, 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN).

[10] F. Ongaro, "Enhancing Quality of Service in Software-Defined Networks", Department of Computer Science and Engineering Master Degree in Computer Engineering, 2013-2014.

[11] C. Xu, B. Chen, and H. Qian, "Quality of Service Guaranteed Resource Management Dynamically in Software Defined Network", Journal of Communications Vol. 10, No. 11, November 2015.

[12] A. Khanna, A. Mishra, V. Tiwari, and P.N. Gupta, "A Literature Based Survey On Swarm Intelligence Inspired Optimization Technique", International Journal of Advanced Technology in Engineering and Science, Volume No 03, Special Issue No. 01, March 2015.

[13] J. Galán-Jiménez, "Minimization of Energy Consumption in IP/SDN Hybrid Networks using Genetic Algorithms", 978-3-901882-99-9 c 2017 IFIP.

[14] A. Sabih, Y. Al-Dunainawi, H. S. Al-Raweshidy, and M. F. Abbod, "Optimisation of Software-Defined Networks Performance Using a Hybrid Intelligent System", Advances in Science, Technology and Engineering Systems Journal, ISSN: 2415-6698, Vol. 2, No. 3, 617-622 (2017).

[15] L. Lingxia, V. C. M. Leung, and L. Chin-Feng, "Evolutionary Algorithms in Software Defined Networks: Techniques, Applications, and Issues", ZTE COMMUNICATIONS, August 2017 Vol.15 No. 3.

Received 6 May 2019; Accepted 5 September 2019

 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352