PERFORMANCE EVALUATION USE CASES FOR VIDEO AND VoIP TRAFFIC USING RYU CONTROLLER AND MININET FRAMEWORK-PART II

Pantelimon-Teodor TIVIG[1], Borcoci EUGEN[2]

SDN represents a novel and very versatile technology to cope with the future of networking challenges, tremendous traffic increase and also to support the developments of customizable multi-tenant, multi-domain and slicing networks. Open issues are still present, related with interoperability, experimental aspects, architectural, smart bandwidth allocation for content related networks, optimal controller placement, manual configuration, and programmability. This paper achieves couple of experiments, testing QoS parameters like (delay, jitter) for video streaming, and VoIP, highlighting possible issues in delivering of all types of traffic between source and subscribers, which finally impact the quality-of-experience (QoE) metrics.

Keywords: Software Defined networking (SDN), Quality of Service (QoS), Distributed Internet Traffic Generator(D-ITG), Quality-of-experience (QoE)

1. Introduction

This paper is an extended version of the work [1] and is devoted to further expand, with several use cases experiments, testing QoS parameters, by using D-ITG for synthetic traffic generation, into the SDN architecture and Mininet, to instantiate the network topology [1].

The purpose of the experiments is to obtain data measurements, related to quality parameters for voice traffic, involving usage scenarios with particular voice codecs and with different header techniques and scenarios for video streaming transmission on demand and also to identify possible congested points, when many customers compete, for the same video content. The final purpose is to use the collected data and machine learning techniques in future experiments, which develop custom network policies, that can be applied to specific network parts, with the scope to alleviate any congestion or buffering scenario, that could impact the user experience as a result of voice and video intense utilization.

[1] Prof., Doctoral School of ETTI, University POLITEHNICA of Bucharest, Romania, e-mail: eugen.borcoci@elcom.pub.ro

2 PhD Student, Doctoral School of ETTI, University POLITEHNICA of Bucharest, Romania, e-mail: pantytivig@yahoo.com.

Traditionally the control and data functionalities are placed into an individual router, and the complexity of control plane increases exponentially, as we have more and more routers, with standalone embedded control plane, for each network equipment [2] [3]. SDN architecture is comprised of three-layered, which are: Application layer, Control layer, and Infrastructure layer. The application layer is running the application on physical or virtual network equipment [4]. The Control layer represents the central point of the network and is fulfilled from hardware components, connected to a controller; the hardware represents the network equipment [5].

The need for collaboration, entertainment and well-being has driven to a wide adoption of multimedia devices. The conclusion is that video services are responsible for generating an important percentage of Internet traffic and it is imperative to determine traffic statistics as bit rate from the customers, so that the network and video service providers can better face the nowadays demands, for higher levels regarding bandwidth and Quality of Experience (QoE) [6][7]. A report that shows the effect of bit rate related to 4K video is Cisco annual report, that mentions that the effect at about 15 to 18 Mbps are more than double the HD video bit rate and nine times more than Standard-Definition (SD) video bit rate [1] [8].

The remainder of the paper is divided as follows; the literature of the domain is presented in section 2. Section 3 explains the D-ITG traffic generator, section 4 explains the methodology, section 5 the experimental results and final section 6 concludes the paper.

2. Literature review

The authors in [9] shape the traffic for every client at a fixed value, evaluating the traffic shaping performance, when there is a competing for video flows, over a bottleneck link using SDN. This approach is inappropriate, due to the fact that when the number of clients expands, the available bandwidth will be beneath the required bandwidth. The results revealed that individual traffic shaping provides improved results comparing with the aggregate traffic shaping. An ISP can use QoS/QoE mapping and adjust the application to assess the user experience proposed [10], after the user consumed the video content, by making adjustments if it is necessary.

3. Software traffic generators involved in the study- D-ITG

Distributed Internet Traffic Generator (D-ITG) (see Fig. 1) [11] is a software that generates traffic, which precisely complies with the patterns defined by the interdeparture time between packets (IDT) and by the packet size (PS) stochastic processes. It is an open-source and completely free. This traffic

generator has been built in module to emulate the source of diversified protocols, like the following: VoIP (G.711, G.723, G.729, Compressed RTP and Voice Activity Detection), Telnet TCP, UDP, ICMP, DNS.

4. Methodology

Mininet [12] and OpenVswith [13] is used to instantiate the network topology in the current paper. Many researchers assess the functionality of the SDN control, using the Mininet platform, for emulating OpenFlow switches, host and links, achieving fast prototyping. To define the network topology presented in Fig. 2, the Python [14] code from below Fig.1 was involved. The goal is to determine the performance statistic parameters, for QoS regarding voice and video, including latency and jitter, by using the D-ITG traffic generator a custom network topology was created with depth=3 and every leaf switch has three hosts each.

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Controller,
RemoteController, OVSController
from mininet.node import CPULimitedHost, Host,
Node
from mininet.node import OVSKernelSwitch,
UserSwitch
from mininet.node import IVSSwitch
class CusomTopo( Topo ):
   "Minimal topology with a single switch and two
hosts"
   #Creating switches and links to connect them
   def build( self ):
      for a1 in range(1):
         a1 = self.addSwitch('s1')
      for a2 in range(1):
         a2 = self.addSwitch('s2')
         self.addLink(a1, a2)
      for a3 in range(1):
         a3 = self.addSwitch('s3')
         self.addLink(a3, a1)
      for a4 in range(1):
         a4 = self.addSwitch('s4')
         self.addLink(a1, a4)
      for a5 in range(1):
         a5 = self.addSwitch('s5')
      for a6 in range(1):
         a6 = self.addSwitch('s6')
         self.addLink(a5, a6)
      for a7 in range(1):
         a7 = self.addSwitch('s7')
         self.addLink(a5, a7)
      #Host defining and connecting to each switch
      for h1_ in range(0,3):
         h1 = self.addHost('h1_%s' %(h1_+1))
         self.addLink(a3, h1)
      for h2_ in range(0,3):
         h2 = self.addHost('h2_%s' %(h2_+1))
         self.addLink(a4, h2)
      for h3_ in range(0,3):
         h3 = self.addHost('h3_%s' %(h3_+1))
         self.addLink(a6, h3)
      for h4_ in range(0,3):
         h4 = self.addHost('h4_%s' %(h4_+1))
         self.addLink(a7, h4)

topos = {
   'minimal': CustomTopo
}
```

Fig. 1. Python script for generating the proposed architecture

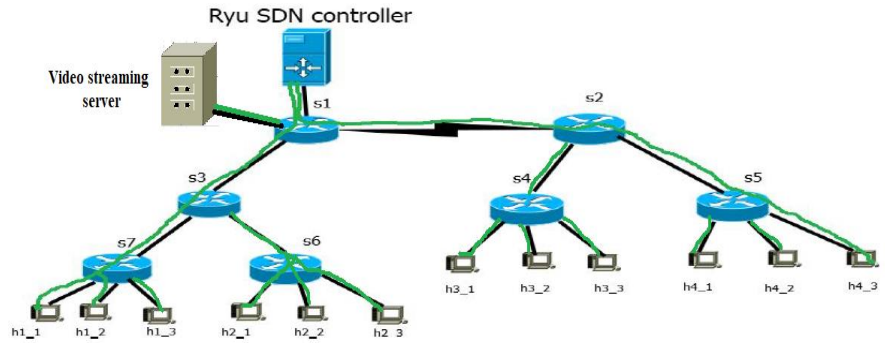

Fig. 2 The network topology

The architecture has 6 switches and 12 hosts emulated by using Linux Namespaces [15] connected to Ryu. The code of the Python script for topology instantiating is provided above, in Fig. 2. In the next subsection there will be conducted video and VoIP traffic test, using D-ITG tool.

5. Traffic Tests using D-ITG

The D-ITG feature that we will use here, in the mentioned network topology, is the capability to generate variable length traffic size.

I. UDP tests

Before we start it is performed a verification of the application, into Fig. 3.

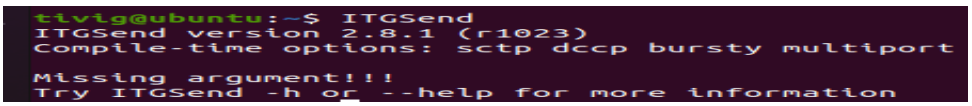

Fig. 3. Verification D-ITG if is properly installed

ITGRecv command for receiver
ITGSend command for sender

A. Staring the experiment

The D-ITG experiment will start on h4_1, server side, Figs. 4 and 5.

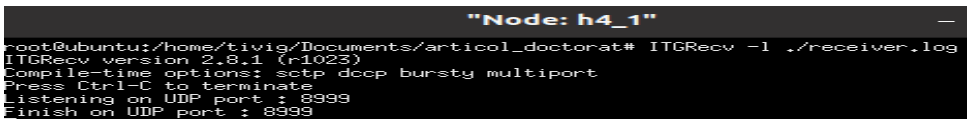

Fig. 4. Starting the D-ITG

a. Start the sender

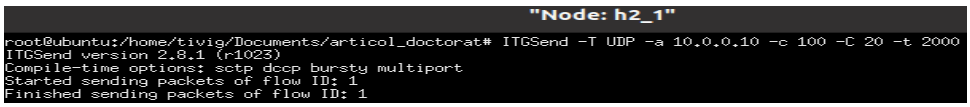

Fig. 5. TCP and UDP traffic flow

The options are explained below:
-T UDP Protocol -UDP
-a 10.0.0.10 destination ip
-t 2000 represents the test duration in milliseconds (20 seconds)

b. Verify the recorded flows

c. Find out the result

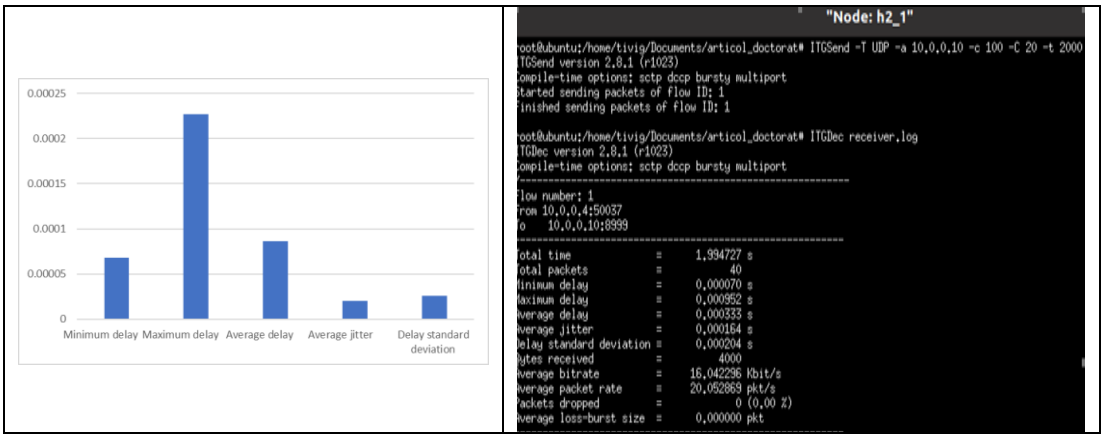

Fig. 6. Traffic results

The obtained result is structured into two parts, as can be depicted in Fig. 6. The first part of the result is represented by flow-specific information, concerning UDP flow and is mapped with FLOW Number. The second part of the result it is represented by a summary of the received information.

d. Multiflow UDP test with 3 flows

For this test we created a script file with incremental packet size rate. The cript is presented into the Fig. 7.

```
-T UDP -a 10.0.0.11 -c 80    -C 30  -t 10000
-T UDP -a 10.0.0.11 -c 400   -C 20  -t 15000
-T UDP -a 10.0.0.11 -c 1024  -C 10  -t 25000
```

Fig. 7. Packet rate variation data

With the script we generate 3 flows:
1st flow generate 100 pkts/per second, having packet size 1024 byte for 1 seconds.
2nd flow generate 80 pkts/per second, having packet size 900 bytes for 15 seconds.
3rd flow generate 60 pkts/per second, having packet size 600 bytes for 20 seconds.

e. Start the receiver on h4_2 (see Fig. 8)

```
root@ubuntu:/home/tivig/Documents/articol_doctorat# ITGRecv -l ./received.log
ITGRecv version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Press Ctrl-C to terminate
Listening on UDP port : 8999
Finish on UDP port : 8999
Finish on UDP port : 8999
Finish on UDP port : 8999
Finish on UDP port : 8999
```

Fig. 8. Starting the receiver

The log file received.log is stored locally and can be decoded via ITGDec.
f. Start the sender on h2_2 (see Fig. 9), using the data from the Fig. 7

```
root@ubuntu:/home/tivig/Documents/articol_doctorat# ITGSend itgscript.sh
ITGSend version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Started sending packets of flow ID: 3
Started sending packets of flow ID: 1
Started sending packets of flow ID: 2
Finished sending packets of flow ID: 1

Finished sending packets of flow ID: 2

Finished sending packets of flow ID: 3
```

Fig. 9. Starting the sender

g. Verify the flow

We can see that there are 3 flows, each represents individual flow we sent, as can be notice from Fig. 10.

II. TCP Test with D-ITG

D-ITG TCP test provides more granular control, we can program the Number of packets/s, Packet Size and more.

a. Start the server on h3_2 (see Fig. 11)

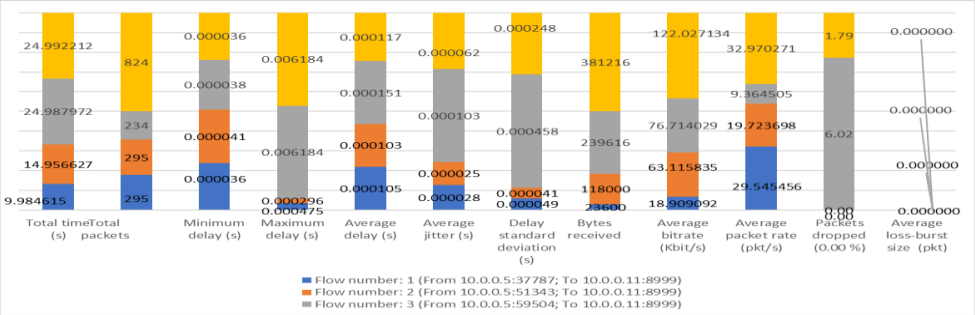

Fig. 10. Result checked from individual flows

```
"Node: h3_2"
root@ubuntu:/home/tivig/Documents/articol_doctorat# ITGRecv -l ./receiver.log
ITGRecv version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Press Ctrl-C to terminate
Listening on TCP port : 8999
Finish on TCP port : 8999
Listening on TCP port : 8999
Finish on TCP port : 8999
```

Fig. 11. Starting of TCP Test with D-ITG, receiver side

b. Start the sender on h1_2 (see Fig. 12)

```
"Node: h1_2"
root@ubuntu:/home/tivig/Documents/articol_doctorat# ITGSend -T TCp -a 10.0.0.8 -t 16000
ITGSend version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Started sending packets of flow ID: 1
Finished sending packets of flow ID: 1

root@ubuntu:/home/tivig/Documents/articol_doctorat# ITGSend -T TCP -a 10.0.0.8 -c 2000 -C 150 -t 16000
ITGSend version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Started sending packets of flow ID: 1
Finished sending packets of flow ID: 1
```

Fig. 12. Starting of TCP Test with D-ITG, sender side

c. Verifying the result (see Fig. 13)

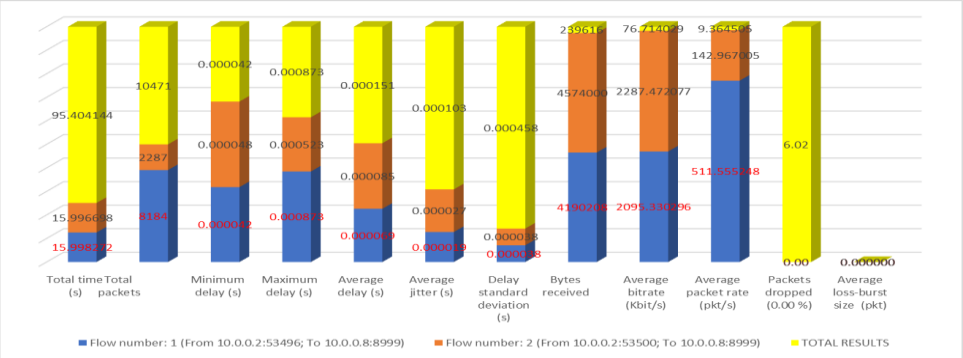

Fig. 13. TCP Test with D-ITG result check

d. Packet distribution (Size and Rate)

Inter-departure time option (Time factor)

(i) Constant distribution (see Fig. 14)

- C rate Constant

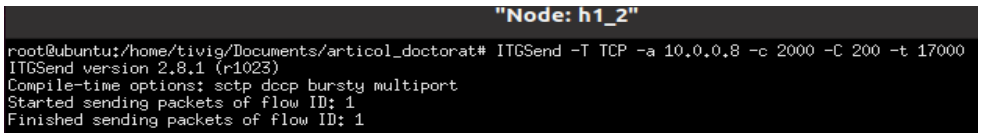

Fig. 14. TCP test, constant distribution

(ii) Uniform distribution (see Fig. 15)

Fig. 15. TCP test, uniform distribution

e. Poison distribution (see Fig. 16)

Fig. 16. TCP test, poison distribution

f. Packet size variation (see Fig. 17)

-c packet size Constant (512 bytes are default),

-u minimum packet size maximum packet size Uniform distribution,

-o mean Poisson distribution.

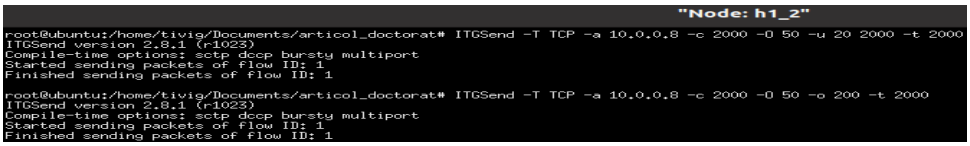

Fig. 17. TCP test, packet size variation

g. Results analysis (see Fig. 18)

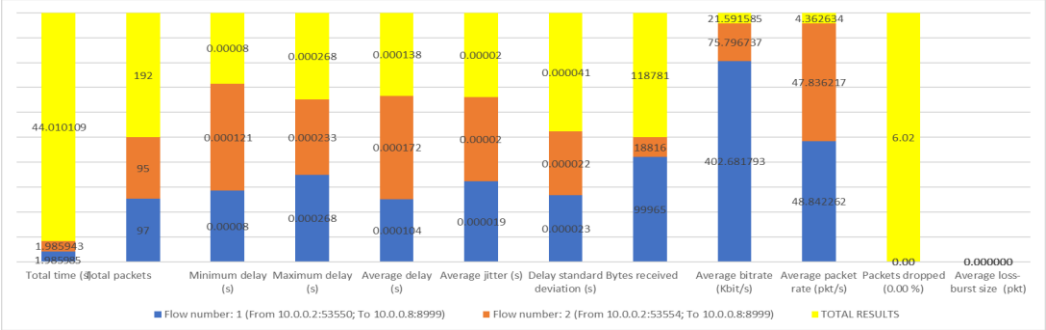

Fig 18. TCP test, packet size variation results

III. VoIP Tests

The performance for VoIP is analyzed using voice codecs G711.1, G723.1 and G729.2 with RTP and CRTP header techniques. The performance of this type of traffic is analyzed based on the parameter's latency and jitter. We will conduct tests-based by hope distance: 1 hop, 2 hops and 3 hops. Will start with one hope distance, where the client is located on h1_1 and the other end is located on h0_1. In the 2-hop configuration, the server remains located at h1_1 and the client connects at h2_1.

A. Latency

The latency pattern adopting CRTP scheme, as can be seen from Fig. 20 is different from RTP scheme Fig. 19. It can be noticed the improvement in latency for 1 hope distance, but also in the case of two hops away, when using CRTP scheme for all mentioned voice codecs. The average delay decreases with the number of hops, but there is a slight difference between the value for two hop distance. The improvement is due to the CRTP technique, that uses a compression scheme, compared with RTP header process. The compression technique of the CRTP header process, makes the G729.2 to obtain the smallest value of the average latency.

B. Jitter

The Jitter is defined as the variation in the latency of received packets. From Figs. 21 and 22, the values of average jitter are increasing with the use of

the CRTP scheme, for all voice codecs in the case 1 hop and 2 hops scenario, contrary with the previous two graphs.

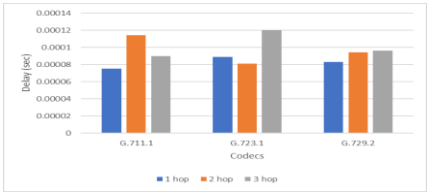

Fig.19. Delay comparation with RTP header

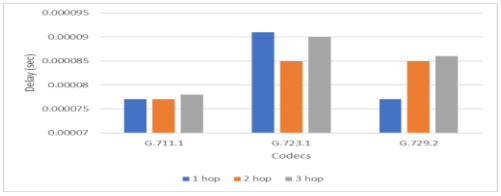

Fig. 20. TCP test, packet size variation

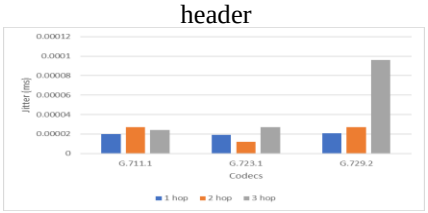

Fig.21. Jitter comparation with RTP header

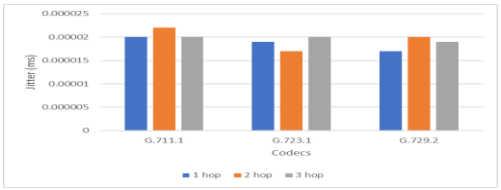

Fig. 22. Jitter comparation with CRTP header

The jitter value for G.711.1 code is slightly the same, into the RTP header test and obviously better, than with the CRTP header scheme. The results proved that the voice quality for G.729.2 is better than for G.711.1 and for G.723.1 in terms of jitter, during data transmission.

IV. Video streaming

The topology used for testing a video streaming scenario, as much closer to a scenario at scale, is the same with the one showed into the Fig. 1. The network is made of: a video server h1_1, eleven video clients (h1_2, h1_3, h2_1, h2_2, h2_3, h3_1, h3_2, h3_3, h4_1, h4_2, h4_3), seven open flow switches (s1, s2, s3, s4, s5, s6 and s7) and the SDN controller. On the server side, for the video streaming test, the VLC media player is used as a video streaming server. On each client side, VLC media player is used with a client role. On the server side of the streaming video, a simple HTTP (Hypertext Transfer Protocol) server bound to port 8080 is used. To view the desired video content, the existing transfer bit rate, which reference at the amount of bit rate, with the chosen resolution without any interference. For instance, YouTube requires 725 Kb/s for 360p and 2.5 Mb/s for 720p. The video animation “Aladdin – Română”, with two resolutions of 1270x720p, 30 MB size and 1906x1080p, 47 MB size, and the duration of 2:56, was used for streaming over Mininet. We used the video code H.264 with encoding bitrate.

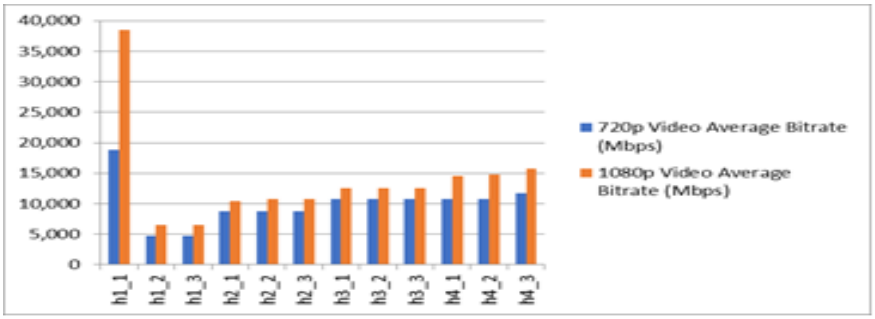

Fig. 23. Video Packet Transmissin with two resolutins

In this test, the HTTP video server hosted on h1_1 sends the video packets to all clients by using TCP mode and the number of packets sent can be seen in Fig. 23. The clients receive traffic streams from the HTTP video server and hosts h1_2, h1_3 connect to the same switch s3, to the video server. Host h1_2, its number of packets is lowest, in the graph. The three clients h4_1, h4_2 and h4_3 are attached to the same switch s5 and traversed three intermediate switches, from the video server to each client, their packet amount is an approximately equal and higher, than the other video clients. Then, the three other hosts, h3_1, h3_2 and h3_3 are streaming videos, via the next switch s4, so the number of packets is close to each and the medium range as their switches are located.

Table 1

Comparation of average bit rate

Video Streaming Hosts		720p Video Average Bitrate (Mbps)	1080p Video Average Bitrate (Mbps)
Host	h1_1	18.761	38.459
Host	h1_2	4.758	6.459
Host	h_3	4.765	6.458
Host	h2_1	8.765	10.458
Host	h2_2	8.766	10.766
Host	h2_3	8.767	10.766
Host	h3_1	10.766	12.566
Host	h3_2	10.766	12.566
Host	h3_3	10.766	12.566
Host	h4_1	10.766	14.565
Host	h4_1	10.766	14.764
Host	h4_3	11.766	15.766

The average transmitted bit rate at each host is compared in Table 1. It can be noticed that the rates are different, as their packet transmission amount and also could be dependent on the distance from the server and subscribers. The closest video customers obtain the highest rate. The bitrate is crucial because they determine the video quality, the streaming performance, and the final impact to

the user experience. More bits per second means more information and more information means more details and better-quality results on the screen. The next customers connected to the next switch require intermediate bitrates and the far end video customers obtain the lowest rate.

If we compare the bitrate value for one of the customers, h1_2 closer to the video server, which has 4.758 Mbps with the far end customer h4_4 which receive 11.766 4.758 we can conclude that the lowest value of the h1_2 customer result, in a lower quality of the video, compared with h4_3 which receive a high value for bit rate. One of the reasons for this low value is a congestion on the link, towards h1_2, which could result in degradation, not only of video quality, but also significant packet loss. A congested link is defined by the offers of lower data rate than the required bitrate of present streaming flow, which could result in a buffering experience, at the customer side. To significantly improve the user experience by alleviating the buffering scenario, for a particular client, with other customers competing for the same video content, the bitrate requires to be high in any competition scenario.

6. Conclusions

This paper reports SDN traffic statistics, measuring a couple of QoS parameters such as the number of packets, data transmission rate and the average bit rate in video streaming, using different resolutions. The experiments used RYU SDN controller, D-ITG, Python programing language for instantiating the network topology and Mininet framework. In this work, we assessed the performance of VoIP in multi-hop scenario, by using header techniques like CRTP and RTP. The statistics were made for G.711.1, G.723.1 and G.729.2 voice codecs, analyzing two important parameters: jitter and delay. The recorded values for jitter are low as the recommendation by National Institute of Standards and Technology, which are less than 40ms.

The video experiments revealed important data, regarding bitrate for video streaming competing scenarios, that bitrate decreased values happened, not only due to the long distance between the server and the client and also for a subscriber, placed in the proximity of the server, showing possible congestion and packet loss.

The data for this experiment will be used in future research, involving machine learning technique, to instantiate dynamic network policies and to maximize the bit rate received by each client and also to avoid a bottleneck network traffic, that could affect the user perceived experience.

REFERENCES

[1]. *T. P. Tivig, E. Borcoci,* Performace Evaluation Experiments for Video and VoIP traffic with RYU Controller and Mininet Framework, journal Scientific bulletin of University Politehnica of Bucharest, 2021.

[2]. *H. Babbar and S. Rani,* "Emerging prospects and trends in software-defined networking," Journal of Computational and Theoretical Nanoscience, vol. 16, no. 10, pp. 4236–4241, 2019.

[3]. *A. A. Abdelaziz, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, "*SDN-Based load balancing servicefor cloud servers," IEEE Communications Magazine, vol. 56, no. 8, pp. 106–111, 2018.

[4]. *M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li,* "Software-defined networking: State of the art and research challenges," Computer Networks, vol. 72, pp. 74–98, 2014.

[5]. *C. Chen-Xiao and X. Ya-Bin,* "Research on load balance method in SDN," International Journal of Grid and Distributed Computing, vol. 9, no. 1, pp. 25–36, 2016.

[6]. *P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race,* Towards network-wide QoE fairness using open flow-assisted adaptive video streaming, in Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, ACM, 2013, pp. 15-20.

[7]. *P. Panwaree, J. W. Kim and C. Aswakul,* Packet Delay and Loss Performance of Streaming Video over Emulated and Real OpenFlow Networks, 29th International Technical Conference on Circuit/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, July 1-4, 2014, pp. 777- 779.

[8]. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual- internet-report/white-paper-c11-741490.html

[9]. *J. J. Quinlan, A. H. Zahran, K. Ramakrishnan, and C. J. Sreenan,* "Delivery of adaptive bit rate video: balancing fairness, efficiency and quality," in Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on IEEE, 2015, pp. 1-6.

[10]. *Hsu et al.,* "The Implementation of a QoS/QoE Mapping and Adjusting Application in software-defined networks", 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG), 2016.

[11]. http://traffic.comics.unina.it/software/ITG/manual/.

[12]. http://mininet.org/.

[13]. https://www.openvswitch.org/.

[14]. https://www.python.org.

[15]. https://man7.org/linux/man-pages/man7/namespaces.7.html.