Enhancing QoE for Video Streaming Considering Congestion: A Fault Tolerance Approach

Baris Yamansavascilar
Computer Engineering
Bogazici University
Istanbul, Turkey
baris.yamansavascilar@boun.edu.tr

Ahmet Cihat Baktir
Computer Engineering
Bogazici University
Istanbul, Turkey
cihat.baktir@boun.edu.tr

Atay Ozgovde
Computer Engineering
Galatasaray University
Istanbul, Turkey
aozgovde@gsu.edu.tr

Cem Ersoy
Computer Engineering
Bogazici University
Istanbul, Turkey
ersoy@boun.edu.tr

***Abstract*—Since tremendous amount of traffic is generated in modern networks as a result of the mobility and video streaming, the congestion issue is faced more frequently in the networks. Accordingly, failures and performance losses in networks due to congestion result in deteriorated Quality of Experience (QoE) from the end user perspective that may cause financial and reputation loss for the service provider. Even though the new video streaming paradigm, Dynamic Adaptive Streaming over HTTP (DASH), is proposed as a solution for the changing condition of the networks, it is not sufficient considering the heavily loaded links that show the symptoms of link failures. Therefore, the flexible implementation of the data plane fault tolerance scheme that can be applied for other problems like congestion in networks is crucial. Thus, in this study, we apply the data plane fault tolerance approach in the Software-Defined Network to improve the QoE of DASH clients in the case of congestion rather than the failure. To detect the congestion in the network level, we use the Bidirectional Forwarding Protocol (BFD) that is originally implemented for link failures. In our experiments, we investigate the effect of the BFD interval, video segment size, and traffic load on QoE parameters. Our results show that if the fault tolerance approach is applied using a small BFD interval with a large segment size, QoE parameters are noticeably enhanced considering the non-applied case.**

***Index Terms*—SDN, Fault Tolerance, Dynamic Adaptive Streaming over HTTP (DASH), Quality of Experience (QoE)**

I. Introduction

Video streaming is the most influential traffic type of the Internet since it occupies 73% of the overall traffic data volume [1]. Moreover, it is expected to occupy 82% of all consumer Internet traffic by 2021 according to the forecast report by Cisco [2]. Apart from its major role in the traffic composition, video streaming also forms a separate category with respect to other applications since it is continuously evaluated by the end user. This evaluation reflects itself in terms of the Quality of Experience (QoE) of the user which is essential for the video service providers since low QoE may result in financial as well as reputation loss for them.

Currently, Dynamic Adaptive Streaming over HTTP (DASH) is the widely accepted technology for video streaming. DASH, in accordance with the stateless character of HTTP, can receive video segments in independent connections, which in turn allows for versatile and dynamic streaming operation. This is in contrast with traditional streaming protocols such as Real-Time Streaming Protocol (RTSP) that work in a stateful manner by tracking the state of the clients during the streaming. Thus, DASH features render video streaming to be carried out in a much more dynamic way so that it provides many advantages including the adaptation to the network conditions [3], [4].

Even though the adaptive nature of DASH provides important flexibility considering the unstable network conditions, its capabilities are insufficient to handle the congestion case. Due to temporary inadequate link capacity, adaptive bitrate feature of the DASH client would not be sufficient to leverage the deteriorating QoE after a serious traffic load. Assuming a continuum of network performance degradation, link failure can be considered as an extreme case of congestion in which delay becomes infinity since they show similar symptoms depending on the severity of the congestion. Bearing with this view, fault tolerance solutions originally developed for link failures can, in fact, serve as candidate methods to deal with congestion when carefully adjusted to this new context.

There are two main approaches for the data plane fault tolerance considering the link failures: (i) restoration (the reactive approach) and (ii) protection (the proactive approach). In restoration, the new rules for the affected flows are computed considering the recent network view, while in protection, the new rules are predefined in case of a failure. Both approaches have advantages and disadvantages regarding their applications on the network [5]. However, considering the congestion problem, the application of the restoration approach is more suitable since a predefined number of affected flows are rerouted rather than all affected flows based on the recent network condition.

Although it seems promising to mitigate congestion using fault tolerance methods, sensitivity of the mechanisms for detecting a data plane fault is also crucial. There are three essential failure detection methods used in networks including Loss of Signal (LoS), Link Layer Discovery Protocol (LLDP), and Bidirectional Forwarding Detection (BFD). Among these methods, the BFD protocol has superiority since it is designed only for failure detection and its detection speed outperforms the others [6], [7]. Thus, its use in a fault tolerant system provides many benefits for both end users considering QoE and service providers to meet the expected service quality.

In this study, our main goal is to apply failure detection and

rerouting methods of the data plane fault tolerance scheme on a congestion case to improve QoE for DASH clients. We use the Software-Defined Networking (SDN) paradigm in our network as its central view provides opportunities for innovative solutions [8]. We investigate the impact of three parameters including the BFD interval, video segment size, and traffic load on the QoE. Thus, main contributions of our study can be summarized as:

1) We detect the congestion in the link layer operating BFD rather than the application layer metrics that is generally used by video applications to adapt themselves for new conditions. Detecting congestion in the network level allows us to implement a global solution as opposed to each client adopting themselves in a reactive manner.
2) After its successful detection, we employ the data plane fault tolerance mechanism to solve the congestion problem for DASH in SDN. As a result, we apply the same treatment for a different problem that shows similar symptoms.
3) We explore the impact of BFD intervals on the QoE parameters along with video segment size to come up with a feasible operational range where fault tolerance approach proves useful for the congestion case.

The rest of this paper is organized as follows. In Section II, we elaborate the related works including DASH, fault tolerance, and congestion studies that use SDN as their networking paradigm. Section III provides the details of our BFD-based congestion detection system. Section IV presents our experimental results. Finally, we conclude our study in Section V.

II. Related Works

SDN offers many benefits to address networking challenges using the OpenFlow protocol [9] that are hard to circumvent with traditional approaches. Accordingly, recent studies working on DASH use SDN as the main networking paradigm in order to exploit the benefits of the centralized controller and, thus, enhance QoE. Mkwawa et al. [10] proposed a video quality management scheme that considers the traffic intensity. They compared the number of stalls of the DASH video streaming when their proposed scheme is in use and not. As expected, the number of stalls were fewer when they used their solution. In [11], Bentaleb et al. focused on QoE unfairness for multiple clients in the network since when the number of clients increases, QoE is unfair because of bandwidth sharing and network resource underutilization. Afterwards, they improved their scheme in [12], considering scalability issues of clients, communication overhead, and client heterogeneity. Likewise, Bagci et al. [13] studied QoE fairness among clients. They used the network slicing concept and manipulated TCP windows to prevent QoE fluctuations.

On the other hand, fault tolerance in SDN data plane has been studied well in the literature. Studies use several methods to provide the communication after the failures based on restoration and protection approaches. In the restoration approach, the common behavior is that studies use the controller for failure notification and then calculate new routes to maintain communication. Kim et al. [14] proposed an SDN fault-tolerant system that recovers from multiple link failures in the data plane. They used VLANs to compute the routing paths. Li et al. [15] on the other hand developed a restoration approach by using a local optimal failover method in order to reduce the path calculation time. Some studies also considered reliability in this manner. In [16], Yuan et al. designed a system based on the Byzantine model to tolerate faulty switches in order to enhance reliability. Moreover, Song et al. [17] focused on the control-path reliability which is an important consideration for out-of-band controllers whose network view can be affected by the data plane failures.

Apart from the restoration approach, some studies used the protection approach in order to provide fast recovery by solving the fault tolerance problem in the data plane without involving the controller. In [18], authors built a system that permits switches to send faulty link information to only the relevant switches in order to prevent traffic flood and increase the network performance in centralized networks. Likewise, Kempf et al. [19] supported a monitoring function for failure detection in the data plane without involving the controller. To this end, they generated monitoring messages within the source switch and process them in another destination switch. In [20], Ramos et al. developed a proactive failure recovery scheme by carrying the information of alternative paths in the packet headers. Thus, when a link failure happens, their system uses the alternative path information in order to maintain communication without consulting the controller. Adrichem et al. [21] used Bidirectional Forwarding Detection (BFD) protocol per link to detect failures and then compared the performance of different BFD detection intervals in terms of milliseconds.

SDN is also used for the congestion case by several studies. In [22], authors modified the TCP receive window of ACK packets at controller in order to avoid the network congestion. To perform this, they deployed a queue management scheme in OpenFlow-switch that notifies the controller when the queue passes the given threshold. Kim et al. [23] on the other hand considered the dynamic changes in the network traffic and proposed their reinforcement learning based technique, Q-learning, for the routing of flows to prevent congestion. Cheng et al. [24] indicated the shortage of ternary content addressable memory (TCAM) of OpenFlow switches that cause bottleneck for scalable flow management. Therefore, they applied flow aggregation using VLANs to prevent congestion for a failure recovery case. In [25], authors carried out a congestion control mechanism in Mobile Edge Computing (MEC) environment using SDN to considering congestion. They classified the network traffic as delay tolerant and delay sensitive so that they buffered the delay tolerant flows in MEC servers during the peak hours in order to prevent congestion.

To the best of our knowledge, this is the first study that applies a fault tolerance approach to improve QoE in case a network-based problem like congestion. Moreover, investigat-

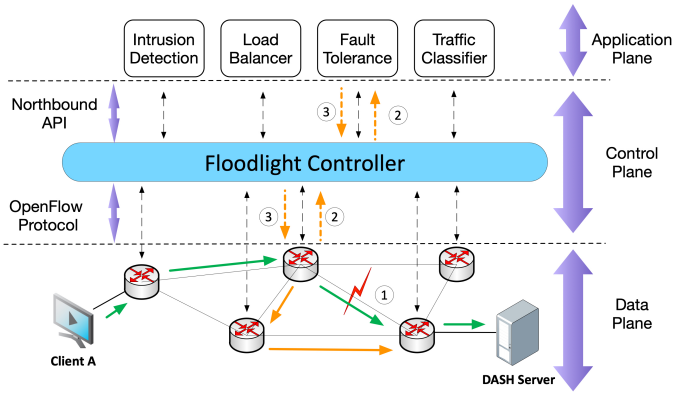


Fig. 1: The design of the BFD-based Congestion Detection System.

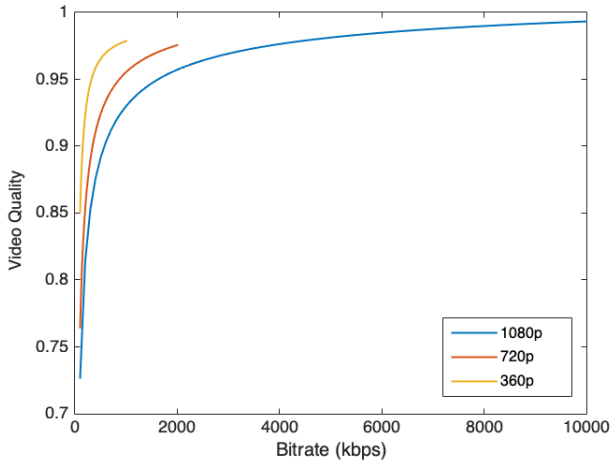


Fig. 2: Relation between video quality and bitrate for three different resolutions.

ing the impact of BFD interval on the QoE distinctly separates this study from existing works.

III. BFD-based Congestion Detection System

We designed our system based on the data plane fault tolerance mechanism applying the restoration approach rather than protection. Since video streaming can resist network changes for seconds due to its buffering mechanism, rerouting flows based on the recent network view would be beneficial. Similar to the fault tolerance problem, our BFD-based Congestion Detection System includes a detection phase and the action phase. However, since there is no need to reroute all affected flows in the congestion problem, it is separated from the fault tolerance. Figure 1 shows the main aspects of our design.

A. Congestion Detection

The first step is the detection of the congestion using the BFD in the case of a heavy traffic load as shown in Figure 1. BFD is run on the switches to which the observed link is connected. These two switches are called as a pair regarding to BFD. Each switch conveys a control packet including the current state of the monitored link to its pair switch. When a switch receives the control message from its pair, it sends an echo message to it with the session status.

The failure detection time, T_d, is based on the BFD message transmit interval, T_i, that can be manually configured by the network administrator considering the given services. For example, if real-time applications such as VoIP are widely used in the network, the interval must be very low considering the 50ms recovery time [6]. On the other hand, if there is multimedia traffic like video streaming, the interval may be in seconds due to the buffer mechanism of the applications. Thus, T_d is computed based on the T_i and the detection time multiplier M as given in Equation (1). M value is used to prevent false positives.

$$T_d = (M + 1) * T_i \quad (1)$$

B. Rerouting Flows

After detecting the congestion using BFD, the problem on the link is reported to the controller via the OpenFlow protocol. Subsequently, the controller informs the modified fault tolerance application about the problem. In the application, flows passing through that link are first extracted from the flow pool in which all active flows are held. Afterwards, the predefined percentage of flows are selected for rerouting and the new rules are created for them. As a result, the congested link is relieved.

C. DASH Client Module

In our system, each DASH client reports their QoE parameters including the bitrate, video quality value, latency, and number of quality switches for the evaluation of the effect of congestion properly. All these parameters except the video quality value are extracted from the DASH.js API [26]. However, obtaining the objective video quality value, which is the most distinctive component of the QoE [4], from the video streaming is required to create an appropriate formula that reflects the human subjective measurement. To perform this, the Structural Similarity Index (SSIM) [27] is widely used.

SSIM is used to measure the similarity between the original and transmitted images so that it is possible to assess the objective image quality and video quality. Since higher bitrates ensure higher similarity, the essential parameter for the perceptual quality value is the bitrate of the video streaming. However, this relationship is not linear as shown in Figure 2 [28]. Therefore, the generalized formula of the perceptual quality is extracted by applying the curve fitting based on the bitrate and resolution values given in Table I. The formula is given in Equation (2) and corresponding coefficients are given in Table II. In the equation, *f(x)* represents the video quality, and *x* variable denotes the bitrate.

$$f(x) = ax^b + c \quad (2)$$

TABLE I: Bitrates for different screen resolutions

Resolution	Bitrate (kbps)
1080p	100, 200, 600, 1000, 2000, 4000, 6000, 8000
720p	100, 200, 400, 600, 800, 1000, 1500, 2000
360p	100, 200, 400, 600, 800, 1000

TABLE II: Coefficients of the generalized video quality function [28]

Resolution	Power Series Model			Goodness of Fit	
	a	b	c	Adjusted R^2	RMSE
1080p	-3.035	-0.5061	1.022	0.9959	0.006011
720p	-4.85	-0.647	1.011	0.9983	0.002923
360p	-17.53	-1.048	0.9912	0.9982	0.002097

IV. Performance Evaluation

We investigated three factors in our experiments: the impact of the BFD interval, the traffic load, and the video segment size on the QoE parameters. For each experiment, we used Mininet [29] for SDN emulation deploying Open vSwitch [30] for switches since it supports both BFD and OpenFlow protocols. On the other hand, we used DASH.js for the video clients. To generate additional traffic and thus cause congestion, we run the iPerf tool [31] on Mininet.

A. The Scenario of Experiments

For each experiment, the topology shown in Figure 3 was used. We limited the capacity of all links as 50Mbps by deploying five DASH clients connecting to Switch 1 and one DASH server attaching to Switch 6. We ensured that all DASH clients obtain the video segments via the same route as indicated in Figure 3 as green arrows. On the other hand, four iPerf clients were connected to Switch 2 to cause congestion by sending their packets to the iPerf server which is located at Switch 5. Moreover, Big Buck Bunny short movie, the duration of which is 10 minutes, is used for streaming the 1080p resolution video. After the streaming was started for each DASH client, T1, T2, T3, and T4 iPerf clients began to send their packets at 50th, 80th, and 110th second respectively to cause congestion. These iPerf clients generated the same amount of UDP traffic and induced congestion on the link between Switch 2 and Switch 5.

We took three parameters as the variable in the experiments: BFD Interval, Traffic Load, and Video Segment Size. We used 1 second and 10 seconds segments to evaluate the effect of the segment size. To observe the impact of the traffic load, we generated 40 Mbps (80% Load), 45 Mbps (90% Load), and 49 Mbps (98% Load) traffic using only the iPerf clients. Finally, to evaluate the influence of the BFD intervals in our system, we used T_i as 100ms and 1000ms respectively by taking M value as two. Thus, the T_d values were 300ms and 3000ms respectively. Moreover, we compared these results with the non-BFD case.

Consequently, we conducted our experiments for 18 different cases considering those three parameters. For each case, we recurred our tests as 6 times. Since each test lasts 10-11 minutes, the duration of our experiments was 18 hours.

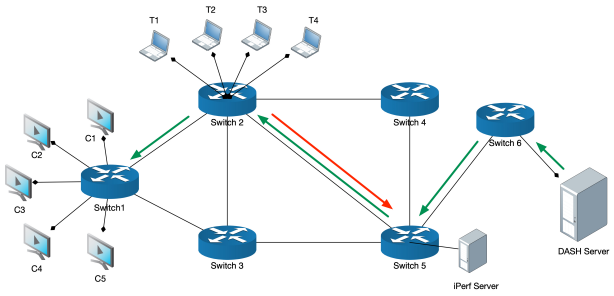


Fig. 3: The topology and video traffic route

To evaluate the results, we analyzed four QoE parameters including the bitrate, quality value, latency and number of quality switches from each client and calculated the average values for each case.

B. The Effect of Segment Size

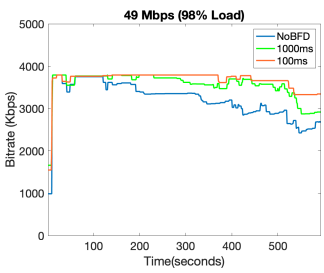
Our experiments showed that streaming with the big segment size is more stable than the small segment size considering the congestion conditions on the link. Experimental results for 49 Mbps (98% Load) traffic on the congested link given in Figure 4 represent that fluctuations and change of the QoE parameters including the average bitrate, video quality, latency, and number of quality switches between representations are higher in 1-sec segment size compared with 10-sec segment size. This pattern is the same for 45 Mbps and 40 Mbps traffic loads. Since a small segment size needs more HTTP requests to transmit video segments, it is affected by the network conditions more than the big segment sizes.

C. The Effect of Traffic Load

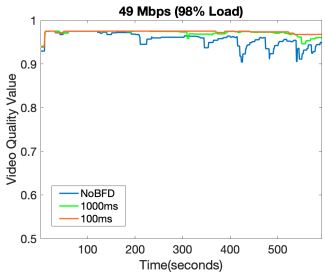
To evaluate the impact of the traffic load, we measured the average video quality of clients based on SSIM and the number of quality switches including all cases in our experiments. Our results demonstrated that when the traffic load increases, the video quality decreases considering both segment sizes with the non-BFD case as shown in Figure 5a and 5b. For each traffic load, the video quality with 10-sec segment size is better than the 1-sec segment size for the non-BFD case due to its buffer capacity. On the other hand, if we use BFD for the congestion detection, the video quality is improved.

Considering the non-BFD case for the 80% and 90% traffic loads, the average video quality with 1-sec segment size is not so affected while the average video quality with 10-sec segment size is decreased. However, for the 98% traffic load, the video quality is poor when we used 1-sec segment size while it is acceptable for 10-sec segment size.

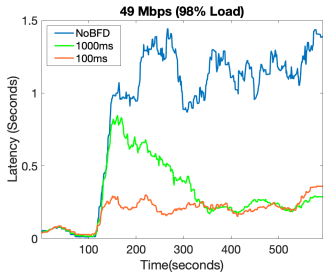
On the other hand, the effect of the traffic load on the number of quality switches is shown in Figure 6a and 6b. The results show that the quality switch count between representations for the 10-sec segment size is the lowest for 80% load and highest for the 90% load when the BFD is not used. This is originated by the fact that 80% load is not heavy for the 10-sec segment size so that the quality change is low, while the quality is affected by the 90% load



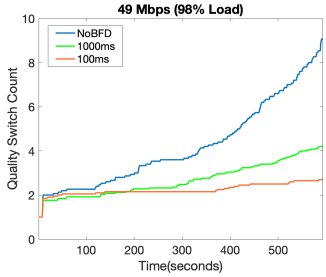
(a) Average bitrate for 10-sec segment size



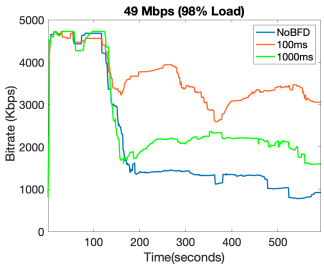
(b) Average video quality for 10-sec segment size



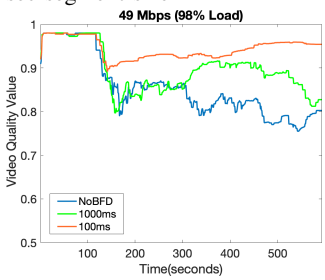
(c) Average latency for 10-sec segment size



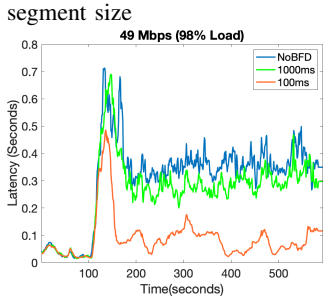
(d) Average number of quality switches for 10-sec segment size



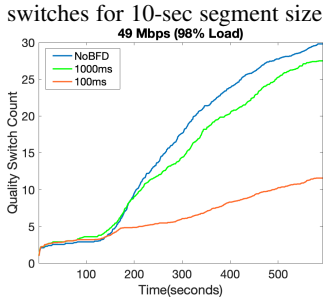
(e) Average bitrate for 1-sec segment size



(f) Average video quality for 1-sec segment size

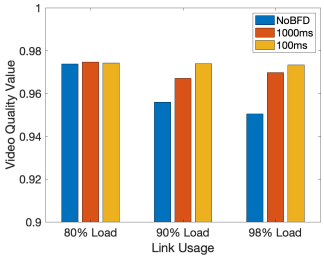


(g) Average latency for 1-sec segment size

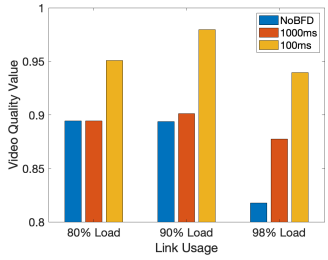


(h) Average number of quality switches for 1-sec segment size

Fig. 4: The change of QoE parameters for congested link with 98% load. The QoE parameters are affected by the congestion after 150th second. If our scheme is used with BFD mechanism, the QoE parameters are improved for each case.

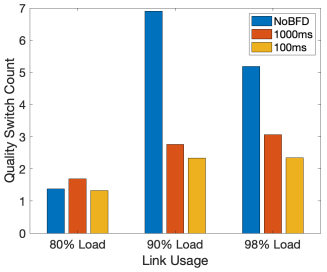


(a) The impact of the traffic load on the average video quality using 10-sec segment size

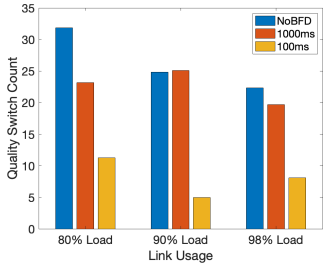


(b) The impact of the traffic load on the average video quality using 1-sec segment size

Fig. 5: The impact of the traffic load on the average video quality with respect to different segment sizes



(a) The impact of the traffic load on the average quality switch count using 10-sec segment size



(b) The impact of the traffic load on the average quality switch count using 1-sec segment size

Fig. 6: The impact of the traffic load on the quality switch count with respect to different segment sizes.

that cause quality switches. Moreover, since 98% load is the most influential for the quality, the quality cannot fluctuate so that the switch count is not higher than the case of 90% load. Besides, considering the 1-sec segment size, the count of switches between representations decrease for higher traffic loads since they cause worse video quality respectively.

D. The Effect of BFD Intervals

Our results showed that the impact of BFD is crucial in the case of congestion. In Figure 4, the results show that using BFD fixes the poor outputs of each QoE parameters after the 150th second at which the traffic load starts to influence video streaming. Moreover, it is clearly observable that T_i with 100ms outperformed T_i with 1000ms regarding QoE parameters since its sensitivity is more delicate so that it detects the congestion earlier. Apart from the 49 Mbps shown in Figure 4, this pattern is also the same for other traffic loads including 45 Mbps (90% load) and 40 Mbps (80% load).

On the other hand, the BFD effect is also noticeable in Figure 5a and 5b considering the video quality based on traffic load. For 10-sec segment size with 80% traffic load, the effect of BFD interval is limited since the traffic could not cause congestion that induces to prevent BFD control packets considering their message interval, T_i. However, for 1-sec segment size with 80% traffic load, T_i with 100ms is affected by the traffic so that the quality is improved. However, considering the 90% and 98% traffic loads, T_i with 1000ms is also affected by the congestion. Moreover, the number of quality switches between representations is reduced when we

use BFD as shown in Figure 6a and 6b. The number of quality switches is the lowest for the T_i with 100ms regarding 10-sec and 1-sec segment sizes.

V. Conclusion

In this study, we applied the data plane fault tolerance approach, restoration, in the SDN to improve the QoE of DASH clients. On the other hand, we used the BFD protocol, which is originally designed to detect failures between network nodes, to detect congestion on the path through which the video flows passing. We investigated the effect of the video segment size, traffic load, and BFD intervals on several QoE parameters that reflect the subjective opinion of the users. We used 1 second and 10 seconds segment sizes; 100ms and 1000ms BFD intervals; 40 Mbps, 45 Mbps, 49 Mbps traffic loads with the capacity of 50Mbps. Our results showed that video streaming with a large segment size is more stable than the small segment size for the congestion case. On the other hand, since the BFD interval with 100ms is more sensitive to the traffic load, it detects congestion earlier than the 1000ms interval so that the output of QoE parameters is better than the latter.

For the future work, we plan to consider the number of DASH clients, quality switch algorithm used in DASH and the percentage of rerouted flows to investigate their effect on QoE parameters in case of congestion.

Acknowledgment

This work is supported by the State Planning Organization of Turkey, under the TAM project with the Grant No. 2007K120610. and the Galatasaray University Research Foundation under the Grant No. 18.401.003

References

[1] C. V. N. Index, "The zettabyte era: Trends and analysis," *Cisco white paper*, 2017, accessed on: June 20, 2018. [Online]. Available: https://bit.ly/2uBPyaa.

[2] ——, "Cisco visual networking index: Forecast and methodology 2016-2021," *White paper, CISCO*, 2017.

[3] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[4] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[5] P. Fonseca and E. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys & Tutorials*, 2017.

[6] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Openflow: Meeting carrier-grade recovery requirements," *Computer Communications*, vol. 36, no. 6, pp. 656–665, 2013.

[7] N. L. van Adrichem, B. J. van Asten, and F. A. Kuipers, "Fast Recovery in Software-Defined Networks," in *2014 Third European Workshop on Software Defined Networks*. IEEE, sep 2014, pp. 61–66. [Online]. Available: http://ieeexplore.ieee.org/document/6984053/

[8] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[9] 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.

[10] I.-H. Mkwawa, A. A. Barakabitze, and L. Sun, "Video quality management over the software defined networking," in *Multimedia (ISM), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 559–564.

[11] A. Bentaleb, A. C. Begen, and R. Zimmermann, "Sdndash: Improving qoe of http adaptive streaming using software defined networking," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 1296–1305.

[12] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "Sdnhas: An sdn-enabled architecture to optimize qoe in http adaptive streaming," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2136–2151, 2017.

[13] K. T. Bagci, K. E. Sahin, and A. M. Tekalp, "Compete or collaborate: Architectures for collaborative dash video over future networks," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2152–2165, 2017.

[14] H. Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner, and N. Feamster, "Coronet: Fault tolerance for software defined networks," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*. IEEE, 2012, pp. 1–2.

[15] J. Li, J. Hyun, J.-H. Yoo, S. Baik, and J. W.-K. Hong, "Scalable failover method for data center networks using openflow," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–6.

[16] B. Yuan, H. Jin, D. Zou, L. T. Yang, and S. Yu, "A practical byzantine-based approach for faulty switch tolerance in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 825–839, 2018.

[17] S. Song, H. Park, B.-Y. Choi, T. Choi, and H. Zhu, "Control path management framework for enhancing software-defined network (sdn) reliability," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 302–316, 2017.

[18] M. Desai and T. Nandagopal, "Coping with link failures in centralized control plane architectures," in *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*. IEEE, 2010, pp. 1–10.

[19] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takács, and P. Sköldström, "Scalable fault management for openflow," in *Communications (ICC), 2012 IEEE international conference on*. IEEE, 2012, pp. 6606–6610.

[20] R. M. Ramos, M. Martinello, and C. E. Rothenberg, "Slickflow: Resilient source routing in data center networks unlocked by openflow," in *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*. IEEE, 2013, pp. 606–613.

[21] N. L. Van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *Software Defined Networks (EWSDN), 2014 Third European Workshop on*. IEEE, 2014, pp. 61–66.

[22] Y. Lu and S. Zhu, "Sdn-based tcp congestion control in data center networks," in *Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance*. IEEE, 2015, pp. 1–7.

[23] S. Kim, J. Son, A. Talukder, and C. S. Hong, "Congestion prevention mechanism based on q-leaning for efficient routing in sdn," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 124–128.

[24] Z. Cheng, X. Zhang, Y. Li, S. Yu, R. Lin, and L. He, "Congestion-aware local reroute for fast failure recovery in software-defined networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 11, pp. 934–944, 2017.

[25] M. Nasimi, M. A. Habibi, B. Han, and H. D. Schotten, "Edge-assisted congestion control mechanism for 5g network using software-defined networking," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2018, pp. 1–5.

[26] "Dash.js," accessed on: December 9, 2018. [Online]. Available: http://cdn.dashjs.org/latest/jsdoc/index.html.

[27] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

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

[29] R. L. S. De Oliveira, A. A. Shinoda, C. M. Schweitzer, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*. IEEE, 2014, pp. 1–6.

[30] "Open vswitch," accessed on: December 16, 2018. [Online]. Available: http://www.openvswitch.org.

[31] "iperf," accessed on: December 9, 2018. [Online]. Available: https://iperf.fr.