Janus - A Software-Defined Networking MPEG-DASH Video Streaming Load Balancer

Edenilson Jônatas dos Passos
Graduate Program in Apllied Computing (PPGCA)
Department of Computer Science (DCC)
Santa Catarina State University (UDESC)
Joinville, Brazil
edenilson.passos@yahoo.com

Adriano Fiorese
Graduate Program in Apllied Computing (PPGCA)
Department of Computer Science (DCC)
Santa Catarina State University (UDESC)
Joinville, Brazil
adriano.fiorese@udesc.br

***Abstract*—Recently, with popularisation of video streaming service, new video distribution technologies have been created. Currently, one of the most promising ones is the Moving Picture Expert Group Dynamic Adaptive Streaming over HTTP or MPEG-DASH. Even so, with the limitation of the TCP/IP network structure, the end user Quality of Experience (QoE) may be affected. One issue that can affect user QoE is the workload of content distribution servers. Thus, the unbalancing of server's workload comprising user's attendance can lead to a content server provider non optimised choice. This work presents two load-balancing solutions between MPEG-DASH video servers based on Software-Defined Networks, using as a balancing workload metric the throughput of the content server as well as the CPU load.**

***Index Terms*—Load balancing, MPEG-DASH, SDN, Flows manipulation**

I. Introduction

The use of the Internet on video transmission plays a relevant role in the business models of the current content providers. This is beneficial to users as it provides great control over when and where content can be viewed. A survey conducted by The Nielsen Company found that out of a total of 30,000 participants from 61 countries, 65% say they consume some Video On Demand content [1]. In addition, according to [2], by the year 2022, video traffic on the Internet is expected to reach 82% of the total.

With the popularisation of streaming services, and consequently the overloading of the links as well as the restrictions imposed on this type of traffic by ISPs, companies began to develop adaptive video formats and content using the HTTP protocol as delivery method. Thus, the standard ISO/IEC 23009-1 or Moving Picture Expert Group Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [3] appeared.

A Content Delivery Network (CDN) is responsible for storing content that customers want as well as routing the requests to the closest server to the client. In this way, a CDN is responsible for load balancing between content servers [4], [5]. In these cases, load balancing is needed since, according to [6], server congestion is the factor that most affects the Quality of Experience (QoE) of the user because it is perceived as interruption and noise in the content's image and audio.

A contemporary approach to load balancing in CDNs for video content delivery is based on serving multiple client requests across multiple servers distributed in the CDN. To do this, the assignment of which server will attend to which client is done by means of the custom deployment of Domain Name Server (DNS) servers by the CDN provider. Thus, the DNS entries are adapted according to the adopted load balancing policy, in order to change the addresses of the servers that will serve the clients, hence performing the load balancing between the content servers. However, large-volume and long-term flows, such as video-on-demand (VoD) traffic, are hardly manageable using this DNS approach. The introduction of Software-Define Networking (SDN) is promising to solve this problem. Particularly, SDN instantiations, such as the one using OpenFlow protocol, allow the transparent redirection of flows in the network without the need for manipulation of infrastructure services [7].

Therefore, along these lines, the present article deals with the proposition of an architecture for load balancing in the MPEG-DASH video distribution network with the aid of SDN to perform the manipulation of necessary flows by means of the protocol Openflow version 1.3 and the Ryu controller. The balancing performed is intended to alleviate the workload of the content source servers and uses a combination of server throughput and CPU utilisation as balancing metric.

This paper is organised as follows. Section 2 discusses related work. Section 3 architecture and operation modes of the proposed solution are detailed. Section 4 presents evaluation of the proposal. Finally, Section 5 presents final considerations comprising this work.

II. Related Work

In [8] an adapted Dijkstra algorithm is presented for load balancing on a CDN using SDN technology. According to the justification used, algorithms that use random selection approaches or the Round Robin one, when selecting a route to the content server, can select the longest route (hops related) and/or with congestion, dramatically decreasing service performance. With the proposed algorithm, the intention is to select the shortest path and with the least congestion.

978-3-903176-24-9

The use of artificial intelligence for load balancing is addressed in [9]. There, the SDN approach is used to redirect the video streams according to the result of the proposed load-balancing algorithm. That work's main goal is to save bandwidth by selecting the best possible path between client and a content server without congestion.

Thus, looking at the works surveyed on load balancing in content distribution systems (especially video), it is possible to note possibility of improvements. In particular, considering the SDN paradigm for the establishment of load balancing between content servers executed directly at the level of the network infrastructure (layers 2,3 and 4). Additionally, using the throughput and CPU utilisation metrics of the servers involved, as well as monitoring and using these values make load balancing highly dynamic and scalable, whether it is adding clients or servers.

III. Proposed Janus Solution

This section introduces the entitled Janus proposed approach to load balancing between MPEG-DASH content servers. This architecture provides two operating modes: static and dynamic.

A. Proposed Architecture: Static Mode

In static mode, the Ryu controller is the main actuator of the system. It performs the key functions such as obtaining the throughput metric of the involved video servers and deciding which one is the most appropriate to the client, generating and installing the traffic redirection rules in the OpenFlow switch. The approach adopted for the performance of the controller in this case is reactive. Thus, once the client attempts to access the web page containing the video content page, that request data packet is forwarded to the controller, if there is no corresponding rule in the OpenFlow switch.

Figure 1 presents the architecture of the proposed solution in static mode, highlighting the activities performed and their interactions as an application belonging to the controller. In addition, Figure 1 indicates the numerical sequence of execution of these activities as a sequence of events.

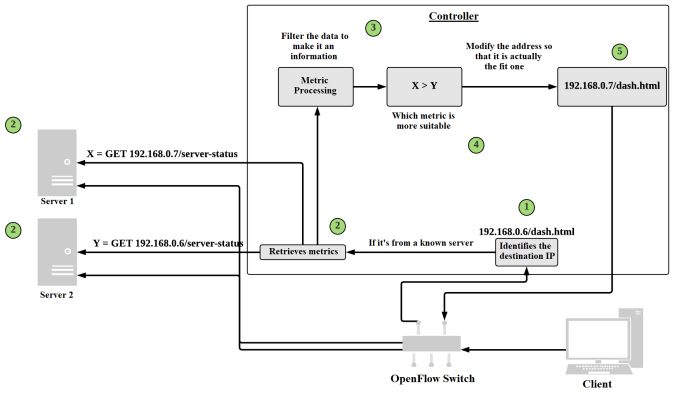

Fig. 1. Proposed Janus Load Balancing Architecture

In this way, according to the proposed architecture, the data packet representing the client request, upon arriving at the OpenFlow switch and forwarded to the controller, is checked in order to identify the requesting client and the content to be delivered. To do this, by establishing the TCP connection and executing the HTTP request of the content, a function identifies the source IP address (1). With the source address, the next step is to check the destination address (1). If the destination address is one of the known and available content servers, the next step is to obtain the load balancing metric values from these servers (2). Values of throughput of the servers is accomplished by the load balancing application, which is developed to the controller. This is performed a single time during the delivering of the video, soon after the process of obtaining the origin IP. That is, with each new TCP connection to the HTTP service, addressed to one of the servers requesting MPEG-DASH video, the metric values used are redeemed. Therefore, load balancing occurs only because of connection establishment (and consequent HTTP session).

After obtaining the values of the metric used, the balancing application evaluates which server has the highest throughput (3, 4) in order to choose one. This is due to the assumption that the one with the highest throughput is able to receive more connections and prone to lower latency. Furthermore, the next and final step is the generation and installation of the OpenFlow switch rules for IP and MAC addresses and the destination port of content traffic based on the best server chosen (5).

B. Proposed Architecture: Dynamic Mode

The dynamic redirection approach is relevant to load balancing since it allows greater flexibility in selecting the most suitable server to receive the MPEG-DASH connection. In this approach, the behaviour of the redirection process is similar to that of the static mode. That is, by noticing the possibility of redirecting to a server whose metric is most appropriate, the controller changes the address fields and logical ports in such a way that the data stream is transferred to and from the chosen server. However, in the process of dynamic redirection, besides the verification of the metrics of the servers when making a new connection, there is also the periodic monitoring of the metrics. Thus, at each given time interval, the metrics of the servers involved are retrieved and if there is a possibility of video flow redirecting to a more suitable server, this action is performed during the video playback, without any perception by the user.

One of the limitations of the static approach is the inability to handle the transfer rate limit. Thus, the use of the throughput metric works to some extent relative to the number of requests met and customers requesting. From that point, the server throughput does not improve with the receiving of more requests, on the contrary, more requests will deteriorate the server's processing performance, leading to a drop in the QoE of users who are already connected. For this reason, in the dynamic approach, the Equation 1 was idealised in order to attenuate such limitation. It indicates that the metric used for load balancing (BC) is a composition between the throughput and the inverse of the server CPU utilisation. That is, the

higher the throughput and the lower the CPU consumption the greater the chance that the server will be used to service a client.

$$BC_i = Ttransfer_i + \frac{1}{\%CPU_i} \quad (1)$$

The metric retrieving frequency was chosen based on the datasets used since they are 126 and 181 seconds long, respectively. Therefore, the time interval is 60 seconds.

IV. Experiments and Results

It is noteworthy that the idealised experiments have as main objective to observe the video reproduction behaviour at the end user computer.

A. Test Environment

All the experiments were performed on Linux virtual machines with help of the VirtualBox tool. Mininet emulator was used to simulate client hosts using an OpenFlow swicht and Ryu controller. Content servers were executed in a particular machine.

Regarding the available video/audio content, two datasets of Creative Commons (CC) license videos offered for testing by YouTube were chosen. The first dataset is called ***CAR CENC***. This video has 181 seconds of duration, and it is available in 6 video qualities (different resolutions) ranging from 256 $\times$ 144 to 1920 $\times$ 1080 pixels and an audio track. In this case, both video and audio are available in chunks of 2 seconds.

The second dataset is called ***FEELINGS VP9***. This dataset has VP9 video compression and therefore its size is considerably smaller than the previous one. However, it presents only high definition quality, i.e., Full HD is not available. This video content has 136 seconds of duration, and it is available in 6 video qualities ranging from 426 $\times$ 240 to 1280 $\times$ 720 pixels and an audio track.

B. QoE Evaluation

In addition to the proposed approach of traffic-oriented load balancing, based on the throughput metric (static mode) and on the throughput and CPU utilization, three other approaches were developed. One of them uses an algorithm to choose the server at random. There is also the Round Robin algorithm executed between MPEG-DASH content servers. Moreover, a third approach, called *Withouth SDN*, was also developed, presenting an approach without load balancing.

To obtain the results, 10 tests were performed for each dataset of each approach and the mean value of those runs were used as a response.

The metrics for evaluating the proposed approaches through the experiments are related to the measurement of the client's QoE, as follows: **a)** The average of bit rate for each second of video execution. Thus, the higher the rate, the better the quality of the video displayed. **b)** The number of times the quality (resolution) of the video has changed. The lower this number, the better the Quality of Experience. **c)** The page load time. The smaller the better. **d)** The time for the first frame of the video to appear on the player. The smaller the better. **e)** The number of stalls in the video. The smaller the better. The ideal is zero. **f)** The average bandwidth consumed by the client.

	Without SDN	Random	Round Robin	Proposed approach Static	Proposed approach Dynamic
Page loading (ms)	574.30 σ = 47.46	819.90 σ = 243.78	987.10 σ = 227.51	679.80 σ = 354.21	282.6 σ = 22.19
First frame (ms)	1078 σ = 55.81	1371.80 σ = 402.50	1486.20 σ = 278.96	1141 σ = 410.58	935 σ = 67.67
Quality switches	5.3 σ = 2.86	3.7 σ = 2.14	3.6 σ = 1.01	3.5 σ = 1.36	5.3 σ = 3.06
Stalls	0.8 σ = 0.97	1 σ = 1.26	0	0.6 σ = 0.91	1.4 σ = 0.91
Bandwidth usage (Mbps)	2.39 σ = 0.26	2.11 σ = 1.36	3.88 σ = 0.24	3.47 σ = 0.59	1.99 σ = 0.49

TABLE I
CAR - FINAL RESULTS

	Without SDN	Random	Round Robin	Proposed approach Static	Proposed approach Dynamic
Page loading (ms)	542 σ = 37,38	819,10 σ = 298,97	1066,40 σ = 49,84	525,80 σ = 450,55	286,3 σ = 37,8
First frame (ms)	1144,10 σ = 118,53	1242,30 σ = 312,76	1679,40 σ = 402,50	1011,80 σ = 582,86	1011,80 σ = 82,15
Quality switches	3 σ = 0	3 σ = 0	3 σ = 0	3 σ = 0	3 σ = 0
Stalls	0	0	0,2 σ = 0,6	0,4 σ = 0,8	0
Bandwidth usage (Mbps)	0,30 σ = 0,09	1,08 σ = 1,29	2,41 σ = 0,10	2,39 σ = 0,15	0,34 σ = 0,15

TABLE II
PARACHUTE - FINAL RESULTS

Tables I and II presents the results obtained. Thus, it is possible to note that for the page load time metric, the proposed architectural modes are more efficient than all other approaches when the dataset is Parachute (Table II), especially the dynamic mode that presented greater consistency and speed. Regarding Car (Table I), the static mode is only lower than the approach without SDN. One of the possible justifications is that in the approach without SDN, there is in fact no need to choose a server because there is no load balancing.

The same occurs with the wait time for the first frame of the video to appear. The architectural modes are more efficient in the Parachute dataset. Again, the dynamic mode is more consistent. However the static approach, loses to the method without SDN in the Car dataset. The justification for this is

identical to that one of the previous metric.

For the next metric, we counted the number of times the video quality changed during playback on the client. First of all, it is worth mentioning that the video is expected to have a minimum number of qualities change. In this case, this amount is 3. This is due to the fact that, when the video is requested, the transmission starts at a lower quality, usually the minimum in the first frame because the player needs to make sure that there is indeed enough bandwidth to increase quality. Therefore, 3 quality changes are usually noticed in the initial seconds. As the second dataset size is smaller, all implemented methods have achieved the best result (3 quality changes). In the dataset car, however, being a uncompressed bigger video content file of maximum quality of 1920 $\times$ 1080 pixels, there were more quality changes in all methods. This is because this video in its maximum quality requires a lot of bandwidth, and depending on the operation of the MPEG-DASH player, when there is not enough bandwidth, the quality of the required video segments is diminished. In some cases, especially on the without SDN and Janus dynamic mode approaches, there are more quality exchanges mainly in the middle of the playback, which negatively affects the quality of user experience. One of the explanations for this is due to the fact that the tests were performed with the bandwidth limit imposed by the *Traffic Monitor* program at 10Mbps and at times there was bandwidth instability, forcing the player to decrease the requested quality of the videos.

Comprising the metric that quantifies the number of stalls, it is important to note that a stall usually occurs when there is not enough bandwidth to consume the video within a certain amount of time, and the player can not decrease the video quality at optimal time for playback continue without interruption. The results were positive in all tests, that is, there were none or few stalls. Although in some situations stalls actually occurred, their occurrence was at the beginning of the video (first frames), so it was imperceptible to the user. This stall behaviour in the second zero of the video is due to the ladder behaviour that the player uses for playback, especially in the initial seconds. Because of this, QoE is little affected. The best result was the Round Robin approach with no stall in the car dataset and an average of 0.2 in the parachute dataset.

The video bit rate variation was also monitored comprising time. When analysing Figure 2 it is noted that among the approaches tested, the one that obtained the lowest mean of bit rate was the one without SDN. The other approaches, however, were quite similar. At the beginning, from the second 1 to 10, all of them have similar behaviour due to the ladder effect previously discussed. From the second 10 a slight variation between them is remarkable. Nonetheless, from the second 40 to the second 70, it is possible to notice a significant decrease in the mean of the bit rate of the without SDN one, as well as in the Janus dynamic approach. The bit rate drops in the dynamic approach generally occurred when server switching was performed, so in some situations due to this change the quality was decreased but soon returned to the highest one.

In the second dataset (parachute) the behaviour of all approaches were extremely similar (for that reason the corresponding Figure was omitted). This is due to the fact that this dataset corresponds to a video which highest resolution is 1280 $\times$ 720 pixels, which means it is smaller than the car dataset, in addition to the VP9 compression. For these reasons the download of this video was fast.

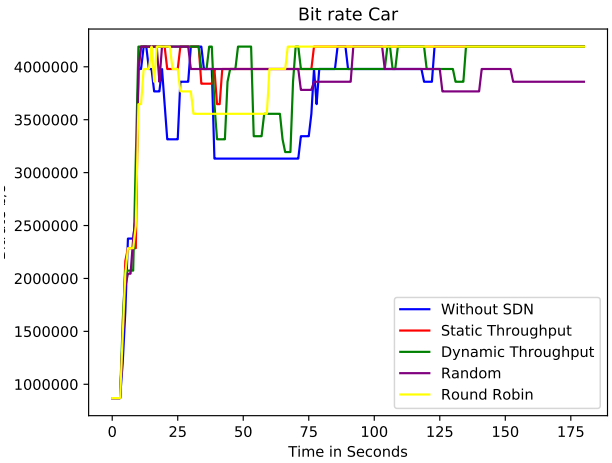

Fig. 2. Final result, average bitrate per approach - Car

In general, according to the performed experiments, the proposed Janus architecture has shown to be promising since it can reduce the response time of the MPEG-DASH video content request, especially in the dynamic mode. On the other hand static mode can maintain the high bitrate, thus preserving the QoE.

V. Conclusion

With Internet popularization and technological advances related to architecture and network structure, video consumption has become a practice in the daily lives of many people. However, despite these advances and constant improvements, there are still several challenges to be overcome. One of them is load balancing so that the content distribution network can support a high demand for requests and does not detract from the end user Quality of Experience.

This work presents a possible solution or at least mitigation to this problem. To do so, an architecture that uses the Software-Defined Networking paradigm to intercept the packets during video content reproduction was devised, and using the analysis of the throughput metric and CPU processing rate of the available content servers, the most appropriate choice of server to deliver the content is possible. The main objective of the evaluation was to highlight that although it is a solution that modifies the connection flows, there is no perception of this redirection in the reproduction of the content consumed by the user and therefore of the quality of experience. Notwithstanding, more experiments are needed regarding the horizontal scalability of the proposed approach to verify its full applicability.

REFERENCES

[1] T. C. Nielsen, "How worldwide viewing habits are changing in the evolving media landscape," http://www.nielsen.com/content/dam/nielsenglobal/eu/docs/pdf/Nielsen-global-video-on-demand.pdf, 73, Avenue des Pléiades B-1200 Brussels, 3 2016, acesso 15 Ago. de 2018.

[2] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper," 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[3] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 04 2011.

[4] T. Bourke, *Server load balancing*, 1st ed. O'Reilly, 2001.

[5] V. Cardellini, M. Colajanni, and P. Yu, "Dynamic load balancing on web-server systems," *IEEE Internet Computing*, vol. 3, no. 3, pp. 28–39, 1999. [Online]. Available: http://ieeexplore.ieee.org/document/769420/

[6] B. Doshi, C. Kumar, P. Piyush, and M. Vutukuru, "Webq: A virtual queue for improving user experience during web server overload," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, June 2015, pp. 135–140.

[7] M. Wichtlhuber, R. Reinecke, and D. Hausheer, "An sdn-based cdn/isp collaboration architecture for managing high-volume flows," *IEEE Transactions on Network and Service Management*, vol. 12, p. 15, 03 2015.

[8] J.-R. Jiang, W. Yahya, and M. Ananta, "Load balancing and multicasting using the extended dijkstra's algorithm in software defined networking," *Frontiers in Artificial Intelligence and Applications*, vol. 274, pp. 2123–2132, 01 2015.

[9] 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. [Online]. Available: http://www.sersc.org/journals/IJGDC/vol9_no1/3.pdf