# Monitoring Metrics for Load Balancing over Video Content Distribution Servers

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

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

***Abstract*—Cloud computing and video streaming services have been in constant expansion in recent years. Along with it, the demand for computing resources has also increased significantly. In this context, monitoring the use of these resources is crucial to maintain a satisfactory level of Quality of Service and, consequently, Quality of Experience, especially in video transmission services. This work discusses a new method of monitoring resources and quality of service metrics on content servers involving CPU utilization and server throughput, which is obtained in a distributed way. For that, a distributed collector system that is based on a modified version of the ring election algorithm is developed to retrieve the Quality of Service metrics in each server. Evaluation experiment results show that there are no performance gains on the system such as the content loading faster for the user, there are however, improvements in terms of the whole system scalability. The greater the number of servers for monitoring, the better the approach is compared to the traditional method of monitoring resources through request and response.**

***Index Terms*—Load balancing, SDN, Monitoring**

## I. Introduction

The cloud computing and video streaming service markets have been growing steadily in recent years. With the advent of the COVID-19 pandemic, these numbers have become even more significant. According to *Global Industry Analysts* [1] cloud computing services revenues reached an approximately revenue of US$313.1 billion in 2020 and they estimate that by the year 2027 this number could reach around US$947 billion.

In this sense, the use of computational services is a striking and determining characteristic of the evolution of societies. Everyday human activities, such as transport and education, are increasingly mediated or dependent on technologies that involve computing. One of the human facets that benefit from this evolution in particular is the consumption of online content. In particular, such consumption can range from information about everyday factual events (news), online learning, online business services, to audio/video based entertainment. Thus, it is essential that the proper management of these services is carried out, allowing not only their use by as many people as possible, but also in order to produce a positive effect on relations between humans and, consequently, on society, as well as a catalytic effect on the economy.

Thus, the situation of video transmission services presents similar characteristics in terms of the growth of cloud services, as according to [2] in 2020 this market segment achieved approximately US$100 billion and suggests that by 2026 this number exceed US$200 billion. Furthermore, according to [3], by the year 2022, Internet video traffic should reach 82% of the total.

With such growth, the opportunity for new technologies and approaches to network infrastructures is promising. This promise is justified since at this growing pace service providers are facing trouble to maintain adequate quality of service. In this context, there are methods to alleviate this problem. Load balancing is one of them and its techniques can be classified according to the way they work. One of them involves server provider selection algorithms. It is worth noting that one of the most important functions of a content distribution network with an infrastructure based on replication is the method of choosing the server that will reply to the client's request. According to [4], the optimal choice of provider made by the load balancer can benefit the system as a whole in several factors, such as reducing the chance of failure and overload, providing robustness, improving scalability, reducing the response time of services, resulting in greater customer satisfaction and considerably reducing the cost of system maintenance.

Load balancing can be addressed by several approaches. One of them is based on performance metrics. Generally speaking, when faced with a request, the load balancing model selects the most capable server to meet the demand at that time. This server choice process can be based on one or more pre-defined metrics or on the behaviour of the network as a whole.

Thus, as an evolution of [5], this paper presents an approach to monitor and consequent recover of metrics that make up the load balancing indicator, which aims to maximize the resource utilization of the content source servers. The Software-Defined Networking (SDN) paradigm is also used, allowing the load balancing approach to be based on metrics and acting directly at the session layer, that is, instead of a load balancing server being needed, such balancing occurs directly in the packet

978-3-903176-51-5  

switching equipment compatible with OpenFlow (SDN) in the network segment where the various clustered servers are located. In addition, this work employs the use of a modified version of the ring election algorithm [6] to collect the metrics in a distributed manner relieving the controller's workload. Furthermore, the main contribution of this work is focused on relieving the workload from the controller in a way that regardless of the number of servers in the network meant to be monitored, the controller monitoring workload will be little or not affected at all.

The remainder of this work is organized as follows. Section II, discusses related work. Section III gives work background and it details the proposed solution approach. Section IV-B presents the proposal evaluation and discussion. Finally, Section V presents the final considerations of this work.

## II. Related Work

In [7], the authors use the scaling algorithm (*Hill Climbing Algorithm* (IAMLBHC)) to improve the response time and cost of processing in the cloud. However, the greatest effort is dedicated to reducing the final cost to the user, since the use of the scaling algorithm aims, in the cloud, to allocate resources to the desired virtual machine efficiently, fulfilling the requirements with the lowest possible value. The metric for VM selection, which in this case when compared to the present work can be considered as a server, is the least allocated. In other words, the VM with the least allocated resources and the lowest price will be chosen. It is also worth noting that the focus is on cloud storage systems.

The work [8] uses a modified version of the *Weighted Least Connections* (WLC) algorithm. Such algorithm works in order to keep records of each server with its active options and then redirect the connection to the one that it least prefers. The authors however use a combination of CPU, network, memory and disk metrics to accomplish such redirection. In this way, the server that has the lowest value, based on the previous metrics, is defined as the one that has the least resources used and thus, receives the new connection.

A work that uses artificial intelligence for load balancing was addressed in [9]. In it, the SDN approach was used to redirect the video streams according to the result of the proposed algorithm. This algorithm is based on a neural network that uses as its main metrics the bandwidth usage, packet lost rate, latency and the number of hops. However, the work does not focus on content distribution networks. The work's main objective is to save bandwidth by selecting the best possible path without congestion.

In [10], load balancing is performed through the help of software-defined networking. A bandwidth monitoring application between servers is proposed in such a way that there is a certain threshold to be used for each server and when this threshold is exceeded, the application detects an overload of work on that server. From there, the application redirects the connection flow to a server that has a bandwidth usage below the previously stipulated threshold and also the packet loss and latency factors are taken into account.

The work presented in [11] brings an elastic solution based on OpenFlow for video transmission services in content distribution networks. In the presented architecture there is a load balancer that aims to balance the system workload by monitoring resources such as RAM memory, CPU utilization, number of active clients and latency. The idea is to manage connection flows in such a way that the load balancer redirects connections to the server that has the least workload and therefore the least resources being used. Despite promising results, the focus of the work is the solution architecture that involves other components such as an orchestrator.

Thus, looking at the most recent works that deal with load balancing and monitoring of resources in content distribution systems (especially video), it is possible to recognize the possibility of improvements. Especially taking into account the use of the SDN paradigm to establish load balancing between content servers, created directly at the network infrastructure level (layers 2, 3 and 4). Furthermore, the use of the throughput and CPU utilization metrics of the servers involved, as well as the monitoring and on-demand utilization of these values makes load balancing highly dynamic and scalable, whether adding clients or servers. In addition, the goal of recent works is focused on the operation of load balancing, often leaving aside how the performance metrics rescue procedure takes place, which is an important contribution that this work also seeks to explore.

## III. Proposed Solution

This section presents the proposed approach to monitoring metrics for load balancing over content servers. The approach consists of two elements, content server metrics monitoring application and an SDN network consisting of the controller and SDN-compatible packet switching equipment (OpenFlow) that connects these servers. In this context, the monitored metrics are transformed into a load balance indicator for each server by means of a mathematical equation. Thus, these servers and their indicators go through the election process and then, with the election completed, the details of the elected server, that is, the one most apt to receive the connection, is sent to the controller which is responsible for carrying out the next specific steps to be performed for the correct redirection of user video requests.

### A. Previous Proposed Approach - Request Based

The load balancing presented in our previous work [5] is classified into two approaches, static and dynamic, both centralized on the SDN controller. The static load balancing approach checks for the possibility of user connection redirection before starting the content delivery/transmission. In the dynamic approach, in addition to the initial verification, there is also the periodic verification of the possibility of redirecting the connection and consequently, the content transmission. The advantage of the dynamic approach refers to the flexibility of the system, since there is the possibility of changing servers during content transmission, making it easier to balance the system when the workload of each server tends to increase.

Thus, in both cases, the Ryu controller is the main actuator of the system, performing the main functions such as obtaining the throughput and CPU utilization metrics of the content servers involved and making the decision of which is the most suitable for generating and installing the traffic redirection rules in the *switch* OpenFlow.

Figure 1 represents a sequence diagram depicting the static approach. Such sequence is started with the request made by the client for certain content, which in the proposed scenario, is the MPEG-DASH video. The request, when detected by the OpenFlow switch is then forwarded to the controller. The next step is to identify the purpose of such request. If it is an HTTPS request and the destination IP is from one of the content servers, the metrics are retrieved. In this way, such recovery is executed sequentially for each server by the controller. Therefore, at every new connection the controller detects if the rules have been met and if so, will start the metrics recovery process and then redirect the connection when necessary.

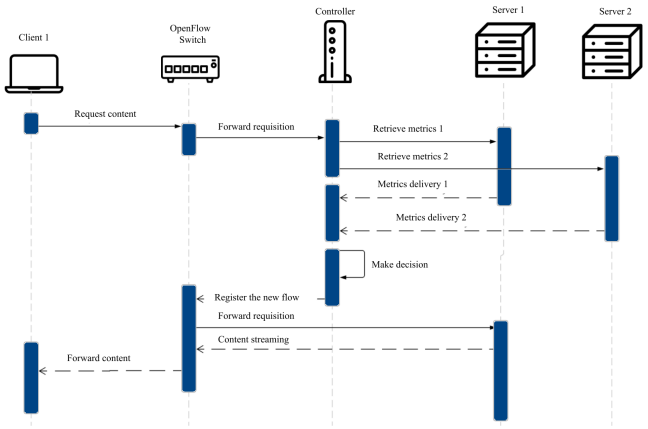


Fig. 1. Sequence of events from request-based static load balancing approach

After executing the metrics recovery process, the calculation to obtain the load balancing indicator is performed according to Equation 1. Using the throughput and CPU utilization metrics, Equation 1 generates the indicator for load balancing (BC) which is a composition between the throughput and the inverse of the server's CPU utilization. In other words, the higher the transfer rate and the lower the CPU consumption, the greater the chance that the server will be used to serve a new client. With the indicator calculated for all servers, the controller checks which indicator value is the highest, defining the corresponding server as the one that will service the client's request. After the most suitable content server is defined, the controller starts the rules installation process in the OpenFlow switch according to the connection destination data, with the chosen server's destination IP, MAC and logical port.

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

In the dynamic approach, the redirection process is similar to the static approach. That is, when there is the possibility of redirection to a server whose metric is more suitable, the controller changes the connection details in such a way that the data flow is transferred to and from the chosen server. However, in the dynamic redirection process, in addition to checking servers metrics when making a new connection, there is also periodic monitoring of metrics. Thus, at a given time interval, the metrics of the servers involved are retrieved and, if there is a possibility of redirection to a more suitable server, this action is performed during the video playback, without any user perception.

The dynamic approach works as follows. At the end of the metrics retrieval process, the controller checks the possibility of redirection, that is, if there is another server that has a higher indicator than the one serving the content at that moment. If so, the controller will remove the flow from the session, which is then momentarily passed through the controller. Then, a new rule is installed covering the current client, but with new connection destination details, with IP, MAC and logical port. As the new redirected server is identical, i.e., all clustered servers are clones with the same video catalogue content available, it has the continuation of MPEG-DASH video segments and therefore the continuation of streaming is unaffected.

The process still continues, since even after the successful redirect, the previous connection remains open and with that, resources are consumed and therefore wasted. Thus, to solve this problem, the controller produces two TCP packets to end the previous session and thus free up the resources being used. By inspecting the last packet received by the controller, it is possible to perform a calculation regarding the FIN+ACK to close the connection. With this information, the controller sends this packet to the previous server, which also responds with a FIN+ACK. Finally, the server replicates with ACK for the connection to be finally terminated.

In the dynamic approach, both request based and election based, there is a periodicity to retrieve the metrics. In the test environment, this periodicity was determined regarding the size of the content made available on the content server, which is 181 seconds. Therefore, a 60-second periodicity was selected. The reason for using this periodicity is due to the fact that, in a more frequent case, it could cause unwanted effects in the video playback, such as stalls for example, and in a less frequent case, the balancing could have no effect.

### *B. Proposed Approach - Election Based*

In this current proposal, the metrics recovery for the load balancing indicator calculation is carried out in a distributed way with the aid of a modified ring election algorithm. In this way, the controller is only responsible for the data packet flow manipulations.

The execution of the ring election algorithm takes place concurrently on all servers and it was developed using Python language with socket programming. The election result is the choice of the server most capable of serving certain content at that particular moment. When the election finishes, a message is sent by the elected server containing its data to a particular

SDN controller's TCP port in order to provide the contact information necessary to client's request/reply flow redirection configuration.

The periodicity of performance metrics rescue is identical to the requisition based approach. Determined regarding the size of the content made available on the content server, the a 60-second periodicity was selected to this approach as well for the same reasons. A more frequent case, could result on unwanted effects in the video playback, such as stalls for example, and in a less frequent case, the balancing could have no effect.

Thus, when the election process is started, each server has the task of reading its own throughput and CPU utilization metrics and from there, calculating its load balancing indicator. This indicator represents an identification number for the election algorithm, that is sent to other servers unidirectionally according to a logical ring construction among them. At the end of the election process it is selected, from the list of identifiers, the one with the highest score, which is then sent along with the associated server IP address and logical port to the controller. In the controller, this information is used to configure the rules for redirecting traffic to and from the client.

This new election-based approach assumes that, if each server retrieves its own metrics, the response time and latency problems regarding the controller in a network must be mitigated since, as they act locally, such values tend to be significantly smaller. Furthermore, in the approach whose controller is responsible for retrieving the metrics, as more servers are added to the system for monitoring, the greater the controller's workload. Therefore, as the metrics are retrieved periodically, there is a possibility of controller workload saturation. Also, depending on the geographic location of each server, there is the possibility of more delays in relation to the response time of each one of them, plus the latency for sending these metrics. Therefore, decentralizing this process tends to help with the scalability of the system. Figure 2 represents how this approach works.

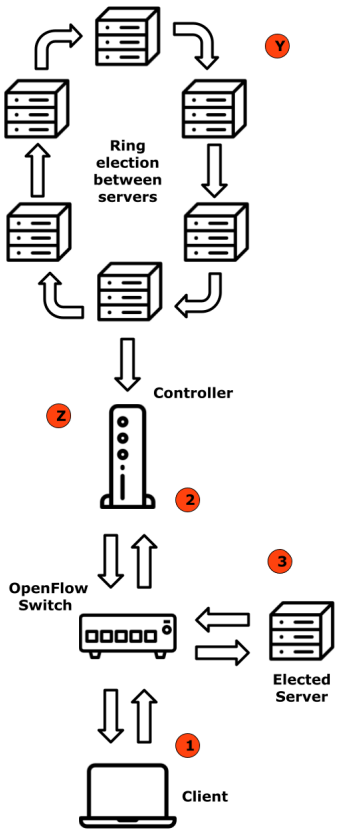


Fig. 2. How the election-based approach works

### C. Implementation

For the instantiation of the proposed approach, in addition to the implementation of the load balancing application with the Ryu controller, in the absence of a CDN for testing, an infrastructure that simulates client access to several content servers, under manipulable network conditions, is required. In this sense, the implementation scenario adopted has as elements *n* content servers, a controller, an OpenFlow switch and *n* clients, as shown in Figure 3.

Thus, for the particular case of this work, due to hardware limitation, but without loss of objectivity, only two MPEG-DASH content servers were created, both identical. That is, both have the same content to be made available. With regard to customers, even though they are on the Mininet network, and therefore virtualized, they were provided with external access including the Internet, so that they could access the content servers. The Mininet topology used was the single topology, with one switch and N hosts.

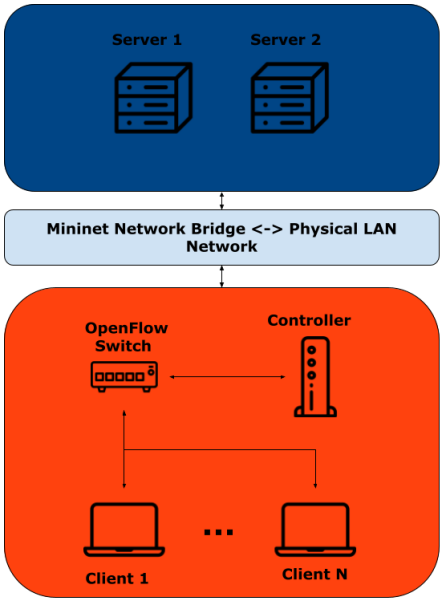


Fig. 3. Deployment scenario testbed - based on distinct networks

The proposed implementation scenario was designed so that the network created by the Mininet emulator can access the physical LAN network (representing the WAN network in a

real scenario), in which the servers are hosted. This connection occurs through a bridge that connects these two interfaces through the Open Virtual Switch (OVS) switch.

More specifically, the configuration of virtualized clients (consumers of MPEG-DASH video content), existing in the Mininet network, is carried out so that they receive IP addresses from the real physical LAN network that has access to the Internet, as shown in Figure 4. Furthermore, the environment thus configured allows the Ryu controller to exercise its activity on the OVS switch that connects such clients to external servers (WAN).

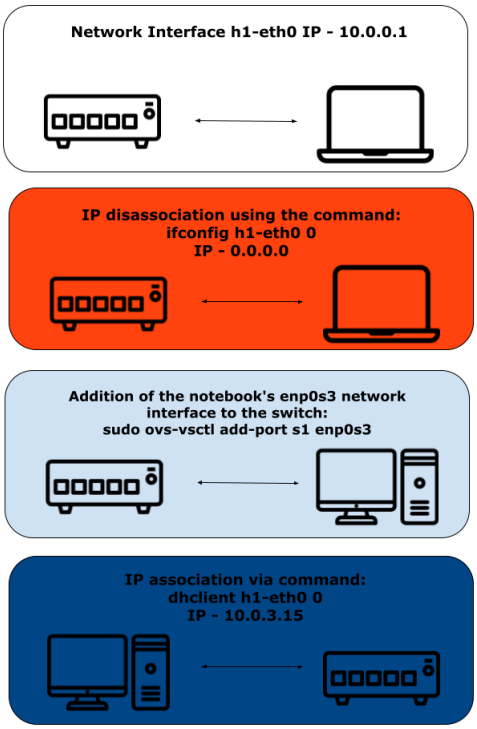


Fig. 4. Host IP configuration - step by step

To simulate a more realistic scenario in terms of MPEG-DASH video consumption, it was idealized that all clients created within the Mininet network could use an Internet browser to perform the visualization of the received content. In this case, the Google Chrome browser, with changes to run in administrator mode, was used because it allows the easier use of MPEG-DASH content player. In this case, the player *Dash.js* [12] was used.

To retrieve the throughput metric values in MPEG-DASH video servers, the Apache server provides a tool known as *server-status* which launches an HTML web page with all possible information and performance metrics from the server. This page is accessed by the election algorithm process of the server itself, which processes the data received in order to obtain the throughput and CPU utilization rate of the server. Thus, using Equation 1, the server load balance indicator is used in the ring election process.

The development of the proposed load balancing algorithm was performed based on the switch layer 2 application of the Ryu controller's available library. From this application, the necessary modifications were performed in such a way that when an MPEG-DASH content request is received in the *switch* OpenFlow, all TCP packets sent to the destination content server's IP that intend to initiate a connection will not actually be delivered directly to the server, but redirected to the controller for processing.

Thus, once the load balancing application receives the connection details, it modifies the network data plane by installing flow rules whose action must rewrite the IP address, MAC and destination port of all future client packets to the IP address, MAC and port of the chosen server. The same is done for the flow in the opposite direction, that is, from the server to the client. In this way, the redirect will be performed automatically and transparently and the user/customer will not know that this exchange has been made.

## IV. Experiments and Results

The ideated experiments aim to evaluate the premise of improving scalability with the election-based approach as well as verify the behavior of content reproduction. Thus, the evaluation metrics are related to process time variation with the addition of a new server, response time of the HTML page that makes the content available, first frame appearance time and video quality variation during playback.

### A. Testbed

All experiments were performed on Linux virtual machines with the aid of the VirtualBox tool. The first device used is a desktop computer. Its role is to run the virtual machine that hosts the Mininet network, containing the hosts, the OpenFlow (OpenVSwitch) switch and the controller. The operating system used on this machine was Linux Ubuntu with 2 GB of RAM with a 2 core processor at 4.2 GHz. The second used device is a notebook. The virtual machine specifications are similar to the previous ones. However, it has an inferior processor running at only 3.2 GHz, it was only used as a content server, running an Apache server. In addition, another notebook with a 2.9 GHz processor is used as the second Apache content server.

The standard ISO/IEC 23009-1 or Moving Picture Expert Group Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [13] was chosen due to its dynamic video quality selection. In such standard, the video content is made available in several resolutions in small segments. The client video player will run depending on the network bandwidth available at that particular moment and if there is a need to lower or increase the quality, as the content is available in segments, the video will not be stopped and just the quality change will be perceptive instead of stalls. This is particularly beneficial for dynamic load balancing systems once its segmentation system makes it possible to change servers mid playback.

Modifications (functions) were implemented in the MPEG-DASH player to allow the analysis of the user quality of

experience, that is, to allow tracking the behavior of certain statistical metrics of video transmission during content playback such as bitrate variation.

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

To simulate an environment close to a real scenario, the *Traffic Monitor* tool was used to edit some network settings. The bandwidth has been limited to 31Mbps and a latency of 30ms has also been added. These settings were used because according to [14] the average bandwidth in Brazil in the year 2021 is 31 Mbps and the average ping is 31ms.

### B. Evaluation of the Proposed Approach

To obtain the results, 10 executions of the test dataset of each proposed approach were carried out, and the average value of these executions was used as the final value. At each test, the controller was terminated as well as the Mininet network and the browser cache on the clients was emptied. The metrics for evaluating the proposed approaches through the experiments performed are related to the measurement of the customer's QoE, as follows:

- The average bitrate for every second of video play. The higher the rate, the better the quality of the displayed video.
- The number of times the video quality (resolution) has changed. The lower this number, the better the quality of end-user experience.
- The page load time. The lower the better.
- The time for the appearance of the first frame of the video in player. The smaller the better.
- The number of freezes (stalls) in the video. The smaller the better. The ideal is zero.

Figure 5 shows the average bitrate for the two evaluated approaches, the election-based one, proposed in this work, and the requisition-based one, available in the literature. Note that there was minimal difference in the comparison, characterized by the speed at which the request-based approach achieves maximum video quality. However, this is not relevant in determining that such an approach is in fact superior to the requisition-based approach. Furthermore, an important takeaway is that the proposed approach does not suffers from any anomaly during video reproduction.

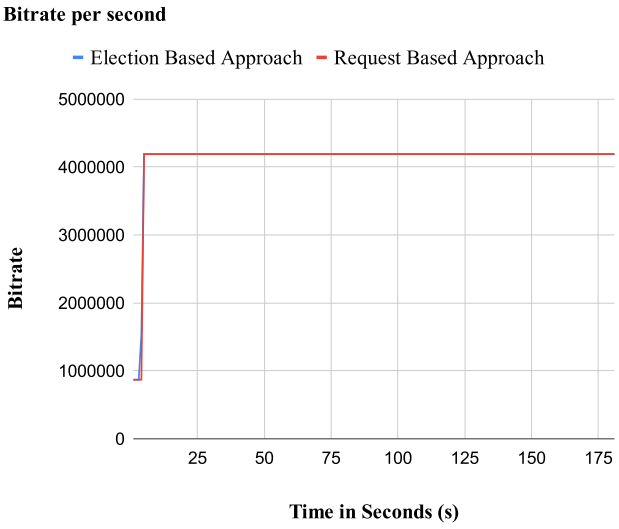


Fig. 5. Bitrate per second of the two approaches

Figure 6 illustrates the response time of the two evaluated approaches. Again, the difference presented by the results is too little that it is not significant enough to determine the best solution.

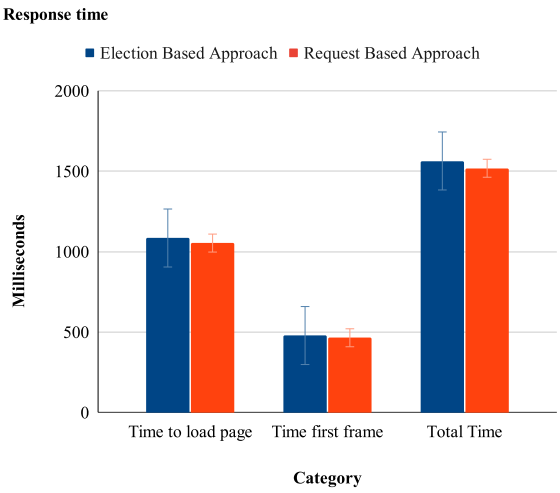


Fig. 6. Response Time

The result that determined an advantage of the election-based approach over the request-based approach is illustrated in Figure 7. There is a clear difference between the scalability of each approach. In the experiments, the difference that the addition of a new server makes to the request-based approach is notable, with each server adding roughly 0.2 seconds. In the election-based approach, with every new server added, the amount of time is considerably less than in the request-based approach with each server costing around 0.05 seconds. A possible explanation for this result is based on the fact that the controller responsible for each metric retrieval is dependent

on the response time along with the latency of each server. Thus, when added together, these metrics affect the total retrieval time and load balancing indicator calculation. In the election-based approach, servers retrieve their own metrics, so the response time and latency are minimal. However, in the election process, when there is an exchange of information between servers, although in the experiments performed there are no notable differences, in a real scenario there is the possibility of the situation being reversed.

Regarding the number of video quality/resolution changes during play, both approaches showed no changes during media play except for the initial seconds to achieve and maintain maximum quality. The stalls did not occur in any of the approaches.

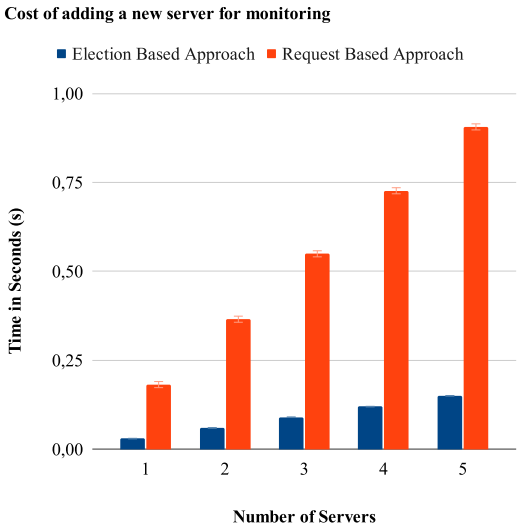


Fig. 7. Scalablity - Server addition time cost

## V. Final Considerations

This work presented a new approach to load balancing in content servers. To this end, an architecture that uses the paradigm of Software-Defined Networking to intercept packets during video content playback was devised, and through the analysis of the throughput and CPU processing rate metrics of the available content servers, it performs the choice of the most suitable server to deliver the content in order to maximize its throughput. Also, to alleviate the controller's workload, the recovery of the load balancing indicator is carried out in a distributed manner through the distributed election approach.

With the tests presented, it is possible to observe a trend of increasing controller workload as the number of content servers increases and thus, performance problems can arise, such as increased response time and constant quality changes during playback of the video. Therefore, an election-based approach that uses a modified version of the ring distributed election algorithm was proposed. After its analysis, it is possible to verify a possible gain regarding to system scalability, and consequently alleviation of the controller workload when compared to the literature request-based approach. It is also verified that, despite several processes of flow manipulation and metrics retrieval occurring during video playback, it is possible to notice that user quality of experience had not losses.

However, there is a need for large-scale tests to fully validate the proposed approach. Moreover, a tradeoff analysis considering number of messages sent in the logical ring among servers and the used bandwidth, between the two ways of retrieving the load balancing indicators is necessary.

The proposal itself can be enhanced. Eventually, new metrics can be added in order to model servers workload. Furthermore, enhancements are planned comprising the election algorithm regarding how to automatically set up its running periodicity and minimizing election traffic messages.

## Acknowledgments

This work received financial support from the Coordination for the Improvement of Higher Education Personnel - CAPES - Brazil (PROAP/AUXPE) 0093/2021.

## References

[1] I. Global Industry Analysts, "Cloud computing services - global market trajectory & analytics," 2021.

[2] J. Stoll, "Ott video revenue worldwide from 2010 to 2026," 2021.

[3] 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

[4] B. Sahoo and A. Prusty, "Heuristics load balancing algorithms for video on demand servers," *International Journal of Artificial Intelligent Systems and Machine Learning*, vol. 5, 08 2013.

[5] E. J. d. Passos and A. Fiorese, "Janus - a software-defined networking mpeg-dash video streaming load balancer," in *2019 15th International Conference on Network and Service Management (CNSM)*. Halifax, NS, Canada: IEEE, 2019, pp. 1–5.

[6] E. Chang and R. Roberts, "An improved algorithm for decentralized extrema-finding in circular configurations of processes," *Communications of the ACM*, vol. 22, no. 5, pp. 281–283, May 1979.

[7] M. Zedan, G. Attiya, and N. El-Fishawy, "Load balancing based active monitoring load balancer in cloud computing," in *2021 International Conference on Electronic Engineering (ICEEM)*. Menouf, Egypt: IEEE, 2021, pp. 1–6.

[8] X. Ren, R. Lin, and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*. Beijing, China: IEEE, 2011, pp. 220–224.

[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

[10] S. Yilmaz, A. M. Tekalp, and B. D. Unluturk, "Video streaming over software defined networks with server load balancing," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 722–726.

[11] P. A. L. Rego, M. S. Bonfim, M. D. Ortiz, J. M. Bezerra, D. R. Campelo, and J. N. de Souza, "An openflow-based elastic solution for cloud-cdn video streaming service," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–7.

[12] dash.js, "A reference client implementation for the playback of MPEG DASH via javascript and compliant browsers.: Dash-industry-forum/dash.js," 2012, disponível em https://github.com/Dash-Industry-Forum/dash.js. Acesso 13 Out. de 2018.

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

[14] Etrality, "Internet speed index," https://www.speedcheck.org/internet-speed-index/, 2021, accessed: 2021-11-09.