International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 14, Number 6 (2021), pp. 530-536

Construction of a client-server scenario for the simulation of interactive video streaming services

Wilmar Yesid Campo_Muñoz[1], Gabriel Elías Chanchí Golondrino[2] and Isabel Cristina Baos Prado[3]

[1] Ph.D., Facultad de Ingeniería, Programa de Ingeniería Electrónica, Universidad del Quindío, Colombia.

[2] Ph.D., Facultad de Ingeniería, Programa de Ingeniería de Sistemas, Universidad de Cartagena, Colombia.

[3]Esp., Facultad de Ingeniería, Programa de Ingeniería Civil, Universidad del Quindío, Colombia.

*[1]*Corresponding Author (ORCID: 0000-0001-8585-706X).*

[2]ORCID: 0000-0002-0257-1988, [3]ORCID: 0000-0003-4076-3320

Abstract

The services supported by video streaming technology are the largest consumers of bandwidth in today's data networks, whether being wired or wireless. Among these services there is the interactivity feature which must be considered when it comes to getting to know the traffic flow behaviour of the video on demand, the quality of service, and the quality of the experience where pausing, forwarding, and rewinding correspond to the interactivity functions in these types of services and whose effect is stopping or generating data bursts. Thus, in this paper the authors present as a contribution the construction of a video streaming client-server scenario that allows to model the interactivity of these services. For this task, the Opnet Modeler tool is used using the module that represents layer seven of the OSI (Open Systems Interconnection) tower, where the construction is carried out through a Finite State Machine (FSM). Finally, the proposed scenario is validated, and the statistics generated by the interactivity processes are analysed. Thus, the scenario proposed in this paper can serve as a reference for the characterization and customization of video streaming services in different application contexts.

Keywords: FSM, Interactivity, Opnet Modeler, Video streaming.

I. INTRODUCTION

The services supported by video streaming technology are the largest consumers of bandwidth in current data networks, whether being wired or wireless [1-3]. This way, the management of this type of services is essential in such a way as to allow the elephant and mice flow [4] [5]. Different approaches have been built such as testbed and simulation models [6], [7]. These approaches aim to determine the behaviour of the elephant flow associated with video streaming services as in the case of the video on demand provided by OTT (over the top) companies [8]. However, these proposals have focused on determining quality of service (QoS) parameters, and they even try to estimate the quality of experience (QoE) [9-11].

On the other hand, a relevant characteristic which is considered in modern networks is low latency that, among other potentialities, allows interactivity even in video on demand services [12], [13]. Therefore, interactivity is a characteristic to consider when it comes to knowing the behaviour of the traffic flow of video on demand, determining QoS parameters, or estimating QoE where pause, forward, and rewind correspond to the interactivity functions in these types of services [14] and whose effect is stopping or generating data bursts.

Considering the aforementioned, the contribution of this paper is proposing a client-server scenario for the simulation of interactive video streaming services. For doing this, the characterization of the traffic components of the video streaming service is used; that is, the representation by means of probability density functions (PDF) of the group of pictures (GOP) and the audio presented in [15] and [16]. The PDF of the GOP and the audio are programmed in the server using the Opnet (Optimized Network Engineering Tool) Modeler [17]. It starts with the construction of a module that represents layer 7 of the OSI tower (Open Systems Interconnection) called application, which by means of an FSM the behaviour of the server is represented, creating a state for each video that can be transmitted, and that is characterized by a PDF. In the same way, the client is characterized, but in this case with a greater diversity of states in the FSM since it is the client who generates pausing, rewinding, or forwarding; that is to say, it is the active component which requests interactivity. Thus, the simulation scenario presented here can be used in any type of network infrastructure, regardless of being a wireless mobile network or a wired network in order to carry out different types of studies that involve video services and their interactivity functions.

The rest of the paper is organized as follows: Section II presents the materials and methods corresponding to the construction of the server and the client. Section III presents the results and discussion, first the validation process and then the interactivity functions and their effects on the video. Finally, section IV presents the conclusions.

II. MATERIALS AND METHODS

This section describes the logical process of each of the states that make up both the client and the server where interactivity is achieved.

II.I The server node

The model in charge of representing the interactive behaviour of the video streaming services was developed by means of an FSM which was built through the Proto-C programming

language, supported by the Opnet Modeler tool. Fig. 1 shows the process associated with the server node made up of an FSM consisting of 15 states, 13 forced, 2 unforced, and their respective transitions. The forced states of green colour are characterized because when the process enters them, it executes the input Proto-C code, which is built in the upper half of the circle and immediately executes the exit code, which is built in the lower half of the circle to enter the next corresponding state. Unlike these, when the process enters a non-forced state which is identified with a red colour, the state input code is executed and waits for a previously determined interruption to occur in order to execute the exit code and enter the corresponding state, see Fig. 1.

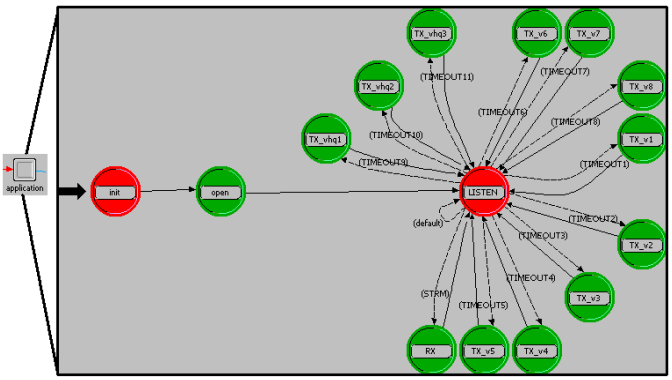


Fig. 1. Server model at the process level, application module.

Similarly in Fig. 1, the application module for the server can be seen at the process level, which has six different types of states whose assigned names are: init, open, LISTEN, RX, TX, and TX_vhq. Each of these states is described below.

II.I.I Init state of the server node

It is the initial state of the process in which the variables necessary for the operation of the model are initialized. Init initializes the variables responsible for containing the PDFs with the parameters that characterize the components that determine the behavior of the interactive video streaming services. These variables correspond to the time between frames of each component of the GOP and the audio. This state also contains the variables, pause duration, and forward and rewind time. Finally, a countdown-type interruption is encountered in this state that causes the process to exit the init state after a certain amount of time; in this case, a 200-second wait to ensure that all component modules of the node start correctly

II.I.II Open state of the server node

This state is in charge of opening the session where the application module can communicate with the module of the OSI tower lower layers. In order to communicate the server with the client in such a way that the information flows through the different layers of the node taken from the OPNET Modeler libraries, it is necessary to open a passive connection; which is, waiting for the client to request the server the use of a video. This session also serves to pass important connection data from the application module to the lower layer modules such as the transport protocol to be used and the service name. This state determines the node identification which is the name that has been assigned to the serving node at the network level.

II.I.III LISTEN state of the server

This state allows you to listen to the channel to determine when the client makes a request. The state has no input or output source codes. However, it allows the process to move to the different states for video transmission and to the receiving information state. As it can be seen in Fig. 1, the LISTEN state is an unforced state, and it is connected to several states through different interruptions. In addition, it has an interruption that takes it to itself, called "default", which is used for preventing infinite loops.

In Fig. 1, there are two types of interruptions that cause the process to exit the LISTEN state which are TIMEOUT and STRM. The former is an automatic or countdown type interruption, which means external interruptions such as traffic flows are not required but only the summoning of the interruption from the same or any other state. This function has as a parameter the time in which the interruption will occur. The interruption times have been defined in the forced states connected to the LISTEN state, including the RX state. Then, when the process enters any of these states, an interruption time is defined that will make the process return to the same state or to a different one depending on the task being carried out.

On the other hand, the STRM interruption that connects the LISTEN state with the RX state has been defined as a stream type interruption. To activate this interruption, it is necessary that the application module receives a packet from the lower module; in other words, from the lower layer of the OSI tower. To make it happen, a packet must be sent from the client to the server.

II.I.IV RX state of the server node

The RX state is activated only when the server receives a packet from the client. Its function is to provide control for the transmission of the different videos to the client depending on what he requests. When the server starts, its process reaches the LISTEN state and waits until any of the interruptions that take it to another state are activated. If the client does not make requests, the server process remains in the LISTEN state because the interruptions for the transmission of the videos are activated in the RX state. When the client transmits a packet requesting a video, the RX state analyzes it and determines the video that the client is requesting. Then, the RX state schedules an interruption that will take the process from the LISTEN state to the TX state corresponding to the video that the client has requested. Since the RX state is a forced state (see Fig. 1), once its input code has been executed, it immediately returns to the LISTEN state. The other task that this state performs is activating the interactivity functions in the video playback: pause, forward, and rewind.

II.I.V TX state of the server node

In the process developed for the server, there are 11 states called TX, which correspond to 11 videos that have been characterized by different PDFs in [15] and [16]. These states perform the tasks of transmitting the videos and controlling the interactivity functions. For this purpose, the state allows to transmit each type of video frame which are the PDF of the GOP components and the audio in the exact order with the time use between frames. In addition, there are flags that indicate whether to pause, forward, or rewind. The mean of each PDF was used for each time component between frames because the times between frames are not constant but are determined precisely by the PDFs. For the control of forwarding and rewinding during playback, it must be ensured that their length does not exceed the remaining reproduction time or that the video has finished.

For controlling playback pauses, it is first determined whether the client has sent this type of request. If so, the pause length is determined. After the pause time has elapsed, an interruption is programmed that will cause the process to restart the transmission. Also, if a pause occurs, the process will not read the frame transmission code but will immediately exit the state to wait for the pause time to expire.

II.II. The client node

This node is responsible for making video requests to the server, as well as obtaining important statistics to determine the traffic behavior in the model. In the client node, the model must also be modified in the application layer.

In Fig. 2, the client node is seen at the process level built by means of an FSM. The process shown in Fig. 2 is housed in the application module of the client node, and it is responsible for carrying out data collection and packet transmission tasks to perform video requests and interactivity functions. It is observed that the states that make up the process within the application module are: init, open, CLIENT, idle, pause, forward, RX, rewind, TX_RQ, Stat and END. The description for each one of them is next.

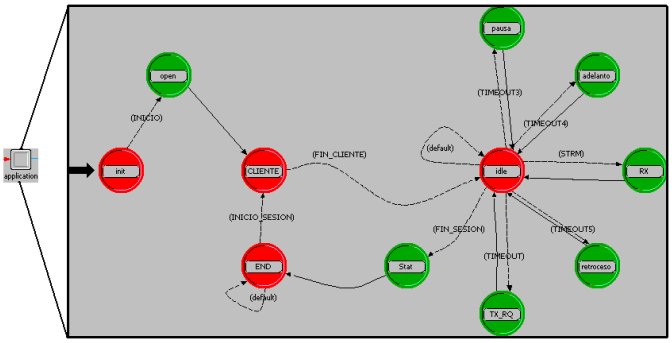


Fig. 2. Client model at the process level. Application module

II.II.I Init state of the client node

After the system starts, the init state of the process will only run once. This node extracts the attributes of the Client node in which the information corresponding to the address of the client and the server is found. The address of the Client node is self-assigned, while the address of the server it connects to is configured as server. This attribute has been created manually in order to facilitate its configuration for users of the proposed scenario who do not have knowledge about it, since it is also possible to configure this parameter directly in the code. As it was done for the init state on the Server node, the Client node has also scheduled 200 waiting seconds to ensure that all component modules of the node start correctly.

II.II.II Open state of the client node

This state is responsible for opening an active connection. This is because it is the client who will make the requests for videos and interactivity functions to the server. This node has the following parameters: node ID, remote address (server address), remote port (server port), transport protocol, service name, application ID, parameter configuration for RSVP protocol, and the type of service (ToS). The other parameters configured for the active session are identical to those configured for the passive session on the server.

II.II.III Client state of the client node

This state was created to be used as a starting point for the process each time a new session is started, being this defined as the interval in which a certain amount of videos are played, determined by the Zipf function described in [18]. The task that this state performs is controlling to delimit the sessions and analyze results such as number of reproductions per video, duration of the session, and time between sessions.

II.II.IV Idle state of the client node

This state is analogous to the LISTEN state of the server node and its function is to pause when the process is idle. Fig. 2 shows that six states are derived directly from it, which are: pause, forward, RX, rewind, TX_RQ, and Stat. When the process enters this state, it pauses until an interruption occurs that takes it to another state. In addition, it has a default interruption to prevent the creation of infinite loops.

II.II.V Pause state of the client node

This state is used to control playback pauses, which means its task is to transmit a packet to the server which indicates that the pause in the video transmission is being requested. As Fig. 2 shows, this state is entered through the activation of an interruption that is programmed within the TX_RQ state, which determines if the current video playback is going to pause; and then, it defines the position; in other words, the playback time when the pause starts for programming the interruption time.

II.II.VI Forward state of the client node

This state was developed to provide control to the forward option in video playback. Its task is to transmit a request to the server so that forwarding in the reproduction takes place. The way to reach this state is by activating the TIMEOUT4 interruption (see Fig. 2). This interruption is activated in the

TX_RQ state when there is forwarding during playback. As in the pause state, the forward interruption is programmed in a time that depends on the user's behavior.

II.II.VII RX state of the client node

This state is responsible for performing two actions when the client receives a packet from the server. The first one consists of obtaining statistics about the video that is being played and sending messages to the simulator console with important data such as the video played and the number of frames of the video. In Fig. 2, it is observed that to enter this state, the MTS interruption needs to be activated. When a video is played by the client, each packet that is sent from the server, and that contains information about this video, triggers the interruption and causes the client node application process to go from the idle to the RX state. The second action consists of taking a record of the simulation time when information on the requested video begins to be received. This record is used to determine the length of the video played.

II.II.VIII Rewind state of the client node

This state is used to control playback rewinding requested by the client. As observed in Fig. 2, the TIMEOUT5 interruption must be activated to enter this state. As for the interruptions for the pause and forward states, this interruption is programmed in the TX_RQ state.

II.II.IX TX_RQ state of the client node

The TX_RQ state is responsible for performing several important tasks to represent the user behaviour, such as requests to the server for playback, pause, forward, and rewind. In this state, the algorithm is carried out to determine the video to be played, whether or not there are interactivity functions, as well as to determine the number of videos to be played in a session, which is considered to be 30 minutes long [18].

II.II.X Stat state of the client node

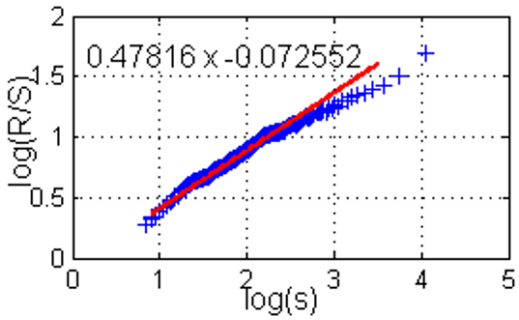
This state was developed with the purpose of keeping statistics about the videos played on each session. It is a forced state, so the code is executed within it; and immediately, the process goes to the END state.

II.II.XI END state

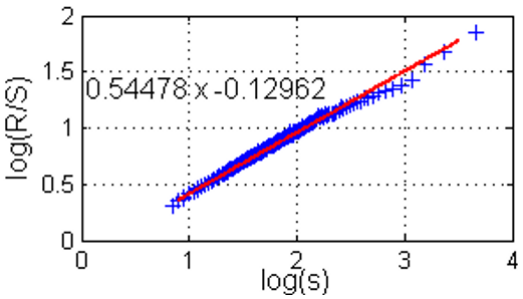
The END state was included in order to serve as a pause to the process when a session has ended and a new one is about to start. As in Fig. 2, this is an unforced state since it requires the activation of the interruption. This interruption is activated thanks to the programming carried out in the RX state when all the videos in a session have finished playing. Furthermore, this state has a default interruption to prevent infinite loops.

III. RESULTS AND DISCUSSION

For the validation process of the proposed scenario, the Hurst exponent (H) was used, which allows to determine whether a series of data follows a self-similar behavior along a period [19]. Thus, the results are compared between the real flow and the simulated model. The method used was the Rescaled Range Analysis (R/S) [20]. Then, the Hurst exponent for video 6 is presented in Fig. 3, as this is the one that presents the most significant difference from the 11 characterized videos between simulated and real traffic. The Hurst exponent is the straight line obtained by graphing log (R/S) vs Log (s).



Real traffic, video 6



Simulated traffic, video 6

Fig. 3. Hurst exponent

Table 1 shows the numerical values for the exponent H for both the real traffic and the simulated traffic for the eleven programmed videos. Obtaining their difference and the average value of that difference between the eleven videos when analyzing the values, it is found that the difference between the real and simulated data for the exponent H is 3.5%. Therefore, it can be concluded that the simulated scenario is statistically similar to the real traffic by 96.5%. considering the self-similarity parameter.

The statistics of the interactivity processes are presented next, which are obtained in a basic scenario made up of a point-to-point connection between a client and a server, in such a way that they allow to observe the events that have occurred as a result of the interactivity processes.

Table 1. Hurst exponent for programmed videos

Videos	1	2	3	4	5	6	7	8	9	10	11
Real	0.46	0.61	0.57	0.54	0.53	0.53	0.48	0.50	0.47	0.51	0.52
Simulated	0.44	0.56	0.64	0.55	0.53	0.54	0.53	0.55	0.54	0.55	0.54

III.I Pause duration

Fig. 4 is composed of three statistics, the first one (upper part) is the "pause duration" statistics, the second one (center) belongs to the "request" statistics, and the third one (lower part) corresponds to the "video duration" statistics. This comparison is made in order to determine the influence of pauses on video playback duration. To do this, the playback of the video in which the pause is taking place is determined, and its duration is evaluated. Thus, when a pause occurs, the duration of the pause will be added to the video duration statistics. Therefore, it is possible to corroborate the information given in the "pause duration" statistics by comparing the original duration of the video with the duration of the paused video.

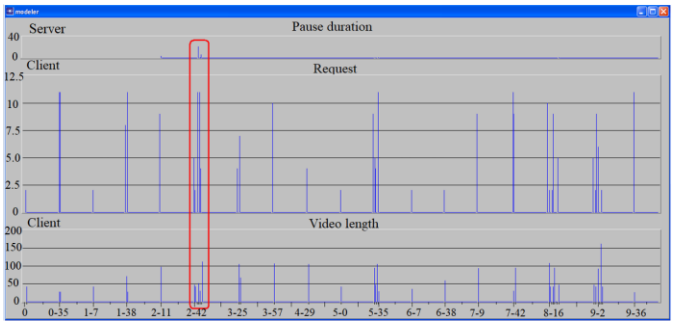


Fig.4. Statistics comparison: pause and video duration, and requests

Similarly, Fig. 4 shows a red box that frames one of the sessions produced during the simulation. It can be seen that this is the sixth session. This particular session is taken as an example because there is a pause of considerable duration with regard to the duration of the videos.

Fig. 5 shows a close-up of session 6 (area framed within the red box in Fig. 4), where it is observed that the Request and Video length statistics is made up of the reproduction of five videos out of the eleven possible. As an example, the data set framed within the black box of Fig. 5 is taken. Since the "requests" statistic shows the videos that the client requires for their reproduction, and the "pause duration" statistics shows the pauses; it can be seen a pause in the playback of video 11. The yellow boxes show the highest point values for the bars framed within the black box. Observing these values, the pause lasts approximately 20 seconds and the total time for playback is 50 seconds, as can be seen in the yellow box at the bottom of Fig. 5 (Video duration section), This means that the length of the video without pauses is 30 seconds. This shows that the model is working correctly in the simulation of pauses.

Another aspect that can be seen in Fig. 5, is that there is a difference of 26 seconds between the register of the "pause duration" and "video duration" statistics. The video duration statistics is recorded every time a video is finished playing, and this is done within the application module of the client node. On the other hand, the pause duration statistics is recorded within the application module of the server node, and it is done every time the client requests a pause. Taking this into account, it can be concluded that 26 seconds elapse between the moment when the pause occurs and when the video is finished playing. If this value is subtracted from the time that the pause lasts, which is 20 seconds, it can be deduced that the pause occurs when there are 6 seconds left to play; in other words, the pause occurs in second 24 of the video since it has a 30 second duration.

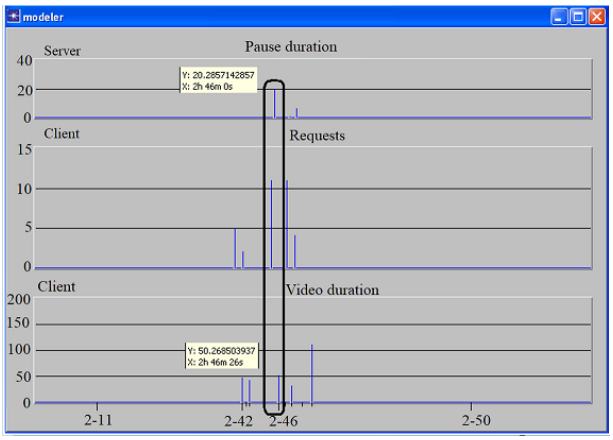


Fig.5. Statistics zoom: pause and video duration, and requests

III.II Forward length in video playback

Fig. 6 presents the effect caused when forwarding appears in the playback duration of the videos selected by the client. Thus, the set of data to be analyzed has been framed in the red box. Since it is forwarding during playback, the time that is forwarded must be subtracted from the total duration of the video that is reproduced; this is because forwarding a video is skipping information that is not being transmitted.

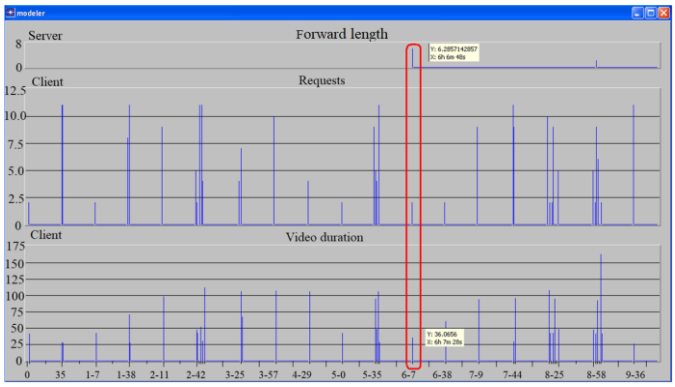


Fig.6. Statistics comparison: forward length, requests, and video duration

Analyzing the red box in Fig. 6 in the "request" section, it can be seen that video 2 is being played, which is 42 seconds long. On the other hand, it also has been forwarded approximately 6.28 seconds. Therefore, if the forward time is subtracted from the duration of the video, the total time for the playback of this video should be approximately 36 seconds. In the yellow box of the "video duration" statistics of the videos in Fig. 6, the duration for video 2 was approximately 36.1 seconds. So, it is concluded that the forward simulation in the model is working correctly.

III.III Rewind length

Fig. 7 takes the "request" and "video duration" statistics to observe the effect of rewinding during playback. When there is a rewind in the reproduction of a video, there is information that is repeated, which means, the server transmits the same information twice. Therefore, on this occasion, finding the total playback time of a video in which a rewind has occurred, the time of the rewind length must be added to the total time of the video duration.

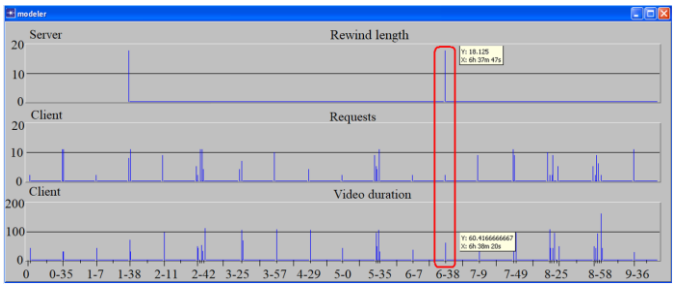


Fig.7. Statistics comparison: rewind length, requests, and video duration

The data for the analysis is found in the red box in Fig. 7. The yellow boxes show the values for the highest points of the framed lines for the "rewind length" and the "video duration". As it can be seen, the request for video 2 is being made, which has a duration of 42 seconds. On the other hand, the rewind length produced is approximately 18 seconds. If this time is added to the total duration of video 2, the time that the client must use to play the video is approximately 60 seconds as Fig. 7 shows. So, it is concluded that the rewind simulation in the model is working correctly.

IV. CONCLUSIONS

This paper has defined a scenario that allows to represent the behavior of the traffic between a client and a server. The simulated scenario responds to an event-oriented model, which considers conditions that must be satisfied for each type of event to be activated; and consequently, their respective actions. For doing this, the FSMs were used being the programming of the scenario the one in charge of controlling actions such as forwarding, rewinding, and pausing generated by clients.

For the conformation of the scenario, one must start from existing models in the simulation environment for the adaptation or modification of their nodes, processes, and states, reprogramming them and developing the interactivity functionalities. This is necessary whether generating a real representation of the services or requesting the representation in the simulated scenario of personalized characteristics, in this case, those corresponding to the interactivity processes of the real environment such as pause, forward, and rewind.

Including the process of self-similarity within the validation process of the simulated scenario brings an important contribution. For this reason, the implementation of the statistical method of rescaled range (R/S) was used. Then, it was found that the simulated scenario, considering the self-similarity parameter, is statistically similar to the real traffic in more than 96%.

Acknowledgments

To the university program "Teaching and Research with OPNET". To the Multimedia Distribution Systems group DMMS of the University of Oviedo. The authors thank the University of Quindío and the University of Cartagena for the support provided in the development of this research.

REFERENCES

[1] K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," in 2017 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017, Jun. 2017, pp. 68–73, doi: 10.1109/FMEC.2017.7946410.

[2] J. Lozano, A. Castro, B. Fuentes, J. M. González, and Á. Rodríguez, "Adaptive QoE measurement on videostreaming IP services," 2011, Accessed: Jun. 04, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/6103992.

[3] T. M. Hautala, I. Suliman, J. J. Lehtomäki, and T. Saarinen, "Performance evaluation of videostreaming on a heterogenous multihop mobile network," in IEEE Vehicular Technology Conference, 2004, vol. 59, no. 5, pp. 2744–2747, doi: 10.1109/vetecs.2004.1391420.

[4] H. Yahyaoui, S. Aidi, and M. F. Zhani, "On Using Flow Classification to Optimize Traffic Routing in SDN Networks," Jan. 2020, doi: 10.1109/CCNC46108.2020.9045216.

[5] H. Thiri Zaw and A. Htein Maw, "Traffic management with elephant flow detection in software defined networks (SDN)," Int. J. Electr. Comput. Eng., vol. 9, no. 4, pp. 3203–3211, 2019, doi: 10.11591/ijece.v9i4. pp 3203-3211.

[6] A. Ceco and S. Mrdovic, "Test Bed for Network Protocols Optimization," 2018, doi: 10.1109/TELFOR.2018.8611846.

[7] S. González, W. Castellanos, P. Guzmán, P. Arce, and J. C. Guerri, "Simulation and experimental testbed for adaptive video streaming in ad hoc networks," Ad Hoc Networks, vol. 52, pp. 89–105, Dec. 2016, doi: 10.1016/j.adhoc.2016.07.007.

[8] S. Kiani Mehr, P. Jogalekar, and D. Medhi, "Moving QoE for monitoring DASH video streaming: models and a study of multiple mobile clients," J. Internet Serv. Appl., vol. 12, no. 1, pp. 1–26, Dec. 2021, doi: 10.1186/s13174-021-00133-y.

[9] A. Dias, A. B. Reis, and S. Sargento, "Improving the QoE of OTT Multimedia Services in Wireless Scenarios," in Proceedings - IEEE Symposium on Computers and Communications, Jun. 2019, vol. 2019-June, doi: 10.1109/ISCC47284.2019.8969709.

[10] T. Oliveira and S. Sargento, "QoE-based Load Balancing of OTT Video Content in SDN Networks," in Proceedings - IEEE Symposium on Computers and Communications, Jun. 2019, vol. 2019-June, doi: 10.1109/ISCC47284.2019.8969720.

[11] V. S. Elagin, I. A. Belozertsev, B. S. Goldshtein, A. V. Onufrienko, and A. G. Vladyko, "Models of QOE ensuring for OTT services," May 2019, doi: 10.1109/SOSG.2019.8706748.

[12] G. Cheung, Z. Liu, Z. Ma, and J. Z. G. Tan, "Multi-stream switching for interactive virtual reality video streaming," Proceedings - International Conference on Image Processing, ICIP, 2018. .

[13] K. Fukava, K. Mori, K. Imamura, Y. Matsuda, T. Matsumura, and S. Mochizuki, "Design and Implementation of Ultra-Low-Latency Video Encoder Using High-Level Synthesis," Dec. 2019, doi: 10.1109/ISPACS48206.2019.8986365.

[14] L. Rossetto et al., "Interactive Video Retrieval in the Age of Deep Learning - Detailed Evaluation of VBS 2019," IEEE Trans. Multimed., vol. 23, pp. 243–256, 2021, doi: 10.1109/TMM.2020.2980944.

[15] W. Y. Campo Muñoz, H. Fabio Bermudez, and E. Astaiza Hoyos, "Characterization of traffic of the video streaming service based on lexical analyzers," Ingeniare, vol. 26, no. 3, pp. 448–458, 2018, doi: 10.4067/S0718-33052018000300448.

[16] W. Y. Campo-Muñoz, E. Astaiza-Hoyos, and L. F. Muñoz-Sanabria, "Modelado de tráfico del servicio de video bajo demanda mediante NS-3," DYNA, vol. 84, no. 202, pp. 55–64, Jul. 2017, doi: 10.15446/dyna.v84n202.61650.

[17] W. Y. Campo-Muñoz, G. E. Chanchí-Golondrino, and M. C. Camacho-Ojeda, "Uso de técnicas de emulación en la construcción de un modelo de tráfico para un servicio multimedia," Ingeniería, investigación y tecnología, 2017.

[18] M. T. González-Aparicio, R. García, J. L. Brugos, X. G. Pañeda, D. Melendi, and S. Cabrero, "Measuring temporal redundancy in sequences of video requests in a News-on-Demand service," Telemat. Informatics, vol. 31, no. 3, pp. 444–458, Aug. 2014, doi: 10.1016/j.tele.2013.10.006.

[19] G. Millán, R. Osorio-Comparán, and G. Lefranc, "Preliminaries on the Accurate Estimation of the Hurst Exponent Using Time Series," Mar. 2021.

[20] J. Beran, Statistics for long-memory processes. CRC Press, 2017.