Volume: 11 Issue: 05 Pages: 4386-4392(2020) ISSN: 0975-0290

Reinforcement Learning in DASH

Koffka Khan
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: koffka.khan@gmail.com
Wayne Goodridge
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: wayne.goodridge@sta.uwi.edu.com

ABSTRACT

Reinforcement Learning is an important class of optimization which has recently been used in the area of dynamic adaptive streaming over HTTP (DASH). Though DASH is very popular method of video delivery in recent years it is plagued with problems when multiple players share a bottleneck link. Thus, this area has become a very active area of research. Two works which implement Reinforcement Learning in DASH are selected and their performances compared against the Conventional DASH player. It is shown that SDP works well for various network conditions.

Keywords – **Reinforcement Learning; optimization; DASH; video; bottleneck; network.**

Date of Submission: Aug 22, 2019 Date of Acceptance: Dec 23, 2019

I. INTRODUCTION

At the end of 1979, we first focused on what is now known as strengthening learning. This was simply the idea of a learning system that wants something, adapting its behavior to maximize a particular signal from its environment. While reinforcement learning had clearly motivated some of the earliest computational learning studies, most of these researchers had gone on to other things, such as classification of patterns, supervised learning, and adaptive control, or had completely abandoned learning [33]. As a result, relatively little attention was paid to the special issues involved in learning how to get something from the environment. Since then, the field has come a long way, developing and maturing in various directions.

Reinforcement learning in machine learning, artificial intelligence, and neural network research has gradually become one of the most active research areas [36]. The contributions to establishing and developing relationships to the theory of optimum control and dynamic programming were particularly important. The general issue of learning to attain objectives from interaction is still far from being solved, but our knowledge of it has been considerably enhanced. We can now position constituent concepts, such as temporal-difference learning [8], dynamic programming [6], and function approximation [4], within a meaningful standpoint to the entire problem.

The Internet traffic [37] due to video applications is increasing thanks to the diffusion of new devices such as tablets, smart phones, Smart TVs which are connected to the Internet through broadband wired and wireless connections. Video streaming, in the form of user-generated video distribution – such as in the case of Youtube – or to deliver movies and TV series – such as in the case of Netflix – is the application that is driving this growth.

Even though the TCP has been regarded in the past as unsuitable to transport video flows, today videos are streamed through HTTP with the TCP, and web browsers implementing the HTML5 [15] standard are now able to reproduce videos without the use of any external plug-in.

The most common approach to distribute video is the progressive download streaming: the video content is encoded at a given bitrate and it is sent to the user as any other file through a HTTP connection [27]. The issue with this approach is that, even though TCP connections [2] are elastic, the video content transported through the TCP socket is not elastic; thus, a persistent mismatch between the encoding bitrate and the net-work available bandwidth may result in playout interruptions. Another drawback of such an approach is that mobile devices, such as tablets or smart phones, may not be able to play a high definition video due to their limited computational resources.

To tackle these issues, the video content must be made adaptive [24], [18], [23], [22], [20], [21]. Among the approaches proposed so far, the stream-switching is gaining momentum due to its deployment and implementation simplicity. With this approach, the video is encoded at different bitrates and resolutions, the video levels, and the encoded videos are logically or physically divided into segments of fixed durations. The stream-switching controller decides, for each video segment, the video level to be streamed, see Figure 1.

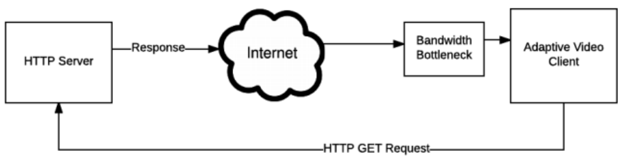

Fig. 1. Conventional Adaptive Streaming

From the architectural standpoint, two different approaches can be used to implement a stream-switching algorithm: the client-side, that places the controller at the client, and the server-side, that implements the controller at the server. It has been shown that the client-side algorithms proposed so far generate an on-off traffic pattern at steady-state that can lead to unfairness when many video flows share a bottleneck (see Figure 2) [14], [25], [16]. Moreover, in [1] it has been established that the adaptive video players of three popular video streaming services were not able to get a fair share when coexisting with a TCP greedy flow. Authors name this issue the "downward spiral effect" and ascribe its cause to the on-off traffic pattern described above; authors suggest increasing the segment size and filter bandwidth estimates to tackle this issue.

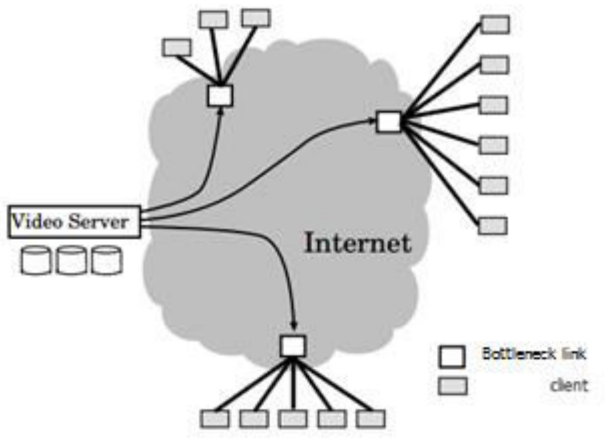

Fig. 2. Bottleneck Links in video distribution

In the presence of competing HTTP-based adaptive streaming (HAS) clients the TCP throughput does not always faithfully represent the fair-share bandwidth [25]. Three performance issues that can take place when two or more adaptive streaming players share a network bottleneck and compete for available bandwidth are instability, unfairness and utilization [25] It is shown that in the case of two competing video flows Adaptive video streaming players provide a received video rate that oscillates around the fair share, but with an increased number of video level switches [14]. Depending on the temporal overlap of the ON-OFF [16] periods among competing players, they may not estimate their fair share correctly. In the case where both players overestimate their fair share, they may request a video representation with a higher bitrate than the fair share, which causes network congestion. Consequently, the players measure that their TCP throughput is lower than their previous fair share estimate, and so switch down to a lower video bitrate level. This creates a repeating oscillatory scenario, so inducing instability.

A scenario can also occur where some players are requesting chunks with lower bitrates than the other players. This can occur as some players observe a throughput lower than the fair share, while others observe a throughput that is more than the fair share. This means that some players overestimate its fair share. When some players overestimate their fair share, it can be that the system of players converge to a stable equilibrium, but unfair [1]. This occurs as the players with the larger fair share estimates request higher bitrate video levels. Even in the case where two players estimate their fair share correctly, bandwidth underutilization can still be prevalent. This occurs as both players request the same lower video bitrate level, which causes underutilization, even though stability and fairness still exist. In reality, several other factors can play an important role in the appearance and extent of instability, unfairness and underutilization, such as the exact player adaptation algorithm, TCP dynamics, bandwidth fluctuations, and the variability of the video encoding rate.

We group these problems into three categories: The first relates to the stability of the players in terms of requested bitrates and video quality. The second is the unfairness among competing players. The third is the potential bandwidth underutilization when multiple adaptive players compete. They are described as follows:

Instability: The instability [25] for player i at time t is given in Equation below, where $w(d) = k - d$ is a weight function that puts more weight on more recent samples. k is selected as 20 seconds.

$$Instability = \frac{\sum_{d=0}^{k-1} \left| r_{i,t-d} - r_{i,t-d-1} \right| * w(d)}{\sum_{d=0}^{k-1} r_{i,t-d} * w(d)}$$

Unfairness: Let $JainFair_t$ be the Jain fairness index (cf. Equation 10) calculated on the average received rates [14], r_i, (cf. Equation below) at time t over all players. The unfairness at time t is defined as $\sqrt{1 - JainFair_t}$. A lower value implies a fairer allocation.

$$r_i = \frac{downloaded\ bytes}{time\ interval}$$

$$JFI = \frac{(\sum_{i=1}^{n} r_i)^2}{n \sum_{i=1}^{n} r_i^2}$$

The utilization metric [16] is defined as the aggregate throughput during an experiment divided by the available bandwidth in that experiment (cf. Equation below, where tp_i is the throuput at time i and bw is the experimental available bandwidth).

$$Utilization = \frac{\sum_{i=0}^{n-1} tp_i}{bw}.$$

A discussion on Reinforcement Learning is given in Section II. DASH approaches using Reinforcement Learning is explored in section III. Experimental setup is given in section IV. Results are given in section V and finally, the conclusion is given in Section VI.

II. Reinforcement Learning

Reinforcement learning is learning what to do — how to link scenarios to actions — to maximize numerical reward amount(s). The learner is not told what actions to take, as in most forms of machine learning, but instead, by trying them, he must find out which actions give the most reward. In the most exciting and difficult cases, scenarios can influence not only the present reward but also the next scenario and all the rewards that follow. The fundamental concept is merely to capture the most significant elements of the actual problem that a learning agent is confronting in order to attain a objective. The purpose of the agent is to include only these three aspects — sensation, action, and goal — in their easiest possible forms without trivializing any of them, see Figure 3.

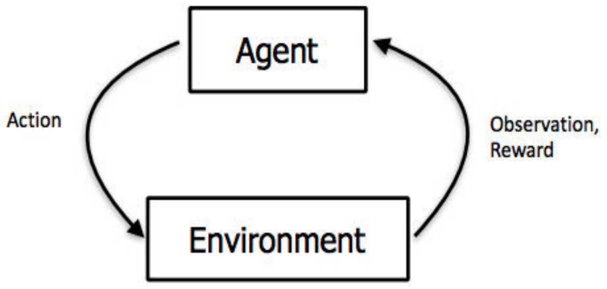

Fig. 3. Reinforcement Learning

The crux of reinforcement learning is capturing essential parts of a problem, when a learning agent interacts with its surroundings, while trying to accomplish a goal. The agent should be able to "sense" its surroundings and make decisions, on the conditions it observes. The agent desires to complete one or more goals. In achieving its goals, the agent attempts to maximize some reward given the actions it takes. Thus, the agent discovers most fruitful actions by checking the rewards for each goal. On a stochastic task or assignment, each action attempt is done multiple times to gain a reliable expected reward estimate [33] considers the whole problem of a goal-directed agent interacting with its surroundings. Reinforcement learning agents: (a) have explicit goals, (b) can sense aspects of their environments and (c) can choose actions to influence their environments. The trade-off between exploration and exploitation is one of the challenges that arise in reinforcement learning and not other types of learning. A reinforcement learning agent must prefer actions he has attempted in the past and discovered to be efficient in generating reward in order to achieve its goal(s). To achieve reward, the agent must exploit what he already knows, but he must also explore to create choices for better action in the future. The agent must attempt a variety of actions and gradually favor those that seem to be the best. However, each action must be attempted many times in a stochastic assignment to obtain a credible estimate of its anticipated reward. Another key feature of reinforcement learning is that it specifically considers entire problem of engaging with an uncertain environment by a goal-directed agent [5]. All reinforcement learning agents have specific objectives, can sense elements of their settings and can choose activities to impact their environments.
Four main sub-elements of the reinforcement learning system can be identified: a policy, a reward function, a value function and, optionally, an environment model [3]. A policy defines the behavior of the learning agent at a particular time. A policy is a mapping of actions to be taken in those states from perceived environmental states. Policies can generally be stochastic. A reward function describes the target of a problem of reinforcement learning. It maps each perceived state (or state-action pair) of the environment to a single number, a reward, indicating that that state's intrinsic desirability. The sole goal of a reinforcement learning agent is to maximize the combined long-term reward it gets. The value function specifies the long-term usefulness. The value of a state is the total percentage of reward that an agent can expect to collect from that state into the future. Values indicate the long-term desirability of states and the rewards available in those states after taking into account the states that are likely to follow [7].
Rewards are primary in a sense, while values are secondary as indicators of rewards. There could be no values without rewards, and the only purpose of estimating values is to obtain more reward. These are the values that we are most concerned with when taking decisions and evaluating them. Action choices are made based on value judgments [33]. We are looking for actions that bring the highest value, not the highest reward, because these actions receive the greatest reward for us in the long run. The derived amount called value in decision-making and planning is the one we are most interested with. Regretfully, determining values is much harder than determining rewards. The environment basically gives rewards directly, but values must be estimated and re-estimated from the sequences of observations that an agent makes throughout his lifetime. A method for the efficient estimation of values is the most important component of almost all reinforcement learning algorithms. A model of the environment is the fourth and final element of certain reinforcement learning systems. This is something that imitates the environment's behavior. For example, the model could predict the resulting next state and next reward given a state and action. Modern reinforcement learning covers the range from low-level, trial-and-error learning to deliberative, high-level planning [33].

III. REINFORCEMENT LEARNING DASH-BASED APPROACHES

Researchers in [28], [32], [17] apply reinforcement learning in quality selection algorithms for streaming video. In multi-player competitive environments these reinforcement learning approaches are able to achieve fairness. A coordination proxy facilitates co-ordination among players. Thus, these approaches fall under network-assisted DASH. These approaches learn and adapt their policy depending on network conditions, yielding low overhead. [11] progressively maximize a pre-defined QoE-related reward function. By utilizing this action, players are able to learn an optimal request strategy. An autonomous RL-learning agent [26] provides adaptive video streaming in best effort networks. The agent learns an optimal control strategy regarding user-QoE, without the need for implementation of a complex heuristics. Researchers in [9] use an original reinforcement learning approach to develop better adaptation agents. Agents gradually improve, by taking into account, both user's behaviour and usage context.

A. *Method A [35]*

HTTP Adaptive Streaming (HAS) is becoming the de-facto standard for Over-The-Top video streaming. A HAS video consists of multiple segments, encoded at multiple quality levels. Allowing the client to select the quality level for every segment, a smoother playback and a higher Quality of Experience (QoE) can be perceived. Although results are promising, current quality selection heuristics are generally hard coded. Fixed parameter values are used to provide an acceptable QoE under all circumstances, resulting in suboptimal solutions. Furthermore, many commercial HAS implementations focus on a video-on-demand scenario, where a large buffer size is used to avoid play-out freezes. When the focus is on a live TV scenario however, a low buffer size is typically preferred, as the video play-out delay should be as low as possible. Hard coded implementations using a fixed buffer size are not capable of dealing with both scenarios. In this paper, the concept of reinforcement learning is introduced at client side, allowing to adaptively change the parameter configuration for existing rate adaptation heuristics. Bandwidth characteristics are taken into account in the decision process, thus allowing to improve the client's bandwidth-awareness. Focus in this paper is on actively reducing the average buffer filling. Authors show that using the proposed learning-based approach, the average buffer filling can be reduced by 8.3% compared to state of the art, while achieving a comparable level of QoE.

B. *Method B [10]*

HTTP Adaptive Streaming (HAS) is becoming the de facto standard for Over-The-Top (OTT)-based video streaming services such as YouTube and Netflix. By splitting a video into multiple segments of a couple of seconds and encoding each of these at multiple quality levels, HAS allows a video client to dynamically adapt the requested quality during the playout to react to network changes. However, state-of-the-art quality selection heuristics are deterministic and tailored to specific network configurations. Therefore, they are unable to cope with a vast range of highly dynamic network settings. In this letter, a novel Reinforcement Learning (RL)-based HAS client is presented and evaluated. The self-learning HAS client dynamically adapts its behaviour by interacting with the environment to optimize the Quality of Experience (QoE), the quality as perceived by the end-user. The proposed client has been thoroughly evaluated using a network-based simulator and is shown to outperform traditional HAS clients by up to 13% in a mobile network environment.

IV. EXPERIMENTAL SETUP

A virtual network is setup on the same host machine creating a custom emulation framework. Our setup consists of client players, video servers, and a bottleneck link. The server resides on a Windows 10 machine. All experiments are performed on a Windows 10 client with an Intel(R) Core(TM)i7-5500U CPU 2.40GHz processor, 16.00 GB physical memory, and an Intel(R) HD Graphics processor. It serves video data to the client(s) who are on a Ubuntu operating system hosted on VMware. The virtual machine is allocated 12GB of physical memory.

TAPAS [13] is installed on Ubuntu 15.04 Linux. The TAPAS Adaptive Video Controller client makes different video segment bitrate level requests to the Apache server. TAPAS allow multiple instances of the player to be created enabling multi-client scenarios. This work involves the interaction between adaptive streaming algorithm at the controller and TAPAS player (cf. Figure 6). All traffic between clients and servers go through the bottleneck, which uses VMware settings which allow bandwidth limits to be set during the experiment. TAPAS support both the HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) format. Algorithms that uses Method A and Method B was tested and shown to work on both MPEG-DASH [34], and Apple HTTP Live Streaming (HLS). This makes it useful for video on demand (VOD) [29] and live streaming [12], for example, real-time video chats. However, the MPEG-DASH standard is used for testing in this research paper, because it makes the experiments more comparable to the ones in the research literature, for example, [30].

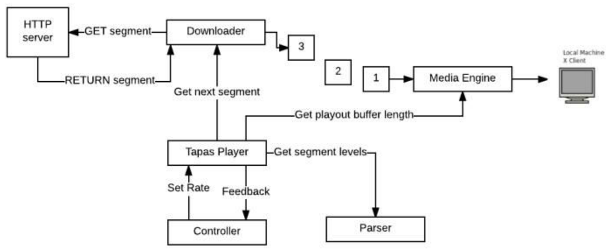

Fig. 4. TAPAS adaptive player

The ten-minute-long MPEG-DASH video sequence "Elephant's Dream"[1] is encoded at twenty different bitrates, between 46 Kbps to 4200Kbps and five different resolutions, between 320x240 to 1920x1080, is used to run the experiments (cf. Table II). The video is encoded at 24 frames per second (fps) using the AVC1 codec [31]. Fragment duration of 2s is used and is recorded in the mpd playlist accordingly. All the DASH files (.m4s fragments and .mpd playlists) are placed on the Apache server. We implemented three client-side algorithms in the TAPAS controller. The conventional approach is present by default and is used as a baseline in which to compare against other algorithms. TAPAS is lightweight in built, thus allowing the same receiving host to run a large number of separate video player instances at the same time at different command line interfaces. Thus, it allows the multi-client scenarios which are essential to the work in this paper.

V. RESULTS

The first experiment illustrates five players competing at a 20Mbps bottleneck link. Table 1 gives the results. Method A outperforms Method B and the Conventional.

Table 1

	Method A	Method B	Conventional
Utilization	0.94	0.91	0.68
Unfairness	0.015	0.021	0.124
Instability	0.112	0.132	0.311

The second experiment illustrates five players competing at a 20Mbps bottleneck link and stopping or pausing during the experiment. Table 2 gives the results. Method A outperforms Method B and the Conventional.

Table 2

	Method A	Method B	Conventional
Utilization	0.95	0.90	0.71
Unfairness	0.009	0.028	0.136
Instability	0.123	0.137	0.356

The third experiment illustrates five players competing at a 100Mbps bottleneck link with increasing number of players up to 20. Table 3 gives the results. Method A outperforms Method B and the Conventional.

Table 3

	Method A	Method B	Conventional
Utilization	0.87	0.83	0.59
Unfairness	0.024	0.030	0.173
Instability	0.135	0.164	0.384

The fourth experiment illustrates five players competing at a 20Mbps bottleneck link in bandwidth varying conditions. Table 4 gives the results. Method A outperforms Method B and the Conventional.

Table 4

	Method A	Method B	Conventional
Utilization	0.85	0.81	0.55
Unfairness	0.029	0.038	0.398
Instability	0.161	0.196	0.401

The fifth and final experiment illustrates five players competing at a 20Mbps bottleneck viewing different videos (note in previous experiments the same video was used by all players while in these additional videos were used: Tears of Steel, Sintel, Big Buck Bunny and The Swiss Account). Table 5 gives the results. Method A outperforms Method B and the Conventional.

Table 5

	Method A	Method B	Conventional
Utilization	0.85	0.81	0.55
Unfairness	0.029	0.038	0.398
Instability	0.161	0.196	0.401

VI. Conclusion

Reinforcement Learning is an important class of optimization which has recently been used in the area of dynamic adaptive streaming over HTTP (DASH). Though DASH is very popular method of video delivery in recent years it is plagued with problems when multiple players share a bottleneck link. Thus, this area has become a very active area of research. Two works which implement Reinforcement Learning in DASH are selected and their performances compared against the Conventional DASH player. It is shown that Reinforcement Learning works well for various network conditions. However, one method outperforms the others in the experiments conducted. In future work multipath schemes [19] should be used together with reinforcement learning techniques to better DASH-based approaches.

References

[1] Akhshabi, Saamer, Lakshmi Anantakrishnan, Ali C. Begen, and Constantine Dovrolis. "What happens when HTTP adaptive streaming players compete for bandwidth?." In Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, pp. 9-14. ACM, 2012.
[2] Allman, Mark, Vern Paxson, and Wright Stevens. "TCP congestion control." (1999).
[3] Aranibar, Dennis Barrios, and Pablo Javier Alsina. "Reinforcement learning-based path planning for autonomous robots." In EnRI-XXIV Congresso da Sociedade Brasileira de Computaç ao, p. 10. 2004.
[4] Baird, Leemon. "Residual algorithms: Reinforcement learning with function approximation." In Machine Learning Proceedings 1995, pp. 30-37. Morgan Kaufmann, 1995.
[5] Barto, Andrew G. "Intrinsic motivation and reinforcement learning." In Intrinsically motivated learning in natural and artificial systems, pp. 17-47. Springer, Berlin, Heidelberg, 2013.
[6] Bellman, Richard. "Dynamic programming." Science 153, no. 3731 (1966): 34-37.
[7] Boyan, Justin A., and Andrew W. Moore. "Generalization in reinforcement learning: Safely approximating the value function." In Advances in neural information processing systems, pp. 369-376. 1995.

[8] Bradtke, Steven J., and Andrew G. Barto. "Linear least-squares algorithms for temporal difference learning." Machine learning 22, no. 1-3 (1996): 33-57.

[9] Charvillat, Vincent, and Romulus Grigoraş. "Reinforcement learning for dynamic multimedia adaptation." Journal of Network and Computer Applications 30, no. 3 (2007): 1034-1058.

[10] Claeys, Maxim, Steven Latre, Jeroen Famaey, and Filip De Turck. "Design and evaluation of a self-learning HTTP adaptive video streaming client." IEEE communications letters 18, no. 4 (2014): 716-719.

[11] Claeys, Maxim, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, and Filip De Turck. "Design and optimisation of a (FA) Q-learning-based HTTP adaptive streaming client." Connection Science 26, no. 1 (2014): 25-43.

[12] De Cicco, Luca, Saverio Mascolo, and Vittorio Palmisano. "Feedback control for adaptive live video streaming." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 145-156. ACM, 2011.

[13] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms." In Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming, pp. 1-6. ACM, 2014.

[14] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)." In 2013 20th International Packet Video Workshop, pp. 1-8. IEEE, 2013.

[15] Hoy, Matthew B. "HTML5: a new standard for the Web." Medical reference services quarterly 30, no. 1 (2011): 50-55.

[16] Jiang, Junchen, Vyas Sekar, and Hui Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." IEEE/ACM Transactions on Networking (ToN) 22, no. 1 (2014): 326-340.

[17] Jiang, Nan, Viswanathan Swaminathan, and Sheng Wei. "Power evaluation of 360 vr video streaming on head mounted display devices." In Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 55-60. ACM, 2017.

[18] Khan, Koffka, and Wayne Goodridge. "B-DASH: broadcast-based dynamic adaptive streaming over HTTP." International Journal of Autonomous and Adaptive Communications Systems 12, no. 1 (2019): 50-74.

[19] Khan, Koffka, and Wayne Goodridge. "Energy aware Ad-Hoc on demand multipath distance vector routing." International Journal of Intelligent Systems and Applications 7, no. 7 (2015): 50-56.

[20] Khan, Koffka, and Wayne Goodridge. "Future DASH Applications: a Survey." International Journal of Advanced Networking and Applications 10, no. 2 (2018): 3758-3764.

[21] Khan, Koffka, and Wayne Goodridge. "QoE in DASH." International Journal of Advanced Networking and Applications 9, no. 4 (2018): 3515-3522.

[22] Khan, Koffka, and Wayne Goodridge. "SAND and Cloud-based Strategies for Adaptive Video Streaming." International Journal of Advanced Networking and Applications 9, no. 3 (2017): 3400-3410.

[23] Khan, Koffka, and Wayne Goodridge. "Server-based and network-assisted solutions for adaptive video streaming." International Journal of Advanced Networking and Applications 9, no. 3 (2017): 3432-3442.

[24] Khan, Koffka, and Wayne Goodridge. "S-MDP: Streaming with Markov Decision Processes." IEEE Transactions on Multimedia (2019).

[25] Li, Zhi, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. "Probe and adapt: Rate adaptation for HTTP video streaming at scale." IEEE Journal on Selected Areas in Communications 32, no. 4 (2014): 719-733.

[26] Menkovski, Vlado, and Antonio Liotta. "Intelligent control for adaptive video streaming." In 2013 IEEE international conference on consumer electronics (ICCE), pp. 127-128. IEEE, 2013.

[27] Mogul, Jeffrey C. The case for persistent-connection HTTP. Vol. 25, no. 4. ACM, 1995.

[28] Petrangeli, Stefano, Maxim Claeys, Steven Latré, Jeroen Famaey, and Filip De Turck. "A multi-agent Q-Learning-based framework for achieving fairness in HTTP Adaptive Streaming." In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-9. IEEE, 2014.

[29] Ramesh, Sridhar, Injong Rhee, and Katie Guo. "Multicast with cache (mcache): An adaptive zero-delay video-on-demand service." IEEE Transactions on Circuits and Systems for video technology 11, no. 3 (2001): 440-456.

[30] Sodagar, Iraj. "The mpeg-dash standard for multimedia streaming over the internet." IEEE multimedia 18, no. 4 (2011): 62-67.

[31] Stockhammer, Thomas. "Dynamic adaptive streaming over HTTP--: standards and design principles." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 133-144. ACM, 2011.

[32] Supancic III, James, and Deva Ramanan. "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning." In Proceedings of the IEEE International Conference on Computer Vision, pp. 322-331. 2017.

[33] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[34] Thang, Truong Cong, Quang-Dung Ho, Jung Won Kang, and Anh T. Pham. "Adaptive streaming of audiovisual content using MPEG DASH." IEEE Transactions on Consumer Electronics 58, no. 1 (2012): 78-85.

[35] van der Hooft, Jeroen, Stefano Petrangeli, Maxim Claeys, Jeroen Famaey, and Filip De Turck. "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients." In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 131-138. IEEE, 2015.

[36] Wiering, Marco, and Martijn Van Otterlo. "Reinforcement learning." Adaptation, learning, and optimization 12 (2012): 3.

[37] Williamson, Carey. "Internet traffic measurement." IEEE internet computing 5, no. 6 (2001): 70-74.

Biographies and Photographs

Koffka Khan received the M.Sc., and M.Phil. degrees from the University of the West Indies. He is currently a PhD student and has up-to-date, published numerous papers in journals & proceedings of international repute. His research areas are computational intelligence, routing protocols, wireless communications, information security and adaptive streaming controllers.

Wayne Goodridge is a Lecturer in the Department of Computing and Information Technology, The University of the West Indies, St. Augustine. He did is PhD at Dalhousie University and his research interest includes computer communications and security.