Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

Performance of Q-Learning algorithms 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

Q-Learning is an important class of stochastic 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 Q-Learning in DASH are selected and their performances compared against the Conventional DASH player. It is shown that Q-Learning works well for various network conditions.

Keywords – **Q-Learning; stochastic; optimization; DASH; video; bottleneck; network.**

Date of Submission: Sep 09, 2019 Date of Acceptance: Oct 22, 2019

I. INTRODUCTION

Computing devices capable of displaying high definition video have become commonplace, and high-speed wireless networks are available in most populated areas in some developing and developed countries [10]. An important application of these technologies is video streaming at home or cooperate settings. Consequently, the number of video streaming providers targeting the video streaming market has exploded.

On-demand adaptive video streaming [17] over unreliable networks with limited bandwidth includes various scenarios involving many adaptive players sharing a bottleneck link. These bottleneck links become more unreliable when players

compete in the presence of TCP long-lived flows [55], when there is time-varying bandwidth present, when players start, stop or pause, when players download different videos, and when more players start to download videos. Achieving a satisfactory QoE for the user is intractable and involves trade-offs among players.

In the past ten years, the Internet has become a standard medium for multimedia delivery. The Internet today is prolific with applications that use video data. As a result, a number of video streaming services have been established over the past decade. Old-fashioned RTSP streaming [50] eventually evolved to

HTTP-based streaming protocols. This shift lead to a better user-perceived Quality of Experience (QoE) [38], [59], [31]. Progressive download uses HTTP [14] as a protocol and succeeded traditional streaming. Today, HTTP adaptive video streaming has become the de facto standard. Real-time multimedia delivery has tight latency constraints, and data arriving too late is essentially useless. To create the illusion of motion, video frames should be played between 24-30 frames per second [2]. Efficient media compression creates interdependence between packet contents and codecs, so packet losses and late arrivals of video data can be detrimental.

Since the Internet does not provide a constant, guaranteed bandwidth for the video stream, the network can only support the video bitrate on a best-effort basis [57]. If the network bandwidth is not sufficient to support the video bitrate, then the decoder at the client-end starts to consume the video data at a greater rate than at which new data is being received from the network. The decoder eventually runs out of video data to decode, which results in a screen freeze (video stalls or rebuffering events). In order to avoid this consequence without having to introduce costly and complex guaranteed bandwidth mechanisms, playout buffers and stream switching solutions has become industry standards [30].

Optimization techniques [42], [54], [4], [21] have been used in an effort to help adaptive video players select appropriate segments during the streaming process. One very promising technique is Q-learning [41], [18], [47]. Q-learning is a model-free reinforcement learning algorithm. Q-learning's objective is to learn a policy that informs an agent what to do under what conditions. It does not involve a model and, without having adaptations, it can address problems with stochastic transitions and rewards. Q-learning finds an optimal policy for any finite Markov decision process (FMDP) [44] in the sense that it maximizes the expected value of the total reward over any and all successive steps, starting from the current state. Q-learning can define an optimal policy for any specified FMDP given an infinite period of exploration and a partially random policy. "Q" refers to the function returning the reward used to provide the reinforcement and can be said to represent the "quality" of an action made in a specified state.

A discussion on solving Q-learning casted problems is given in Section II. Then the literature is explored to extract some Q-learning examples in DASH in Section III. DASH multiplayer competition is discussed in section IV. Implemented Q-learning streaming algorithms in this work are given in section V. Our experimental setup is illustrated in section VI while results are given in section VII. Finally, the conclusion is given in Section VIII.

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

II. Q-LEARNING

Q-learning is a type of model-free reinforcement learning [56]. It is a technique of providing data based on value to tell what action an agent should take. It gives agents the ability to learn how to behave optimally in the fields of Markov experiencing the effects of actions without requiring them to construct domain maps. Learning is similar to the technique of temporal differences (TD) used by Sutton [34], [20]. An agent attempts to take action in a specific state and assesses its implications with regard to the instant reward or punishment it gets and its estimate of the value of the state to which it is being applied. By constantly testing all actions in all states, it learns the best overall, measured by a long-term discounted reward. It is regarded off-policy because the q-learning function learns from actions outside the present policy, such as taking random actions, so there is no need for a policy. In particular, q-learning aims at learning a strategy that maximizes the overall reward. Q-learning is an off policy algorithm that aims to discover the best action to take considering the present situation.

Consider a computing agent moving around some discreet, finite world, selecting one from every step of a finite set of actions. The world is a Markov process regulated by the agent as a controller. At step n, the agent is equipped to register the world's state, an action can be selected accordingly. The agent gets a probabilistic reward, the mean value of which relies only on the state and action, and the state of the world is likely to change to under the law. The agent's job is to determine an optimal policy that maximizes the complete discounted reward anticipated. By discounted reward, we imply that the rewards received in future steps are less than the rewards now received. Under a policy, the value of the state is immediately calculated to carry out the action recommended by the policy. It then moves to a valuable state, with the likelihood connected with that specific policy. DP's theory [15], [58], [22], [43] assures us that at least one stationary policy is optimal. DP offers a number of techniques to calculate the state value and an optimal stationary policy. The challenge facing a Q-learner, is to establish an optimal stationary policy without discovering these values previously. The Q value is the anticipated discounted reward for carrying out an action in a given state and subsequently following the resulting policy. The purpose of Q-learning is to approximate the Q values for an optimal policy. The experience of the agent in Q-learning consists of a series of separate stages or events. The agent in an event:

1. observes its present state
2. selects and executes an action
3. observes the following state
4. receives an immediate payoff
5. use the learning factor to adjust its prior Q values

Realize that values may not correctly represent the policy they implicitly define in the early phases of learning.

When q-learning is done, we generate what is called a q-table or matrix that follows [state, action] formation and initialize our values to zero. After an event, we will update and store our q-values. This q-table is a reference table for our agent to choose the best q-value-based action. The next step is simply for the agent to interact with the environment and make updates in our q-table Q[state, action] to the state action pairs. An agent interacts in 1 of 2 ways with environment. The first is to use the q-table as a reference and view for a specified state all feasible actions. Then the agent chooses the action based on those actions ' max value. This is known as exploiting since we make a choice using the data we have at our disposal. The second way to behave (do an action) is by acting randomly. This is called exploring. We pick an action at random instead of choosing actions based on the max future reward. It is important to act randomly because it allows the agent to explore and discover new states that can not otherwise be chosen during the phase of exploitation. Using an epsilon variable you can balance exploration/exploitation and set the importance of how many times you want to explore vs exploit.

After each move or action, the updates take place and end when an event is finished (reaching some terminal point). For instance, a terminal state can a state of fulfilling some required goal. The agent will not learn much after a single episode, but it will eventually converge and discover the ideal q-values or q-star with sufficient exploration (steps and events). Thus, another way of writing the agent in an event are as follows:

1. Agent starts in a state takes an action and receives a reward
2. Agent selects action by referencing Q-table with highest value OR by random
3. Update q-values

Simply define the learning rate as how much you accept the fresh value versus the old value. In order to balance instant and future reward, a discount factor is used. Reward is the value obtained at a specified state after the completion of a particular action. A reward may occur at any time step or only at the time step of the terminal action. Usually a programming library is used to take the highest future reward and apply it to the present state's reward. What this does is impact the current action by the possible future reward. Thus, q-learning assigns future rewards to present actions to assist the agent in selecting the greatest return action in any specified state.

III. QL IN DASH

A solution for wireless adaptive multimedia QoS provisioning is given in [12]. It utilizes Q-learning to model its solution. Researchers formulate the constrained Markov decision problem using call admission control and bandwidth adaptation. Q-learning does not require the explicit state transition model to solve the Markov decision problem. Therefore, researchers apply more general and realistic assumptions to the underlying system model for this approach, than in previous schemes. A Q-Learning based formalism to optimize Quality of Service (QoS) scheduling is proposed in [40]. A HAS player dynamically learns the optimal behavior corresponding to the current network environment using Q-Learning [16]. Researchers use a tunable reward function, which considers multiple aspects of video quality [7]. However, the performance in a variable networking environment was

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

unsatisfactory. In order to optimize QoE, the DASH player dynamically learns the optimal behaviour, corresponding to the current network environment. Researchers in [3], model the adaptive streaming problem as a Partial Observable Markov Decision Process (POMDP) and solved it with Q-learning. Modelling is possible as end-user possesses partial information about the network state based on the throughput it receives [35]. The service layer control mechanism gracefully degrades video quality, depending on the connection status. Hence, the end-user is able to find the most appropriate quality level for his/her stream.

IV. DASH AND MULTIPLE COMPETING PLAYERS PROBLEM

The concept of adaptive video streaming is based on the idea to adapt the bandwidth required by the video stream to the throughput available on the network path from the stream source to the client [26]. These algorithms can live at the server [25], at an intermediate network device [24] or at the client [28]. With client-side protocols it is the player that decides what bitrate to request for any fragment, improving server-side scalability. A benefit of this protocol is that the player can control its playback buffer size by dynamically adjusting the rate at which new fragments are requested. The adaptation is performed by varying the quality of the streamed video. Multiple video segments constitute a video stream lasting from as little as 2 seconds to as much as having a 10 second chunk delivery rate. Segments are encoded and stored on the server in numerous quality versions, termed representations. Each version has a unique resolution, bitrate and/or quality. A client downloads segments using HTTP GET statements [52]. However, with adaptive streaming a client might request subsequent segments at different quality levels to manage varying network conditions, based on an estimation bandwidth. To do this it uses a manifest file that contains information about the video segments. Protocols and standards such as MPEG Dynamic Adaptive Streaming over HTTP (DASH), Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming (MSS) or Adobe HTTP Dynamic Streaming (HDS) typically use a media playlist that contains a list of uniform resource identifiers (URIs) that are addresses to media segments. The process of determining the ideal representation for each segment to enhance the user's experience is pivotal to adaptive streaming. The controller algorithm estimates the network bandwidth and chooses the next bitrate level corresponding to the available network bandwidth. Variations in the available bandwidth will result in jerky playback and disruption of the video playback if the throughput falls below the bit rate requirement of the video. This is the major challenge in adaptive video streaming. Selecting appropriate bitrate levels helps to maximize the user experience. Generally, higher bitrates and resolutions will give better user experience. For example, if a client approximates that there is 9.5Mb/s available in the network, it might request the server to stream video compressed to the highest video rate available, 9.5Mb/s, or the next rate below, 9.3Mb/s. If the client picks a video rate that is too high, the viewer will experience annoying re-buffering events; if they pick a streaming rate that is too low, the viewer will experience poor video quality.

Adaptive streaming uses the HTTP/TCP protocol stack to transmit video Web traffic. Thus, the development of this wave of HTTP-based streaming applications is not referred to as adaptive streaming over HTTP. The use of HTTP/TCP protocols for video streaming is because of the advantages that HTTP/TCP offers. It allows standard web servers and caches to be used increasing its' cost effectiveness. Another advantage is that all firewalls are configured to support HTTP connections [39]. In addition, is allows better scaling as HTTP is stateless and the streaming session is managed by the client, thus reducing the load on the server. However, HTTP/TCP use reveals further challenges as adaptation is on top of TCPs congestion control algorithm, which forms nested control loops. As the throughput of the TCP connection depends on both the link capacity and the amount of congestion, the throughput can vary significantly over time [28].

In the presence of competing HTTP-based adaptive streaming (HAS) clients the TCP throughput does not always faithfully represent the fair-share bandwidth [23]. 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[32]. 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 [33]. Depending on the temporal overlap of the ON-OFF [29] periods among competing players, they may not estimate their fair share correctly [9]. 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. 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,

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

bandwidth fluctuations, and the variability of the video encoding rate [1]. 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 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.

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

Unfairness: Let $JainFair_t$ be the Jain fairness index (cf. Equation 10) calculated on the average received rates [19], 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 more fair 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 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 throughput at time i and bw is the experimental available bandwidth).

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

V. SDP DASH-BASED APPROACHES

A. Method A [36]

HTTP Adaptive Streaming (HAS) is becoming a key technology for audiovisual broadcasting, although the varying conditions of the networks imply an uneven video quality. So, there are currently many client players endowed with proprietary adaptation algorithms aiming at maximizing the perceived quality. Looking for the best user's Quality of Experience (QoE), an adaptation algorithm based on the Q-Learning method is proposed by authors. This analysis identifies the variables that best capture the system dynamics and establishes a formulation for the characteristic functions of this Reinforcement Learning method. In addition, suitable parameter configurations for the algorithm are analyzed together with its convergence. Experimental results confirm the ability of the proposed solution to control efficiently the selection of the segment qualities, so that quality switches are minimized, and the occurrence of freezes is diminished. Moreover, a performance comparison with other strategies shows that this novel approach outperforms them. Finally, the adaptiveness of this strategy to changing environments is also assessed.

B. Method B [6]

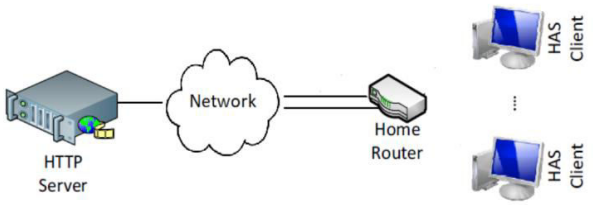
HTTP Adaptive Streaming (HAS) is becoming a key technology for audiovisual broadcasting, although the varying conditions of the networks imply an uneven video quality. So, there are currently many client players endowed with proprietary adaptation algorithms aiming at maximizing the perceived quality. Looking for the best user's Quality of Experience (QoE), an adaptation algorithm based on the Q-Learning method is proposed. This analysis identifies the variables that best capture the system dynamics and establishes a formulation for the characteristic functions of this Reinforcement Learning method. In addition, suitable parameter configurations for the algorithm are analyzed together with its convergence. Experimental results confirm the ability of the proposed solution to control efficiently the selection of the segment qualities, so that quality switches are minimized, and the occurrence of freezes is diminished. Moreover, a performance comparison with other strategies shows that this novel approach outperforms them. Finally, the adaptiveness of this strategy to changing environments is also assessed.

VI. EXPERIMENTAL SETUP

A virtual network is setup (cf. Figure 1) 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.

Figure 1: Streaming Setup

TAPAS [27] 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) [53] and Dynamic Adaptive Streaming over HTTP (DASH) format [5]. Algorithms that uses Method A and Method B was tested and shown to work on both MPEG-DASH [49], and

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

Apple HTTP Live Streaming (HLS) [37]. This makes it useful for video on demand (VOD) [13]and live streaming [51], 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, [11], [48], [8].

The ten-minute-long MPEG-DASH video sequence "Elephant's Dream" 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 [36]. 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.

VII. RESULTS

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

Table 1

	Method A	Method B	Conventional
Utilization	0.87	0.93	0.68
Unfairness	0.029	0.013	0.124
Instability	0.189	0.128	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 B outperforms Method A and the Conventional.

Table 2

	Method A	Method B	Conventional
Utilization	0.89	0.94	0.71
Unfairness	0.038	0.016	0.136
Instability	0.178	0.134	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 B outperforms Method A and the Conventional.

Table 3

	Method A	Method B	Conventional
Utilization	0.82	0.87	0.59
Unfairness	0.045	0.039	0.173
Instability	0.210	0.183	0.384

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

Table 4

	Method A	Method B	Conventional
Utilization	0.79	0.85	0.55
Unfairness	0.129	0.089	0.398
Instability	0.199	0.168	0.401

VIII. CONCLUSION

Q-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 Q-learning in DASH are selected and their performances compared against the Conventional DASH player. It is shown that Q-learning works well for various network conditions. However, one method outperforms the others in the experiments conducted.

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] Barron, Ann E. "An overview of digital video." Journal of Computing in Higher Education 7, no. 1 (1995): 69-84.

[3] Capitan, Jesus, Matthijs TJ Spaan, Luis Merino, and Anibal Ollero. "Decentralized multi-robot cooperation with auctioned POMDPs." The International Journal of Robotics Research 32, no. 6 (2013): 650-671.

[4] Chadha, Aaron, Eirina Bourtsoulatze, Vasileios Giotsas, Yiannis Andreopoulos, and Sergio Grce. "Deep-learning based precoding techniques for next-generation video compression." (2019).

[5] Chen, Lulin, and Shan Liu. "Method and system of adaptive application layer fec for mpeg media transport." U.S. Patent Application 16/074,014, filed February 7, 2019.

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

[6] Claeys, Maxim, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, and Filip De Turck. "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming." In Proceedings of the 2013 Workshop on Adaptive and Learning Agents (ALA), Saint Paul (Minn.), USA, pp. 30-37. 2013.

[7] Claeys, Maxim, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, and Filip De Turck. "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming." In Proceedings of the 2013 Workshop on Adaptive and Learning Agents (ALA), Saint Paul (Minn.), USA, pp. 30-37. 2013.

[8] De Cicco, Luca, Giuseppe Cilli, and Saverio Mascolo. "ERUDITE: a deep neural network for optimal tuning of adaptive video streaming controllers." In Proceedings of the 10th ACM Multimedia Systems Conference, pp. 13-24. ACM, 2019.

[9] 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.

[10] Depover, Christian, and François Orivel. Developing countries in the e-learning era. Unesco, 2013.

[11] Dubin, Ran, Raffael Shalala, Amit Dvir, Ofir Pele, and Ofer Hadar. "A fair server adaptation algorithm for HTTP adaptive streaming using video complexity." Multimedia Tools and Applications 78, no. 9 (2019): 11203-11222.

[12] Fei, Yu, Vincent WS Wong, and Victor Leung. "Efficient QoS provisioning for adaptive multimedia in mobile communication networks by reinforcement learning." Mobile Networks and Applications 11, no. 1 (2006): 101-110.

[13] Fickle, Richard C., José A. Róyo, and Timothy Bruce Aron. "Video-on-demand (VOD) management system and methods." U.S. Patent 9,027,063, issued May 5, 2015.

[14] Fielding, Roy, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. "Hypertext transfer protocol–HTTP/1.1." (1999).

[15] FONNESBECK, CHRISTOPHER J. "Solving dynamic wildlife resource optimization problems using reinforcement learning." Natural Resource Modeling 18, no. 1 (2005): 1-40.

[16] Gadaleta, Matteo, Federico Chiariotti, Michele Rossi, and Andrea Zanella. "D-DASH: A deep Q-learning framework for DASH video streaming." IEEE Transactions on Cognitive Communications and Networking 3, no. 4 (2017): 703-718.

[17] Golubchik, Leana, John CS Lui, and Richard R. Muntz. "Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers." Multimedia Systems 4, no. 3 (1996): 140-155.

[18] Hu, Junling, and Michael P. Wellman. "Nash Q-learning for general-sum stochastic games." Journal of machine learning research 4, no. Nov (2003): 1039-1069.

[19] 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.

[20] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." Journal of artificial intelligence research 4 (1996): 237-285.

[21] Katsavounidis, Ioannis. "Encoding techniques for optimizing distortion and bitrate." U.S. Patent Application 16/034,303, filed January 24, 2019.

[22] Keerthi, S. Sathiya, and B. Ravindran. "A tutorial survey of reinforcement learning." Sadhana 19, no. 6 (1994): 851-889.

[23] 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.

[24] 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.

[25] 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.

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

[27] Khan, Koffka, and Wayne Goodridge. "Variants of the Constrained Bottleneck LAN Edge Link in Household Networks." International Journal of Advanced Networking and Applications 10, no. 5 (2019): 4035-4044.

[28] Khan, Koffka, and Wayne Goodridge. "What happens when adaptive video streaming players compete in time-varying bandwidth conditions?." International Journal of Advanced Networking and Applications 10, no. 1 (2018): 3704-3712.

[29] Khan, Koffka, and Wayne Goodridge. "What happens when adaptive video streaming players compete with Long-Lived TCP flows?." International Journal of Advanced Networking and Applications 10, no. 3 (2018): 3898-3904.

[30] Kua, Jonathan, Grenville Armitage, and Philip Branch. "A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP." IEEE Communications Surveys & Tutorials 19, no. 3 (2017): 1842-1866.

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

[31] Kuipers, Fernando, Robert Kooij, Danny De Vleeschauwer, and Kjell Brunnström. "Techniques for measuring quality of experience." In International Conference on Wired/Wireless Internet Communications, pp. 216-227. Springer, Berlin, Heidelberg, 2010.

[32] Lederer, Stefan, Christopher Müller, and Christian Timmerer. "Dynamic adaptive streaming over HTTP dataset." In Proceedings of the 3rd Multimedia Systems Conference, pp. 89-94. ACM, 2012.

[33] 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.

[34] Lin, Long-Ji. "Self-improving reactive agents based on reinforcement learning, planning and teaching." Machine learning 8, no. 3-4 (1992): 293-321.

[35] Marinca, Dana, Dominique Barth, and Danny De Vleeschauwer. "A Q-Learning solution for adaptive video streaming." In 2013 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT), pp. 120-126. IEEE, 2013.

[36] Martín, Virginia, Julián Cabrera, and Narciso García. "Design, optimization and evaluation of a Q-Learning HTTP Adaptive Streaming Client." IEEE Transactions on Consumer Electronics 62, no. 4 (2016): 380-388.

[37] Müller, Christopher, Stefan Lederer, and Christian Timmerer. "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments." In Proceedings of the 4th Workshop on Mobile Video, pp. 37-42. ACM, 2012.

[38] Nam, Hyunwoo, Kyung-Hwa Kim, Jong Yul Kim, and Henning Schulzrinne. "Towards QoE-aware video streaming using SDN." In 2014 IEEE Global Communications Conference, pp. 1317-1322. IEEE, 2014.

[39] Nelson, Dean S. "Reverse HTTP connections for device management outside a firewall." U.S. Patent 6,553,422, issued April 22, 2003.

[40] Ouferhat, Nesrine, and Abdelhamid Mellouk. "A QoS scheduler packets for wireless sensor networks." In 2007 IEEE/ACS International Conference on Computer Systems and Applications, pp. 211-216. IEEE, 2007.

[41] Peng, Jing, and Ronald J. Williams. "Incremental multi-step Q-learning." In Machine Learning Proceedings 1994, pp. 226-232. Morgan Kaufmann, 1994.

[42] Poularakis, Konstantinos, George Iosifidis, Antonios Argyriou, Iordanis Koutsopoulos, and Leandros Tassiulas. "Distributed Caching Algorithms in the Realm of Layered Video Streaming." IEEE Transactions on Mobile Computing 18, no. 4 (2018): 757-770.

[43] Powell, Warren B. "Perspectives of approximate dynamic programming." Annals of Operations Research 241, no. 1-2 (2016): 319-356.

[44] Puterman, Martin L. "Markov decision processes." Handbooks in operations research and management science 2 (1990): 331-434.

[45] Qian, Feng, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. "Optimizing 360 video delivery over cellular networks." In Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, pp. 1-6. ACM, 2016.

[46] Ramaswamy, Kumar, and Jeffrey Allen Cooper. "Bit depth remapping based on viewing parameters." U.S. Patent Application 16/099,402, filed July 25, 2019.

[47] Rummery, Gavin A., and Mahesan Niranjan. On-line Q-learning using connectionist systems. Vol. 37. Cambridge, England: University of Cambridge, Department of Engineering, 1994.

[48] Shi, Wanxin, Qing Li, Chao Wang, Gengbiao Shen, Weichao Li, Yu Wu, and Yong Jiang. "LEAP: learning-based smart edge with caching and prefetching for adaptive video streaming." In Proceedings of the International Symposium on Quality of Service, p. 16. ACM, 2019.

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

[50] Sorokopud, Gennady, Aharon Satt, and Liron Langer. "Real Time Streaming Protocol (RTSP) proxy system and method for its use." U.S. Patent Application 10/852,995, filed February 9, 2006.

[51] Sripanidkulchai, Kunwadee, Bruce Maggs, and Hui Zhang. "An analysis of live streaming workloads on the internet." In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp. 41-54. ACM, 2004.

[52] 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.

[53] Sugunan, Remesh Kousalya, William P. Franks, Nithin Raj Kuyyar Ravindranath, Vinod Jatti, and Praveen Girish. "Method and apparatus to efficiently smooth adaptive content playback in http live streaming." U.S. Patent Application 15/727,062, filed April 11, 2019.

[54] Tang, Kexin, Nuowen Kan, Junni Zou, Xiao Fu, Mingyi Hong, and Hongkai Xiong. "Multiuser Video Streaming Rate Adaptation: A Physical Layer Resource-Aware Deep Reinforcement Learning Approach." arXiv preprint arXiv:1902.00637 (2019).

[55] Wang, Bing, Jim Kurose, Prashant Shenoy, and Don Towsley. "Multimedia streaming via TCP: An analytic performance study." In Proceedings of the 12th annual ACM international conference on Multimedia, pp. 908-915. ACM, 2004.

Volume: 11 Issue: 02 Pages: 4190-4197 (2019) ISSN: 0975-0290

[56] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8, no. 3-4 (1992): 279-292.

[57] Wu, Dapeng, Yiwei Thomas Hou, Wenwu Zhu, Ya-Qin Zhang, and Jon M. Peha. "Streaming video over the Internet: approaches and directions." IEEE Transactions on circuits and systems for video technology 11, no. 3 (2001): 282-300.

[58] Yee, Richard. Abstraction in control learning. University of Massachusetts at Amherst, Department of Computer and Information Science, 1992.

[59] Zinner, Thomas, Oliver Hohlfeld, Osama Abboud, and Tobias Hoßfeld. "Impact of frame rate and resolution on objective QoE metrics." In 2010 second international workshop on quality of multimedia experience (QoMEX), pp. 29-34. IEEE, 2010.

AUTHOR DETAILS:

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.