A Quality Selection Mechanism Using a Deep Q-Network for Seamless Video Streaming Services

*Iseul Kim, **Seongjun Hong, *Sungwook Jung, **Kyungshik Lim
*School of Computer Science and Engineering, Kyungpook National University
**School of Computer Science and Engineering and Software Technology Research Center
Kyungpook National University
dltmf0689@gmail.com, seong889@gmail.com, squirrelswj@gmail.com, kslim@knu.ac.kr

***Abstract*—There has been an increasing demand on providing seamless video streaming services. One of the major efforts to meet it is the Dynamic Adaptive Streaming over HTTP(DASH) standard. However, its performance heavily depends on the client's adaptive quality selection algorithm that is not included in the standard. The existing conventional algorithms are basically based on a procedural algorithm that is not easy to capture and reflect all variations of dynamic network and traffic conditions in real network environments. To overcome this limitation, this paper proposes a novel quality selection mechanism based on the Deep Q-Network(DQN) model, the DQN-based DASH ABR(DQN_{ABR}) mechanism. The proposed mechanism adopts a new reward calculation method based on five major performance metrics to reflect the current conditions of networks and devices in real time. The DQN_{ABR} system is implemented as a separate server that is connected via a high-speed LAN to the simulation host where the ns-3 DASH simulation software is performed. Experimental results show that the proposed mechanism quickly selects a suitable quality even in high error rate environments, significantly reducing the frequency of quality change compared to the existing algorithm and improving the average quality during video playback.**

***Index Terms*—Adaptive video streaming, DASH, Deep Q-Network, Quality of Experience(QoE)**

I. INTRODUCTION

WITH rapid development of wireless networks and wide spread of high-performance wireless devices such as smartphones and tablets, a huge demand for video streaming services in the Long Term Evolution(LTE) environment has been ever increasing. From the viewpoint of content providers and telecommunication companies, the enhancement of the Quality of Experience(QoE) has become a major issue as it greatly impacts their revenues. The QoE is, however, heavily influenced by the performance of the client's adaptive quality selection algorithm in the dynamic adaptive streaming over HTTP(DASH) standard[1].

There has been many studies on enhancing quality of video streaming services in dynamic network environments[12,14]. In most of studies, however, clients the select quality of requesting video segments by a procedural algorithm which tries to analyze severe fluctuation of network dynamics in real time[2-4]. With this approach, there might be some limitations on enhancing user-experienced quality of video streaming services. One of the most important reasons is that it is not easy for a conventional procedural algorithm to capture and reflect all variations of dynamic network and traffic conditions in real network environments.

To overcome this limitation, this paper proposes a novel quality selection mechanism based on the Deep Q-Network(DQN) model, the DQN-based DASH ABR(DQN_{ABR}) mechanism. The goal of the mechanism is to select requesting video segments with higher quality if seamless video streaming services are not violated. This implies that under the same bad network conditions the existing conventional approach merely tends to request video segments with better quality even in case of frequent interruption of video playback. In particular, the prediction model of the DQN_{ABR} is based on an integrated equation of the QoE evaluation that incorporates throughput, buffer occupancies, and quality information simultaneously.

The remainder of the paper is organized as follows: Section 2 introduces related works. In Section 3, we describe the proposed model in detail. Section 4 shows an experimental analysis of the proposed mechanism and we conclude in Section 5.

II. RELATED WORKS

A. Dynamic Adaptive Streaming over HTTP(DASH)

To provide video streaming services on the Internet, DASH changes video quality dynamically during video playback using an algorithm that analyzes network conditions and device performance in real time. The DASH server maintains different encodings of the same video content based on a combination of different criteria such as resolution, bitrate, number of frames per second, and compression rate. The encoded video contents are segmented based on the playback time, and information

* Corresponding author: Kyungshik Lim (kslim@knu.ac.kr)

about the segmented video contents is managed in a Media Presentation Description(MPD). After receiving the MPD from the server in the very first time of communication the DASH client requests an appropriate quality of a segment to the server. The client then receives and plays the requested segment, and continues to manage network metrics, analyze them using the DASH Adaptive Bitrate(ABR) algorithm, and select a quality of the next requesting segment[5].

The DASH ABR($DASH_{ABR}$) algorithm operates using four representative rules: throughput rule, insufficient buffer rule, switch history rule, and dropped frame rule. The rules inherently handle some metrics from network conditions and device capabilities such as throughput, latency, buffer length, number of frames dropped, and number of quality changes[6]. Since with those metrics the $DASH_{ABR}$ follows a procedural algorithm to determine a quality of the requesting segment, however, there might be some limitations to consider various combinations of those metrics in dynamic network environments. It should be noted that the DASH standard does not include a quality selection algorithm and a media player whose empirical operations are open[7].

B. Deep Q-Network(DQN)

Reinforcement learning is one of the machine learning methods, which combines the advantages of supervised learning and unsupervised learning. The agent performs learning, makes decisions in the current state of the environment, and executes an action according to the decision. When an action is performed, the current state of the environment changes to another state, which is evaluated and rewarded depending on the action. Another action is then performed again in the changed state. Therefore, reinforcement learning learns how to maximize the reward value by making an optimal decision process for the action to be performed through selecting a series of actions and compensating for them [8].

There is a Q-learning technique[9] which is a simple and well-known method for single-agent learning in reinforcement learning. Q-learning is a way of expressing values for all states and actions in a given environment and updates the table according to the reward received from the action. However, there is a limitation in applying the Q table to the complex environment because the calculation speed is delayed, and the calculation result becomes inaccurate when the Q table is used to solve a large problem in a large number of environments.

To solve this problem, a DQN approach[10] has been proposed that replaces the Q table with a deep neural network. This approach can be combined with a variety of deep learning models to achieve better performance in complex environments. In this paper the proposed mechanism applies the DQN approach to the quality selection algorithm, making it possible to calculate the reward using both the network state and the quality selection information over time and to define a model for determining a quality that reflects the reward. This makes it unnecessary to create a user-defined algorithm covering all variations of network conditions unlike the conventional algorithm and to generate the dataset explicitly that is needed by a supervised learning like the Deep Neural Network learning. Furthermore, a simple adoption of a new reward calculation method can give an easy way to achieve different goals of other video streaming services.

C. Quality of Experience(QoE)

In video streaming services, the major metrics associated with a user's QoE include buffer length, number of quality changes, number of rebuffering events, duration of rebuffering, and the average video quality[3,11,13].

The buffer length indicates that the video can be played back from the current playback time. If the buffer is empty, the video playback stops until sufficient data is received for playback. Therefore, maintaining the adequate buffer length according to the fluctuation of network conditions can improve the QoE[3]. The frequent conversion of video quality might degrade the user's QoE. Therefore, reducing unnecessary quality changes could improve the QoE[3]. The number of rebuffering is the number of events that the buffer is empty and video playback is stopped and a major cause of performance degradation[11]. The duration of rebuffering is the duration between when the buffer is empty and video playback is resumed, and the longer the duration of rebuffering, the lower the QoE[13]. Average video quality implies the overall performance of video streaming services, and the higher the average quality, the better the user's QoE[17]. In this paper, therefore, the proposed mechanism adopts all the above five metrics so that a new reward calculation method are introduced to reflect the current conditions of networks and devices in real time.

III. DQN BASED INTELLIGENT $DASH_{ABR}$ (DQN_{ABR}) SYSTEM

A. The DQN_{ABR} mechanism

Fig. 1 shows the overall architecture of the DQN_{ABR} system. This system consists of four major modules: network environment module, agent module, reward calculation module, and the replay memory module.

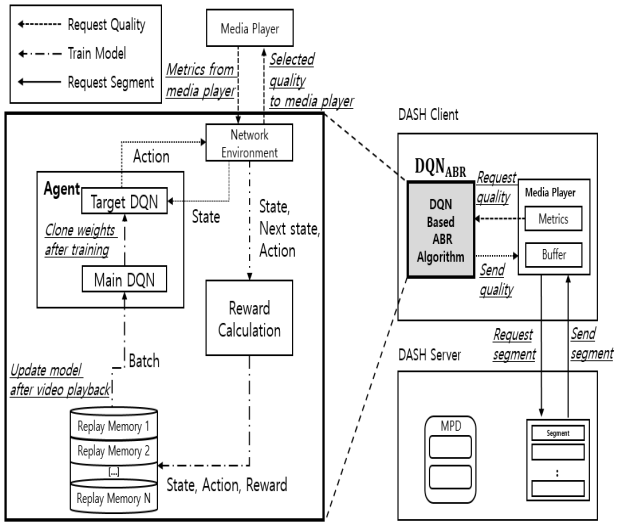

Fig. 1. The overall architecture of the DQN_{ABR} system

The agent module has a Double DQN[18] structure with the target DQN and the main DQN. The network environment module receives network state information in real time from the media player and sends it to the agent module. The target DQN of the agent module determines a quality, represented as an action in Fig. 1, that can receive the highest reward in the current network state and returns it back to the network environment module. The reward calculation module calculates the reward on receiving the selected quality and both the current and next network state information. The reward is then stored to the replay memory module together with network state information and the selected quality as a 3-tuple. After video playback is finished, a predetermined number of randomly chosen 3-tuples are repeatedly fed into the main DQN until a repetition count is reached. Once a learning is finished, the learned main DQN is then copied to the target DQN and the learned model is used to determine the next quality in the next round.

B. *Learning process of the DQN_{ABR} mechanism*

1) *Definition of learning metrics*

The learning metrics used in this experiment are defined as throughput, latency, buffer length, quality selected from previous state, and buffer target.

2) *Reward calculation*

The reward indicates a degree of the QoE improvement for a selected quality of segment. It is obtained from a reward calculation formula that uses the above metrics. The reward calculation formula can be set differently according to the goal to be learned with different learning results. It is therefore, important to find an appropriate reward calculation formula according to the goal. In this experiment, two reward formulas are defined and their performance is compared.

The reward is calculated for every video segment using the current quality, difference between the previous and current qualities, the current buffer length, difference between the previous and current buffer length, and together with the above buffer length and buffer target metrics. Each is calculated as shown in equation (1) to equation (5) below.

$$R_{quality} = a_k \tag{1}$$

$$R_{qswitch} = a_k - a_{k-1} \tag{2}$$

$$R_{buffer} = \begin{cases} b_{a_k} & if(b_{a_k} > 0), \\ -100 & otherwise \end{cases} \tag{3}$$

$$R_{bswitch} = b_{a_k} - b_{a_{k-1}} \tag{4}$$

$$R_{btarget} = \begin{cases} bt_{a_k} & if(bt_{a_k} = 60)\ and\ (R_{bswitch} > 0), \\ 0 & otherwise \end{cases} \tag{5}$$

$$R1 = (R_{quality} * 0.3) + (R_{qswitch} * 0.3) + (R_{buffer} * 0.1) + (R_{bswitch} * 0.1) + (R_{btarget} * 0.2) \tag{6}$$

$$\mathrm{R2} = (R_{quality} * 0.1) + (R_{qswitch} * 0.1) + (R_{buffer} * 0.3) + (R_{bswitch} * 0.3) + (R_{btarget} * 0.2) \tag{7}$$

$R_{quality}$ is the quality selected in step *k*, and the higher the quality, the higher the reward. $R_{qswitch}$ is the difference between the qualities selected in step *k* and step *(k-1)* where the change from low quality to high quality returns good reward. R_{buffer} is a reward that is determined by b_{a_k}, the buffer length after downloading a segment with a_k quality. If the buffer length becomes 0 after downloading the requesting segment, R_{buffer} is set to -100 which means bad reward to avoid rebuffering. Otherwise, the higher the buffer length, the higher the reward. $R_{bswitch}$ is the value of b_{a_k} minus $b_{a_{k-1}}$, and the larger the value, the better the reward. $R_{btarget}$ is a reward calculated via bt_{a_k} so that the playback time for high quality of segments is maintained as long as possible while that of low quality is maintained as short as possible. The reason that bt_{a_k} is set to 60 is as follows. In the existing DASH algorithm, if the maximum quality is selected, there is no further change to higher quality, and the buffer target is set to 60 and holds the video time to play at the highest quality. If you select a quality that is not the highest quality, it is set to 20 to limit the duration of low quality[6]. In this experiment, we defined the buffer target as the same value as the existing DASH. Therefore, it is set to receive a good reward when bt_{a_k} is 60 and $R_{bswitch}$ is greater than or equal to zero.

The two reward formulas used in the learning were defined as R1 in equation (6) and R2 in equation (7). The R1 and R2 functions calculate how well a_k, the quality chosen in step *k*, improves on the five goals associated with QoE. Each term of those equations has a weight that can be adjusted according to the goal to be learned. R1 gives a high weight on the quality term so that average video quality is mainly improved. R2 has a high weight on the buffer length term, so that the number of rebuffering is reduced by keeping the average buffer length high.

3) *Experience Replay*

When the Q function is approximated through the DQN learning, a problem of falling into the local minimum may occur. A way to overcome this problem is the experience replay technique[15]. First, all the metrics (state, action, reward, next action) to be used in the learning process are stored in the replay memory. Instead of using the latest transitions one by one in the learning phase, it is a way to learn several randomly selected replay memories. In this experiment, all the metrics are stored in the replay memory during video playback, and the procedure of learning 100 times at the end of a video playback, where each selects 60 metrics randomly, is repeated 1,300 times.

4) *Epsilon-greedy exploration*

In the Q-learning algorithm, the initial matrix or the network parameters are all randomly initialized. Therefore, if one selects an action with the highest Q value, the action will be performed with the first strategy it finds. Epsilon-greedy exploration[8] is a technique to solve this problem. Basically, the action with the highest Q value is selected and taken a random action with a certain epsilon (ε). In the early stage of learning, therefore, there is a high probability of randomly choosing an action. When the learning progresses and the Q function converges the random probability value is gradually decreased from 1.0 to 0.1. In this experiment, a random epsilon equation, ε = 1/max(step count/100,1), was used. For example, the random probability is 1 in case of less than 100 learnings and 0.2 in case of greater than 500 learnings.

5) Deep Neural Network(DNN) design

The inputs of the designed neural network are the current throughput, delay time, buffer length, previous quality, buffer target and the output is the quality. Three hidden layers were added between the input and output layers, where the number of nodes in each layer is 64, 32, and 16, respectively.

IV. Experiment and Evaluation

In order to evaluate the performance of the proposed mechanism, simulation was performed using Network Simulator-3(NS-3). The bandwidth between the DASH server and the DASH client is set to 40Mbps, considering the conventional LTE environment. The delay time and the loss rate are set to various values to represent dynamic network conditions. Experimental results were evaluated by comparing the average quality, number of quality changes, number of rebuffering, duration of rebuffering, and the buffer length of the DQN_{ABR} with the $DASH_{ABR}$.

A. Configuring the simulation environment

The DASH client in the proposed mechanism consists of the conventional media player and the DQN_{ABR} system that is implemented as a separate server connected via a high-speed LAN. Both systems are connected based on socket communication and perform DASH based streaming service. The bandwidth of communication between the DASH server and the DASH client is fixed at 40Mbps. The delay times were set to 10ms, 30ms, 50ms, and 100ms, and the loss rates were set to 10^{-3}, 10^{-2}, and 10^{-1}, respectively. In this way, learning was performed a combination of the 12 settings. The test environment was the same as the learning environment. The video used for the learning and testing was a Big Buck Bunny animation that has a length of 640 seconds in time. The length of one segment was 4 seconds, and the quality level of each segment was represented by an index of 10 levels from 0 to 9. Table 1 shows the resolution and representation rates of each quality level.

TABLE I
Representation rate for 4-sec dataset

Quality level	Resolution	Representation rates
0	320 x 180	200 kbps
1	320 x 180	400 kbps
2	480 x 270	600 kbps
3	640 x 360	800 kbps
4	640 x 360	1000 kbps
5	789 x 432	1500 kbps
6	1024 x 576	2500 kbps
7	1280 x 720	4000 kbps
8	1920 x 1080	8000 kbps
9	3840 x 2160	12000 kbps

B. Performance measurement results

From Fig. 2 to Fig. 5, the average values of video quality, buffer target, buffer length, and rebuffering count are shown when the same video with a length of 640 seconds in time is repeatedly played three times in each network state, respectively. Fig. 6 to Fig. 11 represents the same performance metrics as the above figures as time continues to vary.

In the legend of the graph, DASH indicates the result of the existing DASH algorithm, DQN_reward1 is the result of quality selection using DQN_{ABR} using the reward calculation formula R1 of equation (6), and DQN_reward2 is the result of quality selection using the reward calculation formula R2 of equation (7).

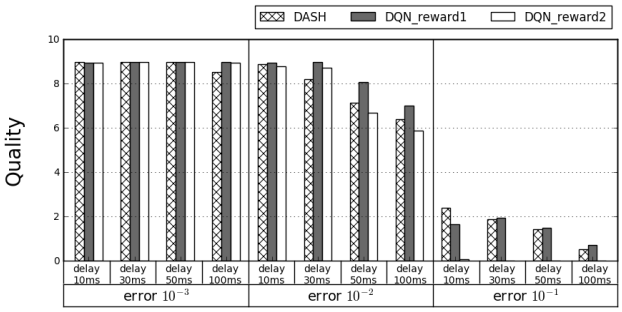

Fig. 2. Average video quality

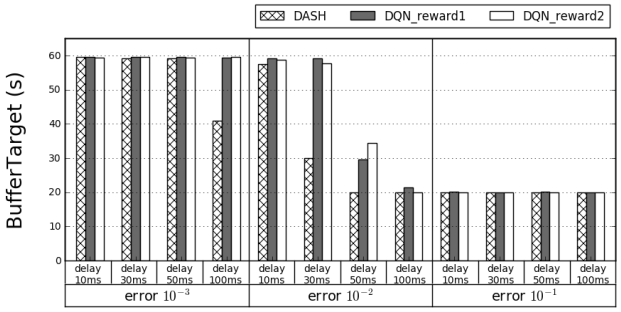

Fig. 3. Average buffer target

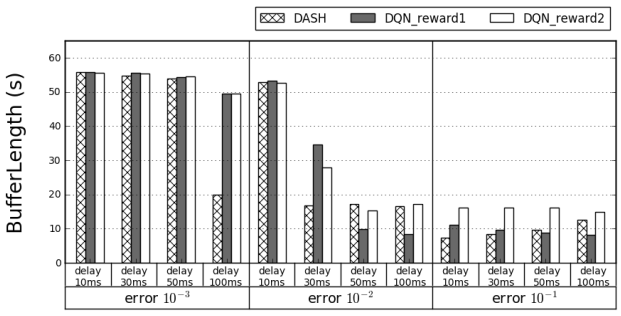

Fig. 4. Average buffer length

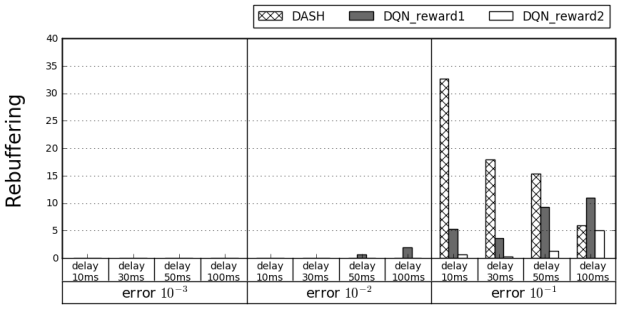

Fig. 5. Average rebuffering count

The average quality of the video shown in Fig. 2 shows that the overall quality of DQN_reward1 is higher than that of DQN_reward2 and DASH. In Fig. 4, it can be seen that DQN_reward2 mostly maintains the average buffer length at a higher level. This result shows that the learning was performed as intended by giving more weight on the interesting term of the reward calculation formula.

In Fig. 2, the highest quality was selected for both DASH and DQN at the lowest error rate of 10^{-3}. Also, Fig. 5 shows that there is no rebuffering even after downloading the highest quality. Therefore, in a network environment with an error rate of 10^{-3}, choosing the highest quality quickly and reducing the number of quality changes in a way to improve QoE. However, Fig. 6 shows that when the delay time is 100ms, the proposed model quickly selects an appropriate quality and the frequency of the quality change becomes low, but the existing algorithm is not easy to find an appropriate quality and results in frequent changes. Therefore, the existing algorithm is not stable due to the longer delay time in a low error rate environment, but the proposed model is more stable. Fig. 7 and Fig. 8 show that overall buffer length and buffer target are kept higher over time, respectively, resulting in much better performance compared to the existing algorithm. In more detail, the results show a similar performance when delay time is short and error rate is low. Although delay time is long, the results confirm that an appropriate quality is determined directly with less number of intermediate quality changes and thus the buffer length is kept long.

In Fig. 5, it is shown that the average number of rebuffering in an environment with a very high error rate of 10^{-1} is much less than that of the existing algorithm. Since the video playback usually stops due to rebuffering, it is obvious that the rebuffering frequency will have a very bad effect on QoE. This means that the existing DASH algorithm does not reflect the network environment in real time and thus choose an unsuitable quality. Fig. 9, Fig. 10, and Fig. 11 show a situation with an error rate of 10^{-1} and a delay time of 10ms over time. DQN_reward1 consistently selected a high quality, resulting in rebuffering. However, DQN_reward2 which selects a quality with the consideration of buffer length keeps a stable buffer length and reduces the number of rebuffering occurrences. In Fig. 9, Fig. 10, and Fig. 11 the vertical lines drawn at 640 seconds in the x axis indicates the length of the video used in the test. In these graphs, DASH requested a download of the last segment in 700 seconds, DQN_reward1 in 630 seconds, and DQN_reward2 in 640 seconds. This indicates that about 60 seconds of rebuffering occurred while playing the video with the existing algorithm. This result shows that the proposed model reduces the number of rebuffering and the duration of rebuffering by choosing the appropriate quality so that the buffer is not empty, reflecting the current network condition in real time.

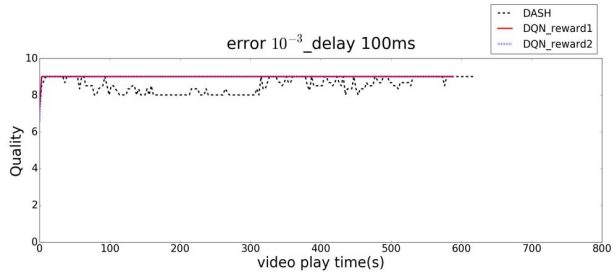

Fig. 6. Average video quality over time

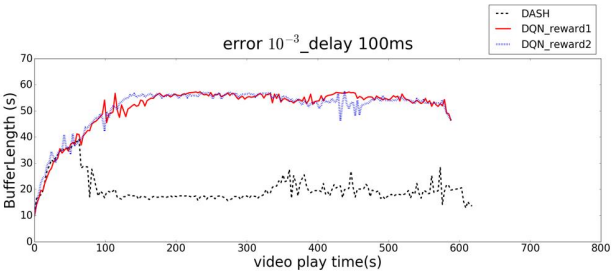

Fig. 7. Average buffer length over time

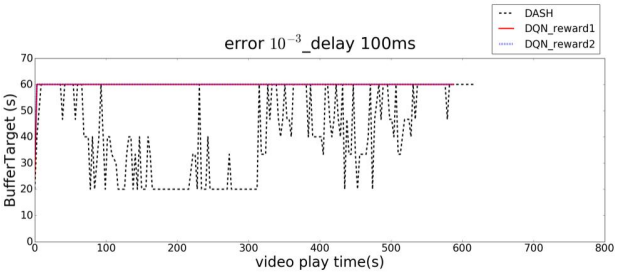

Fig. 8. Average buffer target over time

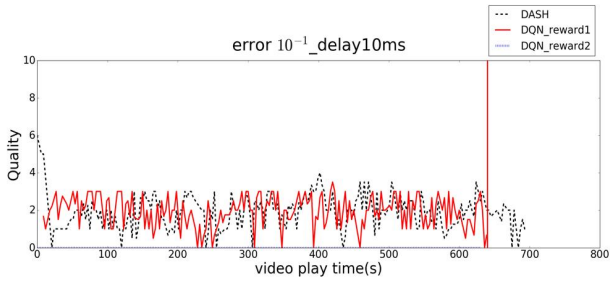

Fig. 9. Average video quality over time

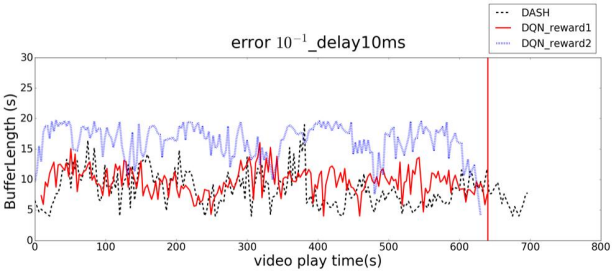

Fig. 10. Average buffer length over time

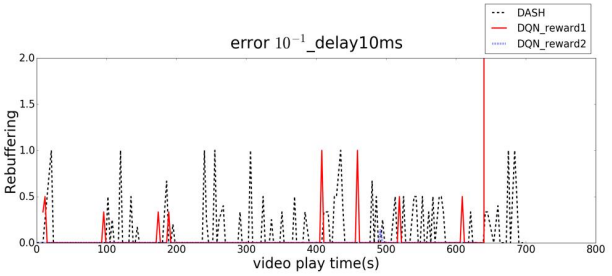

Fig. 11. Average rebuffering count over time

Through this experiment, the existing algorithm caused frequent quality changes in low error rate environments and lowered the average quality of the video. In the meanwhile, the proposed model quickly selected the appropriate quality, reduced the number of quality changes, increased the average quality, and kept the buffer length higher. Also, in an environment with a very high error rate, the existing algorithm frequently selected a quality not suitable for the network conditions and the buffer became frequently empty. However, the proposed model maintains the buffer length properly by selecting the appropriate quality according to the current network state and reduces both the number and the duration of rebuffering. Therefore, according to the QoE criterion defined above, the proposed model shows higher performance than the existing algorithm, and it is confirmed that it can provide seamless video streaming service.

V. CONCLUSION

In this paper, we propose a quality selection mechanism based on a Deep Q-Network for seamless video streaming services in the LTE network environment. The mechanism introduces an integrated equation of the QoE evaluation that incorporates throughput, buffer occupancies and quality information simultaneously.

We defined the criteria for evaluating QoE for performance evaluation. We simulated the LTE network and experimented the proposed mechanism to compare with the existing algorithm. Experimental results show that the proposed mechanism quickly selects a suitable quality in low error rate environments, apparently reducing the frequency of quality change compared to the existing algorithm and improving the average quality during video playback.

In the environment with high error rate, the proposed model chooses a quality suitable for the current network conditions so that it can avoid a situation where the buffer becomes frequently empty and reduce the number of rebuffering events compared with the existing algorithm. In particular, DQN_reward2, which gives more weight on the buffer length, reduces the number of rebuffering events by 90% compared to existing algorithm and selects the best video quality without interruption in various network environments. These results show that the proposed mechanism increases the frequency of selecting the appropriate quality suitable for the network conditions. It is also shown that the use of the DQN_{ABR} mechanism as a quality selection technique can be a way to improve the QoE of adaptive streaming services significantly in a dynamic network environment. Introducing other learning metrics, reward calculation methods, variation of weights on each metrics can give other ways to achieve seamless video streaming services, and remains for further study.

VI. ACKNOWLEDGMENT

This research was partly supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the National program for Excellence in Software supervised by the IITP(Institute for Information & communications Technology Promotion). (2015-0-00912) and by the BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (21A20131600005).

REFERENCES

[1] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP,” ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, pp. 325-338, 2015.

[2] T. Stockhammer, “Dynamic Adaptive Streaming over HTTP – Standards and Design Principles,” ACM conference on Multimedia systems, MMSys 2011, pp. 133-144, 2011.

[3] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, “A Survey on Quality of Experience of HTTP Adaptive Streaming,” IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 469-492, 2015.

[4] D. Zegarra Rodríguez, R. Lopes Rosa, E. Costa Alfaia, J. Issy Abrahão, and G. Bressan, “Video Quality Metric for Streaming Service Using DASH Standard,” IEEE Transactions on Broadcasting, vol. 62, no. 3, pp. 628-639, 2016.

[5] J. Kua, G. Armitage, and P. Branch, “A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1842-1866, 2017.

[6] Dash Industry Forum Reference Player Documentation(2017). https://github.com/Dash-Industry-Forum/dash.js/wiki (accessed Jan., 11, 2018).

[7] M. Claeys, S. Latre, J. Famaey, and F. De Turck, “Design and Evaluation of a Self-Learning HTTP Adaptive Video Streaming Client,” IEEE Communications Letters, vol. 18, no. 4, pp. 716-719, 2014.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore “Reinforcement Learning: A Survey,” Journal of Artificial Intelligence Research, vol. 4, pp. 237-285, 1996.

[9] C. J. C. H. Watkins, and P. Dayan, “Q-learning,” Machine Learning, vol. 8, no. 3-4, pp.279–292, 1992.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., “Human-level control through deep reinforcement learning,” Nature 518, vol. 7540, pp. 529-533, 2015.

[11] A. C. Dalal, D. R. Musicant, J. Olson, B. McMenamy, S. Benzaid, B. Kazez et al., “Predicting User-Perceived Quality Ratings from Streaming Media Data,” IEEE International Conference on Communications, pp. 65-72, 2007.

[12] L. Amour, M. S. Mushtaq, S. Shouihi, and A. Mellouk, "QoE-based framework to optimize user perceived video quality," IEEE LCN 2017, pp. 599-602, 2017.

[13] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, “Initial delay vs. interruptions: Between the devil and the deep blue sea,” 2012 Fourth International Workshop on Quality of Multimedia Experience, pp. 1-6, 2012.

[14] K. Jia, Y. Guo, Y. Chen, and Y. Zhao, "Measuring and predicting quality of experience of DASH-based video streaming over LTE," WPMC 2016, pp. 102-107, 2016.

[15] S. Adam, L. Busoniu, and R. Babuska, “Experience Replay for Real-Time Reinforcement Learning Control,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 2, pp. 201-212, 2012.

[16] T. R. Henderson, M. Lacage, and G. F. Riley, “Network Simulations with the ns-3 Simulator,” ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2008, pp. 17-22, 2008.

[17] P. Juluri, V. Tamarapalli, and D. Medhi, “QoE management in DASH systems using the segment aware rate adaptation algorithm,” IEEE/IFIP Network Operations and Management Symposium, NOMS 2016, pp. 129-136, 2016.

[18] H. V. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-Learning,” AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 2094-2100, 2016.