Markov Decision Processes for Bitrate Harmony in Adaptive Video Streaming

Koffka Khan, Wayne Goodridge
The University of the West Indies
Email: koffka.khan@gmail.com, wayne.goodridge@sta.uwi.edu

***Abstract*—It has become common for many devices to share bottleneck links when users watch streaming video. When the DASH standard is used for adaptive video streaming over HTTP it has been found that good Quality of Experience (QoE) among video players become a critical issue. Markov Decision Processes (MDP) is one attempt at optimizing the streaming process by adopting a policy for maximizing particular QoE parameters. This paper proposes a novel approach called SHARE that uses a state-array representation consisting of a quality measurement called Data Rate Ratio (DRR) from each player in the network. A third-party network device collects the DRR values of players. Further it uses a MDP based on discretized DRRs to generate policies for better bitrate selection at runtime, using a unique reward function. A three player model is presented. Based on comparisons with other methods, the result shows that players adopting these policies obtains good QoE across various metrics, such as, bandwidth utilization, unfairness, re-buffering ratios, instability and average quality, with minimal possible trade-offs.**

Keywords*—*Bottleneck links; DASH; Markov Decision Process; adaptive streaming; Quality of Experience (QoE)

I. Introduction

Recently there is a huge increase in Internet video traffic [1]. This results in a greater possibility of more than one video player sharing a bottleneck link and competing for bandwidth. There are many heuristic approaches that attempt to address multiplayer user QoE. They aim to select player bitrates either based on available bandwidth, bandwidth-based approaches [2], or based on buffer occupancy, buffer-based approaches [3]. There are also approaches which use both bandwidth and buffer-based approaches [4]. However, current heuristic approaches are unpredictable and do not give consistent results as it relates to good QoE.

In this paper the term good QoE means the best allocation of system resources so that each player in a multi-player streaming environment gets low buffer underruns, high network bandwidth utilization, low unfairness, low instability and high average video quality. Thus, good QoE attempts to get better values for these five parameters, with minimal trade-offs, relative to existing methods. Interestingly, when there are two or more players sharing a network bottleneck link, good QoE becomes a critical issue [4].

Another attempt to model adaptive video streaming dynamics is stochastic-based. Markov Decision Processes [5] are used to help the player select future video segments. This method offers a tool for optimizing decision making when outcomes are partly random and partly under the control of the decision maker. The process goes through a finite set of states. At each state, the decision maker can choose a particular action from a given set. State transitions are random and the transition probabilities are different for different actions. Each action is associated with a reward and sometimes a penalty. Revenue (reward minus penalty) is used to evaluate the outcome for a given action taken at a given state. Solving an MDP problem means finding the best action for each state that will maximize the overall revenue. Once solved, the actions corresponding to the states are called the optimal policy. One method to solve an MDP is by using dynamic programming [6].

It is computationally complex to create MDP models that comprehensively take into account all the parameters that affect video streaming. Indeed most of the processing is done offline and few methods select more than five parameters. However, it is the selection of these key parameters that have the most influence on the video quality and consequently on the end-user viewing experience. In this paper we use a normalized ratio derived from incoming data rates, Data Rate Ratio (DRR), as the primary QoE parameter to build our MDP model. This parameter is exchanged amongst players via a third-party network device, with the goal of each player discovering each other DRRs.

At the client the proposed method treats the system as a blackbox and the result is measured by the QoE performance metrics, buffer underruns, bandwidth utilization, unfairness, instability and average quality. This method is common with fully bandwidth-based approaches. However, existing methods may also use the system as a whitebox. It observes client-side parameters, for example, buffer levels and other internal states of the system, which it further uses to optimize the system. The performance of the system is measured and optimized by those internal state parameters. This method is common with fully buffer-based approaches. Based on the literature our proposal is a stochastic bandwidth-based method.

In the MDP model, state transition matrices are created for each player in the adaptive streaming environment. A novel reward function is proposed based on player transitions between discretized DRRs. This directly influences the players to contribute in QoE optimization. This optimization is achieved by appropriately rewarding the discretized DRR parameter. The result is an optimal streaming policy geared towards good QoE video bitrate selection for all segments.

The contributions of this paper are as follows: 1) we propose a Markov decision process optimization problem for DASH player rate adaptation using a novel discretized DRR parameter; 2) we propose a new method called SHARE that results in a different policy for each player rather than the same policy for each player as in the case of other methods that use MDPs; and 3) we compare the performances of the new method with the conventional, the MDP-DASH (Markov Decision Process DASH) [7], and the QC-DASH (Quality Control DASH) [8] client-side algorithms.

This paper is organized as follows. Section II reviews the MDP-DASH and QC-DASH attempts made in the area of adaptive streaming. In Section III the new method is presented. This is a distributed decision making strategy based on Markov decision processes (MDPs), so that each player can make locally optimal decisions based on shared observations. Section IV gives the experimental settings and results. Finally, Section V consists of the conclusion and future work.

II. Related Work

There is a growing body of literature on the use of MDP to optimize video streaming. We now outline different ways MDPs are applied to adaptive video streaming. In [9] and [10], researchers found that bandwidth can vary severely in different locations. Adaptive streaming is modelled as an MDP problem to cope with varying network conditions [11]. The power consumption problem of video decoding can be effectively modelled as an MDP [12]. An MDP to optimize rate adaption of streaming video where the uncertainty in network bandwidth is modelled as a Markov chain with its own bandwidth states is given in [13] and [14]. In [8], [15], a stochastic dynamic programming (SDP) technique was proposed for rate adaption in DASH players, where the system rate is determined based on client buffer occupancy and bandwidth conditions. However, none of these studies used concepts similar to the SHARE approach to obtain their MDP policies.

Further, we implement two MDP models from the literature in this paper. These two models are described in the following paragraphs. These two MDP formulations model the details of SHARE closest to the ones investigated. However, they differ in the stated ways: (1) Both methods use different parameters to devise their MDP model. SHARE uses the single DRR parameter, while the other models use up to three parameters. (2) They use formulas to derive their state transition probability values, whereas SHARE uses historic and run-time experimental data to obtain these probabilities. (3) The methods use different rules to reward and penalize various actions, when compared to SHARE. Though, bitrate switches are used in all approaches, this is the sole parameter for the reward functions in SHARE. (4) The methods produce one policy for all the players to use, whereas SHARE creates a unique policy for each player. We believe that these fundamental changes to the MDP model will produce improved streaming policies. However, in multi-player multi-video bottleneck scenarios the complexity of the SHARE model increases more rapidly than other approaches. With MDP modelling this is an inherited problem, but the multiple polices creates a greater challenge for SHARE-based player streaming.

We label the two studied models as MDP-DASH (Markov Decision Process DASH) and QC-DASH (Quality Control DASH). The goal of MDP-DASH [7] is to explore different methods to reduce decision making overhead for DASH-based adaptive video players. The states depend on the quality level of the downloaded chunk and the time available before the chunks playback deadline (current buffer occupancy measured in time). The actions (decisions) are the quality level of the next chunk to be downloaded. Higher rewards are given for 'watching' a chunk in higher quality. There is a penalty for missing a deadline as well as switching quality from the present chunk to the next. For a given action (chunk size), state transition probabilities are calculated based on the Cumulative Distribution Function (CDF) of the network bandwidth. The CDF allows calculation of the probability of a given buffer occupancy when the next chunk is downloaded. The buffer occupancy, together with the action (quality level decision), defines the next state. Transition probabilities will change with different CDF's. Different CDF's lead to different MDP strategies.

QC-DASH [8] uses Stochastic Dynamic Programming (SDP) to solve the MDP and aid adaptive video streaming. The three parameters to compute the state transition matrix are buffer level, average channel bandwidth and quality. The authors designed a cost function that penalizes situations that may lead to a reduction of the QoE. This computation is done offline, where the control policies map the environment information to the client requests. The main result is that the average quality requested with their algorithm is higher, but it also involves a related number of quality switches among segments.

Both MDP-based methods significantly improve QoE in terms of unfairness, re-buffering ratios, instability and video quality. However, we believe that if players knew about the state of their counterparts, then they can make better decisions regarding what bitrates they request from the server. Primarily, we believe that players should select bitrates that lead to good QoE.

III. Model Description

In an adaptive video streaming model there are n adaptive video players. Each player, i, could receive different video data from different servers. Each video consists of various quality levels, $Q^i_{j,k}$ where j represents the actual video quality levels $(1, 2, \cdots, L_{max})$, and k represents the server number $(1, 2, \cdots, n)$. Each quality level supplied at the server side corresponds to a received video bitrate value at the client side, $B^i_{j,k}$.

The DRR, $R^i_{j,k}$, is a measure of the received bitrate quality or levels a video player i is currently experiencing and has the definition given in (1). The strong players have higher discretized values than weak players.

$$R^i_{j,k} = \frac{B^i_{j,k} - B^i_{1,k}}{B^i_{L_{max},k} - B^i_{1,k}} \tag{1}$$

Each player P_i periodically calculates its DRR value and broadcast it to an external third-party device. The broadcast is made up of the DRR, sequence number and the IP address of the broadcasting player. The goal is for the external device to have a state array containing the most up-to-date DRR values for all the players in the network. The normalization of the requested bitrates into DRR values give the advantage that two or more players with different quality levels can communicate information about their current quality. For the first 2 minutes the device uses historic data from a conventional player [16].

TABLE I. State Arrays for 3 Adaptive Players with 3 Discretized DRRs

State ID	Array	State ID	Array	State	Array	State	Array
1	111	8	132	15	223	22	321
2	112	9	133	16	231	23	322
3	113	10	211	17	232	24	323
4	121	11	212	18	233	25	331
5	122	12	213	19	311	26	332
6	123	13	221	20	312	27	333
7	131	14	222	21	313		

It then listens at 2 minute intervals. It calculates the player policies using MDPs, see Section III-D. The player policies are then sent to the respective players. This method enables players to cope with changing network conditions.

A. State and Action Space for MDP

The state array is made of a discrete representation of the DRR value of each player. Thus, for the three player model there would be three discretized DRR values at a single time stamp. Let us take the first position in the state array, which is allocated for player one. If the DRR value falls below 0.33 then a discretized DRR value of 1 is assigned to the first player. If the DRR value falls between 0.33 percent and 0.66 percent, inclusive, then a discretized DRR value of 2 is assigned to the first player. Finally, if the DRR value is above 0.66 then a discretized DRR value of 3 assigned to the first player. The same is done for the second and third players, who occupy the second and third positions of the state array. To summarize: the three player model would have three players at position one, two and three in the state array and each position can have any of three discretized DRR values, 1, 2, or 3. Note that players with same discretized DRR does not mean same quality or same bandwidth utilization. This is because the single discretized DRR value can represent many different bitrates or levels in the video. However, the discretized DRR values 1, 2, and 3 does indicate increasing bitrate values. Hence, a player with a discretized DRR value of 3 does indicate that the discretized DRR was obtained from a higher bitrate than one at 2. Table I shows all the possible state arrays for a 3 player model with 3 discretized DRRs.

In order to apply MDP techniques, the state transition model of the SHARE method needs to be devised. A rate decision is made at stage k for segment $k+1$, so the total number of stages equals the number of segments K. For stage k, we denote the state array as u_k. Once segment k has been completely downloaded, the video player P_i controller applies control action $a_k^{P_i}$ to determine the quality level for segment $k+1$ based on the information in state array u_k. At every time instance, each player may experience different bitrates, thus their individual 'combined' mapping to the global state may differ. Henceforth, each player may have a different policy and action given the value of state array u_k.

B. State Transition Probabilities for MDP

A decision made at the current state will influence the transition probability of reaching to a specific state at the next step for a given player. For the MDP model, if 3 players with 3 discretized DRRs are used, each transition matrix will be 27×27. For a video with m levels, there are m possible actions that can result from each player in the system having state array u_k. For a player P_i, given any state u_k and action a_k, the transition probability of the MDP is given in (2).

$$P(u_{k+1}|u_k, a_k) = Pr(u_{k+1}|u_k, a_k(u_k)) \tag{2}$$

There are m transition matrices. These transition matrices initially can be calculated by collecting historic data from a configured network containing n video players using the conventional method to HAS. For each player, P_i and each state array u_k, counts can be taken of the number of times the state array changes to u_{k+1} for a given video quality level.

C. Reward Function for MDP

In an MDP the effectiveness of an action has to be evaluated. In order to do this we define a reward value r_k related to action a_k at stage k. The reward value is a function of state array u_k given in (3).

$$r_k = R(u_k) \tag{3}$$

To best explain how the reward function works consider using 3 players with 3 discretized DRRs, where the first element in the array corresponds to a discretized DRR value of player 1, the second to the discretized DRR value of player 2 and the third to the discretized DRR value of player 3. We consider the first case where all three players have different discretized DRR values, the maximum, the middle and the minimum. Thus, one player will have a discretized DRR of 1, one player will have a discretized DRR of 2, and one player will have a discretized DRR of 3. Here, the player with the minimum discretized DRR value is rewarded a value of 2 for moving up by 1 from its current quality level, a value of 1 for moving up by 2 from its current quality level or a value of 0 for any other movements. The player with the maximum discretized DRR value is rewarded a value of 2 for moving down by 1 from its current quality level, a value of 1 for moving down by 2 from its current quality level or a value of 0 for any other movements. The player with the middle discretized DRR value gets a reward of 0. The result is that large quality shifts are penalized, while smaller shifts are rewarded.

Now consider the case where one player obtains the minimum discretized DRR value (1 player has a discretized DRR value of 1) and the two other players have a tie for the maximum discretized DRR value (2 players have discretized DRRs of either 2 or 3). The player with the minimum discretized DRR value is rewarded a value of 2 for moving up by 1 from its current quality level, a value of 1 for moving up by 2 from its current quality level or a value of 0 for any other movements. This encourages upward quality shifts for the player with low quality. The other two players IP addresses are sorted in ascending order. The one with the higher IP is chosen as the player with the highest discretized DRR value. This player is rewarded a value of 2 for moving down by 1 from its current quality level, a value of 1 for moving down by 2 from its current quality level or a value of 0 for any other movements. The player with high quality is able to reduce its high bitrate selection request. The player with the middle discretized DRR value gets a reward of 0, as with the first case. The antithesis to the previous case is where one player obtains the maximum discretized DRR value (1 player has a discretized DRR value of 3) and the two other players have a tie for the minimum discretized DRR value (2 players have discretized DRRs of either 1 or 2). In this case the player with the lowest discretized DRR value (now the tie is broken by selecting the player with the smaller IP address) is rewarded a value of 2 for moving up by 1 from its current quality level, a value of 1 for moving up by 2 from its current quality level or a value of 0 for any other movements. The player with the maximum discretized DRR value is rewarded a value of 2 for moving down by 1 from its current quality level, a value of 1 for moving down by 2 from its current quality level or a value of 0 for any other movements. Again the weak and strong players are allowed to adjust their bitrates, but note that these shifts in bitrates may result in the player staying with the same discretized DRR value. This occurs as a single discretized DRR value may represent many different bitrates. The player with the middle discretized DRR value gets a reward of 0, as with the first and second cases.

Further, consider the last case where all players have the same discretized DRR value. The last case has three alternatives. The first alternative is where all 3 players have the highest discretized DRR values (3 players have a discretized DRRs of 3). The second alternative is where all 3 players have mid range discretized DRR values (3 players have a discretized DRRs of 2). The third alternative is where all 3 players have low discretized DRR values (3 players have a discretized DRRs of 1). These cases are simple as they represent players in a convergent state. Hence, no value reward is given to players. The tendency of the players based on the increases and decreases in bitrate requests, is to eventually converge towards discretized DRR harmony, that is, when all players have the same discretized DRR values. Here, a convergence occurs amongst video player discretized DRR values. However, one may argue that all players at discretized DRR values of 1 or 2 is not ideal. Consequently, it can be said that a distributed streaming system under the second or third alternative would not thrive, even if discretized DRR harmony is present. This is because end users may suffer from sub-optimal QoE, due to players making a request for low level bitrates. But the effects of (Transmission Control Protocol) TCP at the transport layer comes into player in the latter two alternatives. TCP would detect the low bandwidth usage of the bottleneck link and the players would slowly ramp up their bitrate requests. This effect would result in the discretized DRR harmony having a greater tendency of moving towards higher discretized DRRs. In additions because the DRR values are discretized, alternatives one and two may be fruitful for some players who require lower bitrates for good QoE. For example, when players are downloading videos with different qualities or in networks with very low bandwidths. Note that at most two players are involved in the adjustment of quality levels at stage k and the r_k can be defined as follows:

$$r_k = \begin{cases} 0 & \text{if quality movement is more than two,} \\ 1 & \text{if quality movement is two,} \\ 2 & \text{if quality movement is one.} \end{cases}$$

Where r_k given the values 1 or 2 apply when all players have different discretized DRRs or a tie occurs between two players, and the value 0 for other cases. Note that there are m reward matrices where m represents the number of video quality levels.

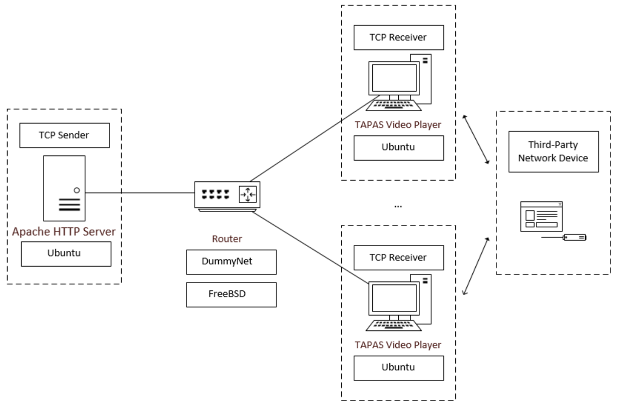

Fig. 1. Testbed setup.

D. Finding Optimal Policy for MDP

The core problem of MDPs is to find a policy for each video player controller. This is a function that given a state array will return an action (the next quality to request from the server). The MDP finds a policy for a player that maximizes the cumulative reward function of that player. Value iteration [3] is a well known algorithm for finding such a policy and we use MATLAB pre-build tools to solve the MDP formulation in this paper. In the case with up to five players, three discretized DRRs values can be used. However, to reduce complexity and state space size with more than five players the number of discretized DRRs values can be reduced to two. This is based on the assumption that as the number of players increase the need for having granular discretized DRR values decreases.

IV. EXPERIMENTS

The emulated environment consists of three TAPAS [17] client video players, a third-party device, a Dummynet [18] bottleneck link and a single server machine. Fig. 1 shows the network architecture used. The client, third-party device and server machines used the Ubuntu 15.04 operating system and the bottleneck link used FreeBSD. In Fig. 1, the router acts as the bottleneck link. The maximum throughput allowed at the bottleneck link is set to 5 Mbps. The video on the server contains 327 video segments and is 10.88 minutes long. Each segment is 2 seconds in length. The video consists of seven different quality levels, ranging from 46 kbps to 4.2 Mbps. Screen resolutions for the different quality levels are 320x240, 480x360, 1280x720, and 1920x1080. The four resolutions are divided amongst seven bitrates. Thus, there are similar resolutions available with different bitrates. The media type for the video is MP4, which is encoded at 24 frames per second (fps).

The conventional method is used to generate the data that is needed to calculate the transition matrices. Players share DRR values using a broadcast mechanism at each client as described in Section III. Twenty-five experiments are run, each with 6500 state changes. State transition matrices are created as described in Section III-B. There are seven levels so this results in seven different 27×27 state transition matrices for each player. In addition, reward matrices are created as described in Section III-C. These seven transition and reward matrices per player are used as inputs to the MDP method that is executed in MATLAB.

The output of the MDP process is 3-tuple policy. The action (client bitrate request level) given by a policy depends on the actual videos used and network conditions. An example of the output for this experiment is given in Table II. During runtime the clients will share DRR value information. In this way they will be able to form their own state arrays. The state arrays are identical across all players in the network. Then depending on the network conditions at the client a look up of action based on the respective policy is made possible.

See Table II for a sample policy[1]. Recall that the video used has seven quality levels. Assume the lowest three bitrates (level 1, 2 and 3) represent a discretized DRR of 1, the next two (level 4 and 5) by a discretized DRR of 2 and the last two (level 6 and 7) a discretized DRR of 3. As an example the state array of player 1, 2 and 3 consists of three values, 3 2 2. The value 3 indicates the discretized DRR value of player 1. The player then looks up the value it should request. The quality level then requested is 5. Likewise, the second player looks up its own state which has a discretized DRR value of 2. Player 2 follows its own policy and requests a quality level of 5. The third player observes that its discretized DRR value is 2. It follows its own policy and requests a quality level of 7. This is a good example of how the reward function affected the policy. The weak player with a discretized DRR value of 2 (tie is broken by selecting player with smaller IP address) got a next bitrate request of 7, while the strong player drops its bitrate request to 5. The other player keeps it bitrate high and so requests a quality level of 7. Note that the global state of the players are important during streaming at any time instant. However, the players can be downloading different segments from different videos from alternative servers and the method will still be effective. Synchronicity of player are required by SHARE. However, start or pause-time should not affect the effectiveness of the method as the players would now be reacting to the changed DRRs and what quality segments their respective policy requests for them to download. The policies shows that there is a general trend towards selecting better bitrates with a harmony existing when players tend to move to lower or higher rates adjusting to bottleneck network conditions. A close look at the values in the policies show that the reward function plays a vital role in the decision a player makes to select its next bitrate.

The following MDP client player algorithms are implemented along with the proposed method:

1) The conventional controller [16]
2) MDP-DASH [7]
3) QC-DASH [8]

A. QoE Evaluation Metrics

The metrics used to evaluate the performance of the client players are:

[1]This policy is the actual policy for the single bottleneck experiment in Section IV-B

TABLE II. Sample Policies for Players 1, 2 and 3

State Array	Player 1	Player 2	Player 3
111	1	1	1
112	6	1	1
113	1	4	1
121	4	6	4
122	4	5	5
123	7	6	1
131	1	7	1
132	6	6	5
133	7	7	1
211	1	1	6
212	6	4	5
213	7	4	5
221	5	4	5
222	5	5	5
223	7	7	6
231	4	1	4
232	6	5	5
233	6	7	5
311	1	4	1
312	6	4	7
313	7	1	6
321	5	6	7
322	5	5	7
323	6	7	6
331	1	6	4
332	6	6	7
333	6	6	6

1) Buffer Underruns [19]: when the playout video rate is larger than the download bandwidth the video buffer gets depleted rapidly and becomes empty, a buffer underrun occurs. This causes the video to stall until more bits refill the buffer. The re-buffering ratio is the total time spent re-buffering divided by the total experiment time as defined in (4).

$$rebuf\_ratio = \frac{totalRebuffingTime}{totalExperimentTime} \quad (4)$$

.

2) Bandwidth Utilization [20]: the aggregate throughput or bitrate during a time interval divided by the available bandwidth in that interval [9] and is defined in (5).

$$utilization = \frac{\sum_{t=0}^{N-1} tp_t}{bw} \quad (5)$$

where tp_t is the throughput at time t, bw is the bandwidth for N time steps

3) Unfairness [21]: measured by 1 minus the Jain fairness index. The Jain fairness index is defined in (6).

$$JainFair_t = \frac{(\sum_{j=1}^{z} tp_j)^2}{z \sum_{j=1}^{z} tp_j^2} \quad (6)$$

where tp_j is the throughput of player j and z is the number of players.

4) Instability [16]: a measure of the total quality differences between successively downloaded segments. The instability for player j receiving video at quality Q_j at time t is given by Equation 7.

$$ins_j = \frac{\sum_{d=1}^{k} Q_j[t-d] - Q_j[t-d-1] \cdot w(d)}{\sum_{d=1}^{k} Q_j[t-d] \cdot w(d)} \quad (7)$$

where $w(d) = k - d$ is a weight function that puts more weight on more recent samples. k is selected as 20 samples in our experiments.

5) Average Quality [15]: considers the geometric mean of all the received video segments. It not only maximizes video quality, but also minimizes quality variance among N segments.

$$average\_quality = \sqrt[N]{r_1 r_2 r_3 \cdots r_N} \tag{8}$$

B. Results

The experimental results for the single constrained bottleneck experiment are presented in Table III. Conditions:

- *Video: Elephants dream [2]. The video duration is 10.79 min.*
- *The available bandwidth is set to 5 Mbps.*

TABLE III. QoE METRICS: SINGLE CONSTRAINED BOTTLENECK

	CONV.	MDP-DASH	QC-DASH	SHARE
Utilization Index	0.68	0.73	0.72	0.74
Unfairness Index	0.297	0.116	0.133	0.014
Re-buffering ratio	0.012	0.008	0.010	0.001
Instability	0.013	0.009	0.010	0.001
Average Quality	3.19	3.42	3.27	3.43

The SHARE distributed MDP method outperforms the conventional, MDP-DASH and QC-DASH methods. It performs the best in terms of channel utilization, fair sharing, re-buffering, stability and quality. This shows that the optimal policies produced by SHARE are well suited for bottleneck environments. MDP-DASH outperforms QC-DASH in all metrics. Finally, QC-DASH outperforms the conventional in all metrics.

However, it must be noted that SHARE is a good QoE sharing-based method. Its main purpose is to help adaptive video streaming players get good QoE relative to their video quality levels. Thus, the overall network utilization and the individual user experiences among the three players will be better than the conventional, MDP-DASH and QC-DASH. However, it must be noted that the re-buffering ratio and average quality value are very close to that of MDP-DASH. In this context SHARE can be said to provide high bandwidth utilization, fairness and stability with minor improvements (possible trade-offs may occur here) in re-buffering and video quality when compared to other approaches.

Further experiments under time-varying bandwidth conditions are considered. The bandwidth at the bottleneck link is changed at various times throughout the experiment. Conditions:

- *Video: Elephant's dream[2]. The video duration is 10.79 min.*
- *The available bottleneck bandwidth capacity, is initialized to 5 Mbps.*
- *At* $2/10$, $4/10$, $6/10$ *and* $8/10$ *of the experiment duration the bottleneck bandwidth capacity drops by* $1/2$ *for 60s, respectively.*

In the presence of time-varying bandwidth, SHARE did the best in all metrics, (Table IV) and player policies in Table V. This exhibits that SHARE can act in an optimal manner in the presence of bandwidth variations. The MDP-DASH performs better than QC-DASH only in fairness. The bandwidth utilization is better because of the extra parameter in the QC-DASH MDP model which considers this metric. The conventional performs the worst. SHARE does not seem to have any trade-offs when network conditions are time-varying. However, a closer look shows that the instability index is closest to the other three approaches compared to the other metrics. Stability is treated as the trade-off for a community of SHARE players in time-varying network conditions.

TABLE IV. QoE METRICS: TIME-VARYING BANDWIDTH

	CONV.	MDP-DASH	QC-DASH	SHARE
Utilization Index	0.56	0.59	0.60	0.67
Unfairness Index	0.343	0.165	0.187	0.081
Re-buffering ratio	0.093	0.072	0.050	0.002
Instability	0.037	0.019	0.027	0.011
Average Quality	2.45	2.73	2.87	3.05

Overlapping ON periods are a huge problem for competing adaptive video streaming players at a bottleneck link [22]. The lowering of bitrate requests at the application layer for the player with the higher discretized DRR reduces the ON effect of this player caused by accelerating TCP rates at the network layer. This gives the player with the lowest discretized DRR an opportunity to now claim a larger portion of the bandwidth by increasing its bitrate request. This holds more so for the constrained bottleneck link compared to the experimental condition of a time-varying bandwidth being present. Note that players using the SHARE approach in time-varying network conditions have a stability trade-off. This is expected as there is less bandwidth for players so chances at having successful download request attempts aimed at increasing and decreasing bitrates becomes less likely. However, the time-varying bandwidth experiment shows that even with the reduction in bandwidth the SHARE approach still allows the player with the lowest bitrate to have a better chance at increasing the TCP transmission rate at the transport layer. The result of this probabilistic increase and decrease in bitrate requests may eventually reduce the oscillatory effects associated with ON-OFF traffic patterns.

V. CONCLUSION

A novel framework called SHARE for using MDPs in adaptive video streaming in bottleneck network environments is proposed which involves a discretized DRR state array of multiple client players. A reward function is presented, which penalizes large jumps in quality level shifts. The output of the MDP process is a separate policy for each client player. We compare with other MDP models to show the effectiveness of the SHARE method. SHARE's design gives good QoE for players. However, possible trade-offs in buffering and video quality may exist. Future work includes measuring the performance of the SHARE method in experiments with varying start times, in the presence of long-lived TCP flows, and multi-video scenarios.

REFERENCES

[1] Z. Li, Y. Huang, G. Liu, F. Wang, Z.-L. Zhang, and Y. Dai, "Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices," in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*. ACM, 2012, pp. 33–38.

[2]http://www.itec.uni-klu.ac.at/dash/?page_id=207

TABLE V. POLICIES FOR PLAYERS 1, 2 AND 3 IN NETWORK WITH TIME-VARYING BANDWIDTH

State Array	Player 1	Player 2	Player 3
111	2	2	8
112	1	1	1
113	1	1	1
121	1	1	1
122	6	6	8
123	1	1	1
131	1	1	1
132	1	1	1
133	8	8	8
211	1	1	1
212	6	8	7
213	7	7	6
221	6	6	6
222	8	5	6
223	8	7	8
231	1	1	1
232	8	8	7
233	8	4	6
311	1	1	1
312	1	1	1
313	8	8	8
321	1	1	1
322	7	8	6
323	7	5	8
331	8	8	8
332	8	4	8
333	1	1	6

[2] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2013, pp. 19–24.

[3] J. Park and K. Chung, "Client-side rate adaptation scheme for http adaptive streaming based on playout buffer model," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 190–194.

[4] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 97–108.

[5] J. Filar and K. Vrieze, *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[6] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[7] A. Bokani, M. Hassan, and S. Kanhere, "Http-based adaptive streaming for mobile clients using markov decision process," in *Packet Video Workshop (PV), 2013 20th International*. IEEE, 2013, pp. 1–8.

[8] S. García, J. Cabrera, and N. García, "Quality-control algorithm for adaptive streaming services over wireless channels," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 50–59, 2015.

[9] P. Deshpande, X. Hou, and S. R. Das, "Performance comparison of 3g and metro-scale wifi for vehicular network access," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 301–307.

[10] J. Yao, S. S. Kanhere, and M. Hassan, "An empirical study of bandwidth predictability in mobile computing," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. ACM, 2008, pp. 11–18.

[11] D. Jarnikov and T. Özçelebi, "Client intelligence for adaptive streaming solutions," *Signal Processing: Image Communication*, vol. 26, no. 7, pp. 378–389, 2011.

[12] N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. Van Der Schaar, "Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems," *IEEE transactions on multimedia*, vol. 15, no. 2, pp. 268–278, 2013.

[13] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in *Proceedings of the 3rd multimedia systems conference*. ACM, 2012, pp. 167–172.

[14] M. Xing, S. Xiang, and L. Cai, "Rate adaptation strategy for video streaming over multiple wireless access networks," in *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012, pp. 5745–5750.

[15] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over http with scalable video coding," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 149–154.

[16] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[17] L. De Cicco, V. Caldaralo, V. Palmisano, and S. 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*. ACM, 2014, pp. 1–6.

[18] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[19] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis, "Trickle: Rate limiting youtube video streaming." in *USENIX Annual Technical Conference*, 2012, pp. 191–196.

[20] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and experimental evaluation of network-assisted strategies for http adaptive streaming," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 3.

[21] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

[22] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. 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*. ACM, 2012, pp. 9–14.