Restless Video Bandits: Optimal SVC Streaming in a Multi-user Wireless Network

S. Amir Hosseini and Shivendra S. Panwar
Department of Electrical and Computer Engineering, NYU Tandon School of Engineering

Abstract—In this paper, we consider the problem of optimal scalable video delivery to mobile users in wireless networks given arbitrary Quality Adaptation (QA) mechanisms. In current practical systems, QA and scheduling are performed independently by the content provider and network operator, respectively. While most research has been focused on jointly optimizing these two tasks, the high complexity that comes with a joint approach makes the implementation impractical. Therefore, we present a scheduling mechanism that takes the QA logic of each user as input and optimizes the scheduling accordingly. Hence, there is no need for centralized QA and cross-layer interactions are minimized. We model the QA-adaptive scheduling and the jointly optimal problem as a Restless Bandit and a Multi-user Semi Markov Decision Process, respectively in order to compare the loss incurred by not employing a jointly optimal scheme. We then present heuristic algorithms in order to achieve the optimal outcome of the Restless Bandit solution assuming the base station has knowledge of the underlying quality adaptation of each user (QA-Aware). We also present a simplified heuristic without the need for any higher layer knowledge at the base station (QA-Blind). We show that our QA-Aware strategy can achieve up to two times improvement in user network utilization compared to baseline algorithms in a real network setting.

I. INTRODUCTION

Video and real time applications are the largest consumer of mobile wireless data (40% during peak consumption) in North America, and it is predicted that this trend will continue. This calls for more intelligent usage of the available spectrum and more bandwidth conserving techniques for video delivery. For this purpose, adaptive video delivery over HTTP has been standardized under the commercial name DASH. DASH can also be implemented using the scalable extension of the video codec H.264/SVC and H.265/SHVC. In SVC, each segment is encoded into a base layer containing the minimum quality representation, and one or more enhancement layers for additional quality. Apart from higher flexibility in segment delivery, SVC also benefits the network in terms of caching efficiency and congestion reduction at the server. These benefits have led to efforts for commercial deployment of SVC. For instance, Vidyo and Google have begun a collaboration for implementing SVC on WebRTC using the VP9 codec. The process of delivering adaptive video using SVC in a wireless network can be broken into two separate tasks:

Quality Adaptation (QA): QA determines the order in which different layers of different segments must be requested by the user and is performed by an end-to-end application specified by the content provider (Netflix, Amazon, etc.). The adaptation policy is not specified in the DASH standard and therefore, depending on the user device, video application, content provider, etc., different vendors can use different policies.

Scheduling: In multi-user wireless networks, where the bottleneck is typically the access link, the base station determines how the time-frequency resources are shared among users. This task is referred to as scheduling, and it is a design choice of the network service provider. In general, the scheduling policy should ensure high Quality of Experience (QoE) and utilize the wireless resources efficiently.

The above two tasks can either be implemented independent of each other as shown in Figure 1a or jointly by the base station as illustrated in Figure 1b. A joint optimization would require considerable cross-layer functionality at the base station making it impractical and overly complex. It would also call for coordination between content providers and network operators which is undesirable because it forces the content provider to give away control over its content delivery process, which it may be reluctant to do for business reasons. On the other hand, separately optimizing the two tasks provides inferior system performance compared to the joint case. In this paper, we combine the two schemes in Figure 1 and we design a scheduling policy that adapts itself to any arbitrary QA policy that is implemented on each end user (QA-adaptive scheduling). In our proposed system model, end users can deploy any QA provided by the content provider. The network then takes the QA of each user as input and optimizes the scheduling accordingly. As a result, content providers will still have full control over the adaptation process and the scheduling will be adaptive to the underlying QA. Furthermore, this separation between QA and scheduling may allow service providers, bound by the evolving net neutrality rules, a new option to maximize QoE without explicit, and therefore possibly discriminatory, cooperation with content providers. The recently developed MPEG’s Server and Network Assisted DASH (SAND) technology, offers standardized messaging schemes and protocol exchanges for service providers and operators to enhance streaming experience while also improving network bandwidth utilization. The exchange of QA logic between content provider and the network can be done within the SAND framework.

For this purpose, we first formulate the QA-adaptive scheme and the joint optimization using the concept of Restless Ban-
Many others have investigated the joint problem of QA and scheduling for scalable video in wireless networks [16–25]. Among these papers, some have deployed network utility maximization techniques for solving the joint problem [16, 17, 20, 21, 25]. In [18], a gradient based method is used in which, in every time slot, the base station solves a weighted rate maximization problem to update the gradient. The majority of these schemes are myopic and obtain optimality in a real-time fashion which makes them suitable for live streaming events.

The authors of [23, 24] model the problem first as a Multi-user Markov Decision Process and solve it using an iterative sub-gradient method. Unlike the previous papers, the proposed schemes are foresighted, i.e., the effect of each decision on future actions is taken into account. However, they require complex iterative computations in every time slot. In all the above papers, scheduling and QA are jointly optimized and therefore suffer the shortcomings discussed in Section II.

Other papers have proposed simple collaboration mechanisms between content providers and network operators with the goal of improving existing QA schemes for DASH [26–29]. The main argument in these papers is that current QA mechanisms that fully rely on client based adaptation fail to deliver acceptable performance in terms of fairness, stability, and resource utilization. By providing network assistance through the exchange of system statistics between the network and the client, the QA policies can be improved. In [28], an OpenFlow assisted control plane orchestrates this functionality. An in-network system of coordination proxies for facilitating resource sharing among clients is proposed in [29]. The authors of [25] develop a scheme that leverages both network and client state information to optimize the pacing of different video flows. In [27], the bitrate of each requested stream is throttled to a certain range and a proportional fair scheduler shares the resources among the streams. In our work, we deploy a similar collaboration mechanism between the network and the content provider but for the purpose of optimizing the scheduling policy given that users may deploy any arbitrary QA.

To the best of our knowledge, no prior work has considered optimal QA-adaptive scheduling for scalable video in wireless networks. Furthermore, our proposed solution aims at optimizing the scheduling procedure in a foresighted manner and is suitable for video on demand, where buffering of content is possible.

III. System Model

In this section, we start by describing the network and video models used in the formulation. We then present a matrix representation to model arbitrary QA.

A. Network Model:

The network under consideration consists of $N$ users from the set $\mathcal{N} = \{1, 2, \cdots, N\}$ and a base station. The total bandwidth is denoted by $W_{tot}$ and is divided into $M$ equal subchannels, as in OFDMA. At each time slot, the base station
chooses $M$ ($M \leq N$) users and allocates time-frequency resources to each of them. We assume that for each user $n$, the channel has flat fading and the capacity of each subchannel follows a Markov chain with transition matrix $C_n$, where $C_{n,i,j} = P(c_{n,t+1} = j | c_{n,t} = i)$. The states of this Markov chain $c_{n,t}$ are taken from a finite set $C$ and represent the maximum achievable data rate per subchannel, which is a function of the available modulation and coding schemes in the network. We also assume that the channel variation is slow enough so that the download rate remains constant over one time slot.

### B. Video Model:

The users are streaming scalable video, each encoded into equal length segments of $\tau_{seg}$ seconds. The segments are encoded into $L$ quality layers. Throughout the paper, we refer to each individual layer of a segment as a sub-segment. We assume that sub-segments of the same layer are of equal rate and the layer rates are denoted by $Q = \{q_1, \ldots, q_L\}$. At each time slot, users receive rewards based on the quality of the video segments that are played back in that time slot. As measure for QoE, we use a reward function that maps the rate of the video that is played back to the perceived quality as follows [30]:

$$R = \begin{cases} e^{-\phi \left( \frac{R_{pen}}{R_{max}} \right)^{-\theta}} + \phi, & \text{no re-buffering} \\ R_{pen}, & \text{re-buffering} \end{cases}$$

where $R_{pen}$ is the rate of the video that is being played back and $R_{max} = \sum_{l=1}^{L} q_l$ is the maximum rate of that segment when all layers are present. The constants $\phi$ and $\theta$ are video-specific parameters of the quality model, and it is shown in [30] that after averaging over numerous video sequences, their values are equal to 0.16 and 0.66, respectively. If the playback header reaches a segment for which the base layer is not delivered, playback stalls and re-buffering occurs. In order to account for this in the reward function, we assign a penalty for all instances of re-buffering denoted by $R_{pen}$, with a value depending on the sensitivity to re-buffering. By setting the value of $R_{pen}$, we implicitly determine our desired delay-quality trade-off. Needless to say, in order to penalize re-buffering, $R_{pen}$ should be set to a value less than the reward obtained by only having the base layer. The lower the value of $R_{pen}$, the higher the penalty. The main purpose of using adaptive video instead of constant rate video is the ability to decrease the quality of the video whenever there is risk of re-buffering. Hence, we suggest a low value for $R_{pen}$ throughout our simulation study in order to avoid re-buffering as much as possible.

### C. Quality Adaptation:

In order to model delivery scenarios with arbitrary QA, we develop a matrix representation of the end user buffer and call it the policy matrix. In this section, we describe how the policy matrix for each QA is derived. In Section [LV] the policy matrix is used for the formulation of the optimization problem.

We define the policy matrix $P_{\pi_n}(c_{n,t})$ as a binary transition matrix representing the QA policy $\pi_n$ that is applied when user $n$ is in channel state $c_{n,t} \in C$ at time $t$. Assuming the policy to be stationary, we can drop the time index from now on. For a buffer limit of $b_{max}$, the policy matrix determines all possible sub-segment deliveries that are allowed by policy $\pi_n$ in channel state $c_n$ in one time slot. Since each layer can have any number of sub-segments between 0 and $b_{max}$, the size of the policy matrix is $(b_{max} + 1)^L \times (b_{max} + 1)^L$. Each row of this matrix represents a particular buffer state at any time slot prior to selecting the next sub-segments to deliver, and each column represents the state of the buffer right after policy $\pi_n$ is applied. Hence, if the element in the $i^{th}$ row and $j^{th}$ column of $P(c_n)$ is 1, it means that the policy chooses to download those sub-segments for which the buffer state changes from $i$ to $j$.

Figure 2 illustrates the concept of policy matrix with a simple example. It shows an end user buffer streaming a video that is encoded into a base and two enhancement layers. Suppose that under the current channel conditions, the user can receive one sub-segment in the current time slot ($c = 1$ Mbps). The current buffer state is denoted by $i = (6, 4, 1)$, showing the number of sub-segments per layer. Assume that the next sub-segment to be requested is from the second enhancement layer. However, since the slot is one second, one segment of the video will be played back and the final state of the buffer will be $j = (5, 3, 1)$. Therefore, the $i^{th}$ row of the policy matrix is constructed as follows (other rows are constructed in a similar fashion):

$$P(c_n)_{i,j} = \begin{cases} 1, & \text{if } j = (5, 3, 1) \\ 0, & \text{otherwise} \end{cases}$$

With this technique, any arbitrary QA mechanism can be modeled as a set of policy matrices, each representing a particular channel state. For the remainder of our analysis, we consider three different QA policies:

1) **Diagonal Buffer Policy (DBP):** Results from existing research [7], [8] suggest that it is optimal to pre-fetch lower layers first, and fill higher layers after. In this policy, which we call the diagonal policy, the user starts pre-fetching sub-segments from the lowest layer until the difference between the sub-segments of that layer and the one above reaches a certain pre-fetch threshold, at which point it switches to the layer above, and this continues for all layers. This way, the difference between the buffer occupancy for each layer with the subsequent upper layer is kept at the fixed pre-fetch threshold. The policy depicted in Figure 2 is an example of the diagonal policy. It should be noted that each two neighboring layers can have different pre-fetch thresholds depending on their respective segment sizes, the additional video quality they provide and design preferences.

2) **Channel Based Policy (CBP):** In this scheme, users conservatively request more base layers whenever they are in bad channel conditions and gradually become more aggressive and request more enhancement layers as the channel condition improves [12].
There are many different ways to design a CBP or BPP policy. In Section IV we describe the particular CBP and BPP policies we used for the simulations.

IV. PROBLEM FORMULATION

In this section, we first formulate the QA-adaptive scheduling as a Restless Bandit (RB). In order to compare the optimal solution of this formulation with a jointly optimal QA and scheduling scheme, we also formulate the latter using a MUSMDP.

A. QA-adaptive scheduling

We assume that similar to Figure 2 QA is determined by the content provider. The scheduler takes the QA of each user as input prior to the start of the streaming process and optimizes the schedule accordingly. Each time the previously requested segments are delivered, users request new segments from the video server. They also send Channel Quality Indicator (CQI) messages to the base station every time slot. The video server sends the requested segments to the base station where they are buffered and scheduled for delivery.

In order to formulate the RB problem, we first model the state space of any user \( n \), denoted by \( S_n \). \( S_n \) is defined as the combination of the instantaneous channel state \( c_n \) and the current state of the buffer. We define the state of the buffer as a vector \( b_n \) representing the number of sub-segments the user has currently stored in the buffer for each layer, i.e., \( b_n = (b_{n,1}, \ldots, b_{n,L}) \). Therefore, the state space can be represented as \( S_n = \{ (c_n, b_n) | c_n \in \mathcal{C}, b_{n,j} \in \{0, \ldots, b_{\text{max}}\} \} \) and is of size \(|\mathcal{C}|(1 + b_{\text{max}})^L\). At each time slot \( k \), the policy taken by the scheduler is in the form of an action vector \( a_n = (a_{n,k})_{1:N} \) of size \( N \), where \( a_{n,k} \) is set to one for scheduled users and zero for the rest. Hence, each user can be modeled as a bandit that is either operated in a slot or not. Since the scheduler does not control the users action due to the arbitrary QA, the bandits are uncontrolled and therefore, satisfy all necessary conditions for RB [6].

The transition from one state to the next depends on the channel transition matrix and the policy matrix as well as if the user was scheduled (active) in that time slot or not (passive). The structure of the transition matrix is similar to the policy matrix with the difference that here we also include the instantaneous channel state. We define two state transition matrices for the active and passive users and denote them as \( H_n^1 \) and \( H_n^0 \), respectively. In the passive case, the user cannot request any new sub-segments and can only play back the existing segments in the buffer. Therefore, the passive policy matrix \( P_n^0 \) indicates transitions for which the occupancy of each layer is decremented by one. If no base layer is left in the buffer, no playback is possible, and therefore, no change occurs in the state of the buffer. We count this as an instance of re-buffering. The same procedure is followed for all channel states and we can write the state transition matrix \( H_n^0 \) as follows:

\[
H_n^0 = C_n \otimes P_n^0,
\]

where \( \otimes \) is the Kronecker product. For the active state transition matrix \( H_n^1 \), we need to create the policy matrices for all channel states \( (\mathbf{P}^{\pi_n}(c_n), c_n \in \mathcal{C}) \) since, depending on the available data rate, a different number of sub-segments can be delivered in every time slot. After determining the policy matrices for all channel states, the active state transition matrix can be derived as:

\[
H_n^1 = \begin{bmatrix}
C_{n,1,1} & \mathbf{P}^{\pi_n}(1) & \cdots & C_{n,1,|\mathcal{C}|} & \mathbf{P}^{\pi_n}(1) \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
C_{n,|\mathcal{C}|,1} & \mathbf{P}^{\pi_n}(|\mathcal{C}|) & \cdots & C_{n,|\mathcal{C}|,|\mathcal{C}|} & \mathbf{P}^{\pi_n}(|\mathcal{C}|)
\end{bmatrix}
\]

The objective function is the expected discounted sum of rewards received by the users throughout the streaming process and is expressed as:

\[
\max_u \mathbb{E}_u \left[ \sum_{k=0}^{\infty} \sum_{n \in \mathcal{N}} R_{s_n,k}^{\pi_n(k)} \right],
\]

where \( \beta \) is the discount factor (\( 0 < \beta < 1 \)) and \( R_{s_n,k}^{\pi_n(k)} \) is the reward received by user \( n \) if it is in state \( s_n,k \) in time slot \( k \) and chooses action \( a_{n,k} \), and is calculated according to (1). The goal of the optimal scheduler is to determine the optimal policy \( u \) in such a way that the expected sum of received rewards is maximized with respect to the resource constraint bounding the number of active users at each time slot to \( M \). We assume that segments that are downloaded in each time slot cannot be played back in the same slot. This is also the case for real video delivery in which after a segment is received, it takes some time for decoding and processing before it becomes available for playback. With this assumption, the immediate reward of a user is independent of the immediate action of the scheduler, and we can write \( R_{s_n,k}^{\pi_n(k)} = R_{s_n,k}^{\pi_n} = R_{s_n,k} \).

Next, we define for every user \( n \) and scheduling policy \( u \), the performance measures \( x_{s_n}^a(u) \), where \( a \) is either zero
or one for the passive and active case, respectively. These performance measures are then defined as follows:

$$x^{a}_{s_n}(u) = E_u \left[ \sum_{k=0}^{\infty} I^a_{s_n}(k) \beta^k \right],$$

where

$$I^a_{s_n}(k) = \begin{cases} 1 & \text{If in slot } k, \text{ user } n \text{ is in state } s_n \\
0 & \text{otherwise} \end{cases}$$

Essentially, $$x^{a}_{s_n}(u)$$ is the expected discounted amount of time that policy $$u$$ assigns action $$a$$ to user $$n$$ whenever the user is in state $$s_n$$. It is proved in [32] that the set of all Markovian policies for user $$n$$ following the transition matrix $$H^a_n$$, forms a polytope which can be represented as follows:

$$Q_n = \left\{ x_n \in R_{+}^{N_s \times \{0,1\}} \left| x^0_n + x^1_n = \alpha_{j_n} + \beta \sum_{i_n \in S_n} \sum_{a \in \{0,1\}} h^{a}_{n,i_n} x^a_{i_n}, j_n \in S_n \right\} , \right.$$  

where $$\alpha_{j_n}$$ represents the probability of $$j_n$$ being the initial state for user $$n$$. Consequently, the RB can be formulated as the following linear program:

$$\max_x \sum_{n \in N} \sum_{s_n \in S_n} \sum_{a \in \{0,1\}} R_{s_n} x^a_{s_n}$$

subject to:

$$x_n \in Q_n, \quad n \in N$$

$$\sum_{n \in N} \sum_{s_n \in S_n} x^1_{s_n} = \frac{M}{1 - \beta} , \quad$$

where $$x = (x_1, \cdots, x_N)$$ and $$x_i = (x^0_{s_i}, x^1_{s_i})_{s_i \in S_i}.$$ The objective function is derived by simply replacing the performance measure from (6) into the original objective function (5). It should be noted that since the quality adaptation is pre-determined, the users are modeled as uncontrolled agents, i.e., the action space only determines if the user is active or passive without specifying what users do in the active mode. This is by definition the classic Restless Bandit (RB) problem including a set of agents (bandits) where a fixed number of them are activated in every time slot. All bandits, whether active or not, change state and receive a reward for the next slot.

Although, based on our network model, the number of active users per slot is kept at a constant $$M$$, RB fixes the average number of active users per slot instead (see the resource constraint (11)). According to RB theory [33], if the size and capacity of the network grow infinitely large ($N, M \to \infty$) while $$\frac{M}{N}$$ remains fixed, the solution to RB asymptotically converges to the case with a constant number of users per slot. Therefore, for a fixed $$\frac{M}{N}$$, the larger the network, the closer RB will get to our desired solution.

We can simplify the problem for the cases in which all users are homogeneous, i.e., they have the same buffer limit, use the same QA, and have similar video and channel characteristics. In this case, the polytope constraint (8) becomes identical for all users.

**Theorem 4.1:** If the users in a network are homogeneous, an allocation policy that results in equal active and passive service time in each state for every user is an optimal allocation policy for RB. In other words, $$x^0_i = x^1_i$$ and $$x^1_i = x^0_i, \forall i,j \in N$$ in the optimal point, where $$x^0_i = (x^0_{s_i})_{s_i \in S_i}$$ and $$x^1_i = (x^1_{s_i})_{s_i \in S_i}$$.

**Proof** Refer to Appendix [A].

**Corollary 4.2:** If the homogeneity conditions hold, RB can be simplified to the following linear program:

$$\max_{x} \sum_{s \in S} \sum_{a \in \{0,1\}} R_{s} x^a_{s}$$

subject to:

$$x \in Q$$

$$\sum_{s \in S} x^1_{s} = \frac{M}{N(1 - \beta)}$$

The number of variables in the above linear program is equal to $$2|S|$$ and it has $$|S| + 1$$ constraints. Hence, we can model a network consisting of multiple homogeneous users using the state space of a single user, and therefore significantly decrease the number of variables and constraints of RB. We can optimize the scheduling for heterogeneous (not homogeneous) users, by grouping them into multiple groups each comprising of homogeneous users, and including only one sample user per group in the optimization. In this case, there will be one polytope constraint for each group and the resource constraint changes to:

$$\sum_{g \in \{1, \cdots, G\}} N_g \sum_{s_g \in S_g} x^1_{s_g} = \frac{M}{1 - \beta},$$

where $$g$$ is the index of the groups and $$N_g$$ and $$S_g$$ represent the number of users and the state space of user in group $$g$$, respectively. The complexity of the problem, therefore only depends on the number of groups and not the number of users per group.

**B. Joint Optimal QA and Scheduling**

In this section, we formulate the joint problem of optimally requesting new segments and scheduling users using the same network and video models described earlier. Similar to the RB, every user is modeled as an independent agent that changes state in a Markovian fashion. In this case, the action space for each user $$n$$ contains the passive action as well as the index of the layer of the sub-segment to be downloaded in the active case and can be represented as $$a_n = \{0, 1, \cdots, L\}$$. The key difference between this formulation and the RB is that previously, we assumed that actions are taken at every time slot, and that within each slot multiple sub-segments can be downloaded if the user is active. Now, we assume that if the user is active, each decision is made once the previous action is fully executed. Therefore, actions will have different
Before download $t = t_0$

After download $t = t_0 + \Delta t$

Initial position of playback
header (u = 0)

Final position of playback
header (u = $\Delta t$)

Downloaded
subsegment

Fig. 3: Buffer evolution model for the MUSMDP. Decisions are made at unequal time intervals, hence, the fraction of playback for each segment ($u$) is included in the state space.

durations depending on the state of the user and the problem becomes a MUSMDP.

In order to properly formulate the MUSMDP, we define the duration of a time slot to be the duration of the shortest possible action $\tau_{\text{slot}} = \frac{\min q_l}{\max C}$, where $q_l$ is the size of the layer $l \in \{1, \ldots, L\}$. Since the time slot duration is generally shorter than the duration of a segment, at every time slot we will have partially played back segments and the fraction of the playback needs to be included in the state space representation as well. Hence, the state space of user $n$ is denoted by $S_n = \{ (c_n, b_n, u_n) | c_n \in C, b_{n,l} \in \{0, \ldots, b_{\text{max},l}\}, u_n \in \{0, \ldots, \tau_{\text{seg}} - 1\} \}$, where $c_n$ and $b_n$ are defined as before and $u_n$ is the number of time slots that have passed since the current segment started playing back. For simplicity of notation we assume that $\tau_{\text{seg}}$ is an integer multiple of $\tau_{\text{slot}}$. Figure 3 illustrates the state space under the new setting in a simple way. In this figure, $u$ is initially zero. After the next sub-segment is delivered, playback continues and the playback header points at $u = \Delta t$.

We can turn a discounted SMDP into a discounted MDP by modifying the transition probabilities and reward function \cite{54}. First, we need to develop policy matrices for each of the actions. For the passive action, we assume that for the duration of one time slot, the user plays back the video in the buffer without adding any segment to it. For the active cases, we generate one matrix per layer $l$ represented by $H^l$. Since the channel transitions occur at every time slot, it is not known beforehand how many time slots each action takes. In order to determine the probability distribution of the duration of action $l$, we first define the random variable $\tau_l$ as the minimum number of time slots required to fully execute action $l$ given that the initial channel state is $c$ as follows:

$$f^l_c(\tau_l) = P \left( \sum_{k=1}^{\tau_l} c_k \geq \frac{q_l}{\tau_{\text{slot}}} \mid \tau_l \geq c \right),$$

where $c_k$ is the available rate in time slot $k$ and it transitions according to the channel matrix $C$. By considering all possible trajectories of channel state transitions, we can determine the joint probability distribution of $\tau_l$ and the final channel state, given the initial channel state, as $f^l(t,j) = P \left( \tau_l = t, c_{\tau_l+1} = j \mid c_1 = c \right)$.

In order to generate the policy matrix for user $n$ for the active cases, we consider the next state as the one in which $b_{n,l}$ is incremented by one and $u_n$ is incremented by the duration of the action. Therefore, the policy matrices $P^l(\tau_l, c_n)$ is also a function of the number of time slots required for executing action $l$. Given that we apply discounting on a slot by slot basis, the policy matrix $H^l_n$ can be written as:

$$H^l_n = \begin{bmatrix} V^l_{1,1} & \cdots & V^l_{1,|C|} \\
\vdots & \ddots & \vdots \\
V^l_{|C|,1} & \cdots & V^l_{|C|,|C|} \end{bmatrix}$$

where:

$$V^l_{i,j} = \sum_{k=1}^{\infty} e^{-sk}f^l_j(k, j)P^k(k, i),$$

where $e^{-s}$ is the discount factor and $l \in \{0, 1, \ldots, L\}$. For the passive case where $l = 0$, we have $f^0_j(k)$ equal to one for $k = 1$ and zero for all higher values since we define the passive action duration to be one time slot.

Suppose that user $n$ makes its $m$th decision at time $\sigma^m_n$. Then, the objective function of the MUSMDP is the following:

$$\max_u \sum_{n \in N} \sum_{m=0}^{\infty} \sum_{k=0}^{\tau_l} e^{-sk} s^m_n R_{s_n}^k,$$

where $e^{-s}$ is the discount factor, $\tau_l$ is the duration of action $l$ under policy $u$ and $R_{s_n}^k$ is the reward obtained by user $n$ in state $s_n$ after $k$ time slots have passed. The above expression can be turned into an equivalent discounted RB using the theorem below:

**Theorem 4.3**: The MUSMDP can also be formulated as a linear program as follows:

$$\max_y \sum_{n \in N} \sum_{s_n \in S_n} \sum_{a \in \{0,1,\ldots,L\}} \tilde{r}_{s_n}^a y_{s_n}^a$$

subject to:

$$y_n \in P_n, \quad n \in N$$

$$\sum_{n \in N} \sum_{a_n=1}^{L} \sum_{s_n \in S_n} y_{s_n}^a \tilde{r}_{s_n}^a = \frac{M}{1 - e^{-s}},$$

where:

$$\tilde{r}_{s_n}^a = \mathbb{E}_u \left[ \sum_{k=0}^{\infty} f^a_{s_n}^k(k)e^{-sk} \right],$$

and $f^a_{s_n}^k(k)$ is defined as \cite{7} and $\tilde{r}_{s_n}^a = \sum_{t=0}^{\infty} \sum_{c \in C} \left( \sum_{k=0}^{t} e^{-sk} R_{s_n}^k f^a_{s_n}(t, c) \right)$. Furthermore, the equivalent polytope constraint turns into the following:

$$P_n = \left\{ y_n \in \mathbb{R}^{|S_n| \times \{0,1,\ldots,L\}} \mid \sum_{a_n \in \{0,1,\ldots,L\}} y_{s_n}^a \right\},$$

and

$$\bar{P}_n = \left\{ y_n \mid \sum_{s_n \in S_n} y_{s_n}^a \right\} \cdot \alpha_n + \sum_{i_n \in \mathbb{N}} \sum_{a_n \in \{0,1,\ldots,L\}} h_{a_n,j_n}^a y_{s_n}^a, j_n \in S_n \},$$
Finally, \( \tau^a_{s_n} \) is the expected discounted duration of action \( a_n \) if the user is in state \( s_n \) when the action is taken and it can be written as \( \tau^a_{s_n} = \sum_{t=0}^{\infty} \sum_{c=0}^{\infty} \left( \sum_{k=0}^{t} e^{-sk} f^a_{s_n}(t,c) \right) \).

**Proof** Refer to Appendix B

C. Evaluation

Now, we evaluate the optimal performance of each of the above scenarios in order to determine the loss incurred by abandoning the jointly optimal scheme for the more practical QA-adaptive method. In order to show this, we solve the above problems with identical system settings described in Table I.

| Parameter                          | Value          |
|------------------------------------|----------------|
| number of users                    | 20             |
| subchannel bandwidth               | 2MHz           |
| channel states per subchannel      | (1,2,5,10) Mbps|
| number of layers                   | 2              |
| video rate per layer               | (1.1) Mbps     |
| buffer limit                       | 20s            |
| segment duration                   | 1s             |
| discount factor per second         | 0.99           |
| \( \tau_{pen} \)                   | 0              |

**TABLE I:** Parameters

Figure 4 shows the sum reward per user, which is the value of the objective function for the RB for each user. In these figures, the reward is plotted as a function of the load on the network, which we define as the average number of users that compete for one subchannel, \( \rho = \frac{N}{N_c} \). In order to vary the load on the network, we vary the number of available subchannels from 4 to 18 in increments of 2. We perform this evaluation for two network settings, with average rates of 4.5 Mbps and 2.5 Mbps, respectively. The QA schemes used are different variations of BPP, CBP, and two DBP schemes. In the BPP-x scheme, base layer segments are requested until the occupancy reaches \( x\% \) of the buffer limit. After that, full quality segments are requested. For CBP, if the channel is in the two low rate states, the user only requests base layer segments. In the third channel state, two-thirds of the resources are spent on requesting base layers and the rest is reserved for enhancement layers. In the best state, only full quality segments are requested. The DBP-x QA policy represents the diagonal policy with a pre-fetch threshold of \( x \) seconds.

In Figures 4a and 4b, the network is assumed to be homogeneous and all users in the network use the same QA. We observe that especially for low load scenarios, choosing a proper QA along with QA-adaptive scheduling will perform close to the jointly optimal case. For high load scenarios, the choice of QA becomes more important and using DBP with a large pre-fetch threshold that fills the buffer with base layer sub-segments up to the buffer limit and then starts downloading enhancement layers performs best, since by pre-fetching base layers, we lower the risk of re-buffering. Based on these results we argue that if an optimal QA-adaptive scheduler is used, an arbitrary QA can perform relatively close to the global optimum of the system. Figure 4c illustrates the performance of the QA-adaptive scheduling mechanism in a heterogeneous system where one half of the users deploy a DBP policy and the other half use a BPP policy. This figure is represents networks in which some content providers use SVC while others use single layered video. This model is useful because once content providers start offering adaptive video with SVC, they have to coexist with services that will still rely on single layered video. Our proposed QA-adaptive model is capable of devising scheduling policies for these mixed environments. Figure 4c shows that by using proper QA schemes, our scheduler performs within 85% of the jointly optimal scheme.

In the next section, we describe the scheduler operation for the QA-adaptive scenario.

V. ONLINE ALGORITHM

The solution of RB gives the long term performance measures, representing the total discounted time each action is applied in each state without explicitly stating what action should be taken in each time slot. In this section, we propose two heuristic algorithms for the QA adaptive scheme that are based on ranking the states of the users in terms of the scheduling priority, based on the optimal solution of RB.

The first algorithm is designed for the case in which the base station knows what particular QA is being used by each user beforehand, which we denote as QA Aware Scheduling. The second algorithm is designed to further simplify this procedure and imitate the functionality of the QA Aware Scheduling without actually knowing the QA of each user. We call this policy QA Blind Scheduling.

A. QA Aware Scheduling Algorithm

We start by defining the dual problem of RB as shown below. Without loss of generality, we show the dual of the simplified problem for the case with homogeneous users.

\[
\min_{\lambda} \sum_{s \in S} \alpha_s \lambda_s + \frac{M}{N(1-\beta)} \lambda \\
\text{subject to:} \\
\lambda_s - \beta \sum_{j \in S} h^0_{s,j} \lambda_j \geq N R^0_s, \forall s \in S, \\
\lambda_s - \beta \sum_{j \in S} h^1_{s,j} \lambda_j + \lambda \geq N R^1_s, \forall s \in S, 
\]

where the set of variables is denoted by \( \lambda = \{(\lambda_s)_{s \in S},\lambda\} \). We define reduced cost coefficients \( \gamma^a_s \) for \( \forall a \in S, a \in \{0,1\} \) as follows:

\[
\gamma^0_s = \lambda^*_s - \beta \sum_{j \in S} h^0_{s,j} \lambda^*_j - N R^0_s, \\
\gamma^1_s = \lambda^*_s - \beta \sum_{j \in S} h^1_{s,j} \lambda^*_j + \lambda - N R^1_s, 
\]

where \( \lambda^*_s \) is the optimal value for \( \lambda_s \).

The reduced cost coefficient \( \gamma^2_s \) represents the rate of decrease in the objective function in RB per unit increase in the variable \( x^a_s \). For example, if in a particular time instant, two users are in states \( s_1 \) and \( s_2 \), respectively, such that \( \gamma^1_{s_1} > \gamma^1_{s_2} \), scheduling the user in \( s_2 \) will cause less reduction.
in the objective function and should be prioritized over the other. By using this characteristic and the fact that due to complementary slackness, either $\gamma_s^a = 0$ or $x_s^a = 0$ for $\forall s, a$, we can derive a ranking scheme for all states as shown in Algorithm 1.

**Algorithm 1 QA Aware Scheduling**

**Step 0:** Solve RB and its dual and determine $x_s^{a,0}$, $x_s^{a,1}$, $\gamma_s^0$ and $\gamma_s^1 \forall s \in S$.

**Step 1:** Set $Q_1 = \{s \in S | x_s^{1} > 0\}$.

**Step 2:** Sort $Q_1$ in descending order of $\gamma_s^0$.

**Step 3:** Set $Q_0 = \{s \in S | x_s^{0} > 0\}$.

**Step 4:** Sort $Q_0$ in ascending order of $\gamma_s^1$.

**Step 5:** Define ranking vector $Q = [Q_1, Q_0]$.

It is easy to argue that states for which $x_s^{a,1}$ is positive should have priority over those for which $x_s^{a,1}$ is zero. Therefore, we prioritize states with $x_s^{a,1} > 0$ and sort them in descending order of their respective $\gamma_s^0$. Then, at the secondary level of priority, we take states with $x_s^{a,0} > 0$ and sort them in ascending order of their $\gamma_s^1$. Thereby, we have a priority list of all states, where in every time slot, the $M$ users that appear highest in the list are scheduled. As a simple example, consider the case where a network has three users. At a particular time slot, the users are in states $s_1$, $s_2$, and $s_3$, respectively. Also, suppose that the value of the reduced cost coefficients and long term performance measures in this time slot are as follows:

| state | $x_s^1$ | $\gamma_s^1$ | $x_s^0$ | $\gamma_s^0$ |
|-------|---------|-------------|--------|-------------|
| $s_1$ | 1       | 0           | 0      | 0           |
| $s_2$ | 0       | 0.2         | 10     | 0           |
| $s_3$ | 0.5     | 0           | 20     | 0           |

**TABLE II:** A sample instance of the QAA algorithm for three users. Note that the numbers are not derived from an actual experiments and serve only as an example.

By following Algorithm 1 for this sample case, we first prioritize user 1 and 3 over user 2 because their respective $x_s^1$ is non-zero. Among users 1 and 3, we pick user 3 because it has a lower $\gamma_s^0$. Therefore, the resulting ranking of users will become $3 - 1 - 2$.

In order to implement QAA, the base station needs to know the channel matrix, the QA of each user and the video characteristics such as number of layers and segment size.
of $\lambda$ in a homogeneous network, which we denote as $\lambda_{\text{avg}}$, is the average throughput of each user. We can calculate $\lambda_{\text{avg}}$ as $\frac{c_{\text{avg}}}{\rho}$, where $\rho$ is the load on the network as defined in previous sections and $c_{\text{avg}}$ is the average capacity of each subchannel. Also, $\mu_{\text{avg}}$ is defined as the average rate of draining the buffer, which depends on the average rate of the video segments being played back. In Appendix C, we derive an expression for $\mu_{\text{avg}}$ given the optimal variables derived from the RB problem.

Figure 6 illustrates the values for $\lambda_{\text{avg}}$ and $\mu_{\text{avg}}$ for different QA and network capacities as a function of load. The encircled points refer to the critical load $\rho^*$ for which the buffer level is constant on average. In order to determine the relation between sections and $c_{\text{avg}}$ played back, in Appendix C, we derive an expression for $\mu_{\text{avg}}$.

In Algorithm 2, we use an auxiliary variable $b_i$ as a measure to represent buffer fluctuations. By starting from an initial value $b_0$ and updating it at each time slot, we can quantify whether the buffer is draining (decreasing $b_i$) or filling (increasing $b_i$). As a scheduling rule, we first consider users with $b_i$ less than a pre-determined threshold $b_{\text{thresh}}$. Among these users, those with better channel are prioritized.

Figure 7 illustrates the scheduling priority heatmap with respect to the instantaneous channel state and the buffer occupancy of both layers. The darker the color, the higher the state appears in the priority list. Also, all users deploy DBP-10s and $c_{\text{avg}} = 4.5$ Mbps. From Figure 6, the critical load for this case is equal to $\rho^* = 2.3$. For a load value larger than 2.3, we observe in Figure 7a that the scheduling priority is highly channel dependent, with users with the highest channel capacity getting the highest priority. Figure 7b shows that for load values smaller than $\rho^*$, the policy begins to become buffer dependent prioritizing users that have less buffer occupancy. A similar trend is observed in Figures 7c and 7d where $c_{\text{avg}} = 3$ Mbps and therefore, $\rho^* = 1.58$. From Figure 8, we can conclude that the above observation is not limited to DBP-10s and applies to a great extent also to cases with DBP-20s and CBP, as shown in Figures 8a-8b and 8c-8d, respectively. Therefore, a scheduling mechanism that leverages this trend can be used for a variety of QA policies which the scheduler does not need to know in advance.

Due to the heterogeneity of wireless networks, users will face rising buffer levels at some times and draining buffer levels at others. From the above analysis we can conclude that these fluctuations in buffer level can be exploited to devise scheduling algorithms without the scheduler knowing the underlying system parameters. Such an algorithm should first provide a measure to quantify fluctuations in the buffer level of each user. This measure will then be used to perform buffer dependent scheduling when buffer is filling, and channel dependent scheduling, when the buffer is draining. Given these guidelines, we devise a simple QA Blind scheduling policy called Buffer Evolution Aware Scheduling (BEAS) shown in Algorithm 2.

In Algorithm 2, we use an auxiliary variable $b_i$ as a measure to represent buffer fluctuations. By starting from an initial value $b_0$ and updating it at each time slot, we can quantify whether the buffer is draining (decreasing $b_i$) or filling (increasing $b_i$). As a scheduling rule, we first consider users with $b_i$ less than a pre-determined threshold $b_{\text{thresh}}$. Among these users, those with better channel are prioritized. If any resources are left, we move to the rest of the users and schedule them by prioritizing users that have fewer base layer segments in the buffer. The update rule for $b_i$ is based on an exponential filter with a smoothing factor $\epsilon$. A larger $\epsilon$ reacts faster to buffer fluctuations while a smaller value results in a smooth representation for the buffer fluctuations. Also, $h(\cdot)$ is a function of the total number of sub-segments delivered in each time slot. In our simulations, we have determined by trial and error that a linear function in the form $h(x) = \alpha x + \beta$ results in the best performance. Algorithm 2 describes this heuristic. In Section VII, we discuss the practical implications of this algorithm in more detail.
Algorithm 2 Buffer Evolution Aware Scheduling

**Initialization:** Let $\epsilon > 0$, for $i \in N$, let $b^k_i = b_0$. The number of layer $l$ segments that are delivered to user $j$ in time slot $k$ is denoted by $n_{j,l}^k$.

for all time slots $k$ do
\[
b_{i}^{k+1} = (1 - \epsilon) b_{i}^{k} - \epsilon \tau_{\text{slot}}
\]

**SCHEDULE:**
\[
B = \{ i | b_{i}^{k} < b_{\text{thresh}} \}
\]
if $|B| < M$ then
Schedule $M$ users from $B$ with the best channel.
else
Schedule all users in $B$.
Schedule $M - |B|$ users from $N \setminus B$ with the lowest base layer occupancy.
end if

**UPDATE:**
for all scheduled users $j$ do
\[
b_{j}^{k+1} = (1 - \epsilon) b_{j}^{k} + \epsilon \tau_{\text{seg}} b_{i}^{k}(\sum_{l=1}^{L} n_{j,l}^k)
\]
end for

end for

VI. SIMULATION RESULTS

In this section, we perform an extensive simulation study evaluating the performance of the algorithms presented in the previous section and compare them with several baseline schemes. Unless mentioned otherwise, the simulation parameters are similar to Table I and the video length is 10 minutes. The QA schemes used in the simulations are designed similar to Section IV. For the BEAS algorithm we use $b_{\text{thresh}} = 0$.

We start by studying the effect of buffer limit on the system performance in Figure 9. It can be seen that increasing the buffer limit beyond 20s will not significantly improve the delivered video quality. Therefore, for the remainder of the simulations, we set the buffer limit to 20s in order to gain a suitable trade-off between computational complexity and average video quality.

**Fig. 7:** Scheduling priority comparison for users with DBP-10s. For the case with $c_{\text{avg}} = 4.5$ Mbps, the critical load $\rho^*$ is equal to 2.3 and for the case with $c_{\text{avg}} = 3$ Mbps, $\rho^* = 1.58$. The region beyond the dashed lines corresponds to buffer values that are not possible.

**Fig. 8:** Scheduling priority comparison for a network with $c_{\text{avg}} = 4.5$ Mbps. The figures on the left side correspond to DBP-20s and the ones on the right correspond to CBP. The region beyond the dashed lines corresponds to buffer values that are not possible.

**Fig. 9:** Reward gained by users for different buffer limit values. The QAs used for the experiment are CBP and DBP with the pre-fetch threshold being the buffer limit.

Next, we move on to comparing the QAA and BEAS algorithms with three baseline algorithms, namely Proportional Fairness (PF) [37], Best Channel First (BCF), and Lowest Buffer First (LBF) [38] algorithms. PF is a very popular scheduling scheme for wireless networks in which users are
sized based on their current channel state normalized by their long term average throughput. BCF is a purely channel dependent scheduling method that only takes the current link conditions of each user and schedules users with the best channel. LBF is a purely buffer dependent scheme in which users that have fewer base layers in the buffer are prioritized.

An important measure for comparison is to determine how each of these algorithms implement the quality-delay trade-off explained in Section III. Therefore, for each scenario under consideration, we show the average fraction of time that each user spends re-buffering, the average fraction of segments that are delivered with only the base layer, and the value of the sum reward per user, which combines both QoE measures into one. It should be noted that while for the re-buffering and reward plots, the x-axis represents the load on the network, the same axis for the video layer plots shows the number of subchannels.

A. Homogeneous System

We first consider the case of homogeneous users in Figures 10 and 11 for channels with $c_{avg} = 4.5$ Mbps and $c_{avg} = 2.55$ Mbps, respectively. By looking at Figures 10a and 11a we observe that in terms of reward, QAA performs very close to the optimum illustrated by the black line, especially for highly loaded networks. Furthermore, we observe that LBF performs better than PF and BCF in high capacity networks with low load while for heavily loaded or low capacity networks, the reverse occurs.

From Figures 10b and 11b we see that the two channel based schemes have poor delay performance. Also, in the low capacity network with high load, LBF also has poor delay performance which is due to the fact that by always scheduling the user with the smallest buffer, it might choose users with very poor channel conditions for which the download takes long. In other words, LBF has poor spectrum utilization which is detrimental in low capacity scenarios and high loads where resources are very scarce. These findings suggest that in order to provide satisfactory delay performance, algorithms should take both channel, and buffer state into account.

Providing enhanced delay performance comes at the cost of delivering segments with fewer higher layers in order to avoid re-buffering. Figures 10c and 11c show the average fraction of segments that were delivered with only the base layer. We can see from these figures that PF and BCF provide on average more segments with maximum quality than the other schemes. This result, together with the delay performance shows that these two schemes tend to over-serve some users, thereby being able to deliver more full quality segments, and under-serve the rest and cause large re-buffering. The QAA scheme adjusts the base layer only fraction with the load on the network, hence, when the load is large, fewer full quality segments are delivered and vice versa. LBF has a poor performance in these figures which is due to the fact that by preferring small buffer users without taking into account the channel conditions, barely any user can get beyond the initial base layer build up phase of DBP-20s. We can also see that

Fig. 10: 20 homogeneous users streaming a 10 minute video with DBP-20s, $c_{avg} = 4.5$ Mbps.

Fig. 11: 20 homogeneous users streaming a 10 minute video with DBP-20s, $c_{avg} = 2.55$ Mbps.
B. Heterogeneous System

Figures 12 and 13 show the performance of the scheduling schemes in non-homogeneous networks based on the discussion in Corollary 4.2. Figure 12 shows the QoE metrics of a network with 20 users, out of which 10 experience $c_{avg} = 4.5$ Mbps and the other 10 experience $c_{avg} = 2.55$ Mbps. Similarly, Figure 13 represents a network with $c_{avg} = 4.5$ Mbps and 20 users. Here, 10 users deploy a DBP-15s QA and the others use CBP. Similar to the homogenous cases, QAA and BEAS outperform the other algorithms in both sum reward and re-buffering. The general trend of the results is similar to the homogeneous case. In Figure 12 similar to 11 due to the presence of users in poor conditions, LBF degrades in performance as the load on the network increases. Also, in Figure 13c we observe that more users are able to deliver full quality segments, which is due to the fact that unlike DBP which starts with downloading only base layers regardless of the channel conditions, CBP is very aggressive in requesting enhancement layers when the channel is in good condition.

![Graphs showing performance metrics for heterogeneous networks](image)

Fig. 12: Heterogeneous network with DBP-20s. Ten users experience $c_{avg} = 4.5$ Mbps and the other ten have $c_{avg} = 2.55$ Mbps.

![Graphs showing performance metrics for heterogeneous networks](image)

Fig. 13: Heterogeneous network with $c_{avg} = 4.5$ Mbps. Ten users use DBP-15s and the other ten use CBP for QA.

do not need the buffer level as a scheduling measure since their goal is to increase network throughput. However, since the delay sensitivity of video is not taken into account in these scheduling schemes, they have poor re-buffering performance and hence, provide lower QoE.

On the other hand, since purely buffer based schemes conservatively try to only avoid re-buffering, they fail at delivering high video quality, especially when the load is high. We can therefore conclude that by using buffer or channel alone, no scheduling policy can deliver satisfactory QoE. BEAS combines the desirable features of channel dependent and buffer dependent scheduling policies into a simple algorithm. By keeping track of the evolution of the buffer state, we can implicitly infer both the capacity and the load of the network. Whenever the buffer level for a user starts to diminish or if the user cannot build up an adequate buffer occupancy, users are scheduled based on the channel state to quickly fill the buffer and prevent re-buffering. If the buffer level grows, since there is no urgency for utilizing the channel efficiently, the scheduler prioritizes users with the lowest segments in the buffer. This also explains why in low capacity networks, where the buffer level is generally low, the gap between BEAS and channel dependent policies narrows.

Another important conclusion from these results is that, especially in wireless networks, even well designed end-to-end QA schemes cannot guarantee QoE if the underlying scheduling at the base station is not designed properly. Also, for a fixed QoE objective, BEAS can deliver up to 30% more users on the same channel as compared to PF.

C. Discussion

From the results in Sections VI-A and VI-B we can draw several conclusions.

Since our QoE model rewards users based on their immediate playback output, it is desirable to always have non-zero segments in the buffer, preferably with as many layers as possible. Purely buffer based schemes (LBF) have an advantage in re-buffering due to the strict priority of users that are in higher risk of draining the buffer. However, for heavily loaded networks and lower data rates, the overall video quality drops because of poor spectrum usage. Therefore, the buffer level alone cannot be used as a reliable scheduling measure. On the other hand, purely channel dependent schemes (PF and BCF)

![Bar graphs showing re-buffering fraction](image)

![Bar graphs showing base-only segment fraction](image)

VII. TESTBED IMPLEMENTATION

In this section, we describe some practical implications of BEAS followed by a testbed implementation. For BEAS to
run in a practical network, the scheduler needs to know the channel quality of each user as well as the state of its buffer. If HTTP is used as application layer protocol, the base station is able to extract data related to the next subsegment to be transmitted from each HTTP request packet that the user sends to the content provider. Therefore, it can accurately estimate the buffer level of each user. However, since more content providers are using HTTPS, this information is encrypted and cannot be retrieved by any intermediate node in the network including the base station. Recently, efforts are being made for estimating the buffer level on the users by measuring TCP/IP metrics. For instance, in [39], a machine learning-based traffic classification method is presented that aims at solving this problem. However, in the absence of these estimation techniques, the buffer state has to be fed back to the base station, in a manner similar to the CQI, as suggested in [7, 26, 40].

We have implemented the scheduling algorithms on the sandbox network located in the orbit testbed. This experimental network consists of 9 nodes equipped with WiFi transceivers. The attenuation of the link between any two nodes can be manually altered from 0 to 63 dB. We use one of the nodes as a base station that contains all video segments and the other nodes act as streaming users. Then, we divide the 60 second long video into 1s long segments and encode them into a base layer and two enhancement layers using the JSVM encoder. We use temporal scalability where the frame rate of the temporal layers is 6, 12, and 24 frames per second. The QA deployed on all users is set to DBP-5s.

For our experiment, in order to generate a heterogeneous wireless channel with fluctuating link capacities, we randomly change the value of the attenuation for each node every five seconds. For half of the nodes, the attenuation value is chosen randomly from 6dB, 9dB and 12dB. For the other half, the possible values for attenuation are 9dB, 12dB, and 15dB. Whenever a segment is fully retrieved by a user, the base station polls all users for their channel state which respond by sending their instantaneous channel state. For the buffer, we simplify the implementation by assuming knowledge of the duration of the segments and the layer index of the transmitted segment at the base station. Therefore, the base station can calculate all users’ buffer state at any instance without the need of an explicit feedback.

Figure 14 shows the performance comparison between the studied algorithms. It can be seen that similar to the simulation results, LBF has better re-buffering performance than BCF and PF, while delivering fewer enhancement layer segments. PF and BCF suffer from higher re-buffering but are able to deliver more enhancement layer segments. The benefits of both schemes are combined into BEAS which has the lowest re-buffering and while it is not always able to deliver many enhancement layers, it outperforms the other schemes in terms of total reward.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we developed a framework for QA-adaptive SVC scheduling in wireless networks. We argue that instead of an overly complex and practically infeasible jointly optimized system, we should separate QA from the scheduler and adapt the scheduling policy to the underlying QA deployed on each user. Using the concept of RB, we formulate the problem as a linear program and solve it in order to obtain long term performance measures. We also formulate the jointly optimal problem as a MUSMDP in order to see the cost incurred by diverging from the jointly optimal scenario. We then develop a primal dual algorithm that performs scheduling in a QA-aware setting. By analyzing the outcome of this algorithm, we propose a heuristic scheduling algorithm that performs QA-blind scheduling with minimal complexity and signaling.

We also perform an extensive simulation study comparing the proposed scheduling algorithms with baseline schemes. Our results indicate that the optimal scheduler should have a joint buffer dependent and channel dependent behavior. By tracking the evolution of the buffer occupancy for each user, the scheduler should prioritize users that are draining the buffer and schedule them based on which user has a better channel quality. We also conclude that while QA schemes are designed to offer a good quality-delay trade-off, if the scheduler is not well designed, the end-to-end QA scheme cannot deliver high QoE in wireless networks. We finally evaluate the performance of the scheduling algorithm with a testbed implementation.

APPENDIX A

We first assume that there is an optimal allocation vector \( \mathbf{x}^* = (x_1^*, \cdots, x_N^*)^T \) (where \( x_k^* = (x_k^{1*}, x_k^{2*})^T \)), in which the above proposition does not hold. Our goal is to show that if we replace the allocation vector \( \mathbf{x}^* \) by the average allocation vector \( \mathbf{x} = \frac{1}{N} \sum_{i \in N} x_i^* \), the new allocation is feasible and does not decrease the value of the objective function. First, we define \( \bar{x} = \frac{1}{N} \sum_{i \in N} x_i^* \) as the average allocation vector for
the active cases. Since \( x_1^* (\forall i \in \mathcal{N}) \) is feasible and therefore non-negative, their average also satisfies the non-negativity constraint. Based on the definition of \( \bar{x}_1 = \frac{1}{\mu} \sum_{i \in \mathcal{N}} x_i^* \), satisfying the resource constraint (11) becomes trivial. In order to check the feasibility of the new allocation for the polytope constraint (10), we rewrite (8) as follows:

\[
x_0^0 - \beta \sum_{i \in \mathcal{S}} h_{is}^0 x_{ij}^0 + x_1^0 - \beta \sum_{i \in \mathcal{S}} h_{is}^1 x_{ij}^1 = \alpha_s, \quad s \in \mathcal{S}. \tag{31}
\]

Therefore, for the optimal point, the set of constraints in (10) can be represented as:

\[
Ax_s^* = A^0 x_0^s + A^1 x_1^s = \alpha \quad \forall i \in \mathcal{N}, \tag{32}
\]

where \( A^0 = (I - \beta H^0T) \) and \( A^1 = (I - \beta H^1T) \) (Note that the user index is omitted for all matrices since they are identical for all users). In order to prove the feasibility of \( \bar{x}_1 \), we need to show that the following holds:

\[
A^0 \bar{x}_0 + A^1 \bar{x}_1 = \alpha. \tag{33}
\]

If we plug in the values for \( \bar{x}_0 \) and \( \bar{x}_1 \), we have:

\[
\frac{1}{\mu} \sum_{i \in \mathcal{N}} (A^0 x_0^i + A^1 x_1^i) = \frac{1}{\mu} \times N \alpha = \alpha,
\]

and feasibility of \( \bar{x}_1 \) is concluded. Finally, we check if the value of the objective function changes if we replace \( x^* \) with \( \bar{x}_1 \). Since the immediate reward in each state is equal across the users, the optimal value of (29) can be written as \( R \cdot \sum_{i \in \mathcal{N}} (x_0^i + x_1^i) \), where \( R = (R_s)_{1 \leq \mathcal{S}} \). By replacing \( x_1^i \) (\( \forall i \in \mathcal{N} \)) with \( \bar{x}_1 \), it is easily verified that the objective value remains constant and the proof is complete.

**APPENDIX B**

The SMDP described by the objective function (19) can be turned into an equivalent MDP which can be solved using the Bellman equation described below for each user \( n \) (34):

\[
V_{s_n}^* = \max_{a_n} \left( r_{s_n}^{a_n} + \sum_{j_n \in \mathcal{S}} M(j_n|s_n, a_n) V_{j_n}^* \right), \tag{34}
\]

where \( j_n \) is the state of user \( n \) after action \( a_n \) is fully executed, and the values for \( r_{s_n}^{a_n} \) and \( M(j_n|s_n, a_n) \) are derived as follows:

\[
r_{s_n}^{a_n} = \mathbb{E}_s \left( \sum_{k=0}^t e^{-sk} R_{s_n}^k \right) \tag{35}
\]

\[
= \sum_{t=0}^\infty \left( \sum_{k=0}^t e^{-sk} R_{s_n}^k \right) f(t|s_n, a_n) \tag{36}
\]

\[
= \sum_{t=0}^\infty \left( \sum_{k=0}^t e^{-sk} R_{s_n}^k \right) f(t|s, a) \tag{37}
\]

\[
= \sum_{t=0}^\infty \sum_{c \in \mathcal{C}} \left( \sum_{k=0}^t e^{-sk} R_{s_n}^k f_{s_n, j_n}^a (t, c) \right) \tag{38}
\]

\[
M(j_n|s_n, a_n) = \sum_{k=0}^\infty e^{-sk} \mathbb{P}(k|j_n|s_n, a_n) = H_{a_n, n, j_n} \tag{39}
\]

From (39) we can see that the transition probabilities of the equivalent MDP are obtained by the \( H^a_n \) matrices derived in (17) and (18). Therefore, similar to the polytope constraint derived in (24), we can derive the equivalent polytope constraint (24).

For the resource constraint (23), we need to show that both sides of the equation represent the expected discounted number of occupied subchannels. The right hand side is defined similar to the resource constraint (15) and for the left hand side, we have:

\[
\mathbb{E} \left[ \sum_{n \in \mathcal{N}} \sum_{t=0}^\infty e^{-st} l_n^i (t) \sum_{k=0}^\infty e^{-sk} \right] = \sum_{n \in \mathcal{N}} \sum_{t=0}^\infty e^{-st} l_n^i (t) \sum_{k=0}^\infty e^{-sk} \tag{40}
\]

\[
\mathbb{E}_t \sum_{k=0}^\infty e^{-sk} = \sum_{t=0}^\infty \sum_{c \in \mathcal{C}} e^{-sk} f_{s_n}^a (t, c) \tag{41}
\]

By substituting (41) into (40) and calling it \( \tau_{s_n}^l \), we will get the resource constraint.

**APPENDIX C**

**CALCULATING THE AVERAGE PLAYBACK RATE \( \mu_{avg} \)**

Without loss of generality, we perform the derivation for the homogenous case. We can write \( \mu_{avg} \) as:

\[
\mu_{avg} = \sum_{l=0}^L \left( \sum_{i=0}^L q_i \right) \tau_l, \tag{42}
\]

where \( q_i \) are defined as in Section (11-B) with \( q_i = 0 \) to represent a playback rate of zero for re-buffering. Also, \( \tau_l \) is the fraction of total streaming time that segments with up to \( l \) layers are played back according to the RB solution, which can be calculated as follows:

\[
\tau_l = \frac{\sum_{s \in \mathcal{S}_l} (x_0^s + x_1^s)}{\sum_{s \in \mathcal{S}} (x_0^s + x_1^s)}, \tag{43}
\]

where \( x_0^s \) and \( x_1^s \) are the optimal solutions of the RB for state \( s \), and \( \mathcal{S}_l \) is the set of all states for which up to \( l \) layers are being played back: \( \mathcal{S}_l = \{ s \in \mathcal{S} | b_i > 0 \ \forall i = (1, \cdots, l) \} \).
APPENDIX D
We denote the set of states for which \( x_t^0 = x_t^1 = 0 \), as \( S' \subset S \). First, we prove that all states in \( S' \) are unreachable under \( u^* \). If for these states, we rewrite the constraints of \( \text{RBOPT} \) using (11), for the optimal points, we will have:

\[
-\beta \left( \sum_{i \in S} h_{i}^0 x_t^0 + \sum_{i \in S'} h_{i}^1 x_t^1 \right) = \alpha_s, \quad s \in S'.
\]

(41)

Let us first assume that \( s \) is not an initial state (\( \alpha_s = 0, \forall s \in S' \)). Since all variables on the left hand side are non-negative, (11) can only hold if \( h_{i}^0 x_t^0 = 0 \) and \( h_{i}^1 x_t^1 = 0, \forall i \in S' \), which, according to the above definition, means that state \( s \) cannot be reached from any state in \( S \setminus S' \). On the other hand, if \( s \) is an initial state (\( \alpha_s > 0 \)), then \( s \notin S' \), otherwise the two sides of (11) cannot be equal. We conclude that the trajectory will start from an initial state that is a member of \( S \setminus S' \) and that from any state belonging to this set, the optimal policy does not allow a transition from \( S \setminus S' \) to \( S \). Now, if state \( s \) is unreachable, we have \( h_{i}^0 x_t^0 = h_{i}^1 x_t^1 = 0, \forall i \in S \) and also \( \alpha_s = 0 \). Therefore, it is easily verified that in order to have feasibility, \( x_t^0 = x_t^1 = 0 \) and the proof is complete.

REFERENCES
[1] Sandvine, “Sandvine global Internet phenomena: Latin America and North America,” Sandvine, Intelligent BoardNetworks, Tech. Rep., June 2016.
[2] Y. Sánchez de la Fuente, T. Schierl, C. Hellge, T. Wiegand, D. Hong, and V. Champel, and O. Oyman, “Applications and deployments of server and network assisted DASH (SAND),” 2016.
[3] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begun, M.-L. Champel, and O. Oyman, “Applications and deployments of server and network assisted DASH (SAND),” 2016.
[4] P. Whittle, “Restless bandits: Activity allocation in a changing world,” Journal of Applied Probability, pp. 287–298, 1988.
[5] J. C. Gittins, “Bandit processes and dynamic allocation indices,” Journal of the Royal Statistical Society. Series B, pp. 148–177, 1979.
[6] S. A. Hosseini, F. Fund, and S. S. Panwar, “(Not) yet another policy for video coding,” in Proceedings of the 7th ACM International Conference on Multimedia systems, 2011, pp. 257–264.
[7] “Vidyo and Google collaborate to enhance video quality within WebRTC,” Press Release, August 2013, http://www.vidyo.com/company/news-and-events/press-releases/vidyo-and-google-collaborate-to-enhance-video-quality-within-webrtc/.
[8] 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 ACM Multimedia Systems Conference, 2012, pp. 149–154.
[9] S. Xiang, L. Cai, and J. Pan, “Adaptive scalable video streaming in wireless networks,” in Proceedings of the 3rd ACM Multimedia Systems Conference, 2012, pp. 167–172.
[10] J. Owani, A. Agarwal, and A. K. Jagannatham, “Optimal scalable video scheduling policies for real-time single-and multiuser wireless video networks,” IEEE Transactions on Vehicular Technology, vol. 64, no. 6, pp. 2424–2435, 2015.
[11] T. Kim and M. H. Ammar, “Optimal quality adaptation for scalable encoded video,” Selected Areas in Communications, IEEE Journal on, vol. 23, no. 2, pp. 344–356, 2005.
[12] B. Kleinberg, J. Kofer, and H. Hellwagner, “An evaluation of TCP-based rate-control algorithms for adaptive internet streaming of H.264/SVC,” in Proceedings of the first annual ACM SIGMM conference on Multimedia systems. ACM, 2010, pp. 157–168.
[13] C. Sieber, T. Hohfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, “Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC,” in Integrated Network Management (IM 2013), IFIP/IEEE International Symposium on, 2013, pp. 1318–1323.
[35] K. G. Murty, *Linear programming*. John Wiley & Sons, 1983.

[36] F. Fund, C. Wang, Y. Liu, T. Korakis, M. Zink, and S. S. Panwar, “Performance of dash and webrtc video services for mobile users,” in *Packet Video Workshop (PV), 2013 20th International*. IEEE, 2013, pp. 1–8.

[37] F. P. Kelly, A. K. Maulloo, and D. K. Tan, “Rate control for communication networks: shadow prices, proportional fairness and stability,” *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[38] Y.-H. Yeh, Y.-C. Lai, Y.-H. Chen, and C.-N. Lai, “A bandwidth allocation algorithm with channel quality and QoS aware for IEEE 802.16 base stations,” *International Journal of Communication Systems*, vol. 27, no. 10, pp. 1601–1615, 2014.

[39] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, “Buffest: Predicting buffer conditions and real-time requirements of http (s) adaptive streaming clients,” in *Proceedings of the 8th ACM Multimedia Systems Conference*, 2017.

[40] S. A. Hosseini, Z. Lu, G. de Veciana, and S. S. Panwar, “SVC-based multi-user streamloading for wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2185–2197, 2016.

[41] “Orbit lab tutorial,” [http://www.orbit-lab.org/wiki/Tutorials](http://www.orbit-lab.org/wiki/Tutorials) accessed: 2017-06-22.