Online Learning for Wireless Video Transmission with Limited Information

Yu Zhang, Fangwen Fu, Mihaela van der Schaar
Electrical Engineering Department, UCLA
yuzhang@ucla.edu, {fwfu, mihaela}@ee.ucla.edu

***Abstract*— In this paper, we address the problem of joint packet scheduling at the application layer as well as power and rate allocation at the physical layer for delay-sensitive video streaming over slow-varying flat-fading wireless channels. Our goal is to find the optimal cross-layer policy that maximizes the cumulative received video quality, while minimizing the total transmission energy. We first formulate the cross-layer optimization using a systematic layered Markov Decision Process (MDP) framework and then propose a layered real-time dynamic programming (RTDP) algorithm for solving this cross-layer optimization problem by combining together the policy update and real-time decision making. This approach reduces the high complexity of the conventionally used offline dynamic programming methods. Moreover, to accommodate the cases when the network environment dynamics (e.g. state transition probabilities) are unknown or non-stationary (e.g. state transition probabilities are changed over time), we further improve our RTDP method by collecting the required network information and estimating the dynamics online, using a model-free approach. Based on this information, a user (a transmitter-receiver pair) can adaptively change its policy to cope in real-time with the experienced environment dynamics. We also prove the convergence of this RTDP method (which complies with the layered architecture of the OSI stack). Finally, our numerical experiments show that the proposed RTDP solutions outperform the conventional offline DP methods for real-time video streaming.**

Keywords- layered markov decision process, dynamic programming, real-time learning, online learning, wireless video transmission.

I. INTRODUCTION

Video streaming over best-effort, packet-switched networks is challenging due to a number of factors, including the high bit-rates and dynamic characteristics exhibited by the video traffic, the time-varying channel characteristics, and the hard delay constraints that need to be fulfilled for real-time applications [1]. This paper addresses the problem of streaming packetized video data over a wireless channel from the transmitter to the receiver. In such a streaming media system, the transmitter encodes the incoming video frames, stores the encoded data in its output buffer and, subsequently, transmits them on demand to a client for playback in real-time [2]. Since wireless channels do not provide any quality of service (QoS) guarantees, the transmitter must perform adaptive packet scheduling as well as transmission power and rate allocation, in response to the environment dynamics experienced by the transmitter or receiver. In this paper, we focus on the problem of real-time video transmission over a single-hop wireless network, analyzing the single user case with a transmitter-receiver pair. Specifically, we consider how, depending on the environment, we can optimally trade off the video quality and transmission energy for real-time wireless video transmission, by adapting the packet scheduling strategy at the application layer and the power and rate allocation strategy at the physical layer. This problem is often treated as a cross-layer optimization problem. Although cross-layer optimization solutions [3][4][5][6] have been extensively investigated in recent years to improve the performance of the wireless user operating in a time-varying, error-prone wireless environment, the current solutions often optimize the current utility without considering the impact on the future performance (i.e. myopic decision [7]) and also, do not comply with the existing layered network architecture.

In [7], a layered framework based on MDP has been established to solve such cross-layer problems by considering the impact of the current selection of cross-layer transmission strategies on the future performance. In this framework, if the states on every layer are defined to capture the environmental dynamics at each time slot and the actions at each layer are also defined in order to determine the received reward and the state transition across time slots, then each layer can use layered dynamic programming (DP) to autonomously determine its optimal action by exchanging only limited information with other layers.

However, the layered DP solutions iteratively search the optimal policy offline, which require a full scan over the entire system state space during each iteration and are, thus, expensive in terms of the required computation resources and overhead. In video applications, the heterogeneity of video data usually leads to a large state space, thus inhibiting the implementation of such offline layered DP methods, even if their resulting solution is optimal. Moreover, and even more importantly, the dynamic wireless network environment, such as the time-varying traffic characteristic and channel conditions, as well as the repeated interaction among multiple users are often difficult to characterize a priori. In this case, the offline DP methods usually perform poorly, due to their inability to adapt to the environmental changes.

To solve these problems, we proposed an alternative approach referred to as RTDP [8] which, instead of updating the state-value function for all the states during one iteration, selectively updates the state-value function for only a subset of states instead of the entire state space. To adaptively adjust the policy to the change in the environment dynamics, we further develop a model-free approach for RTDP, which is called Adaptive RTDP (ARTDP), to work in environments where the a priori environment knowledge is limited or inexistent.

Summarizing, the proposed RTDP method has three main advantages over the conventional offline DP:

1. Unlike the offline DP methods, RTDP is applied online intertwined with the real-time decision making. Therefore, in the actual implementations of the cross-layer optimization, we do not have to wait for a satisfactory policy before the real-time control process must begin.
2. Because RTDP is able to select a subset of states to which the state-value function and policy updates are applied, we can smartly select the states which need to be improved most, based on the experience collected during the decision making process, in order to improve the efficiency of the policy optimization.
3. ARTDP can work efficiently without the network environment knowledge, and can adaptively change its policy based on the online experienced dynamics.

The paper is organized as follows. Section II introduces the considered wireless system. Then, in Section III, we formulate the considered cross-layer problem into a formal MDP problem. Section IV discusses the proposed on-line RTDP method and the online learning under the layered MDP framework. Section V presents the simulation results. Section VI concludes the paper.

II. CONSIDERED SYSTEM MODEL

This paper considers real-time video transmission over a single-hop slow-varying flat fading channel. We focus on the transmission strategy adaptation at the *application (APP)* layer, *MAC* layer, and *physical (PHY)* layer, as depicted in **Error! Reference source not found.** (top). The aim of our work is to optimize the received video quality under certain transmission energy constraints. This cross-layer optimization problem can be modelled as an MDP defined over a joint space of states, transition probabilities, action sets, and reward function over all layers, maximizing the long-term utility of the user (e.g. discounted accumulative reward as in [7]). The considered system model can be formulated as in Figure 1 (bottom).

A. APP layer model

At the APP layer, we assume that the wireless user deploys a delay-sensitive video streaming application, which deploys a video encoder (e.g. H.264) operating at a frame rate f_s. The encoded packets are then injected into the output buffer, and are ready for transmission. We use the activity adaptive model proposed in [9] to model the incoming variable bit rate (VBR) video traffic: after identifying the I frames, the sequence is encoded with a repeating pattern of two B frames followed by a P frame, until the next I frame is reached, but if the interval between two I frames is not a multiple of three, this pattern cannot always be inserted, in which case the last pattern is terminated when the I frame is reached. Here, to simplify our analysis, we neglect the different activity levels of the frames [9], but these can be easily included in the proposed solution and our analysis.

In [6] and [10], it has been shown that partitioning the packets into different priority classes and adjusting the transmission strategies correspondingly for each class can significantly improve the overall received video quality and provide graceful degradation as network congestion levels and channel conditions are changing. Thus, we also divide the packets of the encoded video stream into several priority classes based on their delay constraints and visual impacts on the overall video quality. Assuming that three types of frames exist in this application, which are the I, B, and P frames, respectively, such video traces can then be modelled as a stochastic Markov process, as shown in [9].

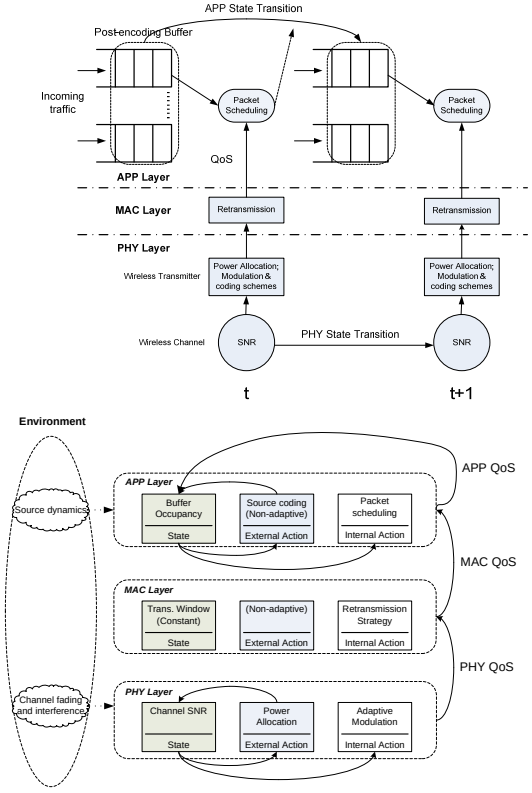

Figure 1. (top) The Architecture of Video Streaming System; (bottom) The corresponding layered architecture

For simplicity, we assume that each frame type corresponds to one priority class (i.e. I, B, P classes), and that inside each class, all packet characteristics are the same and hence, they have the same delay deadline and distortion impact. The average packet length is denoted as L in bits, and a frame contains G_I, G_B, G_P packets on average for class I, B, P, respectively. Meanwhile, the delay deadline for each type of frame is denoted as $D_I T_{frame}$, $D_B T_{frame}$, $D_P T_{frame}$ respectively, where D_I, D_B, and D_P are all integer constants, and T_{frame} is the length of one frame interval. It is easy to tell that the lifetimes of different frames have the relationship as $D_B=D_I-1$ and $D_P=D_I+2$. Moreover, each packet at classes I, B, and P has the distortion impact Q_I, Q_B, and Q_P, respectively. The dependencies[1] between different classes are considered through the distortion impact of the different classes. Since both B and P classes depend on the I class and the B class further depends on the P class, we have the following inequality: $Q_I>Q_B>Q_P$.

[1] The exact dependency between different classes can be expressed as direct acyclic graph (DAG) as in [17]. In this paper, we made the simplification on this dependency for the analysis.

B. MAC layer model

At the MAC layer, the transmission is time slotted. Because the transmission of an APP layer frame can be divided into several transmissions of packet at the MAC layer, we assume that T_{frame} is much larger than the length of MAC time slot, denoted as ΔT in the rest of this paper. To keep our analysis simple due to the space limitations, we do not explicitly consider of the adaptation of the transmission strategies at the MAC layer, but assume that the transmission actions at the MAC layer are all fixed (e.g. each user has a fixed window to access the spectrum in each time slot, and a fixed number of retransmission when performing ARQ). The channel access can be based on time division multiple access (TDMA) or code division multiple access (CDMA). Therefore, the QoS provided by the MAC layer has a deterministic mapping to the QoS from the PHY layer, and we just use the PHY's QoS as variables in our cross-layer optimization for simplicity.

C. PHY layer model

The PHY layer then grabs packets from the output buffer and transmits them through a slow-varying flat-fading wireless channel. The wireless channel in this problem is modelled as a discrete-time block-fading additive white Gaussian noise channel [11], where continuous time is partitioned into discrete time slots, and the Signal-to-Noise Ratio (SNR) is assumed to be constant in each time slot. Here, we assume the length of channel coherence time T_{coh} is equal to the MAC time slot ΔT [3][4] and hence, the Signal-to-Noise Ratio is assumed to be constant within each time slot.

The scheduling in the video streaming consists of two parts then. In each time slot i, the APP layer decides which set of packets in the output buffer to be transmitted, and the PHY layer selects the optimal transmission rate v and the optimal power allocation q to transmit these packets, so as to minimizing the average transmission energy consumption.

III. MDP FORMULATION

The goal in our problem is to design an optimal joint scheduling mechanism over the video encoder at APP layer and the wireless transmitter at PHY layer, with the aim of optimizing the received video quality while minimizing the transmission energy. This can be modelled as a MDP maximizing some discounted accumulative reward as in [7]. Based on the cross-layer model in [7], the MDP problem can be defined over a joint space of states, transition probabilities, action sets, and reward function over all layers. However, note that our problem is significantly different than the cross-layer problem considered in [7] and hence, we first proceed by formalizing the considered joint PHY-APP optimization as an MDP problem.

A. APP layer

1) APP state

The state at the APP layer is defined such that it can capture APP's currently experienced dynamics. In our problem, the APP's state is composed of two parts, the incoming traffic pattern $\xi \in \mathcal{X}$ in each time slot and the post-encoding buffer occupancy $\kappa \in \mathcal{K}$ at the beginning of every time slot: $s_{APP} = [\xi, \kappa] \in S_{APP} = \mathcal{X} \times \mathcal{K}$.

The packet-level time slot in our problem is assumed to be the frame interval, and accordingly, there is only one incoming packet in each time slot. We can use the activity adaptive model in [9] to model the incoming traffic as a Finite-State Markov Chain (FSMC) by defining the state space $\mathcal{X} = \{I, P, B, BB\}$, $\mathcal{X}$ can be augmented easily to also include more sequence type in [9] if needed.

To accurately and uniquely capture the buffer state, we need to find the accurate expression for the state in every priority class. Neglecting the transmission time and decoding time, we assume that the queuing time of the packet in the transmitter's output buffer equals to the time it experiences between encoding and decoding. Thus, any packet that has stayed in the output buffer for its lifetime and is not been successfully transmitted during this duration is assumed to be expired and will be discarded from the buffer. As described in Section II.A, there are three different packet lifetimes corresponding to different packet priority classes, and the packets in each class have different expiration deadlines, thus b can be represented by the current numbers of I-type, P-type and B-type packets in every deadline respectively.

$$\kappa = \left[\tilde{\kappa}_1, \tilde{\kappa}_2, \cdots, \tilde{\kappa}_{D_{\max}}\right], \tag{1}$$

where D_{max}=max{D_I, D_B, D_P} is the largest possible packet lifetime. $\tilde{\kappa}_d = \left(\kappa_{d,I}, \kappa_{d,B}, \kappa_{d,P}\right)$ represents the number of packets from each frame type and having a remaining life time of d time slots. Here we assume an output buffer which is large enough to hold all the incoming traffic.

2) APP action

The actions can be classified into two types at each layer [7]: an *external action* is performed to determine what the next state should be such that the future reward will be improved, and an *internal action* is performed to determine the service (QoS) provided to the upper layers for video streaming in the current time slot.

As the APP layer is the highest layer in our system, it has no internal action to provide service to an upper layer, but only external actions which need to determine the set of packets to be transmitted in every time slot. Accordingly, we have

$$a_{APP} = \left[\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_{D_{\max}}\right], \tag{2}$$

where $\tilde{a}_d = \left(a_{dI}, a_{dB}, a_{dP}\right)$ are the numbers of packets the transmitter takes from the subset of packets with a remaining life time of d time slots.

3) State transition probability

The update on buffer occupancy is an MDP, and the state update will be influenced by the current APP action.

$$\begin{bmatrix} \tilde{\kappa}_1^{t+1} \\ \vdots \\ \tilde{\kappa}_{D_B}^{t+1} \\ \tilde{\kappa}_{D_I}^{t+1} \\ \tilde{\kappa}_{D_P}^{t+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \tilde{\kappa}_2^t - \tilde{a}_2^t \\ \vdots \\ \tilde{\kappa}_{D_B+1}^t - \tilde{a}_{D_B+1}^t + G_B^t \\ \tilde{\kappa}_{D_I+1}^t - \tilde{a}_{D_I+1}^t + G_I^t \\ \tilde{\kappa}_{D_P+1}^t - \tilde{a}_{D_P+1}^t + G_P^t \\ \vdots \end{bmatrix}, \tag{3}$$

where G^t denotes the number of incoming packets in the previous time slot. The buffer state transition probability is then computed as

$$p_\kappa\left(\kappa^{t+1} \mid \kappa^t, \xi^t, a_{APP}^t\right) = \begin{cases} p\left(G^t \mid \xi^t\right) & \text{Eq. (3) satisfied} \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

The incoming traffic is a FSMC, and thus, its transmission probability $\left\{p_\xi\left(\xi^{t+1} \mid \xi^t\right)\right\}$ is time-invariant and not influenced by the action executed at APP layer.

The state transition probability at APP layer is $p\left(s_{APP}^{t+1} \mid s_{APP}^t, a_{APP}^t\right) = p_\xi\left(\xi^{t+1} \mid \xi^t\right) p_\kappa\left(\kappa^{t+1} \mid \kappa^t, \xi^t, a_{APP}^t\right)$.

4) APP reward

The current reward at APP layer is defined as the received application quality for the delay-sensitive application in this time slot. It is composed of two parts as $r\left(s_{APP}, a_{APP}, Z_{APP}\right) = r_{rec} - r_{loss}$, where r_{rec} is the total video quality which is contributed by the packets successfully transmitted and is decodable on the receiver's side in this time slot; and r_{loss} is the total video quality which is contained in the packets dropped in this time slot which expire due to the deadline. Thus,

$$r_{rec} = \sum_{l=1}^{D_{\max}} \sum_h a_{l,h} Q_h, r_{loss} = \sum_h \kappa_{1,h} Q_h - \sum_h a_{1,h} Q_h . \quad (5)$$

5) Action space reduction for packet scheduling

Based on the problem formulation in Section II.A, it is easy to find out that the packet scheduling in every time slot has to search from an action space with the size of $N_{\boldsymbol{a}} = \prod_h \left(\overline{G}_h + 1\right)^{D_h}$, where $\overline{G}_h$ represents the average number of incoming packets when frame of priority class h arrives. Fortunately, with the following two remarks, the action space can be reduced exponentially.

Remark 1: There is no benefit of transmitting packets with longer expiration deadline instead of packets from the same priority class, but with shorter expiration deadline. This is obvious, as transmitting packets who will expire soon can obtain higher foresighted application quality at the same cost.

Remark 2: Using K to denote the number of effective QoS provided by PHY layer in its QoS frontier [7], and $\left\{\tilde{v}_k, 1 \le k \le K\right\}$ represents the transmission rate supported by these QoS. Then the number of packets transmitted in this time slot should be larger than the smallest supported rate.

Remark 2 is explained as follows: Without generality, assume $\tilde{v}_1$ as an example of the smallest supported rate, and the number of packets selected from each priority class are denoted as $\tilde{a}_I, \tilde{a}_B, \tilde{a}_P$ respectively. Then if the number of remaining packets is larger than $\tilde{v}_1$, we have to transmit at the minimum cost of $\tilde{v}_1$. It is straightforward to at least have $\tilde{a}_I + \tilde{a}_B + \tilde{a}_P \ge \tilde{v}_1$

Therefore, the above-mentioned action space size for packet scheduling can be reduced based on the above two remarks.

Proposition 1: The MDP problem with the action space $\boldsymbol{\mathcal{A}}_{APP}$ containing $\prod_h \left(\overline{G}_h + 1\right)^{D_h}$ has the same policy as the MDP problem with the reduced action set $\tilde{\boldsymbol{\mathcal{A}}}_{APP}$ containing $\prod_h \left(\overline{G}_h D_h + 1\right) - O\left(\tilde{v}_1\right)$ actions, where $O\left(\tilde{v}_1\right)$ is the volume of the region $\sum_{h=1}^{H} \tilde{a}_h < \tilde{v}_1$.

Proof: Based on Remark 1, the size of $\tilde{\boldsymbol{\mathcal{A}}}_{APP}$ should be no larger than $\prod_h \left(\overline{G}_h D_h + 1\right)$. Then with the constraints from Remark 2 that the sum rate of all priority classes should be larger than $\tilde{v}_1$, then those actions inside the region of $\tilde{a}_I + \tilde{a}_B + \tilde{a}_P < \tilde{v}_1$ are also discarded. ■

B. PHY layer

1) PHY state

The dynamic we use to characterize the state at PHY layer is the possible received channel SNR in each time slot, denoted by $s_{PHY} = \gamma \in \Gamma = S_{PHY}$. The received signal envelope has the Rayleigh distribution in a typical multipath propagation environment. With additive Gaussian noise, the received instantaneous SNR γ is distributed exponentially with the following probability density function [11]

$$p(\gamma) = \frac{1}{\overline{\gamma}} \exp\left(-\frac{\gamma}{\overline{\gamma}}\right), \quad (6)$$

where $\overline{\gamma}$ is the average SNR which is determined by the allocated transmission power q. Let $\vec{\Gamma} = \left[\Gamma_1, \Gamma_2, \cdots, \Gamma_{N+1}\right]'$ be the received SNR thresholds in increasing order with $\Gamma_1 = 0$ and $\Gamma_{N+1} = \infty$ which partition the total SNR space into N intervals. The channel is determined to be in state n if the SNR is between Γ_n and Γ_{n+1}. The states are thus ordered with decreasing average Bit-Error-Rate (BER) values.

2) PHY action

At each state, the wireless user is able to adapt its modulation scheme to determine the QoS levels to support the upper layer. The selection of modulation level m in each time slot is then the internal action at PHY layer, denoted as $b_{PHY} = m$. To simplify the analysis, here we assume that MPSK is used, and the number of packets transmitted in each time slot is equal to the logarithm of modulation level, as $v = \log_2(m)$ and hence, the selection of m is the rate allocation we would like to discuss in this paper. However, this analysis can also be performed for more sophisticated MAC models.

The external action at PHY layer is the power allocation $a_{PHY} = q$ which could determine the received SNR as the channel state in the next time slot. The power allocation determines the transition of channel SNR, and the rate allocation determines the BER during transmission. Given the current channel SNR γ and transmission rate v, the BER can be determined as [12]

$$\varepsilon_b = 0.05 \times \exp\left[\frac{-6\gamma}{\left(2^{1.9v}-1\right)}\right]. \quad (7)$$

The packet loss rate, as a function of ε_b, depends on the specific packet error-correcting scheme being implemented. In this paper, we suppose that a packet is in error if at least l out of its totally L bits are in error. Then we can characterize packet error rate in terms of BER as

$$\varepsilon_P = \sum_{i=l}^{L}\left(\binom{L}{i}\left(\varepsilon_b\right)^i\left(1-\varepsilon_b\right)^{(L-i)}\right). \quad (8)$$

Thus, we can define the effective data rate in each time slot as $\tilde{v} = v \times \varepsilon_p$, which represents the actual number of packets can be transmitted (with all the retransmission and error correction considered already).

3) State transition probability

As said in Section II.C, we assume that a one-step transition in the FSMC model corresponds to the channel state transition after one frame interval ΔT. The transition only happens from a given state to its two adjacent states. The transition probability $P_{n,n+1}$ from state n to $n+1$, can be approximated by the ratio of the level crossing rate at threshold Γ_{n+1} and the average number of packets per second staying in state n. Similarly, the transition probability $P_{n,n-1}$ is approximated by the ratio of the level crossing rate at threshold Γ_n and the average number of packets per second staying in state n. Hence, they can be approximated as in [11],

$$p_\gamma\left(s_{PHY}^{t+1} \mid s_{PHY}^t, a_{PHY}^t\right) = \begin{cases} \dfrac{F\left(\Gamma_{j+1}\right)T_p}{\phi_j}, & s_{PHY}^t = \widehat{\Gamma}_j,\ s_{PHY}^{t+1} = \widehat{\Gamma}_{j+1} \\ \dfrac{F\left(\Gamma_{j-1}\right)T_p}{\phi_j}, & s_{PHY}^t = \widehat{\Gamma}_j,\ s_{PHY}^{t+1} = \widehat{\Gamma}_{j-1} \\ 1 - \dfrac{F\left(\Gamma_{j-1}\right)T_p}{\phi_j} - \dfrac{F\left(\Gamma_{j+1}\right)T_p}{\phi_j}, & o.w. \end{cases}, \quad (9)$$

where T_p is the transmission time for one packet; $N(\Gamma)$ is the level crossing rate of level Γ for the SNR process, and is expressed as

$$F(\Gamma) = \sqrt{\frac{2\pi\Gamma}{\overline{\gamma}\left(a_{PHY}\right)}} f_m \exp\left(-\frac{\Gamma}{\overline{\gamma}\left(a_{PHY}\right)}\right), \quad (10)$$

where f_m is the maximum Doppler frequency.

The steady-state probability of each state is

$$\phi_j = \int_{\Gamma_j}^{\Gamma_{j+1}} p(\gamma)\, d\gamma = \exp\left(-\frac{\Gamma_j}{\overline{\gamma}\left(a_{PHY}\right)}\right) - \exp\left(-\frac{\Gamma_{j+1}}{\overline{\gamma}\left(a_{PHY}\right)}\right). \quad (11)$$

4) PHY reward

The reward at the PHY layer is the negative of the total energy consumed. The modulation energy cost is defined as the energy-rate function [13]: $c_{PHY\_in} = N_0\left(2^{2v}-1\right)/\gamma$, where N_0 denotes thermal noise. The transmission energy cost is defined as: $c_{PHY\_ex} = q\Delta T$. We have $c_{PHY} = c_{PHY\_ex} + c_{PHY\_in}$.

C. Utility function

By combining the application quality at the APP layer and the incurred transmission energy at the PHY layer, the total reward in an individual time slot t can be defined as

$$R^t\left(\boldsymbol{s}^t, \boldsymbol{a}^t, Z_{APP}^t\right) = r^t\left(s_{APP}^t, a_{APP}^t, Z_{APP}^t\right) - \lambda_{PHY} c_{PHY}^t\left(b_{PHY}^t\right), \quad (12)$$

where λ_{PHY} is a positive parameter which trade off between the application quality and operation cost.

The reward in Eq. (12) can also be further decomposed into two parts: one is the internal reward $R_{in}^t\left(\boldsymbol{s}^t, Z_{APP}^t\right) = r^t - \lambda_{PHY} c_{PHY\_in}^t\left(\xi_{PHY\_in}^k\right)$; and the external reward $R_{ex}^t\left(\boldsymbol{s}^t, \boldsymbol{a}^t\right) = -\lambda_{PHY} c_{PHY\_ex}^t\left(a_{PHY}^t\right)$, which includes the transmission energy incurred due to the external actions.

We assume that the state transitions at different layers are synchronized such that in each time slot, the wireless user has constant state and performs static actions. Thus, with the formulation of MDP, we try to optimize the foresighted cross-layer decision [7], finding the optimal policy π by maximizing the cumulative reward over the infinite time horizon, which means the wireless user is able to take the impact of the current actions on the future reward into consideration, as

$$V = \mathbb{E}_\pi\left\{\sum_{t=0}^{\infty} \mu^t R^t\left(\boldsymbol{s}^t, \boldsymbol{a}^t, Z_{APP}^t\right)\right\}, \quad (13)$$

where μ is a discounted rate within $(0,1)$. When $\mu = 0$, the problem turns into the optimal myopic decision problem, where the video user only considers maximizing the reward received in the current time slot, thus we do not consider the $\mu = 0$ case. The reasons for us not considering $\mu = 1$ are as follows: (i) for delay-sensitive video applications, the data needs to be sent out as soon as possible to avoid missing delay deadlines, and (ii) the undiscounted sum of rewards is not guaranteed to be finite and to converge [16]. In DP algorithms, V is also called state value function or value function.

IV. REAL-TIME DP SOLUTION

A. Layered offline DP solution

In [7], a layered dynamic programming solution has been proposed for the MDP problem formulated in Section III. It takes the idea of value iteration [16], and uses a layered DP operator to allow each layer to optimize its own policy autonomously, based on the information exchanged with the other layers. Each layer l first selects its own internal actions, which, combined with the QoS provided by the lower layers, determines the QoS Z_l supported to the upper layer. Then, the DP-based optimization starts from the highest layer in a sequential fashion. Each layer optimizes its own external and internal actions, and then passes the value function downwards to the lower layers for their optimization. The resulting DP solution is summarized in Algorithm 1 and the corresponding DP operator at each layer is shown in Table 1, where

$$\tilde{V}\left(s_{PHY}^{'}, s_{APP}, a_{APP}, Z_{APP}\right)$$
$$= R_{in}\left(s_{APP}, Z_{APP}\right) \quad , \quad (14)$$
$$+\mu \sum_{s_{APP}^{'} \in \mathcal{S}_{APP}} p\left(s_{APP}^{'} \mid s_{APP}, a_{APP}, Z_{APP}\right) V\left(s_{PHY}^{'}, s_{APP}^{'}\right)$$
$$\tilde{V}_{PHY}\left(s_{PHY}, a_{PHY}\right)$$
$$= -\lambda_{PHY} R_{ex}\left(s_{PHY}, a_{PHY}\right) \quad . \quad (15)$$
$$+ \sum_{s_{PHY}^{'} \in \mathcal{S}_{PHY}} p\left(s_{PHY}^{'} \mid s_{PHY}, a_{PHY}\right) V_{PHY}\left(s_{PHY}^{'}\right)$$

The value function V in the DP algorithm is the expectation of the foresighted utility function, as in Eq. (13) and R is the myopic utility function at that instant. The state space $\mathcal{S}$ then, is the composite space comprising of the incoming traffic state space $\mathcal{X}$, the buffer occupancy $\mathcal{K}$ at APP layer, and the channel state space Γ at PHY layer, i.e., $\mathcal{S} = \mathcal{X} \times \mathcal{K} \times \Gamma$, where $\times$ denotes the Cartesian product. The state transition from $\boldsymbol{s}$ to $\boldsymbol{s}^{'}$ when action $\boldsymbol{\alpha}$ is taken is given by

$$p\left(\boldsymbol{s}^{'} \mid \boldsymbol{s}, \boldsymbol{\alpha}\right) = p_{\xi}\left(\xi^{'} \mid \xi\right) p_{\kappa}\left(\kappa^{'} \mid \kappa, \xi, a_{APP}\right) p_{\gamma}\left(\gamma^{'} \mid \gamma, a_{PHY}\right). \quad (16)$$

Algorithm 1: Algorithm for layered DP solution

Initialize V arbitrarily, e.g. $V(\boldsymbol{s}) = 0$ for all $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$
$\Delta \leftarrow 0$
Repeat
For each $\boldsymbol{s} \in \boldsymbol{S}$
  $v \leftarrow V(\boldsymbol{s})$
  Perform layered DP
  $\Delta \leftarrow \max\left(\Delta, |v - V(\boldsymbol{s})|\right)$
Until $\Delta < \theta$ (*a small positive number*)
Output the optimal deterministic policy $\pi = \pi(\boldsymbol{\mathcal{S}})$

Table 1: DP operator at each layer

	DP operator at each layer
APP	$V_{PHY}\left(s_{PHY}^{'}\right) = \max_{\substack{a_{APP} \in \mathcal{A}_{APP} \\ Z_{APP} \in \mathcal{Z}_{APP}}} \left[\tilde{V}\left(s_{PHY}^{'}, s_{APP}, a_{APP}, Z_{APP}\right)\right]$
PHY	$V(\boldsymbol{s}) = \max_{a_{PHY} \in \mathcal{A}_{PHY}} \left[\tilde{V}_{PHY}\left(s_{PHY}, a_{PHY}\right)\right]$

B. Why is a real-time solution needed?

The layered DP algorithm presented in the previous sub-section updates the state-value function and the policy for every state in every iteration, therefore this algorithm is called as **Synchronous Dynamic Programming (SDP)** as every state is updated synchronously. SDP is an offline solution, for one have to wait until the state-value function converges to V^{*} before it can implement the policy into real-time decision making, and the environment dynamics have to be known. Such approach can guarantee the policy to be optimal, but it is really computationally expensive sometimes, especially when the system state space is large. For example, if there are n_s states and n_a admissible actions for any state, then each SDP iteration, which consists of updating each state-value function and the policy exactly once, requires at most $O\left(n_s^2 n_a\right)$ computations. For illustration, if we assume that each I frame contains 20 packets, B frame of 5 packets and P frame of 8 packets, and the lifetime of three frames are 5, 4, 7 ΔT, respectively, then the buffer state space $\mathcal{K}$ itself contains $21^5 \times 6^4 \times 9^7 = 2.5 \times 10^{17}$ states, and thus, the entire state space $\mathcal{S}$ would be so large that the computation cost is unaffordable. Moreover, the network environment in the video streaming process is usually not stationary but changes over time. For example, the switch of video content will lead to the variation on the statistical property of the incoming data traffic, and the channel characteristics (e.g. SNR) of the wireless channel also varies from time to time. Therefore, the policy implemented during the control over the streaming process should be adaptively adjusted along with the dynamic change of the environment. Unfortunately, SDP can not make such policy adjustment online due to its offline characteristics. Every time the environment changes, SDP has to re-compute the policy, then implements it online. As we can see from the above, such offline-computing-and-online-implementing approach is so inefficient and expensive that is almost impossible to be used in real-time applications when the state space $\mathcal{S}$ inflates.

Table 2: RTDP operator at each layer

	DP operator at each layer
APP	$V_{PHY}\left(s_{PHY}^{'}\right) = \max_{\substack{a_{APP} \in \mathcal{A}_{APP} \\ Z_{APP} \in \mathcal{Z}_{APP}}} \left[\tilde{V}^{t}\left(s_{PHY}^{'}, s_{APP}^{t}, a_{APP}, Z_{APP}\right)\right]$
PHY	$V^{t+1}\left(\boldsymbol{s}^{t}\right) = \max_{a_{PHY} \in \mathcal{A}_{PHY}} \left[\tilde{V}_{PHY}^{t}\left(s_{PHY}^{t}, a_{PHY}\right)\right]$

Therefore, in this paper we propose the new RTDP algorithm to solve the optimal state-value function and policy update online. Our approach is to modify the offline and synchronous policy update, and consider performing the policy update and the real-time decision making of the policy in the video streaming process concurrently. The policy update and the real-time decision making interact as follows:

1. *Decision making*: In every time slot t, the decision on packet scheduling and power and rate allocation for the video streaming process is made based on the most up-to-date policy and the current system state;
2. *Policy update*: The value function and the policy of the current state are updated.

In RTDP, let $\boldsymbol{s}^{t}$ be the last state visited by the system in the t_{th} time slot, and $V^{t}(\boldsymbol{s})$ is the most up-to-date value function for all $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$. The system then executes action $\boldsymbol{\alpha}^{t}$, which can maximize the value function for the current state $\boldsymbol{s}^{t}$, that is

$$V^{t+1}\left(\boldsymbol{s}^{t}\right) = \max_{\boldsymbol{\alpha}^{t}} \left[R\left(\boldsymbol{s}^{t}, \boldsymbol{\alpha}^{t}\right) + \mu \sum_{\boldsymbol{s}^{'}} p\left(\boldsymbol{s}^{'} \mid \boldsymbol{s}^{t}, \boldsymbol{\alpha}^{t}\right) V^{t}\left(\boldsymbol{s}^{'}\right) \right]. \quad (17)$$

and the DP solution (optimal policy) for $\boldsymbol{s}^{t}$ is then updated as $\pi\left(\boldsymbol{s}^{t}\right) = \boldsymbol{\alpha}^{t}$.

Similar to [7], this centralized RTDP operator can be decomposed across the OSI layers using a layering computation as shown in Table 2, where

$$\tilde{V}^{k}\left(s_{PHY}^{'}, s_{APP}^{t}, a_{APP}, Z_{APP}\right) = R_{in}^{t}\left(s_{APP}^{t}, Z_{APP}\right)$$
$$+\mu \sum_{s_{APP}^{'} \in \mathcal{S}_{APP}} p\left(s_{APP}^{'} \mid s_{APP}^{t}, a_{APP}, Z_{APP}\right) V^{t}\left(s_{PHY}^{'}, s_{APP}^{'}\right), \quad (18)$$

$$\tilde{V}_{PHY}^{t}\left(s_{PHY}^{t}, a_{PHY}\right) = -\lambda_{PHY} R_{ex}\left(s_{PHY}^{t}, a_{PHY}\right)$$
$$+ \sum_{s_{PHY}^{'} \in \mathcal{S}_{PHY}} p\left(s_{PHY}^{'} \mid s_{PHY}^{t}, a_{PHY}\right) V_{PHY}\left(s_{PHY}^{'}\right). \quad (19)$$

Before presenting the complete algorithm of RTDP, we first analyze the convergence of such layered RTDP operator.

C. The convergence for layered RTDP

In this part, we prove that the layered RTDP Algorithm converges to the optimal value function V^* and the optimal policy π^* generated from SDP, under certain sufficient conditions.

To prove the convergence of layered RTDP, we first prove its equivalence to an asynchronous algorithm, and then use the Asynchronous Convergence Theorem to get the result.

Lemma 1: Layered RTDP is a special case of Asynchronous Dynamic Programming [16], where only one state is updated in every time slot. ■

Definition 1: There is a sequence of nonempty sets $\{X(k)\}$ with

$$\cdots \subset X(k+1) \subset X(k) \subset \cdots \subset X(0) \tag{20}$$

satisfying the following two conditions:

(a) *(Synchronous Convergence Condition)* We have

$$f(x) \in X(k+1), \qquad \forall k \ and \ x \in X(k) \tag{21}$$

Furthermore, if $\{y^k\}$ is a sequence such that $y^k \in X(k)$ for every k, then every limit point of $\{y^k\}$ is a fixed point of f.

(b) *(Box Condition)* For every k, there exists sets $X_i(k) \subset X_i$ that $X(k) = X_1(k) \times X_2(k) \times \cdots \times X_n(k)$.■

The Synchronous Convergence Condition implies that [14] the limit points of sequences generated by the (synchronous) iteration $x := f(x)$ are fixed points of f, assuming that the initial x belongs to $X(0)$. The Box Condition implies $x \in X(k)$ and $x' \in X(k)$, if we replace the j_{th} components of them, we still obtain two elements of $X(k)$.

Lemma 2: (Asynchronous Convergence Theorem [14]) If the Synchronous Convergence and Box Conditions hold for the sequence of nonempty sets $\{X(k)\}$, and the initial solution estimate $x(0) = \{x_1(0), \ldots, x_n(0)\}$ belongs to the set $X(0)$, then every limit point of $\{x(k)\}$ is a fixed point of f as defined in Definition 1.

Proof: The proof of Lemma 2 can be found in [14]. ■

Hence, as Lemma 1 has shown that layered RTDP is a special asynchronous algorithm, if we view the update of layered RTDP as some mapping f, we just need to prove the set of value functions under this mapping satisfies the Synchronous Convergence and Box Conditions, and then use Lemma 2 to prove layered RTDP's convergence.

First, we prove that the mapping defined by layered RTDP is a contraction mapping.

Definition 2: Let $\tilde{s}_{APP} = (s_{PHY}, s_{APP}, s'_{PHY})$ be the composite state at APP layer. Given this composite state, we can define the mapping at APP layer as

$$T_{APP}^{\tilde{s}_{APP}}(V) = \max_{\substack{a_{APP} \in \mathcal{A}_{APP} \\ Z_{APP} \in \mathcal{Z}_{APP}}} \left\{ \tilde{V}\left(s'_{PHY}, s_{APP}, a_{APP}, Z_{APP}\right) \right\}. \tag{22}$$

Similarly, at PHY layer, the mapping can be defined as

$$T_{PHY}^{s}\left(V_{PHY}\right) = \max_{a_{PHY} \in \mathcal{A}_{PHY}} \left\{ \tilde{V}_{PHY}\left(s_{PHY}, a_{PHY}\right) \right\}. \tag{23}$$

The layered operation of RTDP can be represented as the following iteration

$$V_{PHY}\left(\tilde{s}_{APP}\right) = T_{APP}^{\tilde{s}_{APP}}\left(V^t\right), V^{t+1}(\boldsymbol{s}) = T_{PHY}^{\boldsymbol{s}}\left(V_{PHY}\right). \tag{24}$$

The value function update T can be represented as the composition of the above two mappings

$$V^{t+1}(\boldsymbol{s}) = T\left(V^t(\boldsymbol{s})\right) = T_{PHY}^{\boldsymbol{s}}\left(T_{APP}^{\tilde{s}_{APP}}\left(V^t\right)\right). \tag{25}$$

■

Lemma 3: The value function update T is a contraction mapping.

Proof: Let us first look at $T_{APP}^{\tilde{s}_{APP}}$, for any two different value functions V and $\tilde{V}$, it is verify that

$$\left| T_{APP}^{\tilde{s}_{APP}}(V) - T_{APP}^{\tilde{s}_{APP}}\left(\tilde{V}\right) \right| \le \mu \left\| V - \tilde{V} \right\|_\infty I. \tag{26}$$

where $\|.\|_\infty$ is the maximum norm and I is the all-one vector $(1,1,\cdots,1)$. So $T_{APP}^{\tilde{s}_{APP}}(V)$ is a contraction mapping.

Similarly, $T_{PHY}^{\boldsymbol{s}}\left(V_{PHY}\right)$ is also a contraction mapping.

Therefore, the value function update $T(V(\boldsymbol{s}))$ is also a contraction mapping on V, as the composition of two contraction mappings. ■

With all the above preparation, we can now prove the major theorem in this section.

Theorem 1: The layered RTDP algorithm converges to the optimal value function as long as every state in the state space $\mathcal{S}$ is visited infinite times.

Proof: As the state space $\boldsymbol{\mathcal{S}}$ in our video streaming problem is assumed to be finite, denoted as $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_{N_s}\}$, where N_s is the size of $\boldsymbol{\mathcal{S}}$. Here we define

$$x(0) = \left\{V^0(\boldsymbol{s}_1), V^0(\boldsymbol{s}_2), \cdots, V^0\left(\boldsymbol{s}_{N_s}\right)\right\} = \{0, 0, \cdots, 0\} \tag{27}$$

as the initial value functions on all states and $X(0) = \Re^N$. Apparently, $x(0) \in X(0)$.

Then we set $X(k) = \left\{ x \in \Re^N \mid \left\| x - x^* \right\| \le \mu^k \left\| x(0) - x^* \right\| \right\}$. If we define the value function update $T(.) = f(.)$ and the value functions after *t* updates

$$x(t) = \left\{V^t(\boldsymbol{s}_1), V^t(\boldsymbol{s}_2), \cdots, V^t\left(\boldsymbol{s}_{N_s}\right)\right\}, \tag{28}$$

where $V^t(\boldsymbol{s}_i) = (T)^t\left(V^0(\boldsymbol{s}_i)\right)$ for all i, we have $x(t) \in X(t)$, and $\{X(t)\}$ is a sequence of nonempty sets satisfying the Synchronous Convergence and Box Condition in Definition 1. Based on Lemma 2, every limit point of $x(t)$ is a fixed point of T.

Assuming $x^* = V^*(\boldsymbol{\mathcal{S}})$ represents the fixed point of the mapping T in lemma 3, that is,

$$x^* = T\left(x^*\right), \tag{29}$$

We have

$$\lim_{t \to \infty} x(t) = x^* \ \text{and} \ \lim_{t \to \infty} V^t(\boldsymbol{s}_i) = V^*(\boldsymbol{s}_i) \tag{30}$$

for all *i*.

So in layered RTDP, if we make sure every state to be visited infinite times, t is guaranteed to go infinite and all the value functions can converge to optimal. ■

The only condition necessary for RTDP to converge is that every state should be visited infinite times, though this is not realistic in real applications, it tells us that the more a state is visited, the closer its value function will approach the optimum. To achieve it, we use a randomized learning policy to make sure that the system always continues to visit each

state. An example of such a learning policy is a form of Boltzmann exploration [16]:

$$Pr(\boldsymbol{\alpha} \mid \boldsymbol{s}, t, Q) = \frac{e^{\beta_t(\boldsymbol{s})} Q^t(\boldsymbol{s}, \boldsymbol{\alpha})}{\sum_{a \in A} e^{\beta_t(\boldsymbol{s})} Q^t(\boldsymbol{s}, \boldsymbol{\alpha})}, \quad (31)$$

where $Q^t(\boldsymbol{s}, \boldsymbol{\alpha})$ is the value function of taking action $\boldsymbol{\alpha}$ in state $\boldsymbol{s}$ for time slot *t*; $\beta_t(\boldsymbol{s})$ is the state-specific exploration coefficient for time slot t, which controls the rate of exploration in the learning policy. If we assume $\beta_t(\boldsymbol{s})$ to be infinite in the limit, then this learning policy has the following two properties:

1. Each action is executed infinitely often in every state that is visited infinitely often;
2. In the time limit, the learning policy is greedy with respect to the Q-value function with probability 1.

Learning policies satisfying the above conditions are denoted as *GLIE*, which stands for "Greedy in the Limit with Infinite Exploration" [15].

It has been proved in [15] that for any Reinforcement Learning algorithm that converges to the optimal value function and whose estimates stay bounded, using GLIE learning policies will ensure a concurrent convergence to an optimal policy, and therefore the GLIE policy in layered RTDP also converges for sure.

So finally, the operation of layered RTDP is summarized in Algorithm 2.

Algorithm 2: Algorithm for layered RTDP solution

Initialize V arbitrarily, e.g. $V(\boldsymbol{s}) = 0$ for all $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$
Start from initial state $s^{(0)}$
In the t_{th} time slot $(t \geq 0)$
Layered value function update as in Table 2
Policy update:
Select the action to be performed based on the probability

$$Pr(\boldsymbol{\alpha} \mid \boldsymbol{s}, t, Q) = \frac{e^{\beta_t(\boldsymbol{s})} Q^t(\boldsymbol{s}, \boldsymbol{\alpha})}{\sum_{a \in A} e^{\beta_t(\boldsymbol{s})} Q^t(\boldsymbol{s}, \boldsymbol{\alpha})}$$

Update policy for state $\boldsymbol{s}^t$, $\boldsymbol{\alpha}^t = \pi(\boldsymbol{s}^t)$
State transition: $\boldsymbol{s}^{t+1} \xleftarrow{a^t} \boldsymbol{s}^t$

D. The layered adaptive RTDP

The RTDP described above requires full prior knowledge of the network environment underlying the Markovian decision problem, such as state transition probabilities. When this knowledge is not available (this is commonly known as a Markovian decision problem with limited information), online learning methods should be combined with RTDP to adaptively learn the environment dynamics and adjust the policy in real-time. The key challenge for the online adaptation here is how to perform decision making in an unknown environment and how to adapt to the changing dynamics online.

The general idea for doing the online adaptation in RTDP is to first estimate a system model using an online system identification method before its operation (i.e. policy update and decision making) in each time slot. Subsequently, the policy update and decision making are performed using the improved system model. This method is called layered Adaptive RTDP (layered ARTDP), or RTDP with incomplete knowledge, and the RTDP in the previous section could be referred as RTDP with complete knowledge.

Assuming that the transition probabilities are unknown and need to be estimated, and the estimation is improved at each time slot t, denoted as $\left\{p^t(\boldsymbol{s}_j \mid \boldsymbol{s}_i, \boldsymbol{\alpha})\right\}, \forall i, j$. Let $N_{ij}^{\boldsymbol{\alpha}}(t)$ be the observed number of times before time slot t that action $\boldsymbol{\alpha}$ was executed when the system was in state $\boldsymbol{s}_i$ and made a transition to state $\boldsymbol{s}_j$ afterwards; and $N_i^{\boldsymbol{\alpha}}(t)$ denote the number of times action $\boldsymbol{\alpha}$ was executed in state $\boldsymbol{s}_i$ before time slot t. The state transition probability at time slot t can then be approximated as

$$p^t(\boldsymbol{s}_j \mid \boldsymbol{s}_i, \boldsymbol{\alpha}) = \frac{N_{ij}^{\boldsymbol{\alpha}}(t)}{N_i^{\boldsymbol{\alpha}}(t)}. \quad (32)$$

The operation of layered ARTDP is summarized in Algorithm 3.

Algorithm 3: Algorithm for layered ARTDP Solution

Initialize V arbitrarily, e.g. $V(\boldsymbol{s}) = 0$ for all $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$
Start from initial state $s^{(0)}$
In the t_{th} time slot $(t \geq 0)$
System Identification: $p^t(\boldsymbol{s}_j \mid \boldsymbol{s}^{t-1}, \boldsymbol{\alpha}^{t-1}) = \frac{N_{ij}^{\boldsymbol{\alpha}^{t-1}}(t)}{N_i^{\boldsymbol{\alpha}^{t-1}}(t)}$, for all $\boldsymbol{s}_j \in \boldsymbol{\mathcal{S}}$, and $\boldsymbol{s}^{t-1} = \boldsymbol{s}_i$
Layered value function update as in Table 2
Policy update:
Select the action to be performed based on the probability

$$Pr(\boldsymbol{\alpha} \mid \boldsymbol{s}, t, Q) = \frac{e^{\beta_t(\boldsymbol{s})} Q^t(\boldsymbol{s}, \boldsymbol{\alpha})}{\sum_{a \in A} e^{\beta_t(\boldsymbol{s})} Q^t(\boldsymbol{s}, \boldsymbol{\alpha})}$$

Update policy for state $\boldsymbol{s}^t$, $\boldsymbol{\alpha}^t = \pi(\boldsymbol{s}^t)$
State transition: $\boldsymbol{s}^{t+1} \xleftarrow{\boldsymbol{\alpha}^t} \boldsymbol{s}^t$

E. The convergence of layered ARTDP

In this section, we analyze the convergence of the layered ARTDP and provide some sufficient conditions for its convergence.

Theorem 2: The layered ARTDP algorithm converges to the optimal value function and the optimal policy, with probability one if the following conditions are met:

C1. The convergence conditions for the corresponding layered RTDP algorithm are met.
C2. In the limit, every action is executed from every state infinitely often.
C3. The estimates of the state transition probabilities converge to their true value with probability one.

Proof: In the layered ARTDP, let V_t^* denotes the optimal value function based on the estimates of transition probabilities $p^t(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ in the t_{th} time slot. Based on Condition C2, it is easy to tell that $\lim_{t \to \infty} V_t^* \overset{w.p.1}{\to} V^*$. For any small δ, assume

$$\left\| V_t^* - V^* \right\|_\infty < \delta, \; for \; t > T_\delta. \quad (33)$$

For any state $\boldsymbol{s}$, let $t^s(i)$ denote the time that $\boldsymbol{s}$ is backed up for the i_{th} time after T_δ, then we first prove that the following inequality holds for any state $\boldsymbol{s}$

$$\left|V_{t^s(i)+1}(\boldsymbol{s}) - V^*_{t^s(i)}(\boldsymbol{s})\right| < \mu^i \left\|V_{t^s(1)} - V^*_{t^s(1)}\right\|_\infty + \delta, \text{ for any } \delta \quad (34)$$

We use induction to prove it:

$$\begin{aligned}
&\left|V_{t^s(i)+1}(\boldsymbol{s}) - V^*_{t^s(i)}(\boldsymbol{s})\right| = \left|T\left(V_{t^s(i)}\right) - T\left(V^*_{t^s(i)}\right)\right| \\
&\le \mu \left\|V_{t^s(i)} - V^*_{t^s(i)}\right\|_\infty \le \mu\left\{\left\|V_{t^s(i)} - V^*_{t^s(i-1)}\right\|_\infty + \left\|V^*_{t^s(i-1)} - V^*_{t^s(i)}\right\|_\infty\right\} \\
&< \mu\left\{\left\|V_{t^s(i)} - V^*_{t^s(i-1)}\right\|_\infty + \delta\right\} \le \mu\left\{\left\|V_{t^s(i-1)+1} - V^*_{t^s(i-1)}\right\|_\infty + \delta\right\} \quad (35) \\
&\cdots \\
&\le \mu^i \left\|V_{t^s(1)} - V^*_{t^s(1)}\right\|_\infty + \frac{\mu(1-\mu^i)}{1-\mu}\delta \\
&\triangleq \mu^i \left\|V_{t^s(1)} - V^*_{t^s(1)}\right\|_\infty + \delta
\end{aligned}$$

Therefore, as $i \to \infty$, $\lim_{i\to\infty}\left|V_{t^s(i)+1}(\boldsymbol{s}) - V^*_{t^s(i)}(\boldsymbol{s})\right| < \delta$, for any state $\boldsymbol{s}$ and any small δ.

Finally, as $\lim_{t\to\infty} V_t^* \overset{w.p.1}{\to} V^*$, we can draw our conclusion that

$$\lim_{i\to\infty} V_{t^s(i)}(\boldsymbol{s}) \to V^*(s), \ w.p.1. \quad (36)$$

■

V. NUMERICAL RESULTS

In this experiment, we show that our layered RTDP algorithms have the performance close to optimal, and compare the performances of layered SDP, RTDP, and ARTDP.

A. Simulation setting

At APP layer, we use the video trace pattern as described in Section II.A, and assume that the statistic of incoming video traffic complies with the activity adaptive model as in [9] and has four possible states $\{I, P, B, BB\}$. To simplify our analysis, we further reduce the state space into 3 spaces $\{I, P, B\}$, and use the transition probability matrix of the 4-state model to approximate the transition probability between any two types of frames.

The frame rate at the encoder is assumed to be 20Hz, thus the length of one time slot $\Delta T = 1ms$.

The I frame has a life time of $50\Delta T$, and the B frame and the P frame have corresponding life times of $45\Delta T$ and $60\Delta T$ respectively. Each I frame is encoded into 20 packets, B frame is 5 packets, and P frame 8 packets. The average packet length $L = 1000bits$.

At the PHY layer, we assume that the channel has the maximum Doppler frequency $f_m = 50Hz$. The channel SNR has 9 levels, varying from 0 to infinite. The power allocation has 4 levels, $q \in [0.5, 1.0, 1.5, 2.0](mw)$. We use the MPSK modulation, and there are also 4 modulation levels as $\{BPSK, QPSK, 8PSK, 16PSK\}$.

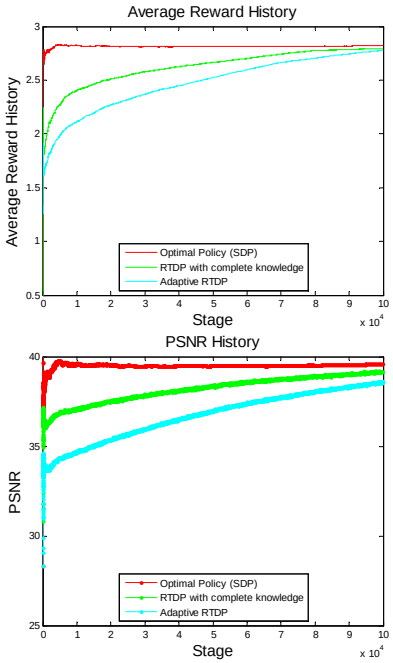

Figure 2. (top) The reward; (bottom) The PSNR received along time in video streaming process with stationary traffic

In this example, we consider how to decide the power allocation and modulation levels, and then select a set from the output buffer, to optimize the video quality received in this application meanwhile minimize the transmission energy at PHY layer.

B. Online adaptation with stationary incoming traffic

In this simulation, we compare the performance of SDP, RTDP, and ARTDP in the video streaming process, with stationary incoming traffic, that is, the traffic pattern and the transition probabilities between different states remain the same.

Figure 2 show the resulting average reward and PSNR achieved by different methods. As can be seen from these figures, the SDP generates the optimal policy, which keeps a constant reward. The RTDP methods will gradually catch up with the performance of SDP as they are learning through the real-time decision control, and their policies are getting closer to optimal.

For the RTDP methods themselves, the performance of RTDP with complete knowledge is initially better than that of ARTDP, which is straightforward as the RTDP has the full accurate knowledge about the network environment and the dynamics before the control operation starts, thus can learn the optimal policy much faster than ARTDP. When the states are visited with enough times, ARTDP can get enough accurate estimation on the system state transition probabilities, and the rate of its policy update would get closer to its non-adaptive competitor.

C. Online adaptation with non-stationary incoming traffic

In this simulation, the incoming traffic is non-stationary. After running 5×10^4 time slots, the incoming video changes into a low motion scenario, and hence, there are less I frames and more B and P frames in the incoming traffic, as well as some change in the transition probabilities between frames.

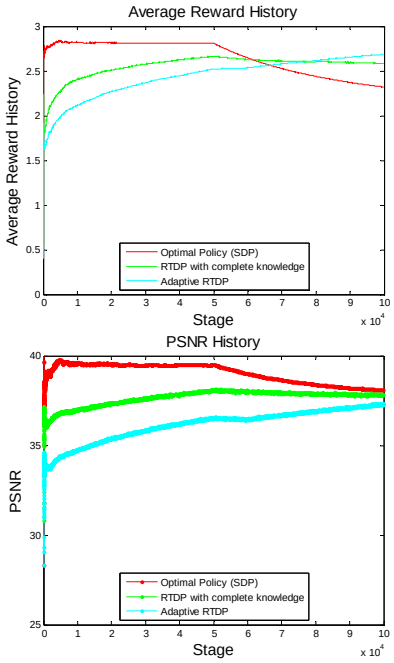

Figure 3. (top) The reward; (bottom) The PSNR received along time in video streaming process with non-stationary traffic

Figure 3 show the performances of the three methods in such a non-stationary environment. Since SDP uses a fixed policy which only fits certain fixed environment, its performance is most severely influenced by the environment dynamics, both in terms of the reward and PSNR.

The RTDP methods, on the other hand, provide much more stable performances compared to SDP. When the incoming traffic changes after 5×10^4 time slots, the reward and PSNR received by RTDP with complete knowledge stops to increase, which is due to the fact that the knowledge about the environment (i.e. state transition probabilities) possessed by RTDP is now out of date. But as RTDP is still in the middle of the learning process and the policy is sub-optimal, its performance does not have severe drop as that of SDP.

The method which fits the non-stationary environment best, is the ARTDP. As we observe, after the environment changes, the performance of ARTDP will slightly fall. But after another 10^4 time slots, ARTDP has again updated its estimate on the state transition probabilities of incoming traffic, and its received reward and PSNR start to rise.

VI. CONCLUSIONS

In this paper, we extend the proposed layered MDP framework in [7] from the setting where all network environment dynamics are considered known to the more realistic case in which the knowledge of the environment dynamics is limited or even completely unknown. Unlike the conventional Synchronous Dynamic Programming method used to solve the MDP problem, we propose a Real-time Dynamic Programming solution, which can also adhere to the layered OSI structure, but is capable of combining the policy update and real-time decision making. This approach can not only avoid the high computational complexity involved in solving the MDP problem, but also, importantly, can adaptively adjust its policy online, given the experienced changes in the environment dynamics. We prove the convergence of the RTDP methods for the considered cross-layer problem, and our experiment results show that the layered RTDP methods obtain near optimal performance in the long term and also exhibit significant advantages in terms of in the non-stationary environment.

REFERENCES

[1] M. van der Schaar, and S. Shankar, "Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms," *IEEE Wireless Commun. Mag.*, vol. 12, no. 4, Aug. 2005.

[2] M. van der Schaar and P. Chou, editors, "Multimedia over IP and Wireless Networks: Compression, Networking, and Systems," Academic Press, 2007.

[3] Q. Liu, S. Zhou, and G. B. Giannakis, "Cross-layer combing of adaptive modulation and coding with truncated ARQ over wireless links," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, May 2005.

[4] Y. J. Chang, F. T. Chien, and C. C. Kuo, "Cross-layer QoS analysis of opportunistic OFDM-TDMA and OFDMA networks," *IEEE J. Select. Areas Commun.*, vol 25, no. 4, pp. 657-666, May, 2007.

[5] M. van der Schaar, Y. Andreopoulos, and Z. Hu, "Optimized scalable video streaming over IEEE 802.11 a/e HCCA wireless networks under delay constraints," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 755-768, June 2006.

[6] M. van der Schaar, and D. Turaga, "Cross-Layer Packetization and Retransmission Strategies for Delay-Sensitive Wireless Multimedia Transmission," *IEEE Transactions on Multimedia*, vol. 9, no. 1, pp. 185-197, Jan., 2007.

[7] F. Fu and M. van der Schaar, "A New Systematic Framework for Autonomous Cross-Layer Optimization", IEEE Trans. Veh. Tech., to appear.

[8] A. G Barto, S. J Bradtke, S. P Singh, "Learning to act using real-time dynamic programming", Artificial Intelligence, Elsevier, 1995.

[9] D. S. Turaga and T. Chen, "Hierarchical Modeling of Variable Bit Rate Video Sources", Proc. of the 11th Packet Video Workshop.

[10] A. Albanese and M. Luby, "PET-priority encoding transmission," in High-Speed Networking for Multimedia Application. Norwell, MA: Kluwer, 1996.

[11] Q. Zhang, Saleem A. Kassam, "Finite-State Markov Model for Rayleigh Fading Channels," IEEE Trans. On Communications. Vol. 47, No. 11, November 1999.

[12] S. T. Chung, A. J. Goldsmith, "Degrees of Freedom in Adaptive Modulation: A Unified View," IEEE Trans. On Communications, Vol. 49, No. 9, September 2001.

[13] W. Chen, U. Mitra, M. J. Neely, "Energy-efficient Scheduling with Individual Packet Delay Constraints over a Fading Channel," Wireless Networks, Springer, 2008.

[14] D. P. Bertsekas, J. N. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods", Prentice Hall, 1989.

[15] S. Singh, T. Jaakkola, M. L. Littman, C. Szepesvari, "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms", Machine Learning, Springer, 2000.

[16] R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction", MIT Press, 1998.

[17] P. Chou, and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390-404, 2005.