LEARNING FOR CROSS-LAYER OPTIMIZATION

Fangwen Fu, and Mihaela van der Schaar

Multimedia Communications and Systems Lab, Electrical Engineering Department,
University of California Los Angeles (UCLA)
{fwfu, mihaela}@ee.ucla.edu

ABSTRACT

Cross-layer optimization solutions have been proposed in recent years to improve the performance of network users operating in a time-varying, error-prone wireless environment. However, these solutions often rely on ad-hoc optimization approaches with known environmental dynamics experienced at various layers by a user and violate the layered network architecture of the protocol stack. This paper presents a new theoretic foundation for cross-layer optimization, which allows each layer to autonomously *learn* the environmental dynamics, while maximizing the utility of the wireless user by optimally determining what information needs to be exchanged among layers. Hence, this cross-layer framework does not change the current layered architecture. The experimental results demonstrate that the proposed layered learning framework achieves near-optimal performance.

Index Terms— Cross-layer optimization, layered MDP, learning, information exchange, environmental dynamics.

1. INTRODUCTION

The Open Systems Interconnection (OSI) model [1] is a layered, abstract organization of the various communications and computer networks protocols. To optimize the different protocol parameters from different layers, the wireless stations (WSTAs) need to consider the dynamic wireless network “environment” resulting from the repeated interaction with other stations, the experienced time-varying channel conditions and, for delay-sensitive applications, the time-varying source characteristics. Moreover, it should be noted that a WSTA needs to jointly optimize the selected protocol parameters within each layer such that the utility of the WSTA is maximized. The joint optimization of the transmission strategies at the various layers is referred to as cross-layer design [2][3]. Recently, various cross-layer design methods have been proposed in order to jointly adapt the transmission strategies at each layer of the OSI stack to the rapidly varying environment and often scarce network resources by assuming that the environmental dynamics are known [4][7].

The advantage of the layered architecture is that the designer or implementer of the protocol or algorithm at a particular layer can focus on that layer without worrying about the rest of stack [3]. However, most existing cross-layer design solutions advocate improving the system utility by violating the current layered architecture of wireless networks. These cross-layer interactions create the dependencies among the layers which will affect not only the concerned layer but also other layers. Hence, such solutions are undesirable because they require a complete redesign of current networks and protocols and thus, require a high implementation cost [3].

Furthermore, most existing cross-layer design solutions aim at maximizing the WSTA’s utility by jointly adapting the transmission strategies across multiple layers to the known environmental dynamics [4][5]. These solutions, however, neglect that the environmental dynamics are generally unknown and the off-line cross-layer optimization will lead to worse performance.

Unlike the previous works that jointly optimize the cross-layer strategies in a centralized way, we propose a layered MDP solution to drive the cross-layer optimization. In this layered MDP framework, each layer makes its transmission decision (i.e. selects the transmission strategies, e.g. packet scheduling in the application (APP) layer, retransmission in the MAC layer and modulation selection in the physical (PHY) layer) in an autonomous manner, by considering the dynamics experienced at that layer as well as the information available from other layers. Importantly, using this layered optimization framework, we do not change the current layered architecture of the protocol stack. Moreover, the current algorithms and protocols currently implemented at each layer also remain unaffected, as the proposed framework requires only the exchange of information across layers and the optimization of available parameters at each layer. To exchange information across multiple layers, we define a message exchange mechanism in which the content of the message captures the performed transmission strategies and experienced dynamics at each layer.

To capture the unknown environment dynamics, we propose a layered learning algorithm to allow each layer to autonomously update its own transmission strategies based on the experienced environment dynamics.

The rest of the paper is organized as follows. Section 2 discusses the problem settings for the cross-layer optimization and formulates the cross-layer design as an MDP problem. Section 3 presents a layered value iteration algorithm for optimally solving the layered MDP. Section 4 discusses the on-line learning based on the layered MDP framework. Section 5 gives the simulations results. The paper concludes in Section 6.

2. CROSS-LAYER PROBLEM STATEMENT

We consider one WSTA transmitting its time-varying traffic to another WSTA (e.g. base station) over a wireless network (e.g. wireless LAN, cellular network, etc.). We also assume that there are L participating layers[1] in the protocol stack. Each layer is indexed $l \in \{1,...,L\}$ with layer 1 corresponding to the lowest participating layer (e.g. PHY layer) and layer L corresponding to the highest participating layer (e.g. APP layer). The WSTA interacts with the dynamic environment at various layers in order to maximize the application utility.

2.1 States

In this paper, the state of the layers is defined such that future transmission strategies can be determined independent of the past history given the current state. In other words, the state encapsulates all the past information required for future strategy adaptation. We refer to this type of state as Markovian state. When considering the layered architecture of current networks, we are able to define a state $s_l \in \mathcal{S}_l$ for each layer l. Then, the state of the entire WSTA is denoted by $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$, with $\boldsymbol{\mathcal{S}} = \prod_{l=1}^{L} \mathcal{S}_l$.

2.2 Actions

In a layered architecture, a WSTA takes different transmission actions in each state of each layer. The transmission actions can be classified into two types at each layer l: an external action is performed to determine the state transition, and an internal action is performed to determine the service provided to the upper layers for the packet(s) transmission.

The external actions at each layer l are denoted by $a_l \in \mathcal{A}_l$, where $\mathcal{A}_l$ is the set of the possible external actions available at layer l. The external actions for the WSTA in all the layers are denoted by $\boldsymbol{a} = [a_1,...,a_L] \in \boldsymbol{\mathcal{A}}$, where $\boldsymbol{\mathcal{A}} = \prod_{l=1}^{L} \mathcal{A}_l$. The internal actions are denoted by $b_l \in \mathcal{B}_l$, where $\mathcal{B}_l$ is the set of the possible internal actions available at layer l. The internal actions are performed by the WSTA to efficiently *utilize* the wireless medium given the network resource allocation and its own resource budget (e.g. power constraint), by providing the QoS required by the supported applications. The internal actions for the WSTA across all the layers are denoted by $\boldsymbol{b} = [b_1,...,b_L] \in \boldsymbol{\mathcal{B}}$, where $\boldsymbol{\mathcal{B}} = \prod_{l=1}^{L} \mathcal{B}_l$. Hence, the action at layer l is the aggregation of external and internal actions, denoted by $\xi_l = \begin{bmatrix} a_l & b_l \end{bmatrix} \in \mathcal{X}_l$, where $\mathcal{X}_l = \mathcal{A}_l \times \mathcal{B}_l$. The joint action of the WSTA is denoted by $\boldsymbol{\xi} = [\xi_1,...,\xi_L] \in \prod_{l=1}^{L} \mathcal{X}_l$.

The external actions are performed to drive the state transition. In general, because states are Markovian, the state transition of the WSTA only depends on the current state $\boldsymbol{s}$, the current performed actions, and the environmental dynamics. The corresponding transition probability is denoted by $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{\xi})$.

Due to the layered architecture of the wireless network, the state transition probability can be further decomposed. Using Bayes rule, the transition probability can be rewritten as

$$p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{\xi}) = \prod_{l=1}^{L} p(s_l' \mid \boldsymbol{s}_{1\to l\text{-}1}', \boldsymbol{s}, \boldsymbol{\xi}) \qquad (1)$$

where $\boldsymbol{s}_{1\to l}' = [s_1',...,s_l']$.

In this paper, based on the actions we mentioned before, the transition probability can be decomposed as

$$p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{\xi}) = \prod_{l=1}^{L-1} p(s_l' \mid \boldsymbol{s}_{1\to l\text{-}1}', s_l, a_l) p(s_L' \mid \boldsymbol{s}_{1\to L\text{-}1}', \boldsymbol{s}, a_L, \boldsymbol{b}) \ (2)$$

This decomposition is due to the layered network architecture and enables us to develop a layered MDP framework, which will be presented in Section 3.

2.3 Utility function

The utility gain obtained in layer L is based on the states and internal actions at each layer and it is denoted by $g(\boldsymbol{s}, \boldsymbol{b})$. The transmission cost at layer l represents the cost of performing both the external and internal actions, e.g. the amount of power allocated to determine the channel conditions or the tax (tokens, money) spent for consuming wireless resources. In general, the transmission cost of performing the external (internal) action at layer l is denoted by $c_l(s_l, a_l)$ ($d_l(s_l, b_l)$), which is a function of the external (internal) action and the state of layer l. For illustration, we assume that the reward is defined as

$$R(\boldsymbol{s}, \boldsymbol{\xi}) = g(\boldsymbol{s}, \boldsymbol{b}) - \sum_{l=1}^{L} \lambda_l^a c_l(s_l, a_l) - \sum_{l=1}^{L} \lambda_l^b d_l(s_l, b_l) \qquad (3)$$

where λ_l^a (λ_l^b) is a external (internal) Lagrangian multiplier in layer l, determined by the WSTA to trade off the utility and transmission cost. We assume that the Lagrangian multipliers λ_l^a and λ_l^b are known. The optimal Lagrangian multipliers depend on the available resource budget and can be obtained as in [6]. The reward in Eq. (3) can be further decomposed into two parts: one is the internal reward, which depends on the internal actions, and the other is the external reward, which depends on the external actions. The internal reward is

$$R_{in}(\boldsymbol{s}, \boldsymbol{b}) = g(\boldsymbol{s}, \boldsymbol{b}) - \sum_{l=1}^{L} \lambda_l^b d_l(s_l, b_l), \qquad (4)$$

and the external reward is

---

[1] If one layer does not participate in the cross-layer design, it can simply be omitted. Hence, we consider here only the L participating layers.

$$R_{ex}(\boldsymbol{s},\boldsymbol{a}) = -\sum_{l=1}^{L} \lambda_l^a c_l(s_l, a_l). \tag{5}$$

Hence, the reward is $R = R_{in} + R_{ex}$.

2.4 Foresighted decision making

As described in Section 2.2, the state transition at each layer is controlled by the external actions. For simplicity, we assume that the state transition in each layer is synchronized and operates at the same time scale, such that the transition can be discretized into stages during which the WSTA has constant state and performs static actions. The length of the stage can be determined based on how fast the environment changes. We use a superscript k to denote stage k. Hence, the state of the WSTA at stage $k \in \mathbb{N}$ is denoted by $\boldsymbol{s}^k$ with each element s_l^k being the state of layer l; similarly, the joint action performed by the WSTA at state k is $\boldsymbol{\xi}^k$ with each element $\xi_l^k = [a_l^k, b_l^k]$. The state transition probability is given by Eq. (2) and the stage reward is given by Eq. (3).

Unlike the tradition cross-layer adaptation that focuses on the myopic (i.e. immediate) utility, in the proposed cross-layer framework, the goal is to find the optimal internal and external actions at each stage such that a cumulative function of the rewards is maximized. We refer to this decision process as the *foresighted* cross-layer decision. By maximizing the cumulative reward, the WSTA is able to take into account the impact of the current actions on the future reward.

Specifically, we assume that the WSTA will maximize the discounted accumulative reward, which is defined as

$$\sum_{k=0}^{\infty} (\gamma)^k R(\boldsymbol{s}^k, \boldsymbol{\xi}^k \mid \boldsymbol{s}^0) \tag{6}$$

where γ is a discounted rate with $0 \le \gamma < 1$ and $\boldsymbol{s}^0$ is the initial state. Unlike the formulation in [6], where the time-average reward is considered, we use the discounted accumulated reward with higher weight on the current reward. The reasons for this are as follows: (i) for delay-sensitive applications, the data needs to be sent out as soon as possible to avoid expiration, and (ii) due to the unexpected environmental dynamics in the future, the WSTA may care more about the immediate reward. Hence, this needs to be considered when determining the values of γ for a specific cross-layer problem

3. LAYERED MDP FORMULATION

The internal and external actions need to be jointly optimized in order to determine the optimal cross-layer performance. Hence, information exchanges between layers are required. Existing cross-layer optimization frameworks require a central controller to decide the parameter configuration assuming that the complete information from all the layers is available to the central controller [7]. The foresighted cross-layer optimization can be formulated as an MDP and solved using value iteration [8]. However, the problem structure discussed in Section 2 enables us to decompose the MDP into a layered MDP which is defined as follows:

Definition (**Layered MDP with information exchange**) The layered MDP model with information exchange is given by the tuple $\mathcal{M} = \langle \mathcal{L}, \mathcal{S}, \{\mathcal{X}_l\}_{l=1}^{L}, \{\Theta_{l,l+1}\}_{l=1}^{L-1}, \{\Theta_{l,l-1}\}_{l=2}^{L}, p, R, \gamma \rangle$, where

- $\mathcal{L} = \{1,...,L\}$ is a set of L layers, each of which takes the internal and external actions individually.
- $\mathcal{S}$ is a finite set of states, each element $\boldsymbol{s} \in \mathcal{S}$ of which contains $[s_1, \cdots, s_L]$.
- $\mathcal{X}_l$ is a finite set of actions available to layer l, each element $\xi_l \in \mathcal{X}_l$ of which contains the external and internal actions, i.e. $\xi_l = [a_l, b_l]$.
- $\Theta_{l,l+1}$ is the message set sent by layer l to its upper layer $l+1$, where $\theta_{l,l+1} \in \Theta_{l,l+1}$ represents a message sent by layer l to its upper layer $l+1$ (i.e. upward message).
- $\Theta_{l,l-1}$ is the message set sent by layer l to its lower layer $l-1$, and $\theta_{l,l-1} \in \Theta_{l,l-1}$ represents a message sent by layer l to its lower layer $l-1$ (i.e. downward message).
- p is the transition probability function. $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{\xi})$ is the probability of moving from state $\boldsymbol{s} \in \mathcal{S}$ to the state $\boldsymbol{s}' \in \mathcal{S}$ when layer $l \in \mathcal{L}$ performs action ξ_l. We assume that the transition model is stationary and independent of the stage (i.e. time).
- $R : \mathcal{S} \times \prod_{l=1}^{L} \mathcal{X}_l \mapsto \mathbb{R}$ is the system stage reward function which has the form of $R(\boldsymbol{s}, \boldsymbol{\xi})$, i.e. the reward is determined by the state and actions in each layer.
- γ is the discounted factor.

The framework of the layered MDP with information exchange for the foresighted cross-layer optimization problem is illustrated in Figure 1. From this figure, we observe that the layer optimizer is not required to know other layers' state space, action space and dynamics models.

Upward message: At the state $\boldsymbol{s}^k$, by deploying the internal actions, the WSTA can determine for each layer (i) the probability of the packet being successfully received at the destination; (ii) the amount of time it takes to transmit on average; and (iii) the cost associated with its transmission. The transmission result of whether a packet is successfully received, is represented by the average packet loss ratio (PLR) at layer l at stage k, which is denoted by $\varepsilon_l^k(\boldsymbol{s}_{1\to l}^k, \boldsymbol{b}_{1\to l}^k)$ where $\boldsymbol{s}_{1\to l}^k = [s_1^k, ..., s_l^k]$ and $\boldsymbol{b}_{1\to l}^k = [b_1^k, ..., b_l^k]$. The average amount of time spent on transmitting one packet at layer l at stage k is denoted by $t_l^k(\boldsymbol{s}_{1\to l}^k, \boldsymbol{b}_{1\to l}^k)$. The aggregated transmission cost incurred by performing internal actions at layer l is defined by $f_l^k\left(\boldsymbol{s}_{1\to l}^k, \boldsymbol{b}_{1\to l}^k\right) = \sum_{l'=1}^{l} \lambda_{l'}^b d_{l'}\left(s_{l'}^k, b_{l'}^k\right)$.

To compute the internal reward function $R_{in}\left(\boldsymbol{s}^k, \boldsymbol{b}^k\right)$, layer L has to know the packet loss probability, the average amount of time for packet transmission and the transmission cost provided from the lower layers in stage k. We can define a message which captures this information from lower layers. This message is the QoS at layer l which is defined as a three-tuple $Z_l^k = \left[\varepsilon_l^k, t_l^k, f_l^k\right]^T$. The QoS at layer l represents the service layer l provides to its upper layer $l+1$. Using the QoS, layer $l+1$ does not need to know the actions and dynamics at lower layers.

By knowing the QoS Z_{L-1}^k provided from layer $L-1$, layer L can be computed as $R_{in}\left(s_L^k, b_L^k \mid Z_{L-1}^k\right)$. In other words, the internal reward R_{in} is independent of the states and actions in the lower layers, given the QoS Z_{L-1}^k provided from layer $L-1$. The more details about the upward message is presented in [9]. Hence, the upward message is $\theta_{l,l+1} = \mathcal{Z}_l^k$ where $\mathcal{Z}_l^k$ is the necessary QoS levels required by the upper layers. The more details about the upward message are presented in [9]

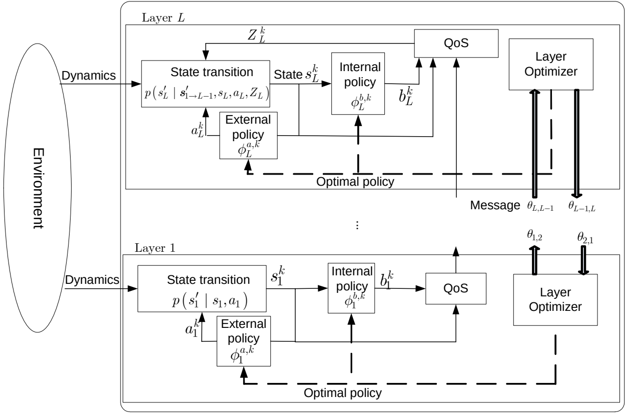

Figure 1. Layered cross-layer optimization framework

Downward message: As in the definition of the layered MDP, each layer is regarded as an autonomous entity that performs its own actions. However, the layers can cooperate via the information exchange to find the optimal state-value function $V^*(\boldsymbol{s})$ as in the value iteration for the central MDP [8]. By decomposing the value iteration for the central MDP given in [9], we can obtain the following theorem.

Theorem 1: The state-value function $V^*(\boldsymbol{s})$ corresponding to the optimal policy can be obtained using a layered value iteration algorithm. At iteration n, each layer performs a sub-value iteration which is given in Table 1.

The proof is omitted here due to the space limitation and can be found in [9].

The layered value iteration is performed as follows: at each iteration n, layer L performs the sub-value iteration as in Eq. (7) to obtain the state-value function $V_{n,L-1}^*\left(\boldsymbol{s}_{1\to L\text{-}1}'\right)$ which services as future state-value function at layer $L-1$. Then, in general, layer l performs the sub-value iteration as in Eq. (8) based on the future state-value function from layer $l+1$ to generate $V_n^*\left(\boldsymbol{s}_{1\to l-1}'\right)$. Finally, layer 1 performs the sub-value iteration as in Eq. (9) to generate the state-value function $V_{n,L}^*\left(\boldsymbol{s}_{1\to L}\right)$ which is $V_n^*(\boldsymbol{s})$ as in the centralized value iteration.

Then the message exchanged from layer $l+1$ to layer l is $\theta_{l+1,l} = \left\{V_{n-1}^*\left(\boldsymbol{s}_{1\to l}'\right)\right\}$.

Table 1. Sub-value iteration at each layer.

Layer	Sub-value iteration form at iteration n
L	$V_{n,L-1}^*\left(\boldsymbol{s}_{1\to L\text{-}1}'\right) = \max\limits_{a_L \in \boldsymbol{\mathcal{A}}_L, Z_L \in \mathcal{Z}_L} \left[R_{in}\left(s_L, Z_L\right) - \lambda_L c_L\left(s_L, a_L\right) + \gamma \sum\limits_{s_L \in \mathcal{S}_L} p\left(s_L' \mid \boldsymbol{s}_{1\to L-1}', s_L, a_L, Z_L\right) V_{n-1,L}^*\left(\boldsymbol{s}_{1\to L}'\right)\right]$ (7)
l	$V_{n,l-1}^*\left(\boldsymbol{s}_{1\to l\text{-}1}'\right) = \max\limits_{a_l \in \boldsymbol{\mathcal{A}}_l} \left[-\lambda_l c_l\left(s_l, a_l\right) + \sum\limits_{s_l' \in \mathcal{S}_l} p\left(s_l' \mid \boldsymbol{s}_{1\to l-1}', s_l, a_l\right) V_{n,l}^*\left(\boldsymbol{s}_{1\to l}'\right)\right]$ (8)
1	$V_{n,L}^*\left(\boldsymbol{s}_{1\to L}\right) = \max\limits_{a_1 \in \boldsymbol{\mathcal{A}}_1} \left[-\lambda_1 c_1\left(s_1, a_1\right) + \sum\limits_{s_1' \in \mathcal{S}_1} p\left(\boldsymbol{s}_1' \mid \boldsymbol{s}_1, a_1\right) V_{n,1}^*\left(\boldsymbol{s}_1'\right)\right]$ (9)

4. ON-LINE ADAPTATION

In this section, we extend the layered MDP framework to perform on-line adaptation for the case in which the models are unknown.

4.1 On-line adaptation using actor-critic learning

When the dynamics models are unknown, these models can be learned using learning techniques [10]. To highlight the advantage of the proposed layered MDP framework, we use for illustration the actor-critic reinforcement learning algorithm [10] for the on-line transmission strategy adaptation. The actor-critic algorithm separately updates the policy and the state-value function for each state. The *policy structure* is used to select actions at each state and is called the *actor*. The *state-value function* is used to criticize the actions selected by the actor and is called the *critic*. We briefly discuss the state-value function and policy update used in the actor-critic algorithm. The algorithm's details about the algorithm can be found in [10].

4.1.1 State-value function update

During the on-line adaptation, the state-value function $V(\boldsymbol{s})$ is unknown and must be estimated on-line. When performing action $\boldsymbol{\xi}^k$, we can update the state-value function, given the current reward $R\left(\boldsymbol{s}^k, \boldsymbol{\xi}^k\right)$ as follows:

$$V^{k+1}(\boldsymbol{s}^k) \leftarrow V^k(\boldsymbol{s}^k) + \alpha\begin{bmatrix} R(\boldsymbol{s}^k,\boldsymbol{\xi}^k) + \\ \gamma V^k(\boldsymbol{s}^{k+1}) - V^k(\boldsymbol{s}^k) \end{bmatrix} \quad (10)$$

where α is a positive step-size parameter and $V^k(\cdot)$ and $V^{k+1}(\cdot)$ are the estimated future rewards for stage k and stage $k+1$, respectively. Since the real future reward is unknown, $V^k(\boldsymbol{s}^{k+1})$ is used instead to update the state-value function. The update procedure for the value function in the actor-critic learning algorithm is performed in the *state-value function* update module illustrated in Figure 2.

4.1.2 Policy update

The state-value function $V^k(\cdot)$ is used to criticize the selected action. After each action selected by the actor, the critic evaluates the selected action at current state $\boldsymbol{s}^k$ to determine whether the value function at the current state performs better or worse than expected. This evaluation can be defined as the time-difference error as follows:

$$\delta^k = R(\boldsymbol{s}^k,\boldsymbol{\xi}^k) + \gamma V^k(\boldsymbol{s}^{k+1}) - V^k(\boldsymbol{s}^k) \quad (11)$$

If the error δ^k is positive, it means that the tendency to select action $\boldsymbol{\xi}^k$ should be strengthened in the future, while if it is negative, the tendency to select $\boldsymbol{\xi}^k$ should be weakened.

To generate the action, the actor defines a value $\rho(\boldsymbol{s},\boldsymbol{\xi})$ at state $\boldsymbol{s}$ for each action $\boldsymbol{\xi}$ to indicate the tendency to select that action. Then the actor generates the action according to the Gibbs softmax method [10]:

$$\Psi(\boldsymbol{s},\boldsymbol{\xi}) = e^{\rho(\boldsymbol{s},\boldsymbol{\xi})} \Big/ \sum_{\boldsymbol{\xi}' \in \prod_{l=1}^{L} \mathcal{X}_l} e^{\rho(\boldsymbol{s},\boldsymbol{\xi}')} \quad (12)$$

where $\Psi(\boldsymbol{s},\boldsymbol{\xi})$ represents the probability of performing action $\boldsymbol{\xi}$ at state $\boldsymbol{s}$.

The strengthening and weakening of the action can then be implemented by increasing or decreasing the tendency as follows:

$$\rho(\boldsymbol{s}^k,\boldsymbol{\xi}^k) \leftarrow \rho(\boldsymbol{s}^k,\boldsymbol{\xi}^k) + \beta\delta^k(1 - \Psi(\boldsymbol{s}^k,\boldsymbol{\xi}^k)) \quad (13)$$

where β is a positive step-size parameter and reflects the learning rate for the tendency update. The policy update is performed in the *policy* module in Figure 2.

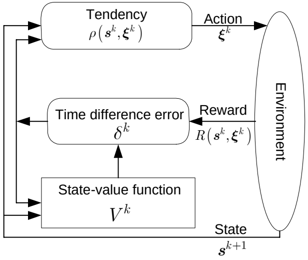

Figure 2. Actor-critic learning structure based on [10]

4.2 On-line adaptation using layered learning

Based on the actor-critic learning algorithm, we develop a layered actor-critic learning algorithm which takes into account the current layered network architecture. In this layered learning algorithm, each layer has its own actor and critic to select the action and criticize the selected action.

4.2.1 State-value update

Recall the value iteration at layer $l \in \{1,\ldots,L\}$. We can define the time-difference error at layer l as

$$\delta_l^k = \begin{cases} R_{in}(s_L^k,Z_L^k) - \lambda_L c_L(s_L^k,a_L^k) + \gamma V^k(\boldsymbol{s}^{k+1}) \\ \qquad - V_{L-1}^k(\boldsymbol{s}_{1\to L-1}^{k+1}) & l = L \\ -\lambda_1^c c_1(a_1^k) + V_1^k(s_1^{k+1}) - V^k(\boldsymbol{s}^k) & l = 1 \\ -\lambda_l^c c_l(a_l^k) + V_l^k(\boldsymbol{s}_{1\to l}^{k+1}) - V_{l-1}^k(\boldsymbol{s}_{1\to l-1}^{k+1}) & o.w. \end{cases} .(14)$$

Then, $V_l^{k+1}(\boldsymbol{s}_{1\to l}^{k+1})$ is updated as

$$V_l^{k+1}(\boldsymbol{s}_{1\to l}^{k+1}) \leftarrow V_l^{k+1}(\boldsymbol{s}_{1\to l}^{k+1}) + \alpha\delta_{l+1}^k, l = 1,\ldots,L-1 . \quad (15)$$

The state-value function $V^{k+1}(\boldsymbol{s}^k)$ is updated as

$$V^{k+1}(\boldsymbol{s}^k) \leftarrow V^k(\boldsymbol{s}^k) + \alpha\left[R_{in}(s_L^k,Z_L^k) - \sum_{l=1}^{L}\lambda_l^c c_l(s_l^k,A_l^k) + \gamma V^k(\boldsymbol{s}^{k+1}) - V^k(\boldsymbol{s}^k)\right] \quad (16)$$

4.2.2 Policy update

Given the state at each layer, the internal actions are independent of the environmental dynamics. In this learning algorithm, the state $\boldsymbol{s}$ is assumed to be known by each layer. From Section 3, we know that the optimal frontier $\mathscr{Z}_L$ only depends on the state $\boldsymbol{s}$, and hence, layer L can select the optimal QoS $Z_L \in \mathscr{Z}_L$. The tendency at layer L is updated using the time-difference error δ_L^k to strengthen or weaken the currently selected action, as follows:

$$\rho(\boldsymbol{s}^k,a_L^k,Z_L^k) \leftarrow \rho(\boldsymbol{s}^k,a_L^k,Z_L^k) + \beta\delta_l^k(1 - \Psi_L(\boldsymbol{s}^k,a_L^k,Z_L^k)) \quad (17)$$

Similarly, the tendency at layer l is updated as

$$\rho(\boldsymbol{s}^k,a_l^k) \leftarrow \rho(\boldsymbol{s}^k,a_l^k) + \beta\delta_l^k(1 - \Psi_l(\boldsymbol{s}^k,a_l^k)) \quad (18)$$

Where Ψ_l is computed similarly to Eq. (12). The mixed action is generated similarly to Eq. (12). The layered learning algorithm is portrayed in Figure 3.

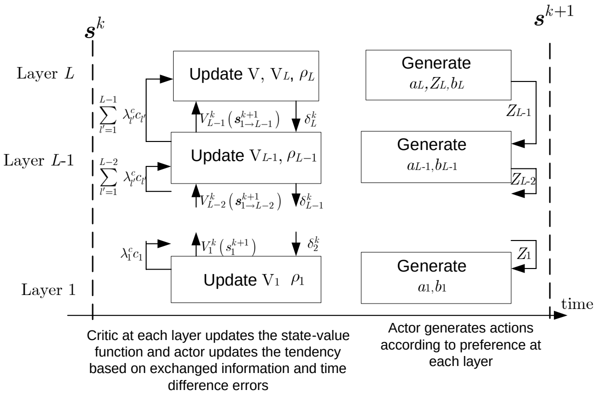
$\boldsymbol{s}^k$
$\boldsymbol{s}^{k+1}$
Layer L
Layer L-1
Layer 1
Update V, V_L, ρ_L
Update V_{L-1}, ρ_{L-1}
Update V_1 ρ_1
$\sum_{l'=1}^{L-1}\lambda_{l'}^c c_{l'}$
$\sum_{l'=1}^{L-2}\lambda_{l'}^c c_{l'}$
$\lambda_1^c c_1$
$V_{L-1}^k(\boldsymbol{s}_{1\to L-1}^{k+1})$
δ_L^k
$V_{L-2}^k(\boldsymbol{s}_{1\to L-2}^{k+1})$
δ_{L-1}^k
$V_1^k(s_1^{k+1})$
δ_2^k
Generate
a_L, Z_L, b_L
Generate
a_{L-1}, b_{L-1}
Generate
a_1, b_1
Z_{L-1}
Z_{L-2}
Z_1
time
Critic at each layer updates the state-value function and actor updates the tendency based on exchanged information and time difference errors
Actor generates actions according to preference at each layer

Figure 3. Proposed layered actor-critic learning procedure

5. ON-LINE ADAPTATION SIMULATION

In this simulation, we show that our layered learning algorithm adheres to the layered architecture, while performing as well as the centralized learning algorithm.

In this simulation, we consider that the WSTA transmits delay-sensitive data to another WSTA and accesses the wireless channel using TDMA (e.g. as in 802.11e HCF). Assume that the time is slotted and divided into frames consisting of N time slots. We consider the optimization of the transmission strategies available at the APP, MAC, and PHY layers, i.e. $L = 3$.

In the PHY layer, the channel gain can be modeled as a finite state Markov chain (FSMC) [11]. To satisfy the service requirement from upper layers, the PHY layer adapt its transmission power level, and the modulation schemes based on the channel dynamics.

In the MAC layer, we consider a more general multi-user scheduling method which allows the WSTA to dynamically compete for the available time slots [12]. At the beginning of each frame, the WSTA competes with other WSTAs for the time to access the spectrum. We can model the amount of time allocated to the WSTA as a finite state Markov chain, which is controlled by the competition bid. Besides competing for the resource, the MAC can also perform error control algorithms (e.g. ARQ) to improve the service provided to the upper layers.

In the APP layer, the WSTA generates delay-sensitive data. The delay-sensitivity is represented by the delay deadlines after which the packets will expire and therefore not contribute to the WSTA's utility. As in [4], we can model the number of packets with the various delay deadlines available for transmission as a Markov chain. The number of packets available for transmission depends on the source coding parameters adaptation as well as the transmission strategies at the lower layers.

The dynamics models (transition probabilities) at each layer are unknown. When using the centralized learning algorithm, a central entity within the WSTA is assumed to update the state-value function (critic) and policy (actor) and to choose the action to be performed. When using the layered learning algorithm, each layer updates its own state-value function and policy using the information from other layers, and performs its own action autonomously. Figure 4 shows the average reward achieved by the traditional learning (shown in Section 4.1) and layered learning algorithms (shown in Section 4.2). When learning for enough time, the layered learning algorithm approaches the optimal solution while the centralized one still has a gap of approximately 0.1. This can be explained as follows: in the centralized algorithm, the wireless user has to learn the joint environmental dynamics and update the transmission strategies at each layer simultaneously, which often requires more time to find the optimal transmission strategies. However, in the layered learning algorithm, each layer updates its own state-value function and tendency based on the exchanged information, and autonomously adapts its transmission strategies to its own experienced dynamics. The proposed autonomous adaptation allows each layer to quickly find its optimal actions in the various states, thereby resulting in an improved performance as opposed to the centralized learning solution.

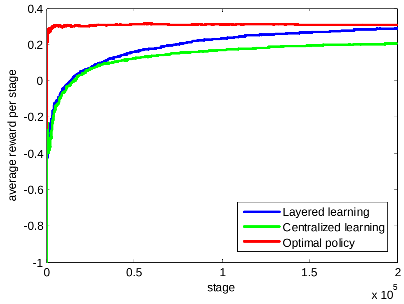

Figure 4. Average reward achieved using both centralized learning and layered learning

6. CONCLUSION

In this paper we formulated the dynamic cross-layer optimization problem as a layered MDP with information exchanges among layers. The layered MDP framework also allows each layer to autonomously learn its environment dynamics on-line. Our results show that the layered learning algorithm outperforms the traditional learning algorithm for cross-layer optimization.

REFERENCES

[1] D. Bertsekas *et al*, "Data networks," Prentice Hall, Inc. Upper Saddle River, NJ, 1987.

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

[3] V. Kawadia *et al*, "A cautionary perspective on cross-layer design," *IEEE Wireless Commun.*, pp. 3-11, vol. 12, no. 1, Feb. 2005.

[4] T. Holliday *et al*, "Optimal Power Control and Source-Channel Coding for Delay Constrained Traffic over Wireless Channels," *Proceedings of ICC'02*, May 2002.

[5] A. Ekbal *et al*, "QoS-constrained physical layer optimization for correlated flat-fading wireless channels," *proceedings of ICC'04*, vol. 7, pp. 4211-4215, June., 2004.

[6] D. Djonin *et al*, "MIMO transmission control in fading channels-a constrainted Markov decision process formulation with monotone randomized policies," *IEEE Trans. Signal Process.*, vol. 55, no. 10, pp. 5069-5083, Oct. 2007.

[7] X. Wang *et al*, "Analyzing and optimizing adaptive modulation coding jointly with ARQ for QoS-guaranteed traffic," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, Mar. 2007.

[8] D. P. Bertsekas, "Dynamic programming and optimal control," 3rd, Athena Scientific, Belmont, Massachusetts, 2005.

[9] F. Fu *et al*, "A new theoretic foundation for cross-layer optimization," Technique report, Dec. 2007, available on http://xxx.arxiv.org/abs/0712.2497.

[10] R. S. Sutton *et al*, "Reinforcement learning: an introduction," Cambridge, MA:MIT press, 1998.

[11] Q. Zhang *et al*, "Finite-state Markov Model for Reyleigh fading channels," *IEEE Trans. Commun.* vol. 47, no. 11, Nov. 1999.

[12] F. Fu *et al*, "Non-collaborative resource management for wireless multimedia applications using mechanism design," *IEEE Transaction on Multimedia*, vol. 9, no. 4, pp. 851-868, Jun. 2007.