Autonomic and Distributed Joint Routing and Power Control for Delay-Sensitive Applications in Multi-Hop Wireless Networks

Zhichu Lin and Mihaela van der Schaar, *Fellow, IEEE*

***Abstract*—Multi-hop wireless networks can provide flexible network infrastructures at a low cost. However, most existing wireless networking solutions are designed for delay-insensitive applications, thereby resulting in poor performance when handling delay-sensitive applications. Traditionally, network design problems are formulated as static optimizations, by assuming the network characteristics remain static. However, these solutions are not optimal when the environments are dynamic. Recently, several research works apply machine learning to maximize the performance of multi-hop wireless networks in dynamic environments, but they either only focus on determining policies at the network layer, without considering the lower-layers' actions, or use centralized learning approaches, which are inefficient for delay-sensitive applications, due to the large delay when propagating messages throughout the network. We propose in this paper a new solution that enables the nodes to autonomously determine their routing and transmission power to maximize the network utility, in a dynamic environment. We formulate the problem as a Markov Decision Process, and propose a distributed computation of the optimal policy. Moreover, we use reinforcement-learning to find the optimized policy when the dynamics are unknown. We explicitly consider the impact of the information overhead on the network performance, and propose several novel algorithms to reduce the information overhead.**

***Index Terms*—Multi-hop wireless networks, cross-layer optimization, informationally-decentralized Markov Decision Process, reinforcement learning.**

I. Introduction

A Multi-hop wireless network can be modeled as a dynamic network, consisting of several interconnected wireless nodes, which aim to jointly optimize the overall network utility, given the resource constraints of the wireless communication channels and also, importantly, the mutual interferences (coupling) resulting when nodes are simultaneously transmitting. In this paper, we study a communication scenario where multiple delay-sensitive streams (e.g. video streams) need to be concurrently transmitted over a multi-hop wireless network. At each hop, a node can optimize its cross-layer transmission strategies (i.e. relay selection, transmission power, etc.) in order to support the transmission of these delay-sensitive streams while explicitly considering the impact of its selected strategy on its neighboring nodes at the various OSI layers (power interference at the physical layer, network congestion at the network layer, etc.).

Numerous solutions for multi-user cross-layer optimization in multi-hop networks, e.g. optimal solutions for joint power-control and routing, already exist. For instance, centralized, flow-based optimization methods have been deployed in [1][2][3] to optimize the performance of (multi-hop) wireless networks by assuming that both the traffic and the wireless environment are static, and finding the optimal cross-layer transmission strategies using methods such as convex optimization and duality theory. Such solutions are, however, not suitable when delay-sensitive applications need to be transmitted across the multi-hop wireless networks, because they involve both substantial communication overheads and, importantly, they incur large delays for propagating the traffic and network information from the different nodes to a network controller (optimizer) and, subsequently, propagating the decision from the controller to the various nodes. Alternatively, distributed approaches based on machine learning have also been proposed in [4][5][6][7], which focus on determining the optimal policy that maximizes the network performance in dynamic environments. In [5], a collaborative reinforcement learning method was proposed for MANET routing. This method can substantially reduce the amount of information exchange among the nodes, compared to the centralized learning method. However, this method does not consider the interference in the physical layer (PHY), which will actually affect the link reliability, and also affect the delay of the packets. Thus, it is not suitable for a multi-hop network which needs to support delay-sensitive applications. In [8], a distributed cross-layer approach based on queuing analysis was proposed to maximize the received video quality from multiple video sources. Although this solution can improve the goodput (defined in this paper as the throughput within a certain delay deadline) in a static network, it is not able to achieve good performance in a dynamic wireless environment, because the nodes choose their cross-layer strategies based only on the immediate experienced delay feedback, and do not consider the effect of their current actions on their future delays.

To address the aforementioned challenges, in this paper, we aim to find a distributed routing and power control algorithm to enable the nodes in the multi-hop network to autonomously optimize the overall performance of multiple delay-sensitive applications under various types of network dynamics, by

Manuscript received August 14, 2009; revised January 11, 2010; accepted June 24, 2010. The associate editor coordinating the review of this paper and approving it for publication was R. Cheng.

The authors are with the Department of Electrical Engineering, University of California, Los Angeles, CA, 90024 USA (e-mail: linzhichu@gmail.com, mihaela@ee.ucla.edu).

Digital Object Identifier 10.1109/TWC.2010.111910.091238

acting solely on their local available information and the limited information exchanges with their neighboring nodes. We will first assume that a delay-driven scheduling scheme is used at the application layer, and then we will formulate the joint routing and power control problem as a Markov Decision Process (MDP) for which we can find the optimal policy that determines the cross-layer transmission strategies to be selected at each node, by assuming that the network dynamics are known. Subsequently, we will propose a distributed computation of the optimal policy, which can enable the nodes to autonomously make decisions in real-time, based on their local information exchanges with other nodes, as well as significantly reduce the delay of propagating these messages back and forth, as in the case of a centralized approach. Using this distributed MDP solution, we will investigate how nodes can autonomously learn the network dynamics online, based on their available information. The online learning enables the users to adapt their cross-layer strategies on-the-fly to the dynamic environment such that they can cooperatively maximize the utility of the delay-sensitive applications. We will also explicitly consider how the information exchanges among nodes will impact the overall network performance in various network scenarios.

In summary, we make the following contributions in this work:

- We deploy foresighted rather than myopic decision making for determining the joint routing and power control solutions. Since both the traffic and the wireless channels are dynamic, a myopic node may not make optimal decisions in the long term. To enable nodes to make foresighted decisions that optimize their long term performance, we propose to use MDP to model the dynamic decision making at each node.
- We propose an efficient distributed computation of the optimal transmission strategy by decoupling the decision making at different nodes and enable the nodes to autonomously determine their optimal cross-layer strategies based solely on their local available information. This decentralized solution significantly reduces the computation complexity as compared to a centralized computation. More importantly, it also enables nodes to make optimal decisions for their cross-layer strategies autonomously, in real-time, by relying only on their local available information. Hence, individual nodes do not require the acquisition of global information for their autonomous decision making, which would have caused a large delay and also high communication overhead.
- We consider a dynamic environment rather than a static one, which means that both the wireless channel and the source traffic are time-varying. However, the distributed nature of the multi-hop networks makes it very difficult for the individual nodes to obtain timely information about the entire dynamic network. Therefore, we propose to use online learning methods to enable the nodes to adapt their transmission strategies to the dynamic environment based on their local information and limited feedback from their neighboring nodes.
- We explicitly consider the impact of information exchange overheads on the overall network performance. On one hand, if nodes acquire more information from other nodes about the dynamic environment, this can help them to derive better policies, but, on the other hand, these exchanges also cause increased overheads, thereby lowering the bandwidth available for the delay-sensitive traffic transmission. We propose different methods to reduce the information exchange overheads, and evaluate their resulting performances.

The paper is organized as follows. Section II presents the considered multi-hop wireless network setting at different layers and formulates the problem using MDP. In Section III, we discuss a distributed computation of the optimal policy based on the factorization of MDP. The proposed method enables wireless nodes to make decisions autonomously, based on their local available information exchanged with their neighboring nodes. In Section IV, we propose to use reinforcement learning methods to enable autonomous wireless users to adapt their transmission strategies online when the network dynamics are unknown, and also discuss various methods to reduce the information exchange overhead as well as the delay for propagating the information. Section V presents our simulation results, and the conclusions are drawn in Section VI.

II. Considered Network Settings and Problem Formulation

A. Network Setting

The network model used in this paper is similar to the one used in [8]. We investigate the performance of transmitting V delay sensitive data streams over a multi-hop wireless network (see Fig. 1). Each data stream corresponds to a source-destination pair. The network consists of H hops with the first hop being source nodes, and we define $M_h = \{m_{h,1}, m_{h,2}, \ldots, m_{h,|M_h|}\}$ to be the set of nodes at the h-th hop $(1 \le h \le H)$. The destination nodes are $\{Dest_1, Dest_2, \ldots, Dest_J\}$.

Delay-sensitive data packets with different delay deadlines are generated by multiple sources in the first hop, and relayed hop-by-hop by the wireless nodes in the multi-hop network until the destinations at the H-th hop receive the packets. We denote the probability that node $m_{1,i}$ has a new source packet of delay-deadline D as $\rho_i(D)$ [8], with $\sum_{D=D_{min}}^{D_{max}} \rho_i(D) \le 1$ and $D \in [D_{min}, D_{max}]$. Note that this summation may be smaller than 1 because a source node may not receive packets in every time slot.

Each source node needs to transmit its traffic to a destination node. Hence, each data packet in the network has a specific destination, and we assume that the relay nodes can extract this information from the IP header of the packet [8]. We assume that each node operates in full-duplex mode, and can only communicate with the nodes in its neighborhood[1], i.e. node $m_{h,i}$ can only send data packets to $m_{h+1,j}$ if and only if $m_{h,i} \in M_h$ and $m_{h+1,j} \in M_{h+1}$. In this paper, we assume that the system is time-slotted and wireless nodes determine their cross-layer transmission strategies at the beginning of

[1]We assume each node can discover its own and also its neighbors' positions, as in geometric routing [9] or the routing method in [13].

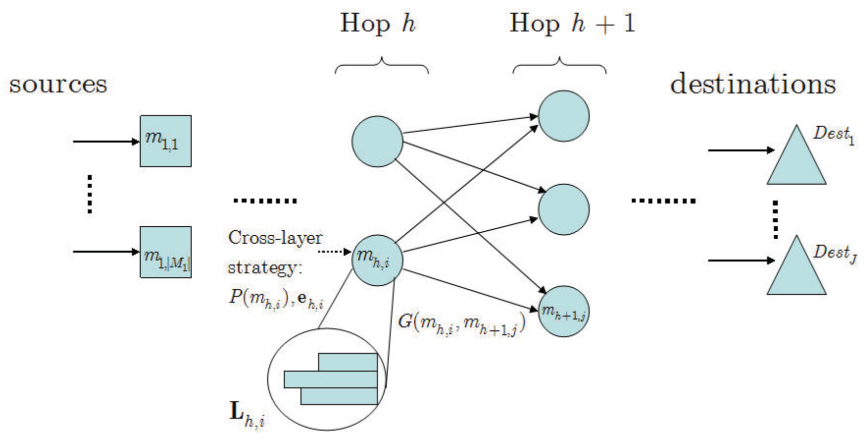


Fig. 1. A multi-hop wireless network with time-varying channels and delay-sensitive source packets.

each time slot[2]. We assume that, in every time slot, each node $m_{h,i}$ can transmit a data packet to one of the nodes in the $(h+1)$-th hop, using a certain transmission power. In this paper we assume that nodes at different hops transmit in orthogonal channels, and hence the error probability of the link between the two nodes $m_{h,i}$ and $m_{h+1,j}$ does not only depend on the power used by the transmitter of $m_{h,i}$, but also it depends on the interferences from the other nodes in the h-th hop. We assume that every node maintains several transmission queues for its received packets with different remaining lifetimes (i.e. time until delay deadline expires). Therefore, data packets may suffer from additional delays due to the congestion experienced at a certain node [8]. Our objective is to design an algorithm which enables the autonomous nodes to cooperatively maximize the performance of the delay-sensitive applications by acting solely based on their available information.

1) Physical layer model: We define a $1 \times |M_h|$ vector $\mathbf{P}_h^t$ to be the vector of transmission powers from the h-th hop to the $(h+1)$-th hop at time t, with its i-th entry being the transmission power of node $m_{h,i}$, i.e. $\mathbf{P}_h^t(i) = P^t(m_{h,i})$. We also define a $|M_h| \times |M_{h+1}|$ channel-state matrix $\mathbf{G}_h^t$ from the h-th hop to the $(h+1)$-th hop at time t as $\mathbf{G}_h^t = G^t(m_{h,i}, m_{h+1,j})$, where $G^t(m_{h,i}, m_{h+1,j})$ is the propagation gain of the channel from node $m_{h,i}$ to node $m_{h+1,j}$ at time t. We further define a $1 \times |M_h|$ noise vector $\mathbf{n}_h = (n_{h,1}, \ldots, n_{h,|M_h|})$ as the noise power levels at the receivers of the h-th hop.

After all the nodes make their decisions on transmission power, the resulting erasure probability of each link can be determined based on $\mathbf{P}_h^t$, $\mathbf{G}_h^t$ and $\mathbf{n}_k$. Assuming $n_{h,i}$ is white Gaussian noise for any $1 \leq h \leq H$ and $1 \leq i \leq |M_h|$, and that all the interferences are treated as noise (similar to [3][13]), the Signal-to-Interference Noise Ratio (SINR) of the link from node $m_{h,i}$ to node $m_{h+1,j}$ equals

$$\begin{aligned}&\mathrm{SINR}^t(m_{h,i}, m_{h+1,j})\\ &= 10\log_{10}\left(\frac{G^t(m_{h,i},m_{h+1,j})P^t(m_{h,i})}{n_{h+1,j}+\sum_{k\in M_h, k\neq i} G^t(m_{h,k},m_{h+1,j})P^t(m_{h,k})}\right)\mathrm{dB}\end{aligned} \tag{1}$$

[2]The time-slotted system is often assumed in the literature [10][11][12]. The length of one time-slot can be determined based on how fast the environment changes. For example, in the simulation we set the length of time slot as 1.0ms.

Each link between two neighboring nodes can be modeled as an erasure channel, with erasure probability [14]

$$p_e^t(m_{h,i}, m_{h+1,j}) = \frac{1}{1+e^{\xi(\mathrm{SINR}^t(m_{h,i},m_{h+1,j})-\delta)}} \tag{2}$$

The parameters of ξ and δ are determined by the coding and modulation schemes at PHY layer [14], and are known to the nodes.

2) Network layer model: We use a $1 \times |M_{h+1}|$ incidence vector $\mathbf{e}_{h,i}^t$ to represent node $m_{h,i}$'s routing decision at time t , i.e. $\mathbf{e}_{h,i}^t(j) = 1$ if node $m_{h,i}$ selects $m_{h+1,j}$ as its relay at time t, and $\mathbf{e}_{h,i}^t(j) = 0$ otherwise. The packets at the nodes are transmitted according to the routing decisions, and received with the corresponding probability, i.e. $1 - p_e^t(m_{h,i}, m_{h+1,j})$ for the erasure channel from node $m_{h,i}$ to node $m_{h+1,j}$. For every time slot, each node can only transmit one packet at a time from its transmission queue. Once a packet is transmitted, it leaves the transmission queue, and we assume that there is no retransmission for the lost packets. A packet is lost with probability $p_e^t(m_{h,i}, m_{h+1,j})$ on the link from node $m_{h,i}$ to node $m_{h+1,j}$ as given by Eq.(2). Upon receiving a packet, the receiving node will look in the RTP header of the packet to determine its remaining lifetime (the header contains the decoding or playout time of the packet), which is then used to determine whether the packet can still get to its destination before its deadline expires. If the packet cannot be delivered before its deadline, because its remaining lifetime D_{remain} is less than the minimum time in which it can reach its destination, it will be dropped; otherwise, the receiver node will put this packet to the end of the transmission queue, which consists of all the packets with the same remaining lifetime D_{remain} . If the length of one time slot is T_{slot}sec, then the minimum (i.e. the best case) delay can be achieved if the packet does not have any waiting delay after the current node. Hence, it can be computed as $\left((H-h) + \sum_{t=D_{min}}^{D_{remain}} L(t)\right) T_{slot}$sec for the h-th hop, which consists of transmission delays for the next $(H-h)$ hops which is T_{slot} times the number of remaining hops, and waiting delay at the current node, which is T_{slot} times the number of packets in the current queue that have remaining lifetimes not larger than D_{remain}, i.e. $\sum_{t=D_{min}}^{D_{remain}} L(t)T_{slot}$ with $L(t)$ being the size of the queue consisting of all the packets with remaining lifetime t . After every time slot, the remaining lifetimes of all the packets in the network are reduced by T_{slot}. The overall utility of the network when supporting delay-sensitive applications is defined as the number of packets received by the destinations within their delay deadlines [8], i.e. $U_{NET} = \sum_{j=1}^{J} N_j P(D_j < d_j)$, where N_j is the number of packets received by the destination node $Dest_j$, D_j is the random delay experienced by all the packets received at $Dest_j$, and d_j is the delay deadline of the video sequence with $Dest_j$ as its destination.

B. An MDP formulation

MDP has been used to solve various wireless networking problems, where the decision maker needs to capture the network dynamics, and take into account the effect of its current action on the future performance, e.g. in [15][16][17][18]. In this paper, we also use MDP to model and capture the network dynamics (time-varying channels and data sources). An MDP can be defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R)$, with $\mathcal{S}$ as its state space, $\mathcal{A}$ as its action space, $P(s'|s,a)$ as the state transition probability and $R(s,a)$ as its reward. We will discuss the formulation in detail in this subsection.

1) States: We define the state of the MDP to include both the dynamics in the wireless channel and also the current queue sizes at every node. Hence, the state at time slot t is $s^t = (\mathbf{G}_1^t, \dots, \mathbf{G}_H^t, \mathbf{L}_1^t, \dots, \mathbf{L}_H^t)$, in which $\mathbf{L}_h^t = \{L_{h,i}^t\}_{i \in M_h}$ is the set of the current queue sizes at the nodes in the h-th hop at time t. We assume that each node maintains several transmission queues, each of which containing packets with the same remaining lifetime. The sizes of these queues are represented by $L_{h,i} = [L_{h,i}(t_{h,min}), \dots, L_{h,i}(t_{h,max})]^T$ in which $L_{h,i}(t) = [L_{h,i}(t, m_{H,1}), \dots, L_{h,i}(t, m_{H,|M_h|})]$ is a vector of the queue sizes with $L_{h,i}(t, m_{H,k})$ representing the size of the queue for destination $m_{H,k}$ that contains packets with remaining lifetime t. If the time slot of the network is T_{slot}sec/slot, then for a node at the h-th hop, the minimum remaining lifetime is $t_{h,min} = (H+1-h)T_{slot}$, and the maximum lifetime is $t_{max} = D_{max} - (h-1)T_{slot})$. We let $\mathcal{S}$ denote the set of all the possible states. To simplify the notation, we also define $s_L^t = (\mathbf{L}_1^t, \dots, \mathbf{L}_H^t)$ as all the current queue states and $s_G^t = (\mathbf{G}_1^t, \dots, \mathbf{G}_H^t)$ as all the current channel states. We note that a centralized controller is needed to collect all the information across the network to have the full knowledge of the current state. We define $s_h^t = (\mathbf{G}_h^t, \mathbf{L}_h^t)$ as the state of the nodes at the h-th hop, which is locally available to the nodes at this hop. We assume the wireless channel gains can be modeled by independent Markov chains as in [19][20][21], i.e.

$$\begin{aligned} &P\left(s_G^{t+1} = (\mathbf{G}_1^{'}, \dots, \mathbf{G}_H^{'}) | s_G^t = (\mathbf{G}_1, \dots, \mathbf{G}_H)\right) \\ &= \prod_{h=1}^{H} P(\mathbf{G}_h^{'} | \mathbf{G}_h) \end{aligned} \tag{3}$$

Note though that these probability distributions of channel gains may be unknown to the network or nodes

2) Actions: We consider the relay selection together with the power levels which each node uses for transmission to the selected relay as the actions of the MDP. Therefore, the action represents both the routing and power control decisions. Note that at each node, we deploy a delay-driven scheduling, where the packet with the earliest delay deadline (i.e. least lifetime) is always transmitted first. Since every node is only transmitting to its selected relay, if node $m_{h,i}$ uses a power of $P_{h,i}^t$ to transmit at time t and makes routing decision $\mathbf{e}_{h,i}^t$, we can define the action of the the node $m_{h,i}$ as $\mathbf{a}_{h,i}^t = \left(\mathbf{P}_h^t(i), \mathbf{e}_{h,i}^t\right)$, which includes both the power levels used for transmission and routing decision of the node. We also define $\mathbf{a}_h^t = \left(\mathbf{a}_{h,1}^t, \dots, \mathbf{a}_{h,|M_h|}^t\right)$ as the action of the h-th hop. The action of the whole network is denoted by $\mathbf{a}^t = (\mathbf{a}_1^t, \dots, \mathbf{a}_H^t)$. We let $\mathcal{A}$ be the set of all the possible actions of the network, i.e. $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_H$, with $\mathcal{A}_h$ being the set of possible actions for the h-th hop.

3) State transition: We now discuss why the state transition of the network will satisfy the Markovian property, i.e. $P(s^{t+1}|s^t, \dots, s^0, \mathbf{a}^t) = P(s^{t+1}|s^t, \mathbf{a}^t)$. First, we note that the channel states are Markovian by the assumption of channel model in Eq.(3), and are not affected by the chosen action (transmission power or relay selection), i.e.$P(s^{t+1}|s^t, \dots, s^0, \mathbf{a}^t) = P(s_G^{t+1}|s_G^t) P(s_L^{t+1}|s^t, \dots, s^0, \mathbf{a}^t)$.Secondly, given the current state and action, the probability of queue sizes at the next time slot is fully determined. This is because the current channel state and action will determine all the erasure probabilities of the links between neighbors by Eq.(2). Therefore, the p.d.f. of the queue size at $(t+1)$ is a function of the queue sizes at time t, action of time t and also channel state at time t, i.e. $P(s_L^{t+1}|s^t, \dots, s^0, \mathbf{a}^t) = P(\mathbf{L}_1^{t+1}, \dots, \mathbf{L}_H^t | s^t, \dots, s^0, \mathbf{a}^t) = P(s_L^{t+1}|s^t, \mathbf{a}^t)$. Finally, by combining these two equations, we conclude that the sequence of states can be represented as a controllable Markov process:

$$P(s^{t+1}|s^t, \mathbf{a}^t) = P(s_G^{t+1}|s_G^t) P(s_L^{t+1}|s^t, \mathbf{a}^t) \tag{4}$$

4) Rewards: The reward function is a mapping of $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$. Based on the network utility U_{NET} in SectionII-A, we define the expected number of received packets at the destinations within a certain delay deadline as the reward of the MDP, i.e. $R(s^t, \mathbf{a}^t) = \sum_{i \in M_H} \left[1 - p_e^t(m_{H,i}, \omega(\mathbf{s}_{H,i}^t))\right]$. In the reward function, $\omega(\mathbf{s}_{H,i}^t)$ is a scheduling scheme which maps $m_{H,i}$'s state to a destination node, i.e. finding the destination node of the packet being transmitted, because for the node in the last hop the relay selection is determined by the specific destination of the packet to be transmitted. We note that any packet which has expired will be dropped once it is received by the H-th hop and thus, the throughput at the destination will be the same as the goodput. It is also easy to verify that the reward is only a function of the state at the H-th hop and its action, i.e. $R(s^t, \mathbf{a}^t) = R(s_H^t, a_H^t)$. The solution to an MDP is a policy π, which is a mapping $\pi: \mathcal{S} \to \mathcal{A}$ with $\pi(\mathbf{s})$ being the optimal action taken in state $\mathbf{s}$. An optimal policy π^* is the policy that maximizes the expected discounted reward, i.e. $\pi^* = \arg\max_{\pi} \mathrm{E}_{P(s'|s,\mathbf{a})}\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}^t, \pi(\mathbf{s}^t))\right]$. We note that, as a special case, if $\gamma = 0$, it becomes an optimization problem of maximizing the immediate reward at the current state,

$$\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} R(\mathbf{s}, \mathbf{a}) \tag{5}$$

which is used in these centralized optimization approaches by assuming a static network environment.

III. Factorization of The MDP

A. *Factorization of the state transition probability*

We note that generally the state transition probability of an MDP can be represented by a $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$ table [22], in which $|\mathcal{A}|$ and $|\mathcal{S}|$ are the numbers of actions and states of the MDP, respectively. For a multi-hop network with H hops, if there are N_A actions and N_S states for each hop, then $|\mathcal{A}| = N_A^H$ and $|\mathcal{S}| = N_S^H$. This exponential complexity makes it computationally intractable to obtain the optimal policy using a centralized algorithm. However, the hop-by-hop structure of the network can be applied to derive a more compact representation of the state transition probability by factorizing it into local state transition probabilities at different nodes, which can further lead to an efficient distributed computation of the optimal policy. Moreover, we can develop distributed algorithms to approach the optimal policy, which can avoid the large delay and high communication overhead caused by the acquisition of global information. We will first present the factorization of the state transition probability in this subsection and subsequently, based on this, provide a distributed and autonomic computation of the optimal policy in the next subsection.

Based on the network state transition probability in Eq.(4), and our assumption about the channels in Eq. (3), we can use the chain rule of conditional probability to rewrite the state transition for the queue size as

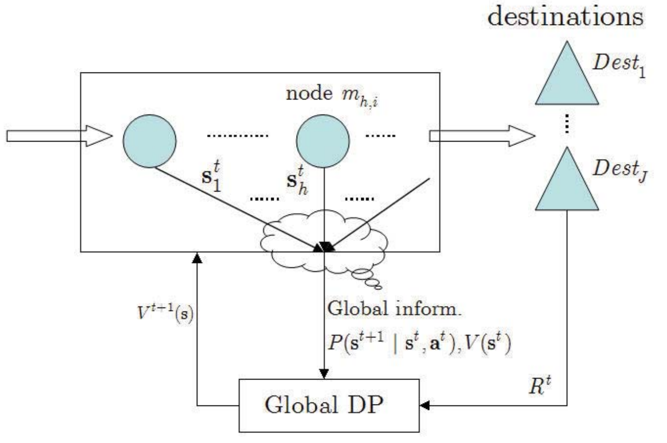
$$\begin{aligned}
&P(s_L^{t+1}|s^t,\mathbf{a}^t)\\
&= P(\mathbf{L}_1^{t+1},\ldots,\mathbf{L}_H^{t+1}|s^t,\ldots,s^0,\mathbf{a}^t)\\
&= P(\mathbf{L}_1^{t+1}|s^t,\mathbf{a}^t)\prod_{h=2}^{H} P(\mathbf{L}_h^{t+1}|s^t,\mathbf{a}^t,\mathbf{L}_1^t,\ldots,\mathbf{L}_{h-1}^t)\\
&= P(\mathbf{L}_1^{t+1}|\mathbf{L}_1^t)\prod_{h=2}^{H} P(\mathbf{L}_h^{t+1}|\mathbf{G}_{h-1}^t,\mathbf{a}_{h-1}^t,\mathbf{L}_{h-1}^t,\mathbf{L}_h^t)
\end{aligned} \tag{6}$$

where $P(\mathbf{L}_1^{t+1}|\mathbf{L}_1^t)$ characterizes the arrival processes at the source nodes of the network. In this case, the state transition at the nodes in the h-th hop is independent from the state transitions at the other hops, and it only depends on the channel states of the $(h-1)$-th hop and the actions of the nodes in the $(h-1)$-th hop, because they together determine the probability of the number of arrival packets at the nodes in the h-th hop. Therefore, the network state transition probability can be factorized as

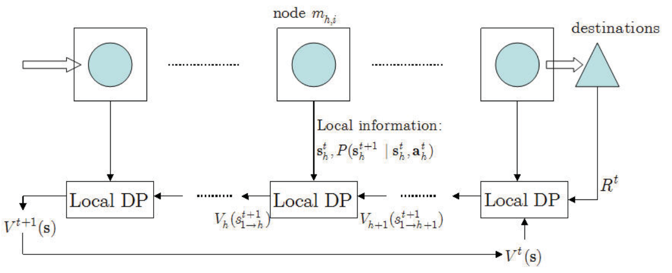
$$\begin{aligned}
&P(\mathbf{s}^{t+1}|\mathbf{s}^t,\mathbf{a}^t)\\
&= P(\mathbf{s}_G^{t+1}|\mathbf{s}_G^t)P(\mathbf{s}_L^{t+1}|\mathbf{s}^t,\mathbf{a}^t)\\
&= \left[\prod_{h=1}^{H} P(\mathbf{G}_h^{'}|\mathbf{G}_h)\right] P(\mathbf{L}_1^{t+1}|\mathbf{L}_1^t)\\
&\times \prod_{h=2}^{H} P(\mathbf{L}_h^{t+1}|\mathbf{G}_{h-1}^t,\mathbf{a}_{h-1}^t,\mathbf{L}_{h-1}^t,\mathbf{L}_h^t)\\
&= P(s_1^{t+1}|s_1^t)\prod_{h=2}^{H} P(s_h^{t+1}|s_h^t,a_{h-1}^t,s_{h-1}^t)
\end{aligned} \tag{7}$$

B. *Distributed computation of the value function*

Given the state transition probability, there are different approaches to compute the optimal policy, such as value iteration and policy iteration [23]. In value iteration, a state-value function $V(\mathbf{s})$ is defined as the expected accumulated discounted reward when starting with state $\mathbf{s}$, i.e. $V(\mathbf{s}) = \mathrm{E}_\pi\left(\sum_{t=0}^{\infty}\gamma^t R(\mathbf{s}_t,,\pi(\mathbf{s}_t))|\mathbf{s}_0=\mathbf{s}\right)$.



(a) A centralized value-function update using global information and centralized DP



(b) A distributed value-function update using local information and local DP

Fig. 2. The comparison of centralized and distributed value-function updates for finding the optimal policy in the dynamic multi-hop network

Since we cannot know the future rewards at current time, the idea of value iteration is to use the current value function $V^t(\mathbf{s})$ as an estimation of future rewards. Hence, the value function can be calculated by iteratively solving the following centralized dynamic programming (DP) problem:

$$V^{t+1}(\mathbf{s}) = \max_{\mathbf{a}\in\mathcal{A}}\left\{R(\mathbf{s},\mathbf{a}) + \gamma\sum_{\mathbf{s}^{'}\in\mathcal{S}} P(\mathbf{s}^{'}|\mathbf{s},\mathbf{a})V^t(\mathbf{s}^{'})\right\} \tag{8}$$

We note that computing the state-value function as in Eq.(8) has two main disadvantages. First, the computing entity needs to know the state transition probability for the entire network, which is distributed across the network, and will require substantial communication overheads to collect them. Secondly, the computation complexity for the summation over all the possible next state, i.e. $\sum_{\mathbf{s}^{'}\in\mathcal{S}} P(\mathbf{s}^{'}|\mathbf{s},\mathbf{a})V^t(\mathbf{s}^{'})$, is very large, because it will be in the size of the network state space. If every hop has N_S possible states, then the network will have a total of N_S^H states, thereby making the computation complexity prohibitive. We illustrated this centralized value iteration using global information and centralized DP in Fig. 2(a). The factorization of the transition probability enables us to explore more computationally efficient methods, which only rely on the local state transition probabilities and thus, enable an autonomous network optimization. Rather than computing the actions for nodes in different hops simultaneously, this DP operator can be solved in a hop-by-hop manner. We first rewrite the DP operator in the state-value update as

$$R(\mathbf{s}_H,\mathbf{a}_H) + \gamma \sum_{\mathbf{s}_1',\ldots,\mathbf{s}_H'} P(\mathbf{s}_1'|\mathbf{s}_1)\prod_{h=2}^{H} P(\mathbf{s}_h'|\mathbf{s}_h,\mathbf{a}_{h-1},\mathbf{s}_h)V^t(\mathbf{s}_1',\ldots,\mathbf{s}_H') \tag{9}$$

Since the action at the last hop does not affect future network states, it is chosen only to maximize the current reward, i.e. $\mathbf{a}_H = \arg\max_{\mathbf{a}} R(\mathbf{s}_H,\mathbf{a})$ and $V_H(\mathbf{s}_1',\ldots,\mathbf{s}_H') = V(\mathbf{s}_1',\ldots,\mathbf{s}_H') + \max_{\mathbf{a}} R(\mathbf{s}_H,\mathbf{a})$. For the nodes in the h-th hop ($1 \le h < H$), a state-value function for the previous hops is computed by solving a local DP problem:

(Note that for simplicity we will use $V_h(\mathbf{s}_1',\ldots,\mathbf{s}_h')$ instead of $V_h(\mathbf{s}_1,\ldots,\mathbf{s}_H,\mathbf{s}_1',\ldots,\mathbf{s}_h')$ in the following parts of this paper.)

$$\begin{aligned}&V_h(\mathbf{s}_1',\ldots,\mathbf{s}_h')\\&=\max_{\mathbf{a}_h}\left[\sum_{\mathbf{s}_{h+1}'} P(\mathbf{s}_{h+1}'|\mathbf{s}_{h+1},\mathbf{a}_h,\mathbf{s}_h)V_{h+1}(\mathbf{s}_1',\ldots,\mathbf{s}_{h+1}')\right]\end{aligned} \tag{10}$$

and this value is passed to the previous hop, as illustrated in Fig. 2(b).

Next, we present the following theorem, which shows that the distributed value-iteration yields a performance which is at least as good as that of optimal policy obtained by the centralized value-iteration.

Theorem 1: The state-value function associated with the optimal policy from the distributed value-iteration is not less than the value-function of the optimal policy from centralized value-iteration.

Proof: For the h-th hop ($1 \le h < H$), the value-function is computed as in Eq.(10) by solving a local DP. If we let $\mathbf{a}^* = \{\mathbf{a}_1^*,\ldots,\mathbf{a}_H^*\}$ be the optimal action obtained by the centralized DP for the current state, then

$$\begin{aligned}&V_h(\mathbf{s}_1',\ldots,\mathbf{s}_h')\\&=\max_{\mathbf{a}_h}\left[\sum_{\mathbf{s}_{h+1}'} P(\mathbf{s}_{h+1}'|\mathbf{s}_{h+1},\mathbf{a}_h,\mathbf{s}_h)V_{h+1}(\mathbf{s}_1',\ldots,\mathbf{s}_{h+1}')\right]\\&\ge \sum_{\mathbf{s}_{h+1}'} P(\mathbf{s}_{h+1}'|\mathbf{s}_{h+1},\mathbf{a}_h^*,\mathbf{s}_h)V_{h+1}(\mathbf{s}_1',\ldots,\mathbf{s}_{h+1}')\end{aligned}$$

Actually the solution of the local DP gives a randomized action rather than a deterministic one, because the optimal action to the problem Eq.(10) depends on the future states, which are unknown at the current stage. ■

The different optimization criterions and computation complexities of three different formulations discussed are summarized in Table I. We also illustrate the message exchange for the exact distributed value-function update and compare it with a centralized value-function update approach in Fig. 2.

IV. Learning the Optimal Policy with Less Information Exchange

A. Why online learning?

In the previous section, we discussed how both centralized and distributed DP can find the optimal policy when the dynamics are known. However, in real wireless environment, the global state transition probability is usually not known by the distributed nodes. Moreover, if a centralized controller is implemented to collect all the information, it will bring both high communication overhead and large delay, and cannot support delay-critical applications well. Hence, a learning method is required to update the value-function online, and adapt the cross-layer transmission strategies on-the-fly.

B. Actor-critic learning

During an online adaptation process, the learning algorithm first chooses its action according to the current state and value-function, and then the network transits to the next state and receives the immediate reward. Suppose we have such a centralized learning node with the knowledge of the current states at all the nodes across the entire network and the value-function for every state, and it can also control the actions of all the nodes. Then, we can define the temporal-difference [22] as:

$$\delta^t(\mathbf{s}^t) = \left[R(\mathbf{s}^t,\mathbf{a}^t) + \gamma V^t(\mathbf{s}^{t+1})\right] - V^t(\mathbf{s}^t) \tag{11}$$

where $\delta^t(\mathbf{s}^t)$ is the difference between current and previous estimation of the value function. The value-function is then updated by

$$V^{t+1}(\mathbf{s}^t) = V^t(\mathbf{s}^t) + \kappa^t\delta^t(\mathbf{s}^t) \tag{12}$$

where κ^t is a positive learning rate.

After the value-function is obtained, policy update can be performed in various ways. We will use the actor-critic (AC) method [24][25] throughout this paper. The AC learning separates the value-function update and policy update. The value-function (i.e. the *critic*), is used to strengthen or weaken the tendency of choosing a certain action. The policy structure, or the *actor*, is a function of state-action pair, i.e. $\rho(\mathbf{s},\mathbf{a})$, which indicates the tendency of choosing action $\mathbf{a}$ at state $\mathbf{s}$. After each update of value-function, $\rho(\mathbf{s},\mathbf{a})$ is updated by $\rho(\mathbf{s}^t,\mathbf{a}^t) = \rho(\mathbf{s}^t,\mathbf{a}^t) + \beta^t\delta^t(\mathbf{s}^t)$, where β^t is the learning rate for policy update at time t. Then, the action is generated by $\pi(\mathbf{s}^t,\mathbf{a}) = \frac{e^{\rho(\mathbf{s}^t,\mathbf{a})}}{\sum_{\mathbf{a}'\in\mathcal{A}} e^{\rho(\mathbf{s}^t,\mathbf{a}')}}$, where $\pi(\mathbf{s}^t,\mathbf{a})$ is the probability distribution of choosing a certain action at state $\mathbf{s}^t$, i.e. the randomized policy at state $\mathbf{s}^t$, which gives a higher probability for choosing an action with a larger tendency ρ . Since the policy update remains the same for various learning algorithms we propose, we will omit this part due to limited space. The centralized AC learning is illustrated in Fig. 3. However, such a centralized learning node cannot exist in a distributed multi-hop network because it requires the full knowledge of the states at each node, which requires a large communication overhead for acquiring this information. Moreover, such distributed information involves transmission delays which are not acceptable for delay-sensitive applications [27], which makes the learning by a centralized learner not practical for on-line adaptation. Alternatively, we can use the learning method based on our distributed value-function update introduced in Section III.B, and the temporal difference for the nodes in the h-th hop becomes

$$\delta_h^t(\mathbf{s}^t) = \begin{cases} V_2^t(s_1^t,s_2^t) - V^t(\mathbf{s}^t), & h=1\\ R(\mathbf{s}^t,\mathbf{a}^t) + \gamma V^t(\mathbf{s}^{t+1}) - V_H^t(s_1^t,\ldots,s_H^t), & h=H\\ V_{h+1}^t(s_1^{t+1},\ldots,s_{h+1}^{t+1}) - V_h^t(s_1^t,\ldots,s_h^t), & \text{else}\end{cases} \tag{13}$$

and update the value-function as $V_h^t(s_1^t,\ldots,s_h^t) = (1-\kappa_h^t)V_h^t(s_1^t,\ldots,s_h^t) + \kappa_h^t\delta_h^t(s_1^t,\ldots,s_h^t)$.

TABLE I
DIFFERENT PROBLEM FORMULATIONS, THEIR OPTIMIZATION CRITERIA, AND COMPUTATION COMPLEXITIES

Formulation	Optimization criterion	Complexity
Centralized without MDP	$\max_{\mathbf{a}} R(\mathbf{s},\mathbf{a})$	$(N_S N_A)^H$
Centralized MDP	$\max_{\pi} \mathrm{E}_{P(s'\|s,\mathbf{a})}\left(\sum_{t=0}^{\infty}\gamma^t R(\mathbf{s}^t,\pi(\mathbf{s}^t))\right)$	$(N_S^2 N_A)^H$
Distributed MDP	$\max_{\pi} \mathrm{E}_{P(s'\|s,\mathbf{a})}\left(\sum_{t=0}^{\infty}\gamma^t R(\mathbf{s}^t,\pi(\mathbf{s}^t))\right)$	$\sum_{h=1}^{H}(N_S^h)^2 N_A$

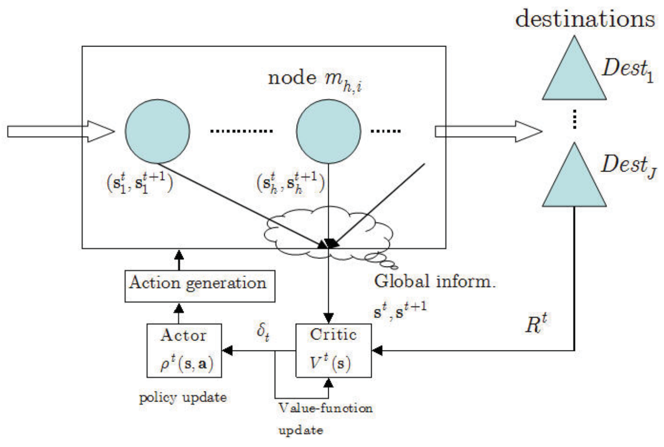


Fig. 3. The centralized actor-critic learner with complete information

C. The trade-off between performance and information exchange overhead

In a network without an independent control channel to exchange information, a large overhead of information exchange will decrease the throughput of data packets, because a certain amount of time slots or frequency bands need to be used for transmitting this information. On the other hand, if the information exchange overhead is reduced, the final value-function that is learned will be suboptimal. Hence, there is a trade-off between the resulting network performance and information exchange overhead. We quantify it as the *performance impact under information exchange constraint*:

$$J(N_{\mathcal{I}},T_{\mathcal{I}}) = \eta(N_{\mathcal{I}},T_{\mathcal{I}}) \max_{\pi_{N_{\mathcal{I}},T_{\mathcal{I}}}} \sum_{t=0}^{\infty}\gamma^t R(\mathbf{s}^t,\pi_{N_{\mathcal{I}},T_{\mathcal{I}}}(\mathbf{s}^t)) \tag{14}$$

If the information exchange is carried out every $T_{\mathcal{I}}$ time slots, with an average information size of $N_{\mathcal{I}}$ bits for each time slot, then we denote $\pi_{N_{\mathcal{I}},T_{\mathcal{I}}}$ as the policy that can be learned under the information exchange constraint of $N_{\mathcal{I}}$ and $T_{\mathcal{I}}$. The effective data ratio of a certain information exchange scheme is $\eta(N_{\mathcal{I}},T_{\mathcal{I}})$, which quantifies the percentage of transmitted data packets in all the transmitted packets, which includes both data packets and packets carrying feedback information [29]. It can be computed as

$$\eta(N_{\mathcal{I}},T_{\mathcal{I}}) = \frac{T_{\mathcal{I}}}{T_{\mathcal{I}} + \lceil \frac{N_{\mathcal{I}}T_{\mathcal{I}}}{L_{packet}} \rceil} \tag{15}$$

where L_{packet} is the size (in bits) of the packet (we assume the data packets and the feedback packets are the same size), and $\lceil \frac{N_{\mathcal{I}}T_{\mathcal{I}}}{L_{packet}} \rceil$ ($\lceil x \rceil = \mathrm{int}(x)+1$) is the minimal number of packets that are needed for feedback.

By increasing the amount of information exchange $N_{\mathcal{I}}$ or decreasing the exchange interval $T_{\mathcal{I}}$, the accumulated reward will be increased because a better policy can be learned; but the overhead is also increased which may decrease the overall performance. Hence, a good trade-off needs to be found for various network setting. Next, we will discuss various methods to reduce the information overhead by either reducing $N_{\mathcal{I}}$ or increasing $T_{\mathcal{I}}$, and evaluate their performances using the metric of Eq.(14) in Section IV.C.

D. An approximate value-function update with less information exchange

We first focus on reducing the information exchange overhead by reducing $N_{\mathcal{I}}$. To learn the exact value-function, the nodes in the h-th hop requires the information with size of N_s^h, which can be very large for the nodes near the destination. Instead of learning the exact value-function, we can let every node to only learn an approximate value function, with a much smaller state space. We can consider the approximate value function as a *function approximation* of the exact value function $V(\mathbf{s})$. Function approximation [24] uses a set of features instead of the complete description of network state to approximate the value function. If we define $V_{h,a}(\mathbf{s}_{\mathcal{F}(h)})$ as the approximate value-function of the nodes in the h-th hop, and $\mathcal{F}(h)$ as the set of hops whose states are used in the approximation, then the size of information exchange for the h-th hop is $N_s^{|\mathcal{F}(h)|}$. The temporal-difference of the nodes in the h-th hop can be computed as:

$$\delta_{h,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t) = \begin{cases} R(s_H^t,a_H^t)+\gamma V_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^{t+1}) - V_{H,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t), & h=H \\ \sum_{k\in\Theta(h)}\theta_{h,k}V_{k,a}^t(\mathbf{s}_{\mathcal{F}(k)}^{t+1}) - V_{h,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t), & h<H \end{cases} \tag{16}$$

and the approximate value-function is updated by

$$V_{h,a}^{t+1}(\mathbf{s}_{\mathcal{F}(h)}^t) = (1-\kappa_h^t)V_{h,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t) + \kappa_h^t\delta_{h,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t) \tag{17}$$

We define $\Theta(h)$ to be the set of nodes that can communicate their approximate value functions to the nodes in the h-th hop. $\theta_{h,k}$ is the weight nodes in the h-th hop uses for the k-th hop's approximate value-function, which must satisfy $\sum_{k\in\Theta(h)}\theta_{h,k}=1$ and $0\le\theta_{h,k}\le 1$.

We summarize four different formulations of our problem, their corresponding value-functions and also computation complexities in Table II. The centralized solution without MDP formulation (Eq.(5)) does not consider the network dynamics and only takes myopic actions. Therefore, it requires

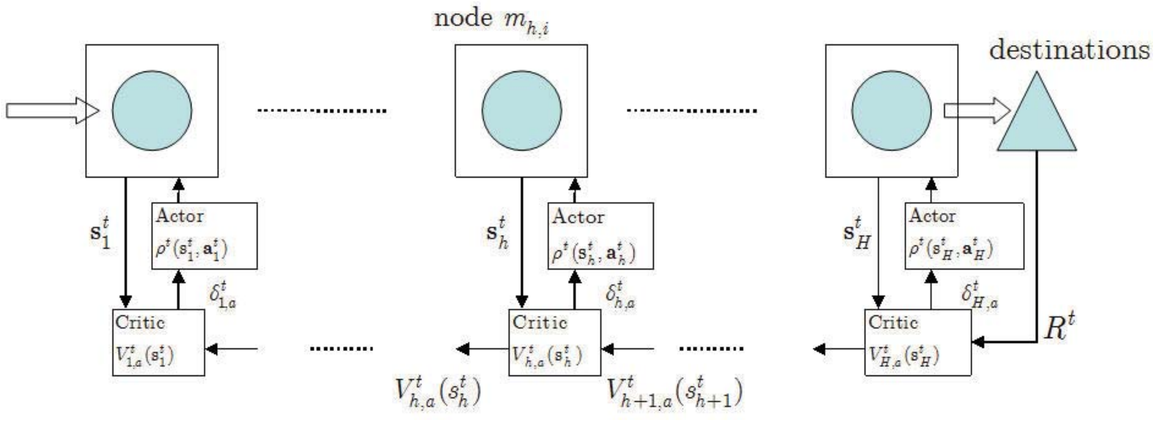

node $m_{h,i}$
destinations
$\mathbf{s}_1^t$
Actor $\rho^t(\mathbf{s}_1^t,\mathbf{a}_1^t)$
$\delta_{1,a}^t$
Critic $V_{1,a}^t(\mathbf{s}_1^t)$
$\mathbf{s}_h^t$
Actor $\rho^t(\mathbf{s}_h^t,\mathbf{a}_h^t)$
$\delta_{h,a}^t$
Critic $V_{h,a}^t(\mathbf{s}_h^t)$
$V_{h,a}^t(s_h^t)$
$V_{h+1,a}^t(s_{h+1}^t)$
$\mathbf{s}_H^t$
Actor $\rho^t(\mathbf{s}_H^t,\mathbf{a}_H^t)$
$\delta_{H,a}^t$
Critic $V_{H,a}^t(\mathbf{s}_H^t)$
R^t

Fig. 4. The distributed approximate actor-critic learning, with $\mathcal{F}(h) = \{h\}$ and $\Theta(h) = \{h+1\}$.

TABLE II
THE VALUE-FUNCTIONS AND COMPUTATION COMPLEXITIES UNDER DIFFERENT FORMULATION

Formulation	Value-function	Complexity
Centralized without MDP	$V(\mathbf{s}) = \max_{\mathbf{a}} R(\mathbf{s},\mathbf{a})$	$(N_S N_A)^H$
Centralized MDP	$V(\mathbf{s}) = \mathrm{E}_\pi\left(\sum_{t=0}^{\infty}\gamma^t R(\mathbf{s}^t,\pi(\mathbf{s}^t))\vert\mathbf{s}_0=\mathbf{s}\right)$	$(N_S^2 N_A)^H$
Distributed MDP	$V_H(\mathbf{s}_1,\ldots,\mathbf{s}_H) = R(\mathbf{s}_H,a_H) + \gamma V(\mathbf{s}_1,\ldots,\mathbf{s}_H)$ $V_h(\mathbf{s}_1,\ldots,\mathbf{s}_h) = \max_{a_h}\left[\sum_{\mathbf{s}_{h+1}} P(\mathbf{s}_{h+1}\vert\mathbf{s}_{h+1},\mathbf{s}_h,a_h)V_h(\mathbf{s}_1,\ldots,\mathbf{s}_{h+1})\right]$	$\sum_{h=1}^{H}(N_S^h)^2 N_A$
Approximate MDP	$V_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^t) = R(\mathbf{s}_H^t,a_H^t) + \gamma V_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^{t+1})$ $V_{h,a}^t(\mathbf{s}_{\mathcal{F}(H)}^t) = \sum_{k\in\Theta(h)}\theta_{h,k}V_{k,a}^t(\mathbf{s}_{\mathcal{F}(k)}^{t+1})$	$\sum_{h=1}^{H}(N_S^{\vert\mathcal{F}(h)\vert})^2 N_A$

a much lower complexity than the centralized MDP approach. The distributed MDP formulation reduces the complexity of the centralized MDP by exploring the factorization of the state transition probability. The complexity is further reduced in the approximated MDP approach, because only a small number of states are used to approximate the exact value-function. The approximate distributed actor-critic learning is illustrated in Fig. 4. A detailed comparison between the centralized actor-critic learning and the approximate distributed learning is presented in Table III.

E. Learning with less frequent feedback

So far we focused on how to reduce the information exchange overhead by reducing the size of exchanged information, i.e. reducing $N_{\mathcal{I}}$. Based on Eq.(14) we can also reduce this overhead by increasing $T_{\mathcal{I}}$. An important limitation of the learning method is that it requires information feedback at the same frequency as the decision making, i.e. the approximate value functions from other nodes need to be updated every time slot. In real multi-hop networks, where an independent feedback channel rarely exists, such high frequency feedbacks will consume a substantial amount of network resources (power, frequency bands, time slots, etc), hence will both decrease the transmission rate and increase delay. Many wireless network protocols consider this tradeoff, and use less frequent feedback schemes. For example, in IEEE 802.11e [28], an option of block acknowledgment (ACK) [29] can improve the quality-of-service for video and audio applications by sending a group of acknowledgments together. We can use a similar idea to reduce the amount of information exchange in our learning method, by lowering the frequency of exchanging approximate value functions.

We consider a transmission protocol in which the information exchange is carried out every $T_{\mathcal{I}}$ time slots[8]. This means that each node does not receive feedback immediately from other nodes about their current approximate value function, until every $T_{\mathcal{I}}$ time slots. For any state visited at time t between two successive feedbacks, i.e. $MT_{\mathcal{I}} \le t < (M+1)T_{\mathcal{I}}$,$M \in \mathbb{Z}_+$, using a similar idea of the n-step TD learning method, we can compute the $n(t)$-step temporal difference ($n(t) = (M+1)T_{\mathcal{I}} - t$) as:

$$\begin{aligned}\delta_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^t) = \quad &(1-\lambda)\sum_{t'=t}^{(M+1)T_{\mathcal{I}}}\lambda^{(M+1)T_{\mathcal{I}}-t'}R_H(\mathbf{s}_{\mathcal{F}(H)}^{t'}) \\ &+\gamma V_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^{(M+1)T_{\mathcal{I}}}) - V_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^t)\end{aligned} \tag{18}$$

when $h = H$, and

$$\begin{aligned}\delta_{H,a}^t(\mathbf{s}_{\mathcal{F}(H)}^t) = \quad &(1-\lambda)\sum_{k\in\Theta(h)}\sum_{t'=t}^{(M+1)T_{\mathcal{I}}}\lambda^{(M+1)T_{\mathcal{I}}-t'} \\ &V_{k,a}^t(\mathbf{s}_{\mathcal{F}(k)}^{t'}) - V_{h,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t)\end{aligned} \tag{19}$$

when $h < H$, with the parameter λ controlling how the value-functions decay with time. Then, the value function update is performed as $V_h^a(\mathbf{s}_{\mathcal{F}(h)}^t) = (1-\kappa)V_h^a(\mathbf{s}_{\mathcal{F}(h)}^t) + \kappa\delta_{h,a}^t(\mathbf{s}_{\mathcal{F}(h)}^t)$.

We note that Eq.(18) is not exactly the same as the standard

TABLE III
THE COMPARISON OF COMPLEXITY OF CENTRALIZED AND DISTRIBUTED ACTOR-CRITIC LEARNING

Value-function update	**Communication overhead**	**Computation complexity**	**Storage space**
Centralized actor-critic	$O(N_A^H)$	Actor : $O(N_A^H)$ Critic : $O(1)$	Value − function : $O(N_S^H)$ Tendency : $O\left((N_S N_A)^H\right)$
Approx. distributed actor-critic	$O(H)$	Actor : $O(N_A H)$ Critic : $O(1)$	Value − function : $O(HN_S^{\lvert\mathcal{F}(h)\rvert})$ Tendency : $O\left(HN_S^{\lvert\mathcal{F}(h)\rvert}N_A\right)$

n-step TD learning in [24], because unlike the standard n-step TD learning , in which $n(t) \equiv n$ for any t, in our method every state is looking ahead with different steps, i.e. $n(t)$ can be different for different t. For example, for the state visited just after a feedback, it will use all the $T_\mathcal{I}$ rewards until the next feedback to compute the temporal difference, and hence it is a $T_\mathcal{I}$-step TD estimation; for the state visited just before a feedback, it only has its immediate reward when the feedback is carried out, and hence it is a 1-step TD estimation. However, this non-uniform $n(t)$-step TD learning still has the following error-reduction property which is similar to the standard n-step TD learning.

Theorem 2: For any MDP with a given policy π and the current estimated value function V, using the non-uniform $n(t)$-step TD learning is guaranteed to reduce the worst case estimation error, i.e. $\max_{s,t} |E_\pi[R_t^{n(t)}|s_t = s] - V^\pi(s)| \leq \gamma \max_s |V(s) - V^\pi(s)|$. ($V^\pi(s)$ is the true value-function under policy π, and $R_t^{n(t)} = (1-\lambda)\sum_{t'=t}^{t+n(t)} \lambda^{t+n(t)-t'} R(t')$ is the discounted accumulated reward in the duration from t to $t+n(t)$.)

Proof: The error-reduction property of standard n-step TD learning [24] guarantees that for any n, $\max_s |E_\pi(R_t^n|s_t = s) - V^\pi(s)| \leq \gamma^n \max_s |V(s) - V^\pi(s)|$, which means the n-step look-ahead reward gives a smaller worst-case error in the estimation of the true value-function. Hence, for any t we have $\max_s |E_\pi(R_t^{n(t)}|s_t = s) - V^\pi(s)| \leq \gamma^{n(t)} \max_s |V(s) - V^\pi(s)|$. Then, taking the maximum over t and s gives $\max_{s,t} |E_\pi(R_t^{n(t)}|s_t = s) - V^\pi(s)| \leq \max_t |\gamma^{n(t)} \max |V(s) - V^\pi(s)|| = \gamma \max_s |V(s) - V^\pi(s)|$. ■

Even when using the $n(t)$-step TD-learning in a distributed and approximate way, we can still conjecture based on this theorem that using a less frequent feedback does not necessary degrade the performance. Moreover, after considering the information feedback overhead, which is quantified as $\eta(N_\mathcal{I}, T_\mathcal{I})$ in Section IV.C, a lower feedback frequency can improve the network performance. This will also be verified by our simulation results in Section V. Algorithm 1 presents the entire information exchange and decision making process (for one node).

Algorithm 1 Using -step TD-learning to reduce the information exchange overhead

```
if $t = kT_\mathcal{I}$ then
    1.1 receive the information feedback, i.e. the approximate value function
    1.2 use the $n$-step TD-learning Eq.(18) to update the approximate value function
    1.3 send the approximate value function and reward information to all the nodes requesting information feedback (this feedback contains the information for the previous $T_\mathcal{I}$ slots.)
else
    2.1 observe the current local state $\mathbf{s}_{\mathcal{F}_h}$, make the estimation of , and take the action given by the current approximate value function
    2.2 record both the experienced state and reward for future feedback to other nodes
```

V. SIMULATION RESULTS

A. Simulation setting

We consider a scenario that two users are transmitting two video sequences to the same destination, through a H-hop wireless network as in Fig. 5. The video sequences, each of which is with 16 frames per GOP, frame rate 30HZ in CIF format, are compressed and transmitted with fixed-length packets. The packets have different delay deadlines, ranging from D_{min} to D_{max}. The arrival processes at the two source nodes are *i.i.d.* with the following probability distribution (we use $d = 0$ for the case of no arrival packet)[8]:

$$P(s(m_{1,j}) = d) = \begin{cases} \rho(d), & d \in [D_{min}, D_{max}] \\ 1 - \sum_{d'=D_{min}}^{D_{max}} \rho(d'), & d = 0 \end{cases}$$

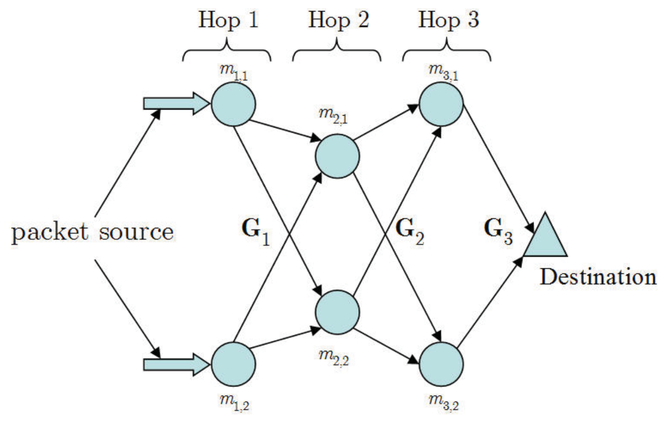
At the physical layer, we assume the channel states at each hop, represented by the channel state matrix $\mathbf{G}_h$, $(1 \leq h \leq H)$, are modeled as independent Markov chains. Each channel state matrix describes the interferences between two adjacent hops, i.e.

$$\mathbf{G}_h = \begin{cases} \begin{pmatrix} G(m_{h,1}, m_{h+1,1}), & G(m_{h,1}, m_{h+1,2}) \\ G(m_{h,2}, m_{h+1,1}), & G(m_{h,2}, m_{h+1,2}) \end{pmatrix}, & h < H \\ \begin{pmatrix} G(m_{h,1}, m_{Dest}) \\ G(m_{h,2}, m_{Dest}) \end{pmatrix}, & h = H \end{cases}$$

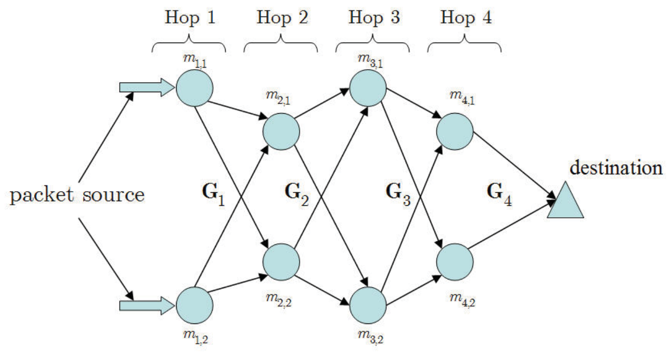
To reduce the number of states in the simulation, we consider each channel state matrix is an independent two-state Markov chain with the state transition probability

$$P(\mathbf{G}_h^{t+1}|\mathbf{G}_h^t) = \begin{cases} 0.8, & \mathbf{G}_h^{t+1} = \mathbf{G}_h^t \\ 0.2, & \mathbf{G}_h^{t+1} \neq \mathbf{G}_h^t \end{cases} \tag{20}$$

The two possible states for $\mathbf{G}_h$ for $1 \leq h < H$ are $\begin{pmatrix} 5, & 1 \\ 2, & 6 \end{pmatrix}$ and $\begin{pmatrix} 1, & 5 \\ 6, & 2 \end{pmatrix}$, and they are $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ for $h = H$. Each node has three possible power levels, which are 0, 10mW and 20mW. The noise power at each node is 1mW. At the network layer, although data packets may get lost during transmission (see Section II.A), we assume that all the feedback packets are not subject to any transmission error.



(a) 1



(b) 2

Fig. 5. Two multi-hop networks, with two source nodes and one destination node.

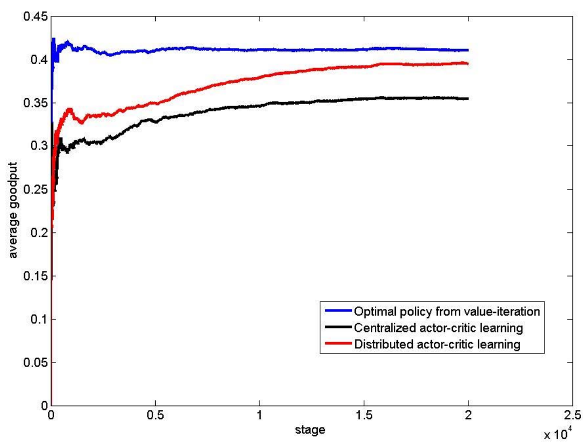


Fig. 6. Average reward from the centralized learning and the distributed learning, and compared with the average reward from the optimal policy.

B. The distributed approximate learning and the exact centralized learning methods

We compare the average reward (i.e. the goodput at the destination) obtained from the optimal policy, which is derived using value-iteration (Eq.(8)), the centralized actor-critic learning (Eq.(11), (12)), and the distributed approximate actor-critic learning (Eq.(16), (17)) for a 3-hop network (i.e. $H = 3$) as in Fig. 5(a). In the distributed approximate learning, we only use the local state, i.e. $\mathcal{F}_h = \{h\}$ for $h = 1, 2, 3$. Due to the huge complexity of value-iteration and centralized learning, which is exponential in the number of hops, we will use a simple setting for this comparison. We assume the maximum queue size at each node is 1. All the packets arriving at the source nodes have delay deadlines of $3T_{slot}$, and the arrival process at each source node has probability distribution $\rho(3T_{slot}) = 0.6$, $\rho(0) = 0.4$. Therefore, for each node there are 2 possible states and the total number of states of the network will be 2^9 (2^6 different queue sizes and 2^3 different channel states). We note that even with this simple setting, the complexity of value-iteration is about $2^9 \times 2^9 \times 4^4 \times 2^2 \approx 2 \times 10^8$.

As shown in Fig. 6, the average reward of the optimal policy serves as an upper-bound for different learning algorithms. The distributed approximate actor-critic learning not only outperforms the centralized learning algorithm, but also approaches the performance bound set by the optimal policy. The advantage of the distributed learning over the centralized one may be surprising at a first sight. However, this result has a simple explanation: in the distributed approximate learning, each actor-critic pair is solving a problem with a much smaller state and action space than the centralized one, and hence, different state-action pairs are visited more frequently and the optimal action can be found much faster. The results show that the distributed approximate actor-critic learning algorithm achieves a good trade-off between performance and complexity, even if each node only uses the local state of its own hop to approximate the value-function.

C. The learning algorithm with less frequent information feedback

With a lower feedback frequency, we have applied the $n(t)$-step TD estimation to update the temporal-difference in Eq.(18) and use this temporal-difference for actor-critic learning. We evaluate the performances (in terms of average goodput per stage but without considering the impact of information overhead) of distributed actor-critic learning with different feedback frequencies ($T_{\mathcal{I}} = 1, 10, 20$ respectively) in the 3-hop network as Fig. 5(a), and the results are shown in Fig. 7. With the n-step TD estimation, although the feedback frequency is reduced by about 90%, the average reward is almost the same or even slightly better compared with the case of $T_{\mathcal{I}} = 1$. This verifies our conjecture that the error-reduction property of $n(t)$-step learning guarantees the performance even when the learning process is distributed.

Next, we compare the performances after taking into account the information overhead with different feedback frequencies. The average reward from the previous simulation will be multiplied by the effective data ratio $\eta(N_{\mathcal{I}}, T_{\mathcal{I}})$. To compute $\eta(N_{\mathcal{I}}, T_{\mathcal{I}})$, we assume $N_{\mathcal{I}} = 30$ bits and $L_{packet} = 1024$ bits, then $\eta(N_{\mathcal{I}}, T_{\mathcal{I}})$ is computed according to Eq.(15). The average goodput after considering the information overhead under different feedback frequencies are compared in Fig. 8, which shows the advantage of using a lower feedback frequency.

In Table IV, we also compare the performances of distributed actor-critic learning with different feedback frequencies by evaluating the quality of received video packets

TABLE IV
THE PSNR FOR VIDEO SEQUENCES, BY USING DIFFERENT FEEDBACK FREQUENCIES

# of hops		Video Sequences					
		Coastguard			Foreman		
3-hop network	Information feedback frequency $(1/\mathcal{T}_I)$	1	$\frac{1}{10}$	$\frac{1}{20}$	1	$\frac{1}{10}$	$\frac{1}{20}$
	Goodput with inform. overhead (packets/time-slot)	0.197	0.357	0.378	0.197	0.357	0.378
	Average queue size/max queue size	0.68	0.49	0.47	0.68	0.49	0.47
	Effective data rate (Kbps)	201	365	388	201	365	388
	PSNR(dB)	27.5	30.0	30.3	30.0	33.1	33.9
4-hop network	Information feedback frequency $(1/\mathcal{T}_I)$	1	$\frac{1}{10}$	$\frac{1}{20}$	1	$\frac{1}{10}$	$\frac{1}{20}$
	Goodput with inform. overhead (packets/time-slot)	0.153	0.278	0.321	0.153	0.278	0.321
	Average queue size/max queue size	0.72	0.55	0.51	0.72	0.55	0.51
	Effective data rate (Kbps)	157	285	329	157	285	329
	PSNR(dB)	26.6	39.4	29.5	28.9	32.3	32.6

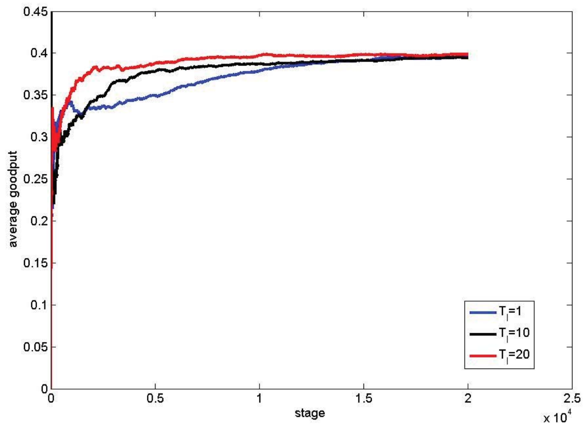


Fig. 7. Average reward (without accounting for information overhead) of the distributed actor-critic learning, with different feedback frequencies ($T_{\mathcal{I}} = 1, 10, 20$, respectively)

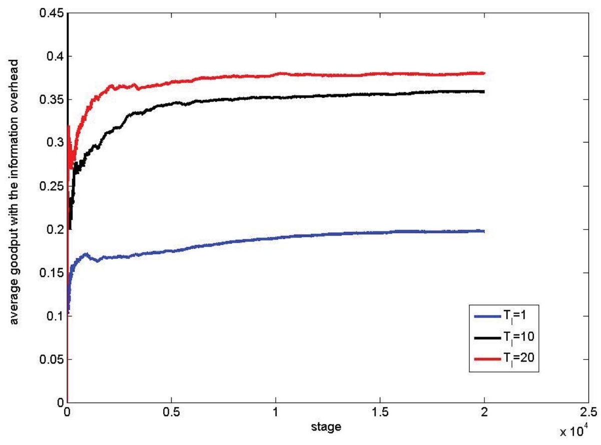


Fig. 8. Average goodput after accounting for information overhead, with different feedback frequencies

(measured by PSNR) after the learning process has converged. The time-slot is assumed to be 1.0ms. We compare their performances in both a 3-hop network as in Fig. 5(a), and also a 4-hop network as in Fig. 5(b). The 4-hop network is constructed by adding a two-node hop in the 3-hop network, and the channels are assumed to be the same as in Eq.(19) and (20).

VI. CONCLUSION

In this paper, we formulated the joint routing and power control problem in wireless multi-hop networks as a Markov Decision Process. Based on the factorization of the state transition probability, we derive a distributed computation method for finding the optimal policy. In order to reduce both the communication overhead and delay incurred due to the inter-node information exchanges, we proposed an on-line learning method which enables the nodes to autonomously learn the optimal policy. Moreover, when the network protocol allows for block acknowledgments to be deployed, an $n(t)$-step learning method is proposed to further reduce the information exchange overhead and improve the network performance.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments which helped us to improve this paper.

REFERENCES

[1] Y. Li and A. Ephremides, “A joint scheduling, power control, and routing algorithm for ad hoc wireless networks,” *Ad Hoc Networks*, vol. 5, no. 7, pp. 959-973, Sep. 2007.
[2] R. Agarwal and A. Goldsmith, “Joint rate allocation and routing for multi-hop wireless networks with delay-constrained data,” Technical Report, Wireless Systems Lab., Stanford University, CA, USA, 2004.
[3] R. Cruz and A. Santhanam, “Optimal routing, link scheduling, and power control in multi-hop wireless networks,” in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 702-711.
[4] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: a reinforcement learning approach,” *Advances in Neural Information Processing Systems*, vol. 6, 1994.
[5] J. Dowling, E. Curran, R. Cunningham, and V. Cahill, “Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing,” *IEEE Trans. Syst., Man, Cybern., Part A*, vol. 35, no. 3, pp. 360-372, May 2005.
[6] E. Forster and A. Murphy, “A feedback-enhanced learning approach for routing in WSN,” in *4th WMAN*, 2007.
[7] P. Stone, “Tpot-RL applied to network routing,” in *Proc. 17th ICML*, San Francisco, CA, 2000.

[8] H. Shiang and M. van der Schaar, "Multi-user video streaming over multi-hop wireless networks: a distributed, cross-layer approach based on priority queuing," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 770-785, May 2007.

[9] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: of theory and practice," in *Proc. 22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, Boston, MA, USA, July 2003.

[10] X. Wang, Q. Liu, and G. B. Giannakis, "Analyzing and optimizing adaptive modulation coding jointly with ARQ for QoS-guaranteed traffic," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, Mar. 2007.

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

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

[13] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89-103, Jan. 2005.

[14] D. Krishnaswamy, "Network-assisted link adaptation with power control and channel reassignment in wireless networks," in *Proc. 3G Wireless Conf.*, 2002, pp. 165-170.

[15] Q. Zhao and A. Swami, "A decision-theoretic framework for opportunistic spectrum access," *IEEE Wireless Commun. Mag.*, vol. 14, no. 4, pp. 14-20, Aug. 2007.

[16] S. Singh, V. Krishnamurthy, and H. V. Poor, "Integrated voice/data call admission control for wireless DS-CDMA systems," *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1483-1495, June 2002.

[17] J. Razavilar, K. Liu, and S. Marcus, "Optimal rate control in wireless networks with fading channels," in *Proc. IEEE 49th VTC*, pp. 807-811, 1999.

[18] E. Altman, "Applications of Markov decision processes in communication networks: a survey," report RR-3984, Aug. 2000, INRIA.

[19] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. Commun.*, vol. 47, no. 11, pp. 1688-1692, Nov. 1999.

[20] Q. Zhang, W. Zhu, and Y. Zhang, "End-to-end QoS for video delivery over wireless Internet," *Proc. IEEE*, vol. 93, no. 1, pp. 123-134, Jan. 2005.

[21] Q. Liu, S. Zhou, and G. B. Giannakis, "Queuing with adaptive modulation and coding over wireless links: cross-layer analysis and design," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 1142-1153, May 2005.

[22] L. Kaelbling, M. Littman, and A. Moore. "Reinforcement learning: a survey," *J. Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[23] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd edition. Athena Scientific, 2000.

[24] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[25] A. Barto, R. Sutton, and C. Anderson, "Neuron-like elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, pp. 835-846, 1983.

[26] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.

[27] H.-P. Shiang and M. van der Schaar, "Queuing-based dynamic channel selection for heterogeneous multimedia applications over cognitive radio networks," *IEEE Trans. Multimedia*, vol. 10, no. 5, pp. 896-909, Aug. 2008.

[28] IEEE Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment: Medium Access Control (MAC) Quality of Service (QoS) Enhancements, IEEE Draft Amendment IEEE P802.11e/D13.0, Jan. 2005.

[29] O. Cabral, A. Segarra, and F. Velez, "Implementation of IEEE 802.11e block acknowledgement policies based on the buffer size," *14th European Wireless Conference*, 2008, pp. 1-7.

Zhichu Lin received his Bachelor and Master degrees both in Electronic Engineering from Tsinghua University, Beijing, China. He has been a graduate student in Electrical Engineering Department at UCLA since 2007.

Mihaela van der Schaar is Professor in the Electrical Engineering Department at University of California, Los Angeles. She received in 2004 the NSF Career Award, in 2005 the Best Paper Award from IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, in 2006 the Okawa Foundation Award, in 2005, 2007 and 2008 the IBM Faculty Award, and in 2006 the Most Cited Paper Award from *EURASIP: Image Communications* journal. She was an associate editor for IEEE TRANSACTIONS ON MULTIMEDIA, IEEE SIGNAL PROCESSING LETTERS, *Circuits and Systems for Video Technology*, *IEEE Signal Processing Magazine*, etc. She also holds 33 granted US patents and three ISO awards for her contributions to the MPEG video compression and streaming international standardization activities. Her research interests are in multi-user communication networks, multimedia communications, processing and systems, online learning, network economics and game theory. She is an IEEE Fellow.