Joint Network Coding and Scheduling for Media Streaming Over Multiuser Wireless Networks

Dong Nguyen, Thinh Nguyen, *Member, IEEE*, and Xue Yang

***Abstract*—We formulate the problem of network-coding (NC)-based scheduling for media transmission to multiple users over a wireless-local-area-network-like or WiMAX-like network as a Markov decision process (MDP). NC is used to minimize the packet losses that resulted from unreliable wireless channel conditions, whereas the MDP is employed to find the optimal policy for transmissions of unequally important media packets. Based on this, a dynamic programming technique is used to give an optimal transmission policy. However, this dynamic programming technique quickly leads to computational intractability, even for scenarios with a moderate number of receivers. To address this problem, we further propose a *simulation-based* dynamic programming algorithm that has a much lower run time yet empirically converges quickly to the optimal solution.**

***Index Terms*—Markov decision process (MDP), media streaming, network coding (NC), packet scheduling, WiMAX.**

I. Introduction

ALTHOUGH there has been a tremendous growth in multimedia applications over the Internet, the packet loss, delay, and time-varying bandwidth of the Internet have hindered many high-quality multimedia applications. These problems manifest more so in wireless networks, which often exhibit higher loss rates and lower bandwidth. Many *above-network-layer* approaches to multimedia streaming over the Internet and wireless networks have been proposed to deal with packet loss, delay, and time-varying bandwidth, ranging from transport protocols and packet-scheduling algorithms [1], [2] to source and channel coding techniques [3], [4]. A number of these techniques are based on the differentiated principle in which data of various importance levels are treated differently under resource constraints. Notably, scalable video coding techniques produce a layered compressed video bit stream that consists of a base layer and several enhancement layers. The base layer contributes the most to the visual quality of a video, whereas the enhancement layers provide successive quality refinements [5]. As such, using a scalable video bit stream, the sender is able to adapt a video bit rate to the current available network bandwidth by sending the base layer and an appropriate number of enhancement layers [5], [6].

That said, for a number of video-streaming applications, their bandwidth requirements are sufficiently small that, even without employing sophisticated techniques, a few of these applications can concurrently run over the existing wireless standards [i.e., IEEE 802.11(b) and (g)]. On the other hand, these standards may not be able to support multimedia applications with much larger bandwidth requirements, e.g., high-definition quality video-streaming applications. In the near future, Internet Protocol television and Video-on-Demand applications will rely on wireless networks to deliver high-quality video from the Internet to any TV set or home computer through a wireless access point or base station. Therefore, it is imperative that an efficient bandwidth-sharing/competing scheme among the wireless applications be employed to satisfy the bandwidth and delay requirements of each application.

Parallel to the advances of wireless technologies is the recent development of the network coding (NC) paradigm, which allows a source to efficiently disseminate information to multiple destinations in a given network topology. In a traditional *forward-and-store* network, packets are forwarded hop by hop, unmodified from the source to the destination. On the other hand, NC techniques allow an intermediate node to combine the data from different input links before sending the combined data on its output links. For many problems such as multicast and broadcast, using appropriate encoding schemes at each intermediate nodes (typically linear combination of input data) can achieve the network capacity. Although the original NC problem is formulated in the context of a wire-line network, it has also been used to reduce the energy consumption and to increase the capacity of wireless ad hoc networks. For example, in [7], Fragouli *et al.* provided an overview of NC and its applications in wireless networks. Wu *et al.* also showed how NC can be used to improve the capacity of information exchange in a wireless ad hoc network [8].

This paper proposes a new NC technique to improve the overall bandwidth efficiency while optimizing multiple concurrent multimedia applications with heterogeneous requirements in a wireless access network. The contributions of this paper include the following: 1) a framework for increasing the bandwidth efficiency of broadcast and unicast sessions in a wireless network based on NC techniques and 2) optimized scheduling algorithms based on the Markov decision process (MDP) to maximize the quality of multimedia applications. We

Manuscript received April 22, 2010; revised September 27, 2010 and December 6, 2010; accepted January 10, 2011. Date of publication February 10, 2011; date of current version March 21, 2011. This work was supported in part by the National Science Foundation under Grant 0845476 and Grant 0834775. The review of this paper was coordinated by Dr. L. Cai.

D. Nguyen is with FPT University, 844 Hanoi, Vietnam (e-mail: dongnv2@fpt.edu.vn).

T. Nguyen is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331-5501 USA.

X. Yang is with the Intel Labs, Intel Corporation, Santa Clara, CA 95054 USA (e-mail: xue.yang@intel.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2011.2112677

first provide a few preliminaries for media streaming, MDP, and NC for wireless networks in Section II. In Section III, we describe a basic NC-based retransmission scheme that improves the bandwidth efficiency of broadcast and unicast sessions in a one-hop wireless network. Next, we present the proposed NC-based scheduling policy using MDP that optimizes multiple concurrent flows under bandwidth and delay constraints. In Section IV, we demonstrate the proposed simulation-based dynamic algorithm as a viable solution for large MDPs. Section V shows how our simulation-based algorithm is used to solve the scheduling problem for the case of erroneous feedback. Simulation results and discussions are provided in Section VI. Finally, we conclude with a few remarks in Section VII.

II. Preliminaries

We first present a brief introduction to multimedia streaming, MDP, and NC for wireless media transmission.

A. Multimedia Streaming

Many approaches to multimedia streaming have been proposed, ranging from network protocols to source and channel coding techniques. From the channel coding perspective, forward-error-correction techniques have been proposed to increase reliability at the expense of bandwidth expansion [3], [9]–[11]. From the source coding perspective, error-resilient coding techniques have been explored to allow the quality of a video to be gracefully degraded in lossy environments [6], [12], [13]. In addition, layered video-coding techniques have been proposed to deal with the heterogeneity and time-varying nature of the Internet by adapting its bit rate to the available bandwidth [5], [14], [15].

Based on the unequal contributions of different video bits, the rate-distortion MDP-based optimization approach to packet scheduling has produced many fruitful results in the past several years [1], [16], [17]. The main idea of this approach is that, using the observations at every single step, the scheduling algorithm chooses the best action to perform (e.g., whether to send a packet or not and which packet to send) to maximize the *expected* video quality under limited network resources. The optimal sequence of actions during a time duration of interest is the solution to the MDP problem, which can be efficiently solved in many settings.

B. MDP

Let us consider a decision maker or a controller who, at every time step, is in charge of making a decision or choosing an action, which can influence the evolution of a probabilistic system. Assuming that the state of the system evolves in discrete time steps, then the goal of the controller is to choose a sequence of actions that maximizes some cumulative system performance metrics (rewards) at the end of some finite or infinite number of time steps. Since the system states and the performance metrics depend on the chosen action at every time step, it is wise for the controller to consider the future states and the associated rewards in the decision-making process at the present state. Finding the optimal sequence of actions is the solution to the MDP problem.

An abstract MDP represents a dynamic system and is specified by a finite set of states S representing the possible states of the system, a set of control actions A, a transition probability P, and a reward function r. The transition probability specifies the dynamics of the system and gives the probability $p(s'|s,a)$ of transitioning to state s' after taking action a in state s. The dynamics are Markovian in the sense that the probability of the next state s' depends only on the current state s and action a and not on any previous history. The reward function assigns a real number to the current state s and the action a taken in that state so that $r(s,a)$ represents the immediate reward of being in state s and taking action a. A policy π is a mapping from states to actions, which defines a controller that takes actions as specified by the policy. We assume that time is discrete and that the control policy selects one action at each time step. Every policy π is associated with a value function V^π such that $V^\pi(s)$ gives the expected cumulative reward achieved by π when starting in state s. The solution to an MDP problem is an optimal policy that maximizes the expected cumulative reward over any finite or infinite number of time steps.

When an MDP ends in a finite number of time steps N, we call it a finite-horizon MDP. Let d_t denote a decision rule, prescribing a procedure for action selection in each state at a specified time step t. In other words, the decision rules are functions $d_t : S \to A$, which specify the choice of action when the system occupies state s at time step t. For each $s \in S$, $d_t(s) = a_t \in A$. A policy $\pi = (d_1, d_2, d_3, \ldots, d_N)$ is a sequence of actions at every time step.

Let U_t^π denote the total expected reward obtained by using policy π from the time t, $t+1, \ldots, N-1$. Thus, for $t < N$, we have

$$U_t^\pi(s_t) = E_{s_t}^\pi \left\{ \sum_{n=t}^{N-1} r_n(s_n, a_n) + r_N(s_N) \right\}. \tag{1}$$

Now, one can compute $U_1^\pi(s)$ using the following recursive equation:

$$\begin{aligned} U_t^\pi(s_t) &= r_t(s_t, a_t) + E_{s_t}^\pi \left\{ U_{t+1}^\pi(s_{t+1}) \right\} \\ &= r_t(s_t, a_t) + \sum_{j \in S} p(j|s_t, a_t) U_{t+1}^\pi(j). \end{aligned} \tag{2}$$

where $p(j|s_t, a_t)$ is the probability of transiting from state s_t to state j when taking action a_t.

Based on (2), it can be shown that the optimal policy $\pi^* = (d^*(s_1), d^*(s_2), \ldots, d^*(s_N))$ can be solved using the backward induction algorithm (BIA) [18] to produce the maximum final cumulative reward

$$\pi^* = \arg\max_{a \in A} E_{s_t}^\pi \left\{ \sum_{n=t}^{N-1} r_n(a_n, a_n) + r_N(s_N) \right\}. \tag{3}$$

The BIA

1) $t = N$, and $U_N^*(s_N) = 0$ for all $s_N \in S$.

2) Substitute $t-1$ for t, and compute $U_t^*(s_t)$ for each $s_t \in S$ by

$$U_t^*(s_t) = \max_{a \in A} \left\{ r_t(s_t, a) + \sum_{j \in S} p(j|s_t, a) U_{t+1}^*(j) \right\} \quad (4)$$

$$d^*(s_t) = \arg\max_{a \in A} \left\{ r_t(s_t, a) + \sum_{j \in S} p(j|s_t, a) U_{t+1}^*(j) \right\}. \quad (5)$$

3) If $t = 1$, stop. Otherwise, return to step 2.

We note that solving a typical MDP problem involves two tasks: 1) modeling and 2) selection of solution tools. In the modeling task, a particular real-world problem is translated into an abstract MDP problem. This involves modeling the states, the actions, the immediate rewards, the transition probabilities, and the desired objective.[1] This modeling process can be hard and often requires domain experts. In other cases, accurately representing the system states may require a large state and action spaces, making it hard to solve a large MDP in practice. Thus, approximate algorithms are typically used to solve large MDP problems in a reasonable amount of time [18], [19].

C. NC for Wireless Media Transmissions

The original NC problem is first studied by Ahlswede *et al.* [20], which shows that the throughput of multicast networks can be significantly improved by appropriate mixing of data at the intermediate network nodes. Chachulski *et al.* shows in a canonical work [21] that NC not only helps improve the multicast throughput but avoids complicated routing and scheduling as well, particularly in wireless ad-hoc or sensor networks. Many other works also exploit these characteristics of NC [8], [22]–[26]. NC can also be considered as a general form of erasure-correcting coding [27], [28].

Recently, NC for wireless media transmissions has been studied [29]–[32]. The main approach of NC for wireless media transmission is not to *network code* every packets equally. Rather, NC is judiciously applied according to media packets of different importance levels and delay requirements. One approach is to use the MDP framework, as proposed by Nguyen *et al.* [29]. This approach describes a basic MDP framework for optimizing the video quality, taking into account the packet importance levels and the constraints on bandwidth and delay. The solution proposed in [29] is to use the classical BIA, which does not scale with large MDPs. In this paper, we propose a heuristic simulation basic dynamic programming algorithm for solving large MDPs. We also extend the MDP framework in [29] to consider the case of erroneous feedback, which was preliminarily studied in [30]. In addition, this paper provides the following: 1) simulation results for the convergence properties of the proposed heuristic; 2) more realistic simulation settings; and 3) a unified view of MDP solutions from its workshop publications [29], [30].

The most related work to ours is that of Seferoglu *et al.* [31] and its extended version in [32]. Both the work in [32] and ours aim to optimize the video quality via NC techniques. However, the differences lie in the formulation of the objective function, the network model, and the solution approach. In [32], the authors describe three algorithms: 1) Network coding for video (NCV); 2) NCVD; and 3) network coding rate distortion optimized (NC-RaDiO). Because NC-RaDiO is built on NCV and NCVD and is the best of the three, we will mainly discuss the differences between NC-RaDio and our work. First, NC-RaDiO explicitly optimizes the rate-distortion function with the incorporation of NC. The authors consider only *one* packet at a time, i.e., the proposed algorithm will choose the packet that minimizes the rate distortion function. In a sense, this is a greedy algorithm since it selects the packet that gives the most value at the present transmission opportunity without taking into account the future. In contrast, our MDP formulation is a sequential decision-making process in which the decision is made at every time step and takes into consideration the future actions to minimize the expected video distortion over a finite time horizon (number of transmission opportunities). In our approach, the rate is implicitly modeled in the constraint on the number of transmission opportunities. The second difference between NC-RaDiO and our work is the network or environmental modeling. RaDiO assumes a more sophisticated network model based on [10]. Specifically, the packet loss probability is a function of round trip time (RTT), which models the queuing delay in a multihop network due to congestion. For a single-hop wireless network, we argue that RTT is perhaps less of an indicator for packet loss, particularly, at the medium access control (MAC) layer, where every successfully transmitted MAC packet is accompanied by an ACK after a prespecified time. For that reason, we assume Bernoulli trial and Gilbert's models for packet losses. In addition, note that NC-RadiO attempts to solve a much harder problem due to the distributed setting, while our work takes a centralized approach. Finally, solution approaches taken by RaDiO and ours are quite different. RaDiO uses continuous optimization via the Lagrangian method, whereas ours uses discrete optimization via a combination of dynamic programming and simulation-based methods, which have been extensively studied in artificial intelligence and optimization communities [33], [34].

III. Wireless Streaming With Network Coding

A. Model and Assumption

We now describe the broadcast and unicast models in wireless local area network (WLAN)-like and Worldwide Interoperability for Microwave Access (WiMAX)-like networks. We show how NC and MDP can be used to increase the bandwidth efficiency while optimizing the concurrent applications based on their requirements. In particular, we are interested in designing a packet-scheduling algorithm running at a WLAN-like access point (AP) or WiMAX-like broadcast station that optimizes multiple concurrent wireless applications. Specifically, we present an optimized packet-scheduling algorithm

[1]The desired objective does not have to be the sum of all the immediate rewards. A popular reward is the discount reward, where the future reward weighs less than the current reward.

exclusively designed for video broadcast and unicast flows from the AP to one or more receivers. The objective of the algorithm is to maximize the visual quality of videos received at the receivers under certain bandwidth and delay constraints.

We make the following assumptions for our model.

1) There are M receivers $R_1, R_2, \ldots,$ and R_M.
2) The AP has a set $\Omega = \{l_1, l_2, \ldots, l_K\}$ of K packets to be delivered to the receivers after some time slots N. In a broadcast setting, all the receivers request all K packets, whereas in a unicast setting, each receiver requests a different subset of Ω. In a semibroadcast setting, there are two or more receivers requesting the same subset of Ω.
3) There is a limit on the total number of time slots N used to transmit these K packets. After N time slots, the AP moves to the next batch of K packets, regardless of whether all current K packets have been successfully received at the intended receivers.
4) Any receiver can cache packets transmitted from the AP to other receivers, even though those packets are not directly useful to themselves.
5) Data are divided into packets, and each is sent in a time slot of fixed duration.
6) The AP knows which packet from which receiver is lost. This can be accomplished through the use of positive or negative acknowledgments (ACK/NAKs).
7) The distribution of packet loss at a receiver R_i follows the Bernoulli distribution with parameter p_i. One can develop a more accurate model [35] and [36], although it will complicate the analysis.

B. Wireless Transmission With NC

Consider a broadcast scenario. Suppose two packets a and b are broadcast from an AP to two receivers R_1 and R_2. In an 802.11x network, if a packet is correctly received, the AP should receive an ACK within an appropriate amount of time after the data packet is sent. Otherwise, the data packet is considered lost and must be retransmitted. Using this scheme, a packet loss at any receiver will require the AP to retransmit that packet. If there are two distinct lost packets at two different receivers, the AP will need at least two retransmissions or a total of four transmissions to successfully transmit both packets a and b to receivers R_1 and R_2, as shown in Fig. 1(a). We now consider an NC technique that requires only one retransmission to recover two lost packets at both receivers. Using this NC scheme, the AP does not immediately retransmit the lost packet a at R_1. Instead, the AP continues to broadcast the next packet until there is a lost packet b at receiver R_2. At this time, the AP broadcasts the new packet $(a \oplus b)$ to both receivers. If R_1 has packet b but not a, and R_2 has packet a but not b, then both receivers will be able to reconstruct their missing packets by simply XOR-ing the packet they have, with the packet $(a \oplus b)$. As shown in Fig. 1(b), R_1 reconstructs a as $b \oplus (a \oplus b)$, and R_2 reconstructs b as $a \oplus (a \oplus b)$. Therefore, one retransmission from the AP will enable both receivers to correctly reconstruct their lost packets. This coding scheme is also considered as a class of maximum-distance-separable or digital fountain codes [37] and, in general, can substantially outperform the traditional retransmission scheme when the loss patterns among many receivers are uncorrelated.

Timeslot	1	2	3	4
Packet sent	a	b	a	b
R_1	x	o	o	o
R_2	o	x	o	o

(a)

Time slot	1	2	3
Packet sent	a	b	a⊕b
R_1	x	o	o
R_2	o	x	o

(b)

Fig. 1. (a) Traditional wireless transmission requiring a total of four transmissions to successfully transmit two packets to two receivers. (b) Wireless transmission with NC requiring only three transmissions.

This NC technique can be readily applied to the unicast setting. Assume that R_1 wants to receive packet a, whereas R_2 wants to receive packet b. Clearly, if R_1 is willing to cache packet b intended for R_2 and R_2 is willing to cache packet a intended for R_1, then the two unicast sessions are now equivalent to a single broadcast session in the previous example.

The key to improving bandwidth efficiency is an efficient generation of XOR packets to enable all the receivers to quickly recover their lost packets. If the packet loss rate is low, the AP has fewer opportunities to broadcast the XOR packets of distinct lost packets at different receivers; thus, there is not much benefits from using NC. In addition, for higher bandwidth efficiency, longer delay of some packets may be necessary to allow packet losses to occur at other receivers, leading to more opportunities for the AP to generate the XOR packets. However, this might not be acceptable for applications with strict playback deadline. Thus, the AP must consider the tradeoff between the delay and the bandwidth efficiency based on the application requirements.

C. Optimal-MDP-Based Packet Scheduling

We discuss the modeling of the set of states S, the set of actions A, the immediate reward $r(s_t, a_t)$, the transition probabilities $P(s_{t+1}|s_t, a_t)$, and the cumulative rewards.

Our packet-scheduling algorithm works as follows: At every time step, the AP sends a packet and waits for an ACK message. If a receiver receives a packet, an ACK is immediately sent back, similar to the 802.11x protocol. If no ACK is received within a specified time frame, the data packet is considered lost. The AP can then choose to send a new packet, retransmit a lost packet, or transmit an XOR packet. We now proceed to model our packet-scheduling algorithm as an MDP of finite horizon N, where N is the maximum number of allowable time slots to transmit K packets.

State Representation: At any given time slot, receiver R_i possesses a subset of packets that belonged to Ω, including the packets that are intended for other receivers. This subset can be represented by an K-bit vector as $(b_j^1, b_j^2, \ldots, b_j^K)$, where $b_j^i \in \{0, 1\}$. $b_j^i = 1$ indicates the presence of packet l_i

at R_j, whereas $b_j^i = 0$ indicates otherwise. Since there are M receivers, a system configuration or state s can be represented by an $M \times K$ matrix with binary entries as

$$s = \begin{bmatrix} b_1^1 & b_1^2 & \cdots & b_1^K \\ b_2^1 & b_2^2 & \cdots & b_2^K \\ \cdots & \cdots & \cdots & \cdots \\ b_M^1 & b_M^2 & \cdots & b_M^K \end{bmatrix}. \quad (6)$$

Thus, there are $2^{M \times K}$ possible states.

Action Representation: At any given time slot, the AP can perform the following: 1) Broadcast any $l_i \in \Omega$; 2) broadcast any XOR packet resulting from XOR-ing the distinct lost packets from different receivers; and 3) broadcast nothing. This implies that the number of possible actions J at any time step

$$J = K + \sum_{i=2}^{L} \binom{L}{i} + 1 \quad (7)$$

where L denotes the number of packets that are lost at one or more receivers. The maximum number of lost packets L is K; however, this case is extremely rare for a large K.

Transition Probability: Given the Bernoulli model with parameter p_i for packet loss at each receiver R_i, it is straightforward to compute the transition probability $P(s_{t+1} = s' | s_t = s, a_t = a)$. For example, consider broadcasting two packets to two receivers, i.e., $K = 2$, and $M = 2$. Let us denote

$$s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad s' = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Suppose that, at time t, the system is in state s, i.e., R_1 has packet l_1 and R_2 has packet l_2; then, choosing action $a =$ "send l_1" in state s will move the system to state s' with probability

$$P(s_{t+1} = s' | s_t = s, a_t = a) = 0 \quad (8)$$

whereas choosing action $a' =$ "send l_2" will move the system to state s' with probability

$$P(s_{t+1} = s' | s_t = s, a_t = a') = 1 - p_1. \quad (9)$$

Reward Modeling: The immediate reward $r(s, a)$ for each pair of a state and an action must be chosen such that the sum of these immediate rewards accurately models our objective. Since our objective is to optimize the quality of multimedia streaming applications, we model the immediate rewards as the sum of the reduction in distortion for one or more receivers upon receiving a particular packet. Thus, maximizing the overall reward is equivalent to minimizing the overall distortion for all the receivers' applications under some bandwidth and delay constraints. In our setting, we know the explicit reward amount $r(s', s)$ when the system moves from state s to state s'. For example, if state s indicates that a receiver has layers 1 and 2, and state s' indicates that a receiver has layers 1, 2, and 3, then moving from state s to state s' would reward us with an amount $r(s', s)$ equal to the distortion reduction contributed by layer 3. Since we know the transition probability between the states under an action a, we can compute $r(s, a)$ as the expected immediate reward by taking action a as

$$r(s, a) = \sum_{j \in S} P(j | s, a) r(j, s). \quad (10)$$

Example: We now present a simple example showing MDP formulations for broadcast and unicast settings with two receivers and two packets. For the state space, there would be a total of 16 states with each state s represented by

$$s = \begin{bmatrix} b_1^1 & b_1^2 \\ b_2^1 & b_2^2 \end{bmatrix}.$$

As for the action space, at any time step, the AP can perform one of four actions: 1) send l_1; 2) send l_2; 3) send $l_1 \oplus l_2$; and 4) send nothing.

As for the transition probabilities, let us denote p_1 and p_2 as the packet loss probabilities at R_1 and R_2, respectively. For each action, there is an associated transition probability matrix. We show two transition probability matrices due to taking actions "sending l_1" and "sending $l_1 \oplus l_2$," respectively. The transition probability matrix for taking actions "sending l_2" and "not sending anything" can be similarly computed.

First, let us consider the transition probability matrix for taking action "sending l_1." This is shown in Fig. 2(a). An entry in row i and column j denotes the transition probability from state i to state j under action "sending l_1." For example, the probability of transition from state 1 to state 4 when sending packet l_1 is $(1 - p_1)(1 - p_2)$. The reason is given as follows: Since state 1 denotes that neither receivers have packets l_1 and state 4 denotes that both receivers have packets l_1, to transition from state 1 to state 4 by sending packet l_1, both receivers must have correctly received l_1, and the probability of this event is equal to $(1 - p_1)(1 - p_2)$. Similarly, other transition probabilities for different states can be computed by using the packet loss probabilities at each receiver.

Let us now consider the transition probability matrix for taking action "sending $l_1 \oplus l_2$." This action is interesting as one transmission by the AP can help two receivers to simultaneously recover two distinct lost packets. Consider a transition from state 10 to state 16 in Fig. 2(b). In state 10, R_1 has l_2 but not l_1, whereas R_2 has l_1 but not l_2. If the AP sends packet $l_1 \oplus l_2$ and the packet is successfully received at both receivers, then both R_1 and R_2 will now obtain $l_1 = l_2 \oplus (l_1 \oplus l_2)$ and $l_2 = l_1 \oplus (l_1 \oplus l_2)$, respectively. The probability of this event is then equal to $(1 - p_1)(1 - p_2)$. Other probability entries can be calculated in a similar manner.

Now, for each action, there is an associated reward matrix. Let us denote r_{ij} as the immediate reward of R_i upon receiving packet l_j. It can be seen that, for broadcast setting, $r_{11} = r_{21}$ and $r_{12} = r_{22}$ since both receivers want packets l_1 and l_2. For unicast setting, we assume that R_1 wants only l_1, whereas R_2 wants only l_2; thus, $r_{12} = 0$, and $r_{21} = 0$. Given this definition, we can express the reward matrix for both unicast and broadcast settings when sending l_1, as shown in Fig. 3(a). For example, the immediate reward when transitioning from state 1 to state 4 under action "sending l_1" is $r_{11} + r_{21}$. The reason is that the reward in state 1 is zero, and with a transition to state 4, both receivers receive l_1 and l_2. Thus, the immediate reward

	States	1 (00 00)	2 (01 00)	3 (10 00)	4 (11 00)	5 (00 01)	6 (01 01)	7 (10 01)	8 (11 01)	9 (00 10)	10 (01 10)	11 (10 10)	12 (11 10)	13 (00 11)	14 (01 11)	15 (10 11)	16 (11 11)
1	00 00	p_1p_2	p_1*(1-p_2)	(1-p_1) *p_2	(1-p_1) *(1-p_2)	0	0	0	0	0	0	0	0	0	0	0	0
2	01 00	0	p_1	0	1- p_1	0	0	0	0	0	0	0	0	0	0	0	0
3	10 00	0	0	p_2	1-p_2	0	0	0	0	0	0	0	0	0	0	0	0
4	11 00	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	00 01	0	0	0	0	p_1p_2	p_1*(1-p_2)	(1-p_1) *p_2	(1-p_1) *(1-p_2)	0	0	0	0	0	0	0	0
6	01 01	0	0	0	0	0	p_1	0	1- p_1	0	0	0	0	0	0	0	0
7	10 01	0	0	0	0	0	0	p_2	1-p_2	0	0	0	0	0	0	0	0
8	11 01	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	00 10	0	0	0	0	0	0	0	0	p_1p_2	p_1*(1-p_2)	(1-p_1) *p_2	(1-p_1) *(1-p_2)	0	0	0	0
10	01 10	0	0	0	0	0	0	0	0	0	p_1	0	1- p_1	0	0	0	0
11	10 10	0	0	0	0	0	0	0	0	0	0	p_2	1-p_2	0	0	0	0
12	11 10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	00 11	0	0	0	0	0	0	0	0	0	0	0	0	p_1p_2	p_1*(1-p_2)	(1-p_1) *p_2	(1-p_1) *(1-p_2)
14	01 11	0	0	0	0	0	0	0	0	0	0	0	0	0	p_1	0	1- p_1
15	10 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2	1-p_2
16	11 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(a)

	States	1 (00 00)	2 (01 00)	3 (10 00)	4 (11 00)	5 (00 01)	6 (01 01)	7 (10 01)	8 (11 01)	9 (00 10)	10 (01 10)	11 (10 10)	12 (11 10)	13 (00 11)	14 (01 11)	15 (10 11)	16 (11 11)
1	00 00	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	01 00	0	p_2	0	0	0	1-p_2	0	0	0	0	0	0	0	0	0	0
3	10 00	0	0	p_1	0	0	0	0	0	0	0	1-p_1	0	0	0	0	0
4	11 00	0	0	0	p_1p_2	0	0	0	p_1* (1-p_2)	0	0	0	(1-p_1) *p_2	0	0	0	(1-p_1) *(1-p_2
5	00 01	0	0	0	0	p_2	1-p_2	0	0	0	0	0	0	0	0	0	0
6	01 01	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	10 01	0	0	0	0	0	0	p_1*p_2	p_1* (1-p_2)	0	0	0	0	0	0	(1-p_1) *p_2	(1-p_1) *(1-p_2)
8	11 01	0	0	0	0	0	0	0	p_1	0	0	0	0	0	0	0	1- p_1
9	00 10	0	0	0	0	0	0	0	0	p_1	0	1- p_1	0	0	0	0	0
10	01 10	0	0	0	0	0	0	0	0	0	p_1*p_2	0	(1-p_1) *p_2	0	p_1* (1-p_2)	0	(1-p_1) *(1-p_2)
11	10 10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	11 10	0	0	0	0	0	0	0	0	0	0	0	p_2	0	0	0	1- p_2
13	00 11	0	0	0	0	0	0	0	0	0	0	0	0	p_1*p_2	p_1* (1-p_2)	(1-p_1) *p_2	(1-p_1) *(1-p_2)
14	01 11	0	0	0	0	0	0	0	0	0	0	0	0	0	p_1	0	1- p_1
15	10 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p_2	1- p_2
16	11 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(b)

Fig. 2. (a) Transition probability matrix associated with action "sending packet l_1." (b) Transition probability matrix associated with action "sending packet $l_1 \oplus l_2$."

	States	1 (00 00)	2 (01 00)	3 (10 00)	4 (11 00)	5 (00 01)	6 (01 01)	7 (10 01)	8 (11 01)	9 (00 10)	10 (01 10)	11 (10 10)	12 (11 10)	13 (00 11)	14 (01 11)	15 (10 11)	16 (11 11)
1	00 00	0	r_{21}	r_{11}	r_{11}+r_{21}	0	0	0	0	0	0	0	0	0	0	0	0
2	01 00	0	0	0	r_{11}	0	0	0	0	0	0	0	0	0	0	0	0
3	10 00	0	0	0	r_{21}	0	0	0	0	0	0	0	0	0	0	0	0
4	11 00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	00 01	0	0	0	0	0	r_{21}	r_{11}	r_{21}+r_{11}	0	0	0	0	0	0	0	0
6	01 01	0	0	0	0	0	0	0	r_{11}	0	0	0	0	0	0	0	0
7	10 01	0	0	0	0	0	0	0	r_{21}	0	0	0	0	0	0	0	0
8	11 01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	00 10	0	0	0	0	0	0	0	0	0	r_{21}	r_{11}	r_{11}+r_{21}	0	0	0	0
10	01 10	0	0	0	0	0	0	0	0	0	0	0	r_{11}	0	0	0	0
11	10 10	0	0	0	0	0	0	0	0	0	0	0	r_{21}	0	0	0	0
12	11 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	00 11	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{21}	r_{11}	r_{11}+r_{21}
14	01 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{11}
15	10 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{21}
16	11 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a)

	States	1 (00 00)	2 (01 00)	3 (10 00)	4 (11 00)	5 (00 01)	6 (01 01)	7 (10 01)	8 (11 01)	9 (00 10)	10 (01 10)	11 (10 10)	12 (11 10)	13 (00 11)	14 (01 11)	15 (10 11)	16 (11 11)
1	00 00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	01 00	0	0	0	0	0	r_{22}	0	0	0	0	0	0	0	0	0	0
3	10 00	0	0	0	0	0	0	0	0	0	0	r_{12}	0	0	0	0	0
4	11 00	0	0	0	0	0	0	0	r_{22}	0	0	0	r_{12}	0	0	0	r_{12}+r_{22}
5	00 01	0	0	0	0	0	r_{21}	0	0	0	0	0	0	0	0	0	0
6	01 01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	10 01	0	0	0	0	0	0	0	r_{21}	0	0	0	0	0	0	r_{12}	r_{21}+r_{12}
8	11 01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{12}
9	00 10	0	0	0	0	0	0	0	0	0	0	r_{11}	0	0	0	0	0
10	01 10	0	0	0	0	0	0	0	0	0	0	0	r_{11}	0	r_{22}	0	r_{11}+ r_{22}
11	10 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	11 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{22}
13	00 11	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{21}	r_{11}	r_{21}
14	01 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{11}
15	10 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r_{21}
16	11 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b)

Fig. 3. (a) Reward matrix associated with action "sending packet l_1." (b) Reward matrix associated with action "sending packet $l_1 \oplus l_2$."

should be equal to the sum of the individual rewards. In the broadcast and unicast settings, this sum is equal to $2r_{11}$ and r_{11}, respectively. Similarly, we can write down the reward matrix for sending $l_1 \oplus l_2$, as shown in Fig. 3(b).

Remarks on the Modeling Complexity and Computational Complexity of BIA: For a small number of receivers, we can use BIA to solve our abstract MDP that corresponds to the scheduling problem. However, the number of states and actions can be exponentially large. Specifically, the number of states is $|S| = 2^{M \times K}$, and the number of actions is $|A| = 2^K$; both exponentially increase as the number of receivers increases. If $M = 6$ receivers and $K = 3$ packets, then the number of states is $|S| = 2^{18}$, and the number of actions is $|A| = 2^3$. From the modeling perspective, BIA requires us to define all state transition probabilities and reward of all transitions. This is infeasible with a large number of states and actions. From the computational perspective, its time complexity of $O(N|S|^2|A|)$ quickly becomes intractable, even for a broadcast session with

a moderate number of receivers. To tackle those issues, we propose an algorithm called simulation-based dynamic programming (SDP).

IV. Simulation-Based Dynamic Programming Algorithm

The time complexity of BIA is $O(N|S|^2|A|)$, which is dominated by the number of states $|S| = 2^{M\times K}$. The number of actions $|A| = 2^K$ is also large but much smaller than the number of states and is manageable for typical scenarios. Therefore, our goal is to devise an algorithm that reduces the modeling complexity and computational complexity. We propose a dynamic programming algorithm based on simulations. The intuition for using such an approach is that, for many problems, going through all the states to determine the optimal actions is not efficient. Rather, through simulations, only the most likely states will be explored for determining the optimal action [33], [38]–[41]. In our paper, we propose a simulation-based method that combines both the dynamic algorithm and the sampling method to solve our large MDP. Our method addresses both the complexity of large state space and modeling process.

A. SDP for Solving a Large MDP

Our proposed SDP algorithm is based on BIA. For a given state s, the SDP algorithm samples each action a in the action space A for a number of iterations N_i to compute the average sampling reward of each tuple (s, a). From those average sampling rewards, the action for a given state that results in the largest reward is the best action. The SDP algorithm samples each action in the action space to determine the next state and the transition reward. The process runs backward from $t = N$ to $t = 1$ similarly as in BIA to find the near optimal policy $\pi^* = \{d^*(s_1), d^*(s_2), \ldots, d^*(s_N)\}$ that produces the maximum final cumulative reward

$$\pi^* = \arg\max_{a\in A} E^{\pi}_{s_t}\left\{\sum_{n=t}^{N-1} r_n(s_n, a_n) + r_N(s_N)\right\}. \quad (11)$$

The SDP algorithm is shown as follows:

The SDP Algorithm

1) Set $t = N$ and $U^*_N(s_N) = 0$ for all $s_N \in S$.

2) Substitute $t - 1$ for t, and compute $U^*_t(s_t)$ for each $s_t \in S$ as follows:

- Sample each action $a \in A$ for N_i iterations, and compute the average

$$\hat{U}_t(s_t, a) = \frac{1}{N_i}\sum_{N_i}\left(r_t(s_t, a) + U^*_{t+1}(j)\right). \quad (12)$$

- Find the highest reward

$$U^*_t(s_t) = \max_{a\in A}\left\{\hat{U}_t(s_t, a)\right\}. \quad (13)$$

- Set

$$a^*(s_t) = \arg\max_{a\in A}\left\{\hat{U}_t(s_t, a)\right\}. \quad (14)$$

3) If $t = 1$, stop. Otherwise, return to step 2.

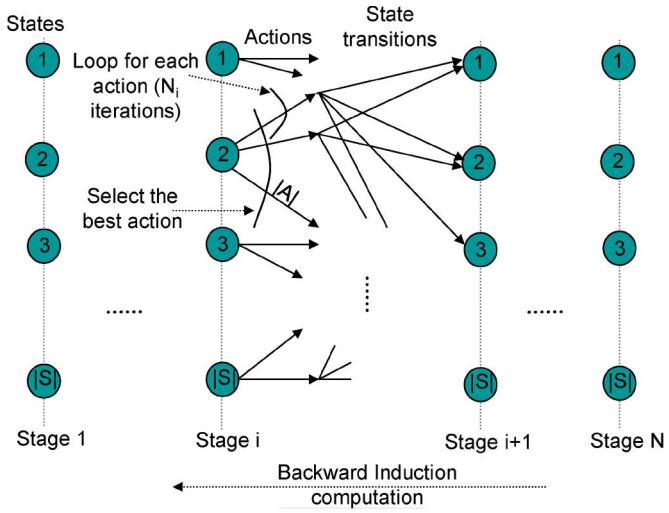


Fig. 4. Graphical illustration of the SDP algorithm. For each state at stage i, all actions in the action space are sampled for N_i iterations, and the mean average reward is computed; the best action is that with the highest mean average reward. The algorithm computes the reward, following the BIA, starting from stage N to stage 1.

In Fig. 4, we provide a graphical illustration of the SDP algorithm.

B. Evaluating the Properties of the SDP Algorithm

We first examine the SDP algorithm to solve our MDP previously defined. Clearly, how close this policy resulted from the SDP algorithm to the optimal policy found by BIA depends on the number of samples per action N_i. If the Markov process is stationary, then it can be shown that the larger the number of samples used, the closer the solution to the optimal one. As N_i goes to infinity, the solution obtained by the algorithm approaches the optimal policy. As an example, we show the empirical results of the convergence rate of the SDP algorithm using a simple setting in which two packets l_1 and l_2 are broadcast to two receivers R_1 and R_2. Assume that, when a receiver correctly receives packet l_1 or l_2, it gets an amount of reward 0.7 or 0.3, respectively. Assume further that l_2 depends on l_1, i.e., l_2 is useful only when l_1 has been received. The packet error probabilities $p_1 = p_2 = 0.1$. As previously noted, there are 16 different states $\begin{bmatrix}0&0\\0&0\end{bmatrix}, \begin{bmatrix}1&0\\0&0\end{bmatrix}, \ldots,$ and $\begin{bmatrix}1&1\\1&1\end{bmatrix}$ and four different actions $a_0 =$ "sending nothing", $a_1 =$ "sending l_1", $a_2 =$ "sending l_2", and $a_3 =$ "sending $l_1 \oplus l_2$". We set the time horizon to be three time steps.

Convergence: As shown in Fig. 5, we have the transmission policies resulted by different numbers of samples per action N_i. In Fig. 5(a1), $N_i = 6$, and the actions are quite different from the actions in the optimal policy in Fig. 5(a3). As the number of samples per action increases to $N_i = 25$, the resulting policy becomes closer to the optimal one. Fig. 5(b) shows the convergence rate to the optimal policy in terms of the amount of average reward per receiver. When increasing the number of samples per action, the reward value gets close to the optimal value.

Complexity: Our simulation-based algorithm, to some extent, avoids the well-known modeling complexity in MDP problems. Specifically, when K and M are large, it is intractable to

	stages			
state	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	3	3	3
5	1	2	2	2
6	1	2	2	1
7	3	3	3	3
8	3	1	1	3
9	1	2	2	2
10	3	3	3	3
11	1	1	1	1
12	1	1	1	1
13	2	3	3	3
14	3	3	3	2
15	2	2	3	2
16	0	0	0	0

(1)

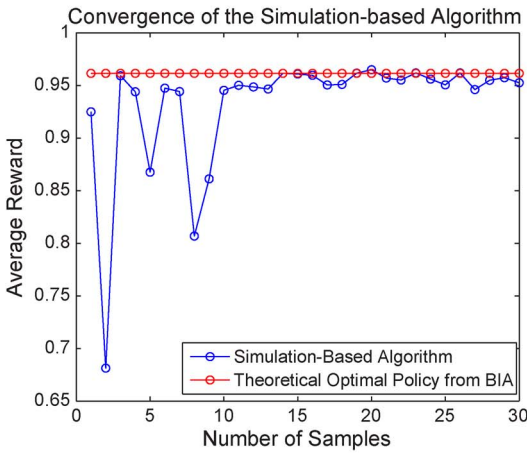
	stages			
state	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	3	1	1
5	1	2	1	1
6	1	2	2	1
7	3	3	3	3
8	3	3	1	1
9	1	2	2	2
10	3	3	3	3
11	1	1	1	1
12	1	1	1	1
13	2	3	2	3
14	3	3	2	2
15	2	2	3	2
16	0	0	0	0

(2)

	stages			
state	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	2	2	1	1
6	1	1	1	1
7	3	3	3	3
8	1	1	1	1
9	1	1	1	1
10	1	1	1	3
11	1	1	1	1
12	1	1	1	1
13	2	2	2	2
14	2	2	2	2
15	2	2	2	2
16	0	0	0	0

(3)

(a)



(b)

Fig. 5. Convergence property of the SDP algorithm. (a) Transmission policies resulting from different numbers of samples per action N_i (a1: transmission policy resulting by running six samples per action; a2: transmission policy resulting by running 25 samples per action; a3: optimal policy by BIA). (b) Policy convergence rate of the SDP algorithm as a function of the samples per action.

explicitly construct the transition probability matrix and transition reward function. Using the proposed SDP method, these are implicitly captured. Second, the SDP algorithm reduces the computational complexity ("the curse of dimensionality" problem). It can be readily seen that the time complexity for this algorithm is $O(NN_i|S\|A|)$. Because N_i is normally much smaller than $|S|$, this time complexity is significantly less than $O(N|S|^2|A|)$ of BIA.

V. Scheduling With Erroneous Feedback

So far, we assume that the feedbacks from the receivers are error free. We now examine the case of erroneous feedback.

POMDPs: With error-free feedback, the AP correctly obtains the states of the receivers and views the state transitions of receivers as a Markov process. With error-prone feedback, however, the AP views those states as hidden states, i.e., partially observable states. The AP gets the observations that are not the actual states of the receivers. It then has to select which packet to send, based on these observations to maximize the sum reward returned at the receivers. This decision-making problem is called partially observable MDPs (POMDPs) [41]–[43].

Example: Consider the example in which two packets l_1 and l_2 are sent to two receivers R_1 and R_2. Assume that the beginning actual state is $s_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and that the AP, at the beginning, also observes the state as $o_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. Now, the AP takes action "sending l_1," and two receivers correctly receive l_1. However, the feedback of the first receiver is erroneous, and the AP will get the observation $o_2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$. Similarly, in the next time step, the AP takes action "sending l_2," which may result in the actual state $s_3 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ and the observation $o_3 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$. As a result, the actual state transition is $s_1 \rightarrow s_2 \rightarrow s_3$, which is different from the observation transition $o_1 \rightarrow o_2 \rightarrow o_3$.

Solving POMDP: The formulation and modeling for POMDP are identical with MDP, except that the state observations are not the actual states. Specifically, the observations are defined as

$$o = \begin{bmatrix} ob_1^1 & ob_1^2 & \cdots & ob_1^K \\ ob_2^1 & ob_2^2 & \cdots & ob_2^K \\ \cdots & & & \\ ob_M^1 & ob_M^2 & \cdots & ob_M^K \end{bmatrix}$$

where entry $ob_j^i = \{0, 1\}$ indicates the observation from the AP on whether the receiver R_j has received packet l_i or not: $ob_j^i = 1$ indicates that the AP observes that R_j has been correctly received l_i, and $ob_j^i = 0$ indicates otherwise. We note again that, if the feedback is error free, then the observation completely describes the state of the receivers, i.e., the observation and the state are the same $(o = s)$. Because of the erroneous feedback, however, it partially describes the receivers' state. The solution to the POMDP is a policy from time step $t = 1$ to $t = N$: $\pi^* = \{d^*(s_1), d^*(s_2), \ldots, d^*(s_N)\}$, where $d^*(s_t)$ is a set of actions at time step t that maximizes the sum reward

$$\pi^* = \arg\max_{a \in A} E_{s_t}^{\pi} \left\{ \sum_{n=t}^{N-1} r_n(o_n, a_n) + r_N(o_N) \right\}. \quad (15)$$

We note that this equation is different from (11) by the observation o_n in the equation. This means that the AP relies on observation o rather than the exact state s as in the MDP to select the action at each time step. Fortunately, to solve our POMDP, we can again use the SDP algorithm.

VI. Simulation Results

We present simulation results that demonstrate the advantages of our scheduling algorithm using NC with SDP. Since a one-hop network is fairly simple, as no routing is necessary, we implement our own network simulator in C/C++. This gives us the flexibility to set network parameters and the new scheduling algorithms, which is harder to do in a sophisticated network simulator such as NS.

We select K to present the number of video layers in each frame and N to present the number of time slots available to transmit these K packets. If the channel is error free, then $N < K$ indicates the shortage of bandwidth, and $N > K$ indicates the redundancy of bandwidth to transmit K packets. In our simulations, we use video frames consisting of three to four layers, i.e., $K = 3$ or $K = 4$, depending on the scenarios. These values of N and K are chosen based on the coding rates of certain video sequences in simulations.

The AP has the scalable video sequence Foreman to be sent to the receivers. Each frame is encoded into four layers l_1, l_2, l_3, and l_4. Each layer is packetized into a packet, resulting to four packets denoted as packets l_1^F, l_2^F, l_3^F, and l_4^F, which correspond to four layers l_1, l_2, l_3, and l_4, respectively. To use NC, packets must have the same size; therefore, additional padding bits are inserted into some packets. Associated with each packet is a reward or a distortion reduction amount in terms of MSE. In particular, we use the distortion reduction $r_1^F = 14.67$, $r_2^F = 10.60$, $r_3^F = 6.85$, and $r_4^F = 5.83$, as provided in [5] and [15]. In addition, since each multimedia packet has an associated playback deadline, our problem is modeled as a finite-horizon MDP. The objective of the proposed SDP is to maximize the total distortion reduction over a given time horizon N.

A. Packet Loss Model as Bernoulli Trials

We assume that the packet losses at all the receivers are independent and follow Bernoulli trials. First, we want to see how close a solution obtained by SDP is to that with a small number of receivers. Specifically, we measure the performances of those algorithms for a broadcast setting in which a Foreman sequence is broadcast to two receivers R_1 and R_2. For SDP, we use 20 and 30 samples per action, $K = 3$, and $N = 5$. As shown in Fig. 6, the NC scheme using BIA gives the optimal scheduling policy, leading to the largest distortion reduction. The performance of the NC scheme with SDP gets closer to that of the NC scheme with BIA as the number of samples increases from 20 to 30. This is in agreement with the fast convergence property of SDP discussed earlier.

Now, we consider the scenario with a higher number of receivers such that it is no longer trivial to write down the analytical expression of the transition probabilities that enables us to run the optimal NC-based BIA algorithm. Fortunately, the simulation-based MDP (SDP) does not need the explicit representation of the transition probabilities. We show the performance of SDP with two other algorithms: 1) the retransmission algorithm (ARQ) without NC and 2) the greedy algorithm with NC.

Retransmission Scheme: The AP sends packets starting from the packet with the largest distortion reduction to that with the smallest distortion reduction, i.e., following the order l_1^F, l_2^F, l_3^F, and l_4^F. Each packet is sent until either it is correctly received at the intended receiver(s) or the number of transmissions exceeds N. After N time slots, regardless of whether the AP successfully sends all four packets, it moves to the next four packets (layers) of the next frame.

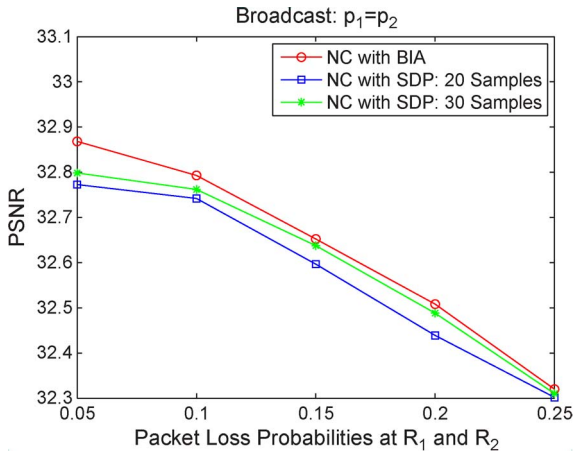


Fig. 6. Average PSNR of video sequences at each receiver versus the packet loss probabilities for the broadcast setting.

Greedy Algorithm With NC Scheme: The scheme selects a packet or a coded packet to send so that the distortion reduction is maximized after every immediate transmission. The AP also maintains the set of actions as in the NC with MDP scheme. However, at each transmission step, the AP observes the feedback to determine the receivers' state and computes the action that provides the largest reward. Essentially, the greedy algorithm optimizes the transmission for one time step.

Fig. 7 shows the average distortion reduction as a function of packet loss probabilities in the broadcast setting for three schemes. In Fig. 7(a), the packet loss probabilities of all receivers vary from 5% to 25%, whereas the total number of transmission opportunities N is kept constant at 5. As seen, the SDP algorithm performs the best, followed by the greedy algorithm with NC and then the non-NC retransmission algorithm. For a fixed $N = 5$, an increase in loss rate results in a decrease in throughput. With a small delay requirement $(N = 5)$, it is critical to schedule the packets to maximize the video qualities at the receivers. When the loss rate is 5% and $N = 5$, the AP has many opportunities to successfully transmit all the packets to receivers, resulting in minimal distortion.

Next, we investigate the performances of these algorithms when the receivers have different packet loss rates. The packet loss rates for R_2, R_3, and R_4 are now set to $p_2 = 0.1$, $p_3 = 0.2$, and $p_4 = 0.3$, and the loss rate for R_1 is varied from 0.05 to 0.25. N is set to 5. As shown in Fig. 7(b), the SDP algorithm provides the highest media quality. It is interesting that the performance gap between the SDP scheme and the other two schemes becomes larger, compared with the case where receivers have identical packet loss rates. This perhaps suggests that SDP is more beneficial when the packet loss rates greatly vary among receivers.

We now consider the concurrent unicast setting in which two receivers R_1 and R_2 receive two different video sequences, i.e., Foreman and Coastguard, respectively. We assume that the

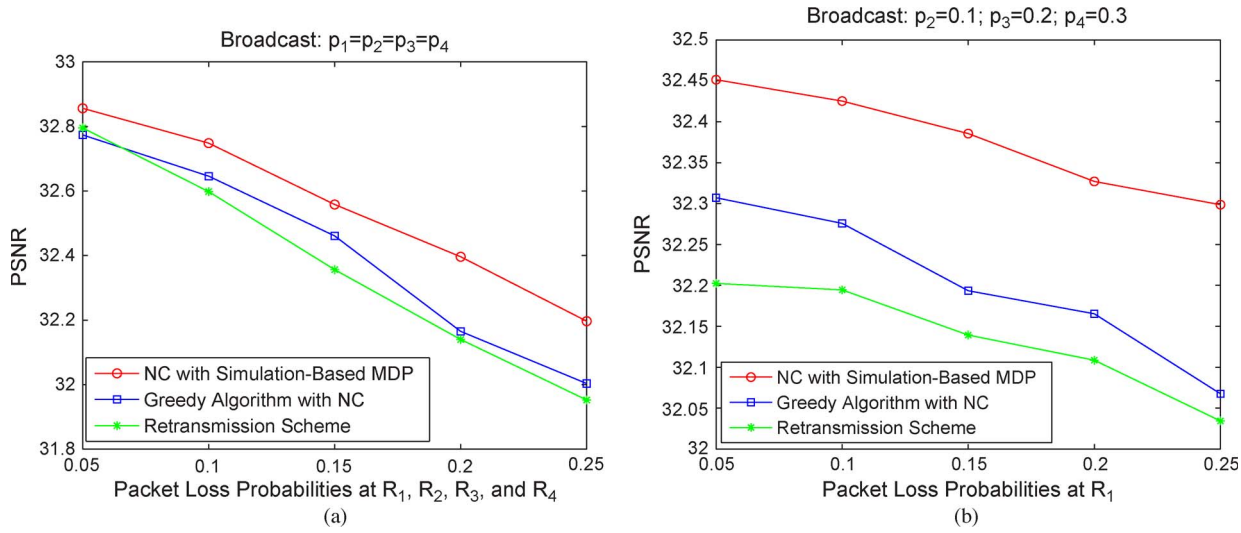


Fig. 7. Average PSNR of video sequences at each receiver in the broadcast scenario with a fixed number of transmission opportunities. (a) Average PSNR versus packet loss probabilities at R_1, R_2, R_3, and R_4. (b) Average PSNR versus packet loss probability at R_1.

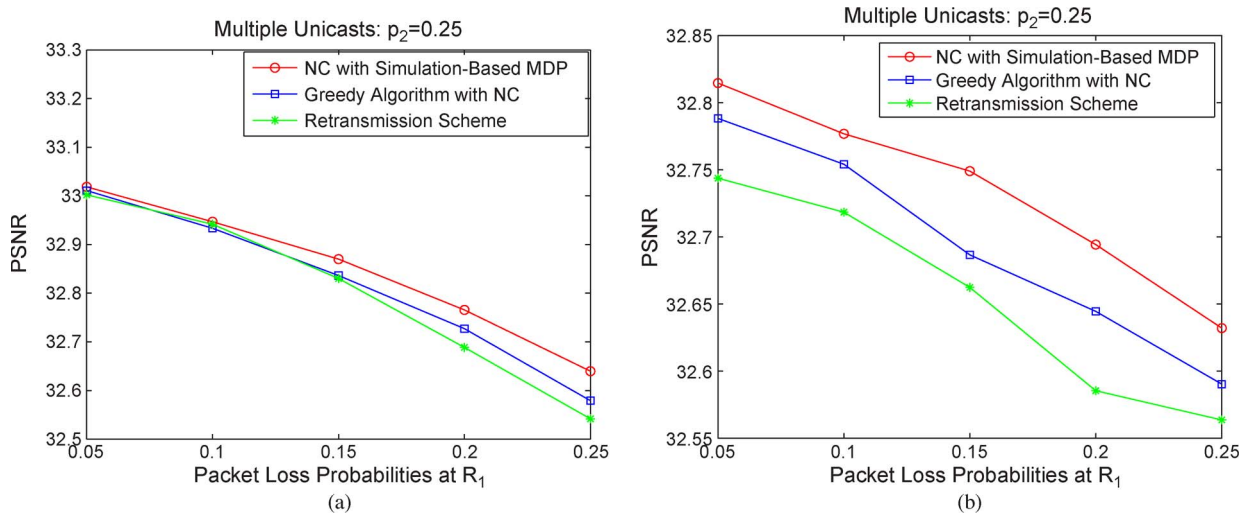


Fig. 8. Average PSNR of video sequences at each receiver in the concurrent unicast scenario. (a) Average PSNR versus packet loss probabilities at R_1 and R_2 ($p_1 = p_2$). (b) Average PSNR versus packet loss probability at R_1.

Foreman video sequence has three packets l_1^F, l_2^F, and l_3^F with their corresponding distortion reduction in MSE $r_1^F = 14.67$, $r_2^F = 10.60$, and $r_3^F = 6.85$, and Coastguard has three packets l_1^C, l_2^C, and l_3^C with their corresponding distortion reduction in MSE $r_1^C = 20.84$, $r_2^C = 15.34$, and $r_3^C = 9.54$. Fig. 8(a) shows the distortion reduction as a function of the loss rates for fixed $N = 7$, and the packet loss probabilities at two receivers p_1 and p_2 identically vary from 0.05 to 0.25. Similarly, Fig. 8(b) shows the distortion reduction when the packet loss rate of R_1 is kept constant at $p_1 = 0.2$ and when the packet loss rate of R_2 is varied from $p_2 = 0.05$ to 0.25. As predicted, the smaller packet loss probabilities provides better performance, as shown in both Fig. 8(a) and (b). Again, the SDP algorithm leads to the largest reward or best video qualities. For the same broadcast and unicast settings, we now examine the scenarios where the loss rates are kept constant and where the number of transmission opportunities N is varied. Fig. 9(a) and (b) shows the rewards as a function of N for three algorithms in the broadcast and unicast settings, respectively. As N increases, there are more opportunities to retransmit the lost packets; thus, the performances of all three algorithms also increase. Again, the SDP performs the best, followed by other algorithms. When N increases to some value, the performances of all three schemes converge to the same maximum value. This implies that there are enough transmission opportunities to correctly send all packets to the receivers, and thus, optimization does not matter much in this case. Still, the SDP converges to the maximum value faster than two other schemes.

We now present the simulation results for the case of erroneous feedback. For the broadcast scenario, the AP broadcasts the Foreman sequence to four receivers R_1, R_2, R_3, and R_4 with their corresponding packet loss probabilities $p_1 = 0.05$, $p_2 = 0.1$, $p_3 = 0.2$, and $p_4 = 0.3$. For the unicast scenario, the AP concurrently unicasts the Coastguard and Foreman sequences to two receivers R_1 and R_2 with packet loss probabilities $p_1 = 0.1$ and $p_2 = 0.25$, respectively. In both settings, we vary the feedback error probabilities from all receivers to the AP from 0% to 20% and keep $N = 5$. As shown in Fig. 10, when the feedback error probability increases, the performance of the retransmission algorithm significantly

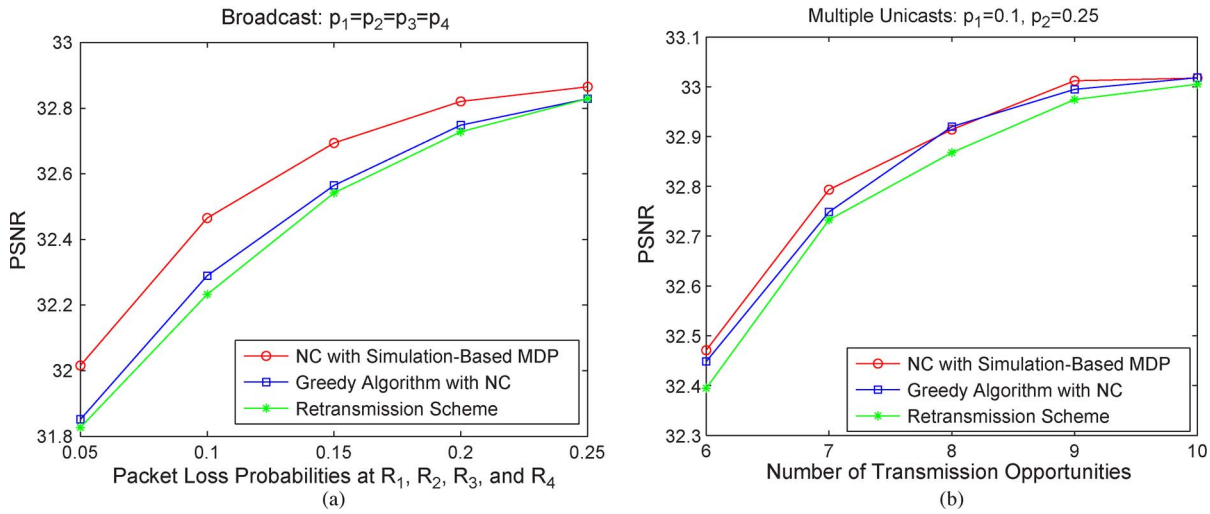


Fig. 9. Average PSNR of video sequences at each receiver versus the number of transmission opportunities. (a) Broadcast to four receivers. (b) Concurrent unicast to two receivers.

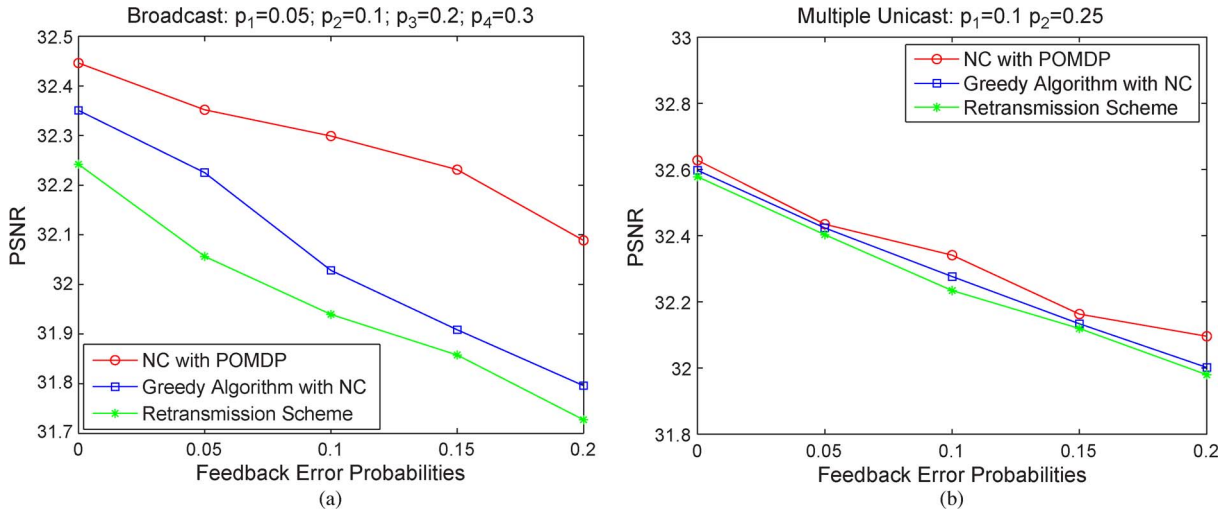


Fig. 10. Average PSNR of video sequences at each receiver versus the number of transmission opportunities for the case of erroneous feedback. (a) Broadcast to four receivers. (b) Concurrent unicast to two receivers.

reduces, whereas the NC framework with POMDP is able to maintain relatively high video quality.

B. Bursty Loss Channel With Two-State Markov Error Model

We present the simulation results for packet loss patterns generated by the Gilbert model, showing that the relative performance gains of the SDP algorithm over other algorithms remain approximately the same. The Gilbert model aims to describe bursty packet losses. The state of a channel is classified into "good" and "bad" states with probabilities p_{good} and p_{bad}, respectively. When the channel is in the good state, the packet loss probability p_{good} is small, and when it is in the bad state, the packet loss probability p_{bad} is much larger. The channel state changes at each transmission slot with transition probabilities $\alpha = p_{\text{good}->\text{bad}}$, $\beta = p_{\text{bad}->\text{good}}$. The stationary probabilities for the channel in good and bad states are $\pi_{\text{good}} = (\beta/\beta+\alpha)$ and $\pi_{\text{bad}} = (\alpha/\beta+\alpha)$, respectively. We evaluate the performances of different schemes for a four-receiver scenario in broadcast setting and for a two-receiver scenario in unicast setting, with each receiver having identical channel conditions. We use Foreman sequence for broadcast setting and Foreman and Coastguard for unicast setting. For simplicity, we set β to a constant value while varying α. Fig. 11 shows the video quality of different transmission schemes. As α increases from 0.05 to 0.25 while β is unchanged, the portion of time that the channel is in "bad" state is larger, leading to the lower average video quality. Overall, the performance gaps among the considered algorithms remain approximately the same.

C. Remarks on the Performance of MDP Algorithms

The simulation results in the previous section show that an improvement in peak signal to noise ratio (PSNR) resulting from using the MDP-based algorithm over the greedy retransmission-based algorithm ranges from 0.1 to 0.5 dB, depending on the scenarios. However, it is important to emphasize several points regarding such modest performances. First, we note that the proposed MDP framework is optimal in the sense that it minimizes the expected distortion subject

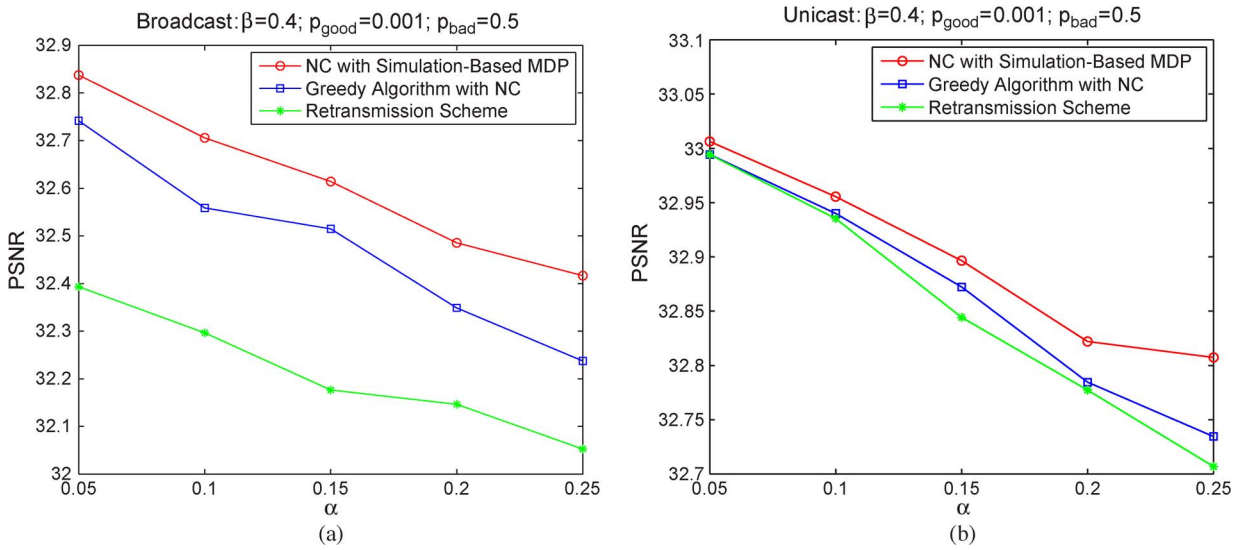


Fig. 11. Average PSNR of video sequences at each receiver versus the state transition probabilities α. (a) Broadcast to four receivers. (b) Concurrent unicast to two receivers.

to the constraint on the number of transmission opportunities. Therefore, ignoring the computational aspect, there cannot be a better algorithm than that proposed in the same setup, at least for the case where the number of receivers is small such that one can analytically compute the transition probabilities. Furthermore, the SDP algorithm will converge to the optimal solution, given a sufficiently large number of samples used. Now, the performance gain of BIA (optimal) over the greedy retransmission-based algorithm is not that much. This indicates that the greedy retransmission-based algorithm is already very good for such scenarios. Second, it is not necessarily the case that the MDP framework always produces modest gain over the greedy algorithm. In fact, there are many factors that make the greedy retransmission-based algorithm perform arbitrarily bad in the broadcast scenario. One such factor is the number of receivers. Specifically, it can be theoretically shown that, as the number of receivers increases, the performance gap between NC- and retransmission-based algorithms becomes larger [44]. One other factor that affects the performance gap between the NC-based and the greedy algorithm is the characteristic of video sequences in consideration. Therefore, our contributions lie mainly in the framework for obtaining the optimal solution and that the actual numerical performance gain is rather dependent on the scenarios, which is hard to characterize.

D. Remarks on the Practicality of the Proposed MDP Algorithms

One of the main drawbacks with the proposed algorithms (BIA and SDP) is the explosion of feedback when there is a large number of receivers in a session. Certainly, a more thorough effort is needed to address this issue. One short-term remedy as done in our simulations and can be extended to real-world scenarios is to artificially limit the number of receivers in a session. Suppose that the number of receivers in a session is limited to M and that if there are $N > M$ receivers wanting to receive the same stream, we can always logically divide a large session into N/M sessions, each containing M receivers. The performance of such a scheme will be suboptimal, but it is an engineering tradeoff for scalability. In fact, we have implemented a protocol with a similar ACK scheme, showing its feasibility on actual 802.11 devices [45].

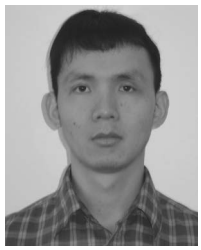
VII. Conclusion and Future Work

In this paper, we have proposed an NC-based scheduling policy at an AP that optimizes the multimedia transmission in both broadcast and concurrent unicast settings in WLAN-like or WiMAX-like networks. In particular, our contributions include the following: 1) an optimized scheduling algorithm based on the MDP to maximize the quality of multimedia applications and 2) simulation-based algorithms to solve large MDP and POMDP problems. Our sampling-based dynamic programming algorithm has two advantages: 1) simplifying the modeling complexity in transforming the scheduling problem to an abstract MDP and 2) reducing the computational complexity. Under typical packet loss rates, the transmission policy found by our MDP framework provides higher media quality than the retransmission and the NC-based greedy methods.

References

[1] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2006.
[2] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast application," in *Proc. Archit. Protocols Comput. Commun.*, Oct. 2000, pp. 43–56.
[3] T. Nguyen and A. Zakhor, "Multiple sender distributed video streaming," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 315–326, Apr. 2004.
[4] W. Tan and A. Zakhor, "Error control for video multicast using hierarchical FEC," in *Proc. 6th Int. Conf. Image Process.*, Oct. 1999, vol. 1, pp. 401–405.
[5] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301–317, Mar. 2001.
[6] W. Tan and A. Zakhor, "Real time Internet video using error resilient scalable compression and TCP friendly transport protocol," *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 172–186, Jun. 1999.
[7] C. Fragouli, J. Le Boudec, and J. Widmer, "Network coding: An instant primer," Swiss Fed. Inst. Technol., Lausanne, Switzerland, Tech. Rep. TR2005010, 2005.

[8] Y. Wu, P. A. Chou, and S. Kung, "Information exchange in wireless networks with network coding and physical-layer broadcast," Microsoft Res., Redmond, WA, Tech. Rep. MSR-TR-2004-78, Aug. 2004.
[9] H. Ma and M. El Zarki, "Broadcast/multicast MPEG-2 video over wireless channels using header redundancy FEC strategies," *Proc. SPIE*, vol. 3528, pp. 69–80, Nov. 1998.
[10] P. A. Chou, A. E. Mohr, A. Wang, and S. Mehrotra, "Error control for receiver-driven layered multicast of audio and video," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 108–122, Mar. 2001.
[11] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degradation over packet erasure channels through forward error correction," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 819–828, Jun. 2000.
[12] G. De Los Reyes, A. Reibman, S. Chang, and J. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 1063–1074, Jun. 2000.
[13] J. Robinson and Y. Shu, "Zerotree pattern coding of motion picture residues for error resilient transmission of video sequences," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 1099–1110, Jun. 2000.
[14] U. Horn, K. Stuhlmuller, M. Link, and B. Girod, "Robust Internet video transmission based on scalable coding and unequal error protection," *Signal Process.: Image Commun.*, vol. 15, no. 1/2, pp. 77–94, Sep. 1999.
[15] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, Mar. 2001.
[16] P. A. Chou and A. Sehgal, "Rate-distortion optimized for receiver-driven streaming over best effort networks," in *Proc. Packet Video Workshop*, Apr. 2002.
[17] M. Kalman and B. Girod, "Techniques for improved rate-distortion optimized video streaming," *ST J. Res.-Networked Media*, vol. 2, pp. 45–54, Mar. 2004.
[18] M. L. Puterman, *Markov Decision Processes Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.
[19] R. Howard, *Dynamic Programming and Markov Decision Processes*. Cambridge, MA: MIT Press, 1960.
[20] R. Ahlswede, N. Cai, R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
[21] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. ACM SIGCOMM*, Aug. 2007, pp. 169–180.
[22] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *Proc. SIGCOMM*, Sep. 2006, pp. 243–254.
[23] Y. Wu, P. A. Chou, and S. Kung, "Minimum energy multicast in mobile ad hoc networks using network coding," in *Proc. IEEE Inf. Theory Workshop*, Oct. 2004, pp. 304–309.
[24] J. Widmer, C. Fragouli, and J.-Y. Le Boudec, "Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding," in *Proc. Workshop Netw. Coding, Theory, Appl.*, Apr. 2005.
[25] W. Chen, K. B. Letaief, and Z. Cao, "A cross-layer method for interference cancellation and network coding in wireless networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2006, pp. 3693–3698.
[26] W. Chen, K. B. Letaief, and Z. Cao, "Opportunistic network coding for wireless networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2007, pp. 4634–4639.
[27] A. Eryilmaz, A. Ozdaglar, and M. Medard, "On delay performance gains from network coding," in *Proc. Conf. Inf. Sci. Syst.*, Mar. 2006, pp. 864–870.
[28] C. Fragouli, D. Katabi, A. Markopoulou, M. Medard, and H. Rahul, "Wireless network coding: Opportunities and challenges," in *Proc. IEEE Mil. Commun. Conf.*, Oct. 2007, pp. 1–8.
[29] D. Nguyen, T. Nguyen, and X. Yang, "Wireless multimedia transmission with network coding," in *Proc. Packet Video*, Nov. 2007, pp. 326–335.
[30] D. Nguyen and T. Nguyen, "Network coding-based wireless media transmission using POMDP," in *Proc. Packet Video*, May 2009, pp. 1–9.
[31] H. Seferoglu and A. Markopoulou, "Opportunistic network coding for video streaming over wireless," in *Proc. Packet Video*, Nov. 2007, pp. 191–200.
[32] H. Seferoglu and A. Markopoulou, "Video-aware opportunistic network coding over wireless networks," *IEEE J. Sel. Areas Commun.—Network Coding for Wireless Communication Networks*, vol. 27, no. 5, pp. 713–728, Jun. 2009.
[33] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[34] A. Gosavi, *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning,* 1st ed. New York: Springer-Verlag, 2003.
[35] D. Nguyen, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," in *Proc. 3rd Workshop Netw. Coding, Theory, Appl.*, Jan. 2007, pp. 1–6.
[36] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb. 2009.
[37] A. Eryilmaz, A. Ozdaglar, M. Medard, and E. Ahmed, "On the delay and throughput gains of coding in unreliable networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 12, pp. 5511–5524, Dec. 2008.
[38] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-Based Algorithms for Markov Decision Processes (Communications and Control Engineering)*. New York: Springer-Verlag, 2007.
[39] H. S. Chang, R. Givan, and E. K. P. Chong, "On-line scheduling via sampling," in *Proc. Artif. Intell. Plan. Syst.*, Apr. 2000, pp. 62–71.
[40] M. Kearns, Y. Mansour, and A. Ng, "A sparse sampling algorithm for near optimal planning in large Markov decision processes," in *Proc. 16th Int. Joint Conf. Artif. Intell.*, Jun. 1999, pp. 1324–1331.
[41] A. Y. Ng and M. Jordan, "PEGASUS: A policy search method for large MDPs and POMDPs," in *Proc. 16th Conf. Uncertainty Artif. Intell.*, Jun. 2000, pp. 41–48.
[42] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artif. Intell. Res.*, vol. 24, no. 1, pp. 195–220, Aug. 2005.
[43] N. Meuleau, K. E. Kim, L. P. Kaelbling, and A. R. Cassandra, "Solving POMDPs by searching the space of finite policies," in *Proc. 15th Int. Conf. Uncertainty Artif. Intell.*, Aug. 1999, pp. 417–426.
[44] T. Tran, T. Nguyen, B. Bose, and V. Gopal, "A hybrid network coding technique for single-hop wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 685–698, Jun. 2009.
[45] R. Edgecombe, "An implementation of a reliable broadcast scheme for 802.11 using network coding," M.S. thesis, Oregon State Univ., Corvallis, OR, 2008.



Dong Nguyen received the B.S. degree in electrical engineering from Hanoi University of Technology, Hanoi, Vietnam, in 2000, the M.S. degree in computer engineering from Yonsei University, Seoul, Korea, in 2005, and the Ph.D. degree in electrical and computer engineering from Oregon State University, Corvallis, in 2009.

Since 2009, he has been a Lecturer with FPT University, Hanoi. His research interests are wireless networking and network coding.

Thinh Nguyen (M'04) received the B.S. degree from the University of Washington, Seattle, in 1995 and the Ph.D. degree from the University of California, Berkeley, in 2003.

He is currently an Associate Professor with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis. He has many years of experience as an engineer for a variety of high-tech companies. He has served as Associate Editor for *Peer-to-Peer Networking and Applications* His research interests include multimedia networking and processing, wireless networks, and network coding.

Dr. Nguyen has served as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the IEEE TRANSACTIONS ON MULTIMEDIA. He has served on many technical program committees.

Xue Yang received the M.S. and B.E. degrees from the University of Electronic Science and Technology of China, Chengdu, China, and the Ph.D. degree from the University of Illinois at Urbana-Champaign.

She is currently a Staff Research Scientist and Technical Lead of several projects with Intel Labs, Intel Corporation, Santa Clara, CA. She is the author of more than 30 journal/conference proceeding papers. He is the holder of more than 30 U.S. and international patents. Her current research interests are wireless networking, mixed networks, mobile virtualization, and positioning.