Opportunistic Network Coding for Video Streaming over Wireless

Hulya Seferoglu, Athina Markopoulou
Electrical Engineering and Computer Science Dept.
University of California, Irvine
Irvine, CA, USA
{*hseferog,athina*}@*uci.edu*

***Abstract*— In this paper, we study video streaming over wireless networks with network coding capabilities. We build upon recent work, which demonstrated that network coding can increase throughput over a broadcast medium, by mixing packets from different flows into a single packet, thus increasing the information content per transmission. Our key insight is that, when the transmitted flows are video streams, network codes should be selected so as to maximize not only the network throughput but also the video quality. We propose video-aware opportunistic network coding schemes that take into account both aspects, namely (i) the decodability of network codes by several receivers and (ii) the distortion values and playout deadlines of video packets. Simulation results show that our schemes significantly improve both video quality and throughput.**

I. Introduction

Providing high quality video over wireless networks is a challenging problem, due to both the erratic and time-varying nature of a wireless channel and the stringent delivery requirements of media traffic. Developments in video compression and streaming, wireless networking, and cross-layer design, are continuously advancing the state-of-the art in wireless video [1], [2]. In this paper, we propose a novel technique for video streaming in a wireless environment inspired by the emerging paradigm of network coding [3], [4].

Our work builds on recent work in [5], [6] that used network coding to improve throughput in a wireless mesh network. In particular, [5], [6] proposed that wireless routers mix packets from different flows, so as to increase the information content of each -broadcast- transmission and therefore the throughput for data applications. In this paper, we build on this idea, and propose a network coding and scheduling scheme for transmitting several video streams over a wireless mesh network.

Our key insight is that the transmission of video streams in a network coding-capable wireless network should be optimized not only for network throughput but also, and more importantly, for video quality. The fact that video packets have unequal importance is well understood and extensively studied in the video streaming community, e.g. for rate-distortion optimized streaming [7]–[9]. The fact that mixing different information flows can increase throughput in multicast networks is well understood in the network coding community [3], [4], [10], [11]. Our work bridges the gap between the two approaches, and proposes a new video-aware scheme for network coding and packet scheduling that improves both aspects, namely video quality and throughput.

We consider a wireless mesh network, in which routers can mix different incoming flows/streams, using simple network coding operations (XOR). The resulting network code is broadcasted to the neighborhood of the router. Nodes in the same neighborhood listen to each other's transmission and store overheard packets; these are used later to decode received coded packets and also to construct new coded packets. The core question in this architecture is how to select the best -according to an appropriate metric- network code for transmission among all possible codes. In [5], [6], a transmitting node chooses a network code that can be decoded by several neighbors at the same time slot; this policy increases the utility of each transmission thus leading to throughput benefits. However, when the transmitted flows are video streams, this is not necessarily the best choice. Video quality can be improved by intelligently selecting network codes that combine those video packets that are decodable by several neighbors but also contribute the most to video quality. In other words, when video streams are transmitted, it is not only the quantity but also the quality/content of information transferred that should be taken into account in the selection of network codes. In this paper, we develop schemes for network code selection and packet scheduling that take into account both (i) the importance and deadlines of video packets and (ii) the network state and the received/overheard packets in the neighborhood.

The paper is organized as follows. Section II discusses related work. Section III gives an overview of the system model. Section IV presents the algorithms for network coding. Section V presents simulation results that demonstrate the benefits of the proposed algorithms over baseline schemes and discusses their complexity. Section VI concludes the paper.

II. Related Work

This work combines ideas and techniques from two bodies of work: video streaming and network coding.

Several network-adaptive techniques have been proposed to support streaming media over unreliable and/or

This work has been supported by a fellowship from the Center for Pervasive Communications and Computing (CPCC) at UC Irvine.

time-varying networks [12]. Supporting video over wireless is particularly challenging due to the limited, time-varying resources of the wireless channel [1], [2]. There is a large body of work on cross-layer design for video over wireless, including [13]–[17]. Packet scheduling is an important control at the medium access control layer. The problem of rate-distortion optimized packet scheduling has been studied in the RaDiO family of techniques [7]–[9]: in every transmission opportunity, media units are selected for transmission so as to maximize the expected quality of received video subject to a constraint in the transmission rate, and taking into account transmission errors, delays and decoding dependencies. Cross-layer approaches exploit the fact that packets in a video stream have different importance and therefore should be treated differently by network mechanisms.

Independently, the network coding paradigm has emerged from the pioneering work in [10], [11], which showed that, in multicast networks where intermediate nodes do simple linear operations on incoming packets, one can achieve the min-cut throughput of the network to each receiver. The linearly combined packets can be utilized at the receivers to recover the original packets by solving a set of linear equations over a finite field. This breakthrough idea inspired significant effort in several directions [3], [4], including studying topologies beyond multicast, such as unicast [18]–[20] and broadcast scenarios. The broadcast nature of the wireless medium offers an opportunity for exploiting the throughput benefits of network coding [21], [22]. The recent work in [5], [6] applied these ideas from the network coding community in the context of wireless mesh networks. [5] implemented a pseudo-broadcast mechanism for 802.11 together with opportunistic listening and a coding layer between IP and MAC that is used to detect coding opportunities and pack packets from different flows into a single transmission, thus increasing network throughput.

Our paper introduces a novel technique for video streaming over wireless that combines the above two approaches. On one hand, we build on [5], [6] to exploit the broadcast nature of the wireless medium and use network coding to pack several packets *from different streams* into a single code for transmission, thus increasing throughput. On the other hand, we construct and select network codes taking into account the importance of video packets (in terms of video distortion and playout deadlines) *within the same stream*, as well as their contribution to the total throughput and video quality. This combined approach allows us to achieve significant video quality improvement while still maintaining the throughput benefits.

III. System Overview

We consider video streaming over wireless mesh networks where intermediate nodes (wireless mesh routers) are able to forward packets to other intermediate nodes and/or clients, as shown in Fig. 1. In this paper, we propose algorithms that can be used at the intermediate node to maximize video quality and throughput. We assume that intermediate nodes can perform simple network

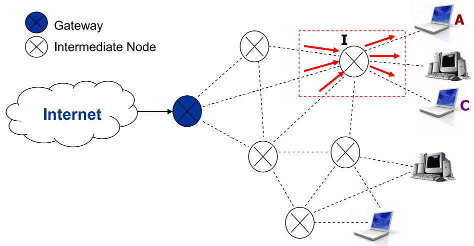


Fig. 1. A wireless mesh network

coding operations (bit-wise XOR) and combine packets from several incoming streams into a single outgoing packet. This packet is broadcasted to the entire neighborhood, thus reaching several nodes at the same time. We assume that nodes can overhear all transmissions in their neighborhood, whether they are intended for them or not; they can decode a network-coded packet using overheard packets. The idea of combining network coding with broadcast to increase the information content per transmission, is well understood in the network coding community. This idea has been recently applied in 802.11-based multi-hop wireless networks and throughput benefits have been demonstrated for data applications [5], [6].

Our key observation is that, when the transmitted flows are video streams, this is not necessarily the best choice and video quality must also be considered. The importance and deadlines of video packets must be taken into account to intelligently select those network codes that contribute the most to the quality of video streams. In this paper, we develop schemes for network coding across different flows, and packet selection within each flow, to improve both video quality and throughput.

Code Selection at the Intermediate Node. Let us consider an intermediate node that receives N packets from different video streams and forwards them to N nodes in its neighborhood. The intermediate node maintains a *transmission (Tx) queue* with incoming video packets. At a given time slot a packet is selected from the Tx queue for transmission. The selected packet is called the *primary packet* and its destination node is called the *target node*. Depending on the network coding scheme, the primary packet may be the first packet from the head of the queue, or any packet in Tx queue that is marked as *active* (*i.e.* not transmitted within the last round-trip time). The primary packet can be thought as the main packet we try to transmit during a time slot, e.g. the packet that would be transmitted by a FIFO policy. In addition to the primary packet, all packets in the queue are considered as candidate *side packets*, i.e. candidates for a transmission in the same time slot together with the primary packet; these are useful to nodes other than the target node. The primary and the side packets are all XOR-ed together into a single packet, called the *network code*. The core question then is:

> *which network code (i.e. XOR of the primary and side packets) to select and transmit so as to maximize the total video quality and throughput.*

TABLE I

TERMINOLOGY

Term	Definition
Primary Packet	The packet selected from the Tx queue before network coding. It must be included in all network codes. It can be thought as the main packet we try to transmit in a given time slot.
Side Packet	Packet in the Tx queue, other than the primary, included in the network code.
Active Packet	Packet in the Tx queue that can be considered as primary. (Not transmitted within the last RTT.)
Inactive packet	Packet in the Tx queue that cannot be considered as primary. (It has already been transmitted within the last RTT, and the acknowledgement is still pending.)
Network Code	The primary and side packets XOR-ed together into a single packet.
Target Node	The intended recipient of the primary packet.
Tx Queue	The output queue of the transmitting node.
Rx Buffer	The receive queue of the receiving node. It stores received packets, destined to this node.
Virtual Buffer	Also maintained at a receiving node. It stores overheard packets, destined to other nodes.

The algorithms addressing this question are the main part of this paper, and will be discussed separately in the next section (IV). In the rest of this section, we describe the remaining components and functions of the system. The terminology is summarized in Table I.

ACKs and Other Functions at the Receiving Nodes. Once the network code is chosen, it is broadcasted to all nodes in the neighborhood. Depending on the channel conditions, some nodes successfully receive it. When the target node receives it, it decodes it (which is guaranteed by the construction of the network code in the next section), stores the primary packet in its *receive (Rx) buffer*, and sends an acknowledgement (ACK) back to the intermediate node. Nodes, other than the target node, overhear the transmitted packet and try to decode it; if they overhear a new packet destined to them, they store it in their Rx buffer and send an ACK back to the intermediate node; if they obtain a packet destined for another node, they store it in their *virtual buffer*. An overheard packet stays in the virtual buffer until an ACK from the target is overheard or until its deadline expires.

Retransmissions and Active/Inactive Packets at the Intermediate Node. The intermediate node waits for a mean round-trip time (RTT) from the time it transmits the network code until it receives an ACK. During that period, all packets that were part of the code stay in the Tx queue but are marked as *inactive*. Inactive packets are not considered for primary transmission (in order to avoid unnecessary duplicate transmissions) but are still considered as candidates for side packets (to increase coding opportunities). When the transmitter receives an ACK, it removes the corresponding packet from the Tx queue. If an RTT expires without receiving an ACK, the packet is marked as *active* again and the process is repeated. A packet stays in the Tx queue until either it is successfully transmitted or its deadline expires; when either of these occur, the packet is removed from the transmission buffer.

Notice that considering any active packet as primary, better utilizes the bandwidth but may lead to reordering. Although this may be a concern for TCP, it is clearly better for video that requires timely delivery and can reorder packets at the playout buffer.

Requirements. We assume the following capabilities available at our system. First, *broadcast* is needed to harvest the benefits of network coding. Although wireless is inherently a broadcast medium, this may be hidden by some communication protocols. We assume that some broadcast capability is available, e.g. 802.11 broadcast or pseudo-broadcast as implemented in [5], [6]. Second, nodes need to know the *contents of the virtual buffers* of all their neighbors, in order to code. In our simulations we assume perfect knowledge of the contents of all virtual buffers. This can be achieved by exchanging and guessing this information, as in [5], [6]; in practice, there will be some error and a slight degradation in the overall performance. Third, nodes must be capable of *coding/decoding* in real time, which is a realistic assumption for simple (bit-wise XOR) operations. Finally, we assume a Tx queue with video packets only.

IV. CODING ALGORITHMS

The main questions in this system have to do with the construction and selection of network codes. The *code construction* problem is concerned with finding candidate codes that guarantee decodability by the target and several other nodes. The *code selection* problem is concerned with finding the best among the candidate codes to optimize video quality and throughput. The first proposed algorithm, NCV, achieves the same throughput gains as in [5] but also intelligently chooses the packets to improve video quality. The second algorithm, NCVD, uses NCV as a building block but considers more coding options, thus further improving video quality and throughput.

A. NCV Algorithm: Network Coding for Video

Assume that there are several video streams coming to an intermediate node that can be mixed together. Depending on the content of virtual buffers at the clients, there may be several combinations of these streams, i.e. several network coding opportunities. The main idea behind the Network Coding for Video (NCV) algorithm is to select the best network code to improve video quality. Let us demonstrate this idea through an example.

Example 1: Consider the example shown in Fig. 1 and let us focus on a single-hop shown in more detail in Fig. 2. Node I receives three independent video streams, e.g. from the Internet through the gateway, destined to its neighbors A, B, C. I maintains a FIFO Tx queue that stores packets $\{A_1, A_2, ...\}$ destined to node A, $\{B_1, B_2, ...\}$ destined to node B, and $\{C_1, C_2, ...\}$ destined to node C. Fig. 2 also shows the contents of the virtual buffers at each client: node A has overheard packets $\{B_1, C_1\}$ and nodes B and C have both overheard packet A_1, from previous transmissions. A_1 is the first active packet from head of the queue and is selected as the primary packet. Any packet (active or inactive) in the output queue, other than A_1, can be chosen as a side

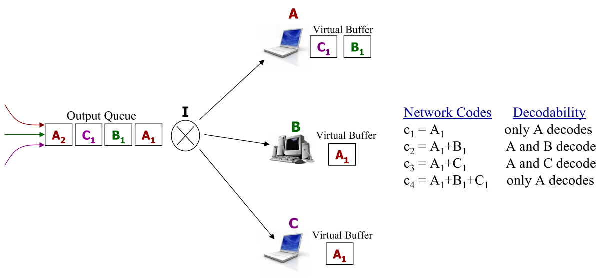


Fig. 2. Example of Network Coding for Video (NCV)

Algorithm 1 The NCV Algorithm

Initialization: $I^i_{max} = 0$, $c^i_{max} = \emptyset$
Choose the first head-of-queue active packet as primary p_i.
Let n_m be the target node of p_i. Let $\{\nu_{m,1}, ..., \nu_{m,\Psi_m}\}$ be the overheard packets at n_m.
for $k = 1...2^{\Psi_m}$ **do**
    $c^i_k = \{p_i\} \bigcup S^m_k$
    Calculate I^i_k with Eq. (3)
    if $I^i_k > I^i_{max}$ **then**
        $I^i_{max} = I^i_k$, $c^i_{max} = c^i_k$
    end if
end for
Choose c^i_{max} as the network code. XOR all packets and transmit

packet, on the condition that the constructed network code should be decoded at node A, i.e. A_1 can be retrieved. To satisfy this condition, side packets that will be used in the network code should already be available at node A; in other words, the decodability of a network code depends on the overheard packets at node A. Network codes $c_1 = A_1$, $c_2 = A_1 \bigoplus B_1$, $c_3 = A_1 \bigoplus C_1$, and $c_4 = A_1 \bigoplus B_1 \bigoplus C_1$ can all be decoded by A and thus are eligible network codes. □

The Code Construction Problem. More generally, consider an intermediate node I with N clients $\{n_1, n_2, ..., n_N\}$. There are Φ packets in the Tx queue, which are denoted with $\{p_1, p_2, ..., p_\Phi\}$. Choose the first active packet, p_i, from the head of the Tx FIFO queue as the primary packet with target node n_m. I will construct and transmit a network code, which consists of p_i XOR-ed together with some side packets. The network code should be constructed so as to guarantee decodability of the primary packet at its target node n_m. For p_i to be decodable at n_m, all $n-1$ side packets must be among the overheard packets at n_m. Assume that Ψ_m packets are overheard at node n_m and denoted by $\{\nu_{m,1}, \nu_{m,2}, ..., \nu_{m,\Psi_m}\}$. Therefore, the candidate network codes are:

$$c^i_k = \{p_i\} \bigcup S^m_k, k = 1, 2, ..., 2^{\Psi_m}$$

where S^m_k is the k^{th} subset of $\{\nu_{m,1}, \nu_{m,2}, ..., \nu_{m,\Psi_m}\}$. Note that, since linear operations are limited to bit-wise XOR, a network code $p_1 \bigoplus p_2 \bigoplus ... \bigoplus p_k$ is completely specified by the set of packets $\{p_1, p_2, ..., p_k\}$ that are XOR-ed together. The complexity of considering all possible network codes is discussed in section V-C. The next step, is to select the best among all candidate codes.

Example 1 Continued. Node A can get packet A_1 from all possible network codes. Codes c_2 and c_3 improve the video quality at node sets $\{A, B\}$ and $\{A, C\}$, respectively. It is clear that c_2 and c_3 are better codes than c_1 and c_4 both for throughput (they are useful to two instead of one node) and video quality. Comparing c_2 to c_3, we observe that they are equivalent in terms of throughput but they may contribute differently to video quality depending on the content of video packets A_1, B_1, C_1. Deciding which candidate code to select between $c_2 = A_1 \bigoplus B_1$ and $c_3 = A_1 \bigoplus C_1$ should depend on the importance and urgency of the original video packets B_1 and C_1. NCV exploits this observation. □

The Code Selection Problem. In order to choose the best code, we first need to define a metric that captures the contribution of each candidate code to video quality improvement. Assume that p_i is the primary packet targeted to node n_m, and $\{c^i_k\}_{k=1}^{k=2^{\Psi_m}}$ are all the candidate codes. Let $I^i_k(n_m)$ be the improvement in video quality at client n_m, when c^i_k is received and decoded:

$$I^i_k(n_m) = \sum_{l=1}^{L_k} (1 - e^k_l) \Delta^k_l g^k_l d^k_l, \tag{1}$$

where each factor in this formula is defined as follows:

- L_k is the number of original packets included in network code c^i_k. At most one out of these L_k packets can be useful to a particular node n_m but different packets are useful to different nodes.
- d^k_l and g^k_l are indicator functions that express whether code k is useful for node n_m. We define $d^k_l = 1$ if c^i_k is decodable at node n_m, or 0 otherwise. We define $g^k_l = 1$ if packet l is targeted to node n_m, or 0 otherwise.
- Δ^k_l is the improvement in video quality (SNR) if packet l is received correctly and on time at client n_m. To compute Δ^k_l, we decode the entire video sequence with this packet missing and we compute the resulting distortion.[1] We assume that this computation is performed at the source offline and that the distortion value is marked on each packet.[2]
- e^k_l is the loss probability of packet l due to channel errors or latency:

$$e^k_l = \int_\tau^\infty p_F(t)dt + \left(1 - \int_\tau^\infty p_F(t)dt\right) \varepsilon_F(s). \tag{2}$$

[1]This is an approximation as the actual distortion that may also depend on the delivery status of prior and subsequent NALs. The distortion model can be extended to capture these correlations [23]. Furthermore, we assume that distortion caused by loss of multiple NALs in the same stream are additive, which is reasonable for sparse losses. These approximations reduce the computational complexity by separating the total distortion into a set of individual distortion functions and optimizing for each one.

[2]For real-time traffic, one can still estimate the distortion by performing online analysis with a delay of a few frames. Most distortion occurs in the first few frames after a loss and breaks after the next I frame; the error depends on the video content of subsequent frames and on the coding decisions. Another approach is to assign distortion values based solely on the GOP structure, ignoring the video content and coding decisions, or to use a model for dependencies [24].

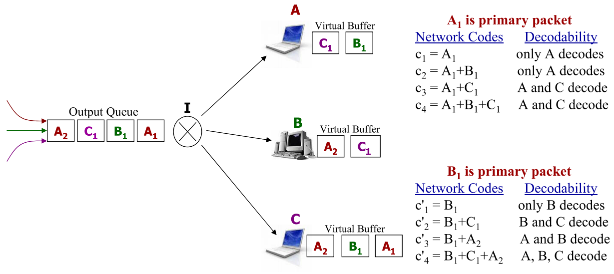


Fig. 3. Example of NCVD

Algorithm 2 The NCVD Algorithm

```
Initialization: c_max = ∅, I_max = 0
for every packet i = 1, ..., , Φ from the head of Tx queue do
    Consider this packet, p_i, as candidate for primary
    Construct all possible codes c_k^i for p_i
    Determine the max improvement I_max^i = max_k I_k^i
    and the corresponding code c_k^i: k = argmax I_k^i as in NCV
    if I_max^i > I_max then
        I_max = I_max^i, c_max = c_k^i
    end if
end for
Choose c_max as the network code. XOR all packets and transmit.
```

The first part in Eq.(2) describes the probability of a packet arriving late; τ is the remaining time until the playout deadline and $p_F(t)$ is the distribution of the forward-trip time. The second part describes the loss probability (of a packet that is still on time) due to effects of the wireless channel, such as noise, fading, interference, etc; $\varepsilon_F(s)$ is the loss probability at state s of the channel.

After defining the contribution of code c_k^i to a single node n_m, $I_k^i(n_m)$, we define the total video quality improvement of code c_k^i as the sum of the video quality improvements at all clients $n_1, ...n_N$,[3] due to code c_k^i:

$$I_k^i = \sum_{m=1}^{N} I_k^i(n_m), \tag{3}$$

The NCV algorithm is summarized in Alg. (1). In each time slot, the NCV algorithm chooses the primary packet p_i and constructs all candidate network codes $c_k^i{}_{k=1}^{k=2^{\Psi_m}}$. Among all candidate network codes, NCV chooses the code that maximizes the total video quality improvement:

$$\max_k I_k^i \tag{4}$$

Depending on the contents of the virtual buffers, there may be no side packets can be used together with the primary packet p_i. In that case, the network code is simply $\{p_i\} \cup \emptyset = \{p_i\}$ and we transmit the primary packet alone.

A note on quality vs. throughput. The objective function I_k^i captures the improvement in video quality that code c_k^i brings to all nodes. Although throughput is not explicitly mentioned, it is implicitly captured. E.g. if we omit the distortion term $(1 - e_l^k)\Delta_l^k$ from Eq. (1), Eq. (3) simply becomes $I_k^i = \sum_{m=1}^{N} \sum_{l=1}^{L_k} g_l^k d_l^k$, which counts the absolute number of packets delivered in a single transmission, i.e. throughput, similarly to NCT and to [5]. Therefore, our metric successfully captures both the number and the quality of packets in a code.

B. NCVD Algorithm: looking into the queue in Depth

As described in the previous section, NCV selects the primary packet from the head of the queue but ignoring packets marked as "inactive". This limits the candidate codes to those that are decodable for this single primary packet. The second algorithm, NCVD looks into the entire Tx queue ("in depth") and considers all, not just the head-of-line, packet as candidates for the primary packet, thus increasing the options for candidate codes, which eventually leads to a better choice for the metric of interest. Note that a different set of candidate codes can be constructed for each primary packet. Let us explain NCVD through the following example.

Example 2: Let us look at Fig. 3. The topology is the same as in Fig. 2, but the contents of the Tx queue and of the virtual buffers are different. Assume that all packets are active packets, i.e. they can all be considered as primary. One option is to select the head-of-line packet A_1 as the primary packet. As discussed in Example 1, the best codes for this primary packet are $c_3 = A_1 \bigoplus C1$ or $c_4 = A_1 \bigoplus B_1 \bigoplus C_1$. A different choice is to select B_1 as the primary packet, which leads to completely different set of candidate network codes (listed on the Fig. 3). Code $c'_4 = B_1 \bigoplus C_1 \bigoplus A_2$ achieves the maximum throughput improvement, and potentially the maximum video quality achievement, depending on the importance and urgency of all packets. This example demonstrates that increasing our options of primary packet, increases the set of candidate codes, and thus can potentially improve both throughput and video quality. □

More generally, NCVD constructs candidate codes $c_k^i, k = 1, 2, ..., 2^{\Psi_m}$ for each candidate primary packet p_i in the Tx queue. Among all constructed codes, NCVD selects the code that maximizes the total improvement in video quality for all clients:

$$\max_{p_i} \max_k (I_k^i), \tag{5}$$

Algorithm 2 summarizes NCVD. NCVD looks in depth d into the Tx queue for candidate primary packets. NCVD($d = 1$) is simply NCV, while NCVD($d = \infty$) considers all packets in the Tx queue. The larger the value of d, the more coding options, the better the performance of NCVD. Because queue sizes are small for real time applications, we can focus on NCVD($d = \infty$), which we simply refer to as NCVD.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed schemes (*NCV* and *NCVD*) in terms of video

[3] If the quality of the encoded video sequences to different clients are significantly different from each other, then the terms $I_k^i(n_m)$ can be normalized by the average PSNR per sequence before summation.

quality and network throughput under different conditions. We compare them to two baseline schemes, one without coding (*noNC*) and one with network coding optimized for throughput (*NCT*) as in [5]. Simulation results show that NCV and NCVD can significantly improve video quality and application-level throughout, without compromising MAC-level throughput. Finally, we discuss the complexity of the proposed schemes.

A. Simulation Setup

First, we describe the communication scenario, wireless channel models and video sequences used, and the baseline algorithms for comparison.

1) Single-Hop Scenario: In this paper, we focus on the evaluation of the single-hop scenario shown in Fig. 2: the intermediate node I receives different video streams which it forwards downstream towards their destinations A, B, C. I can apply different schemes for network coding and packet scheduling in the downlink. The downlink rate is selected to be 300 kbps, and each video stream has a rate of 70 kbps. We assume that the three video streams are the only ones using the downlink, hence, there is no congestion. However, packets may still be lost due to error on the wireless channel, according to models described in the next paragraph, and can also experience a random delay, which we assume to be exponentially distributed with $4ms$ average. The delay budget (playout deadline) for this single hop is set to $100ms$, to allow for queueing, transmission, propagation and a few retransmissions. We also perform simulations for different delay budgets, from 50ms to 200ms. ACKs and retransmissions are used to combat errors, as explained in the system overview.

2) Wireless Channel Model: We consider two channel models to evaluate our algorithms in different conditions. In both cases, packet loss is independent and identically distributed across the three sessions (from I to A, from I to B, and from I to C). Below we specify the packet loss model within a session.

Model I: Packets transmitted in each link experience i.i.d. loss. with a fixed loss probability. We simulated a wide range of effective packet loss rates from 1% up to 20%. The effective loss rate depends on the use of retransmissions, FEC and other mechanisms that mask the error rate on the channel.

Model II: A flat Rayleigh fading channel is modeled by a finite-state Markov chain (MC), where the fading is approximated as a discrete time Markov process with time discretized typically to the channel coherence time [25]. The set of all possible fading gains (equivalently, SNR or BER levels) correspond to the states of the MC, and transitions between these states occur at each interval with certain probabilities. We simulated a two-state model characterized by the parameters specified in [26]: (i) the bit-error-rate (BER) of each state, which depends on the channel signal-to-noise (SNR) level (ii) the state duration and transition probabilities, which depend on the channel coherence time, and in turn on the speed of mobiles and the channel frequency. We assume nomadic or pedestrian wireless clients with 3 km/h speed, and 2.4 GHz channel frequency, which results in an average coherence time of 21 ms. This duration ensures that the channel remains static during a packet transmission. In our experiments, we consider different channel quality levels assigning average SNR levels from the set $\{3, 5, 7, 9\}$ dB, resulting in effective packet loss rates from 1% to 35%.

3) Video Sequences: As our test sequence, we used standard sequences: *Carphone*, *Foreman*, *Mother & Daughter*. These were QCIF sequences encoded using the JM 8.6 version of the H.264/AVC codec [27], [28]. The group of pictures consisted of I and P frames, one I every 10 frames. All encoded sequences had data rates of 70 kbps each and frame rate of 30 fps. Each frame consists of at least one slice. Each slice was packetized into an independent NAL (network abstraction layer) unit, of size 250B. There are two reasons for this choice of packet size. First, due to the time varying nature of wireless channels, it is preferred to have short packet transmissions to avoid the channel variation during a packets transmission. Second, using a fixed size (on the average) simplifies the network coding (bit-wise XOR) operations requiring small or no padding of the packets.

The average peak-signal-to-noise (PSNR) ratio for the encoded sequences, Carphone, Foreman and Mother& Daughter, were $29.95dB$, $28.70dB$ and $40.74dB$ respectively; these PSNR values, of the encoded sequences before transmission, are denoted as "No Error" in the first row of Table II and correspond to the top lines of Fig. 4 . We repeated and concatenated the standard sequences to create longer test sequences of duration 30sec each.

We simulated packet loss by erasing the corresponding NAL units from the RTP stream, according to the packet traces produced by the network simulation. At the receiver side, we decoded the remaining RTP stream with standard error concealment enabled. When an entire frame was lost, we used copy-concealment.

4) Algorithms under Comparison: We compare NCV and NCVD against two baseline algorithms for packet scheduling: no Network Coding (noNC) and Network Coding for Throughput (NCT).

No Network Coding (noNC): This is a FIFO Tx queue without network coding. Consider again Example 1 and Fig. 2: node I stores packets for all three streams destined to nodes A, B, C. In every time slot, I transmits the first packet from the head of the queue. It may require several consecutive retransmissions until the head-of-line packet is successfully transmitted; in 802.11, there is an upper limit in the maximum number of allowed retransmissions. In order to conduct a fair comparison with our schemes, we slightly modify (improve) this scenario by using the same scheme as described in section III.[4]

[4]The summary of the scheme is as follows. After transmission, a packet is marked as *inactive* and is not transmitted as primary for a time duration of a mean RTT; during that period, other packets are transmitted from the FIFO as primary, thus better utilizing the channel. After an RTT, if an ACK is still not received, the packet is marked as *active* and considered again for transmission. Packets whose playout deadlines have expired are removed from the Tx queue.

Network Coding for Throughput (NCT): This is an improved version of the algorithm proposed in [5]. The packet transmission mechanism is the same as in the *noNC* scheme, but network coding is used to maximize throughput, as follows. The packet at the head of the Tx queue is selected as a primary packet; side packets are chosen to be XOR-ed together with the primary packet so as to construct a network code that is useful to the maximum number of receivers possible at this time slot.

There are two improvements in NCT compared to the coding algorithm in [5] that allow NCT to achieve even higher throughput than [5]. First, NCT follows the same ACK and retransmission mechanism described in section III and repeated in Footnote 4: packets with pending acknowledgments are marked as inactive for one RTT, while the channel is used to transmit other packets as primary. In [5] and in general MAC retransmissions, a packet stays at the head of the queue blocking other packets, until it goes through successfully or it exceeds the maximum number of retransmissions. Head-of-line blocking avoids reordering, which may be an issue for TCP traffic. However, in the context of video streaming, the playout buffer can handle reordering. Another difference is that NCT uses an improved version of the coding procedure in [5]: NCT considers all possible subsets of the candidate side packets thus maximizing the number of receivers that can decode; while [5] considers side packets in a sequential order, thus sacrificing some throughput for reduced complexity. Therefore, we use NCT as our baseline for the maximum achievable throughput per transmission using network coding.

The main difference between NCV/NCVD and NCT is that our schemes select side packets to maximize video quality while NCT maximizes throughput. A secondary difference, is that we consider all packets in a queue as candidates for side packets, while NCT, consistently with [5], considers only the earliest packet per flow.

B. Simulation Results

In this section, we present simulation results that compare our schemes to the baselines and demonstrate that NCV and NCVD can improve video quality and application-level throughout, without compromising MAC-level throughput. We report simulation results for the single-hop scenario of Fig. 2, when node I streams sequences *Carphone*, *Foreman*, and *Mother and Daughter* streamed to clients A, B, C, respectively.

1) Video Quality Improvements: Fig. 4 shows the video quality experienced by the clients (PSNR over frame number for parts of the sequences) for the four algorithms under comparison, namely noNC, NCT, NCV, NCVD, as well as for the encoded sequences before transmission (noError). The simulation is performed for wireless channel *Model I* at packet loss rate of 9.4% with 100 ms delay budget; for comparison, the same wireless channel trace is used as input to all 4 algorithms. As expected, there are time periods, during which the channel is bad, the quality degrades for all algorithms. However, the degradation for NCV and NCVD is much less than for NCT and noNC, because NCV and NCVD select network codes to protect and deliver the most important packets on time, thus improving the video quality; in contrast, NCT and noNC treat all packets similarly.

TABLE II

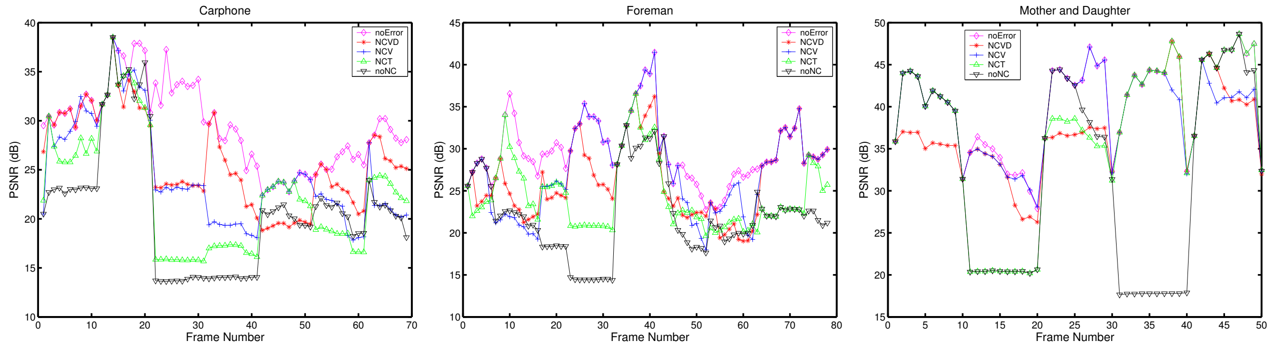
AVERAGE PSNR FOR THE SCENARIO OF FIG.4 ($70kbps$ VIDEO RATE, CHANNEL MODEL I WITH 9.4% LOSS, $100ms$ PLAYOUT DEADLINE)

avg PSNR (dB)	Carphone	Foreman	Mother&Daughter
No Error	29.95	28.70	40.74
NCVD	26.32	26.08	32.87
NCV	23.99	25.03	32.62
NCT	22.40	22.76	30.81
noNC	22.08	21.59	26.92

The average PSNR for each sequence and algorithm is summarized in Table II. As expected, the noNC scheme performs poorly. NCT improves over noNC because it delivers more packets per time slot. NCV improves over NCT because it chooses the most important video packets; although the number of packets does not increase over NCT, their quality does. NCVD further improves over NCV because it considers more candidate codes and opportunities. These numbers are compared to the original encoded sequence (NoError).

Fig. 5 focuses on the client that receives the sequence *Foreman*. The same scenario as in Fig. 4 is considered, but with loss rates varying from 1% to 20%. Fig. 5 shows the average PSNR for each value of packet loss rate and for each algorithm. Clearly, NCV and NCVD outperform NCT (by $2.5-3.5$dB) and noNC (up to $3.5-5$ dB) for all packet loss rates. Another observation (from this and other figures omitted for lack of space) is that the PSNR gain of NCV and NCVD is larger for medium than for very low and very high packet loss rates. For low loss rates, most packets are transmitted successfully, while for high loss rates most packets are lost. In both cases, the number of network coding opportunities decrease. However, even then, the proposed algorithms still achieve a considerable PSNR improvement. The upper part of Fig. 5 shows the video quality for the Foreman sequence, and the lower part shows the PSNR averaged across all sequences.

Fig. 6 evaluates the same scenario as in Fig.5, but for the second wireless channel *Model II*. This is the two-state model, parameterized by the SNR levels (resulting in loss rates from 1% to 35%). The compared algorithms are ranked similar to the previous case and the PSNR improvements from NCV and NCVD are still high. However, there is less improvement compared to channel *Model I*. The difference can be explained by the network coding and code selection opportunities. In *Model I*, consecutive transmissions to the same client experience independent loss, and links to different clients are independent from each other. This independence results in all virtual buffers having roughly the same number of packets, over short time periods. However, in *Model II*, the channel alternates between a good and a bad state; while in a bad state, a client is more likely to experience consecutive losses, while clients in a good state are more likely to receive consecutive successful



(a) Carphone (b) Foreman (c) Mother & Daughter

Fig. 4. PSNR per frame for (parts of) the test sequences, for channel Model I with packet loss 9.4% and delay budget 100ms. Four schemes (noNC, NCT, NCV, NVCD) are compared. "NoError" refers to the encoded sequences without any loss. The average PSNR values (averaged over the entire sequence) are summarized in Table II).

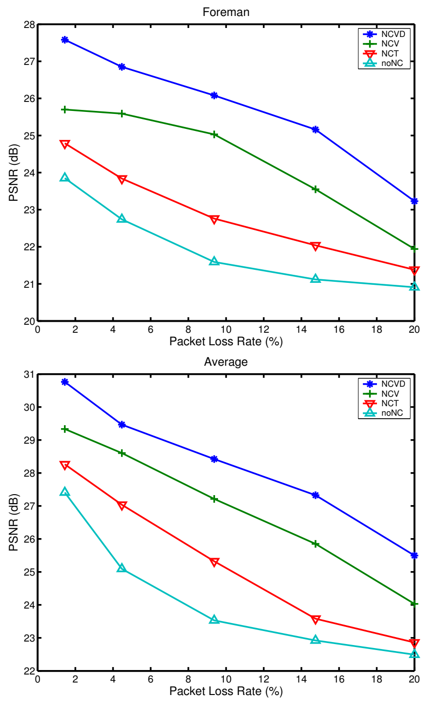


Fig. 5. Video quality observed over wireless channel *Model I*. Average PSNR for *Foreman*, and avg PSNR averaged across all 3 sequences.

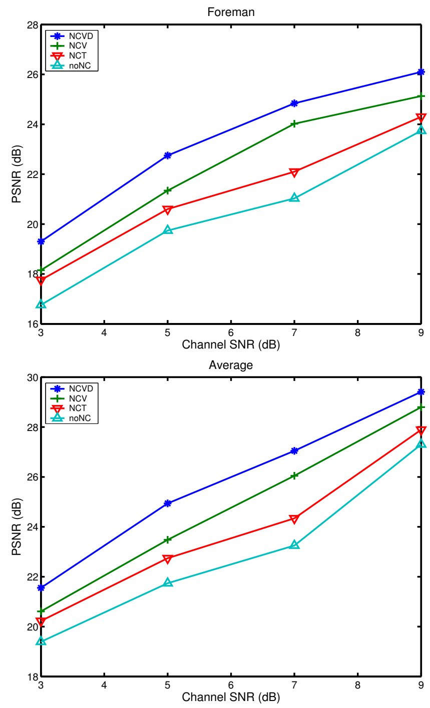


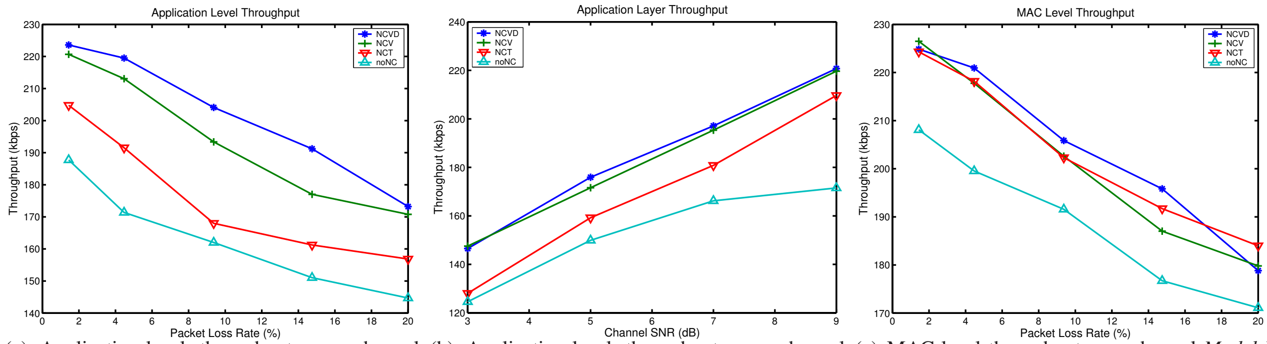
Fig. 6. Video quality observed over wireless channel *Model II*. Average PSNR for *Foreman*, Average PSNR, averaged across all three sequences.

packets. This results in an unbalanced number of packets in the virtual buffers. Since network code construction directly depends on the number of packets in virtual buffers, there are less network coding opportunities, hence less network code selection possibilities. Even when there are less opportunities, we still observe significant quality improvement: NCV and NCVD improves up to 2 and 3dB over NCT and up to 3 and 4dB over noNC, respectively.

In the previous scenarios, we considered a delay budget of 100ms. We now consider channel Model I at 9.4% packet loss rate, with delay values ranging from 50 to 200ms. Fig. 7 depicts the PSNR values for each scenario (first averaged over each sequence and then across the three sequences, to summarize the overall improvement). The figure shows that both NCV and NCVD improves video quality for the entire range of delay values. The improvement is smaller for a tight delay budget, because a tight delay constraint limits the number of retransmissions and the lifetime of packets both at the Tx queue and the virtual buffers, thus decreasing network coding and selection opportunities. However, even with tight delay constraints, there is significant video quality improvement from NCV and NCVD compared to NCT and noNC.

2) Throughput Improvements: The video-aware schemes improve video quality because they explicitly take it into account in the code selection. In this section, we show that, our schemes also significantly improve application-level throughput.

Application throughput. Figures 8(a) and (b) show the total throughput as seen by the application-layer (i.e. NAL units per sec) added over all clients: NCV and NCVD achieve higher throughput as compared to NCT



(a) Application-level throughput over channel *Model I*

(b) Application-level throughput over channel *Model II*

(c) MAC-level throughput over channel *Model I*

Fig. 8. Total throughput (added over all three streams) achieved by the four algorithms.

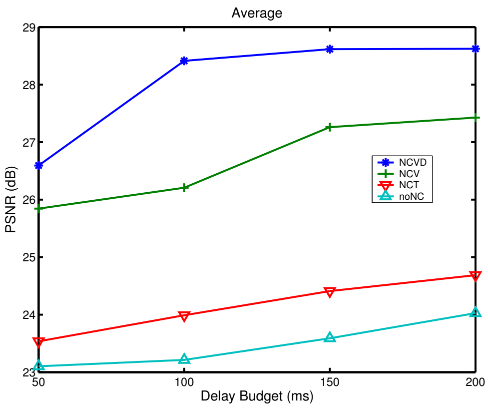


Fig. 7. PSNR values (averaged over each sequence and across sequences) for different delay budgets. Wireless *Model I* is considered.

and noNC. The reason is that NCV and NCVD do not select codes consisting of packets whose deadlines are within one transmission time, while NCT and noNC transmit all packets. Late packets do not contribute to application-level throughput, because they are discarded at the client even if they are received successfully. NCV and NCVD transmit more useful packets to the client even though the number of transmitted packets may be comparable to NCT. Application-level throughput is the most relevant notion of throughput in our context, as in [5].

MAC Throughput. For completeness, we also show that the MAC-layer throughput of NCV and NCVD is very close to that of NCT, which is specifically designed for maximum throughput. Fig. 8(c) shows the MAC-level throughput for the four schemes. As expected, NCT, NCV and NCVD all achieve higher MAC-level throughput than noNC; this is because all three schemes use network coding across streams, which increases the information content per transmission. [5] The second and more interesting observation is that NCV and NCT achieve almost the same MAC level throughput, for loss rates up to 9.4%, and NCT achieves slightly higher throughput for larger loss rates. The reason is that NCV and NCT use the same network code construction schemes but different network code selection mechanisms; therefore, the amount of data delivered at the MAC layer is almost the same for low packet loss rates. For higher packet loss rates, there are more packets with urgent deadline in the Tx queue. NCT transmits all possible side packets even if the their deadline is within one transmission time. NCV selects both primary and side packets considering the deadline and importance; therefore, NCV may purposely *not* choose the code with the largest number of packets, if one or more packets in this code are going to be useless at the receiver. Instead, NCV sends the most useful and effective packets to the client. In summary, NCV achieves the same MAC-level throughput with NCT for loss rates up to 9.4%, and slightly less for higher loss rates. However, and more importantly, NCV always achieves higher application-level throughput than NCT.

NCVD. A third observation from Fig.8(c) is that NCVD achieves not only the highest application-level throughput but also the highest MAC throughput for most loss rates. This is explained by the fact that NCVD looks into the entire queue and has more options to choose from, both in terms of video quality and in terms of absolute number of packets. For very high loss rates (above 20%) NCT achieves higher MAC throughput because NCVD prefers to optimize the code selection for the application level.

Small Queue Sizes. We looked at the buffer occupancy at the transmitter and at the clients and observed that they were small, in the order of 5-10 packets, for the simulated scenarios and delay budgets considered. We also observed that NCV and NCVD further reduces the queue size, compared to NCT and noNC. This is intuitively expected: NCV and NCVD deliver more packets successfully to the client, thus there are less packets waiting for transmission in Tx. Maintaining short buffers has several positive implications. First, short queues are good for bottlenecks shared with TCP. Second, having a small Tx queue allows to consider all packets in the queue for network codes and enjoy the performance gains of NCVD without increase in complexity, even for $d = \infty$. Finally, having small

[5]Note that we plot *effective* packet loss rate, after all possible retransmissions, while actual packet loss rate is higher. Therefore, there are several coding opportunities even for 1% effective loss rate. Note also that, for higher loss rates, the maximum throughput reduces to 170 kbps: the reason is that packets are dropped from the Tx queue when their deadline expires.

virtual buffers means that there are only a few overheard packets to consider in the construction of the network codes, which significantly decreases the complexity.

C. Complexity

NCV constructs candidate network codes and selects the one that maximizes video quality improvement. The complexity comes from considering *all* possible candidate codes. Selecting side packets among all possible subsets of overheard packets at the target node, is exponential in the size of the virtual buffer. However, the complexity of NCV is no worse than the complexity of NCT: they both consider all possible codes but they evaluate them using a different metric. Another important observation is that real-time delay requirements significantly reduce the number of packets in the virtual buffers and therefore the complexity; e.g. in our simulations, a delay budget of 100ms resulted in at most 5 packets in the virtual buffers. Thus, a brute-force approach is feasible for real-time applications.

For a larger delay budget, approximation algorithms for NCV and NCT can be developed. Conflicts between candidate side packets can be represented using a graph, whose vertices represent the overheard packets and edges represent their conflict. Each vertex has a weight corresponding to the expected improvement from including this packet into the code. Two vertices are connected through an edge if the corresponding packets cannot be used together in the same code. The problem of selecting side packets to maximize quality improvement is then reduced to selecting vertices that are not connected to maximize the total weight. This problem is known as the maximum weight independent set problem, and is the complement of the vertex cover problem. Although these are NP-complete problems, they are also well-studied and approximation algorithms have been developed [29].

NCVD runs NCV for each candidate primary primary in the Tx queue and selects the best overall code. The NCVD complexity is linear in the number of packets in the Tx queue, which is also small for real-time applications. The dominant part is still due to the NCV part.

VI. Conclusion

In this paper, we studied opportunistic listening and inter-flow coding for video streaming in wireless networks. In particular, we proposed schemes for network code selection that take into account video considerations, such as the distortion values and the playout deadlines of the video packets. Simulation results show that the proposed schemes improve video quality up to $3-5$dB compared to baseline schemes. Furthermore, they significantly improve the application-level throughput and achieve the same or similar levels of MAC throughput.

We are currently extending this idea in several directions, considering: multihop scenarios, scenarios with a mixture of video and data traffic, and more realistic simulations of the wireless network.

References

[1] N. Färber and B. Girod, "Wireless Video," in A. Reibman, M.-T. Sun (eds.), *Compressed Video over Networks*, M. Dekker, 2000.

[2] Special issue on "Advances in Wireless Video," *in IEEE Wireless Communications Magazine*, Aug. 2005.

[3] "The network coding webpage," *http://www.netcod.org*.

[4] C. Fragouli. J. Widmer and J.Y. LeBoudec, "Network coding: an instant primer," *in Proc. of ACM SIGCOMM CCR*, vol. 36(1), pp. 63-68, Jan. 2006.

[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical network coding," *in Proc. of ACM SIGCOMM*, vol. 36(4), pp. 243-254, Pisa, Italy, Sept. 2006.

[6] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard, "The Importance of Being Opportunistic: Practical Network Coding For Wireless Environments," *in Proc. of Allerton Conference*, 2005.

[7] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *in IEEE Trans. on Multimedia*, vol. 8(2), pp. 390-404, Apr. 2006.

[8] J. Chakareski, J. Apostolopoulos, S. Wee, W. Tan, and B. Girod, "Rate-Distortion Hint Tracks for Adaptive Video Streaming," *in IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15(10), pp. 1257-1269, Oct. 2005.

[9] M. Kalman and B. Girod, "Rate-Distortion Optimized Video Streaming with Multiple Deadlines for Low Latency Applications," *in Proc. IEEE Packet Video Workshop*, Irvine, CA, Dec. 2004.

[10] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *in IEEE Trans. on Information Theory*, vol. 46(4), pp. 1204-1216, July 2000.

[11] S.R. Li, R.W. Yeung, and N. Cai,"Linear network coding," *in IEEE Trans. on Information Theory*, vol. 49(2), pp. 371-381, Feb. 2003.

[12] B. Girod, J. Chakareski, M. Kalman, Y.J. Liang, E. Setton and R. Zhang, "Advances in Network-adaptive Video Streaming," *in Proc. of IWDC 2002*, Capri, Italy, Sept. 2002.

[13] S. Shankar, Z. Hu, and M. Van der Schaar, "Cross Layer Optimized Transmission of Wavelet Video over IEEE 802.11a/e WLANs," *in Proc. IEEE Packet Video 2004*, Irvine, CA, Dec. 2004.

[14] Special issue on "Multimedia over Broadband Wireless Networks," *in IEEE Network Magazine*, vol. 19(1), pp. 25-25, Jan.-Feb 2005.

[15] M. Van der Schaar, S. Shankar,"Cross-Layer Wireless Multimedia Transmission: Challenges, principles, and new paradigms," *in IEEE Wireless Comm. Magazine*, vol. 12(4), pp. 50-58, Aug. 2005.

[16] E. Setton, T. Yoo, X. Zhu, A. Goldsmith and B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," *in IEEE Wireless Comm. Magazine*, vol. 12(4), pp. 59-65, Aug. 2005.

[17] G.M. Su, Z. Han, M. Wu, and R. Liu, "Multi-user Cross-Layer Resource Allocation for Video Transmission over Wireless Networks," *in IEEE Network Magazine*, vol.20(2), pp.21-27, Mar-Apr 2006.

[18] Z. Li, B. Li, "Network coding: The case for multiple unicast sessions," *in Proc. of Allerton Conference*, Sept. 2004.

[19] Y. Wu, P.A. Chou, and S. Y. Kung, "Information exchange in wireless network coding and physical layer broadcast," *in Proc. of IEEE CISS*, Baltimore, MD, March 2005.

[20] T. Ho and R. Koetter, "Online incremental network coding for multiple unicasts," *in DIMACS WG on Network Coding*, Jan. 2005.

[21] S. Deb, M. Effros, T. Ho, D.R. Karger, R. Koetter, D.S. Lun, M. Medard, and N. Ratnakar, "Network coding for wireless applications: A brief tutorial," *in Proc. of IWWAN '05*, London, UK.

[22] A. Ramamoorthy, J. Shi, and R. Wesel, "On the capacity of network coding for wireless networks," *in Allerton*, 2003.

[23] Y. Liang, J. Apostolopoulos and B. Girod, "Analysis of Packet Loss for Compressed Video: Does Burst-Length Matter?," *in Proc. of IEEE ICASSP*, vol. 5, pp. 684-687, Hong Kong, Apr. 2003.

[24] A. Eichhorn, "Modeling Dependecny in Multimedia Streams," *in Proc. of the 14th annual ACM international conference on multimedia*, pp. 941-950, Santa Barbara, CA, 2006.

[25] A. Goldsmith, "Wireless Communications," New York, Cambridge University Press, 2005.

[26] Q. Zhang, S.A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *in IEEE Trans. on Communications*, vol. 47(11), pp. 1688-1692, 1999.

[27] ITU-T Rec. "H.264 : Advanced video coding for generic audiovisual services", 2005.

[28] H.264/AVC Reference Software Version JM 8.6 *http://iphome.hhi.de/suehring/tml/download/*, 2006.

[29] V. V. Vazirani, "Approximation Algorithms," Springer-Verlag 2003.