A Framework for Joint Network Coding and Transmission Rate Control in Wireless Networks

Tae-Suk Kim*, Serdar Vural*, Ioannis Broustis*,
Dimitris Syrivelis†, Srikanth V. Krishnamurthy*, Thomas F. La Porta‡
*University of California, Riverside, †University of Thessaly, ‡Penn State University
{*tskim, svural, broustis, krish*}*@cs.ucr.edu*, *jsyr@uth.gr*, *tlp@cse.psu.edu*

***Abstract*—Network coding has been proposed as a technique that can potentially increase the transport capacity of a wireless network via processing and mixing of data packets at intermediate routers. However, most previous studies either assume a fixed transmission rate or do not consider the impact of using diverse rates on the network coding gain. Since in many cases, network coding implicitly relies on overhearing, the choice of the transmission rate has a big impact on the achievable gains. The use of higher rates works in favor of increasing the native throughput; however, it may in many cases work against effective overhearing. In other words, there is a tension between the achievable network coding gain and the inherent rate gain possible on a link. In this paper[1], our goal is to drive the network towards achieving the best trade-off between these two contradictory effects. Towards this, we design a distributed framework that (a) facilitates the choice of the best rate on each link while considering the need for overhearing and (b) dictates the choice of which decoding recipient will acknowledge the reception of an encoded packet. We demonstrate that both of these features contribute significantly towards gains in throughput. We extensively simulate our framework in a variety of topological settings. We also fully implement it on real hardware and demonstrate its applicability and performance gains via proof-of-concept experiments on our wireless testbed. We show that our framework yields throughput gains of up to 390% as compared to what is achieved in a rate-unaware network coding framework.**

I. Introduction

Network coding (NC) brings the promise of providing significant throughput improvements in wireless networks [1]. However, in most cases it requires nodes to overhear native and/or encoded packets. In a wireless network, overhearing is directly dependent on the channel quality. In particular, more often than not, there may be a mismatch in the quality of (a) the link from the transmitter and its intended recipient and, (b) the link between the transmitter and the neighbor trying to overhear the packet. This mismatch leads to an inherent tension between the use of high transmission rates and providing an opportunity for network coding, as we discuss below.

To aid overhearing, one may simply use the lowest transmission rate that can successfully deliver packets to both the intended recipient and any overhearing node (or sniffer). While this can maximize the network coding gain, it may not yield the best throughput gain. On the other hand, transmitting packets at high rates may decrease the opportunities for encoding packets. We illustrate this with a simple example. Let us consider the topology depicted in Fig. 1; the undirected edges represent the feasible links among the 5 nodes. Jack is an intermediate router who relays packets from Alice and Bob to Chloe and Dave, respectively. To begin with, assume that (i) all the nodes operate in the promiscuous mode (as in [1]), (ii) they use the basic (or lowest), fixed transmission rate R_f, and (iii) all links have a Packet Delivery Ratio (PDR) equal to 1 at this rate R_f. With traditional network coding, Dave is able to overhear Alice's packet A sent to Jack. Similarly Chloe can overhear Bob's packet B, transmitted to Jack. Jack applies a linear encoding function on the two packets (e.g. an XOR operation) to construct the packet $A \oplus B$ and transmits it at rate R_f. Using their overheard packets, Chloe and Dave can decode the packet $A \oplus B$ and thus obtain A and B, respectively.

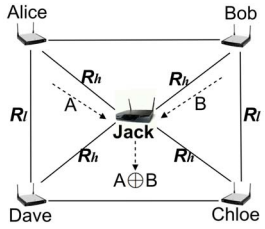

Fig. 1. Selecting high transmission rates may impede the network coding gain. Here, Alice chooses to use a high transmission rate to send her packet A to Jack. It is possible that Dave does not overhear packet A, and hence he cannot decode Jack's $A \oplus B$ encoded packet.

Rate adaptation has a direct effect on NC gain: Now assume that a typical rate adaptation algorithm is used [2], [3], [4]. With this, Alice may use a rate $R_h > R_f$, towards achieving a better performance on the link <Alice→Jack>. However, R_h may be inappropriate on the link <Alice→Dave>; in such a case, it is unlikely that Dave successfully overhears packet A. Thus, although the use of R_h by Alice provides the highest throughput on the link <Alice→Jack>, it also degrades the network coding gain in the long term. If Alice decides on a lower transmission rate R_l ($R_f < R_l < R_h$) with which Dave can overhear packet A, then he will be able to decode the $A \oplus B$ packet sent by Jack. Hence, by settling for a lower *rate gain*, the network enjoys a higher *coding gain*; this, however, may either increase the throughput due to the benefits of coding, or decrease it due to the use of the lower transmission rate.

The decision on which recipient should acknowledge an encoded packet affects the total throughput: Jack will pseudo-broadcast [1] packet $A \oplus B$ to Chloe and Dave. The pseudo-broadcast requires Jack to choose a primary receiver which will send a MAC layer ACK for the encoded packet; we call this receiver the ACKer. The other node acts as a secondary receiver that overhears the packet. For simplicity, assume that (i) Jack sends packet $A \oplus B$ at rate R_h, (ii) the

[1]Prepared partially through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. This work was also partially supported by the US Army Research Office under the Multi-University Research Initiative (MURI) grant W911NF-07-1-0318.

PDR on link <Jack→Dave> is 1, while the PDR on link <Jack→Chloe> is 0.1 at rate R_h, (iii) Jack knows that Dave and Chloe have overheard packets A and B, respectively and, (iv) using reports sent by Chloe and Dave (as with COPE [1]), Jack can determine which packets were received correctly by Chloe and Dave (perhaps after a delay). Intuitively, it would seem that the right choice is for Jack to choose Chloe to be the ACKer. With this, Jack would confidently assume that if Chloe has received the packet, then Dave has, too. In this case, Jack is *expected* to send $A \oplus B$ ten times (since PDR=0.1 on the link with Chloe). By the time Chloe receives her packet, Jack would have delivered only 2 packets (A to Chloe and B to Dave). Let us now consider the case where Jack chooses Dave to be the ACKer. In this case, Jack just transmits $A \oplus B$ once. Later when Jack receives Chloe's report, he will need to retransmit A if she did not get the packet with the single transmission. Jack would then encode packet A again, but with a different, new packet B_2 destined to Dave. Again, on average Jack will have to perform 10 transmissions of A before Chloe receives it. Note however, that now Jack is able to deliver a new packet with each transmission instance to Dave. In other words, in 10 transmissions, Jack delivers one packet to Chloe but 10 packets to Dave. Thus, contrary to our initial expectation, choosing Dave to be the ACKer results in a higher throughput. This trivial example demonstrates that choosing the ACKer for the encoded packet intelligently could have a significant impact on the long-term throughput.

Previous studies on wireless network coding take advantage of the broadcast nature of the medium, and make use of low bit rates in order to increase the network coding gain at wireless routers. However, this approach is conservative; high coding gains are possible at higher bit rates, as long as the PDRs at those rates are sufficiently high. In addition, these studies do not examine clever ACKer selection. Unlike such efforts, in this paper, we consider the two practical challenges described above. Our key contributions are the following:

1. We design a framework for adapting the transmission rate to achieve good trade-offs between coding gain and rate gain: We design and implement a novel distributed rate adaptation framework that considers the potential gains due to network coding. Our framework consists of two modules:

- ***The local transmission rate selection module:*** Every transmitter of a native packet (e.g. Alice in our example) chooses the bit rate such that: (a) a high link-level throughput on the link to the selected router (e.g. link <Alice→Jack>) is achieved, and at the same time, (b) the sniffers successfully overhear the packet with high probability.
- ***The ACKer selection module:*** Every router intelligently selects a bit rate and an ACKer for each encoded packet. The goal is to to maximize the sum of the expected throughputs at all the intended recipients. For this, the router takes into account the probability of successful decoding of the encoded packet for each intended recipient.

2. We perform extensive evaluations of our framework: Our goal is to evaluate our framework in comparison with the popular network coding scheme COPE [1], the design of which does not consider coding-aware bit rate selection. We implement our framework in Click, on top of the COPE implementation [5]. We perform experiments on our wireless testbed, both indoors and outdoors. We observe that our scheme provides practical performance benefits: it improves the total network throughput by up to 270%, as compared to COPE. The primary limitation of the above experimentation is that it is with 802.11b (which supports only 4 rates). To examine the benefits of our approach with 802.11a (with 8 rates) and in larger settings, we also perform extensive NS-2 simulations. We consider different node densities and network coding scenarios. Our simulations demonstrate that with 802.11a, our scheme can outperform COPE by up to 390%.

Our work in perspective: The design of the framework that we propose in this paper is built on top of a prior practical wireless network coding architecture, COPE. However, as we discuss in Section V, our framework is applicable with other network coding architectures as well. In addition, in this paper we focus on local NC topologies; encoded packets traverse at most one hop. As we discuss in Section V, this has applications in wireless LANs.

The rest of this paper is structured as follows. In Section II we discuss previous studies on network coding and differentiate our work. In Section III we present the design of our framework. In Section IV we evaluate our framework through simulations and experiments. In Section V we discuss the scope of our study. Finally, our conclusions form Section VI.

II. Related Work

In this section, we discuss previous relevant NC studies and differentiate our work.

Analytical and simulation work on wireless network coding: Lun et al. [6] show that the problem of minimizing the communication cost can be formulated as a linear program and solved in a distributed manner. Chaporkar and Proutiere [7] show that in multi-rate settings, systems with network coding may actually have smaller throughputs than without coding. They argue that unless appropriate scheduling is applied, network coding may lead to performance degradation in many scenarios. They propose XOR-Sym, a computationally simple network coding technique that can be applied on symmetric routes. With XOR-Sym, packets are decoded at destinations only and not at intermediate nodes along a path. Scheuermann et al. [8] propose noCoCo, a deterministic scheduling scheme for network coding to operate on two-way multihop traffic flows. Seferoglu et al. [9] propose code selection schemes that consider the properties of video traffic. Le et al. [10] provide an upper bound on the number of packets that can be coded together, in any possible coding structure. However, *none of these studies jointly address bit rate selection and network coding towards improving the long-term performance.*

The studies that are mostly relevant to our work are [11] and [12]. Liu and Xue in [11] consider network coding for two-way relaying in a three-node network. They analytically characterize the achievable rate regions for a basic 3-node topology and find the theoretically optimal end-to-end sum rates. However, unlike in our study, they consider only a certain topology where nodes do not need to overhear any packets. Vieira et al. [12] perform simulations to observe how the combination of network coding and bit rate diversity affects the performance of practical broadcasting protocols. They show that it is possible for multi-rate link layer broadcasts and network coding to jointly increase the network throughput in multicast applications. However, they do not propose any bit rate selection algorithm that considers the benefits due to network coding. Further, they do not perform experiments in real settings to observe these effects.

Seferoglu and Markopoulou in [13] provide an understanding of the interplay between application data rate control and

network coding. However, they do not consider the effects of rate adaptation on the network coding efficiency. Seferoglu et al. [14] extend the work in [13] by comparing the optimal application data rate adaptation scheme of [13] to the corresponding optimal coding-unaware scheme. However, they also do not examine the role of bit rate adaptation on coding gain.

Experimental studies on wireless coding: Katti et al. [1] propose COPE, the first implementation of wireless network coding. Their design requires each node to inform the relay about the packets that it has overheard and stored. Since one of the goals of COPE is to increase the number of encoding opportunities, low transmission rates as favorable in order for native packets to be overheard by as many neighbors as possible. The experiments with COPE on a wireless testbed show that even with very simple encoding operations, network coding can provide significant capacity gains. The authors of [1] also study the interactions of network coding with the routing and the higher layer protocols. We refer to COPE later in this paper, since we build our algorithm on top of the design and the implementation of COPE. Rozner et al. in [15] present ER, a scheme that adopts the design of COPE and employs network coding to perform efficient packet retransmissions. With ER, packets that need to be retransmitted are coded together, such that one retransmission can recover multiple packet losses. The authors show that the problem of selecting the optimal set of packets to code together is NP-hard; they propose a set of heuristics that can be followed to make coding decisions. Rayanchu et al. [16] propose CLONE, a suite of algorithms for NC that take into account the potential losses on wireless links. Both [15] and [16] follow COPE's logic regarding the selection of a transmission rate. MORE [17] is a routing protocol for static mesh networks, which performs a random mixing of packets, right before they are forwarded. However, no decisions on rate selection for native or encoded packets are made. In MIXIT [18] the idea is to code *symbols* rather than packets. As with MORE, batches of packets are coded together. *All of these studies are transmission rate unaware. Towards maximizing the coding gain, they implicitly assume that devices use low transmission rates, in order to guarantee that packets are overheard with a high probability.*

Finally, a large body of studies have investigated network coding for wireline networks [19], [20], [21], [22], [23], [24], [25], [26]. These studies, although seminal, do not consider the inherent properties of the wireless medium.

III. Designing our Framework

In this section, we first provide a simple analysis for estimating the total throughput with NC for the topology in Fig. 1[2] We use this analysis to subsequently derive the generic design of our coding aware rate control framework.

A. Throughput estimation with local network coding

Let us return to the example of Fig. 1, where Alice wishes to send a packet to Chloe and Bob wants to send a packet to Dave. To begin with, we make the following assumptions:

1) Alice and Bob use Jack as their only relay to send packets A and B to Chloe and Dave, respectively.
2) A rate set R is available for packet transmissions, and each rate in R has an associated PDR (packet delivery ratio).
3) A scheduling process is assumed [7], [13], [14], according to which packets A and B arrive at Jack before the latter transmits any of these two packets.

Preliminaries: For the purposes of our analysis we define the following notation:

- $P^{r_{Alice}}_{Alice,Jack}$ is the PDR on the link <Alice→Jack>, at bit rate r_{Alice}.
- $P^{r_{ACK}}_{Jack,Alice}$ is the PDR for ACK packets sent in the reverse direction <Jack→Alice>.
- $P^{r_{Alice}}_{\{Alice,Jack\},Dave}$ is the probability that Dave overhears Alice's packet towards Jack (at rate r_{Alice}).
- $N^{r_{Alice}}_{Alice,Jack}$ is the expected number of transmissions needed for successful reception of packet A from Alice to Jack sent at rate r_{Alice}.
- L_{Alice} is the length of the packet that Alice sends, in bits.
- $T^{r_{Alice}}_{L_{Alice}}$ is the transmission time of the packet with length L_{Alice}, transmitted at bit rate r_{Alice}.

The probability that a transmission from Alice is successfully received and acknowledged by Jack is $P^{r_{Alice}}_{Alice,Jack} \cdot P^{r_{ACK}}_{Jack,Alice}$. The successful transmission of a packet follows a Bernoulli process. Given this, the expected number of transmissions of a packet sent by Alice is[3]:

$$N^{r_{Alice}}_{Alice,Jack} = \frac{1}{P^{r_{Alice}}_{Alice,Jack} \cdot P^{r_{ACK}}_{Jack,Alice}}.$$

As shown in Fig. 1, Dave has a link to Alice. The probability that Dave overhears packet A is given by:

$$P^{r_{Alice}}_{\{Alice,Jack\},Dave} =$$

$$\sum_{i=1}^{\infty}\{(1-P^{r_{Alice}}_{Alice,Jack}\cdot P^{r_{ACK}}_{Jack,Alice})^{i-1}\cdot P^{r_{Alice}}_{Alice,Jack}\cdot P^{r_{ACK}}_{Jack,Alice} \cdot\sum_{j=1}^{i}(1-P^{r_{Alice}}_{Alice,Dave})^{j-1}\cdot P^{r_{Alice}}_{Alice,Dave}\}, \quad (1)$$

where $\sum_{j=1}^{i}(1-P^{r_{Alice}}_{Alice,Dave})^{j-1}\cdot P^{r_{Alice}}_{Alice,Dave}$ is the probability that Dave successfully overhears a packet from Alice, when the latter performs exactly i transmissions. The expected number of bits delivered together to Chloe and Dave is:

$$L_t = P^{r_{Jack}}_{\langle Jack,ACKer\rangle,Chloe}\cdot P^{r_{Bob}}_{\{Bob,Jack\},Chloe}\cdot L_{Alice} + P^{r_{Jack}}_{\langle Jack,ACKer\rangle,Dave}\cdot P^{r_{Alice}}_{\{Alice,Jack\},Dave}\cdot L_{Bob}\ ,$$

where $P^{r_x}_{\langle x,ACKer\rangle,y} = \sum_{i=1}^{\infty}(1-P^{r_x}_{x,y}\cdot P^{r_{ACK}}_{y,x})^{i-1}\cdot P^{r_x}_{x,y}\cdot P^{r_{ACK}}_{y,x}$ if node y is ACKer, and $P^{r_x}_{\langle x,ACKer\rangle,y} = P^{r_x}_{\{x,z\},y}$ if node z is ACKer. The above expression is obtained based on the requirement that L_{Alice} bits are delivered to Chloe only if she can overhear Bob's packet; similarly, L_{Bob} bits are delivered to Dave only if he can overhear Alice's packet. The total expected duration for delivering the packets from the sources to the destinations (from the point of inception until Jack receives an ACK from ACKer) is:

$$D_t = N^{r_{Alice}}_{Alice,Jack}\cdot T^{r_{Alice}}_{L_{Alice}} + N^{r_{Bob}}_{Bob,Jack}\cdot T^{r_{Bob}}_{L_{Bob}}$$

[2]This topology represents the simplest case where network coding requires some form of overhearing. Our analysis can be easily extended to more complex local scenarios.

[3]We assume that Alice continues to retransmit the packet until an acknowledgment is received from Jack.

$$+ N^{r_{Jack}}_{Jack,ACKer} \cdot T^{r_{Jack}}_{max(L_{Alice},L_{Bob})} \quad .$$

Then, the expected throughput with network coding enabled is:

$$\Theta_{NC} = \frac{L_t}{D_t}. \tag{2}$$

Our objective: We wish to (a) determine the bit rates at the transmitters (Alice, Bob and Jack) and (b) the choice of the ACKer, such that the expected throughput, computed as in Eq. (2), is maximized. Computing the throughput for the more general case where, n transmitters all use the same relay (Jack) is a challenge. This is because, in such a scenario the candidate rate set is $|R|^n$; it is evident that this rate set grows exponentially with n. In addition, since one can only use rates from among a discrete set, maximizing the expected throughput given our objectives, is an integer programming problem, which in turn is known to be NP-Hard [27].

We also wish to point out that the calculation of Eq. (2) requires each node to have an omniscient view of the state of traffic in the network. In practice however, individual nodes (Alice and Bob) cannot a priori have such a view. In other words, when Alice is about to transmit packet A, she has no idea on whether or not Bob has a packet to send. In cases where Jack only has Alice's packets, he may not wait for Bob's packets to perform network coding; stated otherwise, in such cases, network coding does not take place.

Given these complexities, we instead design a simple distributed heuristic-based rate and ACKer selection algorithm (described below).

B. Our proposed approach

We build our framework on top of a previous, popular coding architecture, COPE. We employ the core coding functionalities of COPE; in particular we leverage reception reports and information exchange among nodes that allow each node to learn about its neighbor states [1] (we provide implementation details in Section IV). Unlike with COPE however, with our proposed framework, each transmitter of native packets, accounts for the quality of the links to potential sniffers while choosing the transmission rate. Stated otherwise, the transmission rate is chosen such that there is a high probability of successful overhearing at the sniffers. The framework is fully distributed and requires each node to only be aware of local information[4].

Composition of our framework: Our proposed framework consists of two component modules.

The local transmission rate selection module: The first module is a *local transmission rate selection* module which primarily operates on native packets. Alice identifies all her one-hop neighbors that are also neighbors of Jack (see Fig. 1); these neighbors are the sniffers. Then, Alice selects the rate that maximizes the expected throughput to Jack, such that the probability of successful overhearing of her packet A, at each of the common neighbors, is greater than a threshold β.

In order to illustrate how this threshold β is determined let us consider the general case where $2N$ nodes share a single relay. These $2N$ nodes can overhear each other when the basic rate is used; at the other rates, this may not necessarily be the case. Let there exist $n(\leq N)$ native packets that are to be exchanged among n distinct pairs of nodes. We denote each source by S_i and its corresponding destination by D_i $(1 \leq i \leq n)$. Given the topology, there is an opportunity for the relay to perform network coding. In our simple example, $N = 2$ and Jack is the relay. Alice and Bob can be mapped to S_1 and S_2 and Chloe and Dave to D_1 and D_2. With our notation for packet overhearing probability in Section III-A, the probability that D_i overhears a packet sent by S_j, with rate r_{S_j} (to D_j) is $P^{r_{S_j}}_{\{S_j,D_j\},D_i}$. In order for D_i to successfully decode its own packet, it should overhear the packets transmitted by each such source S_j. Since the probability that D_i overhears S_j is independent of whether or not it overhears S_l (for $l \neq i, j$), the probability that D_i can successfully decode a received encoded packet to retrieve its own packet, $P_{dec(D_i)}$ is given by:

$$P_{dec(D_i)} = P^{r_{S_1}}_{\{S_1,D_j\},D_i} \times \cdots \times P^{r_{S_{i-1}}}_{\{S_{i-1},D_j\},D_i}$$

$$\times \; P^{r_{S_{i+1}}}_{\{S_{i+1},D_j\},D_i} \times \cdots \times P^{r_{S_n}}_{\{S_n,D_j\},D_i}. \tag{3}$$

Existing network coding systems including COPE, generally allow the encoding of native packets only if the decoding probability at a destination (say D_i) is higher than a certain pre-specified threshold ϑ (0.8 for COPE). In the same spirit, our goal is then to select a transmission rate for each native packet such that the decoding probability at each destination of an encoded packet is greater than or equal to ϑ. However, as one can readily see, one will need to consider all possible rates (beginning with the highest) to determine the best set of rates which satisfy the requirement that this decoding probability requirement is met at all the destinations. This determination takes an exponential number of steps since the product has to be computed for all combinations of rates.

Our approach towards simplifying this requirement is driven by a simple observation. $P_{dec(D_i)}$ is no greater than the minimum of the probabilities $P^{r_{S_j}}_{\{S_j,D_j\},D_i}$, $1 \leq j \neq i \leq n$. Thus, we require that each probability is higher than a certain threshold i.e., we try to maximize this minimum probability. Hence, we set $\vartheta = \beta^n$ and require that each term in the product be at least β. To explain the intuition behind this choice, assume that the probability of D_i correctly overhearing a packet from a source S_j is β. If the overhearing probability from a different source S_l is lower than β then the first condition does not help in the long term; the likelihood of correct decoding is dictated by the ability to decode on the poorer link (between S_l and D_i).

With this requirement, each source (for example Alice) attempts to select the rate that maximizes the expected throughput on the link to the relay (Jack). For our simple example, this translates to the following optimization problem:

$$\begin{aligned} \max_{r_{Alice} \in R} \quad & \frac{L_{Alice}}{N^{r_{Alice}}_{Alice,Jack} \cdot T^{r_{Alice}}_{L_{Alice}}} \\ s.t \quad & P^{r_{Alice}}_{\{Alice,Jack\},Dave} \geq \beta \\ & P^{r_{Alice}}_{\{Alice,Jack\},Bob} \geq \beta. \end{aligned} \tag{4}$$

The ACKer Selection Module: Once Jack obtains the packets from Alice and Bob, he encodes the packets and pseudo-broadcasts the encoded packet to Chloe and Dave. The pseudo-broadcast requires Jack to choose a primary receiver who will transmit a MAC layer acknowledgment for the packet; the other (secondary) receiver simply overhears the transmission. In order to increase the possibility that both Chloe and Dave receive

[4]Note that in the rest of the paper we relax our assumption *3*, in Section III-A as per which, a scheduling process is assumed.

the encoded packet, Jack has to choose his primary receiver intelligently. Towards this, he performs the following steps.

First, Jack will have to determine if Chloe and Dave have indeed overheard the transmissions of the native packets from Bob and Alice. Towards this, we utilize a functionality that is similar to that with COPE. Chloe and Dave periodically send *reception reports* wherein they explicitly indicate if they have in fact overheard packets from Alice and Bob. If such reports are unavailable, our system simply tries to estimate the probability that Dave and Chloe have overheard the requisite packets; this can be done if Alice and Bob indicate the probabilities of successful reception on the links <Alice→Dave> and <Bob→Chloe> for the transmission rates that they have chosen. With this, Jack can compute the decoding probabilities at both Chloe and Dave. If these decoding probabilities are greater than or equal to ϑ, Jack decides to encode the two native packets from Alice and Bob. Note that at some high rates this requirement may not hold; if that is the case, Jack simply forwards the native packets.

Second, Jack unicasts the packet to a primary receiver (say Dave), expecting the other receiver (Chloe) to overhear the transmission. The primary receiver (ACKer) will send a MAC layer ACK to Jack upon the successful reception of the encoded packet; if the ACK is not received, Jack performs retransmission(s). As discussed in Section I, the choice of the primary receiver can have a significant impact on the achievable throughput. Soliciting an ACK from the receiver to which Jack has a *poorer* link may increase the chances that all other recipients have received the encoded packet. However, as seen with our earlier example (Section I), such a decision may not provide high throughput benefits. In that simple example, it was advantageous for Jack to choose the receiver with the better link to be the ACKer. However in more complex settings, the choice may not be as simple. The high level goal of Jack is to maximize the sum of the expected throughput on Chloe and Dave. Thus, Jack finds the solution to the following problem:

$$\max_{\substack{r_{Jack}\in R \\ ACKer\in\{Chloe,Dave\}}} \frac{L'_t}{D'_t} \tag{5}$$

$$\text{where} \quad L'_t = P^{r_{Jack}}_{\langle Jack,ACKer\rangle,Chloe} \cdot P^{r_{Bob}}_{Bob,Chloe} \cdot L_{Alice} + P^{r_{Jack}}_{\langle Jack,ACKer\rangle,Dave} \cdot P^{r_{Alice}}_{Alice,Dave} \cdot L_{Bob}$$

$$\text{and} \quad D'_t = N^{r_{Jack}}_{Jack,ACKer} T^{r_{Jack}}_{max(L_{Alice},L_{Bob})} \quad .$$

Stated otherwise, Jack finds whether the throughput is maximized when Chloe is chosen to be the primary receiver (or ACKer as we call this node) or when it is Dave instead; it then chooses either the appropriate recipient to be the ACKer.

Complexity considerations: Even though the problems defined above (with the two modules of our framework) are discrete optimization problems, the solution space is small. For problem (4) the dimension of the solution space[5] is $|R|$ and for the problem defined in (5) the dimension is $|R| \times 2$. These claims are easily verified by simply examining the structure of the formulated problems. In the first case, one simply needs to examine the throughput with each available rate. In the second case, one needs to examine the throughput with each rate, for each potential node that can be the ACKer.

[5]For example, $|R|$ is 4 for 802.11b, 8 for 802.11a, and 12 for 802.11g.

In the general case, where there are n source-destination pairs, the dimensionality of the first problem is still $|R|$ and that of the second problem is essentially $|R| \times n$. If there is no feasible solution for the problem defined in (4), the basic rate is selected.

We summarize the two components of our framework together, as pseudo-code in Algorithm 1.

```
1  Initialization: tx_rate ← basic rate and max_thr ← 0;
2  At each node p;
3  if a native packet for transmission then
4      if MAC dst q is not the same as final dst then
5          Search for the common neighbor set Φ with q;
6          for j ← 1 to |R| do
7              if r_j satisfies P^{r_j}_{{p,q},π} ≥ β, ∀π ∈ Φ then
8                  Calculate tmp_thr = L_p/N^{r_j}_{p,q} T^{r_j}_{L_p};
9                  if tmp_thr > max_thr then
10                     tx_rate ← r_j;
11     else
12         for j ← 1 to |R| do
13             Calculate tmp_thr = L_p/N^{r_j}_{p,q} T^{r_j}_{L_p};
14             if tmp_thr > max_thr then
15                 tx_rate ← r_j;
16 else
17     // p has an encoded packet to transmit
18     Find the set Ω of MAC destinations of packets XORed;
19     for j ← 1 to |R| do
20         for i ← 1 to |Ω| do
21             Calculate tmp_thr =
               Σ_i P^{r_j}_{⟨p,ACKer⟩,i} · P^{r_1}_{1,i} ··· P^{r_{i-1}}_{i-1,i} ·
               P^{r_{i+1}}_{i+1,i} ··· P^{r_{|Ω|}}_{|Ω|,i} · L_i/N^{r_j}_{p,i} T^{r_j}_{max(L_1,···,L_{|Ω|})};
22             if tmp_thr > max_thr then
23                 tx_rate ← r_j;
24                 ACKer ← i;
```

Algorithm 1: Algorithmic representation of our framework.

IV. Evaluating Our Framework

In this section, we assess the efficiency of our framework via simulations and proof-of-concept experiments, in various topologies and with diverse traffic characteristics.

A. Simulations

We simulate both COPE and our framework with the NS-2 simulator [28].

Simulation design and set-up: We create topologies of various densities and node populations. We consider 802.11a, which supports 8 bit rates (6, 9, 12, 18, 24, 36, 48 and 54 Mbps). We set a fixed transmission power of 18 dBm at every node; the ambient noise is fixed at 1.5e-10 mW. The signal attenuation is modeled by the 2-ray ground propagation model. All simulations consider statically placed nodes. We use fully saturated UDP traffic, where the packet size is set to 1500 bytes. We repeat each experiment 40 times. The exchange of RTS/CTS messages is disabled as with COPE [1].

The efficiency of our framework in small-scale topologies: First, we wish to assess the performance benefits with our framework in topologies with very diverse link qualities. We perform simulations on an 'X' topology, as in Fig. 1, where the link <Jack→Chloe> varies, but is always poorer than the link

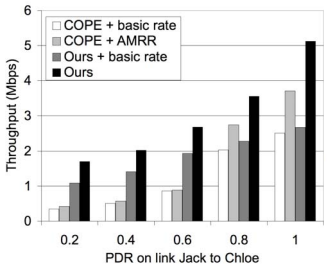

Fig. 2. Our framework provides the highest throughput gains in topologies with diverse link qualities.

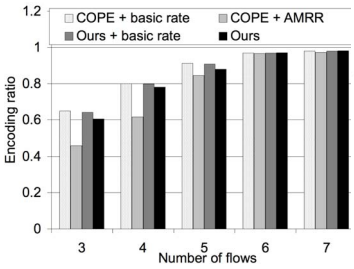

Fig. 3. The coding gain is maintained at high levels with our coding-aware rate selection module.

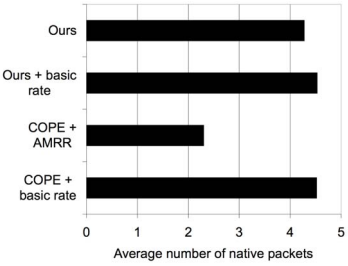

Fig. 4. Our framework enables the router to code of many native packets into every new encoded packet.

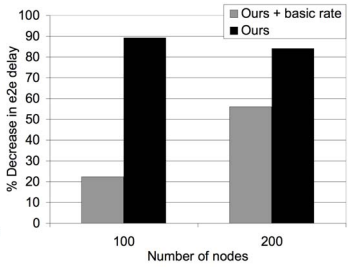

Fig. 5. Our framework achieves a lower end-to-end delay in multi-hop, multi-rate, large-scale topologies.

<Jack→Dave>. We consider the following cases: (i) COPE-basic where, COPE operates at the basic transmission rate (6 Mbps), (ii) our framework with and without our rate adaptation module enabled, and (iii) COPE-amrr, where COPE operates in conjunction with the AMRR rate adaptation algorithm [3]. The total network throughput results are plotted in Fig. 2. A direct observation is that our framework provides up to 305% throughput improvement as compared to COPE-amrr and up to 390% as compared to COPE-basic, depending on the quality of the link <Jack→Chloe>. More specifically:

1. In cases with very poor link qualities, rate unawareness can severely impact the performance: Towards increasing the total throughput, whenever there is a large difference in the PDR among the candidate ACKers, it is better to select an ACKer with good link quality (as discussed earlier). We observe that even if our framework selects the basic rate, the throughput improvements due to selecting the appropriate ACKer is significant: our framework at the basic rate outperforms COPE by 390% (Fig. 2) in terms of the average throughput. The use of AMRR by COPE does not improve performance significantly, since AMRR is coding unaware; it only targets improvements in the local link throughput. This interacts poorly with the random selection of the ACKer with COPE. In our example, whenever COPE picks Dave as the ACKer, Jack selects a high transmission rate (as per AMRR); with this, Chloe cannot receive the encoded packet.

2. High link qualities also favor our framework: In Fig. 2, we observe that when the PDR on all links is high (e.g. 0.8 or higher), our rate selection module boosts the total network throughput, by as much as 75% and 30%, compared to COPE-basic and COPE-amrr, respectively. We also observe that in such cases, the ACKer selection does not significantly impact throughput; this is because with good links, COPE also selects a good ACKer. Unlike COPE however, our framework provides high rate gains on the high quality links; high coding gains are maintained with intelligent rate selection. As a consequence, our framework outperforms COPE-basic and COPE-amrr by 75% and by 30%, respectively.

The case for dense, "wheel" topologies: Next, we simulate wheel topologies, where larger sets of nodes share a common router. A wheel topology is an extended 'X' topology with more transmitters. As per [10], the maximum number of packets encoded together typically cannot be more than 5 with COPE. Thus, here we simulate topologies with up to 6 source-destination flows and with randomly set link qualities. Our goal here is to evaluate our framework in terms of the achieved encoding ratio, i.e., the ratio of the encoded packets at the output of the router to the total number of packets sent by the router. Our simulation results are depicted in Fig. 3. First, we wish to point out that COPE-basic is expected to provide the best encoding ratio, since its design is geared towards achieving high coding gain. In Fig. 3, we observe the following:

1. Our framework provides encoding ratios similar to that with COPE: We observe that our framework, when operating at the basic rate, provides the same ratio as COPE in all the considered scenarios. This is somewhat expected, since the use of a low bit rate aids coding opportunities. We observe that when our rate selection module is enabled, our framework still maintains an encoding ratio that is very close to that of COPE. We verify this finding in a real setting and discuss the observation in the following subsection.

2. COPE-amrr performs poorly in terms of the encoding ratio with few flows: We observe that with few active flows, the combination of COPE and AMRR leads to poor encoding ratios. This accentuates our earlier observations with the 'X' topology, with regards to COPE-amrr; again, this is due to the fact that AMRR is coding unaware. Clearly, one may expect the same behavior with other rate adaptation algorithms [2], [4] that do not consider network coding.

3. Our framework favors the inclusion of more packets into an encoded packet as compared to COPE-amrr: As seen in Fig. 3, with a larger number of flows COPE-amrr provides high encoding ratios. However, the ratio itself is agnostic to the population of native packets that are embedded into an encoded packet. Fig. 4 shows the average number of native packets combined together to form an encoded packet for the 6-flow scenario of Fig. 3. We observe that our framework enables the router to code nearly as many packets as COPE encodes at the basic rate[6]. In contrast, COPE-amrr does not allow the router to code as many packets together.

The potency of our framework in large-scale multi-hop settings: Finally, we perform extensive simulations on two large-scale, multihop topologies, which consist of 100 and 200 nodes respectively. The nodes are randomly and uniformly distributed across a 1000×1000 m^2 square region. We want to evaluate the efficacy of our framework in scenarios with many interfering nodes, and with flows that span a large number of intermediate hops. Throughout these simulations we randomly select source-destination pairs and we initiate fully-saturated UDP flows. Paths are established using the DSR protocol [29].

Our simulation results are plotted in Fig 5. We observe that the application of our framework leads to a significant reduction in the average end-to-end delay, i.e., by as much as 90%

[6]Recall that this observation is aligned with the findings in [10] with regards to the maximum number of native packets that can be combined together with COPE.

compared to COPE. This is due to the combined functionalities of our rate and ACKer selection modules. From Fig. 5 we make the following observations:

1. Our rate selection module provides a dramatic delay reduction in sparser topologies: In such topologies, the levels of interference are not that high; this allows the use of higher transmission rates with our framework. As a result, packets travel faster towards their destinations, thereby suffering lower end-to-end delays as compared to COPE.

2. Our ACKer selection module is mostly beneficial in dense topologies: With increased node density (e.g. the 200-node topology in Fig. 5), the levels of interference are higher. This reduces the probability of successful packet overhearing and reception, thereby incurring lower, diverse delivery probabilities. As we observe in Fig. 5, the strategy "Ours + basic", which has only the ACKer selection, drastically decreases the delay by itself; this demonstrates that the ACKer selection module reduces the delay to a larger extent with the 200-node topology (as compared to the 100-node topology).

B. Implementation and experiments

Next we describe the proof-of-concept implementation of our framework on real hardware. Subsequently we discuss the experiments with our implementation on our wireless testbed.

Testbed description and experimental set-up: We conduct our experiments on a 22-node wireless testbed deployed on the 3rd floor of the Engineering Building Unit II, at UC Riverside. A unique characteristic of the testbed is that it has both indoor and outdoor links; a pictorial representation is shown in Fig. 6. The nodes are based on the Soekris net5501 hardware configuration [30], and run a Debian Linux distribution with kernel v2.6.16.19 over NFS. Each node is equipped with a 500 MHz CPU, 512 Mbytes of memory, and a WN-CM9 wireless mini-PCI card, which carries AR5213 Atheros as the main chip. Every card is connected to a 5 dBi gain external omnidirectional antenna. As discussed in Section V, we repeat a part of our measurements with different hardware in order to validate our findings. We conduct experiments with groups of nodes that form two kinds of topologies: (a) the "cross topology" [1] and (b) the 'X' topology (as in Fig. 1). The relative locations of the nodes with both topologies are shown in Fig. 1; note that in the cross-topology experiments, both nodes of a session exchange packets (i.e., Chloe both sends packets to and receives packets from Alice; Dave similarly exchanges packets with Bob). We experiment with the 802.11b mode of operation, which supports 4 bit rates (1, 2, 5.5 and 11 Mbps); we elaborate on this choice in Section V. Our experiments are performed late at night in order to avoid interference from the collocated campus WLANs. All devices set their transmission powers to 10 dBm. We use fully saturated UDP traffic, where the default data packet size is 1500 bytes. We have disabled RTS/CTS messages, as in [1].

The implementation of our framework: We implement our framework in Click v.1.4.2 [31], as in [5]; this allows us to have a fair comparison against COPE in terms of software efficiency. We use the Madwifi-2005 wireless driver, which is fully compatible with the COPE implementation[7]. We build our framework on top of the publicly available COPE implementation [5]. Note that with this implementation, COPE by default operates at the basic rate of 1 Mbps. We adopt all the Click network coding elements of COPE, which realize the encoding and decoding engines, the construction of an additional packet header, and the functionality that handles the transmission of periodic reception reports. We also use the probing mechanism of the Roofnet routing protocol, SRCR [2]. However, since our framework requires information regarding the PDR at every bit rate, we leverage the ETT framework (instead of ETX as with COPE [1]) for routing [32]. In particular, each node periodically transmits 1500-byte probes and 60-byte probes at every rate. Nodes gather link quality information from each of their neighbors with regards to the PDR on every link (the ratio of probes that were received in both the forward and reverse directions). This PDR information is used by our framework (see Section III) to determine the bit rate and the ACKer that locally maximize the expected total throughput.

Scenario 1: Experiments with the 'X' topology with high quality links: First, we perform experiments on a 5-node 'X' topology, where the PDR on any of the links is above 70% at any transmission rate. As discussed earlier, routers in the 'X' topology can encode up to 2 native packets together. This topology consists of nodes 12, 20, 24, 26, 28. The nodes all have line-of-sight links to each other (as seen in Fig. 6). Our experiments consider different combinations of source-destination pairs and different routers. Our goal here is to quantify the efficiency of our framework in a realistic "good-channel" environment, relative to COPE. For this, we perform 30 experiments, alternating between the use of COPE and our framework. Fig. 7 depicts the throughput gains with our framework as compared to COPE. We observe that our framework outperforms COPE by 186% on average, and by as much as 250%. This is attributed to the fact that our framework efficiently exploits the good channel conditions by utilizing high transmission rates. Recall from our discussion in Section II that COPE targets the maximization of the network coding gain; therefore it typically employs low bit rates. However, with our framework, transmitters of native packets jointly consider the transmission rate and the potential network coding gain (by measuring the PDR of neighbor sniffer links at every rate); hence, while the coding gain is still kept at high levels, the transmission rate is aggressively increased to the extent possible. In Fig. 8, we plot the CDF of the ratio of the number of encoded packets at the relay to the total number of packets that were transmitted through the duration of each experiment. We observe that this ratio is almost the same as with COPE. On the other hand, as one observes in Fig. 9, high bit rates are used with our framework as inherently possible by the supported high PDR values in the considered topology.

Scenario 2: Experimenting with the cross topology with high quality links: Next, we consider the same set of nodes as in the previous scenario; however, our measurements now involve bi-directional traffic flows between sources and destinations. Hence, in contrast with the 'X' topology, the cross-topology routers can encode up to 4 native packets. This can potentially provide a higher coding gain. As observed in Fig. 7, our framework boosts the aggregate throughput of all end users by 209% on average, and by as much as 272%, relative to COPE. Note here that in this set of experiments we again observe that both COPE and our framework project a very similar behavior in terms of the CDF of the ratio $encoded/total$ packets, as in scenario 1 above. However, by comparing the performance of our framework in the two scenarios, we observe that there is approximately a 20% improvement in throughput,

[7]Our framework can be easily modified to be compatible with the latest Click version (v1.7.0rcl) and the latest Madwifi driver (v0.94).

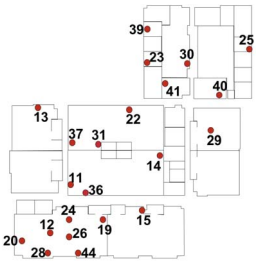

Fig. 6. The deployment of our wireless testbed; nodes are represented by dots along with their IDs.

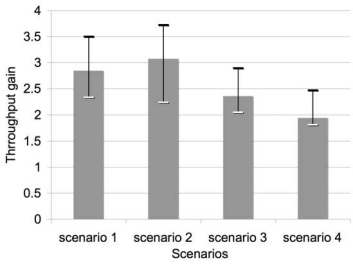

Fig. 7. Our framework outperforms COPE in all considered scenarios in terms of achieved throughput gain.

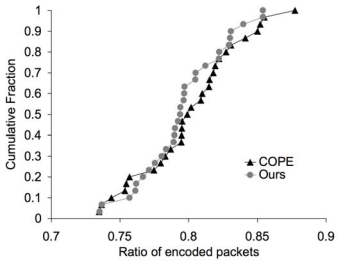

Fig. 8. Our framework achieves a very similar network coding gain as COPE, albeit the use of high bit rates.

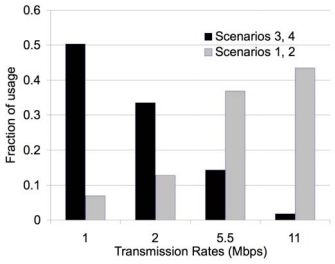

Fig. 9. With our framework, routers exploit the good channel conditions thereby using high bit rates whenever possible.

in scenario 2. This is due to the use of high transmission rates used by the router with our framework (shown in Fig. 9). A higher transmission rate (in the presence of fully saturated traffic) in these topologies results in a larger volume of packets traversing the router in a unit of time. As a consequence, the number of encoding opportunities at the router is higher for the duration of each experiment, and this provides an overall higher long-term throughput.

Scenario 3: Experiments with an 'X' topology comprised of low quality links: Further, we wish to assess the efficacy of our framework in topologies where not all links are of high quality. We consider 'X' topologies (Fig. 1) wherein one of the end receivers, e.g. Chloe, has a poor quality link with Jack and with Dave, although her link with Bob is of high quality. For this, we experiment with various 'X' topologies where the PDR on the link <Jack→Chloe> is poorer than that of the other links. A sample topology consists of nodes 12 (Alice), 28 (Dave), 26 (Jack), 19 (Bob) and 15 (Chloe) in Fig. 6. Here the PDR on the link <Jack→Chloe> is 0.45, while the PDR on the link <Bob→Chloe> is 0.75. We apply fully saturated traffic towards nodes 15 and 28. In Fig. 7, we observe that our framework provides significant throughput benefits even when the router maintains poor quality links with a receiver. Specifically, we observe that our framework outperforms COPE by at least 100% and by as much as 189%. However, note that in this case our framework does not perform as well as in scenarios 1 and 2. This is attributed to the fact that, in contrast to the previous scenarios, the router is now (for most of the time) coerced into using low transmission rates, to increase the probability of reception of the encoded packet by Chloe. Indeed, as observed in Fig. 9, Jack uses rates of 1 and 2 Mbps for most of the time. As expected, this affects the long-term throughput to some extent (although our framework still offers significant gains compared to COPE). We wish to point out here that in these experiments, we observed that Jack selected Chloe to be the ACKer in many cases. This somewhat contradicts our discussion in Section I. Note however that the difference in PDR between the links <Jack→Chloe> and <Jack→Dave> is not too high. Thus, with our framework in this case, Jack does not *sacrifice* the performance on the link to Chloe for the sake of increased total throughput. However, as we observe in our simulations (Section IV-A), routers follow the trend of selecting an ACKer to which they have high quality links, as long as this quality is much higher than the quality of the link(s) with the other candidate receiver(s). Moreover, throughout these experiments we observed that the ACKer selection module alone does not contribute to the throughput gain as much as the rate selection module. This is due to the small number of intended recipients; with such a limited set of receivers, it is highly likely that the random ACKer selection with COPE finds the appropriate ACKer. As we observe in our simulations, though, in dense network settings with many intended recipients, our ACKer selection module significantly contributes towards improving network performance (See Fig. 5).

Scenario 4: Experiments with the cross topology and low quality links: Finally, we examine cross topology scenarios with poor links. In particular, we consider the same physical node locations as in scenario 3. However now we have bidirectional flows, as in scenario 2. Note that the link <Chloe↔Dave> is also poor. As with scenario 3, we observe that the router typically prefers the use of low rates and with this, our framework does not perform as well as in scenario 2. In fact, from Fig. 7 we observe that the throughput gain with our framework is even lower than in scenario 3. This is in contrast with the expectation that due to the possibility of encoding 4 packets together, the gain would be improved. This is directly because of the poor PDR on the link <Chloe→Jack>, which in the sample topology of scenario 3 is 0.4. This forces Chloe into using a low bit rate on the link to Jack. As a consequence, the coding gain is not as high as in scenario 2. In addition, since the link <Dave→Chloe> is also poor, Dave prefers to use lower rates so as to increase the probability of overhearing at Chloe. Hence, Jack receives fewer packets from Chloe and Dave and thus, he does not encode 4 packets together as frequently. In conjunction with Jack using low bit rates, this causes the throughput gains with our framework to be lower than that in the other scenarios. However, note that the gains are still significant relative to COPE.

V. Discussion

The applicability of our framework with other network coding architectures: Our framework was designed and built on top of COPE. The main reason for this decision was that COPE imbibes utilities that facilitate the practical implementation of NC; examples include the exchange of reception reports and the estimation of the link quality though periodic probing. Our framework relies on two functions only: (a) In a generic local NC setting, routers should either explicitly know, or be able to predict whether certain neighbors have sniffed key packets; and (b) Transmitters of native packets need to estimate the quality of each link with their neighbors [2]. Given that such functionalities are typical requirements for any practical NC implementation, we believe that our framework can be applied in conjunction with other NC architectures (such as [16]) with minor modifications.

On the implementation overhead: As discussed in Section IV, we have implemented our framework using the original

COPE Linux implementation [5]. Hence, any practical implementation overheads imposed, as a result of various design decisions with COPE, are inherited by our implementation. More specifically, COPE requires certain packet processing operations in practice; these include packet storage and retrieval, packet padding and encoding/decoding. Such functionalities necessitate the employment of frequent memory copying operations, which are overhead intensive in terms of processing [33]. Indeed, through the course of our experiments, we observe that when the encoding engine (of either COPE or of our framework) was enabled, the CPU utilization with our Soekris boxes was at 97% with 802.11b, at a bit rate of 11 Mbps. We repeated a large part of our experiments with faster hardware (desktop PCs, which carry 1-GHz CPUs and 1-GB RAM, and run a local Fedora Linux installation); we observed that the CPU utilization dropped only by 8% on average. We also performed a small set of experiments with 802.11a; we observed that the desktop PCs could not process sufficiently large numbers of packets transmitted at bit rates higher than 12 Mbps to yield throughputs that commensurate with the applied rate. This is why at this point our experimental evaluation involves proof-of-concept experiments with 802.11b; this inherently supports low bit rates as compared to 802.11a or 802.11g. We expect that future, "lighter" NC implementations, such as [34], will allow the use of high bit rates and will showcase the performance benefits with our framework even more.

On the online computation of PDR: Towards calculating the PDR on every link, our implementation relies on the periodic transmission of probe packets. COPE uses this method as well; we selected this probe-based method for the sake of a fair comparison against COPE. Clearly, however, our framework is compatible with any PDR computation mechanism.

On the usability of our framework: In this paper, we focus on *local* network coding, where an encoded packet travels at most 1 hop away from its sender. Such scenarios are prominent in WLAN deployments [15], where a client sends and receives data through its associated access point (AP) only. In such cases, where the AP essentially plays the role of the router between the client and the rest of the network, access points may employ local NC thereby encoding packets exchanged among their clients. The application of our framework is expected to boost the performance of such networks, especially with delay-sensitive applications, such as real-time video streaming and online gaming [9], [13]. In our future work we plan to consider scenarios where encoded packets traverse multiple hops (such as in the butterfly topology [35]).

VI. Conclusions

Carefully-designed network coding mechanisms can provide significant multi-faceted advantages at various stages of network design. In this paper, we design a transmission rate and ACKer selection framework for extracting the potential benefits from local network coding. Our framework is composed of two modules. With the rate selection module, transmitters adapt the bit rate rate taking into consideration that their packets will be overheard by neighbors. With the ACKer selection module, relays intelligently decide on which from among the intended recipients should acknowledge the reception of an encoded packet. Such a decision is largely based on the the probability of successful decoding for each intended recipient. We evaluate our framework via extensive simulations as well as via a proof-of-concept implementation. Our evaluation demonstrates that our framework provides significant performance benefits, by increasing the total throughput by as much as 390% compared to COPE, in diverse traffic and topological settings.

References

[1] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in The Air: Practical Wireless Network Coding. In *ACM SIGCOMM*, 2006.

[2] J. Bicket et al. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *ACM MOBICOM*, 2005.

[3] S. Pal, S. R. Kundu, K. Basu, and S. K. Das. IEEE 802.11 Rate Control Algorithms: Experimentation and Performance Evaluation in Infrastructure Mode. In *PAM*, 2006.

[4] Onoe Rate Control. http://madwifi.org/browser/trunk/ath_rate/onoe.

[5] COPE WiKi. http://piper.csail.mit.edu/dokuwiki/doku.php?id=cope.

[6] D. S. Lun et al. Minimum-Cost Multicast over Coded Packet Networks. In *IEEE Trans. Inform. Theory, 52(6):931-938*, 2006.

[7] P. Chaporkar and A. Proutiere. Adaptive Network Coding and Scheduling for Maximizing Throughput in Wireless Networks. In *MOBICOM*, 2007.

[8] B. Scheuermann, W. Hu, and J. Crowcroft. Near-Optimal Co-ordinated Coding in Wireless Multihop Networks. In *ACM CONEXT*, 2007.

[9] H. Seferoglu and A. Markopoulou. Opportunistic Network Coding for Video Streaming over Wireless. In *Packet Video*, 2007.

[10] J. Le et al. How Many Packets Can we Encode? - An Analysis of Practical Wireless Network Coding. In *IEEE INFOCOM*, 2008.

[11] C. H. Liu and F. Xue. Network Coding for Two-Way Relaying: Rate Region, Sum Rate and Opportunistic Scheduling. In *IEEE ICC*, 2008.

[12] L. F. M. Vieira, A. Misra, and M. Gerla. Performance of Network Coding in Multi-Rate Wireless Environments for Multicast Applications. In *IEEE GLOBECOM*, 2007.

[13] H. Seferoglu and A. Markopoulou. Distributed Rate Control for Video Streaming over Wireless Networks with Intersession Network Coding. In *Packet Video*, 2009.

[14] H. Seferoglu, A. Markopoulou, and U. Kozat. Network coding-aware rate control and scheduling in wireless networks. In *ICME*, 2009.

[15] E. Rozner et al. ER: Efficient Retransmission Scheme for Wireless LANs. In *ACM CONEXT*, 2007.

[16] S. Rayanchu, S. Sen, J. Wu, S. Banerjee, and S. Sengupta. Loss-Aware Network Coding for Unicast Wireless Sessions: Design, Implementation, and Performance Evaluation. In *ACM SIGMETRICS*, 2008.

[17] S. Chachulski et al. Trading Structure for Randomness in Wireless Opportunistic Routing. In *ACM SIGCOMM*, 2007.

[18] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-Level Network Coding for Wireless Mesh Networks. In *ACM SIGCOMM*, 2008.

[19] R. Ahlswede, N. Cai, S-Y. R.Li, and R. W. Yeung. Network information flow. In *IEEE Trans. Inform. Theory, pp. 1204-1216*, July 2000.

[20] S. R. Li, R. W. Yeung, and N. Cai. Linear network coding. In *IEEE Trans. Inform. Theory*, 2003.

[21] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *IEEE INFOCOM*, 2005.

[22] R. Koetter and M. Medard. An Algebraic Approach to Network Coding. In *IEEE/ACM Trans. on Networking*, 2003.

[23] T. Ho, M. Medard, M. Effros, and D. Karger. On Randomized Network Coding. In *Allerton Conf. on Commun., Control and Computing*, 2003.

[24] T. Ho and R. Koetter. Online Incremental Network Coding for Multiple Unicasts. In *DIMACS Working Group on Network Coding*, 2005.

[25] Y. Wu, P. A. Chou, and S. Y. Kung. Information Exchange in Wireless Networks with Network Coding and Physical-layer Broadcast. In *Microsoft-TR-2004-78*, 2004.

[26] Z. Li and B. Li. Network coding: The Case for Multiple Unicast Sessions. In *Allerton Conference on Communications*, 2004.

[27] C. Papadimtriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[28] The Network Simulator 2. http://www.isi.edu/nsnam/ns/.

[29] D. B. Johnson et al. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, 2001.

[30] UCR Wireless Testbed. http://networks.cs.ucr.edu/testbed.

[31] Click Modular Router. http://read.cs.ucla.edu/click/.

[32] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *ACM MOBICOM*, 2004.

[33] J. Kim et al. A Memory Copy Reduction Scheme for Networked Multimedia Service in Linux Kernel. In *EurAsia-ICT, LNCS 2510, pp. 188195*, 2002.

[34] COPE on the Nokia N800 Phones. http://blogs.forum.nokia.com/blog/frank-fitzeks-forum-nokia-blog/2008/10/06/network-coding.

[35] C. Fragouli, J. Widmer, and J. Y. Le Boudec. Network Coding: an Instant Primer. In *ACM SIGCOMM CCR, vol. 36, pp. 63-68*, Jan. 2006.