Computer Sciences Department

Loss-Aware Network Coding for Unicast Wireless Sessions: Design, Implementation, and Performance Evaluation

Shravan Rayanchu
Sayandeep Sen
Jianming Wu
Suman Banerjee
Sudipta Sengupta

Technical Report #1633

March 2008

UNIVERSITY OF WISCONSIN MADISON

Loss-Aware Network Coding for Unicast Wireless Sessions: Design, Implementation, and Performance Evaluation

Shravan Rayanchu[1] Sayandeep Sen[1] Jianming Wu[1] Suman Banerjee[1] Sudipta Sengupta[2]

[1]University of Wisconsin, Madison, USA.
{shravan,sdsen,jianming,suman}@cs.wisc.edu

[2]Microsoft Research, Redmond, USA.
sudipta@microsoft.com

ABSTRACT

Local network coding is growing in prominence as a technique to facilitate greater capacity utilization in multi-hop wireless networks. A specific objective of such local network coding techniques has been to explicitly minimize the total number of transmissions needed to carry packets across each wireless hop. While such a strategy is certainly useful, we argue that in lossy wireless environments, a better use of local network coding is to provide higher levels of redundancy even at the cost of increasing the number of transmissions required to communicate the same information. In this paper we show that the design space for effective redundancy in local network coding is quite large, which makes optimal formulations of the problem hard to realize in practice. We present a detailed exploration of this design space and propose a suite of algorithms, called CLONE, that can lead to further throughput gains in multi-hop wireless scenarios. Through careful analysis, simulations, and detailed implementation on a real testbed, we show that some of our simplest CLONE algorithms can be efficiently implemented in today's wireless hardware to provide a *factor of two* improvement in throughput for example scenarios, while other, more effective, CLONE algorithms require additional advances in hardware processing speeds to be deployable in practice.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless Communication*

General Terms

Algorithms, Design, Performance

Keywords

IEEE 802.11, Network Coding, Wireless Networks

SIGMETRICS'08, June 2–6, 2008, Annapolis, Maryland, USA.

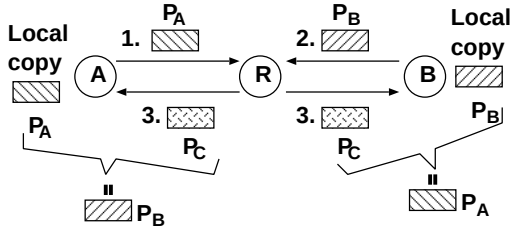


Figure 1: Local network coding example.

1. INTRODUCTION

Network coding [6] is a technique that combines multiple data units at intermediate nodes in a way that allows eventual recovery of these data units at their respective destinations while leading to fewer transmissions on the medium. In wireless environments, network coding can be employed at different layers, e.g., analog network coding at the symbol level in the PHY layer [27, 16], at the byte level in the MAC layer [17], and at the packet level just above the MAC layer [26, 18]. Among these, packet level network coding is readily deployable in existing commodity wireless platforms and is the focus of our work.

Packet level network coding have been proposed to operate at two different scales — global and local. In global network coding, multiple data packets continuously get coded together by different nodes in the network, potentially in a network-wide scale. Examples include MORE [9] and growth codes [15]. In contrast, in the case of local network coding, coding and decoding operations of data packets occur within the local neighborhood of individual relay nodes, e.g., COPE [18] and work by Wu et. al. [26]. We believe that global network coding approaches are likely to be better suited in many sensor networking scenarios that naturally allow for greater path diversity. Similarly, local network coding has greater applicability to wireless mesh networks, that are increasingly being designed by vendors to have single, tightly-controlled paths. The focus of this paper is on understanding the performance of local network coding schemes and are specifically targeted to the common case of unicast flows under lossy wireless environments. Given the complexity in design of efficient, loss-aware, local network coding schemes, we leave a similar treatment of global network coding beyond the scope of this paper.

1.1 Local and opportunistic network coding

COPE is a good example of local network coding technique in which an intermediate node combines locally available packets without requiring coordination with any other node in the wireless network using XOR operations allowing them to be immediately decoded by next-hop neighbors. We explain this with a simple three node example shown in Figure 1. Node A wants to send a single packet (P_A) to node B, while node B wants to send single packet (P_B) to node A. Due to transmission range limitations both paths go via relay node, R. Using standard techniques of packet forwarding, four wireless transmissions would be needed to complete these end-to-end packet transfers — one transmission by A and B to send packet P_A and P_B respectively to R, and two transmissions by R to individually forward these packets to their respectively destinations.

COPE-style network coding allows us to complete this entire packet transfer using three transmissions instead of four as follows. End nodes A and B use one transmission each to forward packets P_A and P_B to node R as before. Unlike common practice, nodes A and B also retain a copy of packets P_A and P_B respectively. Relay node, R, creates a new packet P_C by XORing packets P_A and P_B together ($P_C = P_A \oplus P_B$), and broadcasts it to both the end nodes. On receiving P_C, node A can recover its intended packet, P_B, by XORing the retained copy of P_A with P_C, since $P_A \oplus P_C = P_A \oplus (P_A \oplus P_B) = P_B$. Node B can, similarly, recover packet P_B by XORing P_C with P_A. Using prior terminology [18], we will refer to the original packets (P_A and P_B) as *native* packets while the packet P_C derived as a combination of native packets as a *coded* packet. Since the number of transmissions required to carry the same amount of information is three and four respectively for COPE-style local network coding and no coding, COPE is able to improve the information carrying capacity of the channel by 33%, in this simple example.

Note that the coding operation is only possible if both packets P_A and P_B are available with the relay prior to the time the relay has an opportunity to send out a packet. The coding scheme in COPE is opportunistic in nature because it only performs the coding operation when such coding opportunities arise. If no coding opportunity exists when the relay gets to transmit a packet, the next native packet from its output queue gets transmitted without any further delay.

1.2 Lossy links and network coding

The attractiveness of such local network coding schemes stem from their simplicity. The XOR operation is easy to implement and local decisions at intermediate nodes avoid overheads of global coordination and aggregation of information. While COPE-style network coding leads to be performance gains in many scenarios when compared to no network coding, its performance actually happens to be fairly sub-optimal in lossy wireless environments. By minimizing the number of transmissions necessary to carry information across the wireless channel, such techniques eliminate all redundancy from the traffic. We call such an approach, the minimalist network coding approach. In contrast, we argue that through careful use of redundancy using network coding (which might increase the number of transmissions needed to carry the same volume of information) further performance gains are achievable in face of wireless link losses. We believe that such design is particularly important in urban mesh networks, where loss rates of 30% or higher are observed on a significant fraction of wireless links [5].

The goal of this paper is to study the interaction between lossy wireless environments and local network coding strategies. Even in the setting of local network coding, the design space of loss-aware coding is quite large, and finding optimal solutions proves to be significantly difficult. After systematically exploring this problem space, we converge upon a suite of algorithms for network Coding with LOss awareNEss, or CLONE, that introduces adequate redundancy in local network coding operations, thereby achieving further performance gains. Coding decisions in CLONE algorithms continue to be local and opportunistic in nature, and can be deployed as software-only updates in 802.11-based wireless systems.

Overall, we believe that this is the first piece of work that carefully evaluates the role of redundancy in local network coding, and does so through analysis, simulations, and implementation on a real testbed.

1.3 Key Contributions

Summarizing, the main contributions of this work is threefold: (i) It recognizes the important role of redundancy in creating local network coding solutions for multi-hop wireless scenarios, and that minimization of transmissions is not necessarily the right objective for achieving throughput gains. (ii) It explores the rich design space in creating effective loss-aware local network coding solutions. (iii) It presents a suite of loss-aware network coding algorithms, each with different performance points and computational costs. The first among these can be readily deployed in existing 802.11 wireless hardware to provide throughput gains of a factor of two. This is demonstrated through detailed evaluation on our wireless testbed. The remaining two algorithms can potentially provide even higher throughput gains, but require increase in processing speeds of the wireless hardware to be feasible.

2. DESIGN CONSIDERATIONS AND PROBLEM STATEMENT

The minimalist approach to local and opportunistic network coding is quite efficient in loss-free environments. However, in presence of losses such an approach can limit performance gains. In this section, we discuss how even very simple redundancy mechanisms within network coding can lead to performance improvements over the minimalist approach. Our examples in this section are intentionally simple and will help in building the intuition behind the need for redundancy in local network coding. In addition, the examples and the subsequent problem statement will also expose the rich design space in creating local network coding solutions under lossy environments. Therefore, the performance gains of these schemes over the minimalist network coding approach will be somewhat modest. However, in subsequent sections we will define more sophisticated yet practical, loss-aware, network coding schemes and show their efficiency in providing higher throughput gains.

In both of our examples, we focus on the instants where coding opportunities, indeed, exist at a relay node. Our algorithms, simulations, and implementation, are, however, operate over regular traffic flows where coding opportunities arise in the natural course of operation.

Scheme	$p = 0$	$p = 0.2$	$p = 0.4$	$p = 0.6$
No coding	0.25	0.2	0.15	0.1
COPE	**0.33**	**0.27**	0.2	0.13
COPE-dup	0.25	0.24	**0.21**	**0.16**

Table 1: Performance comparison of delivery rate (successfully delivered packets per packet transmission time) for no coding, COPE, and COPE-dup, under different loss rates in the simple two flow topology.

2.1 Example I: Two flow topology

We first consider the simplest topology of network coding as is shown in Figure 1 with two flows ($A \rightarrow B$) and ($B \rightarrow A$), with all traffic carried through the relay.

For the sake of illustration, let us assume that the links incoming to the relay, i.e., $A \rightarrow R$ and $B \rightarrow R$, are loss-free, and the outgoing links from the relay, i.e., $R \rightarrow A$ and $R \rightarrow B$, have a loss rate of p. (In our algorithms, each of these links are lossy and are assumed to have different loss rates.) The relay node will learn these loss rates through packet loss observations over time. Let us consider the simplest scenario where there are no re-transmissions of lost frames.

The minimalist approach to network coding, in this scenario, is COPE, which will require one single coded packet $P_C = P_A$ XOR P_B to be transmitted to both destinations, as was shown in Figure 1. Let us consider a new network coding scheme, in which the packet, P_C, constructed as above, is transmitted *twice* back-to-back. We will call this new scheme, COPE with duplication, or COPE-dup in short.

Each packet is decodable by a destination node, if either of the two coded packets is successfully received by it. Thus, COPE-dup utilizes four packet transmissions to carry the packets P_A and P_B to their respective destinations (two for the incoming native packets and two for the back-to-back duplicate coded packets).

The probability of a successful end-to-end packet delivery in COPE-dup is $(1 - p^2)$ and requires four transmission slots. Therefore, the achievable delivery rate (in units of successfully delivered packets per packet transmission slot) in COPE-dup is $\left((1 - p^2\right)/4$. In comparison, the probability of a successful end-to-end packet delivery in COPE is $(1 - p)$ and requires three transmission slots (two for incoming native packets, and one for the outgoing coded packet). Therefore, the achievable delivery rate is $(1 - p)/3$.

Table 1 compares the delivery rate of no coding, COPE, and this new scheme, COPE-dup, for different values of p.

It is easy to observe that under higher loss rates, this trivial extension to COPE, although not likely to be very efficient, already leads to a throughput gain of 25% over COPE. Even though COPE requires fewer transmissions to convey packets successfully across the relay node, COPE-dup with its greater redundancy and higher transmission attempt requirements, leads to better overall utilization of the channel that ultimately leads to higher throughput. While COPE-dup and no coding schemes both use four transmissions to convey two packets, the former utilizes network coding to achieve redundancy that provides the necessary performance gains under higher losses.

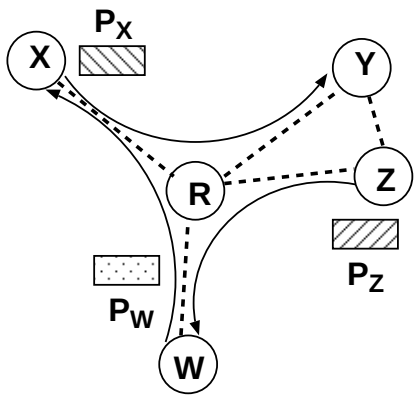


Figure 2: A three flow topology. Dotted lines indicate which nodes are in communication range of each other.

2.2 Example II: Three flow topology

We next consider a three flow topology, shown in Figure 2. In this topology, node W has a packet P_W to forward to node X, node X has a packet P_X to forward to node Y, and node Z has a packet P_Z to forward to node W. All traffic flows through the intermediate relay, R. Let us assume that node Y and Z are in communication range of each other, so that node Y can "overhear" the packet P_Z when its transmitted by the latter to R. We again assume that the only lossy links in this topology are all of the outgoing links from R, i.e.,$R \rightarrow X, R \rightarrow Y, R \rightarrow Z$, and the loss probability of each of these links is p. If no coding is used, that the delivery rate of each flow in the example is $(1 - p)/6$.

We first observe that in this scenario, COPE cannot be used. This is because if we try to combine any set of native packets at the relay node, no more than one destination node will be able to recover its intended packet. For example, if a coded packet, $P_W \oplus P_X$, is transmitted by the relay, then node X will be able to recover packet P_W by combining this coded packet with its local copy of P_X. However, node Y cannot recover packet P_X as it does not locally have a copy of packet, P_W.

Hence, we first define a new minimalist network coding solution for this scenario, and then extend it to induce redundancy.

Scheme I - MIN-code: The minimalist network coding scheme (or MIN-code, for short) requires the relay node to create the following two coded packets, $P_Q(= P_W \oplus P_X)$, and $P_R(= P_X \oplus P_Z)$, and send them out back-to-back. On receiving these packets, each node can successfully decode their intended packet, as follows: (i) node X recovers P_W as $P_W = P_X \oplus P_Q$ where P_X is available locally and P_Q is one of the received coded packets, (ii) node Y recovers P_X as $P_X = P_Z \oplus P_R$, where P_Z was previously overheard by Y when Z had transmitted this packet on the medium, and P_R is a received coded packet, and (iii) node W recovers P_Z as $P_Z = P_W \oplus P_Q \oplus P_R$, where P_W is available locally, and P_Q and P_R are received coded packets.

Thus all three native packets can reach the destination using a total of five transmissions (three due to incoming transmission of native packets to the relay, and two outgoing transmission of coded packets from the relay), instead of six

Scheme	p = 0	p = 0.2	p = 0.4	p = 0.6
No coding	0.167	0.133	0.1	0.067
MIN-code	**0.2**	0.149	0.104	0.064
LOOP-code	0.167	**0.155**	**0.124**	**0.083**

Table 2: Performance comparison of delivery rate (successfully delivered packets per packet transmission time) for no coding, MIN-code, and LOOP-code, under different loss rates for the three flow topology.

in the no coding case.

In MIN-code, the delivery rate of the $W \rightarrow X$ and $X \rightarrow Y$ flows is $(1-p)/5$, where as the delivery rate of the $Z \rightarrow Y$ flow is $(1-p)^2/5$.

Scheme II - LOOP-code: This is an alternate network coding scheme with redundancy (originally introduced in [12]), in which the relay transmits three coded packets: $P_Q(= P_W \oplus P_X)$, $P_R(= P_X \oplus P_Z)$, and $P_S = (P_Z \oplus P_W)$. In this scheme, the packet P_S is being added to the list of transmitted packets to provide the additional redundancy. The consequence of this additional packet is that each node can now recover its intended packets in two different ways. For example, node X can recover P_W using the received coded packets and the local copy of P_X, either as (i) $P_X \oplus P_Q(= P_W)$, or as (ii) $P_X \oplus P_R \oplus P_S(= P_W)$. Similarly, node Y can recover P_X using the received coded packets and the local copy of overheard packet P_Z, either as (i) $P_Z \oplus P_R$, or as (ii) $P_Z \oplus P_S \oplus P_Q$. Finally, node W can recover P_Z, either as (i) $P_W \oplus P_S$, or as (ii) $P_W \oplus P_Q \oplus P_R$.

A packet of a flow, say $W \rightarrow X$, is successfully delivered, if either P_Q is correctly received by X, or both P_R and P_S are correctly received by X. The probability of this event is $1 - p \cdot (1-(1-p)^2) = 1 - 2p^2 + p^3$. The delivery rate of each flow in LOOP-code is, therefore, $(1 - 2p^2 + p^3)/6$.

We present a comparison of the average delivery rate across the three flows for the three schemes, no coding, MIN-code, and LOOP-code, in Table 2. We can see that minimalist coding, that reduces the number of transmissions needed to carry the traffic, is quite efficient in loss-free scenarios, but redundant transmissions plays an important role in improving the delivery rates, as loss rates increase.

2.3 Loss-aware network coding: Problem statement

The general problem of local network coding can be stated as follows: A set of packets, $P_1, P_2, \ldots, P_n$, are available at a relay node, R, to be forwarded to next-hop recipient nodes $N_1, N_2, \ldots, N_n$, respectively.

Each next-hop recipient, N_i, also has a copy of a subset of these packets locally available (this subset does not include packet P_i). There are two ways in which a node N_i will have a packet $P_j (\neq P_i)$ locally available. First, if N_i was the previous-hop node of the packet P_j and was, therefore, the node that had sent this packet to R. Second, if some other node, N_l is the source of packet P_j, and N_i "overheard" the transmission of the packet from N_l to R.

R has probabilistic knowledge of the availability of these packets at the different destination nodes. For example, if node N_i was the previous hop of a packet sent to R, then R knows that this packet is certainly available at N_i. On the other hand, if the packet was sent by another node N_l to R, and if the loss rate on the wireless link $N_l \rightarrow N_i$ is p, when R can "guess" that the packet is available at N_i with a probability p.

Under these assumptions, the key question of local network coding is the following:

In the upcoming transmission opportunities at the relay node, what coded and native packets should the relay node transmit, that maximizes some metric (e.g., throughput, packet delivery rate, etc.) across all of the intended next hop recipients?

COPE answers this question using the following minimalist rule. It creates a coded packet P_q by XORing a set of packets, $\mathcal{P}$, out of the entire set of packets available at R using the following rule. For every packet, $P_w \in \mathcal{P}$, its next-hop recipient N_w already has every packet XORed together in P_q except the packet P_w, i.e., $\forall P_w \in \mathcal{P}, N_w$ has all packets in $\mathcal{P} - \{P_w\}$. Only such a packet, P_q, is transmitted by the relay at each instant.

While, the MIN-code network coding scheme also uses a different minimalist rule to answer the above question, the LOOP-code scheme uses redundant network coding scheme. Clearly, these are not the only three network coding schemes that are possible. There is a whole gamut of different local network coding schemes that can be defined for different topologies and flow scenarios. Some network coding schemes may choose to limit coding operations to create a coded packet out of two native packets at a time, while others might use three or more native packets to create a coded packet. Redundancy is generated through creation of additional packets that lead to multiple ways of decoding one or more native packets at the intended next-hop recipients.

In general, given a set of n native packets, the number of possible coded packets is $2^n - 1$ (we can treat a native packet to be a special case of a coded packet). Therefore, the real challenge in loss-aware network coding design is to choose an appropriate subset of all possible coded packets at each instant in time that should be transmitted by the relay to all its intended recipients. Since exploring this exponential search space is likely to have high computation costs, in the next section we will discuss more tractable versions of this problem that will lead to a practical design of loss-aware network coding.

3. CLONE ALGORITHMS

In this section we present our algorithms for selecting coding strategies that answer the question raised in the previous section. Throughput is our metric of optimization and our network coding strategies are designed to aware of loss conditions on different links. All of our algorithms are *local,* i.e., the algorithms only utilize information that can be determined locally by the relay node within its neighborhood. In addition, the algorithms are *opportunistic,* i.e., the relay node only uses the set of native packets that are already available with itself due to transmissions from its neighbors to create coded packets.

Algorithmic approach and overview: A general network coding strategy may choose to create coded packets, each of which may eventually be constructed of two or more native packets. As discussed before, if the number of native packets available at the relay is n, then the total possible number of coded packets is $2^n - 1$. Let us denote a coding strategy chosen by the relay node (essentially a subset of all the possible coded packets) to be S. For example, in COPE, S consisted of a single packet, shown in Figure 1 as P_C. In

COPE-dup, S consisted of two packets (two occurrences of P_C).

Under the assumption of lossy wireless links, each native and coded packet transmitted by the relay will be received by next-hop node, N_i with some probability. Depending on the packet receptions, N_i will be able to decode its intended packet, P_i, with a probability that is a function of link loss probabilities and the coding strategy, S. We will denote this decoding probability of packet P_i by $q_i(S)$. Assuming all coded packets are of the same size, and are transmitted at the same physical data rate [1], then the total transmission time of all the coded packets is proportional to $|S|$. The expected number of packets decoded as a result of this strategy is $\sum_{i=1\ldots n} q_i(S)$.

Let the metric of interest be aggregate throughput over all next-hop recipients. This metric will be given by the average number of successfully decoded packets per unit time, i.e., $\sum q_i(S)/|S|$. Thus, we can define our optimal coding strategy, S^*, as the one that maximizes $\sum q_i(S)/|S|$.

This maximization formulation turns out to be a fairly complex non-linear program. Therefore, we first consider a simpler coding rule where we limit the maximum number of native packets that are combined in a single coded packet to two. We refer this form of network coding as as *binary network coding*. This form of network coding is somewhat simpler because the total number of possible coded packets is $n(n-1)/2 + n$.

It turns out that even a simplified version of the loss-aware binary network coding problem is $\mathcal{NP}$-hard. In this section, we start by presenting this binary network coding formulation and explain why it is $\mathcal{NP}$-hard. Then we present an initial heuristic for the binary network coding problem, called CLONE-kDSP. Next, we present an even simpler, and more practical algorithm, called CLONE-k-Loop, which is a special case of the CLONE-kDSP algorithm. Both CLONE-kDSP and CLONE-k-Loop provide effective solutions for the binary network coding problem.

For the general version of the network coding problem when more than two native packets can be combined into a single coded packet, we develop another heuristic, called CLONE-MultiXOR, which is greedy in nature. Since this heuristic solves the more general formulation, its performance is better than the heuristics developed for the binary network coding versions.

An important requirement of all these different network coding algorithms is that they need to be computationally very efficient. Our three heuristics, CLONE-k-Loop, CLONE-kDSP, and CLONE-MultiXOR, have increasing computational complexity, so much so that with current processing speeds on wireless nodes, only CLONE-k-Loop is deployable in practice. In simulations we show the superior performance of CLONE-kDSP and CLONE-MultiXOR algorithms (compared to CLONE-k-Loop) by ignoring processing delays. In our implementation, it was only feasible to implement the CLONE-k-Loop algorithm using processing limits of current hardware. Nevertheless, we present all three algorithms in this section, because it is likely that processing speeds will continue to get faster, and it might become feasible to deploy the other two algorithms in the future.

[1]We make these simplifications only for the formulation, but not in our implementation.

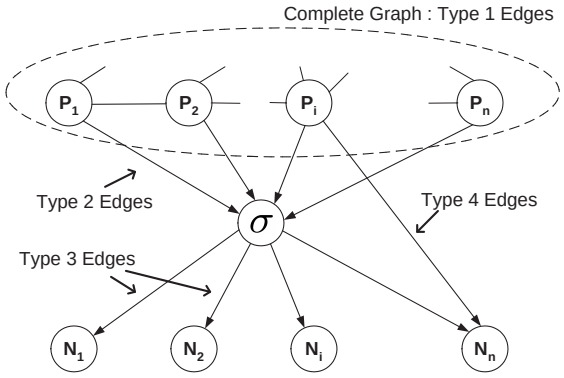


Figure 3: Graph based formulation for binary coding

3.1 Binary network coding

We model the binary network coding problem through a graph-based formulation, as follows. Consider a relay node that needs to deliver n packets $P = \{P_1, P_2, \ldots, P_n\}$ to next-hop nodes $D = \{N_1, N_2, \ldots, N_n\}$, where packet P_i has next-hop N_i.

We model the encoding and decoding operations as a graph $H = (V, E)$ The vertex set of this graph is defined as $V(H) = P \cup D \cup \{\sigma\}$. σ represents a special vertex in this formulation which will be used to model packet availability at different next-hop nodes, as described below. The edges $E(H)$ of the graph are classified into the following types:

1. **Type 1 -** (P_i, P_j): There is a bidirectional edge between all pairs of native packets (P_i, P_j) representing the possibility of transmitting the corresponding coded packet $P_i \oplus P_j$.

2. **Type 2 -** (P_i, σ) There is a directed edge from each P_i to σ denoting the possibility of transmitting the native packet P_i.

3. **Type 3 -** (σ, N_j): There is a directed edge between σ and every node N_j.

4. **Type 4 -** (P_i, N_j): These directed edges denote that next-hop node N_j already has packet P_i (with a certainly probability), possibly through overhearing of an earlier transmission or because packet P_i passed through node N_j in a previous hop.

Coding strategies: A valid coding strategy can be represented as a subset of edges on the induced subgraph of H on vertices $P \cup \{\sigma\}$. Selection of edges (P_i, P_j) imply the transmission of coded packet $P_i \oplus P_j$ while selection of edges (P_i, σ) imply the transmission of native packet P_i.

Now consider any given next-hop N_i. It may not receive all transmitted packets due to packet loss. Consider the subgraph $H(i)$ corresponding to next-hop N_i that consists of (i) all edges of type 3 and 4 above, and (ii) subset of edges from the induced subgraph on vertices $P \cup \{\sigma\}$ corresponding to successfully received native and coded packets. Then, the following lemma states the conditions under which N_i can decode packet P_i:

LEMMA 1. *When coding two packets at a time, next-hop node N_i can decode packet P_i if and only if there is a (directed) path from node P_i to node N_i in graph $H(i)$.*

We will use this basic property of decodability at each next-hop node to derive optimization models for selecting coding strategies at each node.

Decoding strategy: Each receiver node N_i can determine when a packet can decoded by maintaining a simplified version of the above graph, with node s and nodes N_j, $j \neq i$ removed. Edges (P_j, P_k) are included in the subgraph when this node has received coded packet $P_j \oplus P_k$, and edges (P_j, N_i) are included when this node has native packet P_j (either overheard or because that packet passed through this node earlier). Then, by Lemma 1, packet P_i can be decoded whenever a path becomes available from P_i to N_i in this graph.

In our implementation, a relay node which selects a coding strategy, explicitly sends out information (in packet headers) about the possible decoding paths (in the above graph) for each packet, thus making the decoding strategy trivial.

Simplified formulation for binary network coding: Our original formulation of the loss-aware network coding problem to maximize $\sum q_i(S)/|S|$ happens to be a non-linear program and is difficult to solve efficiently. Therefore, we consider a simpler formulation of the problem, as follows. We define a probability threshold parameter, p, which represents a target delivery rate for all packets, P_i to their corresponding next-hops, N_i. Let the successful delivery probability of the link between the relay and each next-hop, N_i, be r_i. Then our modified objective is to choose a coding strategy S that requires the minimum number of transmissions from the relay, *so as to meet this delivery probability threshold for each packet, P_i to next-hop, N_i.*

We can formulate this problem as follows. Let S denote the desired coding strategy, which is a subset of edges from the induced subgraph on vertices $P \cup \{s\}$ corresponding to transmitted native and coded packets. For each i, the probability that packet P_i can be successfully decoded at next-hop node N_i can be computed as follows. Assign a weight of r_i to each edge in S. For edges (P_i, N_j), the weight is the probability estimate that node N_i already has packet P_j (if sure, then this is 1). Each edge of type 3 has weight 1. All other edges have weight zero. The weight on an edge can be interpreted as the probability that the edge is up. Then, the probability $q_i(S)$ that packet P_i can be successfully decoded at next-hop node N_i under coding strategy S is the probability that this graph has a path from node P_i to node N_i. This problem can be formally stated as follows:

$$\min_S |S|$$

subject to

$$q_i(S) \geq d \ \ \forall\ i = 1, 2, \ldots, n$$

The computation of $q_i(S)$ on a general graph is a variation of the "network reliability problem" which is known to be $\#P$-complete [24]. Thus, the problem of even verifying whether a given coding strategy $|S|$ satisfies a given probability threshold is $\#P$-complete. To solve any version of this problem, we need to make the computation of $q_i(S)$ more tractable. We do this by considering only edge-disjoint paths from P_i to N_i in the graph H. Of course, such paths can have the edge (σ, N_i) in common, since that denotes the usage of any native transmitted packet. Hence, we will use the term "edge-disjoint" with this reservation.

The probability $w_i(\pi)$ that a path π from P_i to N_i is up is simply the product of the weights associated with each edge on the path. If we choose a set of k_i edge-disjoint paths $\pi_1^i, \pi_2^i, \ldots, \pi_{k_i}^i$ for delivering packet P_i to next-hop N_i, then we have

$$q_i(S) = 1 - \prod_{k=1}^{k_i} [1 - w_i(\pi_k^i)]$$

The selected paths must have all their edges in $S \cup \{(\sigma, N_i)\}$, so the optimization problem now becomes

$$\min_S |S|$$

subject to

$$1 - \prod_{k=1}^{k_i} [1 - w_i(\pi_k^i)] \ \geq \ p \ \ \forall\ i \quad (1)$$

$$\pi_k^i \ \subseteq \ S \cup \{(\sigma, N_i)\} \ \ \forall\ 1 \leq k \leq k_i, \forall i \quad (2)$$

$$\pi_k^i \cap \pi_{k'}^i \ \subseteq \ \{(\sigma, N_i)\} \ \ \forall\ k, k', i \quad (3)$$

Constraints (1) correspond to the probability threshold that packet P_i can be successfully decoded at node N_i. Constraints (2) denote that the paths must contain only edges corresponding to the (transmitted packets) coding strategy S. Constraints (3) enforce that the paths π_k^i, for a given i, are edge-disjoint (except possibly for edge (σ, N_i)).

While the constraints can be verified in polynomial time for a given set of edge-disjoint paths for each (P_i, N_i) pair, the evaluation of the optimal strategy, S^*, is still $\mathcal{NP}$-hard, as can be shown by reduction from the generalized Steiner network problem [25]. Therefore, instead of attempting an optimal solution to the problem defined above, we define multiple heuristics to minimize this objective function.

3.1.1 *The* CLONE-kDSP *algorithm*

Our first heuristic for binary network coding tries to solve $\min_S |S|$, under Constraints 1 to 3 using the graph based problem formulation presented in the previous section. The algorithm works as follows: Given a set of the (P_i, N_i) pairs, we first generate the graph H and then sort (P_i, N_i) in an increasing order of delivery probability r_i to next-hop node N_i. For each such (P_i, N_i) pair in this sorted order, we generate minimum cost edge-disjoint paths in the graph H from node P_i to node N_i in an incremental manner using the *successive shortest paths* algorithm [3]. Note that our pre-sorting based on delivery probabilities imply that we consider packets destined to lossy next-hop nodes, earlier. For the successive shortest path computation, the cost of an edge is taken to be zero if it has been used by an earlier (P_j, N_j) pair, and 1 otherwise. After each new edge-disjoint path is added, we compute the decoding probability $q_i(S)$. We terminate if the threshold probability p is met or exceeded, or if no more disjoint paths exist from P_i to N_i. The coding strategy, S, is the set of all type 1 edges that are selected as part of the different edge disjoint paths.

3.1.2 *The* CLONE-k-Loop *coding algorithm*

We now present CLONE-k-Loop coding algorithm, a special case of CLONE-kDSP coding algorithm, which essentially outputs a sequence of k Type 1 edges forming a loop. The LOOP-code algorithm presented earlier in Section 1 is an example of CLONE-k-Loop coding algorithm with $k = 3$.

Let us consider R a relay which is surrounded by a set of neighboring nodes N_1, N_2, .. N_k to which it has to send packets P_1, P_2, .. P_k respectively. Now, consider a scenario where the N_1 has packet P_2 already available with it (either because this packet was routed through it or because it overheard a transmission), similarly N_2 has packet P_3 and so on, with N_k having packet P_1 with it. In this scenario, CLONE-k-Loopoutputs a sequence of coded packets $S : C_1(= P_1 \oplus P_2)$, $C_2(= P_2 \oplus P_3)$, $C_4(= P_3 \oplus P_4)$ .. $C_k(= P_k \oplus P_1)$. This gives each node N_i two ways to decode the packet P_i, either as (1) $C_i \oplus P_{(i+1)\text{mod } k}$ or as (2) $P_{(i+1)\text{mod } k} \oplus s_m, (\forall s_m \in (S - \{C_i\}))$. If the loss probability of each of the links is p, it can be shown that the probability that a packet P_i is successfully delivered to N_i using CLONE-k-Loop is given by $(1 - p.(1 - (1 - p)^{k-1}))$.

3.2 General network coding

Selecting a coding strategy while allowing for coding more than two packets at a time is computationally expensive as it involves selecting an appropriate subset of all possible coded packets which is exponential in search space. As an example of the class of algorithms which can code more than two packets, we present a greedy heuristic, CLONE-MultiXOR, described next.

3.2.1 *The* **CLONE-MultiXOR** *heuristic*

Let $P = \{P_1, P_2..P_n\}$ be the packets that have to be forwarded to next-hop nodes $N = \{N_1, N_2, ..N_n\}$. Let $K = \{K_1, K_2..K_n\}$ be the corresponding set of packets already available at these next-hop nodes. Note that each K_i is a subset of P. Let S be the set of packets (native or coded) selected for transmission. Let $NPATH(P_i, N_i, S)$ be the number of ways in which N_i can decode packet P_i using the set of packets S. The CLONE-MultiXOR coding algorithm works as follows:

Let $COMB(K_i)$ be the set of coded packets derived by XORing of all possible subsets of elements of K_i. For each (P_i, K_i), we generate a row of packets, $t_i = P_i \cup \{P_i \oplus COMB(K_i)\}$. For example, if N_1 which is the next-hop node of P_1 has a set of packets $K_1 = \{P_2, P_3\}$ already available with it, then $t_1 = \{P_1, P_1 \oplus P_2, P_1 \oplus P_3, P_1 \oplus P_2 \oplus P_3\}$. A next-hop N_i will be able to directly decode the packet P_i from any of the packets in t_i using the packets present in K_i. We now have a table of packets $T = \{t_1, t_2, ..t_n\}$.

The goal of CLONE-MultiXOR now is to select S, a subset of these packets, such that (1) every next-hop N_i should be able to decode the packet P_i i.e. $NPATH(P_i, N_i, S) \geq 1$ and (2) the number of ways in which a native packet P_i can be decoded by N_i is maximized. CLONE-MultiXOR addresses (1) by selecting at least one packet from each of the rows t_i, thus guaranteeing that each next-hop N_i will able to decode the packet P_i. In order to address (2), CLONE-MultiXOR assigns a *usefulness count* to each of the packets in T. Given a set S of packets which have already been selected for transmission, the usefulness count of a packet d is defined as the *additional* number of ways in which each packet P_i can be decoded by N_i i.e. *usefulness_count*(d) $= \sum_{i=1}^{i=n} \{NPATH(P_i, N_i, S \cup \{d\}) - NPATH(P_i, N_i, S)\}$. Initially, $S = \phi$, thus *usefulness_count*(d) is just the number of rows of T containing d. On every iteration, CLONE-MultiXOR simply selects a packet d with the maximum usefulness count and adds it to S. The usefulness count of each packet in T is then updated and the process is repeated until $|S| = n$. The pseudo-code for CLONE-MultiXOR is presented in Algorithm 1 of Appendix A.

Table 3: Different ways of decoding a packet when using CLONE-MultiXOR for the illustrative topology in Figure 4.

Next-Hop	Decoding sequences
N_1	(C_1), (C_2, C_3), (C_3, C_4, C_5), (C_3, C_5, C_6)
N_2	(C_2), (C_1, C_3), (C_4, C_5), (C_5, C_6)
N_3	(C_6), (C_2, C_5), (C_1, C_3, C_5)
N_4	(C_1), (C_4), (C_2, C_5, C_6), (C_3, C_5, C_6)
N_5	(C_2), (C_1, C_4), (C_3), (C_1, C_5, C_6)
N_6	(C_5), (C_1, C_2, C_4), (C_1, C_2, C_3), (C_1, C_2, C_6), (C_1, C_3, C_6)

4. SIMULATIONS

In this section we evaluate the performance of the CLONE coding schemes presented in Section 3 using simulations on some example network topologies and show their effectiveness in reducing the effective loss rates of the wireless links. We compare the relative performance of various CLONE coding schemes under different link loss rates, overhearing patterns and network topologies. Later in Section 5, we present an implementation of CLONE, followed by the results from a real testbed.

Evaluation Metrics: We use the following evaluation metrics for our simulations and implementation:

- **Post coding loss rate:** This is the effective loss rate as perceived by a receiver after applying the decoding algorithm. That is, this is the perceived loss rate at the receiver *after* all the possible decoding options have been explored.

- **Throughput gain:** We measure the network throughput, which is the sum of end-to-end throughput of all the flows present in the network. We present the corresponding throughput gains, which is the ratio of the measured network throughput with and without CLONE coding schemes applied.

4.1 Results on an Illustrative Topology

In order to illustrate certain key properties of the CLONE coding algorithms, we start with a simple topology shown in Figure 4 where a relay R has six neighboring nodes N_1, N_2, .. N_6. The dotted lines indicate that the nodes are in the communication range of each other and the arrows represent the flow of packets P_i that have to be routed to destination N_i through the intermediate relay. That is, the following six packets have to be forwarded by the relay R, $P_1 : N_5 \rightarrow N_1$, $P_2 : N_6 \rightarrow N_2$, $P_3 : N_1 \rightarrow N_3$, $P_4 : N_2 \rightarrow N_4$,$P_5 : N_3 \rightarrow N_5$ and $P_6 : N_4 \rightarrow N_6$. Figure 4 also shows the set of packets available at each of the nodes N_i. For example, N_1 has packets $\{P_3, P_2, P_4\}$ available with it because N_1 was the source of the packet P_3 and it overheard the packets P_2 and P_4 during their corresponding transmissions by N_2 and N_6. In the absence of coding, the relay needs six transmissions to forward these packets to their corresponding next hops. We now illustrate the performance of each of the coding schemes for this scenario.

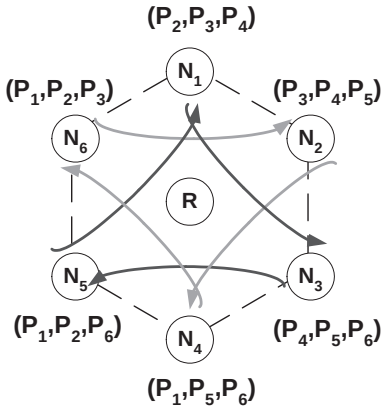


Figure 4: An Illustrative Topology.

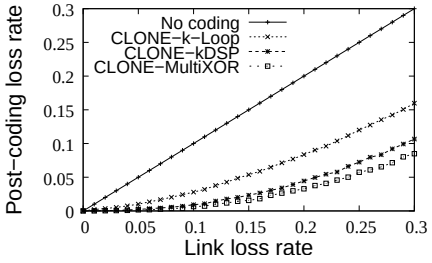


Figure 5: Post coding loss rates on an illustrative topology ($N = 6$).

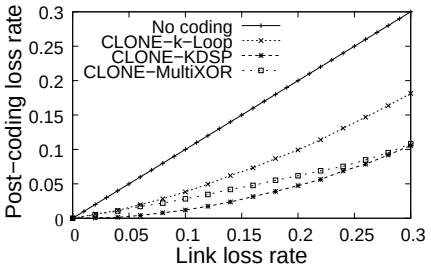


Figure 6: Post coding loss rates for a random topology ($N = 6$).

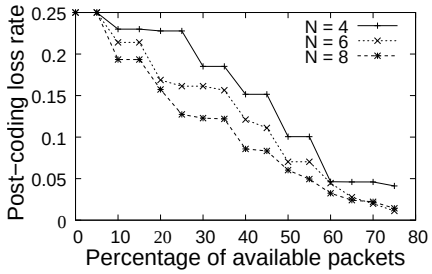


Figure 7: Effect of Relay Node Degree

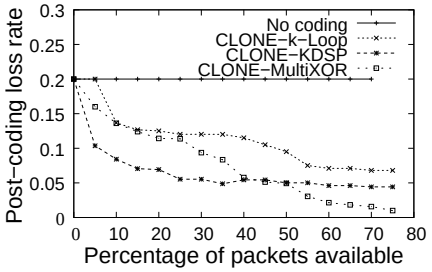


Figure 8: Impact of flow structure and overhearing ($N = 6$)

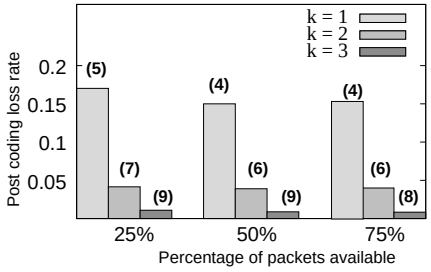


Figure 9: Tradeoffs involved in choosing the value k for CLONE-kDSP

CLONE-k-Loop: In this scenario, CLONE-k-Loop coding scheme identifies 2 loops of length 3: $N_1 \rightarrow N_3 \rightarrow N_5$ and $N_2 \rightarrow N_4 \rightarrow N_6$. This results in a total of six transmissions ($P_1 \oplus P_3$, $P_3 \oplus P_5$, $P_5 \oplus P_1$, $P_2 \oplus P_4$, $P_4 \oplus P_6$, $P_6 \oplus P_2$), thus providing each of the next-hops with 2 ways to decode the packet. As shown in Figure 5 this results in a considerable decrease in the post coding loss rate when compared to the no coding case. For example, when the link loss rate is 10%, the post coding loss rate is reduced to 2%. Further, we observe that the post coding loss rates for all the coding schemes increase with increase in the link loss rate.

CLONE-kDSP : The CLONE-kDSP algorithm transmits the following sequence of packets, given the requirement of providing 2 edge-disjoint paths (i.e. $k = 2$): $P_3 \oplus P_4$, $P_5 \oplus P_3$, $P_1 \oplus P_4$, $P_1 \oplus P_5$, $P_6 \oplus P_1$, $P_2 \oplus P_6$, $P_3 \oplus P_2$. Note that CLONE-kDSP in this case sends out 7 packets instead of six i.e., it increases the number of transmissions by one. However, as shown in Figure 5, this results in post-coding loss rates lower than that provided by CLONE-k-Loop coding (CLONE-kDSP has a loss rate of 0.9% when link loss rate is 10%, while the corresponding loss rate for CLONE-k-Loop is 2%). This is because of the fact that even though CLONE-kDSP with $k = 2$ guarantees providing only 2 edge-disjoint paths, there might be other partially edge-disjoint paths through which a next-hop might be able to decode the packet. For example, in this case N_1 can decode P_1 using any of the following decoding sequences: (1) $P_1 \oplus P_4$, (2) $P_6 \oplus P_1$, $P_6 \oplus P_2$ or (3)$P_1 \oplus P_6$, $P_6 \oplus P_5$, $P_5 \oplus P_3$.

CLONE-MultiXOR : The CLONE-MultiXOR coding algorithm takes advantage of the fact that more than 2 packets can be coded together. For the topology in Figure 4, CLONE-MultiXOR transmits a total of six coded packets, $C_1 : P_1 \oplus P_4$, $C_2 : P_2 \oplus P_5$, $C_3 : P_1 \oplus P_2 \oplus P_5$, $C_4 : P_4 \oplus P_5 \oplus P_6$, $C_5 : P_2 \oplus P_3 \oplus P_6$ and $C_6 : P_3 \oplus P_4 \oplus P_5 \oplus P_6$. In Table 3, we show the corresponding decoding sequences (i.e. the different ways in which a packet can be decoded by a next-hop). Elements of a decoding sequence when XORed together result in decoding the native packet either directly or by a few other XORs with the native packets already present with the next hop. For e.g. next-hop N_1 can decode packet P_1 using ($C_1 \oplus P_4$) or ($C_2 \oplus C_3$) or ($C_3 \oplus C_4 \oplus C_5 \oplus P_3 \oplus P_4$) or ($C_3 \oplus C_5 \oplus C_6 \oplus P_4$). As shown in Table 3, although the total number of packets transmitted by the relay remain the same as no coding case (i.e. 6 transmissions), on an average each next hop is able to decode its packet in *four* ways. CLONE-MultiXOR coding scheme therefore achieves the lowest post coding loss rates (Figure 5) amongst the CLONE schemes discusses. For e.g., when the link loss rate is 10%, the loss rate for CLONE-MultiXOR is 0.6%, lower than that of CLONE-k-Loop (2%) and CLONE-kDSP (0.9%).

4.2 Results on Random Network Topologies

We now present the simulation results on randomly generated network topologies.

Simulation setting: We generate the network topologies for evaluation purpose as follows: We place the relay node R at the origin and define a transmission radius r. We then randomly choose a point at a distance d, where $r/2 \leq d \leq r$ and place a neighboring node with transmission radius equal to d. For each neighboring node N, we randomly

select a node M (not in the range of N) and set up a flow from M to N which is routed through the relay R. The set of packets already available at each of the neighboring nodes (for use in coding) is then calculated based on the transmission ranges. For determining the available coding opportunities, we assume that the relay node has a packet to send to each of its neighboring nodes (which is true under steady state assumptions). We varied the loss rates from 0 to 30%. We evaluated the CLONE schemes presented in the paper on a variety of topologies generated using this approach and found that a number of factors affect the post coding loss rates:

Evaluation #1 – Post coding loss rates: Figure 6 shows the post coding loss rates for CLONE-k-Loop ,CLONE-kDSP and CLONE-MultiXOR in a network where a relay is surrounded by $N = 6$ nodes. As the link loss rates increase, the post coding loss rate also increases. For this topology, we observed that CLONE-k-Loop and CLONE-MultiXOR schemes achieve a considerable amount of reduction in the loss rates while requiring the same number of transmissions as no coding case (i.e. 6 transmissions). On the other hand, while CLONE-kDSP algorithm (with $k = 2$) increased the number of transmissions from 6 to 7 (i.e. it transmitted an extra packet), it performed the best among the CLONE schemes in terms of post coding loss rates. For e.g., as shown in Figure 6, for a link loss rate of 20% in this network, the post coding loss rate for CLONE-kDSP is 4.7% which is lower than that of CLONE-k-Loop (9%) and CLONE-MultiXOR (6.1%).

Evaluation #2 – Impact of flow structure and overhearing: Figure 8 shows the post coding loss rates for CLONE coding schemes on a network with a relay surrounded by $N = 6$ nodes. The link loss rate was set to 20%. We plot the results for varying percentage of packets available at the neighboring nodes. A greater percentage of packets might be available with the surrounding nodes because of an increased overhearing phenomenon in the broadcast wireless medium. We observe that the post coding loss rates decrease with increase in the percentage of available packets due to increased coding opportunities. As shown in Figure 8, the CLONE-MultiXOR coding algorithm lowers the post coding loss rate to 11.4% when 20% of the packets are available and further to 2% when 60% of the packets are available.

Evaluation #3 – Degree of the relay node: For this simulation, we created a topology where a relay is surrounded by $N = 4$ nodes and calculated the post coding loss rates as before. We then successively added 2 nodes to the topology, creating a network with relay surrounded by $N = 6$ and $N = 8$ nodes. Figure 7 shows the results for CLONE-MultiXOR algorithm for varying percentages of the number of packets already available at the neighboring nodes. We see that while the post coding loss rate decreases with the increase in the percentage of available packets at the neighboring nodes, the decrease is much more for a network with higher degree of nodes. This is because greater the degree of the relay node, greater is the possibility of coding several packets together. Figure 7 shows that the post coding loss rates are the lowest for $N = 8$.

Evaluation #4 – Tradeoffs: redundancy Vs transmission minimization: Figure 9 shows the post coding loss rates for CLONE-kDSP on a network where a relay is surrounded by $N = 6$ nodes. The results are plotted for values of $k = 1$, $k = 2$, and $k = 3$ for varying percentage of packets already available at the neighboring nodes. The numbers in parenthesis show the total number of packets transmitted by the relay in each case. The link layer loss rate for this simulation was 20%. We observe that for $k = 1$, CLONE-kDSP reduces the number of packets that need to be transmitted by the relay. Further, greater the percentage of already available packets at the receiver, more is the reduction. Surprisingly, in this case, we also see that $k = 1$, not only reduces the number of transmissions, but also decreases the loss rate. We note that, while CLONE-kDSP algorithm guarantees k disjoint paths, in general there might be more partially disjoint paths available. We also observe that increasing the value of k, while reducing the post coding loss rate considerably can result in increased number of packets that need to be transmitted. Thus, the degree of reliability (determined by the value of k) that can be provided by CLONE-kDSP must be carefully chosen based on the observed link layer loss rates and the set of packets already available at the neighboring nodes.

Summary: We observed that while CLONE schemes achieve low post coding loss rates, the amount of improvement depends on a number factors such as the flow structure, overhearing pattern, degree of the nodes and the link loss rates. In general, we find that among the schemes considered, CLONE-MultiXOR performs the best in terms of improving the loss rates. Algorithms like CLONE-kDSP show that it is important to choose a coding strategy that balances the tradeoffs between redundancy and transmission minimization. However, it is promising to note that even simple schemes like CLONE-k-Loop provide a considerable reduction in the loss rates.

5. IMPLEMENTATION

We have implemented CLONEon an real testbed and studied its performance.Our implementation builds upon the concepts of opportunistic listening and opportunistic coding introduced in [8, 18] and extends these architectures by introducing special packet headers, data structures and by changing the control flow of a router to enable CLONE schemes to code and decode packets. In this section, we provide an overview of the design issues involved in developing CLONE and then present the implementation specifics.

5.1 CLONE: Design Issues

We now highlight the system design issues involved in implementing CLONE so that it integrates well into the current 802.11 systems:

(a) Information about neighbor packet pools: In order for the relay to code packets, it must first know what packets are available with the neighboring nodes. Like in COPE [18], nodes periodically exchange *reception reports* which announce the set of packets available in their respective packet pools. When information from these reports is not available, intelligent *guessing* [18] based on the ETX metric [11] (provided by the underlying routing protocol) is used to populate information about the neighbor packet pools.

(b) Determining the set of packets to code: Each node in the system maintains an output queue of packets that are to be transmitted. At a given point of time, there might be many coding opportunities available to relay node. Searching for a coding opportunity might be com-

putationally expensive, as a node might have to try out $\sum_{m=1}^{m=k} {}^{n}C_{m}$ possibilities when using CLONE-k-Loop (although we might limit the length k to a small number). In order to avoid this, each node also maintains a per next-hop virtual queue (a virtual queue essentially contains pointers to the actual packets in the output queue). The CLONE coding procedures only use the packets from head of each virtual queue to determine the coding opportunities. We note that this might lead to reduction in the number of coding opportunities. However, maintaining virtual queues not only enables efficient coding, but more importantly reduces the possibility of reordering within a flow.

(c) Opportunistic coding: The CLONE coding schemes operate on a set of packets (derived from the heads of the virtual queues) and output a set of coded packets based on the available coding opportunities. We identify this set of packets as a *group*. Each group is uniquely identified using a, group_ID, which is incremented for each group. In our implementation, we take an opportunistic coding approach i.e. we look for coding opportunities that can arise with the head of the output queue and in case none are available, we send it out as one of the packets in the group in order to reduce introducing any delays due to coding.

(d) Packet decoding: CLONE schemes allow the possibility of decoding the native packet using multiple encoded packets. This not only requires a node to buffer the received encoded packets, but also increases the computational complexity as the node would now have to try out all possible ways of XORing these encoded packets to determine whether it can decode its packet. CLONE avoids this by adopting a simple approach: a relay node explicitly sends the list of possible *decode sequences* in special headers annotated on each of the coded packets. Decode sequences explicitly mention which coded packets might be needed to decode a certain packet. Thus, a node only buffers packets which can be useful for the decoding procedure. Further, these buffers are flushed either when a packet has been successfully decoded or after a certain timeout. Since, we selectively buffer packets from only the current group of a relay, we observed that this additional memory overhead in our system was negligible.

(e) Broadcast with Asynchronous Acks: Unlike COPE which tries to reduce the number of packets to be transmitted, CLONE schemes try to improve the link loss rate. That is, CLONE schemes *acknowledge* the fact that transmitted packets can be lost and *recover* from it by providing more than one way to decode a native packet. Using 802.11 synchronous acknowledgments would not be efficient because each coded packet in a group might be useful to multiple next-hops, sending of acks by all these next-hops would cause an ack implosion at the relay. We therefore use 802.11 *broadcast* for sending out the coded packets and we recover from the packet decoding failures by using asynchronous MAC level acknowledgments and retransmits.

(f) Retransmission Timeouts: Scheduling a retransmission timeout for a packet now becomes non-trivial. For e.g. consider packets P_1, P_2, .. P_n which are coded into a group of C_1, C_2, .. C_n packets. When should we trigger a retransmit for a packet P_i ? Depending on the decoding sequences generated, it is possible that the intended next-hop R_i might be able to recover the packet on receipt of packet C_n. We therefore adopt a *conservative* approach – we start the retransmission timer for all the native packets when the last packet of a coded group is transmitted. Though this may slightly delay the retransmit event for a particular packet, the approach would avoid unnecessary retransmits that can otherwise happen due to premature timeouts.

5.2 Implementation Details

We present an overview of CLONE's control flow implementation. Each node maintains a packet pool, per next-hop virtual queues and hash table which gives information about the availability of a particular packet at a neighboring nodes (this is populated using reception reports or by guessing). In addition, each node also maintains a hashmap *group buffer* keyed on the IP address of each neighbor, which buffers potentially useful coded packets and the current group number for this node.

CLONE schemes may generate more than one packet per group, the first of these is transmitted immediately while the rest of the packets of the group are enqueued in a separate queue, pending subsequent transmissions. Whenever a transmission opportunity arises, we first check for any pending packets to transmit. In case such a packet exists we dequeue the first one and transmit it, else, we try to form the next group. Figure 13 in the Appendix C depicts this control flow in CLONE during a transmission opportunity.

On a packet reception, the node checks if the packet is useful to it by checking whether its ID is present in one of the group units. If the packet is useful, the node checks if the group ID on the packet is higher than the sender's current group. If it is, then the the packets belonging to older group are flushed, the new packet is buffered and current group is updated. Further, the node checks if it has enough packets decode its native packet using the information from the decode_sequence_array. In case it is able to decode, then it either forwards the packet (if it is an intermediate hop) or it sends the packet to the higher layer (if it is the destination). Figure 14 in the Appendix C shows this control flow.

The CLONE packet header format and details are present in Appendix B. This additional shim header contributes to about 6 - 9% of typical packet lengths.

5.3 Experimental Results

We implemented CLONE using the Click modular router toolkit [1] that allows it to run as a user space daemon in Linux. Our implementation of CLONE currently runs on a 12-node testbed spanning three floors of our building. Each node on the testbed is a Soekris 4326 [2] running the 2.6.6.19 Linux kernel, and is equipped with Atheros 5212 mini-PCI wireless card. We used the Srcr [7], as the underlying routing protocol which is based on the ETX [11] metric. In all our experiments we measure the aggregate UDP throughput over 802.11g running at 1 Mbps PHY data rate, for a packet size of 1000 bytes.

We first report on the computational complexity of all the three network coding algorithms proposed in Section 3 which shows that processing is still a limiting factor for the CLONE-MultiXOR and the CLONE-kDSP algorithms. Therefore, in the testbed we have deployed a full implementation of the CLONE-k-Loop algorithm alone. The other two algorithms are likely to also provide significant performance gains as our simulation results indicate, and we expect the continued increase in processing speeds to facilitate its feasibility in the future.

Scheme Neighbors	COPE	CLONE-k-Loop	CLONE-kDSP	CLONE-MultiXOR
4	25	37	714	13,489
6	74	83	3,801	31,894

Table 4: Computational complexity of different algorithms. Time reported in μs.

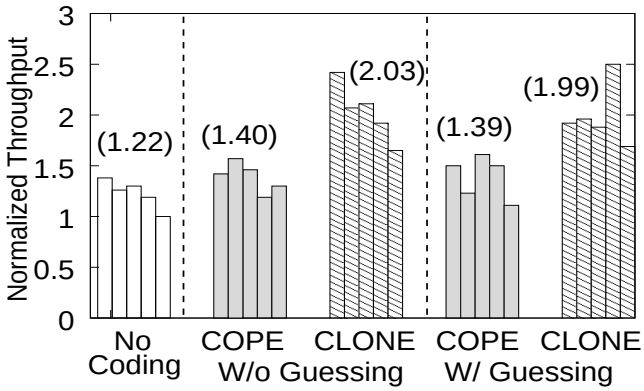


Figure 10: Throughput gains of CLONE and COPE for a 4-node illustrative topology. The numbers in the parenthesis represent the average throughput for each of the cases. Five runs of each are shown.

In our experiments, we first compare the performance of CLONE-k-Loop to both no coding and COPE in a representative four node topology. Subsequently, we explore the performance gains of our CLONE-k-Loop algorithm on the full 12-node testbed.

5.3.1 Computational complexity

The first task in understanding the performance of the three proposed algorithms is the study their computational complexity. Table 4 presents our measurements on the computation times for each algorithm, running within the user level daemon, when combining packets for four and six neighbors, running on a relay node.

Both COPE and CLONE-k-Loop finish their computation within 83 μs. The currently popular 802.11a/g standards use a DIFS value of 34 μs, default CWmin value of 16, and a slot time duration of 9 μs, leading to an average time duration between back-to-back packets of 106 μs. Thus, in a firmware level implementation with careful optimized software optimizations, CLONE-k-Loop should not add any noticeable delays in the data path. The same is not true for the other two schemes. Hence, we believe that further improvements in processing speeds need to occur before the performance gains of these two schemes can be realized in a real wireless system.

5.3.2 Results on a 4-node Illustrative Topology

We now present the relative performance of no-coding, CLONE and COPE schemes on a 4-node topology similar to the one shown earlier in Figure 2, where a relay N routed flows for three of its neighboring nodes. Figure 10 plots the relative throughputs of no coding, COPE and CLONE , the latter two with both guessing enabled and disabled, for 5 different runs. All throughputs were normalized to the lowest performing scenario among all runs. Also, the measured throughputs take the overhead due to CLONE header(∼9%) into account. We observed that throughput improvements when using CLONE were upto a factor of 2, with an average throughput gain of around 57%. The average throughput gains with COPE over no coding were around 21%. This is because, COPE has very few coding opportunities (available due to overhearing) while CLONE formed a simple 3-Loop. In this scenario, CLONE sends out groups of 3 coded packets and each node can decode a packet in two ways. The performance gains stem from the fact that not only does CLONE tolerate losses (of more than 33% on an average in this case), but also avoids the overheads involved in retransmissions. We further note that, the gains of CLONE and COPE due to guessing vary – when CLONE correctly guesses the information about the neighbor packet pools the gains are signficant (a gain of $2x$ in run 4), on the other hand throughput gains would lower down when it fails to guess correctly.

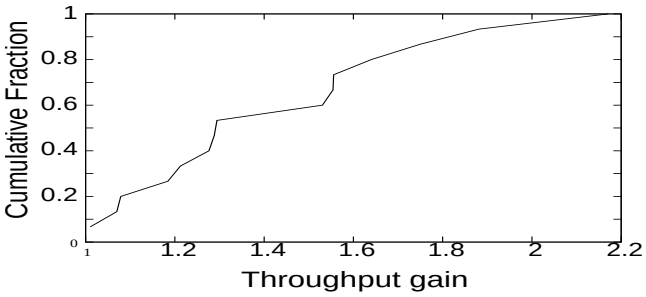


Figure 11: CDF of UDP Throughput gains for CLONE over no coding.

5.3.3 Results on a 12-node Testbed

We repeated the UDP throughput measurements on our 12-node testbed. We set up UDP flows by choosing the source-destination pair in a random fashion. The date rate for each flow was set to 250 kbps and the flows lasted for a duration of 120 seconds. We observed that most of the flows in this experiment were 2-hop flows, while a few of them being 3-hop flows. We the measured the aggregate UDP throughput with and without CLONE. We enabled guessing for CLONE, in order to be able to exploit the ETX information provided by the Srcr routing protocol. Figure 11 shows the CDF of UDP gain for CLONE over the no coding case. We observe that the throughput gains vary from around 10% to more than $2x$. We note that the main source of improvement comes from the fact that CLONE is able to recover from wireless losses, which were of the order of 30% in our experiments. The average throughput gains observed for CLONE were around 43%.

6. RELATED WORK

Network coding was first proposed by Ahlswede et. al. [6] for multicast communication. Subsequently, a large body of work has studied various ways of creating efficient network codes [20, 19, 10, 23, 13]. In wireless environments, network coding has been employed to reduce energy costs [22] and to maximize total flow in the network [4]. Jaggi et. al. [14] have studied the resilience of network coding mechanisms to

attacks under assumptions of various malicious forwarding nodes. These results are quite complementary to our work.

In the context of unicast sessions on multi-hop wireless settings, the benefits of network coding has been demonstrated by Li and Li [21], Wu et. al. [26], and by Katti et. al. in COPE [18]. In particular, COPE demonstrates the potential of the performance gains achievable through *local* network coding and by using XOR operations and is the starting point for the work presented in this paper. The key difference between COPE (as a representative of other such prior work) and the results of this paper is the following. While COPE assumes losses do happen on the wireless path, it does not directly provide protection against losses on the wireless path *through use of network coding structures.* In contrast, our design explicitly uses redundancy in design of network coding structures to guard against the negative impacts of such losses. In particular, we show that the objective of minimizing the number of transmissions needed to carry the information over multiple wireless hops does not lead to best throughput gains. Often, redundant transmissions of packets, in which redundancy is provided through network coding, can lead to significant throughput gains. Thus, the key difference of our work from prior work is in explicit *loss-awareness* on the wireless paths in the design of coding structures.

7. CONCLUSIONS

Network coding as a primitive has been shown to be quite useful in improving traffic carrying capacity in many wired networking scenarios and have direct applications in wireless environments as well. However, to gain the best benefits of this primitive, such a primitive should not be applied in isolation. Instead, it should be effectively combined with characteristics of the environment. This paper shows that network coding, when applied to wireless environments, can lead to further performance gains when it is effectively made aware of the potential losses on different wireless links. We believe that this work should be treated as a starting point for significant further research in understanding interactions of network coding with different aspects of the wireless protocol stack. The availability of the network coding primitive should lead us to revisit the design of protocol mechanisms at different layers, including rate control and contention resolution at the MAC layer, routing and forwarding decisions to facilitate greater coding opportunities at the network layer, congestion control mechanisms at the transport layer, and application-specific coding mechanisms at the application layer as well.

Acknowledgement

We wish to thank our shepherd Zhi-Li Zhang and anonymous reviewers for their helpful comments. S. Rayanchu, S. Sen and S. Banerjee were supported in part by NSF awards CNS-0639434, CNS-0627589, CNS-0627102, CNS-0520152, and CNS-0747177.

8. REFERENCES

[1] The click modular router. `http://www.read.ucla.edu/click/`.

[2] Soekris engineering. `http://www.soekris.com/net4826.htm`.

[3] *Network Flows: Theory, Algorithms, and Application.* Prentice Hall, 1993.

[4] R. A., S. J., and W. R. D. On the capacity of network coding for random networks. In *Allerton Conference on Communications*, 2003.

[5] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM '04*. ACM, 2004.

[6] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yenug. Network information flow. In *IEEE Transactions on Information Theory*, 2000.

[7] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *ACM Mobicom*, 2005.

[8] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *ACM Sigcomm*, 2005.

[9] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM SIGCOMM*, 2007.

[10] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Allerton Conference on Communications*, 2003.

[11] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, 2003.

[12] Q. Dong, J. Wu, W. Hu, and J. Crowcroft. Extended abstract: Practical network coding in wireless networks. In *Mobicom*, 2007.

[13] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. In *IEEE Transactions on Information Theory*, 2005.

[14] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. MÃl'dard. Resilient network coding in the presence of byzantine adversaries. In *INFOCOM*, 2007.

[15] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *ACM SIGCOMM*, 2006.

[16] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: Analog network coding. In *ACM SIGCOMM*, 2007.

[17] S. Katti and D. Katabi. Mixit: The network meets the wireless channel. In *ACM HotNets*, 2007.

[18] 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.

[19] R. Koetter and M. Medard. An algebraic approach to network coding. In *IEEE Trans. Networking*, 2003.

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

[21] Z. Li and B. Li. Network coding: The case for multiple unicast sessions. In *Allerton Conference on Communications*, 2004.

[22] D. S. Lun, N. Ratnakar, R. Koetter, E. A. M. Mdard, and H. Lee. Achieving minimum cost multicast: A decentralized approach based on network coding. In *IEEE INFOCOM*, 2005.

[23] M. Medard, M. Effros, T. Ho, and D. Karger. On coding for non-multicast networks. In *Allerton Conference on Communications*, 2003.

[24] L. Valiant. The complexity of enumeration and reliability problems. In *SIAM Journal on Computing*, 1979.

[25] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. In *Combinatorica*, 1995.

[26] Y. Wu and S. Chou, P.A. Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. Technical Report, MSR-TR-2004-78, Microsoft Research, 2004.

[27] S. Zhang, S. C. Liew, and P. Lam. Hot topic: Physical-layer network coding. In *Mobicom*, 2006.

APPENDIX

A. Pseudo-code for CLONE-MultiXOR algorithm

Algorithm 1 CLONE-MultiXOR

Procedure CLONE-MultiXOR (P, K):
$t_i = P_i \cup \{P_i \oplus COMB(K_i)\}$
$T = \{t_1, t_2, ..t_n\}$
$S = \phi$ *//set of packets selected for transmission*
$R = \phi$
while $|S| \leq n$ **do**
update_usefulness(T, S, R)
select_node(T, R)
end while

Procedure select_node(T, R):
$d_{max} = MAX_{usefulness\_count}\{T - R\}$.
Let $d_{max} \in t_i$.
//if $d_i.usefulness\_count = d_j.usefulness\_count$, *then* $d_{max} = d_i$ *if* $|t_i| < |t_j|$ *otherwise* $d_{max} = d_j$
$S = S \cup \{d\}$ *//select the packet*
$R = R \cup \{t_i\}$ *//eliminate row*

Procedure update_usefulness(T, S, R):
for each element d in $S - R$ **do**
$d.usefulness\_count = \sum_{i=1}^{i=n} \{NPATH(P_i, N_i, S \cup \{d\}) - NPATH(P_i, N_i, S)\}$
end for

B. CLONE Packet format

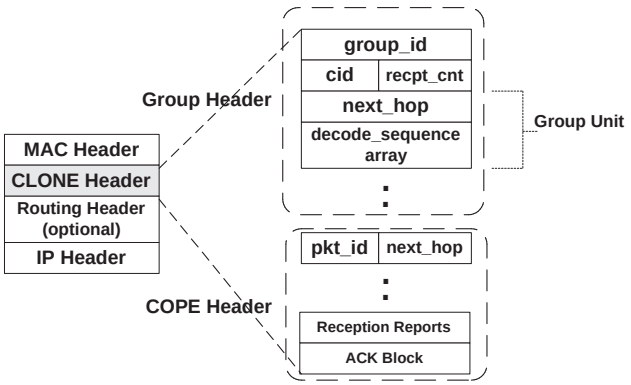
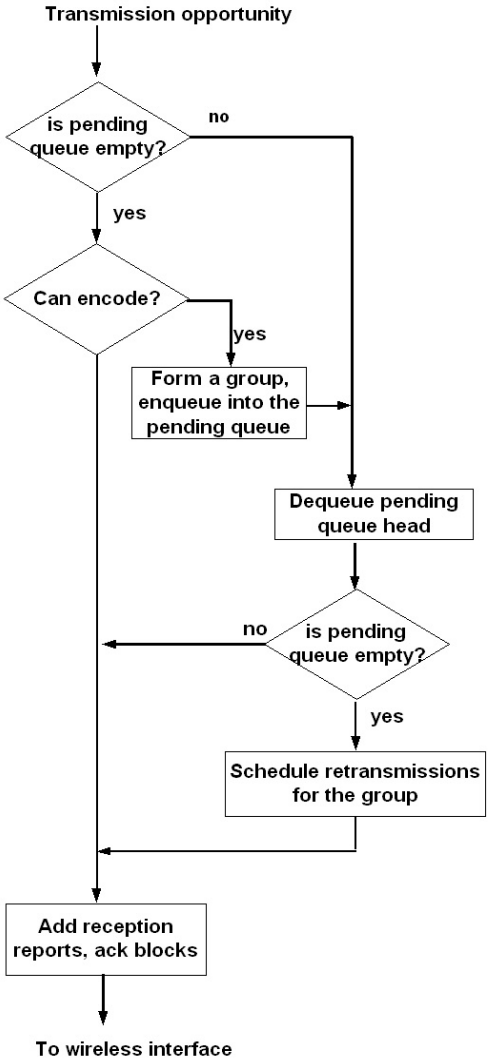


Figure 12: CLONE Header

The packet format of CLONE includes reception reports, asynchronous ack blocks and the XOR headers (stating what are the native packets being coded in this coded packets) compactly in a shim header inserted between the MAC and the IP layer (see Figure 12). This format is analogous to COPE. In addition to these headers, CLONE schemes require a group header, which is of variable length. The format of the group header is shown in Figure 12. It consists of a group ID which when coupled with the IP address, would uniquely identify a group of coded packets. cid represents the id of the coded packet within this group and recpt_cnt informs the number of *group units* that will follow this header. The recpt_cnt essentially represents the number of receivers to whom this coded packet might be useful to. As shown in Figure 12, each group unit contains nexthop which is the ID of the receiver to whom this packet can be useful. It is followed by a decode_sequence_array which is an array of bitmaps representing the set of coded packet ids that have to be XORed to obtain the native packet. In our implementation, we have restricted the size of the array to 4 (maximum number of ways a next-hop can decode a packet) and the size of the bitmap to 16 (maximum number of coded packets per group).

C. Control flow in CLONE implementation

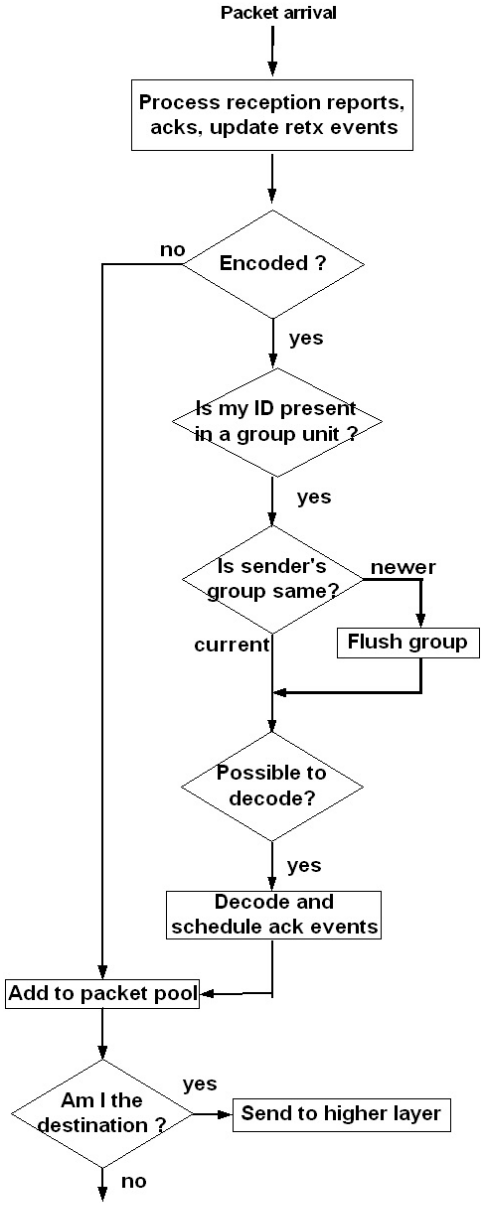
The following figure shows the control flow as implemented on the sender side (relay node) in CLONE.



(a) Sending side

Figure 13: Control Flow : Sender Side

The next figure shows the control flow as implemented on the receiver side (destination node) in CLONE.



(b) Receiving side

Figure 14: Control Flow : Receiver Side