The Importance of Being Opportunistic: Practical Network Coding for Wireless Environments

Sachin Katti Dina Katabi Wenjun Hu Hariharan Rahul Muriel Médard
{skatti@, dk@, wenjun@csail., rahul@csail., medard@}mit.edu

Abstract

This paper applies network coding to wireless mesh networks and presents the first implementation results. It introduces COPE, an opportunistic approach to network coding, where each node snoops on the medium, learns the status of its neighbors, detects coding opportunities, and codes as long as the recipients can decode. This flexible design allows COPE to efficiently support multiple unicast flows, even when traffic demands are unknown and bursty, and the senders and receivers are dynamic. We evaluate COPE using both emulation and testbed implementation. Our results show that COPE substantially improves the network throughput, and as the number of flows and the contention level increases, COPE's throughput becomes many times higher than current 802.11 mesh networks.

1 Introduction

An opportunistic approach to network coding increases the throughput of wireless mesh networks by many folds, a lesson we have learned from the first implementation of network coding in the wireless environment. Starting with the pioneering work of Ahlswede et al [2], which shows that having intermediate nodes in the network mix information from different flows can achieve the broadcast capacity, and including more recent papers which address a variety of coding and decoding options [8, 13–16, 19–21], network coding has established its theoretical ability to improve network throughput. Despite that, very few implementations exist [4, 5], and none for the wireless environment. This paper focuses on how to make network coding work in a mesh wireless network.

How does one apply network coding to a multi-hop wireless network? The current state-of-the-art emphasizes analytical tractability, and thus would first translate the wireless network into a graph where an edge between two nodes means that the radio range allows the two nodes to communicate. Next, one would assume multicast communication, as network coding of multiple unicast flows remains a largely unknown territory. It is also common to assume that the sender and receivers are fixed and given, and that the traffic rates (or distributions) are known, and do not change. In this framework, a few papers show how to run a min-cost flow optimization to find the optimal routing (say the one that maximizes the throughput) [22, 25]. The routing dictates which packets to code together: each node generates linear combinations of the packets on its incoming edges and broadcasts them to neighbors on the outgoing edges.

Unfortunately, wireless mesh networks do not comply with any of these assumptions. First and most importantly, traffic is unicast. Second, senders and receivers are unknown *a priori*; they do not signal their desire to communicate, but just start sending packets. Also, traffic is usually bursty, and the sending rate is unknown in advance even to the sender itself, and varies over time. Also, connectivity in a wireless network is highly variable due to changing channel and medium conditions, and the shared nature of the wireless medium prevents two nearby nodes from transmitting successfully at the same time. In practice, the wireless environment is highly unpredictable and difficult to capture using available models.

This paper introduces COPE, a completely opportunistic approach to network coding. Each wireless node relies on local information to detect and exploit coding opportunities in realtime. In our scheme, all nodes participate in *opportunistic listening*, i.e., they snoop on all communications they hear over the wireless medium. The nodes also annotate the packets they send to tell their neighbors which packets they have heard. When a node sends, it uses its knowledge of what its neighbors have received to perform *opportunistic coding*; the node can XOR multiple packets and send them in a single transmission if each intended receiver has enough information to decode its packet.

Our approach exploits the shared nature of the wireless medium which, for free, broadcasts each packet in a small neighborhood around its path. This creates an environment conducive for coding because nodes in each area have a large and partially overlapping reservoir of packets they can use to decode. For example, Fig. 1 shows how two flows traversing different paths can be encoded

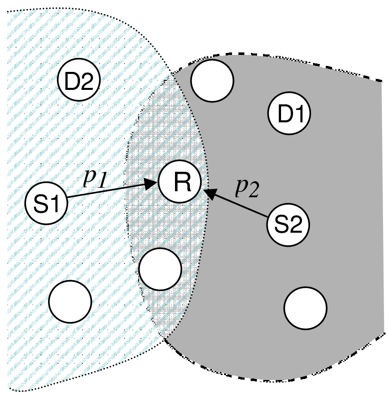


Figure 1: An example of a coding opportunity. There are two flows S1→D1 and S2→D2. Arrows are transmissions, and shaded regions show radio range. D1 hears p_2, and D2 hears p_1. Node R XORs p_1 and p_2 and broadcasts the XOR-ed version. Each destination XORs again with the packet it has heard to obtain the packet destined to it. Thus, R can deliver both packets in a single transmission.

together. By snooping, nodes D_1 and D_2 obtain a copy of the packets traversing their neighborhood. If node R knows what its neighbors have heard, then it can XOR p_1 and p_2 and broadcast the resulting packet. This single transmission allows both destinations D_1 and D_2 to recover their intended packets by XOR-ing with packets they have already heard.

We evaluate our approach using both emulation and real-world implementation. Our results show that an opportunistic application of network coding can substantially improve the throughput of current mesh wireless networks. Depending on the degree of congestion and the number of distinct flows, the throughput with coding may be many times higher than without coding.

This paper has the following contributions.

- It presents the first implementation of network coding to the wireless environment. Our preliminary experiments on a 3-nodes testbed show that network coding almost doubles the throughput.
- It introduces COPE, a network coding protocol for multiple wireless unicast flows. This protocol is online, distributed, and deployable.
- Finally, this paper discusses the challenges in applying the theoretical results in the field of network coding to today's wireless implementations, and presents a set of practical solutions including *pseudo-broadcast*.

2 Requirements

To enable a practical application of network coding to multi-hop wireless networks, one needs to address the following problems:

- *Network coding for unicast applications:* Though most of the theoretical results in network coding are for multicast, the vast majority of Internet traffic is unicast. An application of network coding to the wireless environment has to address multiple unicast flows, if it has any chance of being used. Unfortunately most of the theoretical results on unicast are negative [12, 27]. In particular, with multicast, all receivers want all packets. Thus intermediate nodes can encode any packets together, without worrying about decoding which will happen eventually at the destinations. In contrast, in unicast, packets from multiple flows may get encoded together at some intermediate node, but later their paths may diverge, at which point they need to be decoded. If not, unneeded data will be forwarded to area where there is no interested receiver, wasting much bandwidth.

- *Coping with bursty traffic and dynamic environments:* Prior theoretical work on network coding shows that if the senders, the receivers, and the traffic demands are known *a priori*, it is possible to run a distributed optimization to find the optimal coding strategy [22]. In reality, users start transmitting immediately without allowing time for route optimization to converge. Further, the traffic is usually bursty and the set of senders and receivers keeps changing over time.

- *Broadcast with collision avoidance:* In wireless environments, network coding relies on the broadcast nature of the medium to deliver a single encoded packet to multiple receivers. However, in contrast to unicast, 802.11 broadcast has no collision detection or avoidance mechanism. As a result, broadcast works badly in congested environments where the collision probability is high. However, these are the exact environments that benefit from network coding and its ability to send more information for less bandwidth. One may change the MAC layer completely, but for the short term it may be more desirable to make network coding work with 802.11 as this allows for a practical implementation using off-the-shelf hardware/drivers.

- *Low complexity encoding and decoding:* Traditional network coding uses operations over large finite fields. Decoding operations have quadratic complexity, which becomes too slow for high throughput applications. Further encoding operations are also complicated since they involve multiplications in large finite fields. This makes their use in high throughput applications questionable. Encoding/decoding algorithms should have linear complexity for practical implementation.

- *Working properly with TCP:* Most applications run on top of TCP. Hence, it is essential that the coding scheme has no adverse impact on TCP performance. Two issues are particularly relevant: loss recovery and packet reordering. First, TCP interprets a packet loss as a signal of congestion to which it reacts by halving the transmission rate. Since wireless links usually have higher error rates than what TCP can handle, the 802.11 MAC retransmits lost packets locally at each hop, to mask those losses from TCP. Network coding uses broadcast to deliver multiple packets in a single encoded transmission. In this case, it is unclear how the receivers should ack the reception of their packets. One would need either a mechanism that takes care of the delivering the acks to the sender, or to add enough redundancy to bring the delivery rate to the level acceptable to TCP. Second, since TCP relies on the packet sequence numbers to detect losses, it may confuse packet reordering as a sign of congestion. Thus, a coding scheme that causes packets from the same flow to get out of order may trigger TCP congestion backoff, resulting in low throughput.

3 COPE

Our protocol is designed for wireless mesh networks. It uses network coding for unicast traffic. It assumes no synchronization or prior knowledge of senders, receivers, or traffic rates, any of which may vary at any time. The main characteristic of our approach is opportunism: each node relies on local information to detect and exploit coding opportunities whenever they arise. The scheme has two components: opportunistic listening and opportunistic coding.

3.1 Opportunistic Listening

Wireless is a broadcast medium, creating many opportunities for nodes to hear packets even when they are not the intended recipient. We make all nodes in the network store all packets they hear for a limited amount of time T. For maximal benefit, T should be larger than the maximum packet latency, which is usually on the order of tens of milliseconds. The memory requirements for such storage are relatively low. For example, assuming an 802.11 capacity of 11 Mb/s and a conservative value for T like 500ms, the total amount of storage required is less than 750 kilobytes, which is easily available on PCs, laptops, or PDAs. We call this function *Opportunistic Listening*.

In addition, each node broadcasts *reception reports* to tell its neighbors which packets it has stored.[1]. Reception reports are sent by annotating the data packets the node transmits. A node that has no data packets to transmit periodically sends the reception reports in special control packets.

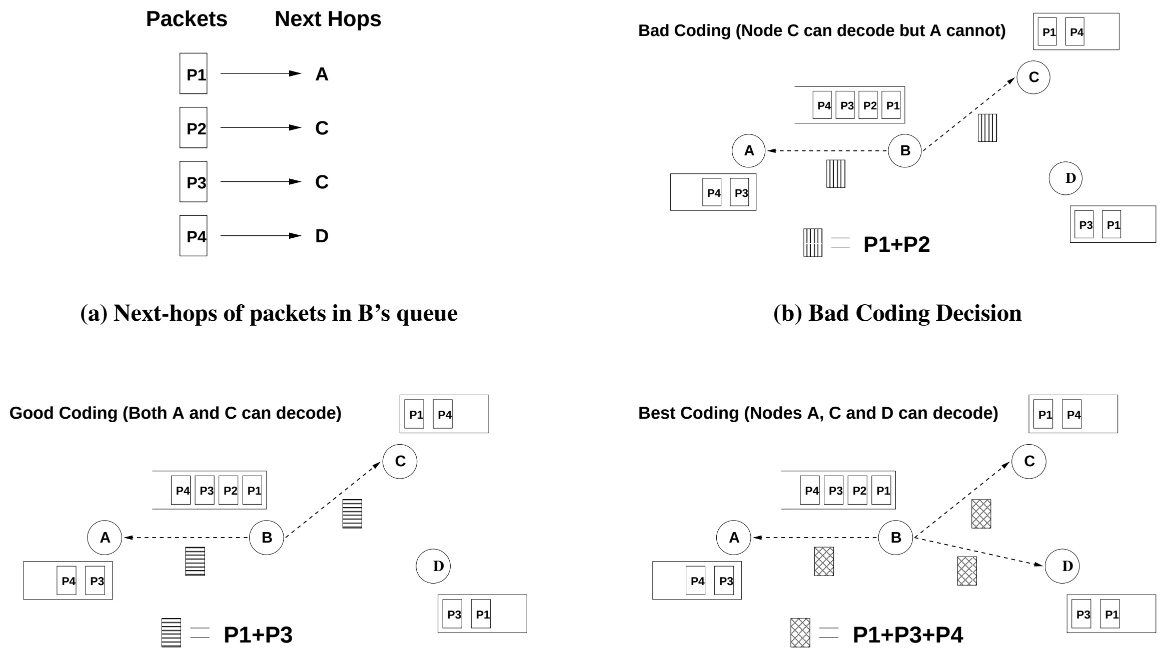
3.2 Opportunistic Coding

The main issue our protocol has to solve is: what packets to code, and how? Each node should answer this question based on local information and without consulting with other nodes. As in current wireless implementations, each node maintains a FIFO queue of packets to be forwarded. When the MAC indicates that the node can send, the node picks the packet at the head of the queue, checks which other packets in the queue may be encoded with this packet, XORs those packets together, and broadcasts the XOR-ed version.

The question, however, is which packets to XOR together to maximize the throughput. A node may have multiple coding options. It should pick the one that maximizes the number of packets delivered in a single transmission. This is best illustrated with an example. In Fig. 2, node B has 4 packets in its forwarding queue p_1, p_2, p_3, and p_4. The table in Fig 2-a shows the next-hop of each of these packets. When the MAC signals to B to transmit, B picks packet p_1 from the head of the queue to transmit it. Assume B knows which neighbor has heard which packets. Now node B has a few coding options; as shown in Fig. 2-b, it could send $p_1 \oplus p_2$. Since C has p_1 in its storage space, it can XOR it again with $p_1 \oplus p_2$ to obtain the packet sent to it, i.e., p_2. But, node A does not have p_2, and thus cannot decode the encoded packet. Thus, if B sends $p_1 \oplus p_2$, it will be a bad coding decision because only one neighbor can benefit from this transmission. Fig. 2-c shows a better coding decision for B. Sending $p_1 \oplus p_3$ allows both neighbors C and A to decode and obtain their intended packets, delivering two packets in a single transmission. But the best coding decision for B would be to send $p_1 \oplus p_3 \oplus p_4$ which allows all three neighbors to receive their intended packets, as shown in Fig. 2-d. In general, a relay node should check various packet combinations to find the largest number of packets that can be delivered in one transmission while still allowing each of the intended recipients to decode its packet.

The above example indicates a simple rule for choosing which packets to code together.

[1]We concatenate the IP address of the sender with the IP sequence number to generate a packet id.



(a) Next-hops of packets in B's queue

(b) Bad Coding Decision

(c) Better Coding Decision

(d) Best Coding Decision

Figure 2: Example of Opportunistic Coding; Node B has 4 packets in its queue, whose next-hops are given in (a). Each neighbor of B has some packets in its storage space. Node B can make a number of coding decisions (b,c,d), but should pick the one in (d) because it maximizes the number of packets delivered in a single transmission.

> To transmit n packets, $p_1, ..., p_n$, to n recipients, $r_1, ..., r_n$, a node can XOR the n packets together only if each intended recipient r_i has all $n-1$ packets p_j for $j \neq i$.

This rule ensures that each next-hop can decode the XOR-ed version and extract the packet intended for it. Whenever a node has a chance to transmit a packet, it tries to find the largest n that satisfies the above rule, i.e., it tries to maximize the number of packets delivered in a single transmission.

But how does a node know what packets its neighbors have? §3.1 describes how a node announces the ids of the packets it stores in special reception reports, allowing its neighbors access to this information. Still, in times of severe congestion, many reception reports may get lost. Further, in time of light traffic, the reception reports may arrive too late, after the node has already made a suboptimal coding decision. Thus, in addition to using the information in the reception reports, each node makes informed guesses about the status of its neighbors. Our scheme allows only one form of guessing: if a node receives a packet transmitted by neighbor A, it assumes that nodes closer to A than itself have also received the packet.[2] Guessing enables nodes to make smarter encoding decisions. More packets could get encoded together as a result, thereby saving wasteful transmissions. On the other hand if the guess is incorrect, the coded packet will not be decodable at some of the intended next-hops. Packets meant for these next-hops will have to be transmitted again, potentially encoded with a new set of packets.

The coding scheme above has a few important characteristics. First, there is no scheduling or assumed synchronization. Second, no packet is delayed; every time the node sends a packet it picks the head of the queue as it would have done in the current approach. The difference is that, when possible, the node tries to overload the transmission with additional information. Third, the scheme does not cause packet reordering as it considers the packets according to their order in the FIFO queue, for both transmission and coding. This is particularly important for TCP flows, which may mistake packet reordering as a congestion signal.

[2] A node learns the location of other nodes in the network by having all nodes flood link state information. This contains the distance to each neighbor measured using the ETX metric [7]. Alternatively, the nodes may use GPS to locate themselves. Then they flood their location to other nodes in the network.

3.3 Opportunistic Routing

Can we further reduce the number of transmissions and improve the throughput? The previous sections assume that the path taken by a packet is pre-decided by the routing protocol. But suppose the routing protocol decides to send packets from node S to D along the path $S \rightarrow A \rightarrow B \rightarrow D$. The routing might have picked this path because, on average, it has the highest delivery probability. Say, however, that when S transmits packet p_i, it happens that both nodes A and B hear the packet. In this case, it would be a waste to have A forward the packet again to B. This observation has been noted in [3] and used to develop an opportunistic routing protocol for mesh wireless networks.

We note a synergy between opportunistic routing and opportunistic coding. An intermediate node encoding packets together can use the reception reports to check if any of the packets has already been received by a node further down its path. If so, that packet is ignored and not transmitted. Additionally, whenever a node hears a packet not sent to it, it checks the path picked by the routing protocol for this packet. The path can be stored in the packet itself, or may be easy to compute at any node as it is the case for geographic routing with known and fixed node coordinates. If the node discovers that it is on the downstream path, it takes responsibility for forwarding the packet towards its destination. To prevent the loss of reception reports from causing duplicate transmissions, each node stores for a limited period the ids of all packets it transited using this form of opportunistic routing. In the example above, node A would learn from B's reception reports that B has a copy of p_i, and thus A cancels the transmission of p_i to B. Node B also knows that it is the next-hop for packet p_i by checking the path assigned to p_i by the routing protocol. Node B keeps state about transmitting packet p_i. This helps preventing duplicate transmissions in case A does not receive the reception report and transmits p_i to B again.

We note that with opportunistic routing there is some chance of duplicate transmissions and packet reordering. However, the chances are so little that neither did occur in our experiments. §6 shows that though opportunistic routing improves the network throughput, its impact is negligible in comparison with the coding. Thus, if these issues create an adversely effect in some environments, one can turn off opportunistic routing.

4 Pseudo Broadcast

Practical artifacts can cause a naïve implementation of network coding to perform poorly. This section shows that implementing network coding directly on top of 802.11 broadcast may produce lower throughput than no coding at all.

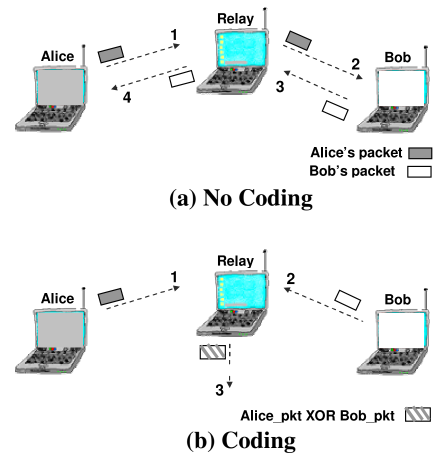


Figure 3: A simplified illustration of COPE, showing how it reduces bandwidth consumption. It allows Alice and Bob to exchange a pair of packets using 3 transmissions instead of 4 (numbers refer to the order of transmission).

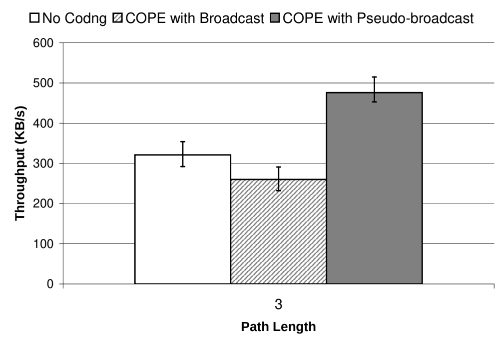


Figure 4: Importance of pseudo-broadcast. The results are for the scenario in Fig. 3. Network coding over 802.11 broadcast has a lower throughput than no coding. In contrast, the throughput with pseudo-broadcast is higher than predicted by the theoretical results because network coding alleviates the mismatch between the input and draining rates at the relay, created by 802.11 MAC's attempt at fairness.

Consider the simple scenario in Fig. 3, where Alice and Bob exchange packets using a relay. The relay XORs Alice's packet with Bob's and broadcasts the XOR-ed version. Both Alice and Bob can obtain each other's packet by XOR-ing the broadcasted packet with the packet they sent. This process requires 3 transmissions. Without coding, the process would require 4 transmissions: Alice sends to the relay which forwards the packet to Bob, and Bob sends to the relay which forwards the packet to Alice. Thus, theoretically, coding should increase the throughput by 25%.

A natural implementation of the coding would use 802.11 broadcast. Fig. 4 shows that this is a bad choice.

The figure compares the total throughput (i.e., the sum of Alice's and Bob's throughputs) under no coding and coding over 802.11 broadcast, and contrasts it against coding over pseudo broadcast, our proposed solution. Surprisingly coding over 802.11 broadcast performs worse than no coding.

To understand why the results in Fig. 4 contradict the theoretical analysis, one needs to grasp the details of the 802.11 MAC, which works in two modes: unicast and broadcast. 802.11 unicast packets are immediately acked by their intended next-hop. The MAC uses the lack of an ACK as a collision signal, to which it reacts by backing off exponentially, allowing multiple nodes to share the medium. In contrast, an 802.11 broadcast packet has many intended receivers, and it is unclear who should ack. Thus, 802.11 broadcast packets are not ack-ed. This means that the sender of a broadcast cannot infer the occurrence of collisions, and thus does not back off. When multiple backlogged nodes share the broadcast channel, each of them continues sending at the highest rate, ignoring the others. The result is a poor throughput caused by high collision rate, like that exhibited in Fig. 4.

Ideally one would design a backoff scheme suitable for broadcast channels (perhaps Idle Sense [11]). But we are interested in an implementation of network coding that can be deployed in the near future. Thus, we need a solution that works well with existing 802.11 cards and drivers.

Our solution, called *pseudo-broadcast*, piggybacks on 802.11 unicast, which has a backoff mechanism. Pseudo-broadcast unicasts packets meant to be broadcast. The link layer destination field is set to the MAC address of one of the intended recipients. An extra header is added after the link-layer header that lists all the intended recipients of the packet. All nodes are put in promiscuous mode, thus they also receive packets not addressed to them. When a node receives a packet with a MAC address different from its own, it checks the extra header to learn whether it is an intended recipient. If so, it does further processing on the packet, else it stores the packet in a buffer as an opportunistically received packet. As all packets are sent using 802.11 unicast, the MAC can detect collisions and backoff properly.[3]

Fig. 4 shows that coding over pseudo-broadcast substantially increases the throughput. Surprisingly, the increase in throughput exceeds the 25% improvement predicted by the theoretical analysis. This is caused by the bandwidth allocation policy of the 802.11 MAC. In an attempt to provide fairness, the 802.11 MAC allocates $\frac{1}{3}$ of the bandwidth to each of the three nodes in Fig. 3. However, without network coding, the relay needs to forward twice as many packets as Alice's and Bob's nodes. This mismatch between the MAC allocated rate and the congestion level of a node is alleviated by network coding, as the larger the number of flows traversing a node, the more opportunities it has to combine multiple packets together in one transmission. In the particular example in Fig. 3a, XOR completely eliminates the mismatch.

[3]The 802.11 MAC uses the link-layer acks also to signal packet loss. The sender retransmits the lost packet increasing the channel reliability. This paper does not address reliability issues, which are the focus of our future work.

5 Addressing the practical requirements

COPE addresses the requirements specified in §2. It allows network coding of multiple unicast flows. This is because it ensures that packets encoded together are always decodable at the next-hop. Hence if two unicast flows diverge at a particular point, they are no longer encoded together.

COPE makes no assumptions about the topology, identity of the senders and receivers, flow rates, arrivals, etc. Each node makes local encoding decisions based on information it has or has received from its neighbors, no global knowledge is necessary. Further, dynamic conditions do not degrade performance – the flexibility that nodes have in making local decisions enables them to adapt on their own.

COPE uses a pragmatic solution to collision problem in 802.11 broadcast. Ideally the MAC protocol should be redesigned to work for broadcast, but this would prevent the coding scheme from being used with present 802.11 software, largely limiting its deployability. We adopt a middle path for this problem by piggybacking broadcast packets on the 802.11 unicast MAC protocol. Hence there is no modification necessary to 802.11 driver software, ensuring interoperability with present systems.

The encoding and decoding algorithms used in COPE are fast, they involve simple bitwise XOR operations. Traditionally, network coding requires operations over large finite fields which are very slow to implement in practice. At very high speeds, these encoding and decoding operations could become the bottleneck for network coding. COPE has linear time decoding and linear time encoding complexity, making it practical to implement for high throughput applications.

One of the design goals for COPE is to ensure interoperability with higher layer protocols and applications, such as UDP, TCP, etc. COPE works fine with UDP as shown by the results in §6. Its interaction with TCP will be the subject of our future work.

6 Performance

We evaluate the performance of COPE using both emulation and real-world implementation.

6.1 Emulation Environment

(a) Emulator: We use the emulation environment Emsim provided as part of Emstar [9], a system for developing and deploying complex wireless network applications. In Emsim, the source code and configuration files are identical to the ones used in a real deployment. Only the radio channel is simulated, the rest of the protocol stack (like the MAC and routing layers) is identical to a real system. Nodes are simulated as independent processes, which have their own clocks, state and packet buffers. The only way they can talk to each other is by sending packets; there is no omniscient control channel for sharing state with each other.

(b) Setup: We have 100 nodes placed in a $800m \times 800m$ area. Senders and receivers are picked randomly. Each flow sends UDP traffic. The senders are always backlogged and thus send as fast as the MAC permits. Our simulations use the 802.11 MAC, which has a default bit rate of 11 Mb/s. The radio channel is simulated to have noise from a normal distribution. The radio power is fixed at 200mW. The channel also has a contention model, hence the 802.11 MAC responds to channel sensing and collisions by backing off. We use a simplified version of geographic routing. Every node knows the co-ordinates of all nodes in the system. Nodes that are less than 50m apart are considered neighbors. The routing protocol picks the neighbor closest to the destination as the next hop.

(c) Metric: We use the sum of the end-to-end throughput of all flows in the network as our evaluation metric. To ensure a fair and conservative comparison, we subtract the bandwidth consumed by reception reports and other metadata from our throughput computation.

(d) Compared Schemes: Our experiments focus on understanding the performance improvement gained from adding more features to the network coding scheme. All coding schemes are built using pseudo-broadcast. We compare the following approaches:

- *No Coding:* This is the current approach which transmits clear packets using 802.11 unicast.
- *2-way Coding:* In this approach, a wireless router XORs any two packets p_1 and p_2 if they are traversing the same pair of neighbors in opposite directions (i.e., p_1 is received from i and should be forwarded to j, and p_2 is received from j and need to be forwarded to i). This is a special case of COPE, which limits the router to XOR-ing a pair of packets and turns off opportunistic listening.
- *COPE With No-Guesses:* This approach is similar to COPE but the router relies solely on the reception reports to discover which packets have been heard by which neighbors.
- *COPE:* This scheme combines opportunistic listening with opportunistic coding, as described in §3.2.
- *COPE With Opportunistic Routing* This approach, described in §3.3, combines opportunistic routing with opportunistic listening and coding.

6.2 Does Network Coding Help in Practice?

In §3.2, we have advocated a simple approach to network coding in wireless environments: each node relies on its local information to detect coding opportunities, and when possible XORs the appropriate packets. However, it is unclear how often such opportunities arise in realistic settings, and whether they can be detected using only local information. Our experiments aim to gauge the expected increase in throughput brought in by COPE.

We evaluate the performance of COPE as the level of congestion in the network increases. Our experiments follow the scenario in §6.1-b. We vary two parameters: the number of flows in the network and the path length.

(a) Throughput as function of the number of flows: Flows are picked randomly by connecting two nodes together. The path length is therefore random and we vary the number of flows in the network. Fig. 5 plots the network throughput as a function of the number of flows in the experiment, for the various schemes in §6.1-d.

The results show that coding can dramatically improve the throughput of congested wireless networks. When the number of flows is small, the coding opportunities are few, and all coding schemes perform similarly to no coding. But, as the number of flows increases, both the coding opportunities and the network congestion increase. In such environments, without coding, the performance deteriorates quickly because of the high level of contention in the network and consequent packet loss due to collisions. In contrast, coding reduces the number of transmissions for the same amount of data, resulting in lower congestion and consequently better performance.

The figure also illustrates the throughput increase gained from improving the coding scheme. 2-way coding consistently performs better than no coding but misses many coding opportunities. Due to opportunistic listening and the ability to encode more than two packets, COPE With No-Guesses achieves a better performance than 2-way Coding. Adding the reception guesses substantially improves performance. As congestion increases, more reception reports get lost, increasing the need for guessing.

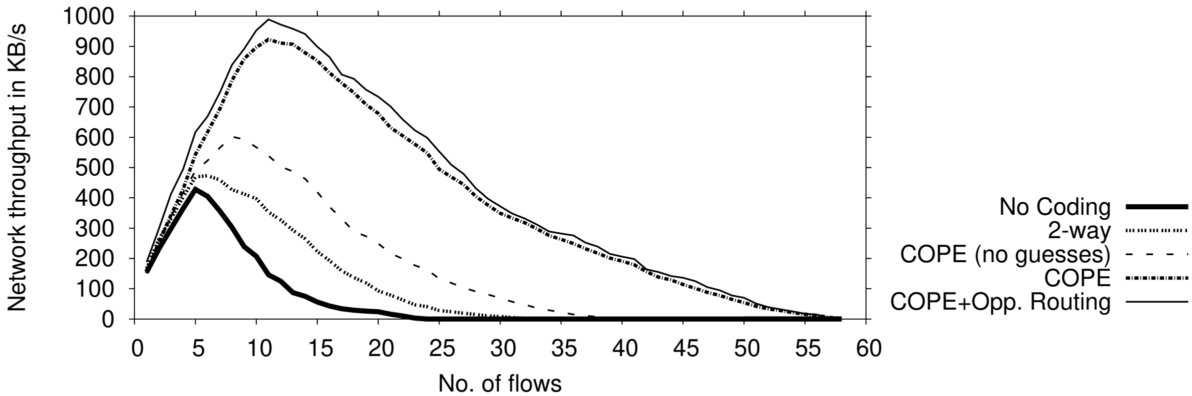


Figure 5: COPE outperforms the current approach which does not use coding. It provides dramatic increase in network throughput particularly as the number of flows increases. Also, each added feature has resulted in an additional improvement in the throughput.

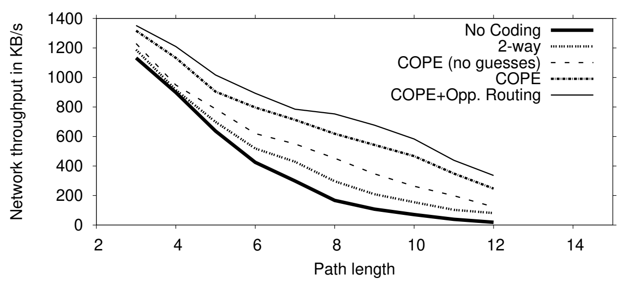


Figure 6: Throughput as a function of the path length. Again coding always outperforms no coding.

This also may indicate the need for a more reliable way of delivering reception reports, potentially by echoing the same report in multiple packets. Finally, allowing the routing to use the information in the reception report to opportunistically skip some hops provides a small boost in performance.

(b) Throughput as function of path length: Next, we evaluate the coding performance as a function of the path length. We fix the number of flows at 10. All flows have the same path length, which we vary from one experiment to another. Fig. 6 plots the throughput obtained in KB/s as a function of the path length. We observe the same trends as before: coding performs substantially better than no coding, and each extra feature added to the coding scheme increases the throughput. But as the path length increases, throughput of all the schemes drops. No-coding and 2-way Coding drop sharply since an increasing number of packets are getting lost before being delivered to the destination. COPE suffers from the same problem, but the drop in throughput is more gradual.

7 Implementation

We implement COPE in an 802.11b multi-hop wireless testbed. Each node in the testbed is a PC equipped with an 802.11b wireless card connected to an omni-directional antenna. The cards are based on the Intersil Prism 2.5 802.11b chipset. They transmit at a power level of 200 milliwatts with RTS/CTS disabled, and operate in 802.11 ad hoc mode. The nodes are placed inside our lab within a few meters of each other, and share the same radio channel. They run the Roofnet software [1], including the Click router [17] and the Srcr routing protocol [6].

Our implementation runs as a user space daemon on Linux. The daemon sends and receives raw Ethernet frames from the wireless device using a libpcap-like interface. The implementation exports a network interface to the user that can be treated like any other network device (for e.g., `eth0`). Applications interact with the daemon as they would with a standard network device provided by the Linux kernel. No modifications to the applications are therefore necessary. The implementation is protocol agnostic and can be used by a wide variety of transport protocols including both TCP and UDP.

8 Implementation Results

We present preliminary implementation results for the simple scenario in Fig. 3, where Alice and Bob send to each other via a relay node. We pick the three nodes randomly from those in the testbed. We setup static routes to ensure the two senders transmit via the relay. Each experiment consists of the senders transmitting UDP packets to each other for 1 minute using standard 802.11 with no coding. This is followed by an idle period of 15 seconds. Then again, the two senders exchange UDP traffic for two min-

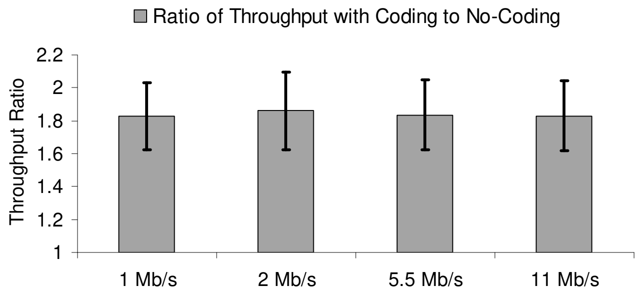


Figure 7: Implementation results for the scenario in Fig. 3. The figure plots the throughput ratio with coding to no-coding, as a function of 802.11 bit rate. It shows that coding almost doubles the throughput.

utes but this time with COPE. Throughput is measured in a 30 second interval once the transmissions have reached steady state. Fig 7 plots the ratios of the throughput obtained with COPE to throughput with no coding.[4] The figure shows both the average and the standard deviation of 10 experiments each with and without coding, for various 802.11 bit rates. COPE nearly doubles the network throughput. Further, different sending rates do not seem to affect this ratio, indicating that the coding gain is independent of the underlying link layer sending rate.

9 Related Work

Recent years have seen a substantial advancement in the theory of network coding. Ahlswede et al started the field with their pioneering paper [2], which shows that having intermediate nodes in the network mix information from different flows increases the throughput and allows the communication to achieve the broadcast capacity. This was soon followed by the work of Li et al, who showed that, for the multicast case, linear codes are sufficient to achieve the maximum flow bounds [19]. Koetter and Médard [16] presented polynomial time algorithms for encoding and decoding, and Ho et al extended these results to random codes [13]. A few papers extended some of these results to wired and wireless networks with lossy links [10,20,21]. Also, recent work studied network coding in the wireless environment [8,23]. In particular, Lun et al studied network coding in the presence of omni-directional antennas and showed that the problem of minimizing the communication cost can be addressed as a linear program and solved in a distributed manner [22].

In contrast to the above work which focuses on multicast, our approach addresses unicast. There is little prior work on network coding for unicast communication [12], and most results are negative [26]. It is known that for a single unicast session, the coding gain is not greater than one. The case for multiple unicast sessions is largely unknown territory, with specific examples present where network coding results in gains [27]. Our work does not focus on optimality; rather it examines whether a simple approach to network coding can improve over current approaches, and shows that network coding increases the throughput of a wireless network, even when all traffic is unicast.

The closest work to our research is that presented by Wu et al in [24]. They focus on duplex flows–i.e., two nodes sending packets to each other in opposite directions through a number of wireless relays. Packets from these two flows traverse exactly opposite paths, and thus can be XOR-ed together and broadcast to their next-hops. In contrast, to ours, the approach in [24] is limited to flows that traverse the same path in opposite directions, does not code more than a pair of packets, and does not benefit from opportunistic listening.

Finally, opportunistic routing was introduced in [3], and is related to ideas in the field of cooperative diversity [18]. We connect this idea with network coding by using the same reception reports for both identifying which packets may be encoded together and opportunistically skipping unnecessary path segments.

10 Concluding Remarks

This paper presents the first implementation of network coding in the wireless environment. We have learnt many important lessons while working on this implementation. Particularly, it pays to be opportunistic. Instead of aspiring to achieve broadcast capacity, we made the nodes use local information to detect coding opportunities and exploit them. Though this may not yield optimal throughput, it allowed a practical integration of network coding into the current stack. The result was a large increase in the network throughput; Depending on the degree of congestion and the number of distinct flows, the throughput with coding may be an order of magnitude higher than without coding. Our approach provides a network coding protocol for multiple wireless unicast flows. It is online, distributed, and deployable.

Our future work extends COPE in a few directions. It includes adding link layer acknowledgment to COPE to mask wireless errors from TCP, conducting extensive experiments over a large 40-nodes wireless mesh network, and running actual applications on top of COPE.

[4] We plot the ratio of the throughputs instead of the actual values because they vary a lot from one experiment to another depending on the nodes' location.

11 Acknowledgments

We thank Srikanth Kandula for helping with the code; Robert Morris, Desmond Lun, Nick Harvey, Allen Miu, and Kyle Jamieson for their comments on the paper. Katti and Katabi are supported by the NSF award ANI-0335256. The opinions and findings in this paper are those of the authors and do not necessarily reflect the views of NSF.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements from an 802.11b Mesh Network. In *ACM SIGCOMM, 2004*.

[2] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network Information Flow. In *IEEE Transactions on Information Theory*, 2000.

[3] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. *ACM SIGCOMM, 2005*.

[4] P. Chou, Y. Wu, and K. Jain. Practical Network Coding. In *51st Allerton Conf. Communication, Control and Computing, Oct. 2003*.

[5] Christos Gkantsidis and Pablo Rodriguez. Network Coding for Large Scale Content Distribution. In *IEEE INFOCOM, 2005*.

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

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM MobiCom '03*, San Diego, California, September 2003.

[8] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Medard, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In *IWWAN*, 2005.

[9] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *ACM Sensys, 2004*.

[10] R. Gowaikar, A. F. Dana, R. Palanki, B. Hassibi, and M. Effros. On the capacity of wireless erasure networks. In *IEEE International Symposium on Information Theory (ISIT 2004)*, 2004.

[11] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless lans. In *ACM SIGCOMM, 2005*.

[12] T. Ho and R. Koetter. Online incremental network coding for multiple unicasts. In *DIMACS Working Group on Network Coding*, 2005.

[13] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *ISIT, 2003*.

[14] T. Ho, B. Leong, Medard, R. Koetter, Y. Chang, and M. Effros. The Utility of Network Coding in Dynamic Environments. In *IWWAN*, 2004.

[15] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transactions on Information Theory*, 2003.

[16] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking, 2003*.

[17] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems, August 2000*.

[18] J. N. Laneman, D. N. C. Tse, and G. W. Wornell. Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *IEEE Trans. Inform. Theory, Dec. 2004*.

[19] S. R. Li, R. W. Yeung, and N. Cai. Linear network coding. In *IEEE Transactions on Information Theory*, 2003.

[20] D. S. Lun, M. Medard, and R. Koetter. Efficient operation of wireless packet networks using network coding. In *International Workshop on Convergent Technologies (IWCT)*, 2005.

[21] D. S. Lun, M. Medard, R. Koetter, and M. Effros. Further results on coding for reliable communication over packet networks. In *IEEE International Symposium on Information Theory (ISIT 05)*, 2005.

[22] D. S. Lun, N. Ratnakar, R. Koetter, M. Mdard, E. Ahmed, and H. Lee. Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding. In *IEEE INFOCOM, 2005*.

[23] A. Ramamoorthy, J. Shi, and R. Wesel. On the capacity of network coding for wireless networks. In *41st Annual Allerton Conference on Communication Control and Computing*, Oct. 2003.

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

[25] Y. Wu and S.-Y. Kung. Distributed utility maximization for network coding based multicasting: a shorted path approach. submitted to IEEE INFOCOM 2006.

[26] Z. Li and B. Li. Network Coding in Undirected Networks. In *CISS 04*, 2004.

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