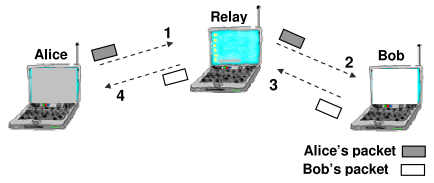
XORs in the Air: Practical Wireless Network Coding

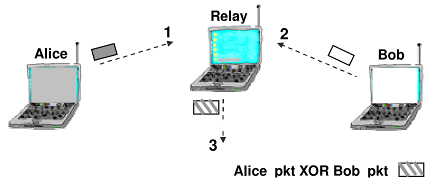
Sachin Katti Hariharan Rahul Wenjun Hu Dina Katabi Muriel Médard Jon Crowcroft

***Abstract*— This paper proposes COPE, a new architecture for wireless mesh networks. In addition to forwarding packets, routers mix (i.e., code) packets from different sources to increase the information content of each transmission. We show that intelligently mixing packets increases network throughput. Our design is rooted in the theory of network coding. Prior work on network coding is mainly theoretical and focuses on multicast traffic. This paper aims to bridge theory with practice; it addresses the common case of unicast traffic, dynamic and potentially bursty flows, and practical issues facing the integration of network coding in the current network stack. We evaluate our design on a 20-node wireless network, and discuss the results of the first testbed deployment of wireless network coding. The results show that using COPE at the forwarding layer, without modifying routing and higher layers, increases network throughput. The gains vary from a few percent to several folds depending on the traffic pattern, congestion level, and transport protocol.**

***Index Terms*— Network Coding, Wireless Networks, Algorithms, Design, Performance, Theory**



(a) Current Approach



(b) COPE

Fig. 1—A simple example of how COPE increases the throughput. It allows Alice and Bob to exchange a pair of packets using 3 transmissions instead of 4 (numbers on arrows show the order of transmission).

I. Introduction

Wireless networks suffer from low throughput and do not scale to dense deployments [1]. To address this problem, we present COPE, a new forwarding architecture that substantially improves the throughput of stationary wireless mesh networks. COPE inserts a coding shim between the IP and MAC layers. It identifies coding opportunities and benefits from them by forwarding multiple packets in a single transmission.

COPE's design is inspired by the theory of network coding. Consider the scenario in Fig. 1, where Alice and Bob want to exchange a pair of packets via a router. In current approaches, Alice sends her packet to the router, which forwards it to Bob, and Bob sends his packet to the router, which forwards it to Alice. This process requires 4 transmissions. Now consider a network coding approach. Alice and Bob send their respective packets to the router, which XORs the two packets and broadcasts the XOR-ed version. Alice and Bob can obtain each other's packet by XOR-ing again with their own packet. This process takes 3 transmissions instead of 4. Saved transmissions can be used to send new data, increasing the wireless throughput.

The main challenge in designing COPE is to extend the idea beyond duplex flows. Few applications send data in both directions, limiting the practical benefits of the scenario in Fig. 1. To achieve large throughput gains, we need to generalize the idea to any topology and traffic pattern. To do so, we exploit the broadcast nature of the wireless medium. We make the nodes snoop on all transmissions and store the overheard packets for a short period. As a result, a router can XOR two packets and deliver them to two different neighbors in a single transmission whenever it knows that each of the two neighbors has overheard the packet destined to the other. Snooping not only extends the benefits of coding beyond duplex flows, but also enables us to code more than pairs of packets, producing a larger throughput increase than that in Fig. 1.

Our work advocates an alternative architecture for wireless mesh networks that significantly improves their throughput. It is based on the following two key principles.

- *Employ network coding.* Current routers forward packets from one link to another. COPE, however, shows that there are practical benefits for allowing the routers to intelligently mix the content of the packets before forwarding them.
- *Embrace the broadcast nature of the wireless channel.* Network designers typically abstract the wireless channel as a point-to-point link, then adapt forwarding and routing techniques designed for wired networks for wireless. In contrast, COPE exploits the broadcast property of radios instead of hiding it under an artificial abstraction.

We summarize the contributions of this work as follows.
(1) COPE provides a general scheme for inter-session wireless network coding. It applies to any topology and an arbitrary number of bursty flows whose duration is not known a priori, and that arrive and leave dynamically. In contrast, prior work on inter-session network coding either focuses on duplex

Sachin Katti, Hariharan Rahul, Dina Katabi, and Muriel Médard are affiliated with the Department of EECS, MIT. Wenjun Hu and Jon Crowcroft are affiliated with the University of Cambridge.

flows [2] or assumes known flow patterns with steady rates and ideal scheduling [3].
(2) COPE is the first integration of network coding into the current network stack, presenting a system architecture that works seemlessly with higher layer protocols and supports both TCP and UDP flows. We also implement COPE in the Linux kernel and evalute its performance in a wireless testbed.
(3) The paper presents the results of the first deployment of network coding in a wireless network. It studies the performance of COPE in a 20-node wireless testbed using state-of-the-art routing protocols, and reveals its interactions with the wireless channel and higher layer protocols. Our findings can be summarized as follows:

- Network coding, used purely as an enhancement of the forwarding layer, while keeping routing and other higher layers unmodified, substantially improves wireless throughput.
- When the wireless medium is congested and the traffic consists of many random UDP flows, COPE increases the throughput of our testbed by 3-4*x*.
- If the traffic does not exercise congestion control (e.g., UDP), COPE's throughput improvement may substantially exceed the expected theoretical coding gain. This additional gain occurs because coding makes a router's queue smaller, reducing the probability that a congested downstream router will drop packets that have consumed network resources.
- For a mesh network connected to the Internet via an access point, the throughput improvement observed with COPE varies depending on the ratio between total download and upload traffic traversing the access point, and ranges from 5% to 70%.
- Hidden terminals create a high collision rate that cannot be masked even with the maximum number of 802.11 retransmissions. In these environments, TCP does not send enough to utilize the medium, and thus does not create coding opportunities. With no hidden terminals, TCP's throughput increases by an average of 38% in our testbed.

Term	Definition
Native Packet	A non-encoded packet
Encoded or XOR-ed Packet	A packet that is the XOR of multiple native packets
Nexthops of an Encoded Packet	The set of nexthops for the native packets XOR-ed to generate the encoded packet
Packet Id	A 32-bit hash of the packet's IP source address and IP sequence number
Output Queue	A FIFO queue at each node, where it keeps the packets it needs to forward
Packet Pool	A buffer where a node stores all packets heard in the past T seconds
Coding Gain	The ratio of the number of transmissions required by the current non-coding approach, to the number of transmissions used by COPE to deliver the same set of packets.
Coding+MAC Gain	The expected throughput gain with COPE when an 802.11 MAC is used, and all nodes are backlogged.

TABLE I—**Definitions of terms used in this paper.**

II. Background and Related Work

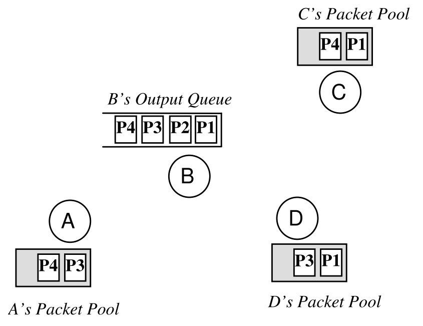
Research on network coding can be divided into two categories: intra-session, where coding is restricted to packets belonging to the same session or flow, and inter-session, where coding is allowed among packets belonging to possibly different sessions or flows. Intra-session network coding has been extensively studied, beginning with the pioneering paper by Ahlswede et al. [4], who show that having the routers mix information in different messages allows the communication to achieve multicast capacity. Li et al. show that for multicast traffic linear codes are sufficient to achieve the maximum capacity bounds [5]. Koetter and Médard [6] present polynomial time algorithms for encoding and decoding, and Ho et al. extend these results to random codes [7]. Further, some work studies intra-session wireless network coding [8]–[16]. In particular, Lun et al. study intra-session network coding and show that the problem of minimizing the communication cost can be formulated as a linear program and solved in a distributed manner [17].

The above work is either theoretical or simulation based and assumes a combination of multicast traffic, optimal scheduling, and known flow patterns with steady rates. In a recent paper, we present a low-complexity algorithm for intra-session network coding and demonstrate via implementation and testbed experiment that intra-session network coding yields practical benefits to both unicast and multicast flows [16].

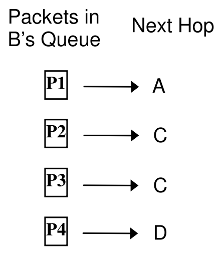
Inter-session network coding, to which COPE belongs, is known to be difficult. It is known that linear codes are insufficient for optimal inter-session network coding [18], and even if we limit ourselves to linear codes, determining how to perform the coding is NP-hard [6]. Hence, recent work has focused on developing heuristics that provide significant throughput gains [2], [3], [19], [20]. Wu et al. [2] describe a physical piggybacking scheme for inter-session network coding in line networks i.e., duplex flows like those in Fig. 1. COPE generalizes that scheme to arbitrary networks. Queue stability and MAC issues for line networks have been explored in [21], [22]. This paper focuses on inter-session network coding but it develops a practical heuristic that bridges the gap between the theory of network coding and practical network design and provides an operational protocol for general unicast traffic.

Related work has also built on our conference publication of COPE [23]. Wu et al. [24] have presented algorithms for mixing packets optimally, Rayanchu et al. [25] have explored routing and MAC algorithms for COPE, and Liu et al. [26] have analyzed the theoretical throughput benefits obtained with COPE style of coding. The technique has also been recently extended to the physical layer [27]–[30].

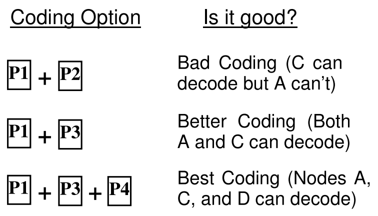
Finally, a rich body of systems research has tackled the problem of improving the throughput of wireless networks. The proposed solutions range from designing better routing metrics [31]–[33] to tweaking the TCP protocol [34], and include improved routing and MAC protocols [35], [36]. Our work builds on these foundations but adopts a fundamentally different approach; it explores the utility of network coding in improving the throughput of wireless networks.



(a) B can code packets it wants to send



(b) Nexthops of packets in B's queue


Coding Option
Is it good?
P1 + P2
Bad Coding (C can decode but A can't)
P1 + P3
Better Coding (Both A and C can decode)
P1 + P3 + P4
Best Coding (Nodes A, C, and D can decode)

(c) Possible coding options

Fig. 2—**Example of Opportunistic Coding; Node B has 4 packets in its queue, whose nexthops are listed in (b). Each neighbor of B has stored some packets as depicted in (a). Node B can make a number of coding decisions (as shown in (c)), but should select the last one because it maximizes the number of packets delivered in a single transmission.**

III. COPE OVERVIEW

We introduce COPE, a new forwarding architecture for wireless mesh networks. It inserts a coding layer between the IP and MAC layers, which detects coding opportunities and exploits them to forward multiple packets in a single transmission. COPE assumes that there is an underlying routing protocol which picks paths between nodes. Before delving into details, we refer the reader to Table I, which defines the terms used in the rest of the paper.

COPE incorporates three main techniques:

(a) Opportunistic Listening: Wireless is a broadcast medium, creating many opportunities for nodes to overhear packets when they are equipped with omni-directional antennae. COPE sets the nodes in promiscuous mode, makes them snoop on all communications over the wireless medium and store the overheard packets for a limited period T (default $T = 0.5s$).

In addition, each node broadcasts *reception reports* to tell its neighbors which packets it has stored. 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.

(b) Opportunistic Coding: The key question is what packets to code together to maximize throughput. A node may have multiple options, but it should aim to *maximize the number of native packets delivered in a single transmission, while ensuring that each intended nexthop has enough information to decode its native packet.*

The above is best illustrated with an example. In Fig. 2(a), node B has 4 packets in its output queue p_1, p_2, p_3, and p_4. Its neighbors have overheard some of these packets. The table in Fig 2(b) shows the nexthop of each packet in B's queue. When the MAC permits B to transmit, B takes packet p_1 from the head of the queue. Assuming that B knows which packets each neighbor has, it has a few coding options as shown in Fig. 2(c). It could send $p_1 \oplus p_2$. Since node C has p_1 in store, it could XOR p_1 with $p_1 \oplus p_2$ to obtain the native packet sent to it, i.e., p_2. However, node A does not have p_2, and so cannot decode the XOR-ed packet. Thus, sending $p_1 \oplus p_2$ would be a bad coding decision for B, because only one neighbor can benefit from this transmission. The second option in Fig. 2(c) shows a better coding decision for B. Sending $p_1 \oplus p_3$ would allow both neighbors C and A to decode and obtain their intended packets from a single transmission. Yet the best coding decision for B would be to send $p_1 \oplus p_3 \oplus p_4$, which would allow all three neighbors to receive their respective packets all at once.

The above example emphasizes an important coding issue. Packets from multiple unicast flows may get encoded together at some intermediate hop. But their paths may diverge at the nexthop, at which point they need to be decoded. If not, unneeded data will be forwarded to areas where there is no interested receiver, wasting much capacity. The coding algorithm should ensure that all nexthops of an encoded packet can decode their corresponding native packets. This can be achieved using the following simple rule:

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

This rule ensures that each nexthop can decode the XOR-ed version to extract its native packet. Whenever a node has a chance to transmit a packet, it chooses the largest n that satisfies the above rule to maximize the benefit of coding.

(c) Learning Neighbor State: But how does a node know what packets its neighbors have? As explained earlier, each node announces to its neighbors the packets it stores in reception reports. However, at times of severe congestion, reception reports may get lost in collisions, while at times of light traffic, they may arrive too late, after the node has already made a suboptimal coding decision. Therefore, a node cannot rely solely on reception reports, and may need to guess whether a neighbor has a particular packet.

To guess intelligently, we leverage the routing computation. Wireless routing protocols compute the delivery probability between every pair of nodes and use it to identify good paths. For e.g., the ETX metric [31] periodically computes the delivery probabilities and assigns each link a weight equal to *1/(delivery probability)*. These weights are broadcast to all nodes in the network and used by a link-state routing protocol to compute shortest paths. We leverage these probabilities for guessing. In the absence of deterministic information, COPE estimates the probability that a particular neighbor has a packet as the delivery probability of the link between the packet's previous hop and the neighbor.

Occasionally, a node may make an incorrect guess, which causes the coded packet to be undecodable at some nexthop. In this case, the relevant native packet is retransmitted, potentially encoded with a new set of native packets.

IV. Understanding COPE's Gains

How beneficial is COPE? Its throughput improvement depends on the existence of coding opportunities, which themselves depend on the traffic patterns. This section provides some insight into the expected throughput increase and the factors affecting it.

A. Coding Gain

We defined the *coding gain* as the ratio of the number of transmissions required by the current non-coding approach, to the minimum number of transmissions used by COPE to deliver the same set of packets. By definition, this number is greater than or equal to 1.

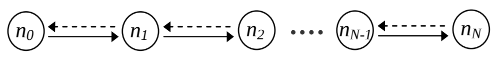
In the Alice-and-Bob experiment, as described in §I, COPE reduces the number of transmissions from 4 to 3, thus producing a coding gain of $\frac{4}{3} = 1.33$.

But what is the maximum achievable coding gain, i.e., what is the theoretical capacity of a wireless network that employs COPE? The capacity of general network coding for unicast traffic is still an open question for arbitrary graphs [3], [37]. However, we analyze certain basic topologies that reveal some of the factors affecting COPE's coding gain. Our analysis assumes identical nodes, omni-directional radios, perfect hearing within some radius, and the signal is not heard at all outside this radius, and if a pair of nodes can hear each other the routing will pick the direct link. Additionally, we assume that the flows are infinite and we only consider the steady state.

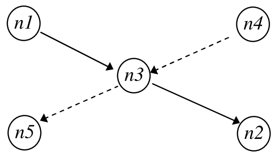
Lemma 4.1: In the absence of opportunistic listening, COPE's maximum coding gain is 2, and it is achievable.
We prove the lemma by showing that the coding gain of the chain in Fig. 3(a) tends to 2 as the number of intermediate nodes increases. The complete proof is in Appendix A.

While we do not know the maximum gain for COPE with opportunistic listening, there do exist topologies where opportunistic listening adds to the power of COPE. For example, consider the "X"-topology shown in Fig. 3(b). This is the analogy of the Alice-and-Bob topology, but the two flows travel along link-disjoint paths. COPE without opportunistic listening cannot achieve any gains on this topology. But with opportunistic listening and guessing, the middle node can combine packets traversing in opposite directions, for a coding gain of $\frac{4}{3} = 1.33$. This result is important, because in a real wireless network, there might be only a small number of flows traversing the reverse path of each other à la Alice-and-Bob, but one would expect many flows to intersect at a relay, and thus can be coded together using opportunistic listening and guessing.

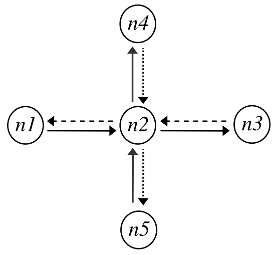
The "X" and Alice-and-Bob examples can be combined to further improve the benefits of coding, as in the cross topology of Fig. 3(c). Without coding, 8 transmissions are necessary for each flow to send one packet to its destination. However,



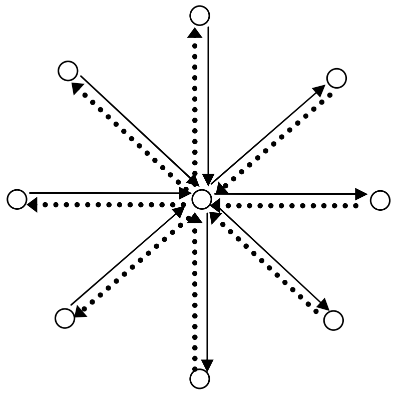
(a) Chain topology; 2 flows in reverse directions.



(b) "X" topology 2 flows intersecting at n_2.



(c) Cross topology 4 flows intersecting at n_2



(d) Wheel topology; many flows intersecting at the center node.

Fig. 3—Simple topologies to understand COPE's Coding and Coding+MAC Gains.

assuming perfect overhearing (n_4 and n_5 can overhear n_1 and n_3, and vice versa), n_2 can XOR 4 packets in each transmission, thus reducing the number of transmissions from 8 to 5, producing a coding gain of $\frac{8}{5} = 1.6$.

We observe that while this section has focused on theoretical bounds, the gains in practice tend to be lower due to the availability of coding opportunities, packet header overheads, medium losses, etc. However, it is important to note that COPE increases the actual information rate of the medium far above the bit rate, and hence its benefits are sustained even when the medium is fully utilized. This contrasts with other approaches to improving wireless throughput, such as opportunistic routing [35], which utilize the medium better when it is not fully congested, but do not increase its capacity.

B. Coding+MAC Gain

When we ran experiments with COPE, we were surprised to see that the throughput improvement sometimes greatly exceeded the coding gain for the corresponding topology. It turns out that the interaction between coding and the MAC produces a beneficial side effect that we call the Coding+MAC gain.

The Coding+MAC gain is best explained using the Alice-and-Bob scenario. Because it tries to be fair, the MAC divides the bandwidth equally between the 3 contending nodes: Alice,

Topology	Coding Gain	Coding+MAC Gain
Alice-and-Bob	1.33	2
"X"	1.33	2
Cross	1.6	4
Infinite Chain	2	2
Infinite Wheel	2	∞

TABLE II—Theoretical gains for a few basic topologies.

Bob, and the router. Without coding, however, the router needs to transmit twice as many packets as Alice or Bob. The mismatch between the traffic the router receives from the edge nodes and its MAC-allocated draining rate makes the router a bottleneck; half the packets transmitted by the edge nodes are dropped at the router's queue. COPE allows the bottleneck router to XOR pairs of packets and drain them twice as fast, doubling the throughput of this network. Thus, the Coding+MAC gain of the Alice-and-Bob topology is 2.

The Coding+MAC gain assumes all nodes continuously have some traffic to send (i.e., backlogged), but are limited by their MAC-allocated bandwidth. It computes the throughput gain with COPE under such conditions. For topologies with a single bottleneck, like the Alice-and-Bob's, the Coding+MAC gain is the ratio of the bottleneck's draining rate with COPE to its draining rate without COPE.

Similarly, for the "X" and cross topologies, the Coding+MAC gain is higher than the coding gain. For the "X", the Coding+MAC gain is 2 since the bottleneck node is able to drain twice as many packets, given its MAC allocated rate. For the cross topology, the Coding+MAC gain is even higher at 4. The bottleneck is able to send 4 packets out in each transmission, hence it is able to drain four times as many packets compared to no coding. This begs the question: what is the maximum Coding+MAC gain? The maximum possible Coding+MAC gains with and without opportunistic listening are properties of the topology and the flows that exist in a network. Here we prove some upper bounds on Coding+MAC gains.

Lemma 4.2: In the absence of opportunistic listening, COPE's maximum Coding+MAC gain is 2, and it is achievable.

The proof is in Appendix B.

Lemma 4.3: In the presence of opportunistic listening, COPE's maximum Coding+MAC gain is unbounded.

The proof, detailed in Appendix C, uses the wheel topology in Fig. 3(d). Assuming N edge nodes, with COPE the bottleneck node, in the center of the wheel, XORs N packets together, and consequently drains its queue N times faster than without COPE. As the number of edge nodes increases, i.e., $N \to \infty$, the gain becomes infinite. While the previous example is clearly artificial, it does illustrate the potential of COPE with opportunistic listening to produce a several-fold improvement in throughput, as in §VII.

Table II lists the gains for a few basic topologies.

V. Making It Work

To integrate COPE effectively within the current network stack, we need to address some important system issues.

A. Packet Coding Algorithm

To build the coding scheme, we have to make a few design decisions. First, we design our coding scheme around the principle of *never delaying packets*. When the wireless channel is available, the node takes the packet at the head of its output queue, checks which other packets in the queue may be encoded with this packet, XORs those packets together, and broadcasts the XOR-ed version. If there are no encoding opportunities, our node does not wait for the arrival of a matching codable packet. COPE therefore lets the node opportunistically overload each transmission with additional information when possible, but does not wait for additional codable packets to arrive.

Second, COPE *gives preference to XOR-ing packets of similar lengths*, because XOR-ing small packets with larger ones reduces bandwidth savings. Empirical studies show that the packet-size distribution in the Internet is bimodal with peaks at 40 and 1500 bytes [38]. We can therefore limit the overhead of searching for packets with the right sizes by distinguishing between small and large packets. We might still have to XOR packets of different sizes. In this case, the shorter packets are padded with zeroes. The receiving node can easily remove the padding by checking the packet-size field in the IP header of each native packet.

Third, notice that COPE will *never code together packets headed to the same nexthop or packets generated by the coding node*, since the nexthop will not be able to decode them. Hence,while coding, we only need to consider non-self generated packets headed to different nexthops. COPE therefore maintains two virtual queues per neighbor; one for small packets and another for large packets (The default setting uses a threshold of 100 bytes). When a new packet is added to the output queue, an entry is added to the appropriate virtual queue based on the packet's nexthop and size.

Searching for appropriate packets to code is efficient due to the maintenance of virtual queues. When making coding decisions, COPE first dequeues the packet at the head of the FIFO output queue, and determines if it is a small or a large packet. Depending on the size, it looks at the appropriate virtual queues. For example, if the packet dequeued is a small packet, COPE first looks at the virtual queues for small packets. COPE looks only at the heads of the virtual queues to limit packet reordering. After exhausting the virtual queues of a particular size, the algorithm then looks at the heads of virtual queues for packets of the other size. Thus for finding appropriate packets to code COPE has to look at $2M$ packets in the worst case, where M is the number of neighbors of a node.

Another concern is *packet reordering*. We would like to limit reordering packets from the same flow because TCP mistakes it as a congestion signal. Thus, we always consider packets according to their order in the output queue. Still, reordering may occur because we prefer to code packets of the same size. In practice, this reordering is quite limited because most data packets in a TCP flow are large enough to be queued in the large-packet queue, and thus be considered in order. We will see in §V-D, however, that reordering might arise from other reasons, particularly the need to retransmit a packet that has been lost due to a mistake in guessing what a neighbor can decode. Thus, we choose to deal with any reordering that might happen inside the network at the receiver. COPE has a module that puts TCP packets in order before delivering them to the transport layer as explained in §V-E.

Finally, we want to ensure that each neighbor to whom a packet is headed has a high probability of decoding its native packet. Thus, for each packet in its output queue, our relay node estimates the probability that each of its neighbors has already heard the packet. Sometimes the node can be certain about the answer, for example, when the neighbor is the previous hop of the packet, or when the reception reports from the neighbor state so. When neither of the above is true, the node leverages the delivery probabilities computed by the routing protocol; it estimates the probability the neighbor has the packet as the delivery probability between the packet's previous hop and that neighbor. The node then uses this estimate to ensure that encoded packets are decodable by all of their nexthops with high probability.

In particular, suppose the node encodes n packets together. Let the probability that a nexthop has heard packet i be P_i. Then, the probability, P_D, that it can decode its native packet is equal to the probability that it has heard all of the $n-1$ native packets XOR-ed with its own, i.e.,

$$P_D = P_1 \times P_2 \times \ldots \times P_{n-1}.$$

Consider an intermediate step while searching for coding candidates. We have already decided to XOR $n-1$ packets together, and are considering XOR-ing the n^{th} packet with them. The coding algorithm now checks that, for each of the n nexthops, the decoding probability P_D, after XOR-ing the n^{th} packet with the rest stays greater than a threshold G (the default value $G = 0.8$). If the above conditions are met, each nexthop can decode its packet with at least probability G. Finally, we note that for fairness we iterate over the set of neighbors according to a random permutation.

Formally, each node maintains the following data structures.

- Each node has a FIFO queue of packets to be forwarded, which we call *the output queue*.
- For each neighbor, the node maintains two *per-neighbor virtual queues*, one for small packets (e.g., smaller than 100 bytes), and the other for large packets. The virtual queues for a neighbor A contain pointers to the packets in the output queue whose nexthop is A.
- Additionally, the node keeps a hash table, *packet info*, that is keyed on packet-id. For each packet in the output queue, the table indicates the probability of each neighbor having that packet.

Whenever the MAC signals a sending opportunity, the node executes the procedure illustrated in Alg. 1. The greedy strategy depicted in this algorithm has a computational complexity that is linear in the number of active neighbors of a node.

1 Coding Procedure

```
Pick packet $p$ at the head of the output queue.
Natives = $\{p\}$
Nexthops = $\{nexthop(p)\}$
if $size(p) > 100$ bytes then
    which_queue = 1
else
    which_queue = 0
end if
for Neighbor $i = 1$ to $M$ do
    Pick packet $p_i$, the head of virtual queue $Q(i, which\_queue)$
    if $\forall n \in$ Nexthops $\cup\{i\}$, Pr[$n$ can decode $p \oplus p_i$] $\geq G$ then
        $p = p \oplus p_i$
        Natives = Natives $\cup\{p_i\}$
        Nexthops = Nexthops $\cup\{i\}$
    end if
end for
which_queue = !which_queue
for Neighbor $i = 1$ to $M$ do
    Pick packet $p_i$, the head of virtual queue $Q(i, which\_queue)$
    if $\forall n \in$ Nexthops $\cup\{i\}$, Pr[$n$ can decode $p \oplus p_i$] $\geq G$ then
        $p = p \oplus p_i$
        Natives = Natives $\cup\{p_i\}$
        Nexthops = Nexthops $\cup\{i\}$
    end if
end for
return $p$
```

B. Packet Decoding

Packet decoding is simple. Each node maintains a *Packet Pool*, in which it keeps a copy of each native packet it has received or sent out. The packets are stored in a hash table keyed on packet id (see Table I), and the table is garbage collected every few seconds. When a node receives an encoded packet consisting of n native packets, the node goes through the ids of the native packets one by one, and retrieves the corresponding packet from its packet pool if possible. Ultimately, it XORs the $n-1$ packets with the received encoded packet to retrieve the native packet meant for it.

C. Pseudo-broadcast

The 802.11 MAC has two modes: unicast and broadcast. Since COPE broadcasts encoded packets to their next hops, the natural approach would be to use broadcast. Unfortunately, this does not work because of two reasons: poor reliability and lack of backoff.

Specifically, in the 802.11 unicast mode, packets are immediately ack-ed by their intended nexthops. The 802.11 protocol ensures reliability by retransmitting the packet at the MAC layer for a fixed number of times until a synchronous ack is received. Lack of an ack is interpreted as a collision signal, to which the sender reacts by backing off exponentially, thereby allowing multiple nodes to share the medium.

In contrast, 802.11 broadcast lacks both reliability and backoff. A broadcast packet has many intended receivers, and it is unclear who should ack. In the absence of the acks, the broadcast mode offers no retransmissions and consequently very low reliability. Additionally, a broadcast source cannot detect collisions, and thus does not back off. If multiple backlogged nodes share the broadcast channel, and each of them continues sending at the highest rate, the resulting throughput is therefore very poor, due to high collision rates.

Our solution is *pseudo-broadcast*, which piggybacks on 802.11 unicast and benefits from its reliability and backoff mechanism. Pseudo-broadcast unicasts packets that are meant for broadcast. The link-layer destination field is set to the MAC address of one of the intended recipients. An XOR-header is added after the link-layer header, listing all nexthops of the packet, Since all nodes are set in the promiscuous mode, they can overhear packets not addressed to them. When a node receives a packet with a MAC address different from its own, it checks the XOR-header to see if it is a nexthop. If so, it processes the packet further, 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.

Pseudo-broadcast is also more reliable than simple broadcast. The packet is retransmitted multiple times until its designated MAC receiver receives the packet and acks it, or the number of retries is exceeded. A desirable side effect of these retransmissions is that nodes that are promiscuously listening to this packet have more opportunities to hear it. Pseudo-broadcast, however, does not completely solve the reliability problem, which we address in the next section.

D. Hop-by-hop ACKs and Retransmissions

(a) Why hop-by-hop ACKs? Encoded packets require all nexthops to acknowledge the receipt of the associated native packet for two reasons. First, encoded packets are headed to multiple nexthops, but the sender gets synchronous MAC-layer ACKs only from the nexthop that is set as the link layer destination of the packet (as explained in the previous section). There is still a probability of loss to the other nexthops from whom it does not get synchronous ACKs. Second, COPE may optimistically guess that a nexthop has enough information to decode an XOR-ed packet, when it actually does not.

The standard solution to wireless losses is to mask error-induced drops by recovering lost packets locally through acknowledgments and retransmissions [39], [40]. COPE too addresses this problem using local retransmissions; the sender expects the nexthops of an XOR-ed packet to decode the XOR-ed packet, obtain their native packet, and ack it. If any of the native packets is not ack-ed within a certain interval, the packet is retransmitted, potentially encoded with another set of native packets.

(b) Asynchronous Acks and Retransmissions: How should we implement these hop-by-hop ACKs? For non-coded packets, we simply leverage the 802.11 synchronous ACKs. Unfortunately, extending this synchronous ACK approach to coded packets is highly inefficient, as the overhead incurred from sending each ack in its own packet with the necessary IP and WiFi headers would be excessive. Thus, in COPE encoded packets are ack-ed asynchronously.

When a node sends an encoded packet, it schedules a retransmission event for each of the native packets in the encoded packet. If any of these packets is not ack-ed within T_a seconds, the packet is inserted at the head of the output queue and retransmitted. (T_a is slightly larger than the round trip time of a single link.) Retransmitted packets may get encoded with other packets according to the scheme in §V-A.

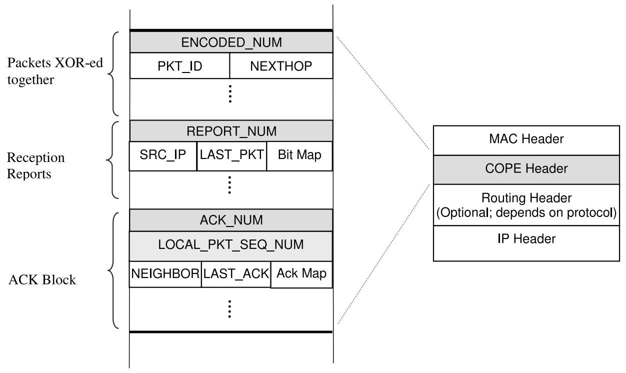


Fig. 4—COPE Header. The first block identifies the native packets XOR-ed and their nexthops. The second block contains reception reports. Each report identifies a source, the last IP sequence number received from that source, and a bit-map of most recent packets seen from that source. The third block contains asynchronous acks. Each entry identifies a neighbor, an end point for the ACK map, and a bit-map of ack-ed packets.

A nexthop that receives an encoded packet decodes it to obtain its native packet, and immediately schedule an ack event. Before transmitting a packet, the node checks its pending ack events and incorporates the pending acks in the COPE header. If the node has no data packets to transmit, it sends the ACKs in periodic control packets–the same control packets used to send reception reports.

E. Preventing TCP Packet Reordering

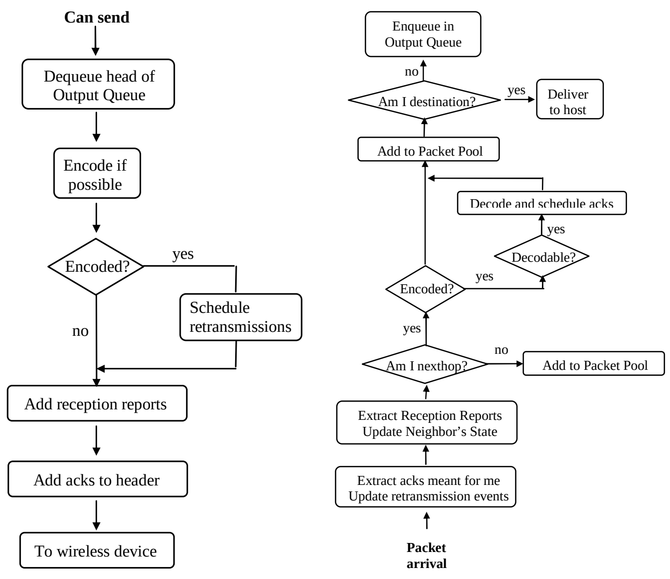
Asynchronous ACKs can cause packet reordering, which may be confused by TCP as a sign of congestion. Thus, COPE has an *ordering agent*, which ensures that TCP packets are delivered in order. The agent ignores all packets whose final IP destinations differ from the current node, as well as non-TCP packets. These packets are immediately passed to the next processing stage. For each TCP flow ending at the host, the agent maintains a packet buffer and records the last TCP sequence number passed on to the network stack. Incoming packets that do not produce a hole in the TCP sequence stream are immediately dispatched to the transport layer, after updating the sequence number state. Otherwise, they are withheld in the buffer till the gap in the sequence numbers is filled, or until a timer expires.

VI. Implementation Details

COPE adds special packet headers and alters the control flow of the router to code and decode packets. This section describes both parts.

A. Packet Format

COPE inserts a variable-length coding header in each packet, as shown in Fig. 4. If the routing protocol has its own header (e.g., Srcr [32]), COPE's header sits between the routing and the MAC headers. Otherwise, it sits between the MAC and IP headers. Only the shaded fields in Fig. 4 are



(a) Sender side **(b) Receiver side**

Fig. 5—Flow chart for our COPE Implementation.

required in every COPE header. The COPE header adds less than 5% overhead to each packet. The coding header contains the following 3 blocks.

(a) Ids of the coded native packets: The first block records meta-data to enable packet decoding. It starts with `ENCODED_NUM`, the number of native packets XOR-ed together. For each native packet, the header lists its `ID`, which is a 32-bit hash of the packet's source IP address and IP sequence number. This is followed by the MAC address of the native packet's `Nexthop`. When a node hears an XOR-ed packet, it checks the list of `Nexthops` to determine whether it is an intended recipient for any of the native packets XOR-ed together, in which case it decodes the packet, and processes it further.

(b) Reception reports: Reception reports constitute the second block in the XOR header, as shown in Fig. 4. The block starts with the number of the reports in the packet, `REPORT_NUM`. Each report specifies the source of the reported packets `SRC_IP`. This is followed by the IP sequence number of the last packet heard from that source `Last_PKT`, and a bit-map of recently heard packets. For example, a report of the form {`128.0.1.9, 50, 10000001`} means that the last packet this node has heard from source `128.0.1.9` is packet 50, and it has also heard packets 42 and 49 from that source but none in between. The above representation for reception reports has two advantages: compactness and effectiveness. In particular, the bit-map allows the nodes to report each packet multiple times with minimal overhead. This guards against reception reports being dropped at high congestion.

(c) Expressing asynchronous acks compactly and robustly: To ensure ack delivery with minimum overhead, we use cumulative acks. Since they implicitly repeat ack information, cumulative acks are robust against packet drops. Each node maintains a per-neighbor 16-bit counter, called `Neighbor_Seqno_Counter`. Whenever the node sends a packet to that neighbor, the counter is incremented and its value is assigned to the packet as a local sequence number, `Local_PKT_SEQ_NUM`. The two neighbors use this sequence number to identify the packet. Now, a node can use cumulative acks on a per-neighbor basis. Each coded packet contains an ack header as shown in Fig. 4. The ack block starts with the number of ack entries, followed by the packet local sequence number. Each ack entry starts with a neighbor MAC address. This is followed by a pointer to tell the neighbor where the cumulative acks stop, and a bit-map indicating previously received and missing packets. For example, an entry of {`A, 50, 01111111`} acks packet 50, as well as the sequence 43-49, from neighbor *A*. It also shows that packet 42 is still missing. Note that though we use cumulative acks, we do not guarantee reliability at the link layer. In particular, each node retransmits a lost packet a few times (default is 2), and then gives up.

B. Control Flow

Fig. 5 abstracts the architecture of COPE. On the sending side, (shown in Fig. 5(a)), whenever the MAC signals an opportunity to send, the node takes the packet at the head of its output queue and hands it to the coding module (§V-A). If the node can encode multiple native packets in a single XOR-ed version, it has to schedule asynchronous retransmissions. Either way, before the packet can leave the node, pending reception reports and acks are added.

On the receiving side, (shown in Fig. 5(b)), when a packet arrives, the node extracts any acks sent by this neighbor to the node. It also extracts all reception reports and updates its view of what packets its neighbor stores. Further processing depends on whether the packet is intended for the node. If the node is not a nexthop for the packet, the packet is stored in the Packet Pool. If the node is a nexthop, it then checks if the packet is encoded. If it is, the node tries to decode by XOR-ing the encoded packet with the native packets it stores in its Packet Pool. After decoding it acks this reception to the previous hop and stores the decoded packet in the Packet Pool. The node now checks if it is the ultimate destination of the packet, if so it hands the packet off to the higher layers of the network stack. If the node is an intermediate hop, it pushes the packet to the output queue. If the received packet is not encoded, the packet is simply stored in the Packet Pool and processed in the same fashion as a decoded packet.

VII. Experimental Results

This section uses measurements from a 20-node wireless testbed to study both the performance of COPE and the interaction of network coding with the wireless channel and higher-layer protocols. Our experiments reveal the following:

- When the wireless medium is congested and the traffic consists of many random UDP flows, COPE delivers a 3-4*x* increase in the throughput of our wireless testbed.
- When the traffic does not exercise congestion control (e.g., UDP), COPE's throughput improvement substantially exceeds the expected coding gain and agrees with the Coding+MAC gain.

- For a mesh network connected to the Internet via a gateway, the throughput improvement observed with COPE varies depending on the ratio of download traffic to upload traffic at the gateway, and ranges from 5% to 70%.
- Hidden terminals create a high loss rate that cannot be masked even with the maximum number of 802.11 retransmissions. In these environments, TCP does not send enough to utilize the medium and does not create coding opportunities. In environments with no hidden terminals, TCP's throughput improvement with COPE agrees with the expected coding gain.

A. Testbed

(a) Characteristics: We have a 20-node wireless testbed that spans two floors in our building connected via an open lounge. The nodes of the testbed are distributed in several offices, passages, and lounges. Paths between nodes are between 1 and 6 hops in length, and the loss rates of links on these paths range between 0 and 30%. The experiments described in this paper run on 802.11a a bit-rate of 6Mb/s. Running the testbed on 802.11b is impractical because of a high level of interference from the local wireless networks.

(b) Software: Nodes in the testbed run Linux. COPE is implemented using the Click toolkit [41]. Our implementation runs as a user space daemon, and sends and receives raw 802.11 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 (e.g., `eth0`). The implementation is agnostic to upper and lower layer protocols, and can be used by various protocols including UDP and TCP.

(c) Routing: Our testbed nodes run the Srcr implementation [32], a state-of-the-art routing protocol for wireless mesh networks. The protocol uses Djikstra's shortest path algorithm on a database of link weights based on the ETT metric [32]. The router output queue is bounded at 100 packets.

(d) Hardware: Each node in the testbed is a PC equipped with an 802.11 wireless card attached to an omni-directional antenna. The cards are based on the NETGEAR 2.4 & 5 GHz 802.11a/g chipset. They transmit at a power level of 15 dBm, and operate in the 802.11 ad hoc mode, with RTS/CTS disabled as in the default MAC.

(e) Traffic Model: We use a utility program called `udpgen` [42] to generate UDP traffic, and `ttcp` [43] to generate TCP traffic. We either use long-lived flows, or many shorter flows that match empirical studies of Internet traffic [44], [45], i.e., they have Poisson arrivals, and a Pareto file size with the shape parameter set to 1.17.

B. Metrics

Our evaluation uses the following metrics.

- *Network Throughput:* the measured total end-to-end data throughput, i.e., the sum of the data throughput of all flows in the network as seen by their corresponding applications. The overhead incurred by the extra coding headers/control packets is therefore taken into account.
- *Throughput Gain:* the ratio of the measured network throughputs with and without COPE. We compute the throughput gain from two consecutive experiments, with coding turned on, then off.

C. COPE in gadget topologies

We would like to compare COPE's actual throughput gain with the theoretical gains described in §IV, and study whether it is affected by higher layer protocols. We start by looking at a few toy topologies with good link quality (medium loss rate after MAC retries $< 1\%$), and no hidden terminals.

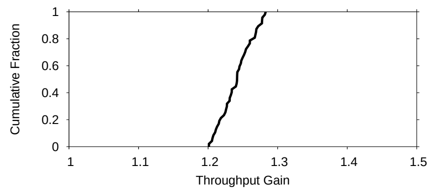
1) Long-Lived TCP Flows: We run long-lived TCP flows over 3 toy topologies: Alice-and-Bob, the "X", and the cross topologies depicted in Figs. 1 and 3. Fig. 6 plots the CDFs of the TCP throughput gain measured over 40 different runs. For the Alice-and-Bob topology the gain, shown in Fig. 6(a), the median gain is close to the theoretical coding gain of 1.33. The difference of $5-8\%$ is due to the overhead of COPE's headers, as well as asymmetry in the throughput of the two flows, which prevents the router from finding a codemate for every packet. Similarly, for the "X"-topology, the gain in Fig. 6(b) is comparable to the optimal coding gain of 1.33. Finally, Fig. 6(c) shows the throughput gain for the cross topology with TCP. The gains are slightly lower than the expected coding gain of 1.6 because of header overhead, imperfect overhearing, and a slight asymmetry in the throughputs of the four flows.

The above experimental results reveal that when the traffic exercises congestion control, the throughput gain corresponds to the coding gain, rather than the Coding+MAC gain. The congestion control protocol, built into TCP, naturally matches the input rate at the bottleneck to its draining rate. When multiple long-lived TCP flows get bottlenecked at the same router, the senders back off and prevent excessive drops, leaving only pure coding gains.

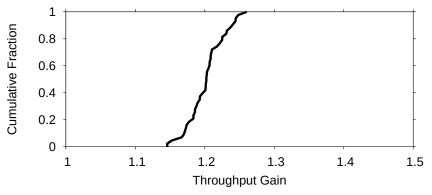
2) UDP Flows: We repeat the above experiments with UDP and evaluate the throughput gains. Fig. 7 plots a CDF of the UDP gain with COPE for the Alice-and-Bob, the "X", and the cross topologies. The figure shows that the median UDP throughput gains for the three topologies are 1.7, 1.65, and 3.5 respectively.

Interestingly, the UDP gains are much higher than the TCP gains; they reflect the Coding+MAC gains for these toy topologies. Recall from §IV that the coding gain arises purely from the reduction in the number of transmissions achieved with COPE. Additionally, coding compresses the bottleneck queues, preventing downstream congested routers from dropping packets that have already consumed bandwidth, and producing a Coding+MAC gain. In §IV, we have shown that the theoretical Coding+MAC gains for the above toy topologies are 2, 2, and 4 respectively. These numbers are fairly close to the numbers we observe in actual measurements.

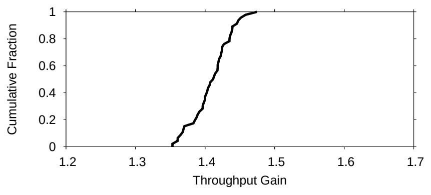
One may wonder why the measured throughput gains are smaller than the theoretical Coding+MAC gain bounds. The XOR headers add a small overhead of 5-8%. However, the difference is mainly due to imperfect overhearing and flow asymmetry. Specifically, the nodes do not overhear all transmitted packets. Further, some senders capture the wireless



(a) TCP gain in the Alice-and-Bob topology

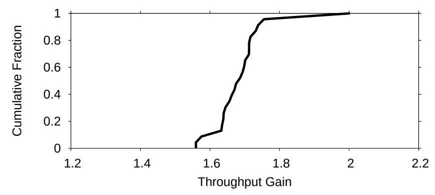


(b) TCP gain in the X-topology

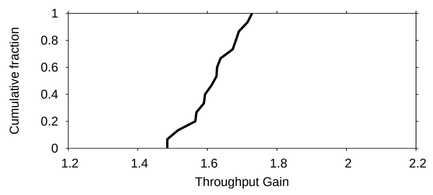


(c) TCP gain in the cross topology

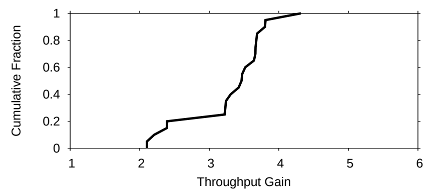
Fig. 6—CDF of throughput gains obtained with COPE, for long-lived TCP flows.



(a) UDP gain in the Alice-and-Bob topology



(b) UDP gain in the X-topology



(c) UDP gain in the cross topology

Fig. 7—CDF of throughput gains obtained with COPE, for UDP flows.

channel sending more traffic in a particular direction, which reduces coding opportunities and overall gain.

In practice, traffic is a combination of congestion-controlled and uncontrolled flows. Further, most TCP flows are short-lived and do not fully exercise congestion control during slow-start.Thus, one would expect COPE's gains to be higher than those observed with long-lived TCP and lower than those observed with UDP. Indeed, we have run experiments for the Alice-and-Bob scenario with short-lived TCP flows with Poisson arrivals and Pareto transfer size. Depending on the flow inter-arrival times, the measured throughput gains vary between the coding gain and the Coding+MAC gain.

D. COPE in an Ad Hoc Network

How does COPE perform in a wireless mesh network? We have advocated a simple approach to wireless network coding where 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 practice, and whether they can be detected using only local information. Thus, in this section, we run experiments on our 20-node testbed to gauge the throughput increase provided by COPE in an ad hoc network.

1) TCP: We start with TCP flows that arrive according to a Poisson process, pick sender and receiver randomly, and transfer files whose sizes follow the distribution measured on the Internet [45].

Surprisingly, in our testbed, TCP does not show any significant improvement with coding (the average gain is 2-3%). The culprit is TCP's reaction to collision-related losses. There are a number of nodes sending packets to the bottleneck nodes, but they are not within carrier sense range of each other, resulting in the classic hidden terminals problem. This creates many collision-related losses that cannot be masked even with the maximum number of MAC retries. To demonstrate this point, we repeat the TCP experiments with varying number of MAC retransmissions with RTS/CTS enabled. Note that disabling RTS/CTS exacerbates the problem further. Fig. 8 plots the end-to-end loss rates for TCP flows as a function of the number of

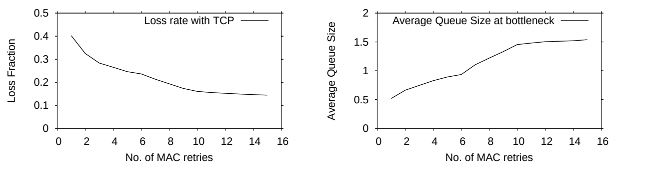


Fig. 8—End-to-end loss rate and average queue size at the bottlenecks for the TCP flows in the testbed. Loss rates are as high as 14% **even after** 15 **MAC retries; TCP therefore performs poorly. The queues at the bottlenecks almost never build up resulting in very few coding opportunities and virtually no gains.**

MAC retransmissions. *These experiments have COPE turned off.* Even after 15 MAC retries (the maximum possible) the TCP flows experience 14% loss. As a result, the TCP flows suffer timeouts and excessive back-off, and are unable to ramp up and utilize the medium efficiently. Fig. 8 plots the average queue sizes at the bottleneck nodes.[1] The bottleneck nodes never see enough traffic to make use of coding; most of their time is spent without any packets in their queues or just a single packet. Few coding opportunities arise, and hence the performance is the same with and without coding.

Collision-related losses are common in wireless networks and recent work has studied their debilitating effect on TCP [46], [47]. Making TCP work in such a setting would imply solving the collision problem; such a solution is beyond the scope of this paper.

Would TCP be able to do better with COPE if we eliminated collision-related losses? We test the above hypothesis by performing the following experiment. We compress the topology of the testbed by bringing the nodes closer together, so that they are within carrier sense range. We artificially impose the routing graph and inter-node loss rates of the original testbed. The intuition is that the nodes are now within carrier sense range and hence can avoid collisions. This will reduce the loss rates and enable TCP to make better use of the medium. We repeat the above experiment with increasing levels of

[1]The few nodes connecting the two floors are where the flows intersect; they are main bottlenecks in our testbed.

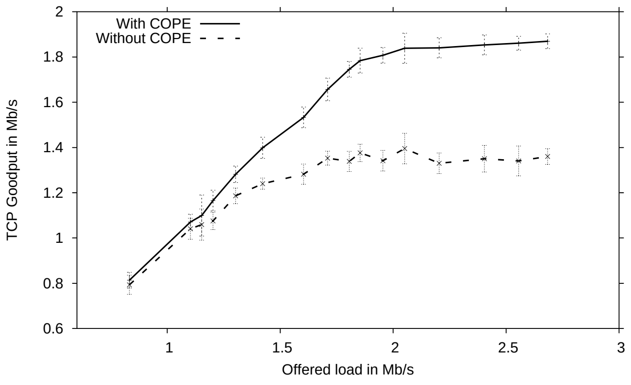


Fig. 9—COPE provides 38% increase in TCP goodput when the testbed topology does not contain hidden terminals.

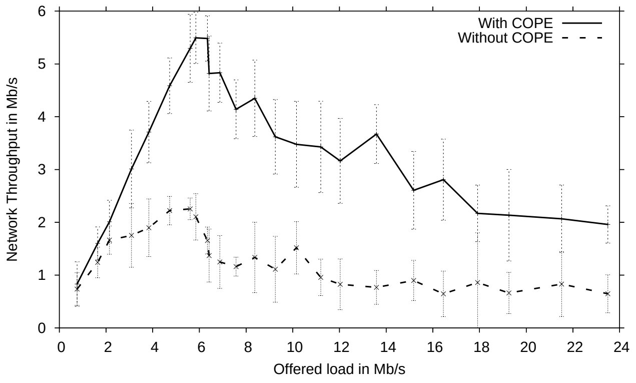


Fig. 10—COPE can provide a several-fold (*3-4x*) increase in the throughput of wireless Ad hoc networks. Results are for UDP flows with randomly picked source-destination pairs, Poisson arrivals, and heavy-tail size distribution.

congestion obtained by decreasing the inter-arrival times of the TCP flows. Fig. 9 plots the network TCP goodput with and without COPE as a function of the demand. For small demands, COPE offers a slight improvement since coding opportunities are scarce. As the demands increase, network congestion and coding opportunities increase, leading to higher goodput gains. As congestion increases beyond a certain level, the throughput levels off, reflecting the fact that the network has reached its capacity and cannot sustain additional load. At its peak, COPE provides 38% improvement over no coding. The medium loss rates after retransmissions are negligible. The TCP flows are therefore able to use the medium efficiently, providing coding opportunities and throughput gains.

2) UDP: We repeat the large scale testbed experiments with UDP. The flows again arrive according to a Poisson process, pick sender and receiver randomly, and transfer files whose sizes follow the distribution measured on the Internet [45]. We vary the arrival rates of the Poisson process to control the offered load. For each arrival rate, we run 10 trials, with coding on and then off (for a total of 500 experiments), and compute the network throughput in each case.

Fig. 10 shows that COPE greatly improves the throughput of these wireless networks, by a factor of 3-4*x* on average. The figure plots the aggregate end-to-end throughput as a function of the demands, both with COPE and without. At low demands

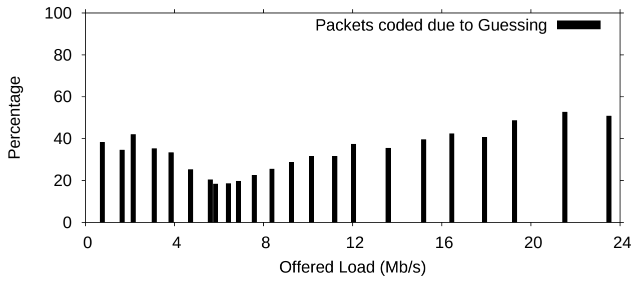


Fig. 11—Percentage of packets coded in the testbed due to guessing, as a function of offered load, for the set of experiments in Fig. 10.

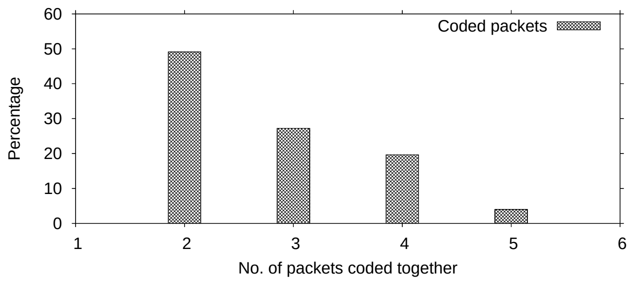


Fig. 12—Distribution of number of packets coded together in the test bed at the peak point of Fig. 10.

(below 2Mb/s), coding opportunities are scarce, and COPE performs similarly to no coding. As demands increase, both network congestion and the number of coding opportunities increase. In such dense networks, the performance without coding deteriorates because of the high level of contention and consequent packet loss due to collisions. In contrast, coding reduces the number of transmissions, alleviates congestion, and consequently yields higher throughput.

It is interesting to examine how much of the coding is due to guessing, as opposed to reception reports. Fig. 11 plots the percentage of packets that have been coded because of guessing for the experiments in Fig.10. It is calculated as follows: If n packets are coded together, and at most k packets could be coded using reception reports alone, then $n - k$ packets are considered to be coded due to guessing. The figure shows that the benefit of guessing varies with demands. At low demands, the bottleneck nodes have small queues, leading to a short packet wait time. This increases dependence on guessing because reception reports could arrive too late, after the packets have been forwarded. As demands increase, the queues at the bottlenecks increase, resulting in longer wait times, and consequently allowing more time for reception reports to arrive. Hence, the importance of guessing decreases. As demands surge even higher, the network becomes significantly congested, leading to high loss rates for reception reports. Hence, a higher percentage of the coding decisions is again made based on guessing.

Let us now examine in greater detail the peak point in Fig. 10, which occurs when demands reach 5.6 Mb/s. Fig. 12 shows the PDF of the number of native packets XOR-ed at the bottleneck nodes (i.e., the nodes that drop packets). The figure shows that, on average, nearly 3 packets are getting coded together. Due to the high coding gain, packets are drained

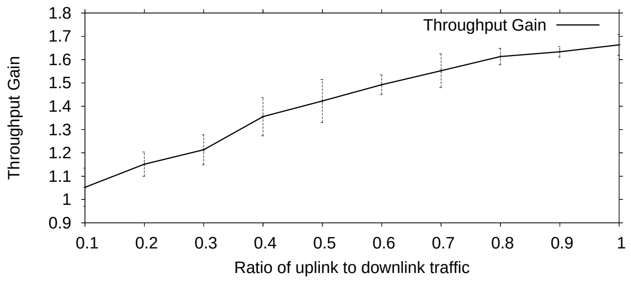


Fig. 13—COPE's throughput gain as a function of the ratio of uplink to downlink traffic at in a congested mesh *access* network.

much faster from the queues of the bottleneck nodes. The result is an average throughput gain of 3-4*x*.

E. COPE in a Mesh Access Network

There is growing interest in providing cheap Internet access using multi-hop wireless networks that connect to the rest of the Internet via one or more gateways/access points [1], [48], [49]. We evaluate COPE in such a setting, where traffic is flowing to and from the closest gateway. We divide the nodes in the testbed into 4 sets. Each set communicates with the Internet via a specific node that plays the role of a gateway. We use UDP flows,[2] and control the experiments by changing the ratio of upload traffic to download traffic. Fig. 13 plots the throughput gains as a function of this ratio.

The throughput gain increases as the fraction of uplink traffic increases. When the amount of uplink traffic is small, gains are correspondingly modest; around $5-15\%$. As uplink traffic increases, gains increase to 70%. COPE's throughput gain relies on coding opportunities, which depend on the diversity of the packets in the queue of the bottleneck node. For example, in the Alice-and-Bob topology, if only 10% of the packets in the bottleneck queue are from Alice and 90% from Bob, then coding can at best sneak Alice's packets out on Bob's packets. Hence, as the ratio of uplink traffic increases, the diversity of the queues at bottlenecks increases, more coding opportunities arise, and consequently higher throughput gains are obtained.

F. Fairness

The access network experiment above illuminates the effect fairness has on coding opportunities. An important source of unfairness in wireless networks is the comparative quality of the channels from the sources to the bottleneck, usually referred to as the *capture effect*. For example, in the Alice and Bob experiment, if the channel between Alice and the router is worse than that between Bob and the router, Alice might be unable to push the same amount of traffic as Bob. Although the 802.11 MAC should give a fair allocation to all contenders, the sender with the better channel (here Bob) usually captures the medium for long intervals. The routing protocol tries to discount the capture effect by always selecting the stronger links; but in practice, capture always happens to some degree.

[2]As mentioned earlier, in the uncompressed testbed, TCP backs off excessively because of collision-based losses from hidden terminals, and does not send enough to fully utilize the medium.

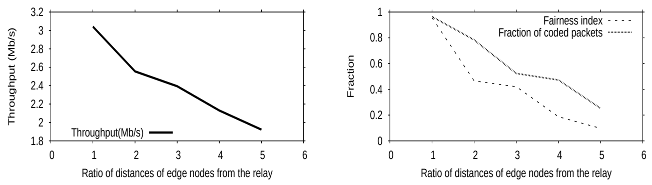


Fig. 14—Effect of unequal channel qualities on coding opportunities and throughput gain in the Alice-and-Bob topology. COPE aligns the fairness and efficiency objectives. Increased fairness increases coding opportunities and hence improves the aggregate throughput.

We study the effect of capture on COPE by intentionally stressing the links in the Alice and Bob topology. We set it up such that both Alice and Bob are equidistant from the router, and compute the total network throughput. We then gradually move Alice's node away from the router, and repeat the experiment and the measurements.

Fig. 14 shows the network throughput as a function of the ratio of Alice's and Bob's distance to the router. It also shows the percentage of coded packets and the *fairness index*, computed as the ratio of Alice's throughput to Bob's. As Alice moves further away, Bob increasingly captures the channel, reducing fairness, coding opportunities, and the aggregate network throughput. Interestingly, without coding, fairness and efficiency are conflicting goals; throughput increases if the node with the better channel captures the medium and sends at full blast. Coding, however, aligns these two objectives; increasing fairness increases the overall throughput of the network.

VIII. Discussion And Conclusion

Finally, we would like to comment on the scope of COPE. The present design targets stationary wireless mesh networks, where the nodes are not resource-constrained. More generally, COPE can be used in multi-hop wireless networks that satisfy the following:

- *Memory:* COPE's nodes need to store recently heard packets for future decoding. Only packets in flight are used in coding; there is no need to store packets that have already reached their destination. Consequently, the storage requirement should be slightly higher than a delay-bandwidth product. (For e.g., an 11 Mb/s network with a 50ms RTT has a delay-bandwidth product of 70 KB.)
- *Omni-directional antenna:* Opportunistic listening requires omni-directional antennas to exploit the broadcast property.
- *Power requirements:* Our current design of COPE does not optimize power usage and assumes the nodes are not energy limited.

The ideas in COPE may be applicable beyond WiFi mesh networks. Note that COPE can conceptually work with a variety of MAC protocols including WiMax and TDMA. One may envision modifying COPE to address the needs of sensor networks. Such a modification would take into account that only a subset of the sensor nodes is awake at any point of time and can participate in opportunistic listening. Sensor nodes may also trade-off saved transmissions for reduced battery usage, rather than increased throughput. Additionally, COPE may be useful for cellular relays. Deploying cellular base stations is usually expensive. A cheap way to increase coverage is to deploy relay nodes that intervene between the mobile device and the base station [50], creating a multi-hop cellular backbone. COPE would allow cellular relays to use the bandwidth more efficiently. Indeed, after the publication of COPE, we have learned that Ericsson has independently proposed a design for cellular relays with a subset of COPE's functionality, where the cellular relay XORs only duplex flows, as in the Alice-and-Bob scenario [50]. This scheme can be extended to make full use of the ideas embedded in COPE.

Our community knows a few fundamental approaches that can improve wireless throughput, including more accurate congestion control, better routing, and efficient MAC protocols. We believe that COPE is an important step forward in our understanding of the potential of wireless networks because it presents a new orthogonal axis that can be manipulated to extract more throughput; namely, how to maximize the amount of data delivered in a single transmission. This is coding, which is an old theme, traditionally used at the physical and application layers. But COPE and a few other recent projects [16], [51] introduce coding to the networking community as a practical tool that can be integrated with forwarding, routing, and reliable delivery.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *SIGCOMM*, vol. 34, no. 4, August 2004.

[2] 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*.

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

[4] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," in *IEEE Transactions on Information Theory*, vol. 46, no. 4, July 2000, pp. 1204–1216.

[5] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," in *IEEE Transactions on Information Theory*, vol. 49, no. 2, February 2003, pp. 371–381.

[6] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

[7] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger, "On randomized network coding," in *41st Annual Allerton Conference on Communication, Control, and Computing*, October 2003.

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

[9] Y. Wu, P. Chou, Q. Zhang, K. Jain, W. Zhu, and S. Kung, "Network planning in wireless ad hoc networks: a cross-layer approach," pp. 136–150, January 2005.

[10] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1906–1918, November 2005.

[11] J. Widmer, C. Fragouli, and J. LeBoudec, "Energy-efficient broadcasting in wireless ad-hoc networks," in *NetCod 2005*.

[12] Y. Sagduyu and A. Ephremides, "Joint scheduling and wireless network coding," in *NetCod 2005*.

[13] Y. Chen, S. Kishore, and J. Li, "Wireless diversity through network coding," *WCNC*, vol. 3, no. 5, pp. 1681–1686, 2006.

[14] X. Bao and J. Li, "On the outage properties of adaptive network coded cooperation (ancc) in large wireless networks," in *ICASSP*, vol. 4, May 2006.

[15] A. A. Hamra, C. Barakat, and T. Turletti, "Network coding for wireless mesh networks: A case study," in *WoWMoM*, June 2006.

[16] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *SIGCOMM*, vol. 37, no. 4, August 2007.

[17] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum cost multicast: A decentralized approach based on network coding," in *IEEE INFOCOM 05*, 2005.

[18] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," in *IEEE Transactions on Information Theory*, vol. 51, no. 8, August 2005, pp. 2745–2759.

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

[20] X. Bao and J. Li, "Matching code-on-graph with network-on-graph: Adaptive network coding for wireless relay networks," in *43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[21] Y. Sagduyu and A. Ephremides, "Network coding in wireless queueing networks: Tandem network case," in *ISIT*, July 2006, pp. 192–196.

[22] ——, "Crosslayer design for distributed mac and network coding in wireless ad hoc networks," in *ISIT*, September 2005, pp. 1863–1867.

[23] S. Katti, H. S. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," in *SIGCOMM*, vol. 36, no. 4, August 2006.

[24] Y. Wu, J. Padhye, R. Chandra, V. Padmanabhan, and P. A. Chou, "The local mixing problem," in *Information Theory and Applications Workshop. San Diego, CA, Feb. 2006.*

[25] S. Rayanchu and S. S. S. Banerjee, "An analysis of wireless network coding for unicast sessions: The case for coding-aware routing," in *IEEE INFOCOM*, May 2007, pp. 1028–1036.

[26] J. Liu, D. Goeckel, and D. Towsley, "Bounds on the gain of network coding and broadcasting in wireless networks," in *IEEE INFOCOM*, May 2007, pp. 724–732.

[27] P. Popovski and H. Yomo, "Bi -directional amplification of throughput in a wireless multi-hop network," in *VTC, 2006*.

[28] T. Aulin and M. Xiao, "A physical layer aspect of network coding with statistically independent noisy channels," in *IEEE International Conference on Communications*, vol. 9, no. 4, June 2006, pp. 3996–4001.

[29] S. Zhang, S. C. Liew, and P. P. Lam, "Physical-layer network coding," in *MobiCom*, September 2006.

[30] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: Analog network coding," in *ACM SIGCOMM*, vol. 37, no. 4, August 2007.

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

[32] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in *MobiCom*, August 2005.

[33] R. Draves, J. Padhye, and B. Zill, "Comparison of Routing Metrics for Multi-Hop Wireless Networks," in *SIGCOMM*, vol. 34, no. 4, August 2004.

[34] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.

[35] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," in *SIGCOMM*, vol. 35, no. 4, August 2005.

[36] 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 *SIGCOMM*, vol. 35, no. 4, August 2005.

[37] Z. Li and B. Li, "Network Coding in Undirected Networks," in *CISS*, 2004.

[38] "Internet packet size distributions: Some observations," http://netweb.usc.edu/ rsinha/pkt-sizes/.

[39] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving tcp performance over wireless links," in *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, December 1997.

[40] p. a. IEEE 802.11 WG, "Wireless lan medium access control (mac) and physical layer (phy) specifications," *Standard Specification,IEEE, 1999.*

[41] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, August 2000.

[42] "udpgen," http://pdos.csail.mit.edu/click/ex/udpgen.html.

[43] "ttcp," http://ftp.arl.mil/ftp/pub/ttcp/.

[44] V. Paxson and S. Floyd, "Wide-area traffic: the failure of poisson modeling," in *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, June 1995, pp. 226–244.

[45] M. E. Crovella, M. S. Taqqu, and A. Bestavros, "Heavy-tailed probability distributions in the World Wide Web," in *A Practical Guide To Heavy Tails*, R. J. Adler, R. E. Feldman, and M. S. Taqqu, Eds. New York: Chapman and Hall, 1998, ch. 1, pp. 3–26.

[46] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, “The impact of multihop wireless channel on tcp throughput and loss,” in *INFOCOM*, vol. 3, April 2003, pp. 1744–1753.

[47] C. cheng Chen, E. Seo, H. Kim, and H. Luo”, “Self-learning collision avoidance for wireless networks,” in *INFOCOM*, April 2006, pp. 1–12.

[48] P. Bhagwat, B. Raman, and D. Sanghi, “Turning 802.11 inside-out,” in *HotNets*, 2003.

[49] “Nokia rooftop wireless routing,” white Paper.

[50] “Definition and assessment of relay based cellular deployment concepts for future radio scenarios considering 1st protocol characteristics. Chapter 5,” https://www.ist-winner.org/DeliverableDocuments/D3.4.pdf.

[51] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein, “Growth codes: Maximizing sensor network data persistence,” in *SIGCOMM*, vol. 36, no. 4, August 2006.

APPENDIX

A. Proof of Lemma 4.1

Proof: We first prove the upper bound of 2. Note that if the intermediate node codes N native packets together, these packets have to be to N different next-hops, by the coding rule of §III(b). In the absence of opportunistic listening, the only routing neighbor that has a packet is the previous hop of that packet. Suppose the intermediate hop codes ≥ 2 packets from the same neighbor. All other neighbors must have $\leq N-2$ packets in the encoded packet, which violates the coding rule. As a result, the intermediate hop can code at most one packet from a neighbor. Without opportunistic listening, this is the only native packet in the encoded packet that this neighbor has. Invoking the coding rule, this implies that the intermediate hop can code at most 2 packets together. This implies that the total number of transmissions in the network can at most be halved with coding, for a coding gain of 2.

Indeed, this gain is achievable in the chain of N links in Fig. 3(a). This topology is an extension of the Alice-and-Bob example where $N = 2$. The no-coding case requires a total of $2N$ transmissions to deliver a packet from Alice to Bob, and vice-versa. On the other hand, in the presence of coding, each of the $N-1$ intermediate nodes on the path can transmit information simultaneously to neighbors on either side by coding the two packets traversing in opposite directions, for a total of $N+1$ transmissions. The coding gain in this case is $\frac{2N}{N+1}$, which tends to 2 as the chain length grows. ∎

B. Proof of Lemma 4.2

Proof: We assume that the network uses the 802.11 MAC, which allocates a fair share to all active nodes. As proved above, in the absence of opportunistic listening, a node can code at most 2 packets together. Hence, a bottleneck node can drain its packets atmost twice as fast, bounding the Coding+MAC gain at 2. This gain is achieved even in the simple Alice-and-Bob experiment as explained above (longer chains result in the same Coding+MAC gain). ∎

C. Proof of Lemma 4.3

Proof: Consider the wheel topology with radius r in Fig. 3(d) with N nodes uniformly placed on the circumference, and one node at the center of the circle. We assume that the nodes use the 802.11 MAC, which allocates a fair share of the medium to all nodes. Assume that when a node transmits, all other nodes in the circle overhear this transmission, except for the diametrically opposed node (i.e., the radio range is $2r - \epsilon$, where $\epsilon \approx 0$). Suppose now that there are flows between every pair of diametrically opposed nodes. Note that nodes on either end of a diameter cannot communicate directly, but can communicate using a two-hop route through the middle node. In fact, this route is the geographically shortest route between these nodes. In the absence of coding, a single flow requires 1 transmission from an edge node, and 1 transmission from the middle node. This adds to a total of 1 transmission per edge node, and N transmissions for the middle node, across all packets. Since the MAC gives each node only a $\frac{1}{N+1}$ share of the medium, the middle node is the bottleneck in the absence of coding. However, COPE with opportunistic listening allows the middle node to code all the N incoming packets and fulfill the needs of all flows with just one transmission, thereby matching its input and output rates. Hence, the Coding+MAC gain is N, which grows without bound with the number of nodes. ∎

Sachin Katti received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, in 2003. He received the M.S. degree in Computer Science from the MIT in 2005. He is currently pursuing his Ph.D. degree in computer science at MIT. His research interests include wireless networks, coding theory, and distributed systems.

Hariharan Rahul received his B.Tech. from the Indian Institute of Technology in 1997, and his M.S. from MIT in 1999, both in computer science. He worked at Akamai Technologies, and is currently pursuing his Ph.D. Degree in Computer Science from MIT. His research interests are in Internet performance measurement, wireless networks, and distributed systems. He has won the President of India gold medal at IIT, and the Professional Services excellence award at Akamai.

Wenjun Hu is a Ph.D. student at the University of Cambridge Computer Laboratory. She has received a BA in Computer Science from the University of Cambridge in 2003. Her research interests are in the area of wireless and mobile networking. For her thesis work she is focusing on applying network coding to wireless mesh networks.

Dina Katabi is a Sloan Associate Professor in the Electrical Engineering and Computer Science department at MIT. She received her M.S. and Ph.D. degrees from MIT, in 1998 and 2003. Her work focuses on wireless networks, network security, routing, and distributed resource management. She has been awarded an NSF CAREER award in 2005, a Sloan Fellowship award in 2006, and the NBX Career Development chair in 2006. Her doctoral dissertation won an ACM Honorable Mention award.

Muriel Médard is the Edgerton Associate Professor in the EECS department at MIT and the Associate Director of LIDS. She received B.S. degrees in EECS and in Mathematics in 1989, a B.S. degree in Humanities in 1990, a M.S. degree in EE 1991, and a Sc D. degree in EE in 1995, all from MIT. Professor Medard’s interests are in the areas of network coding and reliable communications. She was awarded an NSF Career Award in 2001, the IEEE Leon K. Kirchmayer Prize Paper Award in 2002, and the Edgerton Faculty Achievement Award in 2004.

Jon Crowcroft is the Marconi Professor of Networked Systems in the Computer Laboratory, of the University of Cambridge. Prior to that he was professor of networked systems at UCL in the Computer Science Department. He is a Fellow of the ACM, a Fellow of the British Computer Society and a Fellow of the IEE and a Fellow of the Royal Academy of Engineering, as well as a Fellow of the IEEE.