Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2006-049 June 30, 2006

MORE: A Network Coding Approach to Opportunistic Routing

Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

MORE: A Network Coding Approach To Opportunistic Routing

Szymon Jakubczak szym@mit.edu
Michael Jennings mvj@csail.mit.edu
Sachin Katti skatti@mit.edu
Dina Katabi dk@mit.edu

ABSTRACT

Opportunistic routing has the potential to substantially increase wireless network throughput. Prior work on opportunistic routing, however, requires tight node coordination. Different nodes in a network must have knowledge of which packets other nodes have received. Furthermore, the nodes have to agree on which nodes should transmit which packets. Such coordination becomes fragile in dense or large networks.

This paper introduces MORE, a new opportunistic routing protocol that avoids node-coordination. Our design is rooted in the theory of network coding. Routers code packets going to the same destination and forward the coded versions. The destination decodes and recovers the original packets. This approach needs no coordination and provably maximizes network throughput. We have implemented our design and evaluated it in a 25-node testbed. Our results show that MORE provides an average throughput increase of 60% and a maximum of 10-fold, demonstrating that the theoretical gains promised by network coding are realizable in practice.

1 INTRODUCTION

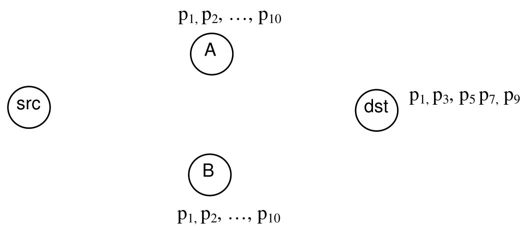
Opportunistic routing exploits the broadcast nature of the wireless medium to increase network throughput [6]. Traditional routing protocols determine the next-hop of a packet before transmission [5, 27, 16]. But wireless is a broadcast medium with dynamic characteristics. Each time a node transmits, a different subset of the nodes may receive the packet. There is always some probability a packet is heard by nodes much closer to the destination than the preselected next-hop. A traditional routing protocol ignores these fortunate receptions and continues forwarding the packet hop-by-hop on the chosen route. In contrast, opportunistic routing does not commit the packet to a particular next-hop; among the nodes that happen to hear the packet the one closest to the destination is picked as the next hop. Hence, opportunistic routing exploits fortunate receptions and multiple paths towards the destination to increase the throughput.

Opportunistic routing, however, is challenging because of the potential for spurious transmissions. Many intermediate nodes may hear the same transmission, and thus forward the same packet, wasting wireless resources, and outweighing the potential throughput gain.

This paper proposes MORE, a network-coding approach to opportunistic routing.[1] The key feature in MORE is that, though intermediate nodes forward packets they hear without consulting with each other, they do not generate spurious transmissions. MORE achieves this feature by building on the theory of network coding. Instead of solely forwarding packets they receive, intermediate nodes forward random linear combinations of packets going to the same destination. The theory shows that such randomly coded packets contain independent information with high probability [13]. To deliver N packets to the destination, it becomes sufficient to deliver *any* N such coded packets, and thus none of the coded packets is redundant. Furthermore, under simplified assumptions of Poisson traffic from source to destination and static loss characteristics, this approach is proven to maximize the throughput [23].

To grasp the intuition underlying MORE, consider the example in Fig. 1, where we would like to deliver a 10-packet file from source to destination. When the source transmits these 10 packets, the destination receives some of them (say the odd packets), while nodes A and B receive all packets. Without coding, nodes A and B need to learn which packets the destination has already received to abstain from transmitting redundant packets. Furthermore, they need to coordinate with each other to ensure that they do not forward the same packets. However, one can recover the 10 original packets from any 10 linearly independent combinations of these packets (a basic result in linear algebra [30]). The destination in our example already has 5 original packets, i.e., 5 linearly independent combinations. Thus, it can recover the file by acquiring any additional 5 linearly independent combinations. Hence, in MORE, A and B broadcast random linear combinations of the packets they receive. Because they are randomly generated, these linear combinations are very likely independent [13]. Once the destination receives 5 additional such coded packets, it recovers the whole file at once using a simple matrix inversion. It also broadcasts an acknowledgement, which causes potential forwarders to stop forwarding packets from this file. Thus, with MORE, potential forwarders need not coordinate or wait to learn who has what.

[1]MORE stands for Multi-path Opportunistic Routing Engine.



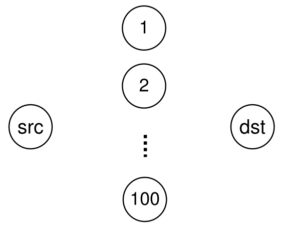
$p_1, p_2, \ldots, p_{10}$
A
src
dst
$p_1, p_3, p_5\, p_7, p_9$
B
$p_1, p_2, \ldots, p_{10}$

Figure 1—A comparison between ExOR and a network coding approach to opportunistic routing. With ExOR, *A* and *B* need to coordinate among themselves and with the destination on which packets to forward and the order of forwarding. Otherwise, they might generate spurious transmissions. With network coding, *A* and *B* transmit random linear combinations of the packets they received ($p' = r_1p_1 + \ldots + r_{10}p_{10}$, where r_i is a random number). They do not need to know which packets the destination received because the destination can retrieve the 10 original packets from any 5 randomly generated linear combinations plus the 5 packets it has already received.

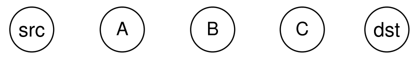
The design of MORE builds upon prior foundational work on opportunistic routing, particularly upon the ExOR protocol [6]. However, MORE departs from prior work in that it does not require potential forwarders to coordinate or consult each other. ExOR, on the other hand, requires tight node coordination. To prevent spurious transmissions, nodes in the network have to coordinate on the following issues. First, they need to exchange information to learn which packets other nodes have received. Second, they must agree on which node should forward which packet. Third, candidate forwarders have to transmit their packets in a particular order. While a forwarder transmits the packets it received, the medium is reserved and other forwarders cannot access it. Such tight coordination becomes fragile in large networks.

This paper makes the following contributions:

- It introduces MORE, a network-coding approach to opportunistic routing. MORE has a few desirable features: It does not require node coordination and thus can easily scale to large networks; it provides reliable data delivery; and its design is supported by theoretical analysis [23].
- The paper presents a prototype implementation and field tests of MORE on a 25-node wireless network, showing that MORE substantially increases wireless throughput. On our testbed, the median throughput gain with MORE is 1.6, while the maximum gain exceeds 10-fold.
- Finally, MORE is the first implementation of wireless network coding with *general linear codes*. To the best of our knowledge, there is one prior implementation of wireless network coding; it XORs packets going to different destinations [17]. In comparison, our implementation is the first to demonstrate the benefits and practicality of general random linear network codes.



(a) Multiple Independent Forwarders



(b) Skipping Hops

Figure 2—Illustration of the benefits of opportunistic routing. In (a), each of the source's transmissions has many independent chances of being received by a node closer to the destination. In (b), though the chosen route has 4 hops, node *B* or *C* may directly hear some of the source's transmissions, allowing these packets to skip a few hops.

2 Motivating Discussion

This section provides some background and motivates a network coding approach to opportunistic routing.

2.1 Benefits of Opportunistic Routing

Opportunistic routing defers the selection of the route until after the packet has been transmitted. Among the nodes that received the transmission, it selects the one with the best route to the destination as the next forwarder. This approach produces two types of gains. First, each transmission has many chances to be received by a node closer to the destination (where distance is measured using loss rate). It has been observed that the losses on the wireless links are approximately independent [26], and we make use of this assumption throughout the paper. Consider the contrived scenario in Fig. 2-a. which we took from [6]. The delivery probability between the source and each of the 100 intermediate nodes is 10%, whereas the delivery probability between an intermediate node and the destination is 100%. Traditionally routing would select one of the intermediate hops as the forwarder. This choice requires each packet to be transmitted an average of 10 times before reaching the intermediate node. The intermediate node forwards the packet using one extra transmission. The total throughput is about 9% of the medium's capacity. In comparison, opportunistic routing requires about two transmissions per packet; one transmission to reach *any* of the intermediate nodes, which happens with probability $1 - 0.9^{100} \approx 0.999$, and a second transmission to reach the destination. Thus, in this scenario, opportunistic routing can potentially increase the throughput by 5-fold, from 9% to 50% of the medium capacity. In more realistic examples, the gain will be less, but continue to be substantial.

The second throughput gain is caused by the ability of opportunistic routing to use long and relatively low-quality links. Consider the scenario in Fig. 2-b, where the best path between the source and destination is SRC-A-B-C-DST. Because of the dynamic and random nature of the wireless medium, some of the source's transmissions may be directly received by node C or the destination itself. A traditional routing protocol ignores these fortunate transmissions and keeps trying to send these packets along the predetermined route. Opportunistic routing on the other hand exploits these happy occurrences to skip some hops, reducing the number of transmissions and increasing the throughput.

2.2 ExOR: Prior Opportunistic Routing Protocol

There is only one prior protocol for opportunistic routing in wireless multi-hop networks. Proposed in [6], ExOR is best described using the topology in Fig. 2-b. The source transmits a batch of packets (10-100 packets). Each packet contains a header that lists all potential forwarders ordered according to their distance from the destination, e.g., C, B, A. Distance is measured using a function of the packet delivery probability. The candidate forwarders buffer the packets they receive and await the end of the batch. Once the source is done transmitting the batch, the forwarders forward the packets in their buffer according to their order in the forwarding list. In our example, node C is the first to transmit the packets it has overheard, followed by node B, then node A. Forwarded packets contain a batch-map that is used by each transmitter to inform the rest of the nodes of its knowledge of which packets have been received by which nodes. The forwarders use the information in the batch-map to abstain from forwarding packets that have already been received by nodes closer to the destination.

The design of ExOR requires tight node coordination. The coordination is imposed using two methods: (1) batch-maps used to exchange information about which nodes received which packets; and (2) a strict order on when a node can forward the received packets. While a forwarder transmits the packets it has received, the medium is reserved and other forwarders cannot access it. Imposing such tight coordination leads to the following undesirable effects:

- It is unclear how long a candidate forwarder should wait before it starts forwarding the packets it received. ExOR requires a forwarder to wait to learn which packets have been received by some downstream forwarders. If the path is long, many downstream forwarders may have received no packets at all, and thus will not generate any batch-maps. In this case, a forwarder has to timeout. But it is difficult to decide on a timeout value. The main gains of opportunistic routing arise from using long but low-quality links. Given the low delivery probability of such links and the dynamism of the loss characteristics in the wireless environment, it is unclear how long a forwarder should wait to ensure that downstream forwarders have not heard any packets.
- Even if the forwarder can estimate a reasonable timeout value, waiting to ensure that downstream forwarders have no packets introduces an unnecessary delay and increases the transfer time.
- Imposing a strict order on the forwarders prevents them from forwarding packets in parallel. This prevents spatial reuse of the bandwidth and reduces the overall throughput.

The contributions of ExOR, as the first opportunistic routing protocol, are highly valuable. But the difficulty of providing tight node coordination in dense or large networks instigates a need to explore alternative approaches to opportunistic routing.

2.3 Network coding is superior to source coding

The description of MORE in §1 might remind some readers of erasure codes [22](e.g., Reed-Solomon codes), a coding scheme that has been successfully used in many systems [8, 7]. Erasure coding is a form of source coding, where to deliver n original packets, the sender codes them into $m > n$ packets. The destination can recover the n original packets from any n coded packets.

Erasure coding at the source, however, suffers from the same limitations as ExOR. Specifically, the destination needs *n distinct* coded packets to recover the original n packets. Even if the packets are coded using erasure codes, candidate forwarders still have to coordinate to ensure that they do not unnecessarily transmit the same coded packet.

MORE acquires its effectiveness from building on network coding. Here the routers themselves can generate new distinct packets by forming random linear combinations of the packets they receive. Since random coding at the routers has been proven to generate distinct and independent coded packets (with an exponentially high probability [12]), the routers do not have to coordinate which node forwards which packets.

3 DEFINITIONS

We start with a few definitions that are used throughout the paper.

(a) Native Packet: a non-coded packet.

(b) Batch: MORE sends packets between a source-destination pair in batches. Only packets in the same batch can be coded together. The batch size, K, may vary from one batch to another.

(c) Coded Packet: a linear combination of native packets belonging to the same batch—i.e., a coded packet is $p' = \sum_i c_i p_i$, where c_i is some number, and the p_i's are native packets from the same batch. Multiplying a packet by a number c_i implies multiplying each byte of the packet with c_i.[2] Addition of 2 packets is done by XORing the corresponding bytes in each packet. Thus a linear combination of packets is created by first multiplying each packet with its corresponding coefficient and then XORing all the packets together. Note that though coded packets are created at a node by linearly combining previously coded packets, they are also linear combinations of the native packets themselves. In particular, assume we create coded packets by linearly combining native packets $p'_j = \sum_i c_{ji} p_i$, where p_i is a native packet. Then we linearly combine these coded packets to create more coded packets as follows: $p' = \sum_j r_j p'_j$, where r_j is some number. The resulting coded packet p' can be expressed in terms of the native packets as follows $p' = \sum_j (r_j \sum_i c_{ji} p_i) = \sum_i (\sum_j r_j c_{ji}) p_i$, i.e., it is a linear combination of the native packets themselves.

(d) Code Vector of a Packet: This is the vector of coefficients that describes how to derive the coded packet from the native packets. More precisely, if the coded packet is expressed as $p' = \sum_i c_i p_i$, where c_i is a number and the p_i's are native packets, then the code vector of this packet is $\vec{c} = (c_1, \ldots, c_i, \ldots, c_K)$. A native packet, p_i, is a special case of a coded packet, for which the element c_i is 1 and the remaining elements are 0's. When a new coded packet is formed by linearly mixing other coded packets, $p' = \sum_j r_j p'_j$, where r_j is a random number, its code vector is obtained as $\vec{c} = \sum_i r_i \vec{c_i}$, where $\vec{c_i}$ is the code vector of packet p'_i.

(e) Linearly Independent Packets: We say that a set of packets are linearly independent if their corresponding code vectors are linearly independent.

(f) Innovative Packet: We say that a node has received an innovative packet if the new packet is linearly independent from the previous packets the node has received. Innovative packets contain new useful information as opposed to linearly dependent packets whose information can be extracted from previously received packets. Packets can either be innovative or non-innovative, and non-innovative packets can be safely discarded.

(g) Closeness to Destination: We define the distance from node x to node y as the expected number of transmissions required to deliver a packet from the first node to the second. This distance metric has been widely used in prior wireless routing protocols. In particular, the ETX metric estimates the above distance [10]. Similar to other routing protocols, MORE uses periodic pings to measure the average delivery probability between any pair of nodes and uses these estimates in a form similar to ETX to measure closeness to destination.

(k) Downstream/Upstream: If node x is closer to the destination than node y, we say that x is downstream of y, and y is upstream of x.

[2]All operations are done in finite fields of size 2^8 allowing us to operate on bytes instead of arbitrary blocks.

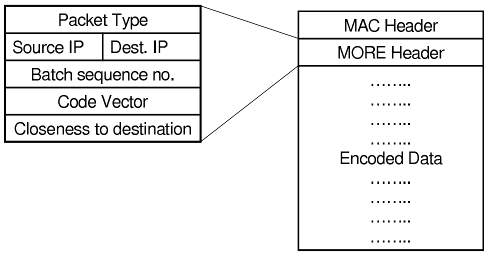


Figure 3—MORE packet header format.

4 MORE In a Nutshell

MORE is an opportunistic routing protocol. Its design is based on the theory of network coding [2]. Routers code packets going to the same destination together and forward these coded versions. The destination decodes and recovers the original packets.

MORE's current design targets stationary multi-hop wireless networks, such as Roofnet and community wireless networks [1, 29, 3]. Nodes in such networks are normally not limited by memory or processing power. We show in §7 that coding and decoding can be done easily on the kind of machines typically used in this environment.

For simplicity, we explain MORE using a single flow. The description extends naturally to multiple flows.

(a) The sender: In MORE, the source sends batches of K packets, where K may vary from one batch to another. When the 802.11 MAC permits, the source creates a random linear combination of the K packets in the current batch and broadcasts the coded packet. Each MORE packet contains the code vector that describes its contents with respect to native packets, the batch sequence number, the distance of the transmitter from the destination, and a type field that identifies data packets from acks (see Fig. 3). The sender keeps transmitting coded packets from the current batch until it receives an ack from the destination acknowledging that it has received K linearly independent combinations, and thus is able to decode the original packets in the batch. At this time, the sender can move on to the next batch.

(b) A forwarder: Nodes listen to all transmissions. When a node hears a packet, it checks whether the packet is innovative, i.e., linearly independent from the ones it has previously received from this batch. This check can be done using simple algebra (Gaussian Elimination [18]). If the packet is not innovative, then it does not add any new information beyond what the node has already heard, and thus it is ignored. Otherwise the node buffers the packet with previously received packets from the same batch. Next, the node checks whether it is closer to the destination than the previous hop of the overheard packet. If it is, then the arrival of this new packet triggers the node to broadcast a linear combination of packets from the same batch as the recently received packet.

(c) The destination: For each packet it receives, the destination checks whether the packet is innovative, and discards non-innovative packets. Once the destination has K innovative packets, it can decode the original K packets using simple matrix inversion:

$$\begin{pmatrix} p_1 \\ \vdots \\ p_K \end{pmatrix} = \begin{pmatrix} c_{11} & \dots & c_{1K} \\ \vdots & \ddots & \\ c_{K1} & \dots & c_{KK} \end{pmatrix}^{-1} \begin{pmatrix} p'_1 \\ \vdots \\ p'_K \end{pmatrix},$$

where, p_i is a native packet, and p'_i is a coded packet whose code vector is $\vec{c}_i = c_{i1}, \dots, c_{iK}$.

Additionally, once the destination receives enough innovative packets to decode the batch, it sends an acknowledgment along the shortest path toward the source. The ack causes the source to stop transmitting any further packets from that batch. As a result, the forwarders stop receiving innovative packets from the batch and consequently stop transmitting linear combinations of that batch. Eventually, the batch will timeout and be purged from the memory of the forwarders.

5 PRACTICAL ISSUES

In §4, we have described the general design of MORE. But for the protocol to be practical, MORE has to address 3 additional challenges, which we discuss in detail below.

5.1 Fast Network Coding

Network coding, implemented naively, can be expensive. As outlined above, intermediate nodes forward linear combinations of the packets they receive. Combining N packets of size S requires NS multiplications and additions. Due to the broadcast nature of the wireless medium, intermediate nodes could receive many packets from the same batch. If a router codes all these packets together, the coding cost may be overwhelming, creating a CPU bottleneck.

MORE employs three techniques to produce efficient coding and ensure the routers can easily support very high bit rates.

- **Code only Innovative Packets:** The coding cost scales with the number of packets coded together. But coding non-innovative packets is not useful; they do not add any information content. Hence, when a MORE forwarder receives a new packet, it checks if the packet is innovative, and throws away non-innovative packets. Since the number of innovative packets in any batch is bounded by the batch size K, discarding non-innovative packets bounds both the number of packets the forwarder buffers from any batch, and the number of packets combined together to produce a coded packet.
- **Operate on Code Vectors:** MORE checks whether packets are innovative using an efficient algorithm that operates only on code vectors. When a new packet is received, checking for innovativeness implies checking if the received packet is linearly independent of the set of packets from the same batch already present at the node. But it is not necessary to perform any operations on the data packet itself. The forwarder node simply checks if the code vectors are linearly independent.[3] The data in the packet itself is not touched; it is just stored in a pool to be used later when the node needs to forward a linear combination from the batch. Thus, operations on individual data bytes happen only occasionally at the time of coding or decoding, while checking for innovativeness, which occurs for every overheard packet, is fairly cheap.
- **Pre-Coding:** When the driver is ready to send a packet, the node has to generate a linear combination of the buffered packets and hand that coded packet to the wireless card. Linearly combining packets involves multiplying individual bytes in those packets, which could take hundreds of microseconds. MORE cannot simply create coded packets and make them available to the MAC for forwarding by storing them in an output queue. If it did, it would run the risk of transmitting coded packets that do not contain the most recently received innovative information. On the other hand, creating these linear combinations once the MAC signals the availability of the medium inserts a delay before the transmission of every packet, and thus reduces the achievable throughput. To avoid this complication, MORE exploits the time when the medium is not available to pre-compute one linear combination, so that the coded packet is ready when the medium becomes available. However, if the node receives an innovative packet before the prepared packet is handed over to the driver, the packet is updated by multiplying the new packet with a random coefficient and adding it to the pre-coded packet. This consumes significantly less time than creating a coded packet from scratch, which requires multiplying and adding all stored packets form that batch.

5.2 How Much to Send on Each Path?

Opportunistic routing uses multiple paths to forward packets instead of restricting itself to the shortest path. Not

[3]Checking independence uses basic algebraic operations. The forwarder maintains a matrix that is formed via *incremental* Gaussian elimination of code vectors of the packets in the same batch. It is a triangular matrix of K rows, but some of these rows are empty (i.e., all zero). To check if the code vector of the newly received packet is linearly independent, the non-empty rows are multiplied by appropriate coefficients and added to it in order so that consecutive elements of the vector become 0. If the vector is linearly independent, then one element will not be zeroed due to a missing row, and the modified vector can be added to the matrix in that empty slot. This process requires only NK multiplications per packet, where N is the number of non-empty rows. A similar technique is used for decoding, requiring $2NS$ multiplications per packet in order to obtain the identity matrix at the end, where S is the packet size.

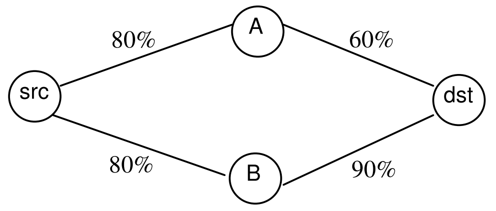


Figure 4—Candidate forwarders should not send the same amount of traffic. The figure shows a wireless topology, where the edges are labeled by their deliverly probability. Though nodes *A* and *B* receive on average the same number of packets, node *B* has a much better path to the destination, and thus should forward more traffic than node *A*.

all paths, however, are equally useful. Consider the scenario in Fig. 4. Nodes A and B receive on average the same number of packets. But since the link from A to the destination is much worse than the link from B to the destination, it would be wasteful to have A forward as many packets as B. Node B is in a much better position to forward most of the flow to the destination. Node A should just supplement the extra information that was received by A but not by B.

Note that the problem outlined above is a general multipath routing problem; it is *not* created by network coding. Indeed it is analogous to the shortest path problem in single path routing. In the shortest path problem, one would like to find the path that minimizes some cost function. In the multipath case, one would like to divide the traffic among all paths connecting the source to the destination as to minimize some cost function. Further, in both cases the natural choice of cost function is the average number of transmissions required per packet.

5.2.1 Minimizing the number of transmissions

MORE addresses the above problem using an optimization framework. It divides the traffic flow between the paths connecting the source and destination as to minimize the expected number of transmissions per packet. Formally, consider the delivery of a single packet from source to destination. Let N be the number of forwarders, z_i be the number of transmissions made by candidate forwarder, i, and x_{ij} is the optimal number of packets that the routing should deliver from i to j. We would like to minimize the total number of transmissions made by all forwarders subject to the constraint of delivering the packet, i.e.:

$$\arg\min \sum_{i\in N} z_i, \qquad (1)$$

Subject to the constraints:

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} 1 & \text{if } i \text{ is the source,} \\ -1 & \text{if } i \text{ is the destination,} \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \qquad (3)$$

$$z_i p(i,J) \geq \sum_{j\in J} x_{ij}. \qquad (4)$$

The first constraint, expressed in Eq. 2, refers to the conservation of information flow. Each forwarder sends out as much flow as it receives, while the source sends out 1 packet and the destination receives 1 packet. The second constraint, expressed in Eq. 3, just states that the traffic flow cannot be negative.

The third constraint embodies the opportunism of wireless networks, which we want to exploit to our advantage. In opportunistic routing, it is sufficient to ensure that when a node transmits a packet, the packet is received by at least one node closer to the destination. For example, in Fig. 4, when the source transmits a packet, it is sufficient to ensure that either A or B gets the packet. Let J be a set of nodes closer to the destination than i, and $p(i,J)$ the probability that at least one node in J receives a packet transmitted by i. The routing has to ensure that despite the lossy nature of the environment, i transmits enough to push the required traffic to downstream nodes, i.e., $z_i p(i,J) \geq \sum_{j\in J} x_{ij}$. Assuming that wireless losses are independent, $p(i,J)$ is simply $1 - \prod_{j\in J}(1-p_{ij})$, where p_{ij} is the delivery probability from i to j. Therefore the value can be obtained by probing individual links. Notice that in general $p(i,J) \geq p_{ij}$, for any node $j \in J$. In Fig. 4 for e.g., the probability that a packet transmitted by the source is received by at least one of the forwarders A or B is 0.96, which is greater than the delivery probability to either forwarder, i.e., 0.8. Thus, traditional routing would have picked one of the forwarders, achieving a smaller traffic flow.

The optimization problem laid out above has two important characteristics. First, each node can solve the optimization locally using any standard min-cost flow algorithm. Furthermore, given the assumption that the link losses are independent, the problem is no more expensive to solve than the shortest path problem. The node needs only the pairwise delivery probabilities to find the solution. Second, similarly to the shortest route computation, the solution of the optimization depends only on the topology and does not change based on traffic. The solution needs to be updated only when the average delivery probability between the nodes changes.

The above optimization setup is relatively standard in multipath wireless routing [24]. Our approach, however, differs from prior work in this area in that we do not use the solution of the optimization problem to find the optimal flow on each path and send traffic according to the optimal flow. Rather, MORE triggers packet forwarding by the reception of a packet (as is the case for current single path forwarding). A node uses packet reception from an upstream node as a signal that it should transmit; the number of transmissions the node should attempt is defined by a *triggering ratio*. Since different nodes have to transmit different numbers of packets, their triggering ratios differ. The triggering ratio for A, for example, should be lower than that for B, since B transmits more packets. We compute the triggering ratio for node i as,

$$T_i = \frac{z_i}{\sum_{j \in \text{upstream}(i)} z_j p(j,i)} \tag{5}$$

The above formula estimates the triggering ratio as a ratio of the number of expected transmissions a node is supposed to make versus the number of expected packets it will receive from upstream nodes. The triggering ratio is therefore an estimate of how aggressive a candidate forwarder should be in attempting to send a linear combination of a recently received innovative packet.

5.2.2 *Example*

Let us explore how the above algorithm works in the topology in Fig. 4. Nodes A and B solve the optimization problem locally using their knowledge of the pairwise delivery probabilities. The solution of the optimization for the simple topology in Fig. 4 is: $z_A = 0.28$, $z_B = 0.93$, and $z_{src} = 1.04$. Note that the source has to perform slightly more than one transmission per packet to make up for wireless losses. Next, each forwarder uses Eq. 5 to compute its triggering ratio, i.e., the average number of transmissions it should make for each packet it overhears. In this scenario, both forwarders receive on average 0.83 transmissions from *src*, therefore $T_A = 0.34$ and $T_B = 1.12$. Thus, as expected node B would forward most of the packets. Every innovative packet that A hears from the source increase A'ss credit counter by 0.34. Once the credit counter is larger than 1, node A broadcasts a random linear combination of the packets it has overheard, and decrements its counter. B does the same but it increases its credit counter by 1.12 for each innovative packet it receives.

5.2.3 *Pruning*

MORE 's solution to the linear optimization above might include forwarders that make very few transmissions (z_i is very small), and thus have very little contribution to the routing. In a dense network, we might have a large number of such low contribution forwarders. Since, the overhead of channel contention increases with the number of forwarders, it is useful to prune such nodes. MORE prunes forwarders that are expected to perform less than 10% of all the transmissions for the batch (more precisely, it prunes nodes whose $z_i < 0.1 \sum_{j \in N} z_j$).

5.2.4 *Sensitivity to delivery probability measurements*

Clearly, the paths that MORE takes and the amount of traffic it pushes on each of them will be affected by the measured delivery probabilities. Thus, the accuracy of the estimate of the delivery probability between a pair of nodes affects the optimality of our solution. Since the delivery probability in a wireless environment is dynamic and hard to accurately measure, any practical implementation of MORE will not be optimal. Such loss of optimality is the norm for any practical implementation of a wireless routing protocol. Consider current wireless routing protocols that try to send along the shortest path, i.e., the path that requires the minimum number of transmissions [5]. These protocols suffer from the same inaccuracy because they rely on measurements of pair-wise delivery probability, which are inaccurate. But in all cases, using these best estimates of delivery probability produces much better performance than ignoring them.

5.3 Stopping Rule

In MORE, traffic is pumped into the network by the source. The forwarders do not generate traffic unless they receive new packets. Thus, it is important to throttle the source's transmissions as soon as the destination has received enough packets to decode the batch. Thus, once the destination receives the K^{th} innovative packet, and before fully decoding the batch, it sends an acknowledgment to the source. To expedite the delivery of acks, they are sent on the shortest path from destination to source. Furthermore, acks are given priority over data packets at all nodes and are reliably delivered using local retransmission at each hop.

When the sender receives an acknowledgment for the current batch, it stops forwarding packets from that batch. If the transfer is not complete yet, the sender proceeds to transmit packets from the next batch.

The forwarders are triggered by the arrival of new packets, and thus stop transmitting packets from a particular batch once the sender stops doing so. Eventually the batch will timeout and be flushed from memory. Additionally, forwarders that hear the ack while it is being transmitted towards the sender immediately stop transmitting packets from that batch and purge it from their memory. Finally, the arrival of a new batch from the sender causes a forwarder to flush all buffered packets with batch sequence numbers lower than the active batch.

5.4 Where does MORE fit in?

MORE 's performance gains are highest for medium to long file transfers. For short transfers smaller than even 16 packets, gains will be insignificant. Hence for such transfers, traditional shortest path routing can be used, while for any transfer greater than 16 packets, MORE can be used. Note that MORE can co-exist with traditional routing, indeed MORE uses traditional routing to send its ACKs to the source.

6 Experimental Environment

(a) Testbed Characteristics: We have a 25-node wireless testbed that spans three floors in our building connected via

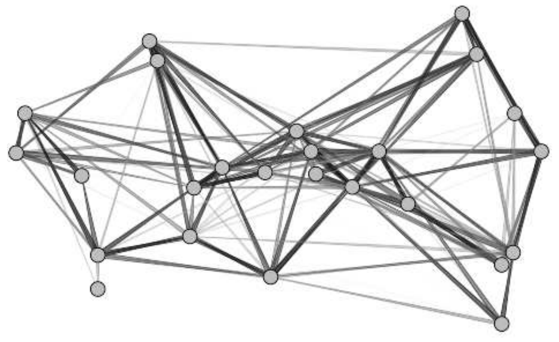


Figure 5—The topology of the testbed. The width and opacity of a link reflect its quality.

Operation	Avg. Time [μs]	Std. Dev. [μs]
Independence check	6.0	1.5
Coding at the source	180	10
Decoding	180	15

Table 1—Computational cost of packet operations in MORE. Note that the coding cost is highest at the source because it has to code all K packets together. The coding cost at a forwarder depends on the number of innovative packets it has received, and is always bounded by the coding cost at the source.

an open lounge. Paths between nodes are between 1 and 5 hops in length, and the loss rates of links on these paths range between 0 and 60%.

(b) 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 18 dBm, and operate in the 802.11 ad hoc mode, with RTS/CTS disabled.

(c) Software: Nodes in the testbed run Linux, the Click toolkit [19] and a the Roofnet software package [1]. Our implementation runs as a user space daemon on Linux, which sends and receives raw 802.11 frames from the wireless device using a libpcap-like interface.

(d) Compared Protocols: Ideally one would like to compare MORE with ExOR [6], but the code for ExOR is currently not available. Comparison with ExOR's numbers are also not possible, since throughput is highly dependent on the topology used, but MORE 's throughputs and gains are in the same range as ExOR. Thus, we compare MORE with Srcr [5], 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. The weights are assigned based on the ETX metric [5], which is an estimate of the number of transmission required to successfully transmit a 1500-byte packet on that link, including the expected number of MAC level retransmissions. The protocol also source-routes the packets to avoid routing loops when link metrics change.

(e) Conducted Experiments: Each of our experiments involves a run of Srcr immediately followed by a run of MORE, between the same source destination pair. We leverage the ETX implementation provided with the Roofnet Software to measure the link delivery probability. Before running an experiment, we run the ETX measurement module for 2 minutes to compute pair-wise delivery probabilities and the corresponding ETX metric. These measurements are then fed to both Srcr and MORE, and used by the two routing protocols for their route selection. Each node then runs the Srcr (Roofnet) routing protocol for 2 minutes and tries to transfer as many packets as possible between the selected node pair(s). MORE runs immediately afterwards and tries to transfer as many packets as possible.

Unless stated differently, the batch size of MORE is set to $K = 32$ packets. The packet size in both MORE and Srcr experiments is 1500B. The queue size at Srcr routers is 50 packets. In contrast, MORE does not use queues; it buffers at most K packets from each active batch.

Finally, most experiments are performed over 802.11b with a bit-rate of 5.5Mb/s. In §*7.2.3*, we allow traditional routing (i.e., Srcr) to exploit the auto-rate feature. For MORE, the concept of auto-rate is not meaningful because MORE broadcasts every transmission to many potential receivers. Thus, we compare Srcr with auto-rate to MORE with a fixed bit-rate of 11Mb/s.

7 IMPLEMENTATION RESULTS

This section presents the results of running MORE in a 25-node wireless testbed. It shows that MORE substantially improves wireless throughput.

7.1 Efficient Coding

Since network coding requires multiplying individual data bytes, it is often a concern whether it could create a CPU bottleneck. MORE provides an efficient implementation of network coding that bounds the number of packets coded together and checks independence by processing code vectors. The cost of coding/decoding packets is incurred primarily when every byte of a packet has to be multiplied with a random number (in a finite field of size 2^8). To optimize this operation, our implementation reduces the cost by using a 64KB lookup-table. The lookup table caches results of all possible multiplications, hence multiplying any byte of a packet with a random number is simply a fast lookup.

Table 6 summarizes the computational cost of using network-coding in MORE. The measurements are taken on a Pentium 4 machine with a 3.2GHz CPU and 512KB of cache. The micro benchmarks indicate that coding and decoding are equally costly. They require on average K finite-field multiplications per byte, where K is the batch size. This

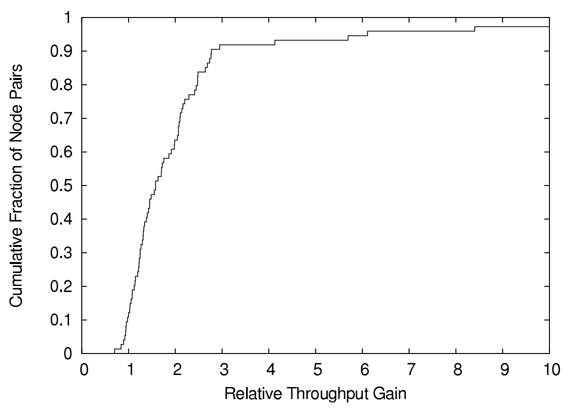


Figure 6—Cumulative distribution of the throughput gain of MORE when compared to Srcr. MORE achieves a median throughput gain of 60%, while some source-destination pairs show as much as $12x$ increase in throughput.

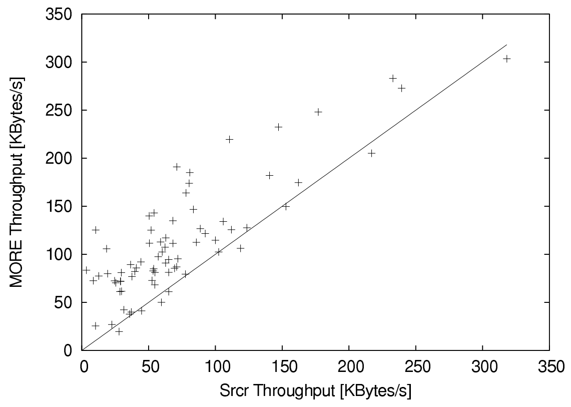


Figure 7—Scatter plot of average throughput attained by MORE and Srcr. Each point represents the throughput of a particular source destination pair. Points above the 45-degree line indicate improvement with MORE. In comparison to Srcr, MORE either improves the flow's throughput or does not affect it. Node pairs which have low absolute throughput with Srcr show the maximum gain with MORE .

ties the choice of K with the maximum achievable throughput. In our setting $K = 32$ and decoding takes on average $180\mu s$ per 1500B packet. This limits the effective throughput to 66 Mb/s, which is much higher than the bit rate of current wireless mesh networks.

7.2 MORE Throughput Gains

We would like to examine whether MORE can effectively exploit opportunistic receptions to improve the throughput. We define the **throughput gain** as the ratio of the throughput under MORE to the throughput under traditional routing, i.e. Srcr, for the same topology and traffic demands.

Fig. 6 plots the CDF of the throughput gain MORE has over Srcr. The CDF is computed over 75 randomly selected

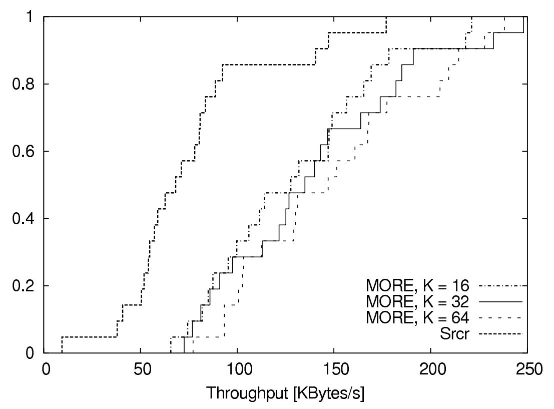


Figure 8—Impact of different batch sizes on MORE's throughput. The figure shows the CDF of the throughput taken over 20 random node pairs. Though larger batch sizes slightly improve the throughput, MORE is relatively insensitive to batch size.

node pairs. The figure shows that the median throughput gain achieved with MORE is about 1.6 –i.e., for our topology, MORE increases the throughput in half of the cases by at least 60%, when compared to state-of-the-art best path routing. It is worth noting that the maximum throughput improvement can be much more than the median. In particular a few node pairs experienced a throughput improvement of $5x$ to $12x$.

We try to identify the scenarios in which MORE is particularly useful–i.e., when should one expect to see a huge throughput gain? Fig. 7 shows the scatter plot for the throughputs achieved under Srcr and MORE for the same source-destination pair. Points on the 45-degree line have the same throughput under both schemes. The figure shows that for the vast majority of source-destination pairs in our testbed, MORE improves the throughput. Even when it does not improve the throughput, using MORE does not really hurt the performance. Also, the figure reveals that the high gain happens at low absolute throughput. This is expected and consistent with the experiments in [6]. A source-destination pair that achieves low throughout under Srcr does not have any special path with substantially better quality. There are usually many low-quality paths connecting the two nodes. By using the combined capacity of all these low-quality paths, MORE manages to greatly boost the throughput of such source-destination pairs. On the other hand, when the best path has very high link quality, opportunistic receptions by nodes not on the best path is of little value.

7.2.1 Batch Size

We picked $K = 32$ as default batch size. However, as shown in Fig. 8, MORE is relatively insensitive to the exact value of batch size. The figure examines the throughput of MORE for batch sizes of 16, 32, and 64, and compares it against the throughput achieved with traditional routing.

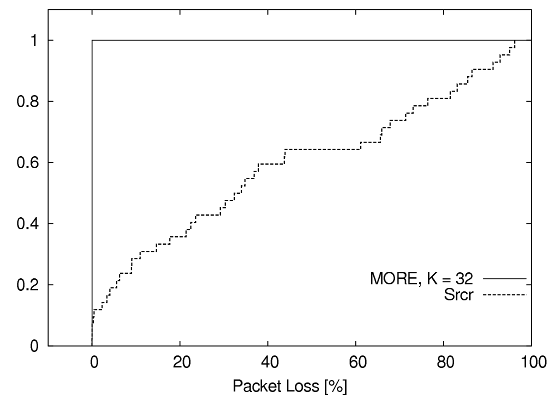


Figure 9—Comparison of end-to-end reliability for MORE and Srcr. The figure shows the CDF of loss rate taken over various source destination pairs. MORE is 100% reliable, Srcr on the other hand exhibits loss rates ranging from 0 to 92%. MORE therefore achieves high throughput as well as perfect reliability.

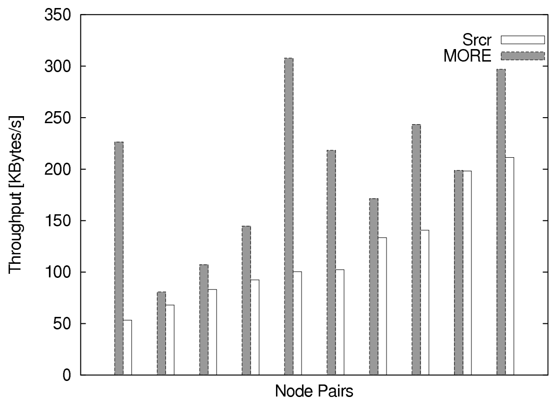


Figure 10—Comparison of throughput achieved by MORE running at 11 Mb/s and Srcr using automatic bit-rate selection, for 10 randomly selected source-destination pairs. MORE outperforms traditional single path routing even when we activate 802.11 smart rate selection to improve the quality of the links.

The figure shows that a larger batch size slightly increases MORE's throughput. But the differences are insignificant. MORE may get a few spurious transmissions between the time the destination decodes a batch and when the source and forwarders stop transmitting packets from that batch. A bigger batch size allows MORE to amortize the cost of these spurious transmissions over a larger number of packets, increasing the overall throughput.

Insensitivity to a range of batch sizes allows MORE to vary the batch size to accommodate different transfer sizes. We expect that for any transfer size larger than 7-10 packets (i.e., a batch larger than 7-10 packets), MORE will show significant advantages. Shorter transfers can be sent using standard single path routing.[4] Also, note that the larger the batch size, the more costly the coding/decoding. Thus, depending on the wireless bit-rate, there will be a bound on how big the batch can be. Transfers larger than the maximum batch size have to be sent using multiple batches.

7.2.2 *Reliability*

MORE ensures reliable delivery of packets; it keeps transmitting until the destination can decode. This is a significant departure from traditional routing which assumes that reliability will be ensured by end-host retransmissions. MORE's reliability is both a necessity and a feature. It is a necessity because network coding does not allow the destination to decode partial information in a batch. In many cases, it is a feature because wireless errors can cause a drop probability too high even for applications that do not require strict reliability. In traditional routing, each forwarder tries to retransmit the packet a fixed number of times at the MAC layer, if the packet still cannot get through it is discarded. Such lost packets have to be recovered via end-to-end retransmissions; these are quite expensive in wireless.

Figure 9 plots the CDF of the loss rate for both MORE and Srcr. The MAC level retransmissions for the Srcr experiments are set to 11. Despite of the large number of MAC retries, traditional routing shows significant packet loss, 60% of the node pairs show loss rates ranging from 1% to 37%, and 20% of the node pairs experience a loss rate larger than 80%. The high loss rates expressed by Srcr comply with prior results [5]; they are explained by Srcr's tendency to pick links with relatively high loss rate in order to reduce the number of hop,s and thus improve the overall throughput. MORE shows no end-to-end loss for all node pairs. Note that for path with error rates larger than 20%, TCP transmissions are likely to halt. In contrast, MORE can cope with such lossy environments and provide reliable data transmission. MORE therefore achieves higher throughput gains *as well as high reliability*.

7.2.3 *Autorate selection*

Wireless networks use automatic rate selection [4] for each link to maximize throughput. MORE on the other hand does not have the concept of a link, it uses a single bit-rate for all nodes. We compare the performance of MORE with traditional routing using automatic bit-rate control. We compare it against a state of the art bit-rate selection algorithm present in the Madwifi [25] driver.

Figure 10 compares the throughputs obtained by Srcr with automatic rate selection and MORE for 10 randomly selected node pairs. MORE achieves better throughput even when traditional routing is combined with automatic rate selection. Paths with low absolute throughput in traditional routing once again show the largest gains. Such paths have low quality links irrespective of the bit-rate used, therefore auto-rate selection does not help these paths.

[4]MORE benignly co-exists with shortest path routing. Indeed, MORE uses shortest path routing to send batch acks.

8 Related Work

Related work spans the following two areas.
(a) Network Coding: 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 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 [21]. Koetter and Médard [18] showed that network coding can be cast into an algebraic framework and polynomial time matrix multiplication and inversion algorithms can be used encoding and decoding. Ho et al. extended these results to random codes [13]. Some recent work has studied network coding in the wireless environment [11, 28]. In particular, Lun et al. have studied network coding in the presence of omni-directional antennae and shown that the problem of minimizing the communication cost can be formulated as a linear program and solved in a distributed manner [24]. All of this work, however, is mainly theoretical; Further, most of it assumes multicast traffic, known sender and receivers, and smooth traffic. In contrast, this paper focuses on a practical implementation and measured throughput improvements.

The closest work to ours is a recent paper by Katti et al. [17] that shows via simulation and a prototype implementation that having the routers intelligently XOR packets going to different destinations substantially improves network throughput. Our work provides a complementary perspective on wireless network coding; it shows that coding packets going to the same destination also greatly increases the overall throughput. Future work should explore the potential of combining the two schemes to further boost network throughput.
(b) Diversity in wireless networks: Opportunistic routing has been introduced by Biswas and Morris, whose paper explains the potential throughput increase and proposes the ExOR protocol as a means to achieve it [6]. Our work builds on this foundation but adopts a fundamentally different approach; It shows how to employ network coding to provide a flexible and robust approach to opportunistic routing.

A number of proposals exist for intelligently picking next hops in wireless networks. Many of them rely on signal strength measurements as an indicator of the quality of the link. Larsson [20] uses RTS/CTS packets to measure received signal strength. Nodes that receive RTS packets reply with their signal strength in the CTS packets. The sender picks the node with the best signal strength as the next hop. CTS packets could collide though, hence modifications have been proposed [15] to impose an order on the transmission of the nodes' replies to RTS packets. Other proposals [9, 31] use historical measurements or geographical distances as an approximation of link quality. The above proposals rely on measurements to pick next hops before a packet is transmitted, MORE picks forwarders after the packets have been received at the forwarders.

Cooperative diversity [14] techniques use opportunistic receptions in wireless networks to provide higher diversity gains. Essentially different nodes receiving the transmission are used as multiple antennas to retransmit the same bits. This leads to duplicate transmissions, which is a problem in dense networks. Further it assumes orthogonal channels or time division multiplexing, which are unrealistic assumptions in practice.

9 Conclusion

Opportunistic routing and network coding are two powerful ideas which may at first sight appear unrelated. Our work combines these ideas in a natural fashion to provide opportunistic routing *without* node coordination. We design a practical system, MORE, that plugs random linear network coding into the current network stack, exploits the opportunism inherent in the wireless medium, and provides significant performance gains. Field tests on a 25-node wireless testbed show that the median throughput gain of MORE compared to traditional single-path routing is 1.6, while the maximum exceeds 10-fold.

References

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert 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*, July 2000.

[3] Bay area wireless user group. http://www.bawug.org.

[4] John Bicket. Bit-rate selection in wireless networks. *M.S. Thesis*, 2005.

[5] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *ACM MobiCom*, 2005.

[6] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. *ACM SIGCOMM*, 2005.

[7] J. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast, 2002.

[8] John W. Byers, Michael Luby, and Michael Mitzenmacher. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In *INFOCOM*, pages 275–283, New York, NY, March 1999. IEEE.

[9] Romit Roy Choudhury and Nitin H. Vaidya.

Mac-layer anycasting in ad hoc networks. *SIGCOMM Comput. Commun. Rev.*, 34(1), 2004.

[10] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM MobiCom*, San Diego, California, September 2003.

[11] 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.

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

[13] 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.

[14] D. Tse J. N. Laneman and G. Wornell. Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *IEEE Transactions on Information Theory*, December 2004.

[15] S. Jain and S. Das. Exploiting path diversity in the link layer in wireless ad-hoc networks. *IEEE WoWMoM Symposium*, 2005.

[16] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. 2001.

[17] S. Katti, D. Katabi, W. Hu, H. S. Rahul, and M. Médard. The importance of being opportunistic: Practical network coding for wireless environments. In *43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[18] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 2003.

[19] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, Aug 2000.

[20] Peter Larsson. Selection diversity forwarding in a multihop packet radio network with fading channel and capture. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):47–54, 2001.

[21] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, Feb 2003.

[22] Michael Luby. Tornado codes: Practical erasure codes based on random irregular graphs. In *RANDOM '98: Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, 1998.

[23] Desmond S Lun, Muriel Médard, Ralf Koetter, and Michelle Effros. Further results on coding for reliable communication over packet networks. In *IEEE International Symposium on Information Theory (ISIT 05)*, 2005.

[24] Desmond S. Lun, Niranjan Ratnakar, Ralf Koetter, Muriel Médard, Ebad Ahmed, and Hyunjoo Lee. Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding. In *IEEE INFOCOM*, 2005.

[25] Madwifi: Multiband atheros driver for wifi. http://madwifi.org.

[26] Allen K. Miu, Hari Balakrishnan, and Can E. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *11th ACM MOBICOM Conference*, Cologne, Germany, September 2005.

[27] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[28] 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*, October 2003.

[29] Seattle wireless. http://www.seattlewireless.net.

[30] G. Strang. Intoduction to linear algebra. *Wellesley Cambridge Press*, March 2003.

[31] M. Zorzi and R. Rao. Geographic random forwarding (geraf) for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, October 2003.