Trading Structure for Randomness in Wireless Opportunistic Routing

by

Szymon Chachulski

mgr inz., Warsaw University of Technology (2005)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007

© Massachusetts Institute of Technology 2007. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2007

Certified by .....
Dina Katabi
Assistant Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Trading Structure for Randomness in Wireless Opportunistic Routing

by

Szymon Chachulski

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Opportunistic routing is a recent technique that achieves high throughput in the face of lossy wireless links. The current opportunistic routing protocol, ExOR, ties the MAC with routing, imposing a strict schedule on routers' access to the medium. Although the scheduler delivers opportunistic gains, it eliminates the clean layering abstraction and misses some of the inherent features of the 802.11 MAC. In particular, it prevents spatial reuse and thus may underutilize the wireless medium.

This thesis presents MORE, a MAC-independent opportunistic routing protocol. MORE randomly mixes packets before forwarding them. This randomness ensures that routers that hear the same transmission do not forward the same packets. Thus, MORE needs no special scheduler to coordinate routers and can run directly on top of 802.11.

We analyze the theoretical gains provided by opportunistic routing and present the EOTX routing metric which minimizes the number of opportunistic transmissions to deliver a packet to its destination.

We implemented MORE in the Click modular router running on off-the-shelf PCs equipped with 802.11 (WiFi) wireless interfaces. Experimental results from a 20-node wireless testbed show that MORE's median unicast throughput is 20% higher than ExOR, and the gains rise to 50% over ExOR when there is a chance of spatial reuse.

Thesis Supervisor: Dina Katabi
Title: Assistant Professor

Acknowledgments

This thesis is the result of joint work done with Michael Jennings, Sachin Katti and Dina Katabi. I could not express enough thanks to Dina for her extraordinary support during these two years at MIT.

I am grateful to Mike and Sachin for the effort put into the project. I owe many thanks to Hariharan Rahul for his much appreciated help with our wireless testbed. I am especially indebted to Micah Brodsky for his insight on wireless communication and invaluable discussions on opportunistic routing. I would also like to thank the developers of Roofnet, Click and MadWifi, the great building blocks of our project.

I wish to thank my fellow lab mates for their great company; Arvind, David, Dan, Jacob, James, Jen, Krzysztof, Nate, Srikanth, Yang and Yuan. Magda and Michel deserve special credit for their support.

Finally, my biggest thanks go to Ewa for her patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless mesh networks are increasingly used for providing cheap Internet access everywhere [7, 4, 33]. City-wide WiFi networks, however, need to deal with poor link quality caused by urban structures and the many interferers including local WLANs. For example, half of the *operational* links in Roofnet [4] have a loss probability higher than 30%. Opportunistic routing has recently emerged as a mechanism for obtaining high throughput even when links are lossy [10]. Traditional routing chooses the nexthop before transmitting a packet; but, when link quality is poor, the probability the chosen nexthop receives the packet is low. In contrast, opportunistic routing allows *any* node that overhears the transmission and is closer to the destination to participate in forwarding the packet. Biswas and Morris have demonstrated that this more relaxed choice of nexthop significantly increases the throughput. They proposed the ExOR protocol as a means to achieve these gains [10].

Opportunistic routing, however, introduces a difficult challenge. Multiple nodes may hear a packet broadcast and unnecessarily forward the same packet. ExOR deals with this issue by tying the MAC to the routing, imposing a strict scheduler on routers' access to the medium. The scheduler goes in rounds. Forwarders transmit in order, and only one forwarder is allowed to transmit at any given time. The others listen to learn which packets were overheard by each node. Although the medium access scheduler delivers opportunistic throughput gains, it does so at the cost of losing some of the desirable features of the current 802.11 MAC. In particular, the scheduler prevents the forwarders from exploiting spatial reuse, even when multiple packets can be simultaneously received by their corresponding

receivers. Additionally, this highly structured approach to medium access makes the protocol hard to extend to alternate traffic types, particularly multicast, which is becoming increasing common with content distribution applications [1] and video broadcast [3, 2].

In contrast to ExOR's highly structured scheduler, this paper addresses the above challenge with randomness. We introduce MORE, MAC-independent Opportunistic Routing & Encoding. MORE randomly mixes packets before forwarding them. This ensures that routers that hear the same transmissions do not forward spurious packets. Indeed, the probability that such randomly coded packets are the same is proven to be exponentially low [16]. As a result, MORE does not need a special scheduler; it runs directly on top of 802.11.

The main contribution of MORE is its ability to deliver the opportunistic routing gains while maintaining the clean architectural abstraction between the routing and MAC layers. MORE is MAC-independent, and thus can enjoy the basic features available to today's MAC. Specifically, it achieves better unicast throughput by exploiting the spatial reuse available with 802.11.

We evaluate MORE in a 20-node indoor wireless testbed. Our implementation is in Linux and uses the Click toolkit [23] and the Roofnet software package [4]. Our results reveal the following findings.

- In our testbed, MORE's median unicast throughput is 20% higher than ExOR. For 4-hop flows where the last hop can exploit spatial reuse, MORE's throughput is 50% higher than ExOR's.
- In comparison with traditional routing, the gain in the median throughput of a MORE flow is 95%, and the maximum throughput gain exceeds $10x$.
- Finally, coding is not a deployment hurdle for mesh wireless networks. Our implementation can sustain a throughput of 44 Mb/s on low-end machines with Celeron 800MHz CPU and 128KiB of cache.

1.1 Motivating Example

MORE's design builds on the theory of network coding [5, 25, 16]. In this section, we use a toy example to explain the intuition underlying our approach and illustrate the synergy between opportunistic routing and network coding.

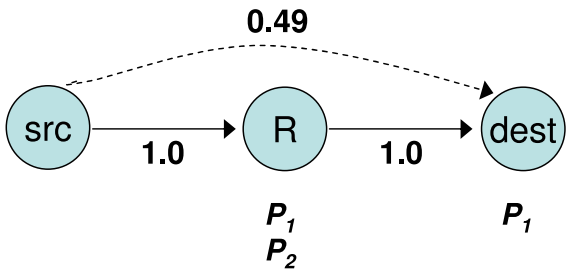


Figure 1-1: **Motivating Example.** The source sends 2 packets. The destination overhears p_2, while R receives both. R needs to forward just one packet but, without node-coordination, it may forward p_2, which is already known to the destination. With network coding, however, R does not need to know which packet the destination misses. R just sends the sum of the 2 packets $p_1 + p_2$. This coded packet allows the destination to retrieve the packet it misses independently of its identity. Once the destination receives the whole transfer (p_1 and p_2), it acks the transfer causing R to stop transmitting.

Consider the scenario in Fig. 1-1. Traditional routing predetermines the path before transmission. It sends traffic along the path "$src \rightarrow R \rightarrow dest$", which has the highest delivery probability. However, wireless is a broadcast medium. When a node transmits, there is always a chance that a node closer than the chosen nexthop to the destination overhears the packet. For example, assume the source sends 2 packets, p_1 and p_2. The nexthop, R, receives both, and the destination happens to overhear p_2. It would be a waste to have node R forward p_2 again to the destination. This observation has been noted in [10] and used to develop ExOR, an opportunistic routing protocol for mesh wireless networks.

ExOR, however, requires node coordination, which is hard to achieve in a large network. Consider again the example in the previous paragraph. R should forward only packet p_1 because the second packet has already been received by the destination; but, without consulting with the destination, R has no way of knowing which packet to transmit. The problem becomes harder in larger networks, where many nodes hear a transmitted packet. Opportunistic routing allows these nodes to participate in forwarding the heard packets. Without coordination, however, multiple nodes may unnecessarily forward the same packets, creating spurious transmissions. To deal with this issue, ExOR imposes a special scheduler on top of 802.11. The scheduler goes in rounds and reserves the medium for a single forwarder at any one time. The rest of the nodes listen to learn the packets overheard by each node. Due to this strict schedule, nodes farther away from the destination (which could potentially have transmitted at the same time as nodes close to the destination due to

spatial reuse), cannot, since they have to wait for the nodes close to the destination to finish transmitting. Hence the scheduler has the side effect of preventing a flow from exploiting spatial reuse.

Network coding offers an elegant solution to the above problem. In our example, the destination has overheard one of the transmitted packets, p_2, but node R is unaware of this fortunate reception. With network coding, node R naturally forwards linear combinations of the received packets. For example, R can send the sum $p_1 + p_2$. The destination retrieves the packet p_1 it misses by subtracting from the sum and acks the whole transfer. Thus, R need not know which packet the destination has overheard.

Indeed, the above works if R sends any random linear combination of the two packets instead of the sum. Thus, one can generalize the above approach. The source broadcasts its packets. Routers create random linear combinations of the packets they hear (i.e., $c_1p_1 + \ldots + c_np_n$, where c_i is a random coefficient). The destination sends an ack along the reverse path once it receives the whole transfer. This approach does not require node coordination and preserves spatial reuse.

The Challenges: To build a practical protocol that delivers the above benefits, we need to address a few challenges.

(a) How Many Packets to Send? In traditional best path routing, a node keeps sending a packet until the nexthop receives it or until it gives up. With opportunistic routing however, there is no particular nexthop; all nodes closer to the destination than the current transmitter can participate in forwarding the packet. How many transmissions are sufficient to ensure that at least one node closer to the destination has received the packet?

(b) Stop and Purge? With network coding, routers send linear combinations of the packets. Once the destination has heard enough such coded packets, it decodes and retrieves the file. We need to stop the sender as soon as the destination has received the transfer and purge the related data from the forwarders.

(c) Efficient Coding? Network coding optimizes for better utilization of the wireless medium, but coding requires the routers to multiply and add the data bytes in the packets. We need efficient coding and decoding strategies to prevent routers' CPU from becoming a bottleneck.

Chapter 2

Background and Related Work

In this chapter, we introduce the reader to the basic concepts and related work in wireless mesh networking, opportunistic routing and network coding.

2.1 Wireless Mesh Networks

The nodes in a wireless mesh network are most often stationary wireless routers equipped with omnidirectional antennae [13, 4, 9]. These routers can either connect to wired LANs or include access points for mobile clients. Moreover, the radio frequency band is unlicensed, so, in order to deploy such a network physically, one only needs to scatter the nodes over the covered area. All remaining configuration can be done in software. Things get a bit more complicated when directional antennae and multiple interfaces are used, but in this work we focus on the simplest deployments. The wireless interfaces are 802.11 commodity hardware, which make such networks affordable and proliferate.

The price for this simplicity is charged to the resulting link quality. A number of physical phenomena can cause bit errors and corrupt transmitted frames, even if no collision occurs [13, 4]. A failed transmission is wasted bandwidth not only for its originator, but also for all the nodes in the wireless channel vicinity that have deferred their transmissions in order to avoid collision. Our performance goal is thus to reduce the total number of transmissions needed to deliver the message across the network.

The loss rates can be prohibitively high even if the destination is just one hop away, and the problem is much aggravated when a packet is dropped after some number of hops. In such case, all past transmissions of this packet could go to waste. The standard solution

to avoid this inefficiency is to add retransmissions in the link-layer [17]. Upon successful decoding the intended receiver transmits back an acknowledgment frame (ACK). The original sender keeps retransmitting the data frame until it receives the ACK. This method, effectively, masks most losses from higher layers. Furthermore, although nodes other than the intended receiver can often successfully decode the data frame, they usually discard them. So the wireless link typically has a simple point-to-point abstraction.

2.1.1 Routing

Given the point-to-point links, the routing algorithm is responsible for picking the best path through the nodes to destination. A number of metrics has been proposed to determine the quality of a path. The most obvious one, borrowed directly from wired networks, is hop count. Intuitively, paths with fewer hops will require fewer transmissions, which in turn reduces delay and bandwidth consumption. However, unless precaution is taken, this metric prefers long hops, which is unfortunate, because link quality generally drops with length. Therefore, the result could be a reduced throughput.

To address this problem, DeCouto et al. propose the ETX metric, defined as the expected number of transmissions necessary to deliver one packet over the path [13]. The ETX of a path is then the sum of the ETX of each hop along it. This in turn, is inversely proportional to the loss rate of the wireless link. Thus, this metric accepts more hops if they are less lossy, but minimizes hop count if good links are available. For example, in Fig. 1-1 the ETX of the path $src \rightarrow R \rightarrow dst$ equals 2 and is smaller than ETX of $src \rightarrow dst$, which is $1/0.49 = 2.04$. Overall, if the loss estimates are accurate for each link, and the losses are independent across links, then ETX yields the expected transmission cost of each packet. In addition, for sake of accuracy, ETX accounts for the probability that the transmission is successfully decoded, but must be reattempted because the 802.11 ACK is lost.

It should be noted that wireless interfaces are often capable of modulating at different bit-rates, and so transmissions of the same-sized packet could take different amounts of time. In that case, during one lengthy transmission at a low bit-rate, a number of high bit-rate ones could be performed. Realizing that, Bicket et al. propose the ETT metric, an extension to ETX, defined as the expected transmission time. This is the metric that powers Srcr, the routing protocol used in Roofnet [9]. Other variants and extensions have also been proposed [14, 31].

(a) Skipping Hops (b) Multiple Independent Forwarders

Figure 2-1: **Benefits of Fortunate Receptions.** In (a), 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. In (b), each of the source's transmissions has many independent chances of being received by a node closer to the destination.

2.2 Opportunistic Routing

As described in the previous section, a widely employed practice in routing in wireless mesh networks is to conceal the underlying broadcast medium behind a point-to-point abstraction. A transmission is not successful unless the designated receiver is able to decode it. Failure to receive at one node, however, does not preclude a success at another. The traditional approach ignores any such receptions, which could turn out to be quite fortunate. To illustrate how could such receptions be beneficial, let us consider two simple scenarios taken from [10].

Suppose that the path used for routing is $src \rightarrow A \rightarrow B \rightarrow C \rightarrow dst$, as shown in Figure 2-1-a. When the source transmits a packet, there is a chance that some nodes further down the route successfully receive the packet. If that is the case, there would be no point for A to forward the packet again, as it would be redundant effort. Moreover, A could actually fail to decode, but other nodes succeed.[1] Rather than discarding the packet at the lucky, though unintended receivers, we can exploit these receptions to skip hops. Thus, a number of transmissions can be spared.

In the second scenario, when the source transmits, each of the 100 possible forwarders has an independent probability of successful reception. So, even if each one fails with probability 0.9, some other succeeds with probability $1 - 0.9^{100} > 0.9999$. If the forwarder is designated ahead of time, the source must therefore perform on average 10 transmissions

[1] Although the link quality decreases with distance, so the chance that a farther receiver succeeds is smaller than that of the nearer node, but a bit-error at A does not determine a bit-error at other nodes.

for each packet. But if any node that successfully receives could be used as a forwarder, this cost drops tenfold.

2.2.1 ExOR

What is necessary to gain from these fortunate receptions is to defer the choice of the next hop forwarder to after the outcome of reception decoding is known. This is the essence of opportunistic routing as proposed by Biswas and Morris in [10]. The proposed protocol, ExOR, aims to leverage all such events by enforcing the simple rule: *of all the nodes that were able to successfully decode the transmission, the one that is closest to the destination should forward it on.* Not only does it ensure that the progress of a packet in each transmission is maximized, but it also prevents redundant transmissions due to replication. In the scenario shown in Fig. 2-1-a, this rule guarantees maximum progress of the packet toward the destination, and in the second scenario from Fig. 2-1-b, it utilizes all candidate forwarders, in both cases preventing redundant replication.

Although simple to define, the rule is quite difficult to implement in a distributed network, without omniscient observers and centralized coordinators. To determine whether it should forward the packet it received, a node must at the very least know whether a better forwarder succeeded in decoding that packet. Reliably exchanging this information after every reception would likely lead to a collapse due to overhead, not mentioning the hardness of avoiding collisions in such a flood of announcements. Instead, ExOR schedules all nodes in order of increasing distance to the destination. For example the schedule in Fig. 2-1-a would be *dst*>*C*>*B*>*A*>*src*. In this way, before the turn of a particular node comes, all nodes closer to the destination get to transmit. This enables propagation of reception status via piggy-backing on data transmissions. But if packets get lost, so are the announcements. To alleviate this problem and amortize the overhead of scheduling, ExOR gathers packets into larger batches.

To better understand why this works, let us consider a snapshot of the schedule. When C is done forwarding the packets it is responsible for, it is the turn of node B. Each transmission from C included (in a bit-map in the header) information on each packet in the batch whether the packet had been received by C or any node closer to destination. If B received at least one of these, it knows what packets to forward: all that B received but none reported to be received by C.

The scheduling is troublesome. The standard 802.11 MAC is a CSMA/CA protocol [17] and does not support such a strict schedule, hence, in order to maintain high utilization, ExOR must rely on fragile timing estimates. Moreover, this scheduling is *one-at-a-time* and thus prevents two transmissions that could otherwise succeed at the same time due to spatial reuse. The nodes are forced to wait until the currently scheduled node has finished. This is adversely affecting the throughput.

2.3 Network Coding

Network coding is a method to improve bandwidth efficiency in networks. The core idea is that the forwarding nodes should merge data contained in distinct packets in such a way that allows for recovery at the destination. This is in contrast with the traditional scheme which treats each packet as a distinct object that must be delivered to the destination intact. As Ahlswede et al. established in their seminal paper, network coding allows one to surpass the capacity limitations of the traditional method and achieve the information flow capacity of the network [5]. Furthermore, as Li et al. shown in [25], it is possible to achieve this capacity using only linear functions when mixing packets together. In such case, each packet transmitted is some linear combination of other packets. Additionally, Ho et al. show that the above is true even when the routers pick random coefficients [16]. Researchers have extended the above results to a variety of areas including content distribution [15], secrecy [11, 18], distributed storage [19], and reliability in DTN [34].

The classical network coding theory is hardly practical. It often assumes that the packets flow synchronously through the network, so that nodes can simply combine the incoming packets as they arrive and apply a single mixing function. In contrast, real-world packet networks are asynchronous and subject to random losses and delays. MORE belongs to a category of *practical* (or *distributed*) network coding schemes, such as those described in [28] and [12]. It addresses this problem by randomness and buffering. The nodes store the packets in buffers as they receive them and form random linear combinations of them when transmitting. Then each packet is described by the coefficients of the linear combination with respect to the raw source packets. This allows decoding by solving a set of linear equations.

As a final note, we should point out that the proposed scheme is not the only way to

apply network coding to routing in multi-hop wireless networks. For example, Katti et al. proposed COPE [21], a protocol which mixes together packets of different flows crossing at a forwarder. This way a single transmission can deliver a number of packets to their respective next hop nodes, where they can be decoded. In contrast, MORE operates on a single flow, and does not decode the packets until they reach the destination.

Chapter 3

MAC-independent Opportunistic Routing and Encoding

In this chapter, we present the design and implementation of MORE, our MAC-independent opportunistic routing protocol, which can exploit fortunate receptions without cumbersome coordination.

3.1 MORE In a Nutshell

MORE is a routing protocol for stationary wireless meshes, such as Roofnet [4] and community wireless networks [33, 6]. Nodes in these networks are PCs with ample CPU and memory.

MORE sits below IP and above the 802.11 MAC. It provides *reliable* file transfer. It is particularly suitable for delivering files of medium to large size (i.e., 8 or more packets). For shorter files or control packets, we use standard best path routing (e.g., Srcr [9]), with which MORE benignly co-exists.

Table 3.1 defines the terms used in the rest of the paper.

3.1.1 Source

The source breaks up the file into batches of K packets, where K may vary from one batch to another. These K uncoded packets are called *native packets*. When the 802.11 MAC is ready to send, the source creates a random linear combination of the K native packets

Term	Definition
Native Packet	Uncoded packet
Coded Packet	Random linear combination of native or coded packets
Code Vector of a Coded Packet	The vector of co-efficients that describes how to derive the coded packet from the native packets. For a coded packet $p' = \sum c_i p_i$, where p_i's are the native packets, the code vector is $\vec{c} = (c_1, c_2, \ldots, c_K)$.
Innovative Packet	A packet is innovative to a node if it is linearly independent from its previously received packets.
Closer to destination	Node X is closer than node Y to the destination, if the best path from X to the destination has a lower ETX metric than that from Y.

Table 3.1: **Definitions used in the thesis.**

in the current batch and broadcasts the coded packet. In MORE, data packets are always coded. A *coded packet* is $p' = \sum_i c_i p_i$, where the c_i's are random coefficients picked by the node, and the p_i's are native packets from the same batch. We call $\vec{c} = (c_1, \ldots, c_i, \ldots, c_K)$ the packet's *code vector*. Thus, the code vector describes how to generate the coded packet from the native packets.

The sender attaches a MORE header to each data packet. The header reports the packet's code vector (which will be used in decoding), the batch ID, the source and destination IP addresses, and the list of nodes that could participate in forwarding the packet (Fig. 3-1). To compute the forwarder list, we leverage the ETX calculations [13]. Specifically, nodes periodically ping each other and estimate the delivery probability on each link. They use these probabilities to compute the ETX distance to the destination, which is the expected number of transmissions to deliver a packet from each node to the destination. The sender includes in the forwarder list nodes that are closer (in ETX metric) to the destination than itself, ordered according to their proximity to the destination.

The sender keeps transmitting coded packets from the current batch until the batch is acked by the destination, at which time, the sender proceeds to the next batch.

3.1.2 Forwarders

Nodes listen to all transmissions. When a node hears a packet, it checks whether it is in the packet's forwarder list. If so, the node checks whether the packet contains new information, in which case it is called an *innovative packet*. Technically speaking, a packet is innovative if

it is linearly independent from the packets the node has previously received from this batch. Checking for independence can be done using simple Algebra (Gaussian Elimination [22]). The node ignores non-innovative packets, and stores the innovative packets it receives from the current batch.

If the node is in the forwarder list, the arrival of this new packet triggers the node to broadcast a coded packet. To do so the node creates a random linear combination of the coded packets it has heard from the same batch and broadcasts it. Note that *a linear combination of coded packets is also a linear combination of the corresponding native packets.* In particular, assume that the forwarder has heard coded packets of the form $p'_j = \sum_i c_{ji} p_i$, where p_i is a native packet. It linearly combines these coded packets to create more coded packets as follows: $p'' = \sum_j r_j p'_j$, where r_j's are random numbers. The resulting coded packet p'' can be expressed in terms of the native packets as follows $p'' = \sum_j (r_j \sum_i c_{ji} p_i) = \sum_i (\sum_j r_j c_{ji}) p_i$; thus, it is a linear combination of the native packets themselves.

3.1.3 Destination

For each packet it receives, the destination checks whether the packet is innovative, i.e., it is linearly independent from previously received packets. The destination discards non-innovative packets because they do not contain new information. Once the destination receives K innovative packets, it decodes the whole batch (i.e., it obtains the native 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 $\vec{c_i} = c_{i1}, \dots, c_{iK}$. As soon as the destination decodes the batch, it sends an acknowledgment to the source to allow it to move to the next batch. ACKs are sent using best path routing, which is possible because MORE uses standard 802.11 and co-exists with shortest path routing. ACKs are also given priority over data packets at every node.

3.2 Practical Challenges

In §3.1, 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.

3.2.1 How Many Packets Does a Forwarder Send?

In traditional best path routing, a node keeps transmitting a packet until the nexthop receives it, or the number of transmissions exceeds a particular threshold, at which time the node gives up. In opportunistic routing, however, there is no particular nexthop; all nodes closer to the destination than the current transmitter are potential nexthops and may participate in forwarding the packet. How many transmissions are sufficient to ensure that *at least one* node closer to the destination has received the packet? This is an open question. Prior work has looked at a simplified and theoretical version of the problem that assumes smooth traffic rates and infinite wireless capacity [26]. In practice, however, traffic is bursty and the wireless capacity is far from infinite.

In this section, we provide a heuristic-based practical solution to the above problem. Our solution has the following desirable characteristics: 1) It has low complexity. 2) It is distributed. 3) It naturally integrates with 802.11 and preserves spatial reuse. 4) It is practical– i.e., it makes no assumptions of infinite capacity or traffic smoothness, and requires only the average loss rate of the links.

Practical Solution

Bandwidth is typically the scarcest resource in a wireless network. Thus, the natural approach to increase wireless throughput is to decrease the number of transmissions necessary to deliver a packet from the source to the destination [10, 13, 9]. Let the distance from a node, i, to the destination, d, be the expected number of transmissions to deliver a packet from i to d along the best path– i.e., node i's ETX [13]. We propose the following heuristic to route a packet from the source, s, to the destination, d: when a node transmits a packet, the node closest to the destination in ETX metric among those that receive the packet should forward it onward. The above heuristic reduces the expected number of transmissions needed to deliver the packet, and thus improves the overall throughput.

Formally, let N be the number of nodes in the network. For any two nodes, i and j, let

$i < j$ denote that node i is closer to the destination than node j, or said differently, i has a smaller ETX than j. Let ϵ_{ij} denote the loss probability in sending a packet from i to j. Let z_i be the expected number of transmissions that forwarder i must make to route one packet from the source, s, to the destination, d, when all nodes follow the above routing heuristic. In the following, we assume that wireless receptions at different nodes are independent, an assumption that is supported by prior measurements [32, 30].

We focus on delivering one packet from source to destination. Let us calculate the number of packets that a forwarder j must forward to deliver one packet from source, s to destination, d. The expected number of packets that j receives from nodes with higher ETX is $\sum_{i>j} z_i(1-\epsilon_{ij})$. For each packet j receives, j should forward it only if no node with lower ETX metric hears the packet. This happens with probability $\prod_{k<j} \epsilon_{ik}$. Thus, in expectation, the number of packets that j must forward, denoted by L_j, is:

$$L_j = \sum_{i>j} (z_i(1-\epsilon_{ij}) \prod_{k<j} \epsilon_{ik}). \tag{3.1}$$

Note that $L_s = 1$ because the source generates the packet.

Now, consider the expected number of transmissions a node j must make. j should transmit each packet until a node with lower ETX has received it. Thus, the number of transmissions that j makes for each packet it forwards is a geometric random variable with success probability $(1 - \prod_{k<j} \epsilon_{jk})$. This is the probability that some node with ETX lower than j receives the packet. Knowing the number of packets that j has to forward from Eq. (3.1), the expected number of transmissions that j must make is:

$$z_j = \frac{L_j}{(1 - \prod_{k<j} \epsilon_{jk})}. \tag{3.2}$$

(a) Low Complexity: The number of transmissions made by each node, the z_i's, can be computed via the following algorithm. We can ignore nodes whose ETX to the destination is greater than that of the source, since they are not useful in forwarding packets for this flow. Next, we order the nodes by increasing ETX from the destination d and relabel them according to this ordering, i.e., $d = 1$ and $s = n$. We begin at the source by setting $L_n = 1$, then compute Eqs. (3.1) and (3.2) from source progressing towards the destination. To reduce the complexity, we will compute the values incrementally. Consider L_j, as given by

Eq. (3.1). If we computed it in one shot, we would need to compute the product $\prod_{k<j} \epsilon_{ik}$ from scratch for each $i > j$. The idea is to instead compute and accumulate the contribution of node i to L_j's of all nodes j with lower ETX, so that each time we only need to make a small update to this product (denoted P in the algorithm).

Algorithm 1 Computing the number of transmissions each node makes to deliver a packet from source to destination, z_i's

```
for i = n ... 1 do
    L_i ← 0
L_n ← 1    {at source}
for i = n ... 2 do
    z_i ← L_i/(1 − ∏_{j<i} ε_ij)
    P ← 1
    for j = 2 ... i − 1 do
        {compute the contribution of i to L_j}
        P ← P × ε_{i(j−1)}    {here, P is ∏_{k<j} ε_ik}
        L_j ← L_j + z_i × P × (1 − ε_ij)
```

Alg. 1 requires $O(N^2)$ operations, where N is the number of nodes in the network. This is because the outer loop is executed at most n times and each iteration of the inner loop requires $O(n)$ operations, where n is bounded by the number of nodes in the network, N.

(b) Distributed Solution: Each node j can periodically measure the loss probabilities ϵ_{ij} for each of its neighbors via ping probes. These probabilities are distributed to other nodes in the network in a manner similar to link state protocols [9]. Each node can then build the network graph annotated with the link loss probabilities and compute Eq. (3.2) from the ϵ_{ij}'s using the algorithm above.

(c) Integrated with 802.11: A distributed low-complexity solution to the problem is not sufficient. The solution tells each node the value of z_i, i.e., the number of transmissions it needs to make for every packet sent by the source. But a forwarder cannot usually tell when the source has transmitted a new packet. In a large network, many forwarders are not in the source's range. Even those forwarders in the range of the source do not perfectly receive every transmission made by the source and thus cannot tell whether the source has sent a new packet. Said differently, the above assumes a special scheduler that tells each node when to transmit.

In practice, a router should be triggered to transmit only when it receives a packet, and should perform the transmission only when the 802.11 MAC permits. We leverage the preceding to compute how many transmissions each router needs to make for every packet it receives. Define the TX_credit of a node as the number of transmissions that a node should make for every packet it receives from a node farther from the destination in the ETX metric. For each packet sent from source to destination, node i receives $\sum_{j>i}(1-\epsilon_{ji})z_j$, where z_j is the number of transmissions made by node j and ϵ_{ji} is the loss probability from j to i, as before. Thus, the TX_credit of node i is:

$$\text{TX\_credit}_i = \frac{z_i}{\sum_{j>i} z_j(1-\epsilon_{ji})}. \tag{3.3}$$

Thus, in MORE, a forwarder node i keeps a `credit counter`. When node i receives a packet from a node upstream, it increments the counter by its TX_credit. When the 802.11 MAC allows the node to transmit, the node checks whether the counter is positive. If yes, the node creates a coded packet, broadcasts it, then decrements the counter. If the counter is negative, the node does not transmit. The ETX metric order ensures that there are no loops in crediting, which could lead to credit explosion.

Pruning

MORE's solution to the 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 $z_i < 0.1\sum_{j\in N} z_j$).

3.2.2 Stopping Rule

In MORE, traffic is pumped into the network by the source. The forwarders do not generate traffic unless they receive new packets. 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 ACK 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 ID's lower than the active batch.

3.2.3 Fast Network Coding

Network coding, implemented naively, can be expensive. As outlined above, the routers forward linear combinations of the packets they receive. Combining N packets of size S bytes requires NS multiplications and additions. Due to the broadcast nature of the wireless medium, routers 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 that ensure the routers can easily support high bit rates.

(a) Code only Innovative Packets: The coding cost scales with the number of packets coded together. Typically, network coding makes routers forward linear combinations of the received packets. Coding non-innovative packets, however, is not useful because 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 innovative packets are by definition linearly independent, 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. Discarding non-innovative packets is particularly important in wireless because the broadcast nature of the medium makes the number of received packets much larger than innovative packets.

(b) Operate on Code Vectors: When a new packet is received, checking for innovativeness implies checking whether the received packet is linearly independent of the set of packets from the same batch already stored at the node. Checking independence of all data bytes is very expensive. Fortunately, this is unnecessary. The forwarder node simply checks if the code vectors are linearly independent.

To amortize the cost over all packets, each node keeps code vectors of the packets in its buffer in a row echelon form. Specifically, they are stored in a triangular matrix M of K rows with some of the rows missing, thus for each stored row, the smallest index of a non-zero element is distinct. 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, as shown in the algorithm:

Algorithm 2 Checking for linear independence of vector u

```
for i = 1...K do
  if u[i] ≠ 0 then
    if M[i] exists then
      u ← u − M[i]u[i]
    else
      {admit the modified block into memory}
      M[i] ← u/u[i]
      return True {rank increased}
return False {discard packet}
```

This process requires only NK multiplications per packet, where N is the number of non-empty rows and is bounded by K. The decoder at the destination runs a similar algorithm for each received packet. The nodes are kept in *reduced* row echelon matrix, so the first non-zero element of each existing row is 1. Decoding requires $2NS$ multiplications per packet in order to obtain the identity matrix at the end, where S is the packet size.

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.

(c) Pre-Coding: When the wireless 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. This inserts significant delay before every transmission, decreasing the overall throughput.

To address this issue, MORE exploits the time when the wireless medium is unavailable to pre-compute one linear combination, so that a coded packet is ready when the medium becomes available. If the node receives an innovative packet before the prepared packet is handed over to the driver, the pre-coded packet is updated by multiplying the newly arrived packet with a random coefficient and adding it to the pre-coded packet. This approach achieves two important goals. On one hand, it ensures the transmitted coded packet contains information from all packets known to the node, including the most recent arrival. On the other hand, it avoids inserting a delay before each transmission.

3.3 Implementation Details

Finally, we put the various pieces together and explain the system details.

3.3.1 Packet Format

MORE inserts a variable length header in each packet, as shown in Fig. 3-1. The header starts with a few required fields that appear in every MORE packet. The type field distinguishes data packets, which carry coded information, from ACKs, which signal batch delivery. The header also contains the source and destination IP addresses and the flow ID. The last required field is the batch ID, which identifies the batch to which the packet belongs. The above is followed by a few optional fields. The code vector exists only in data packets and identifies the coefficients that generate the coded packet from the native packets in the batch. The list of forwarders has variable length and identifies all potential forwarders ordered according to their proximity to the source. For each forwarder, the packet also contains its TX_credit (see §3.2.1). Except for the code vector, all fields are initialized by the source and copied to the packets created by the forwarders. In contrast, the code vector is computed locally by each forwarder based on the random coefficients they

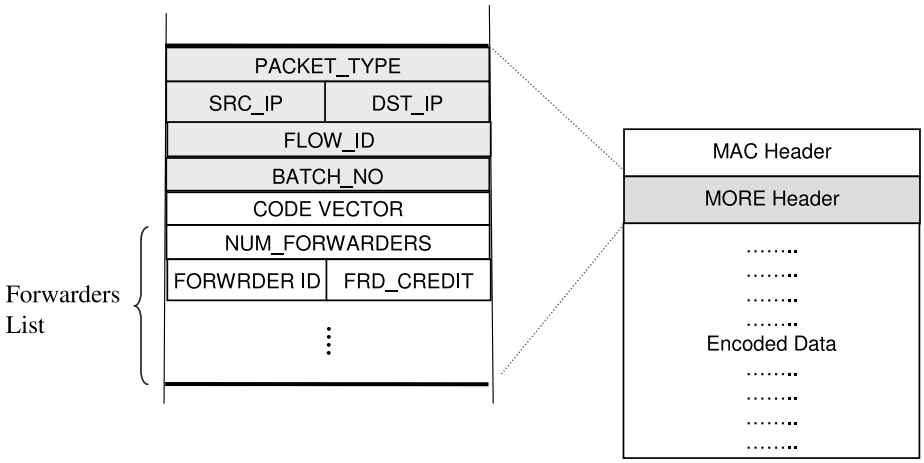


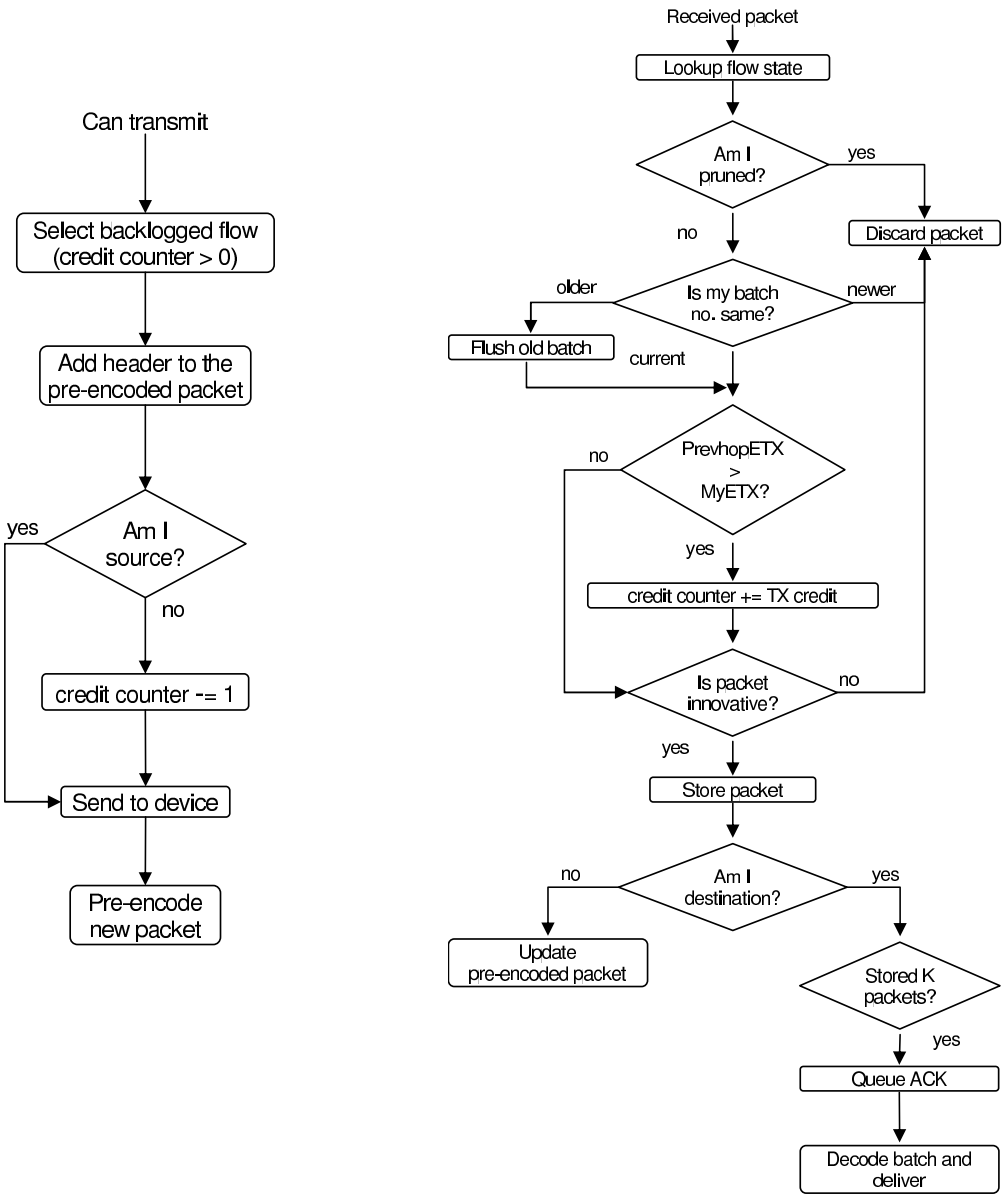
Figure 3-1: **MORE Header.** Grey fields are required while the white fields are optional. The packet type identifies batch ACKs from data packets.

picked for the packet.

3.3.2 Node State

Each MORE node maintains state for the flows it forwards. The per-flow state is initialized by the reception of the first packet from a flow that contains the node ID in the list of forwarders. The state is timed-out if no packets from the flow arrive for 5 minutes. The source keeps transmitting packets until the destination acks the last batch of the flow. These packets will re-initialize the state at the forwarder even if it is timed out prematurely. The per-flow state includes the following.

- The `batch buffer` stores the received innovative packets. Note that the number of innovative packets in a batch is bounded by the batch size K.
- The `current batch` variable identifies the most recent batch from the flow.
- The `forwarder list` contains the list of forwarders and their corresponding TX_credits, ordered according to their distance from the destination. The list is copied from one of the received packets, where it was initialized by the source.
- The `credit counter` tracks the transmission credit. For each packet arrival from a node with a higher ETX, the forwarder increments the counter by its corresponding TX_CREDIT, and decrements it 1 for each transmission. A forwarder transmits only when the counter is positive.



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

Figure 3-2: **MORE's Architecture.** The figure shows a flow chart of our MORE implementation.

3.3.3 Control Flow

Figure 3-2 shows the architecture of MORE. The control flow responds to packet reception and transmission opportunity signaled by the 802.11 driver.

On the sending side, the forwarder prepares a pre-coded packet for every backlogged flow to avoid delay when the MAC is ready for transmission. A flow is backlogged if it

has a positive `credit counter`. Whenever the MAC signals an opportunity to transmit, the node selects a backlogged flow by round-robin and pushes its pre-coded packet to the network interface. As soon as the transmission starts, a new packet is pre-coded for this flow and stored for future use. If the node is a forwarder, it decrements the flow's `credit counter`.

On the receiving side, when a packet arrives the node checks whether it is a forwarder by looking for its ID in the forwarder list in the header. If the node is a forwarder, it checks if the batch ID on the packet is the same as its `current batch`. If the batch ID in the packet is higher than the node's `current batch`, the node sets `current batch` to the more recent batch ID and flushes packets from older batches from its `batch buffer`. If the packet was transmitted from upstream, the node also increments its `credit counter` by its TX_credit. Next, the node performs a linear independence check to determine whether the packet is innovative. Innovative packets are added to the `batch buffer` while non-innovative packets are discarded.

Further processing depends on whether the node is the packet's final destination or just a forwarder. If the node is a forwarder, the pre-coded packet from this flow is updated by adding the recent packet multiplied by a random coefficient. In contrast, if the node is the destination of the flow, it checks whether it has received a full batch (i.e., K innovative packets). If so, it queues an ACK for the batch, decodes the native packets and pushes them to the upper layer.

3.3.4 ACK Processing

ACK packets are routed to the source along the shortest ETX path. ACKs are also prioritized over data packets and transferred reliably. In our implementation, when a transmission opportunity arises, a flow that has queued ACK is given priority, and the ACK packet is passed to the device. Unless the transmission succeeds (i.e., is acknowledged by the MAC of the nexthop) the ACK is queued again. In addition, all nodes that overhear a batch ACK update their `current batch` variable and flush packets from the acked batch from their `batch buffer`.

Chapter 4

Experimental Evaluation

We use measurements from a 20-node wireless testbed to evaluate MORE, compare it with both ExOR and traditional best path routing, and estimate its overhead. Our experiments reveal the following findings.

- MORE achieves 20% better median throughput than ExOR. In comparison with traditional routing, MORE almost doubles the median throughput, and the maximum throughput gain exceeds $10x$.
- MORE's throughput exceeds ExOR's mainly because of its ability to exploit spatial reuse. Focusing on flows that traverse paths with 25% chance of concurrent transmissions, we find that MORE's throughput is 50% higher than that of ExOR.
- MORE significantly eases the problem of dead spots. In particular, 90% of the flows achieve a throughput higher than 50 packets/second. In traditional routing the 10^{th} percentile is only 10 packets/second.
- MORE keeps its throughput gain over traditional routing even when the latter is allowed automatic rate selection.
- MORE is insensitive to the batch size and maintains large throughput gains with batch size as low as 8 packets.
- Finally, we estimate MORE's overhead. MORE stores the current batch from each flow. Our MORE implementation supports up to 44 Mb/s on low-end machines with Celeron 800MHz CPU and 128KiB of cache. Thus, MORE's overhead is reasonable for the environment it is designed for, namely stationary wireless meshes, such as Roofnet [4] and community wireless networks [33, 6].

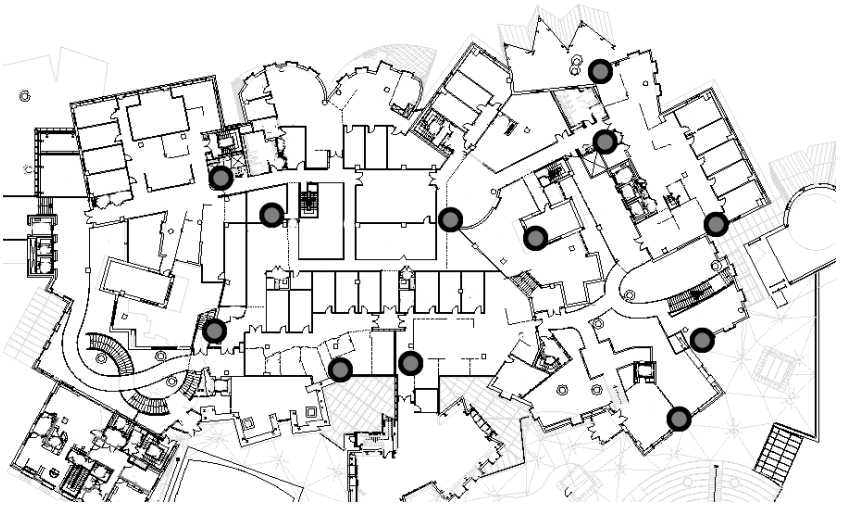


Figure 4-1: **One Floor of our Testbed.** Nodes' location on one floor of our 3-floor testbed.

We will make our code public including the finite field coding libraries.

4.1 Testbed

(a) Characteristics: We have a 20-node wireless testbed that spans three floors in our building connected via open lounges. The nodes of the testbed are distributed in several offices, passages, and lounges. Fig. 4-1 shows the locations of the nodes on one of the floors. Paths between nodes are 1–5 hops in length, and the loss rates of links on these paths vary between 0 and 60%, and averages to 27%.

(b) Hardware: Each node in the testbed is a PC equipped with a NETGEAR WAG311 wireless card attached to an omni-directional antenna. 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 [23] and the Roofnet software package [4]. Our implementation runs as a user space daemon on Linux. It sends and receives raw 802.11 frames from the wireless device using a libpcap-like interface.

4.1.1 Compared Protocols

We compare the following three protocols.

- *MORE* as explained in §3.3.

- *ExOR* [10], the current opportunistic routing protocol. Our ExOR code is provided by its authors.
- *Srcr* [9] which is a state-of-the-art best path routing protocol for wireless mesh networks. It uses Dijkstra's shortest path algorithm where link weights are assigned based on the ETX metric [13].

4.1.2 Setup

In each experiment, we run Srcr, MORE, and ExOR in sequence between the same source destination pairs. Each run transfers a 5 MByte file. We leverage the ETX implementation provided with the Roofnet Software to measure link delivery probabilities. Before running an experiment, we run the ETX measurement module for 10 minutes to compute pair-wise delivery probabilities and the corresponding ETX metric. These measurements are then fed to all three protocols, Srcr, MORE, and ExOR, and used for route selection.

Unless stated differently, the batch size for both MORE and ExOR is set to $K = 32$ packets. The packet size for all three protocols is 1500B. The queue size at Srcr routers is 50 packets. In contrast, MORE and ExOR do not use queues; they buffer active batches.

Most experiments are performed over 802.11b with a bit-rate of 5.5Mb/s. In §4.4, we allow traditional routing (i.e., Srcr) to exploit the autorate feature in the MadWifi driver, which uses the Onoe bit-rate selection algorithm [8]. Current autorate control optimizes the bit-rate for the nexthop, making it unsuitable for opportunistic routing, which broadcasts every transmission to many potential nexthops. The problem of autorate control for opportunistic routing is still open. Thus in our experiments, we compare Srcr with autorate to opportunistic routing (MORE and ExOR) with a fixed bit-rate of 11 Mb/s.

4.2 Throughput

We would like to examine whether MORE can effectively exploit opportunistic receptions to improve the throughput and compare it with Srcr and ExOR.

4.2.1 How Do the Three Protocols Compare?

Does MORE improve over ExOR? How do these two opportunistic routing protocols compare with traditional best path routing? To answer these questions, we use these protocols

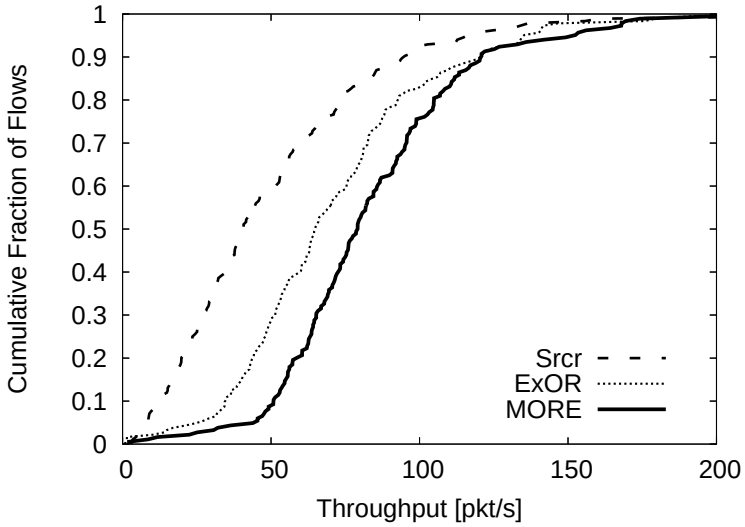


Figure 4-2: **Unicast Throughput.** Figure shows the CDF of the unicast throughput achieved with MORE, ExOR, and Srcr. MORE's median throughput is 22% higher than ExOR. In comparison to Srcr, MORE achieves a median throughput gain of 95%, while some source-destination pairs show as much as 10-12x.

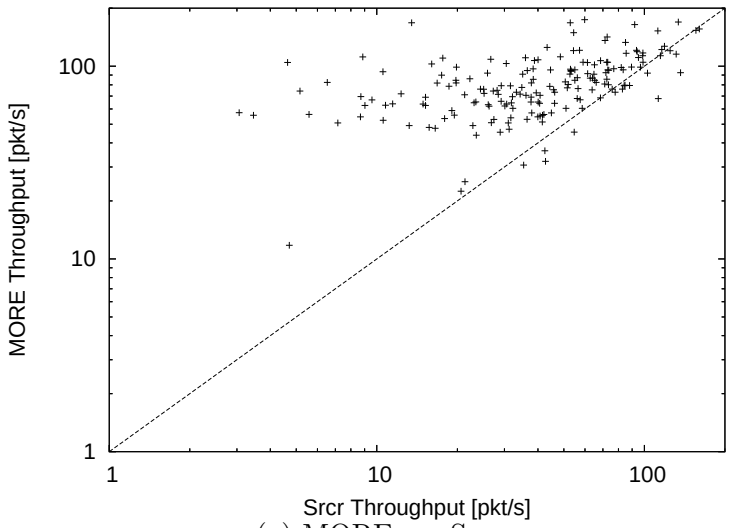
to transfer a 5 MByte file between various nodes in our testbed. We repeat the same experiment for MORE, ExOR, and Srcr as explained in §4.1.2.

Our results show that MORE significantly improves the unicast throughput. In particular, Fig. 4-2 plots the CDF of the throughput taken over 200 randomly selected source-destination pairs in our testbed. The figure shows that both MORE and ExOR significantly outperform Srcr. Interestingly, however, MORE's throughput is higher than ExOR's. In the median case, MORE has 22% throughput gain over ExOR. Its throughput gain over Srcr is 95%, but some challenged flows achieve 10-12x higher throughput with MORE than traditional routing.

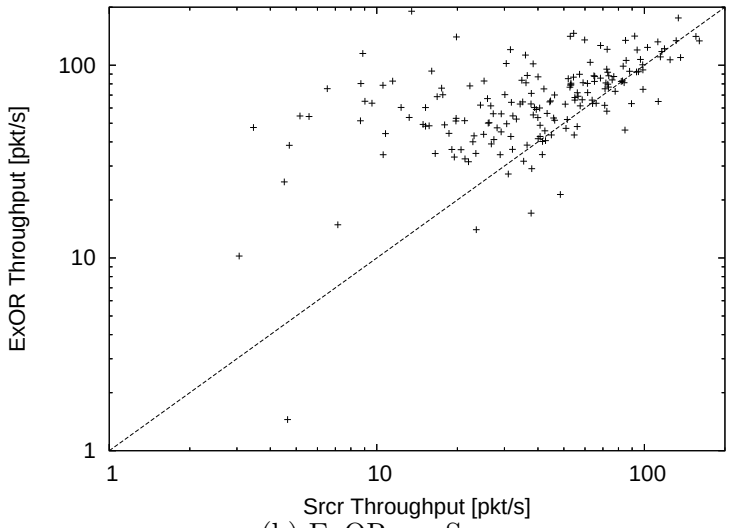
Further, MORE and opportunistic routing ease the problem of dead spots. Fig. 4-2 shows that over 90% of MORE flows have a throughput larger than 50 packets a second. ExOR's 10^{th} percentile is at 35 packets a second. Srcr on the other hand suffers from dead spots with many flows experiencing very low throughput. Specifically, the 10^{th} percentile of Srcr's throughput is at 10 packets a second.

4.2.2 When Does Opportunistic Routing Win?

We try to identify the scenarios in which protocols like MORE and ExOR are particularly useful– i.e., when should one expect opportunistic routing to bring a large throughput gain?



(a) MORE vs. Srcr



(b) ExOR vs. Srcr

Figure 4-3: **Scatter Plot of Unicast Throughput.** Each point represents the throughput of a particular source destination pair. Points above the 45-degree line indicate improvement with opportunistic routing. The figure shows that opportunistic routing is particularly beneficial to challenged flows.

Fig. 4-3a shows the scatter plot for the throughputs achieved under Srcr and MORE for the same source-destination pair. Fig. 4-3b gives an analogous plot for ExOR. Points on the 45-degree line have the same throughput in the two compared schemes.

These figures reveal that opportunistic routing (MORE and ExOR) greatly improves performance for challenged flows, i.e., flows that usually have low throughput. Flows that achieve good throughput under Srcr do not improve further. This is because when links on

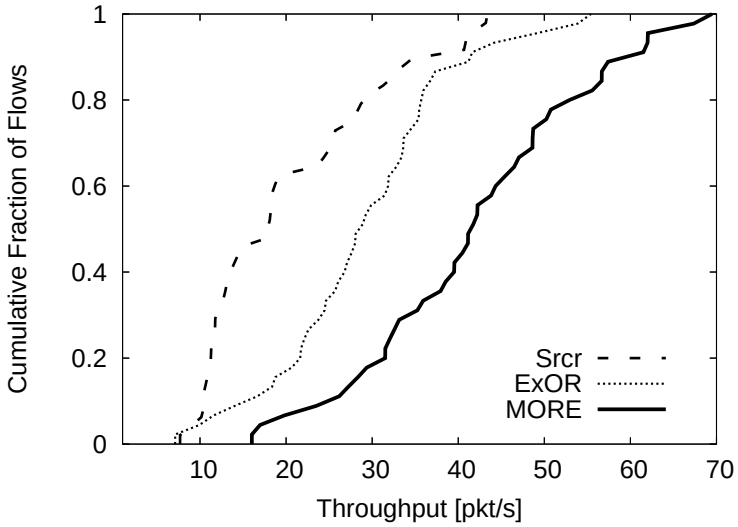


Figure 4-4: **Spatial Reuse.** The figure shows CDFs of unicast throughput achieved by MORE, ExOR, and Srcr for flows that traverse 4 hops, where the last hop can transmit concurrently with the first hop. MORE's median throughput is 50% higher than ExOR.

the best path have very good quality, there is little benefit from exploiting opportunistic receptions. In contrast, a source-destination pair that obtains a low throughput under Srcr does not have any good quality path. Usually, however, many low-quality paths exist between the source and the destination. By using the combined capacity of all these low-quality paths, MORE and ExOR manage to boost the throughput of such flows.

4.2.3 Why Does MORE Have Higher Throughput than ExOR?

Our experiments show that spatial reuse is a main contributor to MORE's gain over ExOR. ExOR prevents multiple forwarders from accessing the medium simultaneously [10], and thus does not exploit spatial reuse. To examine this issue closely, we focus on a few flows that we know can benefit from spatial reuse. Each flow has a best path of 4 hops, where the last hop can send concurrently with the first hop without collision. Fig. 4-4 plots the CDF of throughput of the three protocols for this environment. Focusing on paths with spatial reuse amplifies the gain MORE has over ExOR. The figure shows that for 4-hop flows with spatial reuse, MORE on average achieves a 50% higher throughput than ExOR.

It is important to note that spatial reuse may occur even for shorter paths. The capture effect allows multiple transmissions to be correctly received even when the nodes are within the radio range of both senders [32]. In particular, less than 7% of the flows in Fig. 4-2

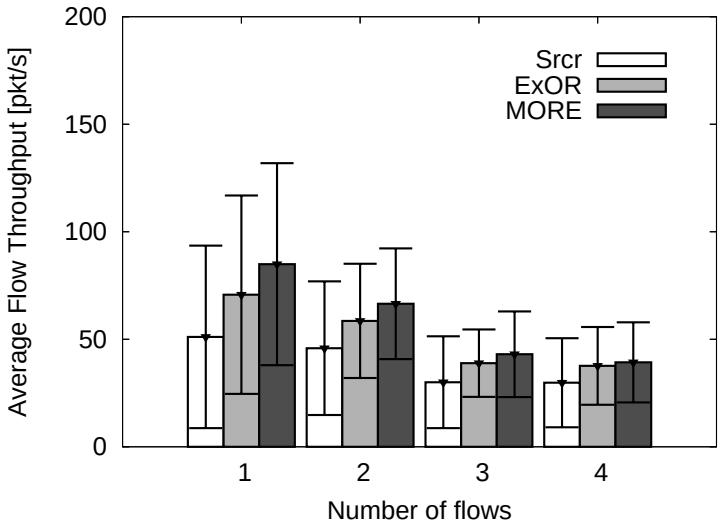


Figure 4-5: **Multi-flows.** The figure plots the per-flow average throughput in scenarios with multiple flows. Bar show the average of 40 random runs. Lines show the standard deviation.

have a best path of 4 hops or longer. Still MORE does better than ExOR. This is mainly because of capture. The capture effect, however, is hard to quantify or measure. Thus, we have focused on longer paths to show the impact of spatial reuse.

4.3 Multiple Flows

One may also ask how MORE performs in the presence of multiple flows. Further, since the ExOR paper does not show any results for concurrent flows, this question is still open for ExOR as well. We run 40 multi-flow experiments with random choice of source-destination pairs, and repeat each run for the three protocols.

Fig. 4-5 shows the average per-flow throughput as a function of the number of concurrent flows, for the three protocols. Both MORE and ExOR achieve higher throughput than Srcr. The throughput gains of opportunistic routing, however, are lower than for a single flow. This highlights an inherent property of opportunistic routing; it exploits opportunistic receptions to boost the throughput, but it does not increase the capacity of the network. The 802.11 bit rate decides the maximum number of transmissions that can be made in a time unit. As the number of flows in the network increases, each node starts playing two roles; it is a forwarder on the best path for some flow, and a forwarder off the best path for another flow. If the driver polls the node to send a packet, it is better to send a packet from the flow for which the node is on the best path. This is because the links on the best path

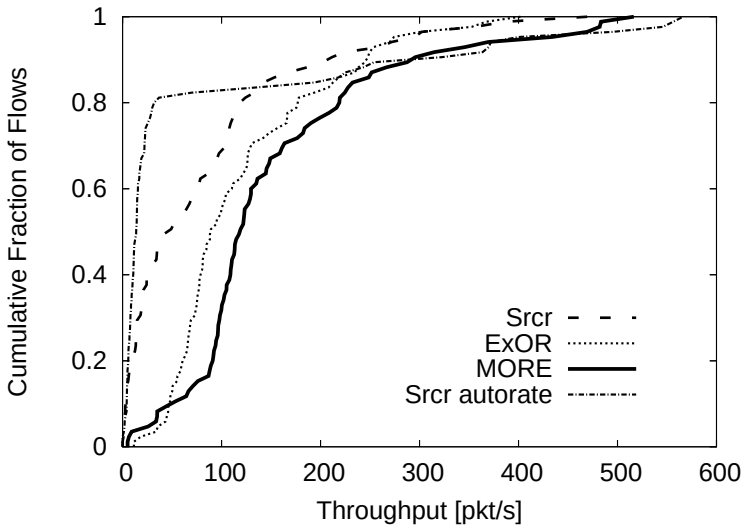


Figure 4-6: **Opportunistic Routing Against Srcr with Autorate.** The figure compares the throughput of MORE and ExOR running at 11Mb/s against that of Srcr with autorate. MORE and ExOR preserve their throughput gains over Srcr.

usually have higher delivery probability. Since the medium is congested and the number of transmissions is bounded, it is better to transmit over the higher quality links.

Also, the gap between MORE and ExOR decreases with multiple flows. Multiple flows increase congestion inside the network. Although a single ExOR flow may underutilize the medium because it is unable to exploit spatial reuse, the congestion arising from the increased number of flows covers this issue. When one ExOR flow becomes unnecessarily idle, another flow can proceed.

Although the benefits of opportunistic routing decrease with a congested network, it continues to do better than best path routing. Indeed, it smoothly degenerates to the behavior of traditional routing.

Finally, this section highlights the differences between inter-flow and intra-flow network coding. Katti et al. [21] show that the throughput gain of COPE, an inter-flow network coding protocol, increases with an increased number of flows. But, COPE does not apply to unidirectional traffic and cannot deal with dead spots. Thus, inter-flow and intra-flow network coding complement each other. A design that incorporates both MORE and COPE is a natural next step.

4.4 Autorate

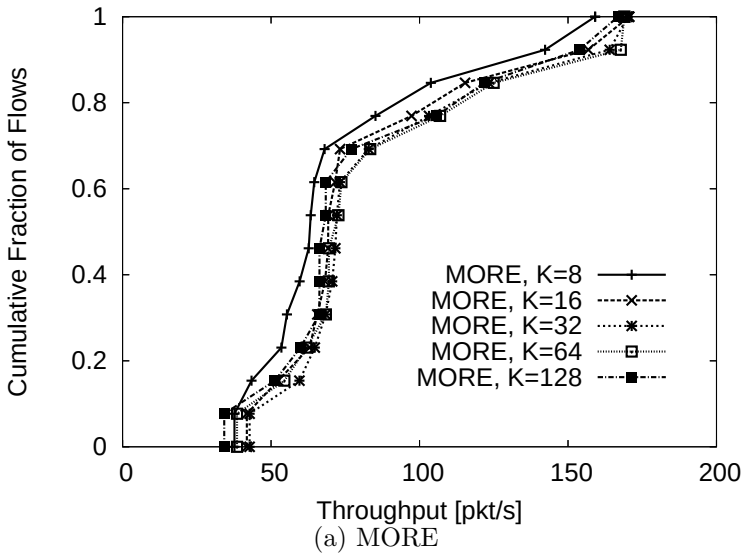
Current 802.11 allows a sender node to change the bit rate automatically, depending on the quality of the link to the recipient. One may wonder whether such adaptation would improve the throughput of Srcr and nullify the gains of opportunistic routing. Thus, in this section, we allow Srcr to exploit the autorate feature in the MadWifi driver [29], which uses the Onoe bit-rate selection algorithm [8].

Opportunistic routing does not have the concept of a link, it broadcasts every packet to many potential nexthops. Thus, current autorate algorithms are not suitable for opportunistic routing. The problem of autorate control for opportunistic routing is still open. Therefore, in our experiments, we compare Srcr with autorate to opportunistic routing (MORE and ExOR) with a fixed bit-rate of 11 Mb/s.

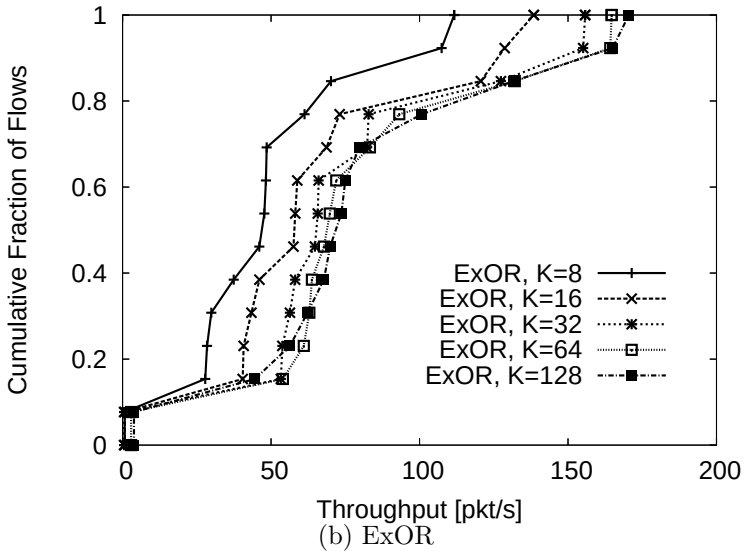
Fig. 4-6 shows CDFs of the throughputs of the various protocols. The figure shows that MORE and ExOR preserve their superiority to Srcr, even when the latter is combined with automatic rate selection. Paths with low throughput in traditional routing once again show the largest gains. Such paths have low quality links irrespective of the bit-rate used, therefore autorate selection does not help these paths.

Interestingly, the figure also shows that autorate does not necessarily perform better than fixing the bit-rate at the maximum value. This has been noted by prior work [35] and attributed to the autorate confusing collision drops from error drops and unnecessarily reducing the bit-rate.

A close examination of the traces indicates that the auto-rate algorithm often picks the lowest bit-rate in attempt to reduce packet loss. However, the improvement in quality of the relatively good links is limited, and a large fraction of the losses is due to interference thus cannot be avoided by reducing the bit-rate. This limited benefit is greatly outweighed by the sacrifice in bandwidth efficiency. In our experiments, the average success rate of all transmissions improves only slightly with autorate from 66% to 68%. At the same time, on average 23% of all transmissions using autorate are done at the lowest bit-rate, which takes roughly 10 times longer than the highest bit-rate. These transmissions form a throughput bottleneck and consume almost 70% of the shared medium time. As shown in Fig. 4-6, this problem affects about 80% of all flows tested.



(a) MORE



(b) ExOR

Figure 4-7: **Impact of Batch Size.** The figure shows the CDF of the throughput taken over 40 random node pairs. It shows that MORE is less sensitive to the batch size than ExOR.

4.5 Batch Size

We explore the performance of MORE and ExOR for various batch sizes. Fig. 4-7 plots the throughput for batch sizes of 8, 16, 32, 64, and 128. It shows that ExOR's performance with small batches of 8 packets is significantly worse than large batches. In contrast, MORE is highly insensitive to different batch sizes.

In both ExOR and MORE, the overhead increases with reduced batch size. ExOR nodes exchange control packets whenever they transmit a batch. Increasing the batch size allows

Operation	Avg. Time [μs]	Std. Dev. [μs]
Independence check	10	5
Coding at the source	270	15
Decoding	260	150

Table 4.1: **Average computational cost of packet operations in MORE.** The numbers for $K = 32$ and 1500B packets are measured on a low-end Celeron machine clocked at 800MHz with 128KiB cache. 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.

ExOR to amortize the control traffic and reduces the chance of spurious transmissions. MORE may make 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 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 traditional routing. Note that MORE benignly co-exists with traditional routing, which it uses to deliver its ACKs.

4.6 MORE's Overhead

Finally, we would like to estimate MORE's overhead and its suitability for deployment in mesh networks like Roofnet [4] and community wireless networks [33, 6].

(a) Coding Overhead: In MORE, the cost of coding/decoding packets is incurred mainly when the packet has to be multiplied by a random number (in a finite field of size 2^8). To optimize this operation, our implementation reduces the cost by using a 64KiB lookup-table indexed by pairs of 8 bits. The lookup table caches results of all possible multiplications, so multiplying any byte of a packet with a random number is simply a fast lookup.

Table 4.1 provides micro benchmarks for coding and decoding in MORE. The measurements are taken on a low-end Celeron 800MHz machine. The benchmarks show that coding and decoding have roughly equal cost. They require on average K finite-field multiplications per byte, where K is the batch size. This ties the choice of K with the maximum achievable throughput. In our setting $K = 32$ and coding takes on average $270\mu s$ per 1500B packet.

This limits the effective throughput to 44 Mb/s, which is higher than the effective bit rate of current wireless mesh networks [20].

(b) Memory Overhead: In MORE like in ExOR, routers do not keep an output queue. Instead, they store the current batch from each flow. This per-flow state is dominated by the storage required to buffer innovative packets from the current batch, which is bounded by $K = 32$ packets. Additionally, as stated above, MORE nodes keep a 64KiB lookup-table. Given that the number of concurrent flows in a mesh network is relatively small, we believe MORE's memory overhead is acceptable.

(c) Header Overhead: MORE's header in our current implementation is bounded by 70 bytes because we bound the number of forwarders to 10. Certain values in the header are compressed to increase efficiency. For example, since routers only keep the current batch, we can represent batch IDs using a few bits. Similarly, we compress the node ID in the forwarder list to one byte, which is a hash of its IP. This works because only nodes whose ETX to the destination is smaller than the source are allowed to participate in forwarding. For 1500B packets, the header overhead is less than 5%. Note that our throughput numbers are computed over the delivered data, and thus they already account for header overhead.

Chapter 5

Theoretical Bounds

MORE attempts to exploit the broadcast capacity of wireless networks with network coding. In this chapter, we extend the practical contribution with analysis of the theoretical bounds on the performance of network coding and opportunistic routing.[1]

5.1 Prior Theoretical Results

In [26], Lun et al. show that the random network coding scheme as described in §3.1 is *capacity–achieving*, i.e., given fixed link throughputs (including broadcast) it can achieve flow throughput arbitrarily close to the maximum flow capacity of the network for a sufficiently large finite field size. This result, however, is insufficient to ensure high throughput in wireless networks, where a new problem emerges of how to share the wireless medium between the forwarders. In wired networks, the bandwidth is isolated in individual links, so if each link operates at full speed, the capacity is maximized. In a wireless network, however, nodes in the same vicinity can rarely transmit simultaneously as their signals would interfere at the receiver, so we must control what fraction of the bandwidth to allocate to each transmitter. As noted in [27], the location-dependent nature of interference makes this problem hard, because it is difficult to predict whether two transmissions would interfere or not.

To simplify the task, we can take on a different approach of minimizing the cost, or the number of transmissions necessary to deliver a packet across the network. This rule of

[1]This work was done after developing the protocol and addressing the practical challenges. As presented and evaluated in previous chapters, MORE does not fully utilize the theoretical results of this chapter.

minimizing the resources (transmissions) spent per unit of service (packet) is most sensible and widely followed in the design of wireless network protocols. For example, the hop-by-hop link-layer retransmissions mentioned in §2.1 avoid the cost of multi-hop retransmissions. The ETX metric, which accounts for the retransmissions needed to reliably relay each packet [13], favors paths that require the fewest transmissions per packet. Lastly, the forwarding rule of ExOR selects after each transmission such forwarder that offers the smallest ETX path to the destination [10], which is roughly equivalent to minimizing the cost incurred by the packet on the remainder of the trip.

To minimize the cost, we must choose for each node, how many transmissions per unit of flow it should make. In [27], Lun et al. propose a linear program for the general minimum cost flow in lossy wireless networks, which bounds the performance of any forwarding scheme. What makes this problem difficult is the exponential number of flow capacity constraints, due to the broadcast nature of the radio medium. To cope with this problem, the authors propose a distributed subgradient-based algorithm and assume that the neighborhood of each node is limited to some constant number of nodes. This bounds the complexity, but precludes dense networks or discards some opportunistic receptions at the very least.

5.2 Contribution

In this chapter, we consider the same linear programming (LP) problem of (uncapacitated) minimum-cost unicast flow in lossy wireless networks and provide the following results:

- We introduce EOTX – a metric for opportunistic transport schemes, which is shown to be equal to the minimum expected number of transmissions necessary among all nodes in order to deliver a single packet from source to sink.
- We propose two efficient algorithms that follow the standard shortest-path framework to compute the metric, and another to compute the flow variables in detail, i.e., solve the minimum cost LP problem. The overall computational complexity is $O(n^2)$, which is a significant improvement over previous work that needed to solve the LP problem with potentially $O(n2^n)$ constraints, where n is the number of nodes in the network.

5.3 Minimum-cost Information Flow

The distinctive characteristic of the wireless medium is its broadcast nature, which allows multiple nodes to receive a single transmission. However, the widely employed practice is to conceal this property in a point-to-point abstraction, in which a transmission can only be received by a predestined node, and subsequently use the model for wired networks. The deficiency of this model becomes clear when we consider the possibility of losses. In a lossy wireless network a transmission might be received by some subset of its neighbors, which cannot be predetermined. An optimal strategy should then exploit all successful receptions.

In this section, we define the network model that gives a simplistic but broadcast-aware view of the lossy wireless medium, and introduce the LP problem that is equivalent to minimum cost flow. The solution to the problem will tell us how many transmissions are necessary to deliver one packet from the source to the destination.

5.3.1 Network Model

The network is comprised of broadcast-capable nodes. A broadcast transmission can be received by each node with some probability. We assume that successful receptions are independent of other transmissions. This requires that the transmissions do not interfere with each other as well as there being no intermittent external interference. The model implies that, whenever node i transmits z_i packets, node j is expected to receive $p_{ij}z_i$ of them, where p_{ij} is the marginal probability that j receives a transmission from i. Since we are actually interested in the expected values, e.g. how many transmissions on average are required to deliver one packet, we will assume throughout this paper that the amounts computed are expectations. This will allow a flow formulation in the next subsection.

Given the broadcast nature of the radio medium, there are no edges in a wireless network graph. Instead, it consists of hyper-edges spanning from the transmitter to each possible set of receiver nodes. The network is fully described by the joint probability distribution of packet reception, i.e., p_{iK} being the probability that a transmission from node i is received by exactly the nodes in set K, given that i transmits. Observe that there could be $O(n2^n)$ hyper-edges in a network of n nodes. We simplify the model and assume that there is no choice of modulation or output power, and hence the distribution p_{iK} is fixed. Note that it is often assumed that the receptions by *individual* nodes are independent given the

transmitter, or

$$p_{iK} = \prod_{k \in K} p_{ik} \prod_{k \notin K} (1 - p_{ik})$$

However, a network could exhibit spatial features that correlate receptions of nearby nodes, e.g. local source of interference. Throughout the paper, we shall work on the general model, except for §5.5, where we describe the algorithm for independent receptions.

5.3.2 Problem Statement

Although the model above is discrete, we are interested in expected values. This allows us to use real-valued flow formulation. The flow in question is interpreted as amount of *information*, so the results apply to any communication protocol that fits the network model, including network coding.

The task is to minimize the (expected) number of transmissions performed by all nodes in order to deliver R units of flow (e.g. packets) from source node s to sink node t. Following the work of Lun et al. we consider the amount of *innovative* packets with respect to the sink of the flow that is passed between two neighbors. We focus on a single sink and denote it x_{ij}. Information cannot be made up, and it is certainly suboptimal to destroy information, so these values are subject to flow conservation constraints:

$$\sum_k x_{ik} - x_{ki} = \begin{cases} R & i = s \\ -R & i = t \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Another set of constraints limits x_{ij} by the number of packets actually received. If node i performs z_i transmissions then:

$$q_{iK} z_i - \sum_{k \in K} x_{ik} \geq 0 \quad , \quad \forall i, K \tag{5.2}$$

where q_{iK} is the probability that *at least one* node in set K receives a transmission from i, i.e., $q_{iK} = \sum_{J \cap K \neq \phi} p_{iJ}$. We shall call them *cost constraints*. Our optimization goal is

simply to minimize the overall network-wide cost of the flow:

$$\begin{aligned} &\min \sum_i z_i \\ &\text{over } z_i,\, x_{ij} \geq 0 \end{aligned} \tag{5.3}$$

Consider the cost constraints (5.2). Note that there is one for every hyper-edge (iK), thus the upper bound on the number of innovative packets over a simple edge, x_{ik} , can be lower than the number of packets received over that edge. For example, let: $z_i = 10$; $q_{i\{j\}} = p_{ij} = 0.5$; $q_{i\{k\}} = p_{ik} = 0.5$; $q_{i\{jk\}} = 0.75$. Then: $x_{ij} \leq 5$; $x_{ik} \leq 5$; $x_{ij} + x_{ik} \leq 7.5$. So, even though nodes j and k together are expected to receive 10 packets, only 3/4 of them are expected to be innovative, due to the expected overlap in information content. Packets that are received by both nodes cannot be counted twice, because their content is the same. More specifically, packets are regarded as innovative if they carry new information, e.g. are linearly independent.

Although, given the probability interpretation, one could require that $K \subset J \implies q_{iK} \leq q_{iJ}$, this can be assumed for the LP, since $K \subset J \implies \sum_{k\in K} x_{ik} \leq \sum_{k\in J} x_{ik} \leq z_i q_{iJ} \implies \sum_{k\in K} x_{ik} \leq z_i \min(q_{iK}, q_{iJ})$.

We can now explain what it means to say that network coding is *capacity–achieving* as shown by Lun et al. in [26]. Namely, if the nodes apply network coding to each transmission and transmit on average z_i times in unit time, then we achieve an information flow per unit time of amount matching the capacity determined by the minimum cut S, T (the maximum amount of flow that can go through it): $R = \sum_{i\in S} \sum_{k\in T} x_{ik} \leq \sum_{i\in S} q_{iT} z_i$. Hence, using network coding, we can indeed achieve flows that are feasible in our model.

5.3.3 Solution

In this section, we first decompose the global problem into a local one. Next we solve the local problem exactly. We begin with a simple observation:

Proposition 1 (Scaling)**.** *If z, x make up an optimal solution for some R, then $\alpha z, \alpha x$ is an optimal solution for αR.*

Proof. This is a general property of LP. If we scale both the free constants and all variables by the same positive factor, then we remain optimal. □

Decomposition

To solve the LP we will decompose it into subproblems, so let us define what constitutes one. We use instances of the LP for some fixed q_{iK}, t, and $R = 1$, but varying the source node s. We will denote the optimal value of the objective of such a problem:

Definition 1.

$$\mathbb{Z}_s = \min \sum_i z_i$$

subject to the flow and cost constraints (5.1)-(5.2)*, for* $R = 1$.

Note that s is a *parameter* here. In other words, $\mathbb{Z}_s$ is the minimum cost needed to transport a unit of flow from s to t. In particular, define $\mathbb{Z}_t = 0$ for completeness. We will also refer to $\mathbb{Z}_s$ as *cost* of node s. The scaling property implies that it costs $R\mathbb{Z}_s$ to deliver R units.

Theorem 1 (Decomposition)**.**

$$\begin{aligned} \mathbb{Z}_s &= \min \left[z_s + \sum_k x_{sk} \mathbb{Z}_k \right] && (5.4) \\ \textit{subject to} \quad \sum_k x_{sk} &= 1 && (5.5) \\ q_{sK} z_s - \sum_{k \in K} x_{sk} &\geq 0 \quad ,\forall_K && (5.6) \\ z_s, x_{sk} &\geq 0 \end{aligned}$$

Note that this is a local problem, where $\mathbb{Z}_k$ and s are parameters. In other words, once we have computed the costs of all the other nodes, we can compute the cost for s by choosing the best feasible distribution of the flow among the other forwarders. Note that this resembles how the standard cheapest path problem can be decomposed as: $d(s) = \min_k (c_{sk} + d(k))$, where $d(s)$ is the minimum cost of a path from s to t, and c_{sk} is the cost of the edge (s, k). To prove the theorem we use:

Lemma 1 (Superposition)**.** *Consider a flow with (non-negative) surplus at every node,*

subject to the constraints:

$$\begin{aligned} \sum_k x_{ik} - x_{ki} &= R_i, \quad i \neq t \\ q_{iK} z_i - \sum_{k \in K} x_{ik} &\geq 0 \quad , \forall_{i,K} \\ z_i, x_{ik} &\geq 0 \end{aligned}$$

(Observe $\sum_k x_{tk} - x_{kt} = -\sum_{i \neq t} R_i$*) Then the optimal min-cost solution is the superposition of the solutions to the subproblems:*

$$\min \sum_i z_i = \sum_i R_i \mathbb{Z}_i$$

That the superposition is feasible follows from the linearity of LP, but we claim that it is optimal. We prove the lemma by induction on the number of nodes in the network.

Proof by complete induction. The base case is when the network contains just the sink $\{t\}$. Then $\mathbb{Z}_t = z_t = 0$, so the case is trivial.

Now assume Superposition holds for networks of less than n nodes, and we are given a network of n nodes. Observe that flow conservation implies it can be decomposed into paths and cycles. Furthermore, the cycles can be safely removed without losing feasibility nor optimality. Therefore there must exist a node, denoted s, such that $x_{is} = 0$ for all i (i.e., first in topological order). Then these constraints can be split into:

$$\begin{aligned} F_i: &\quad \textstyle\sum_{k \neq s} x_{ik} - x_{ki} &&= x_{si} + R_i \qquad i \notin \{s, t\} \\ F_s: &\quad \textstyle\sum_k x_{sk} &&= R_s \\ Q_i: &\quad q_{iK} z_i - \textstyle\sum_{k \in K} x_{ik} &&\geq 0 \quad , \forall_{i \neq s, K} \\ Q_s: &\quad q_{sK} z_s - \textstyle\sum_{k \in K} x_{sk} &&\geq 0 \quad , \forall_K \end{aligned}$$

The constraints on z_i and x_{ik} for $i \neq s$ do not depend on z_s given x_{si}, so we can decompose

the global objective

$$\begin{aligned}\min_{F,Q}\sum_i z_i &= \min_{F_s,Q_s}\left[z_s + \min_{F_i,Q_i}\sum_{i\neq s} z_i\right] = \\ \text{(by Superposition)} &= \min_{F_s,Q_s}\left[z_s + \sum_{i\neq s}(x_{si}+R_i)\mathbb{Z}_i\right]\end{aligned}$$

Note that for the special case when $R_i = 0$ for $i \neq s$ and $R_s = 1$ the objective equals $\mathbb{Z}_s$ (and s is still the first in topological order), so we obtain both Decomposition and Superposition:

$$\begin{aligned}R_s\mathbb{Z}_s &= \min_{F_s,Q_s}\left[z_s + \sum_{i\neq s} x_{si}\mathbb{Z}_i\right] \\ \min_{F,Q}\sum_i z_i &= \min_{F_s,Q_s}\left[z_s + \sum_{i\neq s}(x_{si}+R_i)\mathbb{Z}_i\right] = \sum_i R_i\mathbb{Z}_i\end{aligned}$$

□

Local Problem

Let us focus on a specific source node s, and drop this index from all subscripts. We also order all nodes in ascending cost, $\mathbb{Z}_i$; i.e., node 1 is the node of smallest cost, so $1 = t$, $\mathbb{Z}_1 = 0$. With no loss to generality, we assume the order is strict. The problem at hand is:

$$\mathbb{Z} = \min\left[z + \sum_k x_k\mathbb{Z}_k\right] \tag{5.7}$$

$$\begin{aligned}\text{subject to } \sum_k x_k &= 1 \\ q_K z - \sum_{k\in K} x_k &\geq 0 \quad \forall_K \\ z, x_k &\geq 0\end{aligned} \tag{5.8}$$

Even though there might still be exponentially many cost constraints (5.8), we will show that only one constraint per neighbor is actually required. For each node i consider the set $\{1\ldots i\}$, i.e., of the cheapest i nodes. We will use the shorthand $q_i = q_{\{1\ldots i\}}$ interpreted as the probability that at least one node among the cheapest i receives the transmission. We set $q_0 = 0$ for completeness; as pointed out in §5.3.2, $q_{i+1} \geq q_i$. For now we just relax the

problem and leave cost constraints involving such sets only:

$$q_i z - \sum_{k=1}^{i} x_k \geq 0 \quad \forall i \tag{5.9}$$

We will show that the remaining constraints are satisfied after we find the solution.

Proposition 2 (Water filling)**.** *The optimal solution is to always send as much as possible to the cheapest available neighbor subject to the constraints* (5.9)*. So,* $x_1 = q_1 z$; $x_2 = q_2 z - x_1$; *... until we reach* s*. This gives us:*

$$x_k = (q_k - q_{k-1})z \quad \forall_{k<s} \tag{5.10}$$

$$x_k = 0 \quad \forall_{k\geq s} \tag{5.11}$$

$$z = 1/q_{s-1} \tag{5.12}$$

This result is quite intuitive. Given a choice of where to send the flow, we should definitely prefer the cheaper one of any two candidates, but if we go too far, we will need to perform more transmissions to deliver the flow. We prove it formally:

Proof. We first show that the optimum requires (5.11), i.e., that the flow never goes to a higher cost node. Suppose a solution $\mathbb{Z}$ sets $x_i > 0$ for some $i > s$ (i.e., $\mathbb{Z}_i > \mathbb{Z}_s = \mathbb{Z}$). Then we can improve it by setting $x'_i = 0$ and rescaling the other variables to fit the constraints.

$$\text{Set } f = \frac{1}{1 - x_i},\ x'_k = x_k f,\ x'_i = 0,\ z' = zf$$

This assignment remains feasible, but the objective is improved:

$$\mathbb{Z}' = z' + \sum_k x'_k \mathbb{Z}_k = f\left(\mathbb{Z} - x_i \mathbb{Z}_i\right) < \mathbb{Z} f(1 - x_i) = \mathbb{Z}$$

Now consider the first node i that violates the condition (5.10). Then $q_i z \geq \sum_{k\leq i} x_k = q_{i-1} z + x_i$, and therefore $x_i < (q_i - q_{i-1})z$. Then consider the next node j such that $x_j > 0$. There are two cases:

1. Such a node j exists. In this case, we can improve the solution by shifting the flow to i, which offers lower cost. Let $\delta = \min(x_j, (q_i - q_{i-1})z - x_i)$ and set $x'_i = x_i + \delta, x'_j = x_j - \delta$. We did not break any of the constraints (the only ones affected cover q_i, q_j and x_j),

but we gained in the goal:

$$\mathbb{Z}' = z + \sum_k x'_k \mathbb{Z}_k = \mathbb{Z} + \delta(\mathbb{Z}_i - \mathbb{Z}_j) < \mathbb{Z}$$

2. For all $j > i$, $x_j = 0$. Then $1 = \sum_{k \le i} x_i = q_{i-1}z + x_i < q_i z$ so $x_i = 1 - q_{i-1}z$ and $z > 1/q_i$. Instead, set $z' = 1/q_i$ and $x'_k = (q_k - q_{k-1})z$ for all $k \le i$ and improve:

$$\begin{aligned} \mathbb{Z}' = \quad & z' \left(1 + \textstyle\sum_{k<i}(q_k - q_{k-1})\mathbb{Z}_k - q_{i-1}\mathbb{Z}_i\right) + q_i z' \mathbb{Z}_i \quad < \\ < \quad & z \left(1 + \textstyle\sum_{k<i}(q_k - q_{k-1})\mathbb{Z}_k - q_{i-1}\mathbb{Z}_i\right) + \mathbb{Z}_i \qquad = \mathbb{Z} \end{aligned}$$

Finally, given (5.10)-(5.11), the constraint $q_{s-1}z \ge \sum_{i<s} x_i = 1$ implies that $z = 1/q_{s-1}$ is the minimum. □

The issue with this form of the solution is that to know all q_i we must have the order of all $\mathbb{Z}_i$, so we still need an algorithm to compute that from scratch.

Before we proceed, let us prove the claim that this solution to the relaxed problem is still feasible with respect to the omitted constraints. Observe that our solution is tight on the cost constraints for q_i, so we prove the following:

Proposition 3. *If $q_i z - \sum_{k=1}^{i} x_k = 0$ for all $i < s$, then $q_K z - \sum_{k \in K} x_k \ge 0$ for all $K \subseteq \{1 \dots s-1\}$, and consequently – for all K, since $x_i = 0$ for all $i \ge s$.*

Proof. The assumption implies that for $k < s$, $x_k = (q_k - q_{k-1})z$, therefore the claim can be written as:

$$q_K \ge \sum_{k \in K} (q_k - q_{k-1})$$

Recall from §5.3.1 that (after omitting the source index s) $q_K = \sum_{J \cap K \ne \phi} p_J$ where p_J is the probability that exactly the nodes in set J receive a transmission. So, $q_k = q_{\{1\dots k\}} = \sum_{J \cap \{1\dots k\} \ne \phi} p_J$, which gives us:

$$q_k - q_{k-1} = \sum_{\substack{J \cap \{1\dots k\} \ne \phi \\ J \cap \{1\dots(k-1)\} = \phi}} p_J = \sum_{J \ni k, J \subseteq \{k\dots s-1\}} p_J = \sum_{J : \inf J = k} p_J \tag{5.13}$$

This is not unexpected. After all, both sides represent the probability that k has the smallest

cost among all that successfully receive the transmission. We can conclude that

$$\sum_{k\in K}(q_k - q_{k-1}) = \sum_{k\in K}\sum_{J:\inf J=k} p_J = \sum_{J:\inf J\in K} p_J \leq \sum_{J\cap K\neq\phi} p_J = q_K$$

For any other set K, we have

$$\sum_{k\in K} x_k = \sum_{k\in K\cap\{1\ldots s-1\}} x_k \leq q_{K\cap\{1\ldots s-1\}} z \leq q_K z$$

□

5.4 EOTX

In this section, we arrive at the same result as above, but instead of a flow formulation we use the forwarding rule proposed for ExOR by Biswas and Morris [10]. We introduce EOTX, a new routing metric that computes the optimal expected number of transmissions required to deliver a packet over the network. EOTX is a generalization of ETX, as it computes the expected number of transmissions when all paths between sender and receiver are used, while ETX is limited to single path routing. Thus, EOTX provides a baseline for evaluating opportunistic routing protocols.

We define EOTX as the minimum expected number of opportunistic transmissions that need to be performed in the network in order to deliver a single packet from source to sink. The opportunism is embodied in the forwarding scheme that is based on the rule: *when a packet is broadcast, of all the nodes that received it successfully (including self), only the one should forward it which has the lowest EOTX.*[2] As in the previous section, we focus on node s and drop this index from subscripts. Therefore, for a particular sink node t, the EOTX of node s is defined as:

$$d(s) = \begin{cases} 0 & s = t \\ 1 + \sum_K p_K \min_{k\in K} d(k) & \text{otherwise} \end{cases} \tag{5.14}$$

where as before $p_K = p_{sK}$ is the probability that a packet broadcast by s is received by *exactly* the nodes in set K; except now we also assume K includes the transmitter s itself, so

[2]Note that this requires perfect coordination.

that it also is a candidate forwarder. Note that since s always "receives" own transmission, $p_{sK} = 0$ if $s \notin K$.

This is a notoriously recursive but quite intuitive definition. The expected number of transmissions node s needs is the single initial broadcast followed by the expected number of transmissions of the forwarding node, which is the best among all successful recipients. It is also quite clear that EOTX is the lower bound for the expected number of transmissions needed by any scheme similar to ExOR in the assumed model.

Proposition 4 (Equivalence)**.** *EOTX equals the minimum cost of unit flow from source to sink.*

$$d(s) = \mathbb{Z}_s$$

Proof. Consider all nodes in ascending order of $d(k)$. By identifying all reception events that imply each forwarder we can rewrite EOTX:

$$\begin{aligned} d(s) &= 1 + \sum_K p_K \min_{k \in K} d(k) = \\ &= 1 + \sum_K p_K \, d_{\inf K} = \\ &= 1 + \sum_i d(i) \sum_{K : \inf K = i} p_K = \quad \text{using (5.13)} \\ &= 1 + \sum_i d(i)(q_i - q_{i-1}) \end{aligned}$$

Recall that $q_i - q_{i-1}$ is the probability that node i offers the smallest cost among all nodes that receive the transmission. In particular, the probability that no node cheaper than the source receives the packet is: $q_s - q_{s-1} = 1 - q_{s-1}$. We thus obtain:

$$\begin{aligned} d(s) &= 1 + \sum_{i<s} (q_i - q_{i-1}) d(i) + (1 - q_{s-1}) d(s) \\ d(s) &= \frac{1}{q_{s-1}} \left(1 + \sum_{i<s} (q_i - q_{i-1}) d(i) \right) \end{aligned} \tag{5.15}$$

Furthermore, $d_t = \mathbb{Z}_t = 0$, since $t = 1$. Observe this recursion is equivalent to (5.7), when we plug the solution (5.10)-(5.12) in. The proposition follows by simple induction. □

The result is the same. This is not surprising. After all, the reason why the flow requires

a particular number of transmissions is because it is limited by the capacity of all cuts. The forwarders on the source side of the cut must perform sufficient transmissions to deliver the information to any node on the sink side of the cut. Once it is on the other side, the forwarders should not transmit any more, hence the equivalent opportunistic forwarding rule: *do not transmit if any node closer to destination received.*

Corollary 1. *Thus, there exists* no gain *due to network coding for single unicast wireless flows in the presented model. Network coding (which requires mixing the content of packets) and store-and-forward opportunistic routing (which requires perfect reception feedback) share a common information capacity bound.*

5.5 Algorithms

In the previous sections, we have shown how to reformulate the problem into a local one, and what the solution is. We have also defined EOTX, which is shown to have the same solution. Now, we just need an algorithm to untangle the recursion of the solution.

Although the EOTX metric as defined by (5.14) could be computed directly, covering all possible subsets of the neighbors of each node is computationally expensive. Instead, we use the closed-form (5.15). As noted earlier, our decomposition resembles very much the recursive definition of cheapest path. The only challenge is how to determine the position of the subject node in the final cost-order. The general idea is that we can compute the correct cost of node i as soon as we know the costs of all nodes cheaper than i, even if we do not initially know which nodes are indeed cheaper.

The single step for node i will recompute (5.15) assuming that all costs smaller than the newly computed cost $d(i)$ are correct. It considers all nodes in the cost-order, and admits each as a forwarder until its own cost is lower than the candidate's. We now include the subscript of the transmitter omitted in previous considerations.

Algorithm 3 Procedure *Recompute*(i)

Require: nodes are in ascending order of $d()$

$T \leftarrow 1$ {guess on $q_{i(i-1)}d(i)$}

$k \leftarrow 1$ {start from the sink}

while $T/q_{ik} > d(k)$ **do**

{has better cost, admit as forwarder}

$T \leftarrow T + (q_{ik} - q_{i(k-1)})d(k)$

$k \leftarrow k + 1$

return $T/q_{i(k-1)}$

Observe that if sufficient number of correct costs is given, this procedure returns the correct cost $d(i)$. The resulting global procedure is along the lines of the Bellman–Ford algorithm for shortest (or cheapest) paths in graphs [24]:

Algorithm 4 Computing EOTX in a Bellman–Ford fashion

$d() \leftarrow \infty$

$d(t) \leftarrow 0$

for $n = |V|$ times **do**

sort nodes in order

for all $i \neq t$ **do**

$d'(i) \leftarrow$ *Recompute*(i)

$d() \leftarrow d'()$

Proposition 5. *The algorithm is correct, that is, after $n = |V|$ main iterations $d(i)$ is as defined in* (5.15)*.*

Proof. After i iterations the best i costs are correct. We prove it by induction.

1. Initially $i = 1$ and the single best cost, $d_t = 0$ is correct.

2. At iteration $i + 1$, assume that the claim is satisfied for iteration i. The new $(i + 1)$th best cost will be computed by the algorithm only in terms of smaller costs, which are all correct. Therefore, after this iteration the best $(i + 1)$th cost is also correct.

□

The complexity of the algorithm is $O(n^3)$, but make note that $(q_{ik} - q_{i(k-1)}) < p_{ik}$ so if $p_{ik} = 0$, we do not recompute, and the complexity is $O(n(n \log n + m))$, if $m = |\{(i, j) : p_{ij} >$

$0\}|$. Nevertheless, the BF framework is better fit for parallel (e.g. distributed) computation, in which case the recomputations are all run in parallel. Otherwise, a lot of the work is going to waste, because there is no coordination between relaxations.

All this is, of course, assuming that q_{ik} are readily available and need not be computed (or read from input/storage) by the algorithm. However, the computation is simple if the reception probabilities are independent:

$$\begin{aligned} p_{iK} &= \prod_{k \in K} p_{ik} \prod_{k \notin K} (1 - p_{ik}) \\ \implies 1 - q_{ik} &= (1 - q_{i(k-1)})(1 - p_{ik}) \end{aligned}$$

While this loss independence assumption is not always valid, it is supported as a good approximation by prior measurements [32, 30]. We note that the estimates for marginal p_{ij} are much easier to obtain, e.g. in the fashion employed in ETX [13] using periodic probing and reporting to neighbors.

We propose a different algorithm that incorporates these observations. The concept is to keep partially computed costs for all open nodes, but for the computation use only costs of nodes that have been closed, i.e., such that are certainly correct. When a node has the lowest cost among all open nodes, then it cannot be further improved, so we can extract it for relaxation. This is very much like the Dijkstra's algorithm for cheapest path [24]:

Algorithm 5 Computing EOTX in a Dijkstra fashion for independent losses

$d() \leftarrow \infty, \qquad d(t) \leftarrow 0$
$T() \leftarrow 1$ {guess on $q_{i(i-1)}d(i)$}
$P() \leftarrow 1$ {guess on $1 - q_{i(i-1)}$}
$O \leftarrow V$ {queue of open nodes}
repeat
  $k \leftarrow ExtractMin(O)$ {node to visit}
  for $i \in O : p_{ik} > 0$ **do**
    $T(i) \leftarrow T(i) + p_{ik}P(i)d(k)$
    $P(i) \leftarrow P(i)(1 - p_{ik})$
    $d(i) \leftarrow T(i)/(1 - P(i))$
until $O = \phi$

The complexity of this algorithm is $O(n^2)$ (with trivial priority queue) or $O(n \log n + m)$ (with Fibonacci heaps).

5.6 How Many Transmissions to Make?

In the previous section we have focused on finding the minimum total cost, i.e., the optimal value of the objective, but neglected the actual underlying flow and cost variables x, z throughout the network. To actually achieve the minimum cost, we must ensure that each node performs on average the desired number of transmissions per packet to be delivered.

These numbers are of particular interest in routing protocols based on network coding, such as MORE presented here, rather than explicit reception notification, e.g. ExOR [10]. In the classic store-and-forward model (whether opportunistic or on single path) a node only attempts to output a packet when it is presumed that the packet needs to be forwarded, that is it has not yet been received by the next hop, or a set of candidate next hops. With explicit feedback, the nodes know exactly which packets got through, and keep retransmitting until all of them do. However, obtaining feedback in opportunistic routing requires ExOR to circumvent the MAC. In contrast, MORE operates without feedback, instead using the estimated transmission count requirements to ensure high throughput.

5.6.1 Flow Method

We are interested in z_i for each node i, the number of transmissions it should make per packet transported end-to-end. If we have solved the cost using (5.15), then we obtained the EOTX for each node, $d(\cdot)$. If we follow the algorithm closely, we will notice that it is relatively easy to obtain z_i once $d(i)$ is computed, $z_i = 1/(1 - P(i)) = 1/q_{i(i-1)}$, but we must realize that this is for the subproblem $\mathbb{Z}_i$, i.e., when i is the source and the demanded flow is $R = 1$.

Using the Decomposition+Superposition technique, however, we can compute each z_i as soon as we find the *load* of each node, defined as $L_i = \sum_k x_{ik} = \sum_k x_{ki}$, because according to (5.11), the packets should only flow from higher cost to lower cost nodes. So $L_i = \sum_{k>i} x_{ki}$ and we can compute it by following the topological order. The starting condition is $L_s = 1$.

Algorithm 6 Retrieving optimal z, x from $\mathbb{Z}$ via load distribution.

Require: nodes $\{1 \dots s\}$ are in order of ascending $\mathbb{Z}$

```
for all i do
    L(i) ← 0
L(s) ← 1
for i = s ... 1 do
    z(i) = L(i)/q_{i(i-1)}
    for j < i do
        x(i, j) ← (q_ij − q_{i(j-1)}) z(i)
        L(j) ← L(j) + x(i, j)
```

The added complexity is $O(n^2)$. Observe that while $\mathbb{Z}$ is defined locally and does not depend on the source of the global flow, but rather is computed per-sink, the actual values of z and x depend on the global source. Therefore, the added complexity in practical routing scenarios is greater by a linear factor, and dominates the computation of $\mathbb{Z}$. Furthermore, if implemented in the Distributed Bellman-Ford framework, this algorithm requires much more communication overhead (exchanging $x(i,j)$ for each source of interest). Instead, we suggest to implement it using the link-state scheme for better convergence properties.

5.6.2 EOTX Method

In §3.2.1, we have proposed Algorithm 1, which computes the z variables following directly the opportunistic forwarding rule, assuming only that we know the forwarder priority order. MORE, like ExOR, uses ETX to determine the order. However, observe that if the EOTX (optimal) order is used, then Alg. 1 is equivalent to Alg. 6 in the case when losses are independent, since:

$$\begin{aligned} p_{ij} &= 1 - \epsilon_{ij} \\ L_j &= \sum_{i>j} x_{ij} = \sum_{i>j} (q_{ij} - q_{i(j-1)}) z_i = \sum_{i>j} z_i p_{ij} \prod_{k<j} (1 - p_{ik}) = \sum_{i>j} z_i (1 - \epsilon_{ij}) \prod_{k<j} \epsilon_{ik} \\ z_j &= \frac{L_j}{q_{j(j-1)}} = \frac{L_j}{(1 - \prod_{k<j} (1 - p_{jk}))} = \frac{L_j}{(1 - \prod_{k<j} \epsilon_{jk})} \end{aligned}$$

Note that only when EOTX is used for the order, the resulting z_i always add up to $d(s)$, the EOTX of the source.

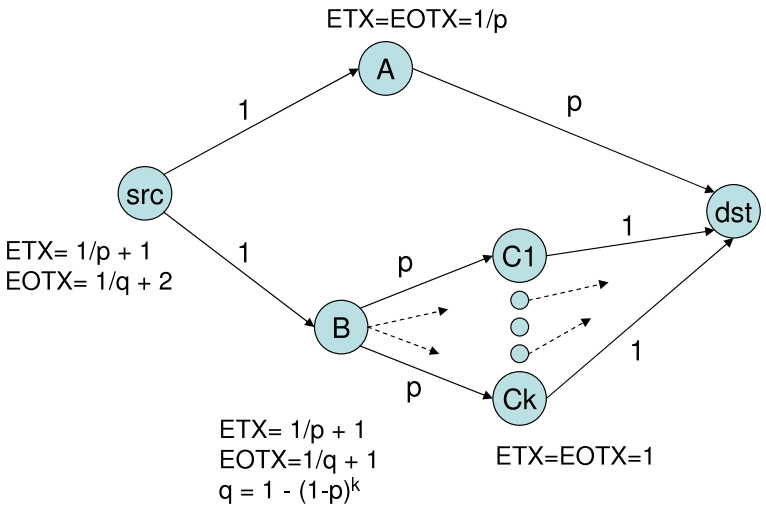


Figure 5-1: **Unbounded Cost Gap.** The numbers on the arrows are marginal success probabilities and it is assumed that the losses are independent. For each node, its ETX and EOTX are listed. ETX-order will always discard B as a forwarder, thus leaving only the path through A. The total cost on this path equals its ETX, which can be driven arbitrarily higher than the EOTX through B.

5.7 EOTX vs. ETX

Although we have shown that the optimal number of transmissions per packet is given by EOTX, both MORE and ExOR use ETX to determine the priority ordering of forwarders, because both pre-date the new metric. Naturally, since EOTX is not computationally harder than ETX, future incarnations of both protocols should use the theoretically exact EOTX.

Yet, one might ask, whether the performance gap caused by the different ordering is at all significant. For a particular network and source-destination pair, we define the gap to be the ratio of the total cost ($\sum_i z_i$) as computed by Algorithm 1 in the case when the ordering is determined by ETX to the case when it is determined by EOTX.

Proposition 6. *The cost gap between ETX- and EOTX-order is unbounded.*

Proof. Consider the network topology shown in Fig. 5-1. The ETX order is $B = src > A > C1 = \ldots = Ck > dst$. In this metric, B is not closer to the destination, hence only A is usable as forwarder. We will then compute a total cost of $\frac{1}{p} + 1$, notably the same as the ETX of the path $src \rightarrow A \rightarrow dst$.

In contrast, by exploiting the multiple independent forwarders, $C1 \ldots Ck$, we can reduce the number of transmissions that B must make to $\frac{1}{1-(1-p)^k}$, so that the total cost of routing

a packet from B is lower than from A when $p < 0.3$ and $k > 1$. The EOTX order is therefore: $src > A > B > C1 = \ldots = Ck > dst$ and the total cost from src is $\frac{1}{1-(1-p)^k} + 2$.

The gap, is the ratio:

$$G = \frac{\frac{1}{p} + 1}{\frac{1}{1-(1-p)^k} + 2}$$

We have that $\lim_{p \to 0^+} G = k$, so by reducing p and increasing k we can drive the ratio to arbitrarily large values. □

Although this result suggests that much improvement is possible, if EOTX is used for forwarder ordering, the contrived topology in Fig. 5-1 is hardly realistic. In particular, it requires a large number of forwarders and very high loss rates. To measure the practical impact, we compute the gap in the topology of our testbed described in §4.1 using marginal success probabilities estimated via probing. We use the same estimates in our protocol evaluation, so we can check if EOTX would affect the forwarder order (and hence z_i) used in our experiments. We compute the gap for each source-destination pair and find that: (1) more than 40% flows would be completely unaffected; (2) among those affected the median gap is 0.2%. We conclude that the difference in the order given by ETX vs. EOTX is insignificant in our experiments.

Chapter 6

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 network coding into the current network stack, exploits the opportunism inherent in the wireless medium, and provides significant performance gains. Field tests on a 20-node wireless testbed show that MORE provides unicast traffic with significantly higher throughput than both traditional routing and prior work on opportunistic routing.

Furthermore, we analyze the theoretical bounds on the performance of wireless network coding and opportunistic routing. We show that the minimum cost wireless unicast flow problem can be solved efficiently despite the exponential number of constraints, and that it could be achieved without network coding but with perfect feedback. The proposed metric - EOTX - serves as the optimal baseline for opportunistic routing protocols.

Bibliography

[1] Calling p2p: Peer-to-peer networks coming to a phone near you, 2005. http://www.econtentmag.com.

[2] Digiweb offers wireless IPTV in Ireland, 2005. http://www.dtg.org.uk/news/.

[3] Ruckus to announce wireless, IPTV deals with 15 telcos, 2006. http://www.eweek.com/article2/0,1895,1989290,00.asp.

[4] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, 2004.

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

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

[7] Pravin Bhagwat, Bhaskaran Raman, and Dheeraj Sanghi. Turning 802.11 inside-out. In *HotNets*, 2003.

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

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

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

[11] N. Cai and R. W. Yeung. Secure Network Coding. In *ISIT*, 2002.

[12] P. Chou, Y. Wu, and K. Jain. Practical Network Coding. In *Allerton*, 2003.

[13] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, 2003.

[14] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MOBICOM*, 2004.

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

[16] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On randomized network coding. In *Allerton*, 2003.

[17] IEEE 802.11 WG, part 11a/11b/11g. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999. http://standards.ieee.org/getieee802/802.11.html.

[18] Sid Jaggi, Michael Langberg, Sachin Katti, Tracy Ho, Dina Katabi, and Muriel Médard. Resilient Network Coding In The Presence of Byzantine Adversaries. In *INFOCOM*, 2007.

[19] A. Jiang. Network Coding for Joint Storage and Transmission with Minimum Cost. In *ISIT*, 2006.

[20] A. Kamerman and G. Aben. Net throughput with IEEE 802.11 wireless LANs. In *WCNC*, 2000.

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

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

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

[24] C.E. Leiserson, R.L. Rivest, T.H. Cormen, and C. Stein. *Introduction to Algorithms*. McGraw-Hill, second edition, 2001.

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

[26] D. S. Lun, M. Médard, and R. Koetter. Efficient operation of wireless packet networks using network coding. In *IWCT*, 2005.

[27] D. S. Lun, M. Médard, and R. Koetter. Network coding for efficient wireless unicast. In *IZS*, 2006.

[28] 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.

[29] MADWiFi: Multiband Atheros Driver for WiFi. http://madwifi.org.

[30] Allen K. Miu, Hari Balakrishnan, and Can E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *MOBICOM*, 2005.

[31] Jun Cheol Park and S.K. Kasera. Expected data rate: an accurate high-throughput path metric for multi-hop wireless routing. In *SECON*, 2005.

[32] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference. In *SIGCOMM*, 2006.

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

[34] Joerg Widmer and Jean-Yves Le Boudec. Network Coding for Efficient Communication in Extreme Networks. In *SIGCOMM WDTN*, 2005.

[35] Starsky H. Y. Wong, Songwu Lu, Hao Yang, and Vaduvur Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *MOBICOM*, 2006.