Multipath Code Casting for Wireless Mesh Networks

Bozidar Radunovic[1], Christos Gkantsidis[1], Peter Key[1],
Steluta Gheorghiu[2], Wenjun Hu[3], and Pablo Rodriguez[4]

[1] Microsoft Research
Cambridge, UK
{bozidar, chrisgk, peterkey}@microsoft.com

[2] Universitat Politécnica de Catalunya
Barcelona, Spain
steluta@ac.upc.edu

[3] Cambridge University
Cambridge, UK
wenjun.hu@cl.cam.ac.uk

[4] Telefonica Research
Barcelona, Spain
pablorr@tid.es

March 2007
Technical Report
MSR-TR-2007-67

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

Abstract

Designing high throughput wireless mesh networks is a challenge, and involves solving interrelated scheduling, routing, and interference problems. In this paper, we exploit the fundamental properties of broadcast medium and path diversity in wireless meshes to implement multipath routing between a source and destination pair. We use network coding for a given unicast source-destination flow to ease the scheduling problem, exploit diversity, and deal with unreliable transmissions. We describe multipath-forwarding algorithms, and show their performance benefits over existing proposals, using simulation, analysis, and a prototype implementation on a small testbed. We propose a rate-scheduling protocol that relies on network coding, which gives over 30% performance improvement for a realistic topology and can double the throughput in certain cases.

1 Introduction

Wireless mesh networks offer a way of creating low-cost and efficient networking, needing no or little infrastructure support. They have applications in diverse settings, ranging from disaster-relief to enabling the wireless office. Yet the inherent variability of the wireless medium, and the inter-related scheduling, routing, and interference problems that need to be solve, make the design of mesh networks that perform well a timely challenge.

One characteristic of wireless mesh networks is that there is usually a large number of paths connecting each pair of source and destination nodes. Moreover, these networks are in nature broadcast: a transmission will be received by many nodes. Therefore it is natural to expect that using multiple paths could improve the performance of the network. Indeed, this was partially shown by Biswas et al. [4]. Previous work has typically used only a few parallel paths, using the same spatial resource, and had to employ a complicated set of scheduling algorithms for packet transmissions. While multipath routing and congestion controlled have been explored in wireline networks, little has been said for wireless networks.

In this work, we use network coding to ease the process of packet scheduling. Moreover, our scheme may use many parallel paths that follow very different routes to reach the destination. Figure 1(a) illustrates the potential improvement of our system. In this figure there are two paths connecting source 1 to destination 4, and the capacity limitations are in the middle of the network. For example, node 2 can forward packets for the flow $1 \rightarrow 4$ at an average rate of r_{23}, and similarly for nodes 5 and 6 with average rate r_{56}. Our goal is to guarantee average rate $r = r_{23} + r_{56}$ for the connection $1 \rightarrow 2$. Assume that source S broadcasts packets at rate $\leq r$. Nodes 2 and 3 will receive (potential overlapping) subsets of those packets. Which packets should nodes 2, 3 forward to their next hops? Can they decide without having to synchronize among them (and avoid the synchronization overhead)?

We use network coding to answer these questions, and achieve the desired throughput gain, by always forwarding packets that are combinations of the previously received packets. The motivational example of Figure 1 is simplistic in the sense that we have suppressed issues that relate to the reliability of the wireless links, for instance we have not discussed the process of guaranteeing that the destination will receive enough packets and be able to decode the original information, and we have ignored fairness issues. We specifically address such issues in this paper, which are connected to choosing the rate of packets in each route r_i and dealing with retransmissions. (Note that we do not adapt the physical layer rate.)

The main ideas in our approach can be summarized in the following:

• Opportunistic routing: Instead of unicasting a packet to a predefined next-hop, a node broadcasts a packet. This packet can be received by one or a few nodes that are next-hops on different paths to the destination. This is similar to [4].

• Load balancing and fault-tolerance: Every source maintains multiple routes to the destination, and when one path becomes more heavily loaded, the

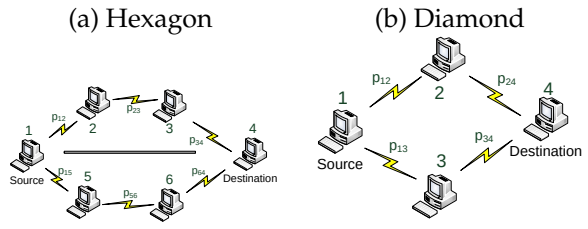

Figure 1: Simple topologies where using multiple paths and network coding increase the performance. In the hexagon topology assume that the bottleneck links are the ones connecting the intermediate nodes 2 and 3, and 5 and 6. In the diamond topology, network coding easies the scheduling of the transmissions of the intermediate nodes 2 and 3, and eliminates the transmission of duplicate packets.

source decreases the load on that path and increases the load of other paths, e.g. [34]. And similarly if one path becomes less reliable. We outline a rate adaptation protocol to implement such load-balancing.

• Network coding: the previous two techniques have been used before, but require a significant protocol overhead to avoid packet duplication, lost, or delay. We introduce network coding at intermediate nodes to randomize information transmission and decrease protocol overhead.

Using these features, we derive our protocol in Section 3. Our scheme encodes packets of the *same* flow. This is unlike the opportunistic coding protocol COPE [16] scheme that encodes packets from multiple simultaneous flows that pass through a particular node. Hence, our proposal is complementary to COPE. We study the behavior of the idea and algorithms and quantify the benefits using numerical simulations (Section 4.1), ns-2 simulations (Section 4.2), and experiments with a lab test-bed (Section 4.3). The main contributions of our paper are:

• We design a system that uses per-flow network coding to exploit the benefits of multipath and opportunistic routing.

• By controlling the rate at which encoded packets are broadcast, we show how it is possible to implement opportunistic routing, direct traffic to appropriate parts of the network (eg, least loaded/ most reliable), and demonstrate how proportional fairness (log utility [18]) increases. We describe a protocol that achieves this by implementing a form of maximal scheduling in a distributed manner [6].

• We evaluate our ideas by simulations using a multipath routing protocol (Section 4), on both simple scenarios (from Figure 1)and a realistic 38-node topology from Roofnet [32]. Our simulations demonstrate gains of more than 30% on average, and more than 100% in some cases.

• We implement a prototype as a proof of concept. It is built on top of Virtual Ring Routing (VRR) [5], and small-scale experiments confirm the performance gains.

2 Motivation and Examples

We now study illustrative examples of topologies (Figure 1) where multipath routing offers advantage. We first look at a diamond topology, and compare multipath routing with and without coding, with single-path routing. We then discuss a hexagonal topology, which allows us to introduce issues of fairness.

2.1 Throughput Benefits

Consider the simple 4-node, 2-hop network shown in Figure 1(b), where node 1 is the source, 4 the destination, and 2 and 3 are relay nodes. Suppose further that node 1 can broadcast to nodes 2 and 3, but that nodes 1 and 4 can not communicate with each other. Our objective is to maximize the information flow f between source and destination, subject to the network constraints, where the flow f is interpreted as the throughput, in say *unique* packets received per unit time.

Specifically, we may assume that source generates fresh packets at a certain rate. These packets are transmitted over the network, encoded at relays, and eventually received by the destination. The received flow rate is the rate at which packets of the source stream can be reconstructed. For more details on network coding, see Appendix.

The routing problem asks which route packets should take: how many should be sent via node 2 and how many via node 3. The scheduling problem determines the packet schedule at each node. Let α_i denote the fraction of time node i is active (hence $\sum_i \alpha_i = 1$). Suppose that nodes broadcast packets at the same rate of 1 packet per time unit, and that packets transmitted by i are successfully received by j with probability p_{ij}. Here $1 - p_{ij}$ is the erasure probability, and incorporates the effects of the physical (PHY) layer, interference, and MAC retransmissions. To ease the description, we set $p_{12} = p_{13} = p$ and $p_{24} = p_{34} = q$ with probabilities assumed independent.

We will use a model similar to the one in [25] to characterize the capacity region of a network with network coding. Define r_{iJ} for a set J to be the information rate on links iJ , that is the rate at which linearly-independent packets from i are received (and may be forwarded) by the nodes in the set J, where J is the set of nodes that can receive information from i.

The r_{iJ} can be interpreted as the carried flow rate. Under our assumptions of independent erasure probabilities, we have

$$r_{i\{J\}} \leq \alpha_i \left(1 - \prod_{j \in \{J\}} (1 - p_{ij})\right). \tag{1}$$

(Clearly $r_{1i} \leq \alpha_1 p$ and $r_{i4} \leq \alpha_i q$ for $i = 2, 3$.) We now explore throughput using different routing and scheduling policies

Multipath routing with Network Coding

Since the same packets may be received by nodes 2 and 3 when node 1 broadcasts, using the union bound (1) we have the constraint

$$r_{1,\{2,3\}} \leq \alpha_1 \left(1 - (1-p)^2\right) = \alpha_1 p_b. \tag{2}$$

which is achievable, where

$$p_b \stackrel{\text{def}}{=} 1 - (1-p)^2$$

is the probability at least one broadcast packet reaches a relay node. The optimal flow has to satisfy $f \leq r_{1,\{2,3\}}$, $f \leq r_{24} + r_{34}$, and with the constraints $r_{1,\{2,3\}} \leq r_{12} + r_{13}$, $r_{1i} \leq r_{i4}$ it is straightforward to show that the unique solution is

$$f_{nc} = \frac{p_b q}{p_b + q}, \quad \alpha_1 = \frac{q}{p_b + q}. \tag{3}$$

Uniqueness follows since we are effectively solving a linear program — given that the p's are assumed fixed, we have a linear objective function maximized over linear constraints, solving for non-negative variables (the α_i and r_{ij}).

Networking coding is implicit in this solution, ensuring that what is received at relay nodes 2 and 3 is useful to node 4; nodes 2 and 3 recode the packets they receive. Indeed, either directly or by adapting the results of [25] we can show that a random linear coding scheme can get arbitrarily close to this rate (with arbitrarily small error probability, provided we can look at large number of packets), where the intermediate nodes generate random linear combinations (over the appropriate field) of packets they receive.

The above relation (3) holds more generally if we have n relay nodes instead of 2, where all links interfere (only one node may transmit at a time). Let 1 be the source, $n+2$ the destination, and $\{2, \cdots, n+1\}$ relays, with $p_{1,n+2} = 0$, $p = p_{1i}$, $q = p_{i,n+2}$ and $p_b = 1 - (1-p)^n$. A rate constraint on the first cut, between the source and the relays, is $f_{nc} \leq \alpha_1 p_b$ and a rate constraint on the second cut is $(1 - \alpha_1)q$. From there we readily obtain the maximal average end-to-end rate $f_{nc} = \frac{p_b q}{p_b + q}$.

Naïve multipath routing

Without network coding, intermediate relay nodes just relay received packets, hence duplicate packets may be received by the destination, and we have to subtract the 'double-counted' packets. As before, $r_{1,\{2,3\}}$ is given from the bound (2), however now the aggregate rate at the destination is bounded above by $(1 - \alpha_1)q - \alpha_1 p^2$, since there are $\alpha_1 p^2$ duplicate packets on the average. In this case, the (unique) maximum rate is given by

$$f_{mp} = \frac{p_b q}{q + 2p}, \quad \alpha_1 = \frac{q}{q + 2p} \tag{4}$$

The equation generalizes when there are n relays, the average rate of duplicated packets is $\alpha_1(np - p_b)$, the rate constraint on the second cut is $(1-\alpha_1)q - \alpha_1(np - p_b)$, hence the maximal average end-to-end rate is

$$f_{mp} = \frac{p_b q}{q + np}. \tag{5}$$

Note that we have $f_{mp} \leq f_{nc}$.

Fixed Routing

With fixed routing, we can only use one path. In this case

$$f_{fr} = \frac{pq}{p+q} \tag{6}$$

with $\alpha = \frac{p}{p+q}$.

Even in this simple scenario, we get a benefit from multipath routing ($f_{nc} > f_{fr}$) provided the links to the relays are not lossless ($p < 1$). The improvement ratio $f_{nc}/f_{fr} = p_b(p+q)/p(p_b+q)$ increases with n, the number of relay nodes, and q, and decreases as p increases— i.e., the less reliable the broadcast links, the bigger the gain. Indeed, with $q = 1$ and small p (very unreliable broadcast links) the ratio is $n + O(p)$, which illustrates the potential benefits. The gain is bounded above by $1/2p + 1/2$.

The benefit of multi-path over single-path routing for various values of p and the number of relays is shown Figure 2, where the relative throughput (f_{nc}/f_{fr}) is plotted. As expected, we see that the benefit is maximal when the first hop is unreliable (p small), second hop is reliable (q large), and the number of path is large. However, in most cases we can expect over 20% improvement, even in such a simple network.

This example is not particularly favorable to multipath routing: in a general network, fixed routing may not choose the best single path route, as it does in this example.

Figure 2 also show the advantage of using network coding rather than naive multipath routing. The figure plots f_{nc}/f_{mp}. We see that network coding always make transmissions more efficient as it eliminates the redundancy, especially when the number of paths is large.

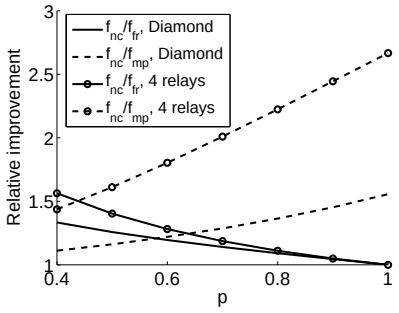

Figure 2: Relative performance improvement using multipath coding over single-path and naive multipath ($q = 0.8$).

2.2 Fairness and scheduling

The previous example assumes that the erasure probabilities are known, in which case the optimal schedule and routing can be simply determined. In practice such probabilities are unknown and variable, and later we show how to construct a scheduling and routing protocol that implicitly estimates such quantities.

We shall also consider the hexagon network, shown in Figure 1 and we suppose 2 and 3 do not interfere with nodes 5 and 6, hence links 2-3 and 5-6 can transmit in parallel.

Suppose now we have an additional flow that uses link 5-6, then how should we be fair to such a flow? If we associate a utility function with a flow, $U(f)$ then seeking to maximize $\sum_f U(f)$ for a given network is equivalent to using a fairness criterion. For example, putting $U(f) = \log f$ give proportional fairness [18], and putting $U(f) = -1/f$ gives TCP type fairness. It turns out that the optimization framework used above carries through if we use concave utility functions U, and we can again show that a unique solution to the scheduling and routing problem exists. We shall implement a form of fairness later by allowing nodes to generate credits according to a rate derived from a utility function.

3 System Design

We now discuss the design choices of our system and the performance implications. We also outline the core of our prototype implementation. Our design strives to (a) guarantee that all packets injected at the source arrive at the destination, (b) use the system resources efficiently to transfer the packets as fast as possible, and (c) ensure fairness among different source - destination flows that are competing for the same resources. To achieve these broad goals, our system uses the following components:

• *Path finding and selection* to identify a set of candidate paths (Section 3.1).

• *Packet encoding and decoding* to divide the packets into generations and code inside the generations (Section 3.2).

• *Error Control* to guarantee that all packets reach the destination (Section 3.3).

• *Rate Control* to decide the next packet to transmit and to ensure fairness among flows (Section 3.4).

We sketch the design of the system in sufficient details to perform simulations and prototype-based evaluation. However, some important protocol details are left for future work, for example, we omit the discussion of imperfect signaling. Moreover, a more detailed evaluation is necessary to study the interaction between our system and higher-level network applications.

3.1 Path selection and multipath forwarding

Current routing algorithms are very efficient for finding the best path (for an appropriate definition of optimality) to route traffic from a source to a destination [5, 15, 30]. Many of them also discover additional (secondary) paths that are used when the primary path fails. In contrast, we need to discover multiple paths to use in parallel to optimize the performance of the routing. These paths are not necessarily disjoint. The use of non disjoint parallel paths has received some attention in [4, 10] but has not been studied comprehensively.

For the simulations performed in the paper, we use a heuristic algorithm described in Section 4.1 for discovering such paths. Our prototype currently uses the underlying routing protocol (VRR) for finding multiple paths [5].

3.2 Packet encoding and decoding

We define a flow as an ordered collection of packets that need to be transferred from one wireless mesh node, the source, to another node, the destination. Observe that our flows may be composed of packets from different upper layer traffic (e.g. TCP), or even from different machines; the only requirement is that they enter and exit the wireless mesh network from the same endpoints.

For the purposes of our simulation studies we have assumed that all packets are of the same size. Our prototype pads short packets to the same size. More efficient solutions remain for future work.

Our system receives from the upper layer a sequence of packets. Those packets are grouped in generations; typically a generation is composed of 32 consecutive packets. The size of the generation has both performance and design implications; for example larger sizes allow for more encoding opportunities and improve the system efficiency, but also require more buffer space. Some discussion on the optimal generation size is given in Section 4.1.1. We encode packets that belong to the same generation only, according to the process described in the Appendix.

The destination tries to decode packets as new encoded packets arrive (see also Appendix). In the worst case, the entire generation will be decoded when enough linearly independent packets from that generation arrive. The decoded packets are delivered to the upper layer sequentially. If the destination is not able to receive enough linearly independent packets, then it uses the retransmission mechanisms described in Section 3.3.

3.3 Error control

In order to solve the linear system to recover coded packets, it is necessary and sufficient to receive the

same number of linear combinations as the number of original packets in the generation. Our system includes two error control mechanisms to ensure the destination receives enough packets - hop-by-hop local recovery and end-to-end retransmissions.

Local recovery is achieved by processing 'passive acknowledgments'. Given the broadcast medium, a sending node can potentially overhear what the next hop transmits and determine if all the next hops have received enough linear combinations between them. If not, the sender retransmits more combinations.

It may happen that some native component is lost at all intermediate nodes, so end-to-end recovery is necessary. This is expensive and should be infrequent. We employ a 'pseudo' end-to-end mechanism. The destination *unicasts* a request toward the source for more combinations for the given generation. If any relay receiving the request has all components for the generation, it could intercept the request and generate retransmissions. Otherwise, the relay forwards on the request toward the source. These retransmissions are also unicast toward the destination. Unicast increases the chance that the request and retransmissions are correctly received.

Note that for both the hop-by-hop and the end-to-end mechanisms, retransmitted packets are in fact new linear combinations of all packets the node has received (or for the source, that the node has sent out).

A final note is due about why we have chosen to implement a retransmission scheme, instead of sending enough redundant packets (which happen naturally using network coding) so that the destination will receive enough packets to decode. We have performed analytical and experimental studies of a scheme without retransmissions and observed that the redundant packets consume a lot of network resources and effectively reduce the performance of the system.

3.4 Rate Control

One of the most challenging problems of implementing a multipath routing algorithm is to decide how to split the rates among the multiple paths. This rate control algorithm needs to adapt fast to variations in link qualities, congestion signals coming from different paths, and react to packet losses.

A simple rate control design is to let the source decide the rate on each path. However, this is inefficient as a delay in feedback from a broken or congested link within the network to the source may be large. Instead, we opt for a distributed rate control algorithm where each relay decide which next-hop node to use for each packet it relays.

Our distributed rate control algorithm is based on the ideas from [24, 33], and, in addition, deals with two issues that are unique in our system. The first relates to the broadcast nature of the transmission; when a relay forwards a packet there are several possible next-hops for a packet and the relay does not know in advance which next-hop node will actually receive the packet. The second issue relates to the use of network coding. In our case every packet is a linear combination of the packets that were available to the sender. As a result the relay does not even need to know which next-hop received what packet (observe that in unencoded transmissions such knowledge is important). Actually, the fact that we forward linear combinations and that we do not care about individual packets easies the design of our rate control algorithm. Note also that we do not adapt physical transmission rates.

With the above motivation in mind, we present a credit-based algorithm for rate control. For each packet generated at the source, the source generates one (node) *credit* and assigns it to the generation the packet belongs to. A credit is further identified by its generation and not by the packet itself. The features of the algorithm are as follows:

1. Credits are created for each packet at the source node, and identified with a generation, not a specific packet.

2. Credits are interpreted as the number of packets to be transferred by the node for a specific flow.

3. Credits are conserved.

4. Credits are declarations of intent, and transferred by a (sending) node before packets are transmitted;

the receiving node only updates such credits when successful packet transmission actually occurs.

5. Nodes also keep track of *transmission-credits*, associated with each subset J downstream nodes, which can be interpreted as the expected number of credits that must be received by *at least* one node in J.

6. When a credit is transferred from a node n to m, transmission credits are increased on all subsets J of downstream nodes that contain m.

7. When a packet is transmitted from node n to m, the transmission credits are decreased for all subsets J of J of downstream nodes that contain m.

8. Backpressure is used to determine where to route packets and where to transfer credits.

We now describe the algorithm in more detail. For node n and flow c let $\mathbf{DST}(n,c)$ be the set of all next-hop (downstream) nodes of node n for flow n. Credits are stored at a node: list $\mathbf{C}(n,c)$ contains all the credits for flow c stored at node n. Each node is responsible for all the credits it has received, and it is obliged to forward each credit to exactly one next-hop node. Credit losses are minimized, and a suitable retransmission scheme is developed, as discussed in Section 3.3.

Transmission credits are associated with broadcast link, as defined in [25]. For all $J \subseteq \mathbf{DST}(n,c)$ we define $\mathbf{TC}(n,c,J)$ to be the list of credits associated with broadcast link (n,J). We want to guarantee that all credits from $\mathbf{TC}(n,c,J)$ will be transmitted to at least one of the nodes from set J.

We also define the cumulative transmission credit

$$\mathbf{CTC}(n,c,m) = \sum_{J \subseteq \mathbf{DST}(nc,c), J \ni m} |\mathbf{TC}(n,c,J)|.$$

It will be used to define the state of congestion of link (n,m). The higher the value of $\mathbf{CTC}(n,c,m)$, the more packets are waiting to be transmitted to node m, hence the more congested the link is.

C update: At any point in time, node n checks the following routing decision

$$\mathbf{RD}: \quad |\mathbf{C}(n,c)| > |\mathbf{C}(m,c)| + |\mathbf{CTC}(n,c,m)| \qquad (7)$$

for each $m \in \mathbf{DST}(n,c)$, and starts transferring one of its credit for flow c to node m. If the routing decision is true for several next-hops, it will select one at random, say node m. The intuition behind this routing decision is as follows: In order to keep network stable, one needs to guarantee that all credit queues will be bounded. To achieve that, we use back-pressure. If either the set of node credit $\mathbf{C}(m,c)$ or the number of cumulative transmission credit $\mathbf{CTC}(n,c,m)$ are large, node n will not forward any more packet to that direction, hence the queue will not build up. We illustrate the convergence issue numerically by example in Section 4.1, Figure 4.

Credit updates are included in headers of actually transmitted packets. That means that node m will be informed about the credits transferred to it before time t only after a packet, broadcast after t, is successfully received by m. At that point, it will update its credit set $\mathbf{C}(m,c)$. More formally, let $z_t^n(n,c,m)$ be the list of all credits transferred from n to m until time t, as seen by node n, and suppose that a packet broadcast by n at time t is successfully received. Then, node m receives $z_t^n(n,c,m)$, and it will add to its credits the set

$$\mathbf{C}(m,c) = \mathbf{C}(m,c) \cup (z_t^n(n,c,m) \setminus z_t^m(m,c,n)).$$

Credits are added to the source node of each flow when fresh packets are created. The rate at which we generate fresh packets will define the efficiency of the scheme and the fairness among flows. We propose a simple scheme, inspired by [24], in which node n, the source of flow c, is allowed to insert $K/\mathbf{C}(n,c)$ credits (and packets) per time unit, where K is an arbitrary constant. The higher K is, the higher the average queue size and delays in the network will be, but also it is less likely that an empty queue will occur, hence the efficiency is higher. We illustrate the performance of the flow control algorithm in Section 4.1, Figure 7.

Observe that the total number of credits change at the sources, since the source insert credits, and at the destinations, where each packet reception reduces the number of credits. (Credits can also be lost by node failures.) Moreover, the rate of credit insertion is controlled by the congestion signals. Hence,

the system does not enter periods of instability when credits are inserted in the system without control.

TC update: Whenever a credit for flow c is transferred from n to m, then a corresponding transmission credit is added to $\mathbf{TC}(n, c, J)$ for all $J \subseteq \mathbf{DST}(n, c)$ such that $m \in J$.

When a packet from flow c is successfully transmitted from node n to node m, we remove one transmission credit corresponding to the packet generation from $\mathbf{TC}(n, c, J)$ (if exists), for all $J \subseteq \mathbf{DST}(n, c)$ s.t. $m \in J$.

Although it may appear that $\mathbf{TC}(n, c, J)$ is simply a union of $\mathbf{TC}(n, c, i)$ for all $i \in J$, this is not always the case. We illustrate the use of $\mathbf{TC}(n, c, J)$ on the example from Figure 3. Initially, $|\mathbf{C}(1, 1)| = 2$, and node 1 transfers one credit to node 2 and one to node 3. Then node 1 broadcasts a packet, which is received by both nodes 2 and 3. This in turn decreases $\mathbf{TC}(1, 1, 2)$ and $\mathbf{TC}(1, 1, 3)$ to empty sets, but we still have $|\mathbf{TC}(1, 1, \{2, 3\})| = 1$. In other words, to successfully finish the credit transfer, we still need to transmit successfully one packet, either to node 2 or node 3.

Scheduling packets: Finally, we need to describe how we transmit packets. We first introduce some notations. Let us denote by $Q(n, J)$ the quality metric of broadcast transmission from n to J. This corresponds to the probability, as measured by n, that at least one node from J will receive a packet transmitted from n. We further define

$$W(n, c) = \sum_{J \subseteq \mathbf{DST}(n,c)} \mathbf{TC}(n, c, J) Q(n, J)$$

and $W(n) = max_c W(n, c), c(n) = \arg\max_c W(n, c)$.

When the routing decision **RD** is satisfied at node n for at least one flow c and its next-hop, node n will calculate $c(n)$ and prepare a random linear combination of the packets buffered for flow c for transmission, from the generation corresponding to the oldest transmission credit. Intuitively, by selecting the flow $c(n)$ the node will maximize its expected reduction in the queue size, similarly to [33]. It will then contend for transmission of the packet in the wireless medium.

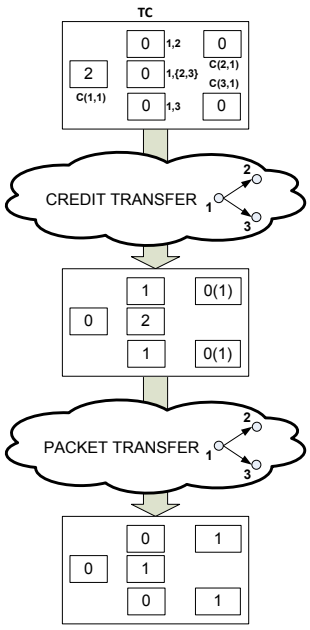

Figure 3: An illustration of rate control: Consider nodes 1, 2 and 3 from the diamond topology depicted in Figure 1(a). First node 1 transfers one credit to each of the nodes 2 and 3. Then node 1 transmits a packet that is successfully received by both nodes 2 and 3. After credit transmission and before packet transmission, nodes 2 and 3 are not yet aware of credit transfers, thus $|\mathbf{C}^2(2, 1)| = 0, |\mathbf{C}^3(3, 1)| = 0$. However, $|\mathbf{C}^1(2, 1)| = 1, |\mathbf{C}^1(2, 1)| = 1$ (values in brackets).

Ideally, to maximize the stability region, nodes should be scheduled to transmit according to the queue sizes **TC** and link qualities Q, as in [24, 33]. This optimal scheduling is hard to implement. A simpler suboptimal scheduling, called *maximal scheduling*, is proposed in [6]. It is polynomial and close to the optimal. In contrast, the scheduling implemented by 802.11 MAC is much more random than the previous two scheduling algorithms and far from the optimal. However, discussing differences between these scheduling algorithms goes beyond the scope of this paper. We will evaluate our approach on different scheduling algorithms: maximal

scheduling in Section 4.1 and 802.11 scheduling in Sections 4.2 and 4.3. We show that our system improves performance in both cases.

4 Evaluation

We analyze the performance benefits of multipath code casting over single-path routing using several methods. We first simulate our protocol using numerical simulations in MATLAB, then we simulate it in ns-2 network simulator, and finally we test it on an experimental test-bed.

4.1 Numerical Simulations

We first want to evaluate how multipath code casting (mc^2) allocates rates over multiple available routes, and quantify the benefits of multi-path routing vs. single-path routing. We also want to understand the effects of different architectural parameters on performance, such as generation size and coding field size. We want to be able to quickly verify these results on relatively large wireless mesh networks (of about 40 nodes).

To achieve this, we developed a simple, event-driven simulator in MATLAB. The input to the simulator is a topology matrix $P = \{p_{nm}\}_{n,m=1\cdots N}$, which defines the probability of successful transmission p_{nm} between any two nodes n, m in a network of N nodes.

We abstract most of the signaling issues. We assume that a node is instantly informed about the success of its transmission. However, we only transfer credits to next-hops once a successful transmission is made. The simulator uses synchronized scheduling. The main scheduling principle is that, if node n is scheduled, then no other node m can be scheduled for which $p_{nm} > 0$ or $p_{mn} > 0$. In this section, we evaluate the results using maximal scheduling [6] described in Section 3.4.

We used a centralized routing algorithm to finds the best paths. We define the quality of path $\rho = \{n_1, \cdots, n_k\}$ to be $C(\rho) = \prod_{i=1}^{k-1} p_{n_i,n_{i+1}}$. We used exhaustive search to find the best 2, 4, or 6 paths according to this metric, for each source-destination pair.

4.1.1 Simple Topologies

We first consider simple networks, the diamond (Figure 1-b) and hexagon (Figure 1-a) topologies. In the diamond topology we set $p_{12} = p_{13} = p$ and $p_{24} = p_{34} = q$, and we fix $q = 0.8$ and we vary p, as we did in Section 2. In the hexagon topology we set $p_{12} = p_{15} = p$ and $p_{23} = p_{34} = p_{56} = p_{64} = q$. We fix $q = 0.8$ and we vary p. Numerical results are shown in Figure 4.

The simulation results match the analysis from Section 2, showing that the proposed matching algorithm achieves the optimal rate control in the case of the diamond and hexagon topologies.

The relative improvement is higher in the case of the hexagon topology. This is due to the fact that links 2-3 and 5-6 can transmit in parallel, and the bottleneck in the hexagon topology is on node 1. In the diamond topology all nodes contend for the MAC access, hence the improvement is less visible. As a general rule we expect that partially disjoint paths may perform better.

We also illustrate the benefit of the coding over naive multi-path routing. Note that naive multipath corresponds to network coding with generation size 1 (thus no possibilities for coding). We also observe that any generation size, larger than 16, performs equally well. However, on larger networks, larger generation sizes may be needed (see Section 4.1.2).

Finally, we illustrate the convergence speed in Figure 4(c). We plot node credits for the diamond topology as a function of the iteration size. We see that the rate control algorithm converges in 50 iterations. However, the actual average end-to-end rate converges in less than 10 slots. We observe similar convergence speed on the Roofnet topology, described next.

4.1.2 Roofnet

In this section we consider the Roofnet network topology [3]. We take the loss measurements avail-

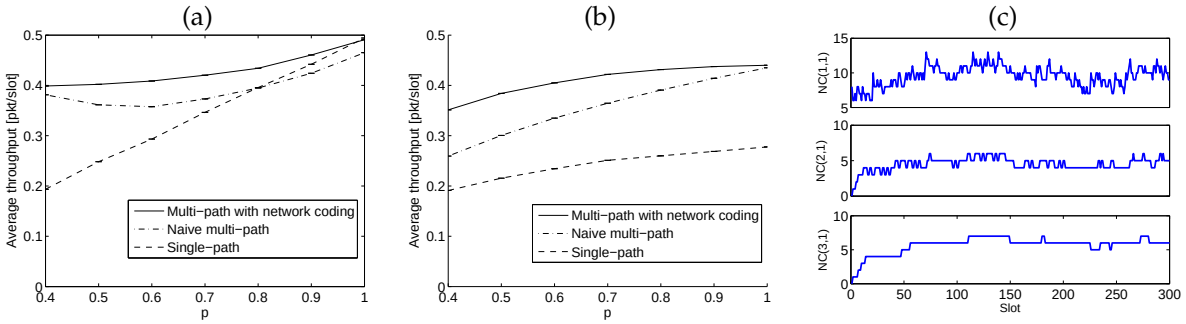

Figure 4: Single-path vs. multi-path for diamond (a) and hexagon (b) topology. For each generation size we run 10 simulations, each transmitting 15MB. Confidence intervals are very small. The improvement of mc^2 is 20% in the case of diamond and almost 100% in the case of hexagon topology. In (c) we illustrate the convergence speed of node credits for the diamond topology. We plot node credits for the diamond topology as a function of the iteration size (note $\mathbf{C}(4,1) = 0$ always).

able from the Roofnet web site [32] for different physical transmission rates, and we feed them to our simulator. We randomly select source-destination pairs. We use GF(2^8) field for coding, and generation size of 32, unless specified otherwise.

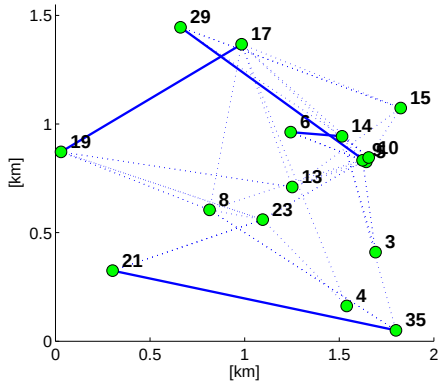

Figure 5: One sample of the Roofnet topology: source-destination pairs 5-29, 21-35, 6-14, 19-17 are connected using solid lines. Links used by our routing protocol are denoted by dotted lines. See [3] and the references therein for the exact topology specification.

Single-flow case: We first consider the Roofnet topology with a single flow, and we evaluate how the performance improves when we increase the number of paths. The results are depicted in Figure 6.

We see that if we use 2 paths, in 20% of cases the improvement of multi-path over single-path is larger than 20%, whereas if we use 4 or 6 paths, for almost 50% of the cases the improvement is larger than 20%. This suggests that the optimal number of paths for Roofnet topology is 4 paths. From Figure 6(b) we see that the performance improvement for 5.5Mbps and 11Mbps PHY broadcast rates is almost the same.

We also see that in a small number of cases (less than 5%), multi-path exhibits worse performance than the single-path. In these cases, the performance drop is higher for a larger number of paths. This is due to the linear dependency in the received packets, as illustrated in Figure 6(c). The number of linearly dependent packets can be reduced if the generation size is increased, which in turns introduces the amount of space needed by coefficients in each packet's header. We believe that the generation size 32 is a good compromise.

Multiple-flow case: The only benefit from mc^2 in the single-flow case comes from the opportunistic routing. When there are several concurrent flows,

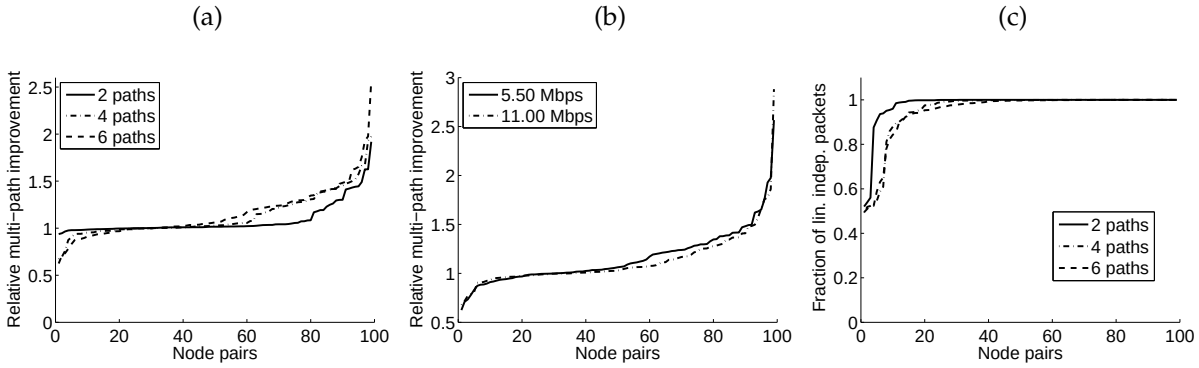

Figure 6: Performance of a single flows on the Roofnet topology. We consider 100 instances of the Roofnet topology with randomly selected source-destination pairs. (a) Cumulative improvement for 2, 4 and 6 path routing over the single-path routing, for PHY rate of 5.5 Mbps. (b) Cumulative improvement of the 6-paths routing over the single-path case for PHY rates of 5.5 and 11 Mbps. (c) Fraction of linearly independent packets received at the destination for the case of 2, 4 and 6 path routings, for PHY rate of 11 Mbps.

mc^2 will also improve the fairness among the end-to-end flows through load-balancing. We illustrate this for the case of 4 concurrent flows in Figure 7.

We first consider the example from Figure 5, and we plot the rates achieved by the 4 flows in Figure 7(a). When all flows use only a single path, we see that the second flow, 21-35, achieves much lower end-to-end rate than the other flows. We see that as we increase the number of paths, the rate of flow 21-35 significantly increases. At the same time the rate of flow 19-17 slightly decreases, but still remains larger than the rate of flow 21-35.

In order to quantify the fairness and the efficiency, we further use the following two metrics. The first is the sum of the end-to-end rates of all the flows in the system, Figure 7(b), which measures the efficiency of the system. The second one is the log utility [18], Figure 7(c), equal to the sum of logs of the end-to-end rates of all the flows in the system, which measures both the efficiency and the fairness of the system.

From the results we see that again in about 50% of the cases the sum of all the rates in the network is increased by at least 20%. We also see that in all of the cases the log-utility of the system has increased.

4.2 Simulations using ns-2

The main drawback of the previous simulation setup are the simplifying assumptions on MAC and PHY. In order to obtain more realistic model of 802.11 MAC and PHY, we used the network simulator ns-2 to simulate the diamond and the hexagon topology.

We did not implement the full rate-control in ns-2, since there are several protocol issues that are out of scope of this paper. Instead, we pre-calculated the credit assignments in the MATLAB simulator and hard-coded them in ns-2.

We simulated the diamond topology for $p_{12} = p_{13} = p_{24} = p_{34} = 0.5$ and the hexagonal topology for $p_{12} = p_{23} = p_{34} = p_{15} = p_{56} = p_{64} = 0.5$. We considered a single CBR flow, varied its rate, and we looked at the effective end-to-end rate. The results are depicted in Figure 8.

For the diamond topology, the system gets saturated at around 150kbps in the case of single-path, and at around 200kbps for the multi-path case, hence the improvement is 42%. In the hexagonal case, the improvement is 35%. Note that the rates in the hexagonal case is lower than in diamond; due to inefficient 802.11 scheduling there are more MAC layer collisions in the hexagonal case.

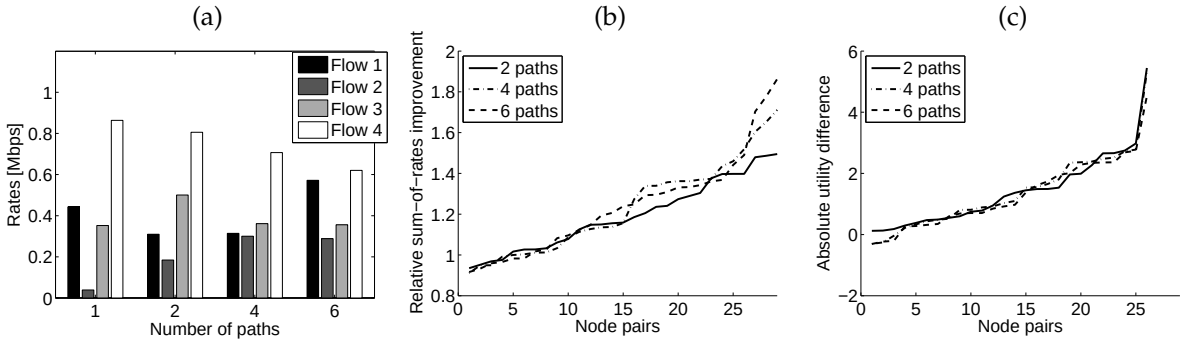

Figure 7: Performance of 4 concurrent flows on the Roofnet topology: (a) Example of end-to-end rate allocation for flows illustrated in Figure 5 (b) Cumulative relative improvement of the sum of rates for 2, 4, and 6 path cases over the single-path case, for PHY rate of 5.5 Mbps. We consider 100 instances of the Roofnet topology with randomly selected source-destination pairs. (c) Cumulative absolute improvement of the log-utility for 2, 4 and 6 path cases over the single-path case, for PHY rate of 5.5 Mbps (here some node pairs are missing as we cannot calculate the utility of a zero rate allocation).

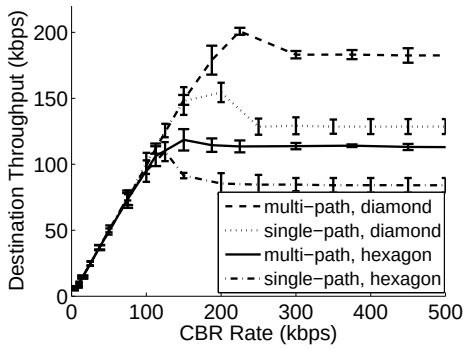

Figure 8: Ns-2 simulation results for the topologies from Figure 1. The x-axis corresponds to the source CBR rate and the y-axis to the effective end-to-end rate.

4.3 Preliminary testbed evaluation

We have built a prototype system implementing a basic sets of mc^2 functionalities, including multi-path forwarding, coding on multiple paths and error control. We focus on small topologies like the diamond topology, where all paths from the source are assumed equally good and hence used simultaneously. The coding rate is split evenly among the paths, and is determined by the number of paths.

We evaluate the performance on a testbed resembling the diamond topology. All nodes are laptops and desktop PCs equipped with Netgear 802.11a/g cards with the Atheros chipset. The experiments are run in 802.11a mode, and the default broadcast rate of 6Mbps is used. We compare throughput of 800K UDP flow transfers for 2 parallel paths and a single path for two configurations of the diamond topology, with high and low loss rates between source and the relay nodes respectively. Different loss rates are produced by moving source very close to or very far from the first hops, and the relays were about 50cm apart in both configurations. For the low loss case, the 2-path scenario showed a 17% throughput improvement, while for the high loss case the improvement was 25%. These results are consistent with the analysis earlier.

Note that the two relays are in the same contention domain, and future work will study the performance on topologies involving relays that cannot hear one another. We will consider desynchronizing their transmissions to avoid collisions at the destinations.

5 Related Work

Multipath routing has a long history in telecommunications. In (wired) wide area networks, multipath routing has generated research activity linked to performance benefits in Internet [8,13] or overlays [2], with authors proposing different routing architectures [20,35,36]. Some of these authors have proposed proportional splitting of traffic across multiple routes, or a form or randomized splitting. There have been recent proposals to combine multipath routing with rate control [14,17,19], jointly optimizing at the network and the transport layer, which is an example of cross-layer optimization [34]. To date, little has been said about how to choose routes, or how to switch between routes when using multipath routing. In our approach, we tie together rate control and routing, without involving the transport layer at this stage. Including and incorporating transport layer controls (such as a TCP-like control) is an extension of our work for future research.

Multipath routing in wireless networks has also received considerable attention, driven by ad-hoc network research. [31] looks at splitting traffic across multiple paths to reduce congestion problems in wireless sensor networks. Marina et al [26] extended AODV to a multipath version, AOMDV, allowing multiple loop-free and disjoint paths, and the general feature of much multipath work has been to only consider disjoint paths (e.g. [22]). The use of semi-disjoint or 'braided' paths is argued for in Moski et al [29] in a mesh context. Theoretical analysis of wireless cross layer design with multi-path routing is done in [24], but without network coding.

Most of the work done in network coding considers multicast transmissions [1,23,27]. Theoretical analysis of network coding applied on unicast transmissions can be found for example in [25]. This result however does not discuss practical routing and rate control issues. A system that proposes a practical use of network coding on unicast transmissions is COPE [16]. Unlike COPE, we apply coding to a single flow, hence our approaches complement.

We said little about how to select routes to choose from. Although our specific implementation used VRR [5], any route discovery protocol such as AODV [30], or DSR [15] can be used, (suitably modified to allow multiple possible routes, e.g. through caching). Draves et al [9] look at three routing metrics to determine best routes, finding that ETX performs best for static configurations, while simple hop-count does better when the sender is mobile. We could use quantities such as ETX in the packet scheduling determination algorithm.

The idea of spatial diversity has been exploited a lot in the field of opportunistic scheduling and MIMO systems. Some theoretical aspects of spatial diversity in networking, as a generalization of multipath, is described in [21]. Some practical implementations are addressed for example in [4, 10, 28]. Multi-Radio Diversity system [28] studies diversity from client to multiple APs and tries to recover packets from corrupted versions received on both APs. [10] also explores the possibilities of recovering partially received packets received on multiple paths but in a sensor network context. Our work is close to ExOR [4], a mesh-networks' routing protocol that exploits spatial diversity. Compared to ExOR, our signalling scheme is much simpler, and in addition we perform load-balancing and ensure fairness.

6 Conclusions

In this paper we have proposed a scheme that uses multiple paths effectively to increase the performance of wireless mesh networks. We have used network coding and a novel credit based algorithm. Network coding eases the process of scheduling packet transmissions, since nodes do not need to coordinate between themselves which packets to forward. The credit-based algorithm ensures that paths with better performance carry most of the traffic, and, also, ensures fairness among multiple flows.

The scheme is motivated by extensive analytical work in the literature.

We have outlined the main challenges in designing our system, and used analysis and simulation to demonstrate the performance improvements of our approach, which can can double the throughput in certain cases. We have also built a prototype, tested on a small network, that further demonstrates the feasibility and advantages of our proposal.

There are several issues that require further investigation, and are currently being planned. We need a more detailed analytical evaluation. We also require a fuller implementation of our ideas in a real system, with a corresponding evaluation in a real environment. However, we believe that our results so far are very encouraging, and more than demonstrate proof of concept and potential benefits of multipath code casting (mc^2).

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46:1204–1216, 2000.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Rao. Improving web availiability for clients with MONET. In *NSDI*, 2005.

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

[4] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *ACM SIGCOMM*, 2005.

[5] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: network routing inspired by DHTs. In *SIGCOMM*, 2006.

[6] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. In *43rd Allerton Conference*, 2005.

[7] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Allerton Conference on Communication, Control, and Computing*, Oct 2003.

[8] C. de Lanois, B. Quoitin, and O. Bonaventure. Leveraging Internet path diversity and network performances with IPv6 multihoming. Technical Report RR 2004-06, Universite catholique de Louvain Tech. report RR 2004-06, 2004.

[9] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM*, 2004.

[10] H. Dubois-Ferrière, D. Estrin, and M. Vetterli. Packet combining for sensor networks. In *Proceedings of ACM SenSys*, 2005.

[11] C. Gkantsidis, J. Miller, and P. Rodriguez. Anatomy of a p2p content distribution system with network coding. In *IPTPS*, 2006.

[12] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *IEEE Infocom*, 2005.

[13] K. Gummadi, H. Madhyastha, S. Gribble, H. Levy, and D. Wetherall. Improving the reliability of Internet paths with one-hop source routing. In *OSDI*, 2004.

[14] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley. Overlay TCP for multi-path routing and congestion control. *IEEE/ACM Trans. Networking*, December 2006.

[15] D. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. In *Mobile Computing*, 1996.

[16] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: practical wireless network coding. In *ACM SIGCOMM*, 2006.

[17] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. *Computer Communication Review*, 35(2):5–12, 2005.

[18] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[19] P. Key, L. Massoulié, and D. Towsley. Combining multipath routing and congestion control for robustness. In *CISS 2006*, 2006.

[20] M. Kodialam, T. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *HotNets*, 2004.

[21] J. N. Laneman and G. Wornell. Exploiting distributed spatial diversity in wireless networks. In *Proceedings of 38th Allerton Conference*, 2000.

[22] S. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *IEEE ICC*, 2001.

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

[24] X. Lin, N. Shroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1452–1463, June 2006.

[25] D. S. Lun, M. Medard, and M. Effros. On coding for reliable communication over packet networks. In *Proc. 42nd Allerton Conference*, 2004.

[26] M. Marina and S. Das. demand multipath distance vector routing in ad hoc networks, 2001.

[27] M. Medard, R. Koetter, and P. A. Chou. Network coding: A new network design paradigm. In *IEEE International Symposium on Information Theory*, Adelaide, Sep 2005.

[28] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proceedings of ACM/IEEE MobiCom*, 2005.

[29] M. Moski and J. Garcia-Luna-Aceves. Multipath routing in wireless mesh networks. In *WiMesh 2005*. IEEE, September 2005.

[30] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *2nd IEEE WMCSA*, 1999.

[31] L. Popa, C. Raiciu, I. Stoica, and D. Rosenblum. Reducing congestion effects in wireless networks by multipath routing. In *ICNP*. IEEE, November 2006.

[32] MIT roofnet - publications and trace data. `http://pdos.csail.mit.edu/roofnet/doku.php?id=publications`, 2005.

[33] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control*, 37(12), 1992.

[34] W.-H. Wang, M. Palaniswami, and S. Low. Optimal flow control and routing in multi-path networks. *Performance Evaluation*, 52, 2003.

[35] W. Xu and J. Rexford. MIRO: Multi-path Interdomain ROuting. In *ACM Sigcomm*, 2006.

[36] R. Zhang-Shen and N. McKeown. "designing a predictable Internet backbone with valiant load-balancing. In *IWQoS*, 2005.

We now present a quick overview of the encoding and decoding operations involved in network coding. The interested reader may find more information in [1, 7, 11, 12, 23, 27] and the references therein.

With network coding information is treated as an algebraic entity on which we can perform arithmetic operations. We view each packet as a vector of elements in a finite field. For example, a packet of size L that is composed as bytes $b_0 b_1 b_2 \dots b_L$, is treated as the vector $(b_0, b_1, \dots, b_L)$, where each b_i is now treated as an element in a Galois Field $GF(2^8)$. For the purposes of this discussion, we assume that all packets have the same size.

Assume that we have n packets; we represent packet i as $\vec{b_i} = (1_{1=i} 1_{2=i} \dots 1_{n=i} b_{i1} b_{i2} \dots b_{iL})^T$, where b_{ik} are the bytes of the original packet, and $1_{j=i}$ are indicator functions that are $1_{j=i} = 1$ iff $i = j$ and $1_{j=i} = 0$ otherwise; the indicator functions are

using to identify the position of the packet in the stream of n packets. The first n entries of the vector (that contain the indicator functions) are also called the *coefficient vector*.

Assume now that the source or an intermediate node has k packets, p_1 through p_k and wants to generate a new encoded packet. In that case, it needs to generate k random elements, say α_1 through α_k, from the base arithmetic field (from $GF(2^8)$ in our case) and compute the new encoded packet as follows:

$$p_{\text{new}} = \alpha_1 \cdot p_1 + \ldots + \alpha_k \cdot p_k \,.$$

(The operations are performed in the base arithmetic field.) Observe that the coefficient vector (i.e. the first elements of the new packet) describes the encoding of the new packet as a linear combination of the original n packets.

When the destination receives n linearly independent packets, i.e. packets with linearly independent coefficient vectors, it can reconstruct the original information using a process that resembles solving a system of linear equations. Moreover, each node can decide whether to receive and store a packet based on the coefficients vector; if the coefficient vector of the received packet can be expressed as a linear combination of the coefficient vectors that are already stored by the node, i.e. the received packet is redundant, then the packet is dropped.

A final note is due about our choice of the base arithmetic field. We have used a Galois Field GF(2^8) which has a size of 256 elements. The size of the field is large enough to guarantee good diversity (at least for networks of sizes of a few hundred nodes) and is small enough to allow efficient encoding and decoding operations.