# Multipath Code Casting for Wireless Mesh Networks

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

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

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

[3] Telefonica Research
Barcelona, Spain
{pablorr,steluta}@tid.es

## ABSTRACT

Designing high throughput wireless mesh networks involves solving interrelated scheduling, routing, and interference problems. In this paper, we exploit the broadcast properties and the path diversity of wireless meshes to implement an efficient multipath routing protocol, Multipath Code Casting ($MC^2$).

In contrast to prior work in opportunistic routing, which required strong coordination across nodes to prevent information repetition, our design is based on network coding and does not require node coordination. Moreover, it provides a unified framework to deal with data transmissions across multiple and, often, unreliable transmission paths. Our design also includes a novel rate-scheduling algorithm that guarantees (proportionally) fair allocation of resources across multiple (multipath) flows, ensures that data use the paths with the best performance, and prevents information overflow by controlling the data rate across each path. Using simulations and a prototype implementation, we show that our algorithms provide over 30% performance improvement compared to traditional singlepath approaches when applied to realistic and other exemplar topologies; in some scenarios, our approach can even double the throughput. Our approach also performs better than 20% compared to other multipath routing schemes.

## 1. INTRODUCTION

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

A characteristic of wireless mesh networks is the large number of paths that typically connect a source to a destination node. It is natural to expect that using multiple paths could improve the performance of the network. However, forwarding data across multiple paths opens the door for new unfairness and congestion control issues. While joint multipath routing and congestion control have been explored in wireline networks, little has been said for wireless networks. In addition, wireless mesh networks are also broadcast and dynamic: (a) a transmission may be received by multiple nodes, and (b) the subset of receiving nodes may change frequently. Biswall et al. proposed opportunistic routing to take advantage of the broadcast and dynamic nature of wireless mesh networks [3]. However, opportunistic routing requires strong coordination between intermediate nodes; otherwise, intermediate nodes may receive and forward the *same* packets wasting wireless resources.

In this paper we propose Multipath Code Casting ($MC^2$), a new approach to opportunistic forwarding. The key idea is the use of network coding at intermediate forwarders, combined with a novel rate and congestion control algorithm. Intermediate nodes forward random linear combinations of the packets they receive, and use a credit-based scheme to ensure that multiple paths can be fairly and efficiently used in parallel, to deliver linearly independent information to the destination without the need for strong node coordination.

Figure 1(a) illustrates the potential of our system. In this figure there are two paths connecting source 1 to destination 4. Assume that the combined capacity of the intermediate paths (i.e., $r_{23}$ and $r_{56}$) is equal to that joining node 1 to nodes 2 and 3. Our goal is to guarantee an 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 potentially overlapping subsets of those packets. If nodes 2 and 3 simply forward the packets they receive, many redundant packets will be sent, and nodes need to coor-


*CoNEXT*'07, December 10-13, 2007, New York, NY, U.S.A.

dinate to prevent that. However, using network coding at each intermediate forwarding node (e.g., 2, 3, 5, and 6), one can achieve a rate of $r$ at the destination by simply collecting enough linearly independent combinations, produced as combinations of previously received packets, and then decoding.

The motivational example of Figure 1(a) is simplistic in the sense that we have suppressed issues that relate to the reliability of the wireless links. For example, we have not described how to guarantee that the destination receives enough independent packets to be able to decode the original information. We address such issues in this paper, and specifically focus on how to chose the rate of packets on each route $r_i$ to maximize throughput while ensuring fairness, and how to deal with retransmissions.

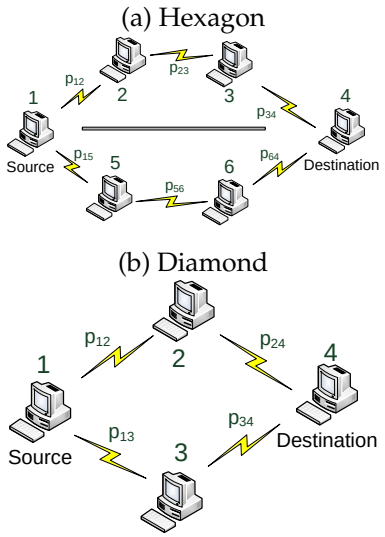


Figure 1: Simple topologies that demonstrate the benefit of using multiple paths and network coding. In the hexagon topology assume that the bottleneck links are the ones connecting the intermediate nodes 2 and 3, and 5 and 6. In the diamond topology, network coding eases the scheduling of the transmissions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

The main contributions of this paper are:

• Opportunistic forwarding using network coding: Due to the nature of the wireless medium, packet transmissions may be received by multiple nodes. To avoid the problem of coordinating which node forwards which packet, we use per-flow network coding at the intermediate forwarding nodes. As a result, we reduce the coordination cost, we eliminate redundant retransmissions, and we achieve a system that scales well for large networks. Network coding provides a unifying framework for multipath and opportunistic routing in wireless mesh networks.

• Multipath rate/congestion control and fairness: We present a novel framework for multipath congestion control and rate allocation across paths. Our algorithm ensures optimal traffic allocation across paths, directing load to those paths that can benefit the most at any time without saturating the network or being unfair to other flows. By controlling the rate at which encoded packets are broadcast, we show how it is possible to direct traffic to appropriate parts of the network and demonstrate how to achieve proportional fairness (log utility [14]) by implementing a form of maximal scheduling in a distributed manner [6].

• We evaluate our ideas using simulations under simple intuitive scenarios and a realistic 38-node topology from Roofnet [24]. Our simulations demonstrate gains of more than 30% on average, and more than 100% in some cases. We also implement a prototype as a proof of concept. It is built on top of Virtual Ring Routing (VRR) [4]. Small-scale experiments confirm the performance gains and demonstrate the practicality and feasibility of using network coding in wireless mesh networks with general random linear codes, as opposed to simple XORs.

## 2. MOTIVATION AND EXAMPLES

We now study two illustrative examples (Figure 1) that motivate the multipath network coding approach. We first look at a diamond topology and compare single path routing with multipath routing with and without network coding. We then discuss a network with hexagonal topology, which carries traffic from two flows; this example allows us to introduce issues related to faireness.

### 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. 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, 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. The *routing problem* needs to decide which route packets should take: e.g. how many should be sent via node 2 and how many via node 3. The *scheduling problem* determines which packet should be delivered by which node.

Let $\alpha_i$ denote the fraction of time node $i$ is active, hence $\sum_i \alpha_i = 1$. Note that this constraint is imposed due to a specific topology of Figure 1(b) where all links mutually interfere; the corresponding constraints would be more complex in a general case. Suppose that nodes broadcast packets at the same rate of 1 packet per time unit, and that packets broadcasted by $i$ are successfully received by $j$ with probability $p_{ij}$. We assume that the $p_{ij}$'s are independent. 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 $p_{24} = p_{34} = q$.

We will use a model similar to the one in [19] 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 develop analytical formulations for different routing and scheduling policies:

**Multipath routing with Network Coding:** Since the broadcasts from node 1 may be received by nodes 2 and 3, the union bound (1) gives the constraint

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

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 $\alpha_i$ and $r_{ij}$).

Networking coding is implicit in this solution. The solution assumes that the transmissions from relay nodes 2 and 3 are of interest to node 4 (i.e. no duplicate transmissions). Indeed, either directly or by adapting the results of [19] we can show that a random linear coding scheme can get arbitrarily close to this rate with arbitrarily small error probability. The result assumes that nodes 2 and 3 encode using a sufficiently large number of packets (i.e. they have to receive many packets from 1 before they start transmitting), and the encoding operations happen over a sufficiently large arithmetic field.

Eq.3 generalizes to $n$ relay nodes instead of 2; again we assume that all relay nodes interfere and, hence, 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 between the source and the relays is $f_{nc} \leq \alpha_1 p_b$ and a rate constraint between the relays and the destination is $(1 - \alpha_1)q$. From there we readily obtain the maximal average end-to-end rate

$$f_{nc} = \frac{p_b q}{p_b + q}. \tag{4}$$

**Naïve multipath routing:** Without network coding and without coordination, 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 we need to discount the duplicate packets. Since there are $\alpha_1 p^2$ duplicate packets on average, the aggregate rate at the destination is bounded above by $(1 - \alpha_1)q - \alpha_1 p^2$. In this case, the (unique) maximum rate is given by

$$f_{mp} = \frac{p_b q}{q + 2p}, \quad \alpha_1 = \frac{q}{q + 2p} \tag{5}$$

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

$$f_{mp} = \frac{p_b q}{q + np}. \tag{6}$$

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

**Fixed Routing:** With fixed routing, we can only use one path. In this case $f_{fr} = \frac{pq}{p+q}$ with $\alpha_1 = \frac{p}{p+q}$.

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 $1/2p + 1/2$.

**Comparison:** 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), the second hop is reliable ($q$ large), and the num-

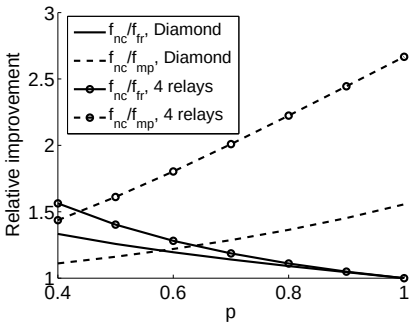


Figure 2: Relative performance improvement using multipath coding over single-path and naïve multipath routing ($q = 0.8$).

ber 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 shows the advantage of using network coding over naïve multipath routing. Network coding always makes transmissions more efficient by eliminating the redundancy, especially when the number of paths is large.

## 2.2 Fairness and scheduling

The previous example assumes knowing the erasure probabilities, hence 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 forwarding protocol that implicitly estimates these.

Let us now consider the hexagon network, shown in Figure 1 and assume 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, setting $U(f) = \log f$ give proportional fairness [14], and putting $U(f) = -1/f$ gives TCP style fairness. In fact 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. Later, we will propose a new algorithm to implement fairness by allowing nodes to generate credits according to rates derived from a utility function.

# 3. SYSTEM DESIGN

$MC^2$ is designed as an extension on the forwarding path between the data link layer and the network layer. Our system strives to (a) guarantee packet delivery from source to destination with low overhead, (b) efficiently use the systems resources to achieve maximum throughput, and (c) ensure fairness among competing flows. These goals lead to the following system components: Multipath forwarding, packet coding and decoding, error control and rate control. We discuss them in detail in this section. Given the space constraint, we have omitted some implementation details (see also [23]).

## 3.1 Multipath forwarding

We rely on the underlying routing protocol to find a set of static paths, and dynamically select the actual paths to use. Ideally, our objective is to select a set of paths that collectively give the best performance; in practice we fix a limit on the number of paths, say $k$, and pick the best $k$ paths. Note, however, that we do not require disjoint paths. that we are only concerned with the next hop for a node on each path, so end-to-end disjointness of paths is not necessary. We expect that the typical number of possible next hops per nodes is between 2 and 4. The convention notion of 'path' is only used for illustration. The paths are chosen based on their quality, and we study the effects of the number of paths used with simulations (Section 4.1). Delivering packets to multiple next hops is easily achieved by exploiting the broadcast medium.

## 3.2 Packet encoding and decoding

$MC^2$ employs random linear codes [17], which are shown to be sufficient for typical network coding scenarios. The source and relay nodes generate network encoded packets using the original packets or other encoded packets; the destination collects encoded packets and, after receiving a sufficient number of them, recovers the original information by a process called decoding.

A packet of $L$ bytes $b_0 b_1 b_2 \ldots b_L$, is viewed as an algebraic vector $(b_0, b_1, \ldots, b_L)^T$, where each $b_i$ is treated as an element in a Galois Field. Assume $n$ original packets, original packet $i$ is represented as

$$\vec{b_i} = [I_{1=i}, I_{2=i}, \ldots, I_{n=i}, b_{i1}, b_{i2}, \ldots, b_{iL}]^T$$

where $b_{ik}$ are the bytes of the original packet, and $I_{j=i}$ are indicator functions, such that $I_{j=i} = 1$ iff $i = j$, and 0 otherwise; the indicator functions identify the position of the packet in the stream of $n$ packets. The first $n$ entries of the vector, which contain the indicator functions, are collectively called the *coefficient vector*.

Generating a new encoded packet $B_{\text{new}}$ using $k$ packets $B_1, \ldots, B_k$ is equivalent to computing the linear combination:

$$B_{\text{new}} = \alpha_1 \cdot B_1 + \ldots + \alpha_k \cdot B_k \,.$$

All operations are performed in the base arithmetic field. The packets $B_i$ can be either original packets ($b_i$'s), or other encoded packets. 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 $n$ original packets. Multiple linearly independent combinations can be generated from the same packets in a *decentralized* fashion, with high probability, by drawing random coefficients from the field [7, 11]; this is a highly desired feature since it implies that the nodes can make coding decisions independently.

Decoding resembles solving a system of linear equations. The destination needs to collect $n$ linearly independent packets. Using the coefficient vectors in each of them, it can determine how to transform the (encoded) packets to reconstruct the original packets.

Our base arithmetic field is a Galois Field $GF(2^8)$ containing 256 elements. This field size is large enough to guarantee good diversity, for networks of a few hundred nodes, and yet is small enough to allow efficient encoding and decoding operations.

We apply the above coding scheme to packets flowing between the same source and destination pair in the wireless mesh networks. These packets could be from different upper layer traffic. Furthermore, packets are divided into generations of 32 consecutive packets each, and only those in the same generation are coded together. Coding per generation has multiple practical advantages such as limiting the overhead of decoding, and reducing the state at the intermediate nodes.

For simulations we assume packets are of the same size, while our prototype implementation pads short packets to ensure this. For convenience we insert the length of the original packets in the beginning of their payloads (hence, the length becomes part of the payload). The length can be retrieved after successful decoding.

Decoding is performed at the destination as new packets arrive. In the worst case, the entire generation will be decoded when (re)transmissions for the generation complete. Decoded packets are delivered to the upper layer according to their initial ordering.

For efficiency, relays and the destination check for and drop linearly dependent incoming packets.

## 3.3 Error control

To recover original packets, it is necessary and sufficient for the destination to receive the same number of linear combinations as the number of original packets in the generation. Our system includes two mechanisms to ensure this, hop-by-hop local recovery and end-to-end retransmissions.

Local recovery is achieved by processing acknowledgments. Acknowledgments can be passive, by overhearing next hop transmissions, or active, by piggybacking acknowledgments in packets flowing in the inverse direction. A source can determine from a passive acknowledgment: (a) if each next hop has received enough packets (the number of packets is determined through credit exchange, as explained in Section 3.4), and (b) if all packets have been received by at least one next-hop. For (b), note that packet IDs are readily available in the coding header of the overheard packet. If either is not met, the sender generates and broadcasts new packets. The local recovery mechanism is included because the broadcast mode of 802.11 does not include reliability provisions.

Local recovery is not enough to protect against node failures, and other failures that disconnect the network. Therefore, end-to-end recovery is also necessary. However, end-to-end recovery is expensive and should be infrequent. We employ a *pseudo* end-to-end scheme. On a timeout, the destination *unicasts* a request for more combinations of a given generation toward the source for the given generation. By unicast, we mean the 802.11 unicast mode, which includes MAC-level acknowledgments and retransmissions. The request will be intercepted by any relay that has all the packets of the generation (or, the source in the worst case); in that case the relay (or, the source) will generate retransmissions. Relays that do not have enough packets forward the request.

The end-to-end scheme cannot detect the loss of all packets from the last generation. This event should be infrequent. We rely on upper layer protocols to deal with such events.

In both recovery mechanisms, retransmissions involve generating *new* linear combinations of all packets that are available in the node that produces the retransmission. As a result, the retransmitting node does not need to know which packet is missing (as in the case of not using network coding); as a result network coding simplifies the design of the retransmission mechanism.

## 3.4 Rate Control

One of the main challenges faced by a multipath routing scheme is to decide how to split the rates among the multiple paths. This rate control algorithm must adapt quickly to varying link qualities, congestion signals from different paths, and packet losses. Rate control is also needed to prevent forwarders from producing more information than they received. This could create an information overflow with many linearly dependent packets flowing over the mesh across different paths. Rate control should also allocate network resources fairly among competing flows.

A simple approach is for the source to decide the rate on each path. This is inefficient, however, since there might be a large delay in sending feedback from a bro-

ken or congested link to the source. Instead, we opt for a distributed approach, where each node decides which next hops to send each packet to.

Our distributed rate control algorithm is based on the ideas from [18,25], and, in addition, addresses two issues unique in our system. First, the broadcast nature of the transmissions means that, 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 will actually receive the packet. Second, in our case every packet is a linear combination of the packets that were available to the sender. Therefore, it is not even necessary for the relay to know precisely which next hop received a particular packet. This observation easies our design of the rate control algorithm. Note that knowledge of individual packets is a requirement with uncoded transmissions.

With the above motivation in mind, we now present a *credit-based* algorithm for rate control, with the following features:

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

2. Credits are interpreted as the number of packets a node should transfer for a specific generation.

3. Credits are conserved inside the network.

4. Credits are declarations of intent, and transferred, as packet annotations, 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 $J$. These are used to aid the hop-by-hop retransmissions.

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 $n$ and successfully received by a set $J$ of downstream nodes, the transmission credits are decreased for all subsets $K$ of $J$.

8. Back-pressure is used to determine where to forward packets and transfer credits.

One can show that our algorithm stabilize the network whenever it is possible. We omit the proof due to lack of space and refer interested readers to [23].

We now describe the algorithm in more detail. For node $n$ and flow $c$ let $\mathbf{DST}(n,c)$ be the set of all nexthop nodes, downstream of node $n$ for flow $c$. Credits are stored as: 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 nexthop node. Credit losses are minimized, using the retransmission schemes of Section 3.3.

Transmission credits are associated with broadcast link, as defined in [19]. 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 $(n,J)$ (the typical number of next-hop neighbors will be 2-3 hence we expect less than 10 broadcast links per flow). We want to guarantee that all credits from $\mathbf{TC}(n,c,J)$ will be transmitted to at least one node 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 on 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 time, node $n$ checks the following forwarding decision

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

for each $m \in \mathbf{DST}(n,c)$, and starts transferring one credit for flow $c$ to node $m$. If the forwarding decision is true for several nexthops, one is selected at random, say node $m$. The intuition behind this decision is that, in order to keep the network stable, one needs to guarantee that all credit queues are bounded. This is achieved through back-pressure. If either the set of node credits $\mathbf{C}(m,c)$ or the number of cumulative transmission credits $\mathbf{CTC}(n,c,m)$ is large, node $n$ will not forward any more packet in that direction, to avoid the queue building up. Note that only the inequality in Eq. 7 matters, the individual values of $\mathbf{C}$ and $\mathbf{CTC}$ are not important. This is common in back pressure flow control algorithms (see [25] for a more detailed analysis). We illustrate the convergence issue numerically in Section 4.1, Figure 4.

Credit updates are included in headers of actually transmitted packets. This means that node $m$ will not be informed of the credits transferred to it before time $t$, until 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 generated on 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 [18], 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 the $K$, the higher the average queue size and delay in the network. Higher $K$ also makes it less likely that an empty queue will occur, resulting in higher efficiency. The performance of the flow control algorithm is depicted in Section 4.1, Figure 7.

Observe that the total number of credits change at the flow source, since the source insert credits, and at the destination, where each packet reception reduces the number of credits. Credits can also be lost in 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 node $n$ to $m$, a corresponding transmission credit is added to $\mathbf{TC}(n,c,J)$ for all $m \in J \subseteq \mathbf{DST}(n,c)$.

When a packet from flow $c$ is transmitted from node $n$ and successfully received by a set $J$ of downstream nodes, we remove one transmission credit corresponding to the packet generation from $\mathbf{TC}(n,c,K)$, if it exists, for all $K \subseteq J$.

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 $\mathbf{TC}(n,c,J)$ for the example in Figure 3. Initially, $|\mathbf{C}(1,1)| = 2$, and node 1 transfers one credit each to node 2 and 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 need to successfully transmit one more packet, to either node 2 or node 3.

The separation simplifies the design in the sense that transport credits **TC** provide channel quality estimates. Also, this implementation is provably stable [23]. Observe that the credit and packet transfers in Figure 3 occur in parallel and do not need to be synchronized.

**Scheduling packets:** Finally, we describe how to transmit packets. 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 $W(n) = max_c W(n,c), c(n) = \arg\max_c W(n,c)$.

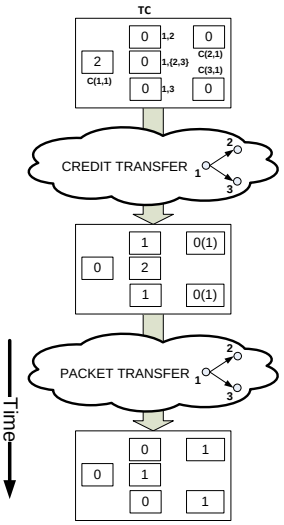


Figure 3: An illustration of rate control: Consider nodes 1, 2 and 3 from the diamond topology in Figure 1(a). First node 1 transfers one credit each to nodes 2 and 3. Then node 1 transmits a packet that is successfully received by both nodes 2 and 3. After credit transmission and before packet transmission, 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).

When the forwarding decision **RD** is satisfied at node $n$ for at least one flow $c$ and its nexthop, node $n$ will calculate $c(n)$ and prepares 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 the expected reduction in its queue size, similarly to [25]. It will then contend for medium access to transmit the packet.

Ideally, to maximize the stability region, nodes should be scheduled to transmit according to the queue sizes **TC** and link qualities $Q$, as in [18, 25]. However, this optimal scheduling is hard to implement. A simpler polynomial and near optimal scheduling, called *maximal scheduling*, is proposed in [6]. In contrast, the 802.11 MAC scheduling is much more random and far from the optimal. While discussing differences between these scheduling algorithms goes beyond the scope of this paper, we will use both maximal scheduling (Section 4.1) and 802.11 scheduling (Sections 4.2 and 4.3) in evaluations. We show that our system improves performance in both cases.

# 4. EVALUATION

We evaluate the performance benefits of $MC^2$ over single-path routing using numerical simulations in MATLAB, ns-2 simulations and preliminary testbed experiments. We take advantage of the different evaluation environments to highlight the performance of different components in our system.

## 4.1 Numerical Simulations

We first want to quickly quantify the benefits of multi-path forwarding over a single-path approach and evaluate the rate allocation of $MC^2$ on different wireless mesh topologies of 40 nodes or more. Therefore, we developed a simple event-driven simulator in MATLAB. We also study the effects of architectural parameters, such as the generation and coding field sizes, in the same framework.

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 away most of signaling issues, and assume that a node is instantly informed about the success of its transmission. However, credits are only transferred to nexthops once transmissions are successful. The simulator uses synchronized scheduling: if node $n$ is scheduled, then no other node $m$, for which $p_{nm} > 0$ or $p_{mn} > 0$, can be scheduled. Furthermore, we use maximal scheduling [6] described in Section 3.4.

The path quality $\rho = \{n_1, \cdots, n_k\}$ is defined as $C(\rho) = \prod_{i=1}^{k-1} p_{n_i,n_{i+1}}$. We used centralized exhaustive search to compute the best 2, 4 or 6 paths according to this metric for each source-destination pair in the given topology.

### *4.1.1 Simple Topologies*

We start with the diamond (Figure 1-b) and hexagon (Figure 1-a) topologies as representative simple networks. In the diamond topology we set $p_{12} = p_{13} = p$ and $p_{24} = p_{34} = q$, as 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 vary $p$ for both. Numerical results are shown in Figure 4.

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

The relative improvement is higher for the hexagon topology. This is due to parallel transmissions along links 2-3 and 5-6, which shifts the bottleneck in the hexagon topology to node 1. In the diamond topology all nodes contend for medium access, hence the improvement is less visible. Generally, we expect disjoint paths to perform better.

The benefit of using coding over naïve multipath routing is shown in Figure 4(a) and (b). Observe that in both cases the optimal performance is 0.5 packets per slot (assuming no packet losses). The reason is that the maximum transmission rate is 1 packet per slot and that nodes compete for the wireless channel. As a result, the source cannot transmit while any of the intermediate nodes transmit; similarly only one intermediate node can transmit to the destination at any point in time. The benefit depends on the topology and the characteristics of the wireless channel. However, in both cases network coding performed significantly better than no coding and naïve multi-path. We have also experimented with different generation sizes (not shown in Figure 4), and observed that any generation size, larger than 16, performs equally well. However, larger generation sizes may be needed for larger networks (see Section 4.1.2).

Finally, Figure 4(c) shows the convergence speed for the diamond topology. We plot node credits as a function of the iteration size. The rate control algorithm converges in around 50 iterations, while 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*

We next consider the Roofnet topology [2] as a more realistic example. Real-world loss measurements for different physical transmission rates are available from the Roofnet web site [24], which we fed to our simulator. Unless specified otherwise, we randomly select source-destination pairs and use GF($2^8$) and generation size 32 for coding.

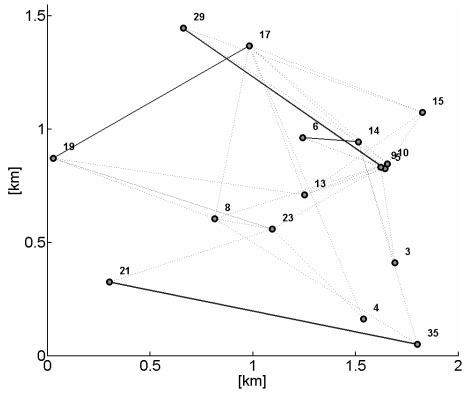


Figure 5: One sample of the Roofnet topology: source-destination pairs 5-29, 21-35, 6-14, 19-17 are connected using solid lines. Links used by our routing algorithm are denoted by dotted lines. See [2] and the references therein for the exact topology specification.

**Single-flow case:** We first study the throughput of a single flow in the Roofnet topology as we increase the number of paths, and summarize the results in Figure 6.

In 20% of cases the improvement of 2-path over single-path exceeds 20%, and such improvement can be obtained in more than half the cases using 4 or 6 paths. This suggests that the optimal number of paths for Roofnet topology is 4. Figure 6(b) shows that the performance improvement for 5.5 Mbps and 11 Mbps PHY rates is almost the same.

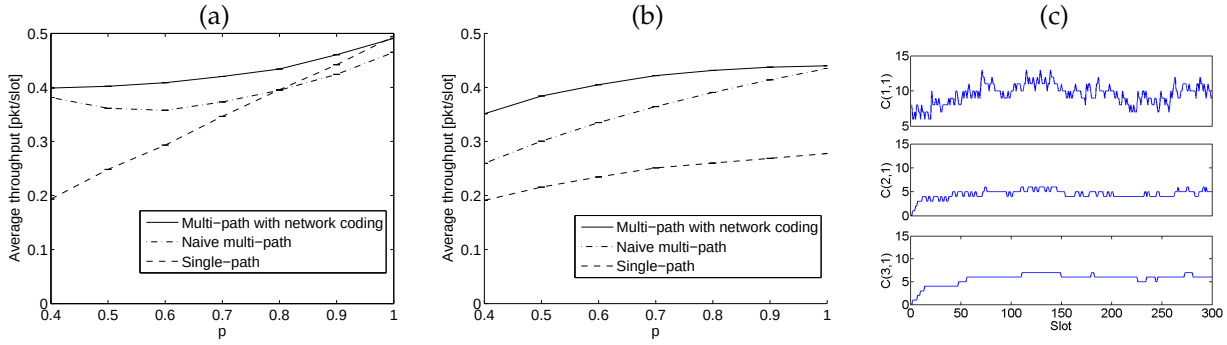


Figure 4: Single-path vs. multipath for diamond (a) and hexagon (b) topologies. For each value of $p$ we run 10 simulations, each transmitting 15MB. Confidence intervals are very small. The improvement of $MC^2$ is 20% in the diamond and almost 100% in the hexagon topology. Part (c) illustrates the convergence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

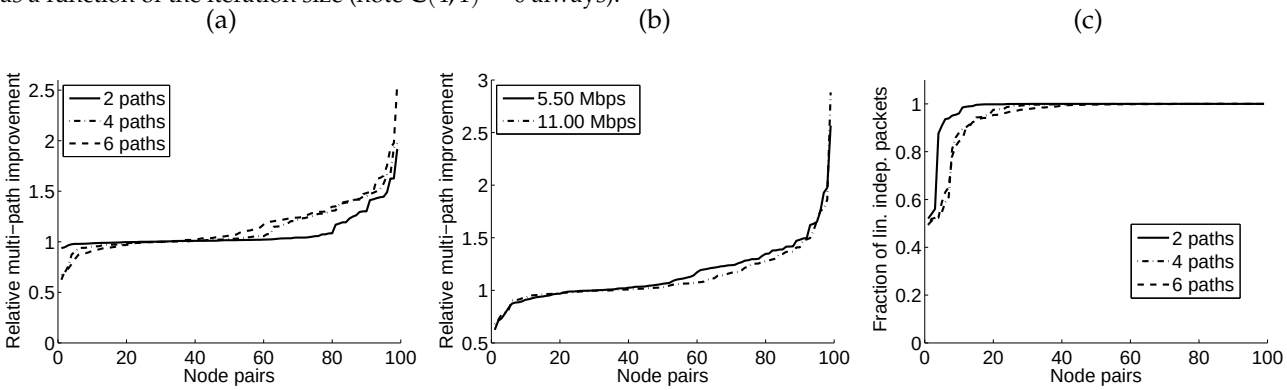


Figure 6: Performance of a single flow in 100 instances of the Roofnet topology, with randomly selected source-destination pairs. (a) Cumulative improvement from using 2, 4 and 6 paths over single-path routing, for PHY rate 5.5 Mbps. (b) Cumulative improvement from using 6 paths over the single-path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

On few occasions (less than 5% of the cases), multipath exhibits worse performance than the single-path counterpart, and the performance worsens as the number of paths increases. This is due to two reasons: dependency in received packets and suboptimal scheduling (see Section 3.4). The linear dependency in the received packets is illustrated in Figure 6(c). The number of linearly dependent packets can be reduced if the generation size is increased, which nevertheless increases the coding header size. We believe that the generation size of 32 is a good compromise.

**Multiple-flow case:** The only benefit from $MC^2$ in the single-flow case is from path diversity, which reduces the overall packet loss. When given several concurrent flows, $MC^2$ will also improve fairness among them through load-balancing. We illustrate this for 4 concurrent flows in Figure 7.

Using the example from Figure 5, we plot the throughput of the 4 flows in Figure 7(a). When all flows use only a single path, the second flow, 21-35, achieves much lower end-to-end rate than the other flows. As the number of paths increases, the rate of flow 21-35 significantly increases. Meanwhile, the rate of flow 19-17 slightly decreases, but remains larger than the rate of flow 21-35.

In order to quantify fairness and efficiency, we use two more metrics. The first is the aggregate network throughput, i.e., the sum of the end-to-end rates of all flows in the system (Figure 7(b)), which measures the efficiency of the system. The second one is the log utility [14], i.e., the sum of logs of the end-to-end rates of all flows in the system, which measures both the efficiency and the fairness of the system (Figure 7(c)).

The results show that, again, in about 50% of the cases the aggregate network throughput increased by at least 20%. In all cases the log-utility of the system increased.

## 4.2 ns-2 simulations

The main drawbacks of the previous simulation setup are the simplifying assumptions for the MAC and PHY. Therefore, we turn to ns-2 for more realistic MAC and PHY models.

For ns-2 simulations, we use CBR traffic over UDP sent at 1 Mbps and two-way ground reflection model.

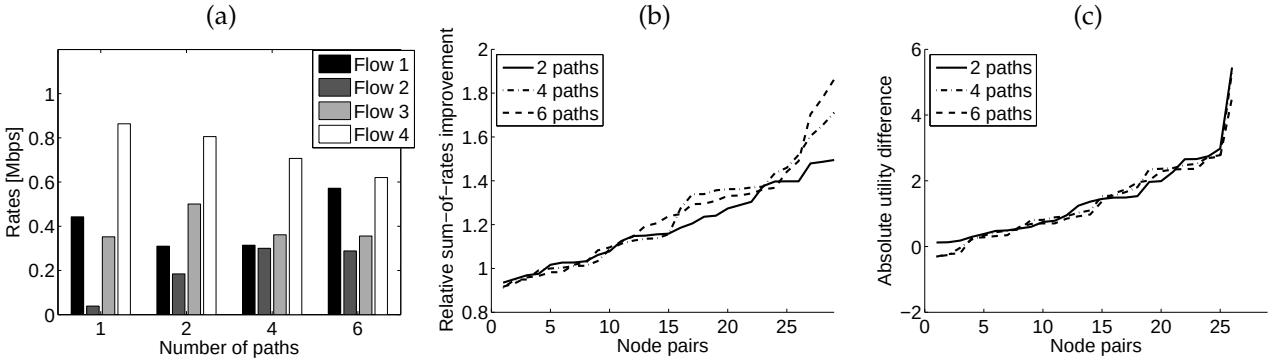


Figure 7: Performance of 4 concurrent flows in 100 instances of Roofnet topology with randomly selected source-destination pairs: (a) Example of end-to-end rate allocation for flows illustrated in Figure 5 (b) Cumulative relative improvement of the aggregate rates using 2, 4 and 6 paths over a single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

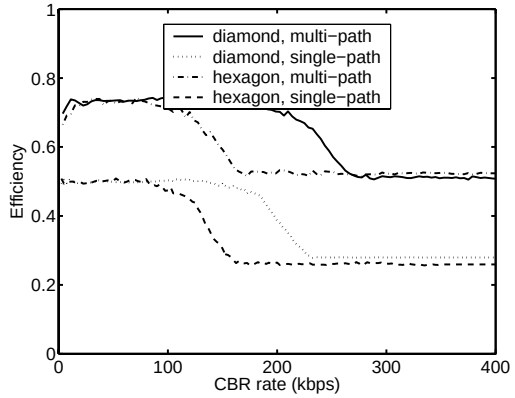


Figure 8: Efficiency for diamond and hexagon topologies with rate control. Source and destination are out of each other's transmission range. They communicate either using two parallel paths and network coding or a single path. The two paths interfere with each other. Loss probability is 0.5 for each link.

We configured the network parameters so that the source and destination cannot communicate directly. All other parameters have ns-2 default values.

We simulated the diamond and hexagon topologies, setting the successful transmission probabilities to 0.5 on all links. We focus on a single CBR flow, vary its offered load, and plot the efficiency in Figure 8. Efficiency is defined as the ratio of destination throughput to source transmission rate.

Note that the efficiency is around 0.5 for single path for low CBR rates and drops to 0.26 as the channel gets utilized at its maximum capacity. In the case of multipath and network coding, the efficiency is 0.75 for low CBR rates and goes to 0.5 for high CBR rates. Thus, using multipath and network coding for high CBR rates almost doubles the efficiency. As expected, the channel gets saturated sooner for the hexagon topology than it does for the diamond topology, since inefficient 802.11 scheduling causes more MAC layer collisions in this case.

## 4.3 Preliminary testbed evaluation

We have built a prototype implementing a basic subset of the $MC^2$ system, including multipath forwarding, coding on multiple paths and error control. Our prototype is built on top of the VRR system [4]. We use VRR's boilerplate functionality to establish a wireless mesh network, and to discover multiple paths between pairs of source and destination nodes. The use of VRR's routing algorithm was a matter of convenience; other routing protocols could be used to discover paths. We extend VRR by changing the data forwarding functionality. Our prototype is built as a Windows kernel driver.

We focus on a single flow in 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. It is signaled to the nexthops in spare bits of the routing header.

The 802.11 broadcast mode is currently used for all transmissions, except for end-to-end requests and retransmissions, to avoid mixed effects from the reliability provided by the 802.11 unicast and our own error recovery schemes. The 'pseudo broadcast' technique used in COPE [12] could yield more benefit, especially in assigning credits and avoiding collisions at the destination. We leave this for future work, and for now study scenarios where the relays are in the same contention domain.

We evaluate the performance on a testbed resembling the diamond topology in Figure 1(b). The nodes are laptops and desktop PCs running Windows, each equipped with a Netgear 802.11a/g card with the Atheros chipset. We used the default broadcast rate, 6Mbps, in 802.11a. The experiements were run for two topology

configurations, with high and low loss rates between the source and the relay nodes respectively. Different loss rates are produced by moving the source away from the relays. The relays were about 50cm apart in both configurations, and placed very close to the destination.

We initiate 800KB UDP transfers for two parallel paths and a single path, and compute the throughput gain, averaged over 10 runs, of $MC^2$ over single-path (Table 1). The results are consistent with the analysis earlier. $MC^2$ benefits from path diversity, especially when given lossy pre-coding hops.

| Scenario | Loss rate | Throughput gain |
|---|---|---|
| Low-loss | 20% | 17% |
| High-loss | 80% | 25% |

Table 1: Results from testbed experiments

To evaluate the overhead of coding and decoding operations, we also performed some microbenchmarks on a machine with 847MHz CPU speed and 256MB RAM. If 32 packets are used in generating a linear combination, it takes around 5.1ms on average. For linear combinations of up to 5 packets components, it takes on average 0.5ms to 1ms to generate, and up to 2ms for generating linear combinations from up to about 12 packets. The check for linear dependency takes negligible time, and decoding takes close to negligible time when averaged over a generation of packets. For memory overhead, currently we keep each full generation for a long time. However, our analytical results show that it is only necessary to keep around 5 packets per generation to ensure diversity in retransmissions. In general, there is much scope in optimising the implementation, which we leave for future work.

## 5. RELATED WORK

Our work draws experience from prior work on multipath routing in wired and wireless networks, rate control and network coding. In particular, our scheme is the first to tie together multipath routing, rate control and error control into a unified framework through network coding.

For wired networks, the most relevant are recent proposals considering cross-layer optimizations of multipath routing and rate control [10, 13, 15, 26]. However, there is little literature on how to select or switch between multiple routes.

For wireless, relevant cross-layer design was studied in [18] through theoretical analysis, and Popa et al considered congestion alleviation in wireless sensor networks using multipath routing [22].

Multipath routing in wireless networks is also based on exploiting spatial diversity at the physical layer [16]. Practical implementations includes ExOR [3] for wireless mesh networks, MRD [20] for one-hop WLAN, and SPaC [8] for sensor networks. Both MRD and SPaC improve loss resiliency by recovering partially corrupt copies of the same packet received from different paths. Our work is close to ExOR, but we permit nodes to forward packets concurrently on multiple paths where possible. Moreover, our scheme performs load-balancing and ensures fairness among different flows.

Network coding was originally proposed to increase the network capacity [1]. Subsequent work has used it to ensure reliability [19] and eliminate redundancy [9, 27]. CodeCast [21] uses network coding to control loss, bound delay, and exploits path diversity for wireless multicast. However, most work is theoretical and/or relies on simulations, focusing mainly on multicast and wireline networks.

COPE is a practical encoding scheme for wireless mesh environments [12]. COPE combines packets from *different* simultaneous unicast flows passing through a node. In contrast, $MC^2$ uses random linear codes [17] to encode packets from the *same* flow. While COPE uses coding to increase the network capacity, we use it to ease the signaling, error recovery and rate control of packets. More recently Chachulski et al. proposed MORE, which also combines packets of the same flow using network coding [5]. In many respects, MORE is similar to $MC^2$ . However, MORE does not explicitly consider multiple flows, fairness, or scheduling; in fact the performance benefit of MORE drops as the number of flows increases.

## 6. CONCLUSIONS

In this paper we have proposed $MC^2$, 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 signaling of packet transmissions, by avoiding strong coordination of which packets to forward between nodes. The credit-based algorithm ensures that good paths carry most of the traffic, and ensures fairness among multiple flows. The scheme is motivated by extensive prior analytical work.

We have outlined the main challenges in designing $MC^2$, and used analysis and simulation to demonstrate the performance improvements of our approach, which can double the throughput in certain cases. We have also built a prototype and tested on a small network, that further confirms the feasibility and advantages of $MC^2$.

For future work, we need more detailed analysis, a fuller implementation and more extensive evaluations in a real environment. Various design and implementation details are open for optimizations, such as reducing the buffer space requirements for coding, decoding and retransmissions, and optimizing coding op-

erations However, our results so far are very encouraging, and more than demonstrate the potential of $MC^2$.

## 7. REFERENCES


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

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

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

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

[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM SigComm*, 2007.

[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*, 2003.

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

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

[10] 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.

[11] T. Ho, M. Médard, M. Effros, and D. Karger. On randomized network coding. In *41st Allerton Annual Conference on Communication, Control and Computing*, Oct 2003.

[12] 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.

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

[14] 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.

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

[16] J. N. Laneman and G. Wornell. Exploiting distributed spatial diversity in wireless networks. In *Allerton Conference*, 2000.

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

[18] 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.

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

[20] 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.

[21] J.-S. Park, D. Lun, Y. Yi, M. Gerla, and M. Medard. Codecast: A network coding based ad hoc multicast protocol. *IEEE Wireless Communications*, (5), 2006.

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

[23] B. Radunovic, C. Gkantsidis, P. Key, P. Rodriguez, and W. Hu. An optimization framework for practical multipath routing in wireless mesh networks. In *MSR-TR-2007-81*, 2007.

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

[25] 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.

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

[27] J. Widmer and J.-Y. L. Boudec. Network Coding for Efficient Communication in Extreme Networks. In *WDTN Workshop*, 2005.