Dimitrios Koutsonikolas (dkoutson@purdue.edu)
Advisor: Prof. Y. Charlie Hu
Work done in collaboration with Prof. Chih-Chun Wang
School of ECE, Purdue University

CCACK: Efficient Network Coding Based Opportunistic Routing Through Cumulative Coded ACKnowledgments

1. Problem and Motivation

Wireless mesh networks (WMNs) are characterized by static mesh routers connected by wireless links to each other and to a few gateway nodes. The WMN routers effectively form a multihop wireless backbone. Recently, the deployment and use of WMNs have increased significantly and several cities have planned deployed WMNs (e.g., [1, 2, 3]). A main challenge in WMNs is to deal with the poor link quality due to urban structures and interference, both internal (among flows in the WMN) and external (from other 802.11 networks). For example, 50% of the operational links in Roofnet [1] have loss rates higher than 30% [4]. Hence, routing protocol design is critical to the performance and reliability of WMNs.

Traditional routing protocols for multihop wireless networks treat the wireless links as point-to-point links. First a fixed path is selected from the source to the destination; then each hop along the chosen path simply sends data packets to the next hop via 802.11 unicast. For example, in Figure 1, a traditional routing protocol would select one of the two available paths, say through node A, to send packets from node S to node D. Due to the broadcast nature of the wireless medium, it is possible that a packet transmitted by the source is not received by node A but by node B instead. In such a case, B discards the packet and the source has to retransmit it until A receives it.

Opportunistic Routing (OR), as first demonstrated in the ExOR protocol [4], has recently emerged as a mechanism for improving unicast throughput in WMNs with lossy links. Instead of first determining the next hop and then sending the packet to it, a node with OR broadcasts the packet so that all neighbors have the chance to hear it and assist in forwarding. In Figure 1, both A and B can forward the packets they receive, thus making faster progress towards the destination; the source only has to retransmit a packet if none of the two nodes receives it.

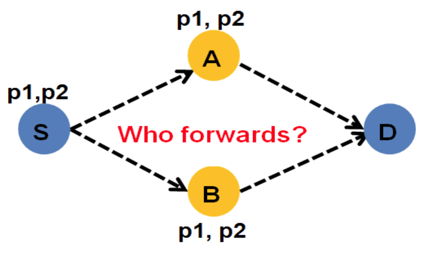


Figure 1. Opportunistic Routing: Coordination is required between A and B.

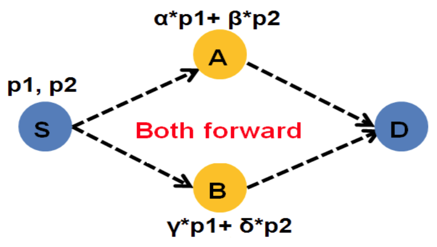


Figure 2. Intra-flow Network Coding: Coordination between A and B is no longer required.

OR provides significant throughput gains compared to traditional routing, however, it introduces a difficult challenge. It may happen that when a node broadcasts a packet, more than one neighbors receive it. For example, in Figure 1, both A and B have received the same two packets p1, p2, from S. Without any coordination, all nodes that hear a packet will attempt to forward it. These duplicate transmissions waste bandwidth, which is a scarce resource in wireless networks. To address this challenge, coordination is required among nodes, so that they can determine which one should forward each packet.

Coordination can also result in significant bandwidth consumption, since it requires nodes to exchange messages. Recently, [5] showed through the design of the MORE protocol, that the use of random **intra-flow network coding (NC)** can address this challenge in a very simple and efficient manner. With NC, a router randomly mixes the packets it has received before forwarding them; instead of sending out individual packets, it broadcasts linear combinations of those packets, as shown in Figure 2. Random mixing at each forwarder ensures that with high probability different nodes that may have heard the same packet can still transmit linearly independent coded packets. Hence, coordination among forwarders is no longer required.

NC has significantly simplified the design of OR protocols and led to substantial throughput gains compared to non-coding based protocols. However, the use of NC introduces a new challenge: **How many coded packets should each forwarder transmit?** We illustrate this challenge with the example in Figure 3. The source S has three downstream forwarding nodes A, B, and C within its transmission range. Assume for simplicity that S has three innovative packets X1, X2, and X3 to send. Instead of transmitting the native packets, S transmits three coded packets X1 + X2 + X3, 3X1 + X2 + 2X3, and X1 + 2X2 + 3X3 in sequence, which are denoted by the corresponding coding vectors (1, 1, 1), (3, 1, 2), and (1, 2, 3). Assume that (1, 1, 1) coded packet is received by C, and the (3, 1, 2) and (1, 2, 3) packets are received by A and by {A,B}, respectively. The downstream nodes A, B, and C have received a sufficient amount of innovative packets. Collectively, the three nodes can now act as the new source and the original source S should stop transmission. However, it is a nontrivial task for S to know whether its downstream nodes have accumulated a sufficient amount of innovative packets.

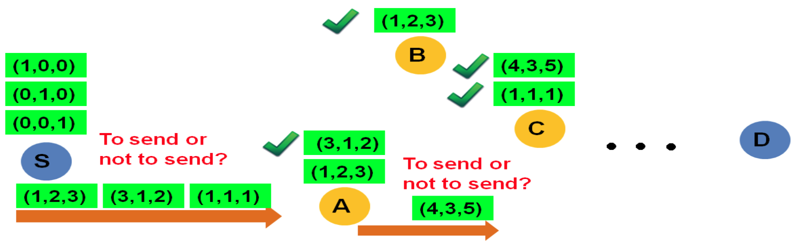


Figure 3. The importance of knowing how many coded packets to transmit.

The intermediate node A faces an even greater challenge. One of its two received packets, the (1, 2, 3) packet, is received by B as well. Without any coordination with B, A sends a useful network-coded packet (4, 3, 5), which is received by C. The same question arises, i.e., whether A should continue sending coded packets or it should stop transmission. From a holistic perspective, collectively B and C have received three independent packets and can now act as the new source. Hence A should stop transmission. Nonetheless, A has limited knowledge about the reception status of the three packets transmitted by S (e.g., A may not know that C has received (1, 1, 1) from S), which makes the decision whether to stop transmission even harder for A than that for the source S.

2. Background and Related work

MORE [5] was the first practical NC-based OR protocol showing that random linear network coding can be used to remove the need for coordination in among forwarders in OR protocols and it can be implemented on commodity hardware. Subsequently, several other NC-based OR protocols [6, 7, 8] were proposed addressing several inefficiencies of MORE. The basic operation of all these protocols is similar. The source works with a batch of *N* data packets and broadcasts linear combinations of those packets in the form:

$$p'_j = \sum_i c_{ji} p_i$$

where p_i's are the native packets and c_{ji}'s are random coefficients chosen from a Galois Field. We call (c_{j1}, c_{j2}, ..., c_{jN}) the *coding vector* of packet p'_j. The coding vector is included in the packet header to inform other nodes of how the coded packet was created from the native packets. Whenever a node hears a coded packet, the node first checks whether the packet is linearly independent (innovative) to the packets it has received in the past by comparing its coding vector to those of the existing packets stored in a buffer B_υ. If innovative, the newly received packet is stored in B_υ. Forwarding nodes also broadcast coded packets by randomly mixing the innovative packets in their B_υ buffer. The destination stores all the innovative packets it receives until it collects *N* such packets. It then decodes the original batch of packets and sends an end-to-end ACK back to the source to inform it to move to the next batch.

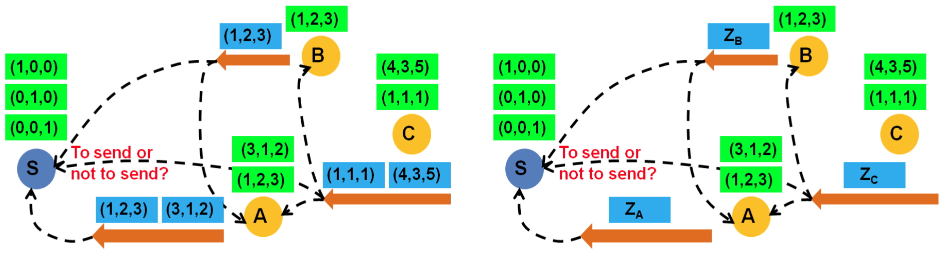
As mentioned above, the main challenge in this class of protocols is *how many new coded packets a forwarder should transmit every time it receives a packet from upstream*. To avoid the overhead of feedback exchange, all these protocols compute offline the expected number of transmissions for each forwarder using heuristics based on periodic measurements of the average link loss rates. Although attractive due to their minimal coordination overhead, these approaches may suffer significant performance degradation in dynamic wireless environments with continuously changing levels of channel gains, interference, and background traffic. In particular, recent WMN studies [9, 10] have shown that, although link loss rates remain relatively stable for long intervals in a quiet network, they are very sensitive to background traffic, resulting in over- or underestimation of the amount of packet every node should transmit. Overestimation causes redundant transmissions, which waste bandwidth. Underestimation may have an even worse impact, since nodes may not transmit enough packets to allow the destination to decode a batch. **This motivates the need for a new approach, one that is oblivious to loss rates**.

3. Approach and Uniqueness

In this work, we propose CCACK, a new efficient NC-based OR protocol. Forwarding nodes in CCACK decide how many packets to transmit in an **online** fashion, and this decision is completely **oblivious to link loss rates**. This is achieved through a novel **C**umulative **C**oded **ACK**nowledgment scheme that allows nodes to acknowledge network coded traffic to their upstream nodes in a simple and efficient way, **with practically zero overhead**. **CCACK is the first NC-based protocol that brings back the traditional approach of feedback exchange, which had been aborted with the advent of network coding; but it exploits network coding to effectively compress this feedback and hide its overhead, while still exploiting its benefits.** Coded feedback in CCACK is not required strictly on a per-packet basis; this makes the protocol resilient to individual packet loss and significantly reduces its complexity.

The need for Cumulative Coded Acknowledgments

Take the scenario in Figure 3 as a continuing example. One naive approach to ensure that S (resp. A) knows when to stop transmission is through the use of reception reports, for which each node broadcasts all the basis vectors of the received linear space to its upstream nodes, as illustrated in Figure 4(a). An obvious drawback of this approach is the size of the feedback messages. For practical network coding with symbol size GF(2^8) and batch size 32 packets, each coding vector contains 32 bytes. To convey a space of dimension $\kappa >> 1$ thus requires κ 32-byte vectors, which is too large to piggyback to normal forward traffic. The unreliability of the wireless channel further exacerbates the problem as the κ feedback messages need to be retransmitted several times until they are overheard by all the upstream nodes.



(a) Uncoded feedback: Each node sends **all** its basis vectors to its upstream nodes. (b) Coded feedback: Each nodes sends a **single** coded vector to its upstream nodes.

Figure 4. Different types of feedback for network-coded traffic.

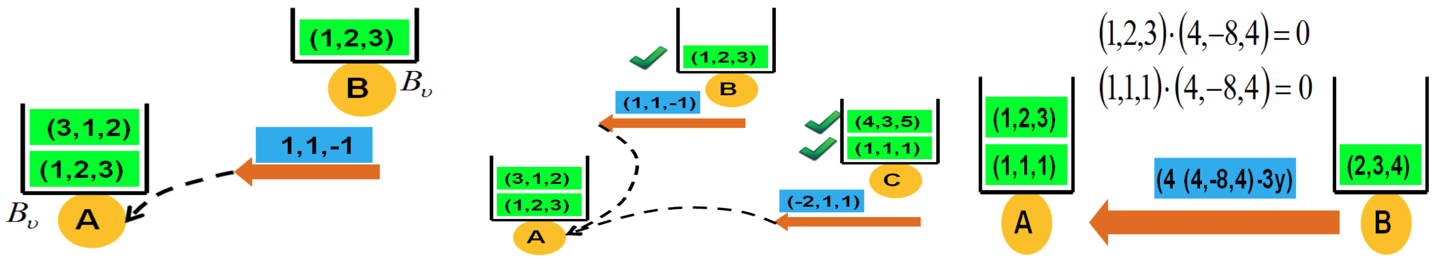
In contrast, in CCACK each node uses a single coded feedback vector to represent the entire space, which may consist of κ >> 1 basis vectors. In the broadest sense, the three coded acknowledgment vectors z_A to z_C in Figure 4(b) serve as a hash for their corresponding spaces. Each single vector can be easily piggybacked to the forward data traffic. This compressed/coded acknowledgment is critical to the efficiency, since in CCACK overhearing any of the data packets of A with piggybacked coded ACK will convey to the upstream nodes the entire space (or most of the space) of A. This thus drastically reduces the need of retransmitting feedback information over the unreliable wireless channel.

Null-Space-Based Feedback

One candidate solution, attractive due to its simplicity, is **null-space-based (NSB)** coded feedback i.e., each node sends to each upstream nodes one vector randomly chosen among all vectors in the null space of the innovative vectors the node has received in the past. Take for example Figure 5(a). Every time node B sends a coded data packet to its own downstream nodes, it also appends to the packet header one vector z_B satisfying:

$$z_B \cdot \upsilon = 0, \forall \upsilon \in B_\upsilon$$

Namely, the inner product between z_B and υ is zero. There may be multiple choices of z_B that satisfy this condition (e.g., in Figure 5(a), z_B can be any vector of the form (-2x-3y, x, y)). z_B is then chosen uniformly randomly among all valid vectors. When the upstream node A overhears a packet from B, it simply needs to compute the inner product of each of its own innovative vectors with z_B. Suppose that z_B is chosen as (1, 1, -1). Since (1, 2, 3) • (1, 1, -1) = 0, A concludes that node B has received packet (1, 2, 3). On the other hand, since (3, 1, 2) • (1, 1, -1) = 2 ≠ 0, A concludes that packet (3, 1, 2) is an innovative packet for B, and hence it should send more coded packets to B.



(a) Basic operation. (b) The collective space problem. (c) The false positive error.

Figure 5. Null-space-based (NSB) coded feedback

Although attractive due to its simplicity, the above NSB feedback scheme suffers from two significant limitations when used in NC-based OR: the **collective space problem**, and the **non-negligible false positive probability**.

The collective space problem. Nodes B and C in Figure 5(b) have already (jointly) received 3 linearly independent packets; they want to convey their space information to A so that A can stop packet transmission. Based on the NSB concept, B and C send z_B = (1, 1, -1) and z_C = (-2, 1, 1), respectively, which are orthogonal to their local innovative vectors. The idea is that, upon the reception of z_B and z_C, A will know that the knowledge spaces of B and C have collectively covered the local knowledge space of A and thus will stop transmission. Nonetheless, when A checks the inner product of the coded feedback and its own innovative packets, we have z_B • (3, 1, 2) = 2 ≠ 0 and z_C • (3, 1, 2) = -3 ≠ 0. Therefore *A thinks that the coding vector (3, 1, 2) is innovative to both its downstream nodes and thus continues transmission even when collectively B and C already have enough information*; it will not stop until one of its downstream nodes has a local knowledge space that covers the local knowledge space of A. This defeats the purpose of OR. This misjudgment is caused because (a) the NSB coded feedback does not convey the collective space of all downstream nodes but only the space relationship between the individual pairs (e.g., A vs. B and A vs. C) and (b) the basis vectors of the downstream nodes may be different from the basis vectors of the upstream node. In Figure 5(b), node B has received the vector (4, 3, 5), which was constructed and sent by A, but A has no knowledge of this vector anymore.

Non-negligible false positive probability. Take Figure 5(c) for example. Node A would like to send two packets to node B and a network coded packet has been received by B already. B sends an orthogonal vector z_B, which is randomly chosen to be any vector of the form z_B = (4x, 4y,−2x − 3y), i.e., any vector orthogonal to (2, 3, 4). Suppose that B happens to choose z_B = (4,−8, 4); this particular vector also happens to be orthogonal to both the innovative vectors of A. Hence, A will wrongfully imply that the knowledge space of B covers the local knowledge space of A and it will send no more packets. Although the probability of such a false positive event is small, its impact to the system performance can be significant. In a multi-hop transmission, any single hop that experiences this false positive event will cause an upstream node to stop transmission prematurely. The communication chain is thus broken and the destination may not be able to receive enough packets to perform decoding.

CCACK design

In addition to the buffer of innovative packets B_υ, nodes in CCACK maintain two additional buffers for each flow: B_u and B_w. B_u stores the coding vectors of all the packets a node receives from upstream nodes, regardless being innovative or not. B_w stores the coding vectors of all the packets the node transmits. Each vector in B_u and B_w can be marked as H (heard by a downstream node) or $\neg H$ (not heard). A vector is marked as $\neg H$ when initially is inserted in either of the two buffers. Upstream nodes mark vectors in B_u and B_w as H, using the inner product of these vectors and the ACK vectors they receive from downstream nodes. ACK vectors are appended to the headers of coded data packets; they are never sent as stand-alone packets.

Solving the collective-space problem. Nodes in CCACK construct the ACK vectors using **all** the forward coding vectors stored in B_u, and not only the innovative vectors stored in B_υ. Also, when a node overhears a packet from a downstream node, it uses the ACK coding vector of that packet to decide whether any of the coding vectors **in $B_u \cup B_w$** are heard by the downstream node. Remember that the simple NSB scheme we described above only considers the innovative packets in B_υ.

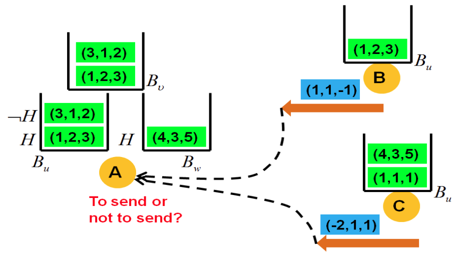


Figure 6. CCACK: solving the collective space problem.

Focusing on B_u and B_w vectors instead of B_υ, this new structure solves the collective-space problem of the NSB coded feedback. Continue our example from Figure 5(b) in Figure 6. By checking the inner products of the vectors in B_u and B_w with z_B and z_C, A knows that the (1, 2, 3) and (4, 3, 5) have been *heard* by downstream nodes. Since the dimension of (1, 2, 3) and (4, 3, 5) is the same as the dimension of B_υ vectors, A stops transmission.

Reducing the false-positive probability.

Each node maintains M different N × N diagonal hash matrices H_1 to H_M where N = 32 is the batch size and each entry of the diagonal of the matrix is randomly chosen from $GF(2^8)$. All nodes in the network are aware of the H_1 to H_M matrices of the other nodes. Nodes construct the ACK vectors using the following algorithm:

```
§ CONSTRUCT THE ACK VECTOR
Start from a $0 \times N$ matrix $\Delta$.
while The number of rows of $\Delta \leq N - 1 - M$ do
    Choose randomly one vector $u \in B_u$.
    for $j = 1$ to $M$ do
        if the $1 \times N$ row vector $uH_j$ is linearly independent to the row space of $\Delta$ then
            Add $uH_j$ to $\Delta$.
            Perform row-based Gaussian elimination to keep $\Delta$ in a row-echelon form.
        end if
    end for
end while
Choose randomly the coding coefficients $c_1$ to $c_N$ such that the following matrix equation is satisfied:
    $\Delta(c_1, \cdots, c_N)^T = (0, \cdots, 0)^T$.
Use the vector $(c_1, \cdots, c_N)$ as the ACK vector.
```

Figure 7. CCACK: The ACK vector construction algorithm.

With this new ACK construction, a vector *u* in the B_u (or B_w) buffer of an upstream node is marked *H* if and only if *u* passes all the following M different tests:

$$\forall j = 1, \ldots, M, \; uH_j z^T = 0$$

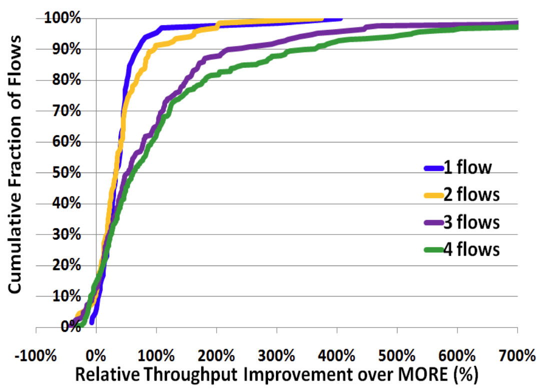
where H_1 to H_M are the hash matrices of the downstream node that sent the ACK vector. It is easy to show that for any vector u in $B_u \cup B_w$ that does not belong to the linear space spanned by the vectors selected by the downstream node, the probability to pass all M tests is $(1/2^8)^M$. In our implementation, we used M = 4, which gives a false positive probability of 2.33×10^{-10}.

4. Results and Contributions

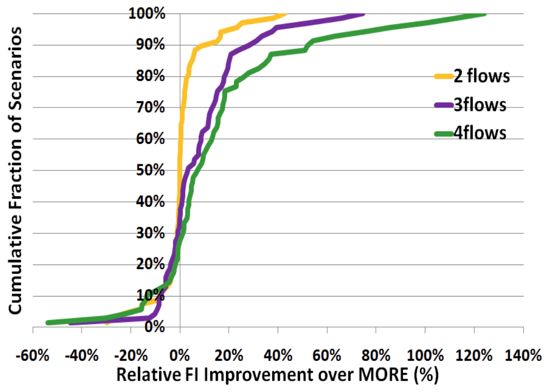
We evaluated the performance of CCACK and compared it against MORE, using extensive simulations. We used the *Glomosim* simulator [11], a widely used wireless network simulator with a detailed and accurate physical signal propagation model. We simulated a network of 50 static nodes placed randomly in a 1000m x 1000m area. The *Two Ray* propagation model was used and combined with the Rayleigh fading model to simulate a realistic environment, where the transmission and interference range are not constant but very significantly around their average values. We varied the number of flows in the network from 1 up to 4. For a given number of flows, we simulated 9 different randomly generated topologies, and for each topology we repeated the simulation 10 times, each time with a different randomly selected set of source-destination pairs. Hence, we simulated a total of 90 scenarios. We compared the two protocols in terms of throughput and fairness. For fairness, we used the popular Jain's Fairness Index (FI) [12], defined as

$$FI = \left(\sum x_i\right)^2 \Big/ \left(n \times \sum x_i^2\right)$$

where x_i is the throughput of flow i and n is the total number of flows. The value of Jain's FI is between 0 and 1, with values closer to 1 indicating better fairness.



(a) Throughput comparison.



(b) Fairness comparison.

Figure 8. Performance comparison between CCACK and MORE using simulations.

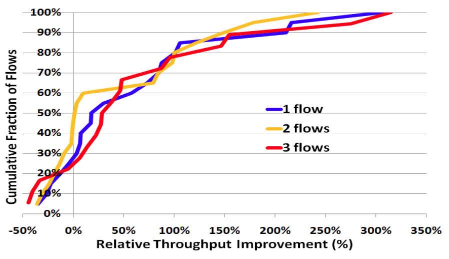
Figure 8(a) plots the Cumulative Distribution Function (CDF) of the relative per-flow throughput improvement of CCACK over MORE for 70 different scenarios with 1, 2, 3, and 4 flows in the network. CCACK significantly improves throughput over MORE; the median improvement is 33% with 1 and 2 flows, 55% with 3 flows, and 62% with 4 flows. Figure 8(b) plots the CDF of per-scenario relative improvement to Jain's FI with CCACK over MORE, with 2, 3, and 4 flows, for the same scenarios. CCACK improves fairness in more scenarios compared to MORE as the number of flows increases: 40% of the 2-flow, 65% of the 3-flow, and 72% of the 4-flow scenarios. Overall, by allowing nodes to make the decision of how many packets to transmit online, rather than relying on offline heuristics, CCACK improves both throughput and fairness compared to MORE.

The benefits of CCACK are more pronounced as the number of flows in the network (and hence the levels of congestion and interference) increase. In those scenarios, MORE's *offline heuristic may completely fail to accurately estimate the necessary number of transmissions* for each forwarder, resulting in starvation for some flows. In Figures 8(a), 8(b), we observe **throughput gains up to 4x, 3.7x, 8.1x, and 20.4x**, in the 1-flow, 2-flow, 3-flow, and 4-flow case, respectively, and **FI improvements up to 74% and 124%** with 3 flows, and 4 flows, respectively.

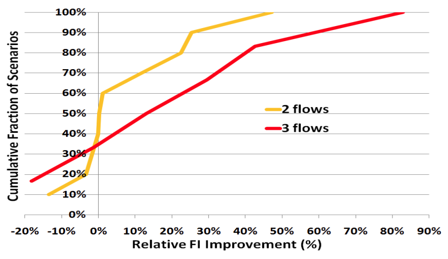
Although CCACK incurs a higher coding overhead than MORE, since routers have to perform additional operations when transmitting/receiving a packet, we note that all these additional operations are performed on N-byte vectors instead of the whole K-byte payload. Therefore, in practical settings (e.g., with N = 32 and K = 1500), **the coding overhead of CCACK is comparable to that of MORE**, which makes the protocol easily deployable.

To demonstrate this, we implemented application layer prototypes of CCACK and MORE on Linux and evaluated their performance on a 22-node 802.11 WMN testbed we have deployed in two academic buildings at Purdue University [13]. A schematic of the testbed is shown in Figure 9(a), and the evaluation results are shown in Figures 9(b), 9(c). Although the small size of our testbed along with the limitations of our implementation limit the potential gains, we observe that the results are qualitatively very similar to the simulation results; CCACK improves both throughput and fairness, by up to 3.2x and 83%, respectively, with average improvements of 11-36% and 5.7-8.3%, respectively, for different numbers of flows, validating the benefits of our approach.

2nd floor
3rd floor
(a) Schematic of MAP testbed.



(b) Throughput comparison.



(c) Fairness comparison.

Figure 9: Performance comparison between CCACK and MORE on MAP testbed.

Overall, our work makes the following contributions:

- We identify a fundamental challenge in the newly emerged class of NC-based OR protocols: **How many coded packets should each forwarder transmit?** We discuss the inefficiencies of existing loss-based approaches to addressing this challenge and show, through our simulation and testbed evaluations, the severe impact such approaches can have on the performance of NC-based OR protocols.

- We propose CCACK, a new efficient NC-based OR protocol. Unlike existing protocols, nodes in CCACK decide how many packets to transmit in an **online** fashion, and this decision is completely **oblivious to link loss rates**. Central to the design of CCACK is a novel **Cumulative Coded ACKnowledgment** scheme that allows nodes to acknowledge network coded traffic to their upstream nodes in a simple and efficient way, with practically zero overhead.
  Our extensive simulation study using a popular wireless network simulator with a realistic physical model shows that CCACK offers significant throughput and fairness improvements over the state-of-the-art MORE protocol. Our experiments on a 22-node WMN testbed confirm the simulation findings.

- CCACK brings a **shift to the design paradigm** of NC-based OR protocols. Existing NC-based OR protocols have identified feedback overhead as a main cause for performance degradation of practical wireless routing protocols and used NC to eliminate the need for feedback exchange, resorting to offline loss-based heuristics. On the contrary, CCACK encodes feedback to exploit its benefits and avoid the drawbacks of offline heuristics, and at the same time to hide its overhead.

This work has been published in IEEE INFOCOM 2010 and an extended version has been submitted to IEEE/ACM Transaction on Networking (ToN).

5. Acknowledgment

This work was supported in part by NSF grants CCF-0845968 and CNS-0905331.

6. References

[1] “MIT Roofnet,” http://www.pdos.lcs.mit.edu/roofnet.
[2] Technology For All (TFA). http://tfa.rice.edu.
[3] Google WiFi. http://wifi.google.com/.
[4] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In Proc. of ACM SIGCOMM, 2005.
[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In Proc. of ACM SIGCOMM, 2007.
[6] X. Zhang and B. Li, “Optimized multipath network coding in lossy wireless networks,” in Proc. of IEEE ICDCS, 2008.
[7] X. Zhang and B. Li, “Dice: a game theoretic framework for wireless multipath network coding,” in Proc. of ACM MobiHoc, 2008.
[8] Y. Lin, B. Li, and B. Liang, “CodeOR: Opportunistic routing in wireless mesh networks with segmented network coding,” in Proc. of IEEE ICNP, 2008.
[9] S. M. Das, H. Pucha, K. Papagiannaki, and Y. C. Hu, “Studying Wireless Routing Link Dynamics,” in Proc. of ACM SIGCOMM/USENIX IMC, 2007.
[10] Yi Li, L. Qiu, Y. Zhang, R. Mahajan, Z. Zhong, G. Deshpande, and E. Rozner“Effects of interference on throughput of wireless mesh networks: Pathologies and a preliminary solution,” in Proc. of ACM HotNets-VI, 2007.
[11] X. Zeng, R. Bagrodia, and M. Gerla, “Glomosim: A library for parallel simulation of large-scale wireless networks,” in Proc. of PADS Workshop, May 1998.
[12] R. K. Jain, D. W. Chiu, and W. R. Hawe, “A quantitative measure of fairness and discrimination for resource allocation in shared computer systems,” DEC-TR-301, Digital Equipment Corporation, Tech. Rep., September 1984.
[13] Mesh@Purdue, “http://www.engineering.purdue.edu/mesh.”.