Designing Coded Feedback for Efficient Network Coding Based Opportunistic Routing

Dimitrios Koutsonikolas, Chih-Chun Wang, and Y. Charlie Hu
Purdue University, West Lafayette, IN, USA
{dkoutson, chihw, ychu}@purdue.edu

1. MOTIVATION

Opportunistic Routing (OR) [1] is a new routing paradigm for improving throughput in lossy wireless mesh networks (WMNs). In OR, a set of nodes in a *belt* around a routing path (i.e., the forwarding nodes or FNs) collaboratively forward packets from the source to the destination. A major challenge in OR is how the FNs coordinate to avoid spurious retransmissions which waste bandwidth.

Recently, [2] showed that the use of *random intra-flow network coding* (NC) can address this challenge in a simple and efficient manner. With NC, the source sends random linear combinations of packets, and each router also randomly mixes packets it already has received before forwarding them. Random mixing at each router ensures that with high probability different nodes that may have heard the same packet can still transmit linearly independent coded packets. However, the use of NC introduces a new challenge: *How many coded packets should each forwarder transmit?*

We illustrate this challenge, with the example of Figure 1. The source S has three downstream FNs A, B, C and three innovative packets X_1, X_2, and X_3 to send. Instead of transmitting the native packets, S transmits three coded packets $X_1 + X_2 + X_3$, $3X_1 + X_2 + 2X_3$, and $X_1 + 2X_2 + 3X_3$ in sequence, denoted by the corresponding coding vectors $(1, 1, 1)$, $(3, 1, 2)$, and $(1, 2, 3)$. Assume that coded packet $(1, 1, 1)$ is received by C, and packets $(3, 1, 2)$ and $(1, 2, 3)$ are received by A and by $\{A, B\}$, respectively. The downstream FNs A, B, C have received a sufficient amount of innovative packets; collectively, they can now act as the new source, and the original source S should stop transmission. However, it is a non-trivial task for S to know whether its downstream FNs have accumulated a sufficient amount of innovative packets.

The same challenge exists for the intermediate FN A. After transmitting a useful coded packet $(4, 3, 5)$, which is received by FN C, A has to decide whether it should continue or stop sending coded packets. Furthermore, 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 of whether to stop transmission even harder for A than for the source S.

To address the challenge in a simple manner, and to minimize the control overhead, recently proposed NC-based OR protocols (e.g., [2, 5]) use *offline* heuristics *based on link loss rates* and the ETX metric. The drawback of these approaches is that performance heavily depends on the accuracy and freshness of the loss rate measurements. Loss rate estimates are obtained through periodic probing and are propagated from all nodes to the source. Apparently, the higher the probing frequency, the higher the accuracy, but also the higher the overhead.

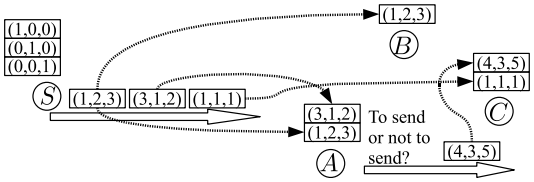


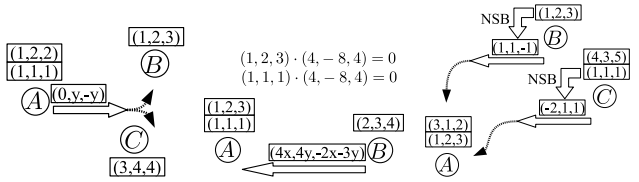
Figure 1: Illustration of the importance of knowing how many coded packets to transmit.

To reduce this overhead, the authors in [2] collect the loss rates only in the beginning of each experiment. In practice, this suggests that loss rate measurements should be performed rather infrequently. Unfortunately, recent WMN studies [3, 4] have shown that, although link metrics 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 FN should transmit. Overestimation causes redundant transmissions, which waste wireless bandwidth. Underestimation may have 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, *oblivious* to loss rates.

In this work, we propose CCACK, a new efficient NC-based OR protocol. FNs 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*. 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.

2. A SIMPLE CODED FEEDBACK SCHEME

Coded feedback has been used in the past in a different context; in [6], a null-space-based (NSB) coded feedback scheme is used to enhance the reliability of an NC-based



(a) NSB Feedback for unreliable multicast.

(b) The false positive error of the NSB feedback.

(c) NSB Feedback for reliable unicast OR.

Figure 2: Null-Space-Based (NSB) feedback for unreliable multicast, the corresponding false positive event, and proposed NSB feedback for reliable unicast OR.

multicast protocol for multimedia applications in mobile ad hoc networks. Take Figure 2(a) for example. A batch of 3 packets are coded together and nodes A, B, and C act as FNs, forwarding network coding traffic towards the members of a multicast group. Let B_v denote the buffer containing the innovative coding vectors received by A (which contains two vectors $(1, 2, 2)$ and $(1, 1, 1)$ in Figure 2(a)).

Since A has received fewer than 3 innovative packets, it informs its neighbor nodes that it needs more packets by appending to each coded packet a vector z_A satisfying

$$z_A \cdot v = 0, \quad \forall v \in B_v. \tag{1}$$

When node B (resp. C) overhears a packet from A, it simply needs to compute the inner product of its own innovative vectors with z_A. Suppose that z_A is chosen as $(0, 1, -1)$. Since $(0, 1, -1) \cdot (1, 2, 3) = -1 \neq 0$, node B must contain an innovative packet for A. B can broadcast its innovative packet and once A receives it, A will be able to decode the entire batch.

Problem 1: The collective space problem. Take Figure 1 for example. Based on the NSB concept, B and C send $z_B = (1, 1, -1)$ and $z_C = (-2, 1, 1)$, respectively, that are orthogonal to their local innovative vectors. When A checks the inner product of the coded feedback and its own innovative packets, we have:

$$z_B \cdot (3, 1, 2) = 2 \neq 0 \text{ and } z_C \cdot (3, 1, 2) = -3 \neq 0.$$

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 have had enough information already*.

This misjudgment is caused by that the NSB coded feedback does not convey the collective space of all downstream nodes but only the space relationship between the individual pairs (i.e., A vs. B and A vs. C). Indeed, each node in the flooding-based multicast protocol in [6] tries to receive (and forward) as many coded packets of a batch as possible; *there is no notion of a set of nodes collectively receiving and forwarding information*, unlike in unicast OR. Therefore, if we apply the NSB coded feedback of [6] to unicast OR, A will not stop transmission 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.

Problem 2: Non-negligible false-positive probability. Take Figure 2(b) 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 satisfying (1), which is randomly chosen to be any vector of the form $z_B = (4x, 4y, -2x - 3y)$. Suppose that B happens to choose $z_B = (4, -8, 4)$. *Since z_B is orthogonal to all the innovative vectors of A, A will wrongfully imply that the knowledge space of B covers the local knowledge space of A. A thus attempts no further transmission.* Although the probability of such a false positive event is small, its impact to the system performance is significant. The reason is that 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 independent packets to decode the current batch.

3. CCACK DESIGN

We now describe the design of our proposed CCACK protocol. Central to the design of CCACK is a novel cumulative coded feedback scheme for NC-based OR that addresses the two problems of the simple scheme in [6].

CCACK Overview. Nodes in CCACK maintain per flow state, which includes three buffers: a packet buffer B_v and two coding vector buffers B_u and B_w. The source and the FNs broadcast randomly mixed packets and store the coding vectors (coding coefficients) of these packets in B_w. Whenever a node overhears a packet, the node first checks whether the packet is innovative by comparing the coding vector to those of the existing packets in B_v. If innovative, the newly received packet is stored in B_v. Regardless being innovative or not, the node also checks whether the newly received packet is from an upstream node. If yes, then it stores the forward coding vector in B_u.

Each coding vector in B_u and B_w can be marked as H (heard by a downstream node) or $\neg\mathsf{H}$ (not heard). A vector is marked as $\neg\mathsf{H}$ when initially is inserted in either of the two buffers, since the node has no information at that time whether any downstream node has heard the packet or not.

Similar to [6], nodes in CCACK embed an additional *ACK vector* in the header of each coded data packet to report a subset of the packets (or coding vectors) they have received (heard) in the past from their upstream nodes. 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, as described in the following.

Solving the collective-space problem. In contrast to [6], nodes in CCACK construct the ACK coding vectors using *all the forward coding vectors stored in B_u*, and not only the innovative vectors stored in B_v. 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 $B_u \cup B_w$*, instead of in B_v, are heard by the downstream node.

Nodes keep checking the rank of the B_u and B_w vectors marked as H. When this rank becomes equal to the rank of innovative packets in B_v for a node A, A will stop transmit-

ting either temporarily, until it receives another innovative packet, or permanently if the rank of the B_v vectors is already equal to N. In both cases, the downstream nodes have received a sufficient number of packets that cover the innovative packets of A from the knowledge space perspective.

Focusing on B_u and B_w vectors instead of B_v, this new structure solves the collective-space problem of the NSB coded feedback. Continue our example in Fig. 2(c). For node A, B_u contains the received coding vectors $(1, 2, 3)$ and $(3, 1, 2)$ while B_w contains the transmitted vector $(4, 3, 5)$. Suppose we reuse the NSB coded feedback for nodes B and C. Then by checking inner products with z_B and z_C, A knows that the $(1, 2, 3) \in B_u$ and $(4, 3, 5) \in B_w$ have been received. Since the rank of $(1, 2, 3)$ and $(4, 3, 5)$ is the same as the rank of B_v vectors, A stops transmission.

Solving the false positive problem. We now describe our new ACK design that reduces the false-positive probability from $\frac{1}{2^8}$ (in [6]) to $\left(\frac{1}{2^8}\right)^M$ for any integer $M \geq 1$.

Each node maintains M different $N \times N$ 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. This is achieved by using the ID of a node as a seed to generate the H_1 to H_M matrices. Nodes construct the ACK vectors using the following algorithm:

§ CONSTRUCT THE ACK VECTOR

1: Start from a $0 \times N$ matrix Δ.
2: **while** The number of rows of $\Delta \leq N - 1 - M$ **do**
3: Choose randomly one vector $u \in B_u$.
4: **for** $j = 1$ to M **do**
5: **if** the $1 \times N$ row vector uH_j is linearly independent to the row space of Δ **then**
6: Add uH_j to Δ.
7: Perform row-based Gaussian elimination to keep Δ in a row-echelon form.
8: **end if**
9: **end for**
10: **end while**
11: Choose randomly the coding coefficients c_1 to c_N such that the following matrix equation is satisfied:

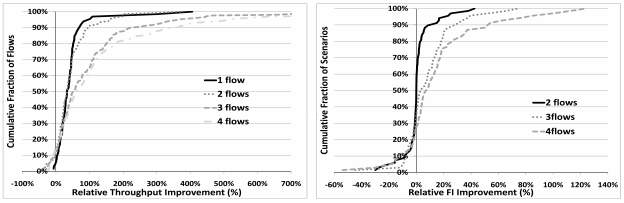
$$\Delta(c_1, \cdots, c_N)^{\mathrm{T}} = (0, \cdots, 0)^{\mathrm{T}}.$$

12: Use the vector $(c_1, \cdots, c_N)$ as the ACK vector.

When a node overhears a packet with an ACK vector z from a downstream node, it uses the inner product to check all its vectors in B_u and B_w and decides whether any of them has been heard by the downstream node. More explicitly, a vector $u \in B_u$ (or B_w) is marked H if and only if u passes *all the following M different tests* (one for each H_j):

$$\forall j = 1, \cdots, M, \quad uH_j z^{\mathrm{T}} = 0, \tag{2}$$

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



(a) Throughput comparison. (b) Fairness comparison.

Figure 3: Performance comparison between CCACK and MORE with different number of flows.

4. EVALUATION

We conducted a preliminary evaluation of CCACK's performance and compared it against MORE using the Glomosim simulator. We simulated a network of 50 static nodes placed randomly in a $1000m \times 1000m$ area. The *TwoRay* propagation model was used and combined with the Rayleigh fading model to make the simulations realistic.

Figure 3(a) plots the CDF of the relative per-flow throughput improvement of CCACK over MORE and Figure 3(b) plots the CDF of per-scenario relative improvement to Jain's Fairness Index (FI) with CCACK over MORE, with 2, 3, and 4 flows, for 70 different scenarios. 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. In addition, CCACK improves fairness in more scenarios as the number of flows increases: 40% of the 2-flow, 65% of the 3-flow, and 72% of the 4-flow scenarios. By quickly and accurately stopping transmissions at nodes whose downstream nodes have collectively received a sufficient number of packets, CCACK particularly benefits challenged scenarios, where flows completely starve with MORE, offering 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 expected to be comparable to that of MORE, which makes the protocol easily deployable.

5. REFERENCES

[1] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In *Proc of ACM SIGCOMM*, 2005.
[2] S. Chachulski, et al. Trading structure for randomness in wireless opportunistic routing. In *ACM SIGCOMM*, 2007.
[3] S. M. Das, et al. Studying Wireless Routing Link Dynamics. In *Proc. of ACM SIGCOMM/USENIX IMC*, 2007.
[4] Y. Li, et al. Effects of interference on throughput of wireless mesh networks: Pathologies and a preliminary solution. In *Proc. of HotNets-VI*, 2007.
[5] Y. Lin, et al. CodeOR: Opportunistic routing in wireless mesh networks with segmented network coding. In *Proc. of IEEE ICNP*, 2008.
[6] J. Sang Park, et al. Codecast: a network-coding-based ad hoc multicast protocol. *IEEE Wireless Communications*, 13(5), 2006.