Code Pruning in Opportunistic Routing through Bidirectional Coding Traffic Comparison

Weiping Wang, Xiaozhuan Chen, Mingming Lu, Jianxin Wang [*],
Xi Zhang [†], and Jie Wu [‡]

Abstract

Opportunistic routing (OR) significantly improves transmission reliability and network throughput in wireless mesh networks (WMNs) by utilizing the broadcast nature of the wireless medium. Through the integration of network coding (NC), the complicated coordination to select the best forwarding node in OR can be bypassed. However, the introduction of NC exacerbates the redundant-packet-transmission problem. To mitigate this issue, existing coded OR protocols either adopt the loss-rate-based approach, employ orthogonal vectors as coded feedback, or pursue the stream-based coded OR model. However, these three solutions suffer inaccuracy and obsolescence of the loss-rate measurement, false-positive/false-negative problem, and unavailability of hop-by-hop stream-based OR, respectively. To address the above problems, we propose a simple but practical coded feedback scheme, Cumulative Coding Coefficient ACKnowledgement (C^3ACK), based on the relevance between forward (coded packets received from upstream nodes) and backward coding traffic (coded packets overheard from downstream nodes), and apply C^3ACK to both batch-based and stream-based coded OR models in order to prune redundant forward and backward coding traffic. Both testbed evaluation and simulation study show that our code-pruning schemes can outperform existing approaches in terms of expected throughput and transmission count.

Keywords: *Coded feedback, code pruning, networking coding, opportunistic routing, wireless mesh networks.*

[*]Weiping Wang, Xiaozhuan Chen, Mingming Lu, and Jianxin Wang are with the School of Information Science and Engineering, Central South University, Hunan, China, 410083 P.R.C. E-mail: {wpwang, xzchen, mingminglu, jxwang}@csu.edu.cn (see http://netlab.csu.edu.cn). Mingming Lu is the corresponding author.

[†]Xi Zhang is with the Department of Electrical and Computer Engineering, Texas A&M University. E-mail: xizhang@ece.tamu.edu

[‡]Jie Wu is with the Department of Computer and Information Sciences, Temple University. E-mail: jiewu@temple.edu

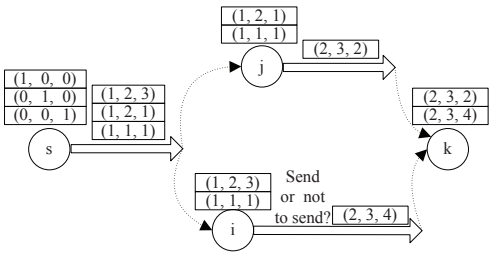

Figure 1: The illustration of the collective-space problem.

1 Introduction

Recently, OR has received an increasing amount of attention because OR, as a new routing paradigm, can significantly improve transmission reliability and network throughput in WMNs [4]. Compared to the single-path routing, instead of relying on an intended receiver to forward a packet, OR utilizes a set of potential packet receivers (candidates) to forward a packet. To reduce redundant-packet transmissions, ExOR [4] exploits a time-consuming coordination mechanism among candidates to select the best forwarder. The relatively complicated coordination has been bypassed through integrating network coding (NC) [7], where the forwarder (source) randomly mixes received *native* packets, which refer to the original packets that have not been coded yet, before forwarding (sending) them. Once the destination receives enough *innovative* packets, where a packet is innovative to a node if it is linearly independent from its previously received packets, it can recover the native packets from the source.

However, existing coded OR protocols, such as MORE [7], exacerbate the problem of redundant-packet transmissions due to the collective-space problem [19], as illustrated by the example shown in Fig. 1, where the source S has two forwarders i and j. S has three native packets p_1, p_2, and p_3, and generates three coded packets $p_1 + 2p_2 + 3p_3$, $p_1 + 2p_2 + p_3$ and $p_1 + p_2 + p_3$ (abbreviated as their corresponding *coding vectors*, i.e., the vector of coefficients that describes how to express a coded packet from the native packets). As shown in Fig. 1, neither i nor j has received enough innovative packets to cover S' knowledge space, which refers to the linear space spanned by the coding vectors, but the knowledge spaces of i and j can collectively cover S' knowledge space. Timely learning downstream nodes' knowledge spaces can help reduce redundant-packet transmissions in coded OR protocols, which, however, is a non-trivial task.

MORE [7] reduces transmission redundancy by controlling the relative transmission frequency of each forwarder through the transmission credits, which is

based on the expected transmission count (ETX) [9]. However, this transmission control scheme relies on accuracy and freshness of the statistical link-loss-rate measurements. To reflect the instantaneous channel condition, orthogonal vector [16, 17, 19] has been utilized as a compressed representation of the knowledge space. However, because an orthogonal vector is a lossy compression, it imposes the *false-positive* problem, which occurs when a node receives a feedback vector orthogonal to its knowledge space, denoting the coverage of its knowledge space, but the orthogonality is actually a coincidence. Although the false-positive problem can be mitigated by reducing the probability of coincidental orthogonality [19], it is at the expense of exacerbating the collective-space problem.

The false-positive problem motivates us to pursue an efficient coded feedback scheme without relying on orthogonal vectors. By carefully examining the forward and backward coding traffics through both theoretical analysis and simulation evaluation, we identify the relevance between forward and backward coding traffics, and propose a simple but efficient coded feedback scheme, named C^3ACK, that helps prune redundant forward coding traffic.

C^3ACK utilizes the observed relevance, appending an additional vector of coding coefficients, which, together with the coding vector, serves as coding feedback, to each node. Surprisingly, one additional vector can achieve approximately the same performance as the optimal solution (i.e., appending all vectors of coding coefficients), with less overheads than that of the orthogonal coded feedback schemes.

Furthermore, as stream-based coded OR protocols [8, 23, 27], which relax the constraint of coding within a batch (a fundamental property of the batch-based coded OR [7, 19]), can reduce the end-to-end transmission delay, and in turn improve transmission throughput, we extend our code-pruning scheme to stream-based OR models by pruning packets *covered* by downstream nodes so that the covered packets will no longer be used for generating new coded packets. Note that a packet is covered by downstream nodes if the coding vector used to express the packet can be linearly expressed by the coding vectors in the collective knowledge spaces of downstream nodes.

The rest of this paper is organized as follows. Section 2 analyzes the false-negative problems associated with the existing orthogonal-vector-based coded feedback schemes. Section 3 motivates, develops, and analyzes our proposed C^3ACK scheme. Section 4 proposes the code-pruning schemes for both the batch-based and stream-based coded OR models. Section 5 compares our code-pruning schemes with MORE, CCACK [19], the ideal scheme, and SlideOR [23] in NSclick [25]. Section 6 further evaluates our code-pruning schemes in a testbed. Section 7 summarizes the related works in this area. Section 8 concludes this paper and discusses future works.

2 The false-negative Issues of Existing Orthogonal-vector-based Feedback Schemes

The simplest orthogonal-vector-based coded feedback scheme is the null-space-based (NSB) coded feedback scheme [19], where each node directly uses orthogonal vectors to acknowledge received coded packets. Take Fig. 1 for example: the vector (x, y, z) orthogonal to node i's knowledge space should be of the form $(z, -2z, z)$, as the orthogonal vector should satisfy $(1, 2, 3) \cdot (x, y, z) = 0$ and $(1, 1, 1) \cdot (x, y, z) = 0$, i.e., $x + 2y + 3z = 0$ and $x + y + z = 0$. Similarly, the vector (x', y', z') orthogonal to node j is of the form $(-z', 0, z')$. Let $z = 1$, then, orthogonal vectors $(1, -2, 1)$ is orthogonal to $(1, 2, 3)$, $(1, 1, 1)$ at node s, because $1-4+3 = 0$ and $1-2+1 = 0$. Similarly, let $z' = 1$, then, $(-1, 0, 1)$ is orthogonal to $(1, 2, 1)$, $(1, 1, 1)$.

2.1 The false-negative issue of the NSB scheme

As a preparation for further analysis, we first analyze the false-negative probability of the NSB scheme. First of all, we give formal definitions of false-negative and false-positive.

Definition 1 *(***False negative***) A false-negative event occurs, when downstream nodes of a node have collectively covered the node's knowledge space, while the node believes its knowledge space has not been covered yet.*

Definition 2 *(***False positive***) A false-positive event occurs, when downstream nodes of a node have not covered the node's knowledge space, while the node believes its knowledge space has been covered.*

Without loss of generality, consider node i with l downstream nodes $i_1, \cdots, i_l$. Let S_i denote node i's knowledge spaces and $R(S_i)$ denote its rank. We first observe that the occurrence of the collective-space problem depends on two premises: (1) node i's knowledge space S_i can be collectively covered by downstream nodes; (2) any single downstream node alone cannot cover its knowledge space.

We then analyze the false-negative probability when receiving an orthogonal vector z_{i_j} from a downstream node i_j with the premises of the collective-space problem satisfied, and have the following theorem concerning the false-negative probability of the NSB scheme.

Theorem 1 *If the premises of the collective-space problem are satisfied, in NSB scheme, for any feedbacked orthogonal vector, the false-negative event occurs with a probability more than* $1 - \frac{1}{256}$*, assuming that the size of the Galois Field is* 2^8.

Proof: If the premises are satisfied, at least one packet q_i exists, which is innovative to any single downstream node's knowledge space, i.e., $\exists q_i$ s.t. $\forall i_j$, $q_i \notin S_{i_j}$, but not innovative to the collective knowledge space of all downstream nodes, i.e., $q_i \in \cup_{j=1}^{l} S_{i_j}$.

Upon receiving a z_{i_j}, node i will determine that its knowledge space has not been covered as long as q_i is not orthogonal to z_{i_j}, which will incur the false-negative event. Note that the probability of q_i orthogonal to z_{i_j} is actually equal to the false-positive probability according to Definition 2, because q_i is not covered by S_{i_j}, while the orthogonality falsely denotes the coverage. As the false-positive probability is approximately $\frac{1}{256}$ [19], the false-negative probability is about $1-\frac{1}{256}$ based on the above discussion.

If more than one such q_i exists, the false-negative probability further increases to $1-(\frac{1}{256})^{\sharp q_i}$, where $\sharp q_i$ denotes the number of such q_i, as the false-negative event occurs only when none of such q_i is orthogonal to z_{i_j}. □

Anyway, it is almost for sure that the false-negative event will occur if the premises of the collective-space problem satisfy. Thus, the false-negative probability can be approximated as the occuring probability of the collective-space problem.

2.2 The false-negative issue of the CCACK scheme

As previously mentioned, CCACK [19] is actually an extension of NSB. The extension is twofold. On one hand, to address the collective-space problem, two additional buffers are introduced to record the coding vectors associated with the received and transmitted packets, respectively. As a node's received packets are from its upstream nodes, a feedback vector generated through the received packets must be orthogonal to certain transmitted packets from the upstream nodes. Hence, it can mitigate the collective-space problem to some extent. On the other hand, CCACK adopts M random hash matrices to simulate the effect of M independent orthogonal vectors. As each independent orthogonal vector incurs a false-positive probability of $\frac{1}{2^8}$, the false-positive probability of M independent orthogonal vectors can reduce to $(\frac{1}{2^8})^M$.

In the case that all packets received and transmitted by an upstream node are received by its downstream nodes, CCACK can solve the collective-space problem. However, CCACK still has a serious false-negative problem, because the above case does not occur so often and the hash matrices have a side effect of increasing the false-negative probability.

In the following, we use M_{rx}^i and M_{tx}^i to denote the sets of coding vectors associated with the received and transmitted coded packets, respectively. M_v^i is

used to denote the set of coding vectors extracted from M_{rx}^i with only innovative packets included. $R(\cdot)$ is used to represent the rank of a matrix or a set. Note that $R(M_{rx}^i) = R(M_v^i) = R(S_i)$, and $|M_{tx}^i| \leq R(S_i)$.

We can analyze the false-negative issue from two perspectives. First of all, even though each vector in M_{tx}^i is the same as a certain vector in $M_{rx}^{i_j}$, it is not necessary that each feedback vector from downstream node i_j will be H-orthogonal to a vector in M_{tx}^i. Thus, we have the following lemma.

Lemma 1 *For any packet at node i, i.e., $\forall q_i \in M_{tx}^i \cup M_{rx}^i$, if it has been received by a certain downstream node i_j, i.e., $q_i \in M_{rx}^{i_j}$, the probability that it can be acknowledged by a feedback vector z_{i_j} from i_j is $\frac{\min(R(S_{i_j}), \llcorner \frac{N-1}{M} \lrcorner)}{R(S_{i_j})}$.*

Proof: Since at most $\llcorner \frac{N-1}{M} \lrcorner$ vectors can be H-orthogonal to a selected feedback vector z_{i_j}, if $R(S_{i_j}) \leq \llcorner \frac{N-1}{M} \lrcorner$, every vector in downstream node i_j will be selected, and the corresponding probability is $\frac{R(S_{i_j})}{R(S_{i_j})} = 1$; otherwise, a vector will be selected with a probability $\frac{\llcorner \frac{N-1}{M} \lrcorner}{R(S_{i_j})}$. In a word, a vector will be selected with a probability $\frac{\min(R(S_{i_j}), \llcorner \frac{N-1}{M} \lrcorner)}{R(S_{i_j})}$. As z_{i_j} is definitely orthogonal to the selected vectors, q_i will be acknowledged by z_{i_j} with the probability $\frac{\min(R(S_{i_j}), \llcorner \frac{N-1}{M} \lrcorner)}{R(S_{i_j})}$. □

The above lemma discusses the false-negative probability in the case that packets in an upstream node are exactly the same as those from downstream nodes. However, it is more general that packets in upstream nodes are not the same as those from downstream nodes. The following lemma will discuss the false-negative probability of the latter case.

Lemma 2 *For any packets at node i, i.e., $\forall q_i \in M_{tx}^i \cup M_{rx}^i$, if it has not been received by any downstream node, i.e., $q_i \notin \cup_{j=1}^{l} M_{rx}^{i_j}$, but it can be linearly expressed by the collective knowledge space S from downstream nodes, then the probability that it can be acknowledged by a feedback orthogonal vector z_{i_j} from downstream nodes is $\omega + (1-\omega) \times (\frac{1}{256})^M$, where $\omega = \frac{\sum_{j=1}^{l} 256^{R(S_{i_j})}}{256^{R(S)}}$.*

Proof: Without loss of generality, assume packet q_i satisfies the premise of this lemma. Two cases exist that q_i can be acknowledged by a z_{i_j}: (1) q_i can be linearly expressed by (the knowledge space of) a certain downstream node alone; (2) case (1) does not satisfy, but q_i is acknowledged by a feedback vector z_{i_j} due to the false-positive possibility.

The probability that q_i can be linearly expressed by downstream node i_j is $\frac{256^{R(S_{i_j})}}{256^{R(S)}}$, as the total number of vectors that can be linearly expressed by the colletive knowledge space S is $256^{R(S)}$, from which only $256^{R(S_{i_j})}$ can be linearly expressed by knowledge space S_{i_j}. As case 1 satisfies if any downstream node alone can linearly express q_i, case 1 holds with the probability of $\omega = \frac{\sum_{j=1}^{l} 256^{R(S_{i_j})}}{256^{R(S)}}$.

Straightforwardly, the probability that q_i cannot be linearly expressed by any single downstream node alone is $1-\omega$. In case 2, the probability of q_i being orthogonal to z_{i_j} is actually equal to the false-positive probability, according to Definition 2. Thus, case 2 holds with the probability $(1-\omega) \times (\frac{1}{256})^M$. Therefore, a packet satisfying the premise of this lemma will be acknowledged by a feedback orthogonal vector with the probability $\omega + (1-\omega) \times (\frac{1}{256})^M$. □

From Lemma 1 and Lemma 2, we can observe that the false-negative issue of CCACK occurs with a non-negligible probability. Thus, we have the following theorem concerning the false-negative probability.

Theorem 2 *CCACK has a non-negligible false-negative probability.*

Proof: First, we need to notice that a packet is not innovative to downstream nodes only if it has either been received by downstream nodes directly, or it can be linearly expressed by downstream nodes. If the packet has been received by downstream nodes, according to Lemma 1, it will not be acknowledged with the probability $1 - \frac{\min(R(S_{i_j}), \llcorner \frac{N-1}{M} \lrcorner)}{R(S_{i_j})}$, which is non-negligible when $R(S_{i_j})$ is larger than $\llcorner \frac{N-1}{M} \lrcorner$. For example, in CCACK's setting, $N = 32$ and $M = 4$, thus, $\llcorner \frac{N-1}{M} \lrcorner = 7$. Then, the false-negative probability is at least $\frac{1}{8}$ when $R(S_{i_j}) > 7$.

If the packet has not been received by downstream nodes, but it can still be linearly expressed by a single downstream node alone, according to Lemma 2, it will not be acknowledged with the probability $1 - \omega - (1-\omega) \times (\frac{1}{256})^M$, which is still non-negligible when the number of downstream nodes is much less than the size of Galois Field. Note that $\omega \leq \frac{l \cdot \max_j 256^{R(S_{i_j})}}{256^{R(S)}} \leq l \times \frac{1}{256}$. For example, if $l = 5$ and $M = 4$, the false-negative probability is approximately 0.98. □

3 Cumulative Coding Coefficient Acknowledgments

3.1 Motivation

The false-positive/negative issue inherent in the orthogonal-vector based feedback schemes motivates us to pursue an efficient feedback scheme independent of orthogonal vectors. We first observe that the false-positive problem originates from

the fact that an orthogonal vector is just a lossy compression of a node's knowledge space. A straightforward approach to this false-positive problem is to feedback all coding vectors instead of the single orthogonal vector, which, however, will incur enormous overheads proportional to the number of innovative packets, and thus reduce effective throughput instead.

Fortunately, the story does not end here. From a macro point view in terms of the knowledge space, it requests all coding vectors to be feedbacked. However, a further investigation of the forward/backward traffic in finer granularity can identify that, statistically, only one innovative feedback coding vector is required for each individual received innovative packet, because all innovative packets are interchangeable in terms of decoding. Moreover, in wireless networks, nodes need to compete for the wireless communication medium. Therefore, it is rare that an upstream node transmits all coded packets and then waits for feedbacks. In general, upstream nodes and downstream nodes will obtain the medium in turn, to forward coded packets and feedback coding vectors, respectively.

However, the existing coded OR models are designed to guarantee the forward traffic with little attention to the backward traffic. Thus, for each received innovative packet, it may be the case that, on average, less than one innovative coding vector will be feedbacked. The above argument is verified through our theoretical analysis and simulation evaluation, as shown in Fig. 2, where η denotes the ratio of forward and backward traffics. Fig. 2 illustrates that in more than 70% of all cases, no less than one innovative feedback coding vector will be overheard for each received innovative packets, while in more than 90% of all cases, no less than one innovative feedback coding vector will be overheard for every two received innovative packets. (The details about the theoretical analysis and simulation evaluation will be shown in Section 3.2.) It is actually an encouraging result, as imbedding two coding vectors in a packet will be sufficient to convey feedback acknowledgement information to upstream nodes in most cases.

The above observation motivates us to design a novel coded-feedback scheme based on piggyback coding vectors, which is fundamentally different from existing orthogonal-vector based schemes in that it exploits the relevance of the forward and backward traffics in the coded OR models. Before formally presenting C^3ACK, we first present our theoretical analysis and simulation evaluation about the result shown in Fig. 2.

3.2 The relevance analysis

In this section, we formally analyze the forward and backward traffic relevance in the coded OR models, so as to derive the expected number of innovative feedback coding vectors for each forwarded innovative packet. For each innovative packet

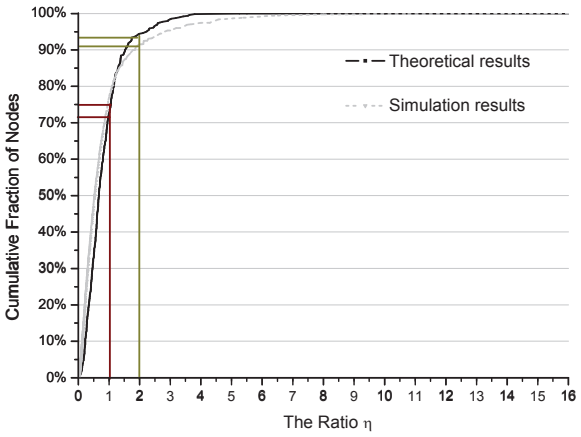

Figure 2: The illustration of the relevance of forward and backward coding traffics.

to be delivered to the destination, the coded OR models [6, 11, 24] analyzed the expected number of packets needed to be forwarded (L_j), and the corresponding ETX (T_j) by each node j, as follows,

$$L_j = \sum_{i>j} (T_i \times (1 - \varepsilon_{ij}) \Pi_{k<j} \varepsilon_{ik}) \tag{1}$$

$$T_j = \frac{L_j}{(1 - \Pi_{k<j} \varepsilon_{jk})} \tag{2}$$

Chachulski [6], Dubois-Ferriere [11], and Lu [24] have proved that, in the coded OR models, all nodes can be prioritized based on ETX. Thus, all nodes can be labeled from the lowest ID (the destination) to the highest ID (the source) in the ascending value of the ETX metric. Therefore, $i < j$ ($i > j$) can be used to denote that node i is closer (farther) to the destination than node j in terms of ETX.

Intuitively, a node closer to the destination should have a higher priority to forward an innovative packet, as it incurs less cost. According to Equation (1), L_j denotes the expected number of packets received by node j from upstream nodes, which are meanwhile not received by its downstream nodes. As those packets are innovative to node j alone, L_j actually reflects the expected number of innovative packets to be forwarded by node j.

A calculation similar to Equation (1) can be used to analyze the backward traffic as follows

$$R_j = \sum_{k<j} (T_k \times (1 - \varepsilon_{kj}) \Pi_{i>j} \varepsilon_{ki}) \tag{3}$$

where R_j reflect the expected number of innovative feedback coding vectors overheard by node j alone.

Thus, the ratio $\eta_j = L_j/R_j$ de facto reflects the relative information-flow rate of the forward traffic against the backward traffic at node j. Intuitively, for each node, the ratio should be approximately equal to 1, because, on average, the volume of the forward traffic should be equal to the volume of the backward traffic. If that is the case, on average, one innovative coded packet in forward traffic can be effectively acknowledged by one coding vector in backward traffic. Thus, no additional effort should be taken to acknowledge packets already covered by downstream nodes.

However, our numeric analysis and simulation evaluation reveal that the ratio is larger than 1 on average, which means overheard coding vectors from backward traffic are insufficient to acknowledge forward traffic. In this case, the value of η reflects the required number of imbedded coding vectors in a coded packet.

To verify the above analysis, we randomly generate 110 different topologies, each of which consists of 50 static nodes randomly distributed within an area of 1000m$\times$1000m, calculate the η value for each node in all topologies based on Eqs. (1) $\sim$ (3), and plot the cumulative distribution function (CDF) of all η values as the theoretical result shown in Fig. 2. We also simulate MORE (with the same simulation environment described in Section 5) by letting the source send batches (32 packets/batch) to the destination. For each node, the numbers of packets delivered from upstream nodes and overheard from downstream nodes are recorded, and the corresponding η value for each node is calculated accordingly. The CDF of η is plotted in Fig. 2 as the simulation result. Both theoretical and simulation results reveal that imbedding two coding vectors in a coded packet will be sufficient to convey feedback information in over 90% of all cases.

As mentioned in the previous section, the fundamental reason for this phenomenon of unequal forward and backward traffics is that the coded OR paradigm [7] was proposed to utilize neighbor nodes' idle transmission capability opportunistically to improve transmission throughput along the forwarding traffic. It has been proved [6, 11, 24] that each node in the coded OR will forward packets along an optimal routing plan to the destination.

3.3 C^{3}ACK formulation

Based on the above analysis, we can insert an additional innovative coding vector, named *backward coding vector*, into the packet header so that the backward coding vector, together with the *native coding vector*, which refers to the coding vector attached in a coded packet as packet overhead in order to express the coded packet from the native packets, form the *feedback coding vectors*. Once a node receives

sufficient innovative feedback coding vectors to cover its knowledge space, it will stop packet transmission.

Formally, we use *coding matrices*[1] M^i_{rx} and M^i_{tx} to represent coded packets received and transmitted by node i in forward traffic, respectively, and also adopt coding matrices M^i_{ox} and M^i_{bx} to denote feedback coding vectors overheard and transmitted by node i in backward traffic, respectively, as shown in Fig. 3, where intermediate node i, j, and k help source s forward packets. In Fig. 3, the white arrow below M^i_{tx} represents forward traffic transmitted by node i, the gray arrow below M^i_{bx} represents the backward traffic transmitted by node i, and the dotted (dashed) arrow denotes received packets in the forward (backward) traffic.

Note that M^i_{tx} is generated from M^i_{rx}, while M^i_{bx} is generated from $M^i_{rx} \cup M^i_{ox}$, because all the innovative packets a node should take responsibility to transmit is from its received packets in the forward traffic, while innovative feedback coding vector can be not only from what it received in forward traffic, but also from what it overheard in backward traffic.

We also introduce an operational matrix M^i_{op}, defined as $M^i_{rx} \cup M^i_{ox}$, to simplify the description. By comparing $R(M^i_{ox})$ with $R(M^i_{op})$, the stopping rule of C^{3}ACK, which decides whether to stop transmission, can be conveniently described as follows: 1) if $R(M^i_{ox}) = R(M^i_{op})$, node i stops transmission, 2) if $R(M^i_{ox}) < R(M^i_{op})$, node i continues transmission. Simply comparing $R(M^i_{ox})$ with $R(M^i_{rx})$ does not work, as M^i_{ox} may contain innovative-packet information directly transmitted from upstream nodes to downstream nodes that does not reach node i. The correctness of the stopping rule will be discussed in Section 3.4.

Although our C^{3}ACK is simple, it is practical and efficient. From the point of view of practicality, simple is good. We illustrate our coded feedback scheme through the example in Fig. 3. Although node s overheard 4 feedback coding vectors, it still decides to continue transmission, as $R(M^s_{ox}) = 2 < R(M^s_{op}) = 3$. As for nodes i and j, since $R(M^i_{ox}) = R(M^i_{op}) = 2$ and $R(M^j_{ox}) = R(M^j_{op}) = 2$, both of them will stop transmission until receiving innovative packets.

3.4 False-positive and false-negative analysis

In this section, we will first prove that C^{3}ACK is false-positive free, and then discuss the false-negative issue. Before that, we need to give the following definitions and lemmas.

[1] A coding matrix is the matrix formed by a set of coding vectors. In the case of no confusion, with somewhat notation abuse to reduce the number of notations, we also use the coding matrix to represent the set of coding vectors that form the coding matrix. Moreover, a coding matrix can be compressed by keeping only the innovative coding vectors to reduce storage overhead.

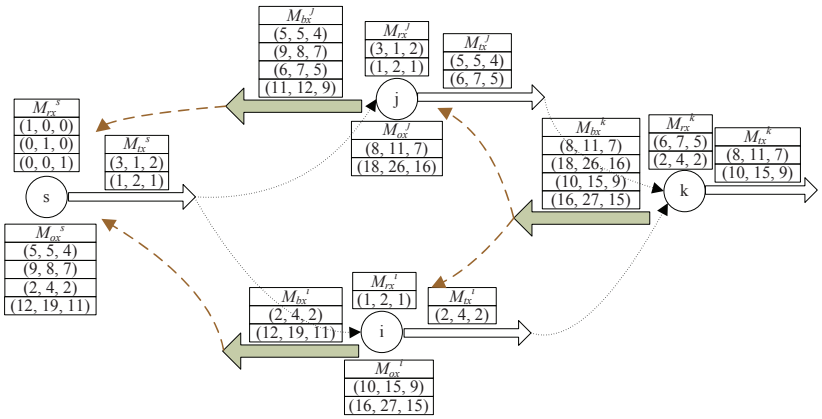

Figure 3: The cumulative coding coefficient acknowledgement scheme in unreliable multicast.

Definition 3 *(**Set linear expression**) Given two coding vector sets A and B, if $\forall a \in A$ can be linearly expressed by B, we say that A can be linearly expressed by B.*

The reason that the stopping rule of C^{3}ACK can be used to reflect the knowledge space coverage is illustrated through the following lemma concerning the set linear expression.

Lemma 3 *Let A and B be two coding vector sets; if A can be linearly expressed by B, then $R(B \cup A) = R(B)$, and if A cannot be linearly expressed by B, then $R(B \cup A) > R(B)$.*

Proof: If A can be linearly expressed by B, all vectors in A can be linearly expressed by B. Hence, $R(B \cup A)$ (the rank of the coding matrix associated with $B \cup A$) is equal to $R(B)$. If A cannot be linearly expressed by B, at least one coding vector in A cannot be linearly expressed by B. Thus, $R(B \cup A) > R(B)$. □

Lemma 4 *Linear expression of the coding vector set is transitive, i.e., if $R(B \cup A) = R(B)$ and $R(C \cup B) = R(C)$, then $R(C \cup A) = R(C)$.*

Proof: Without loss of generality, let A :$\{a_1, a_2, \cdots, a_l\}$, B :$\{b_1, b_2, \cdots, b_m\}$, and C :$\{c_1, c_2, \cdots, c_n\}$. Based on the assumption, for any a_i, we have $a_i = \sum_{j=1}^{m} \alpha_{ij} b_j$, where $i = 1, \cdots, l$ and α_{ij} is an coefficient, and $b_j = \sum_{j=1}^{n} \beta_{jk} c_k$, where $j = 1, \cdots, m$ and β_{jk} is an coefficient. We can derive $a_i = \sum_{j=1}^{m} \alpha_{ij} (\sum_{j=1}^{n} \beta_{jk} c_k) = \sum_{j=1}^{n} (\sum_{j=1}^{m} \alpha_{ij} \beta_{jk}) c_k$. Therefore, A can be linearly expressed by C. By Lemma 3, $R(C \cup A) = R(C)$ holds. □

Lemma 5 *If* $R(B_1 \cup A_1) = R(B_1)$ *and* $R(B_2 \cup A_2) = R(B_2)$*, then* $R((B_1 \cup B_2) \cup (A_1 \cup A_2)) = R(B_1 \cup B_2)$.

Proof: Since A_1 can be linearly expressed by B_1 and A_2 can be linearly expressed by B_2, each element of A_1 and A_2 can be linearly expressed by B_1 and B_2, respectively. Therefore, $A_1 \cup A_2$ can be linearly expressed by $B_1 \cup B_2$ according to Definition 3. According to Lemma 3, $R((B_1 \cup B_2) \cup (A_1 \cup A_2)) = R(B_1 \cup B_2)$ satisfies. □

Lemma 6 *If* $R(B \cup A) = R(B)$*, then* $R((B \cup A) \cup C) = R(B \cup C)$.

Proof: $R(B \cup A) = R(B)$ denotes that A can be linearly expressed by B. $R((B \cup C) \cup A) = R(B \cup C)$ directly follows because A can also be linearly expressed by $B \cup C$. Since $R((B \cup A) \cup C) = R((B \cup C) \cup A)$, $R((B \cup A) \cup C) = R(B \cup C)$ can be easily derived. □

Lemma 7 *For any node* j*, its overheard coding vectors are covered by its downstream nodes' knowledge space, i.e.* $R(\cup_{k<j} M_{rx}^k \cup M_{ox}^j) = R(\cup_{k<j} M_{rx}^k)$.

Proof: We prove this lemma by induction on the closeness to the destination. Because no downstream nodes exist for the destination, the inductive basis is trivial as $R(\cup_{k<d} M_{rx}^k \cup M_{ox}^d) = R(\cup_{k<d} M_{rx}^k) = 0$. For the inductive steps, assume that, for any $k < j$, $R(\cup_{k'<k} M_{rx}^{k'} \cup M_{ox}^k) = R(\cup_{k'<k} M_{rx}^{k'})$.

Then, consider node j. Since its overheard coding vectors are from its downstream nodes with possible packet loss, we have $M_{ox}^j \subseteq \cup_{k<j} M_{bx}^k$, which infers

$$R(\cup_{k<j} M_{bx}^k \cup M_{ox}^j) = R(\cup_{k<j} M_{bx}^k) \tag{4}$$

For any node k, since M_{bx}^k is generated from M_{op}^k, we obtain $R(M_{op}^k \cup M_{bx}^k) = R(M_{op}^k)$ by Lemma 3. Then, applying Lemma 5 can derive

$$R((\cup_{k<j} M_{op}^k) \cup (\cup_{k<j} M_{bx}^k)) = R(\cup_{k<j} M_{op}^k) \tag{5}$$

As $M_{op}^k = M_{ox}^k \cup M_{rx}^k$, $\cup_{k<j} M_{op}^k$ can be written as $(\cup_{k<j} M_{rx}^k) \cup (\cup_{k<j} M_{ox}^k)$, which is equal to $\cup_{k<j}(\cup_{k'<k} M_{rx}^{k'} \cup M_{ox}^k)$ because $M_{rx}^{k'} \cup M_{rx}^{k'} = M_{rx}^{k'}$. Hence, $R(\cup_{k<j} M_{op}^k) = R(\cup_{k<j}(\cup_{k'<k} M_{rx}^{k'} \cup M_{ox}^k))$. The inductive hypothesis, $R(\cup_{k'<k} M_{rx}^{k'} \cup M_{ox}^k) = R(\cup_{k'<k} M_{rx}^{k'})$, can infer $R(\cup_{k<j}(\cup_{k'<k} M_{rx}^{k'} \cup M_{ox}^k)) = R(\cup_{k<j}(\cup_{k'<k} M_{rx}^{k'}))$ by applying Lemma 5. Hence,

$$R(\cup_{k<j} M_{op}^k) = R(\cup_{k<j}(\cup_{k'<k} M_{rx}^{k'})) = R(\cup_{k<j} M_{rx}^j) \tag{6}$$

Also, since $M_{op}^k = M_{rx}^k \cup M_{ox}^k$, it directly follows $M_{op}^k \cup M_{rx}^k = M_{op}^k$, which infers $R(M_{op}^k \cup M_{rx}^k) = R(M_{op}^k)$. It further derives $R((\cup_{k<j} M_{rx}^k) \cup (\cup_{k<j} M_{op}^k)) = R(\cup_{k<j} M_{op}^k)$ by apply Lemma 5. Together with Equation (6), it can derive

$$R((\cup_{k<j} M_{rx}^k) \cup (\cup_{k<j} M_{op}^k)) = R(\cup_{k<j} M_{rx}^j) \tag{7}$$

$R(\cup_{k<j} M_{rx}^k \cup M_{ox}^j) = R(\cup_{k<j} M_{rx}^k)$ can be inferred by applying Lemma 4 to Equations (4), (5), and (7). □

Now, we are ready to prove the following theorem concerning the false-positive free property.

Theorem 3 *C^{3}ACK is false-positive free.*

Proof: This theorem is proved through contradiction. Assume false-positive possibility exists in C^{3}ACK. Then, a node j exists, s.t. $R(\cup_{k<j} M_{rx}^k \cup M_{rx}^j) > R(\cup_{k<j} M_{rx}^k)$, and $R(M_{op}^j) = R(M_{ox}^j)$. According to Lemma 7, we have $R(\cup_{k<j} M_{rx}^k \cup M_{ox}^j) = R(\cup_{k<j} M_{rx}^k)$, which derives $R(\cup_{k<j} M_{rx}^k \cup M_{op}^j) = R(\cup_{k<j} M_{rx}^k \cup M_{rx}^j)$ by applying Lemma 6. In turn, it derives $R(\cup_{k<j} M_{rx}^k \cup M_{op}^j) > R(\cup_{k<j} M_{rx}^k)$ based on the inequality in the assumption. By applying Lemma 6 to the equation in the assumption, we can obtain $R(\cup_{k<j} M_{rx}^k \cup M_{op}^j) = R(\cup_{k<j} M_{rx}^k \cup M_{ox}^j)$, which infers $R(\cup_{k<j} M_{rx}^k \cup M_{op}^j) = R(\cup_{k<j} M_{rx}^k)$ according to Lemma 7. By comparing with the previously derived inequality, we can conclude $R(\cup_{k<j} M_{rx}^k) > R(\cup_{k<j} M_{rx}^k)$, a contradiction. Therefore, C^{3}ACK is false-positive free. □

Before discussing the false-negative issue, we first analyze the relationship between C^{3}ACK and the collective-space problem. A thorough examination of CCACK can identify the essence of the collective-space problem, where feedbacks from downstream nodes cannot construct a collective knowledge space of all downstream nodes. The fundamental reason is that two orthogonal vectors from different downstream nodes are not complementary to each other, because an orthogonal vector generated from a node is usually just one of the plausible vectors orthogonal to the node's knowledge space. Due to the uncertainty, an orthogonal vector cannot uniquely determine its corresponding knowledge space. Thus, two orthogonal vectors cannot infer a collective knowledge space and, hence, are not complementary.

On the contrary, we have the following lemma concerning the complementarity among feedbacks in C^{3}ACK.

Lemma 8 *If the knowledge spaces of two downstream nodes i_j and i_k of a given node i are not inclusive to each other, i.e., $S_{i_j} - S_{i_k} \neq \emptyset$ and $S_{i_k} - S_{i_j} \neq \emptyset$, then the coded feedback from one downstream node can convey information complementary to the knowledge space of the other downstream node.*

Proof: Without loss of generality, consider any coded feedback from node i_k, which contains two coding vectors $q^1_{i_k}$ $q^2_{i_k}$ generated from $M^{i_k}_{rx}$ and $M^{i_k}_{bx}$, respectively. Consider $q^1_{i_k}$, which can be linearly expressed by the vectors in $M^{i_k}_{rx}$. We can conclude $q^1_{i_k} \notin S_{i_j}$. Otherwise, a contradiction can be derived as follows. From the assumption of this lemma, we can infer that at least one vector $q_{i_k} \in M^{i_k}_{rx}$ exists, s.t. $q_{i_k} \notin S_{i_j}$. Otherwise, $S_{i_k} - S_{i_j} = \emptyset$, which contradicts the assumption. Since $q^1_{i_k}$ can be linearly expressed by the vectors in $M^{i_k}_{rx}$, which includes q_{i_k}, the only vector in $M^{i_k}_{rx}$ but not in S_{i_j}, we can easily derive $q_{i_k} \in S_{i_j}$, a contradiction. Therefore, since $q^1_{i_k} \notin S_{i_j}$ denotes the coded feedback from i_k containing information complementary to S_{i_j}, the lemma is proved. □

Theorem 4 *C^{3}ACK is free from the collective-space problem.*

Proof: According to Lemma 8, in C^{3}ACK, the coded feedbacks from downstream nodes with complementary information are also complementary to each other. Thus, those coded feedbacks from downstream nodes can accumulate coding information to build up a collective knowledge space of all downstream nodes. Therefore, C^{3}ACK is free from the collective-space problem. □

Although C^{3}ACK is free from the collective-space problem, it still suffers from the false-negative issue, as each coded feedback can only contain two coding vectors. If the rank of downstream nodes' collective knowledge space is 32, a node needs to receive 16 coded feedbacks to construct the collective space. However, the false-negative problem associated with C^{3}ACK is not so bad as it seems to be at the first glance, because most of the coded feedback from downstream nodes are overheard by upstream nodes as downstream nodes forward packets to their downstream nodes. As shown in Section 3.2, for every two packets forwarded, a coded feedback will be overheard with a probability over 90%.

Moreover, the false-negative problem can be further mitigated by the following two design considerations in C^{3}ACK. First, the backward coding vectors are generated from both received coded packets and overheard coding vectors, as compared with the forward coded packets, which are generated from received coded packets alone. This will potentially increase the complementarity among overheard coding vectors, which in turn mitigates the false-negative problem. Second, downstream nodes will explicitly send acknowledge packets with two coding vectors, named LACK, upon receiving non-innovative packets from upstream nodes. The LACK packet can further mitigate the false-negative problem.

It has been discussed in CCACK [19], in the coded OR models, cost of a false-positive event is considerably higher than that of a false-negative event, because the latter simply causes redundant packet transmission, while the former causes

the failure of decoding at the destination node, due to insufficient coded packets.

4 Code Pruning in Coded OR Models

In this section, we propose our code-pruning schemes for both batch-based and stream-based coded OR models.

4.1 Code pruning in the batch-based coded OR model

For the batch-based coded OR model, we implement code pruning by imbedding our C^{3}ACK into existing batch-based OR protocols, such as MORE [7]. A batch-based OR protocol usually splits a large file into a bunch of batches, each of which consists of N native packets. It requires the source to generate coded packets from native packets within the current batch until the source receives an end-to-end ACK from the destination. Correspondingly, every coded packet generated at any forwarder is, in fact, a linear combination of native packets within the current batch. Upon receiving the end-to-end ACK, the source will proceed to the next batch. Forwarders will stop transmitting coded packets from the current batch upon receiving the end-to-end ACK or coded packets from the next batch. Upon receiving innovative packets, forwarders re-encode received packets and forward them. Forwarder sets are selected based on ETX metric.

During packet transmission, the source and forwarders adopt C^{3}ACK to determine whether their knowledge spaces have been covered or not. C^{3}ACK maintains three coding matrices M_{rx}, M_{ox} and M_{op} for each flow, as shown in Fig. 3. Note that C^{3}ACK does not have to maintain M_{tx} and M_{bx}, which are listed just for illustration. C^{3}ACK also maintains two states: STATE_TX (transmission state) and STATE_IDLE (idle state). Upon receiving an innovative packet, a forwarder checks its state. If its state is STATE_IDLE, it will change state to STATE_TX, and save the native coding vector into both M_{rx} and M_{op}. When overhearing an innovative backward coding vector, it saves the vector into both M_{ox} and M_{op}, and compares $R(M_{ox})$ and $R(M_{op})$. If $R(M_{ox}) < R(M_{op})$, the state sets to STATE_TX. If $R(M_{ox}) = R(M_{op})$, the state sets to STATE_IDLE.

When the MAC layer protocol allows a node to transmit, it generates a coded packet from received coded packets and forwards it if the state is STATE_TX and the credit counter [7] (the transmission condition in MORE) is positive. Once the destination can decode the received coded packets within the current batch, it sends an ACK packet, feed-backed through a predetermined shortest path, to the source to inform of the successful delivery of the batch.

Algorithm 1 provides a packet processing procedure at an intermediate node.

Algorithm 1 Packet processing procedure at forward node i

1: //sending a coded packet P at intermediate node i
2: **if** tx_state_ = STATE_TX **then**
3: construct a coded packet P by random linearly combining of all received coded packets;
4: construct an ACK vector by random linearly combining of all vectors in M_{op}^{i} and embed it in packet P;
5: **end if**

6: //upon receiving an ACK at intermediate node i
7: stop transmitting packets from batch k;
8: forward it to upstream nodes on the shortest path;
9:
10: //upon receiving an LACK at intermediate node i
11: **if** LACK comes from downstream nodes **then**
12: store the LACK vectors of the LACK in M_{ox}^{i} and M_{op}^{i};
13: **if** $R(M_{ox}^{i}) = R(M_{op}^{i})$ **then**
14: set tx_state_ = STATE_IDLE;
15: **end if**
16: **end if**

17: //upon receiving a data packet P at intermediate node i
18: **if** P comes from upstream nodes **then**
19: **if** P is innovative **then**
20: store the coding vector of the packet in M_{rx}^{i} and M_{op}^{i};
21: set tx_state_ = STATE_TX;
22: **else**
23: discard the packet and send a LACK for batch k;
24: **end if**
25: **else**
26: store both the coding vector and the ACK vector of the packet in M_{ox}^{i} and M_{op}^{i};
27: **if** $R(M_{ox}^{i}) = R(M_{op}^{i})$ **then**
28: set tx_state_ = STATE_IDLE;
29: **end if**
30: **end if**

4.2 Code pruning in the stream-based coded OR model

Compared to the batch-based OR, which codes packets separately within batches, the stream-based OR allows coded packets to be transmitted over WMNs as coding streams. More specifically, stream-based OR protocols [8, 23, 27] need to maintain an additional *native-packet queue* and a *coding window* at the source. The coding window stores the *coding basis*, i.e., the native packets used to generate coded packets, and the queue contains a stream of native packets waiting to be put into the coding window. Upon receiving a feedback from the destination containing the acknowledgements of the decoded or seen native packets [23], the source will slide the coding window by removing the acknowledged native packets and filling the coding window with the native packets from the queue.

However, existing stream-based coded OR implementations only support end-to-end coding stream, which does not fully utilize the benefit of stream-based coding. In this work, we design a hop-by-hop stream-based coded OR protocol, named StreamOR, which extends our batch-based OR protocol, C^{3}ACK, by allowing intermediate nodes to remove individual coded packets that have been covered by downstream nodes.

Compared to C^{3}ACK, StreamOR requires the source and forwarders to maintain an additional coded-packet queue and a coding window. During packet transmission, the source and forwarders need to determine whether each individual packet is covered by downstream nodes. If yes, covered packets will be removed from the coding window and new coded packets (if any) will be fetched from the queue to fill the coding window. Note that the coded-packet queue and coding window collectively replace the role of the coding matrix M_{rx} to store the receiving coding vectors. The size of the coding window has a maximum limit. If the incoming coding stream overwhelms the coding window, overflowed coded packets will be saved in the coded packet queue in the manner of FIFO.

For the backward traffic, each overheard innovative coding vector, will be saved to M_{ox}. The novelty of each individual coded packet in the coding window and the coded-packet queue will be evaluated as follows. For any packet q in the coding window or the coded-packet queue, if $R(M_{ox} \cup \{q\}) = R(M_{ox})$, the packet will be considered obsolete and will be removed from the coding window, and a new coded packets will be put into the coding window from the queue.

Note that orthogonal vectors can also be used as feedback for the hop-by-hop stream-based OR model. However, the orthogonal vector suffers from both false-positive and false-negative problems. First, the false-positive problem can be propagated back to upstream nodes, which may finally cause the source to believe that certain packets (not received by the destination) have been delivered to the destination. This will incur serious end-to-end transmission delay, canceling the benefit of

the stream-based coded OR. Second, the false-negative problem can also be propagated back to upstream nodes, which in turn incurs significant unnecessary packet transmissions.

5 Simulation Evaluations

As the size of a testbed is usually limited, which cannot evaluate large-scale WMNs, and the ideal coded feedback scheme (the straightforward approach mentioned in Section 3.1) cannot be implemented in the testbed, we first evaluate our coding pruning schemes in Nsclick [25] with the Madwifi extension [21]. The nsclick simulator embeds the Click modular router architecture [18] into the ns2 simulator [1], such that the routing protocols are developed with Click, while the physical (wireless) medium is simulated using ns2. The advantage of using nsclick is that we can easily transplant our code developed for the simulator to the testbed.

5.1 Simulation environment

The physical wireless medium is simulated through the lognormal shadowing propagation model adopted in ns2 [1]. Let d_{ij} and p_{ij} denote the distance and the delivery probability for the link from node i to node j, respectively. p_{ij} can be represented approximately as a function of d_{ij}, as follows:

$$p_{ij} = \begin{cases} 1 - ((\frac{d_{ij}}{R})^{2\beta}) \times 1/2, & if\ d_{ij} \leq R \\ ((\frac{2R-d_{ij}}{R})^{2\beta}) \times 1/2, & if\ R < d_{ij} \leq 2R \\ 0, & otherwise \end{cases}$$

Here, β is the power attenuation factor ranging from 2 to 6, which is set as $\beta = 2$ in our simulation, and R is defined as the distance satisfying $p_{ij}(R) = 0.5$.

If not specified otherwise, the simulation topology is a network of 50 static nodes randomly deployed in a 1000m $\times$ 1000m area. Our simulation environment in ns2 is based on the 802.11b standard. The transmission range and the carrier-sensing range are the default values 250m and 550m, respectively. We set $R =$ 125m, as $p_{ij} = 0$ when $d_{ij} = 2R$. All routing protocols operate under the fixed bit-rate 11Mb/s. We set the payload size to 1.4KB, while the size of the packet header depends on the routing protocols. The batch size is set to 32 packets for MORE, CCACK, OPTACK and C^{3}ACK. Also, we have the RTS/CTS handshake disabled, as most operational networks do. In each simulation, the source transmits 17MB of file to the destination as fast as possible. In our simulation, two nodes are regarded as neighbors to each other if the link quality between them is sufficient to achieve a transmission success probability higher than 0.1.

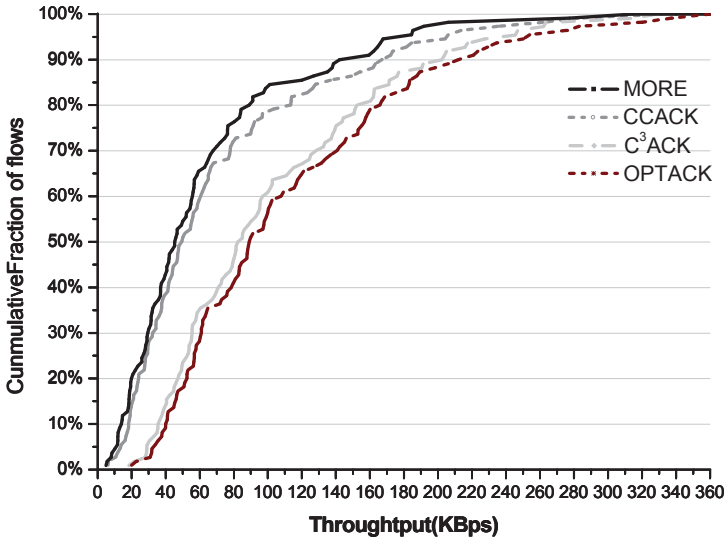

Figure 4: CDF of throughputs achieved with MORE, CCACK, OPTACK, and C^{3}ACK.

5.2 The analysis of simulation results

The simulations are executed on 110 different randomly generated topologies, and each protocol runs 12 times on every topology. In each scenario, every source sends a 17MB file, consisting of 1400-byte packets. Fig. 4 plots the CDF of the throughputs of the 110 topologies achieved with MORE, CCACK, C^{3}ACK, and OPTACK, which allows any node to directly access the knowledge space of its downstream nodes [19]. Although the OPTACK is impractical, it can serve as a benchmark. To maintain fairness, data packets in OPTACK are imbedded with a 32-bit ACK vector, the same size of CCACK and C^{3}ACK. The direct access of downstream nodes' knowledge space is implemented through shared memory in the simulator. As OPTACK does not have the false-positive and false-negative problems, it should have the best performance. The result in Fig. 4 shows that C^{3}ACK outperforms both MORE and CCACK. More specifically, there are about 15% topologies with throughputs greater than 100KBps, and about 20% in CCACK, while there are about 40% in C^{3}ACK. The average throughputs with MORE, CCACK and C^{3}ACK are 63.77KBps, 72.51KBps and 103.40KBps, respectively. Moreover, the OPTACK's expected throughput is 112.49 kb/s, only 9.37% higher than C^{3}ACK. This result illustrates that C^{3}ACK can effectively handle the false-positive and false-negative problems with little performance loss.

Fig. 5 gives the CDF of the relative throughput improvement ratio for C^{3}ACK

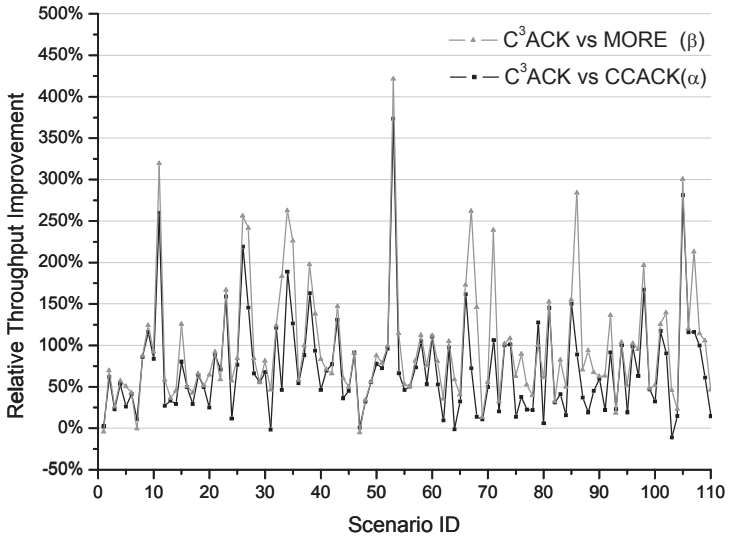

Figure 5: Relative throughput improvement of C^{3}ACK over CCACK, and C^{3}ACK over MORE on each topology.

over CCACK, and C^{3}ACK over MORE on 110 different topologies. To evaluate the throughput gain, we defined $\alpha_i = \frac{T^i_{C^3ACK} - T^i_{CCACK}}{T^i_{CCACK}} \times 100\%$, $\beta_i = \frac{T^i_{C^3ACK} - T^i_{MORE}}{T^i_{MORE}} \times 100\%$, where T^i_{MORE}, T^i_{CCACK} and $T^i_{C^3ACK}$ are the throughputs of different protocols on topology i, respectively. Fig. 5 shows the value of α and β on 110 different topologies, and Fig. 6 plots the responding CDF of α and β. We observe that C^{3}ACK achieves a higher throughput than both CCACK and MORE in 107 topologies out of 110. The average throughput gain of C^{3}ACK over CCACK and MORE are 72.2% and 98.7%, respectively. For some challenged topologies with the mutil-hops and mutil-path from the source to the destination, the throughput achieved with C^{3}ACK is 2-3x higher than CCACK. Furthermore, for some challenged topologies with mutil-hops extremely far away from the source to the destination, the throughput of C^{3}ACK is 2-4x higher than MORE.

We observe that most of the throughput improvement of C^{3}ACK over CCACK distribution is at about 70% on 110 topologies. Only in three topologies, the throughput improvements are negative. The underlying reason is that only one single path exists in these topologies, and the single path has a very limited hop count. CCACK will not have the false-negative problem in a single-path topology, where CCACK will outperform C^{3}ACK, because one orthogonal vector can convey more coding information than two ordinary coding vectors. Thus, in that case,

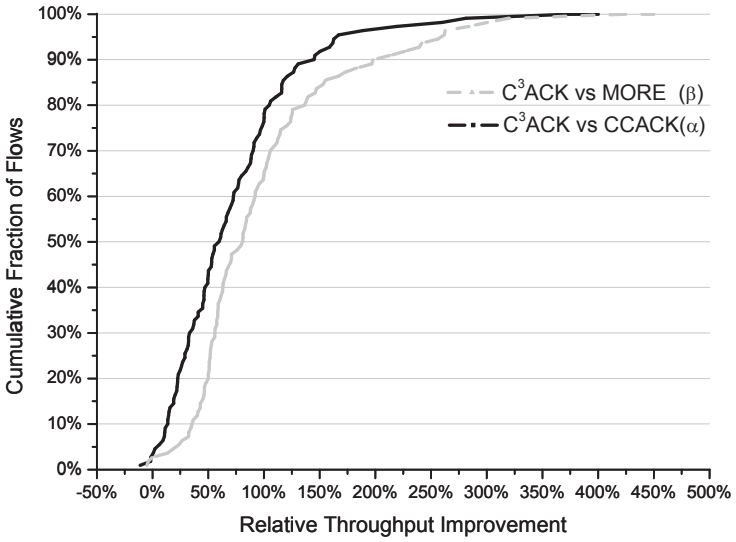

Figure 6: CDF of relative throughput improvement of C^{3}ACK over CCACK, and C^{3}ACK over MORE.

the throughput of C^{3}ACK is slightly worse than that of CCACK. In the worst case, $\alpha = -11.2\%$.

Similarly, in certain scenarios, the values of β are relatively low, 3 out of which are even negative. In the worst case, $\beta = -4.94\%$. The underlying reason is that only short paths exist in those scenarios, which has a very limited hop count. In such a short-path topology, an ACK packet in MORE will be delivered to the source quickly. In that case, C^{3}ACK has no advantage to utilize the delay-bandwidth product as the ACK feedbacked towards the source. Moreover, the packet overhead of C^{3}ACK is slightly larger than that of MORE. Thus, in those cases, MORE will perform slightly better than C3ACK.

To identify where the gain for C^{3}ACK comes from, we count the number of packet transmissions associated with MORE, CCACK, C^{3}ACK, respectively, for every node on all the 110 randomly generated topologies. We also calculate the predicted number of transmissions obtained from the offline ETX-based credit calculation [7] as a baseline because it reflects the ideal case when the online channel can be accurately reflected by the offline measurement, under which MORE can minimize its packets transmissions.

Fig. 7 plots the total number of packet transmissions associated with MORE, CCACK, and C^{3}ACK, respectively, on all the 110 randomly generated topologies. Note that the total number of transmissions actually reflects the overall overheads

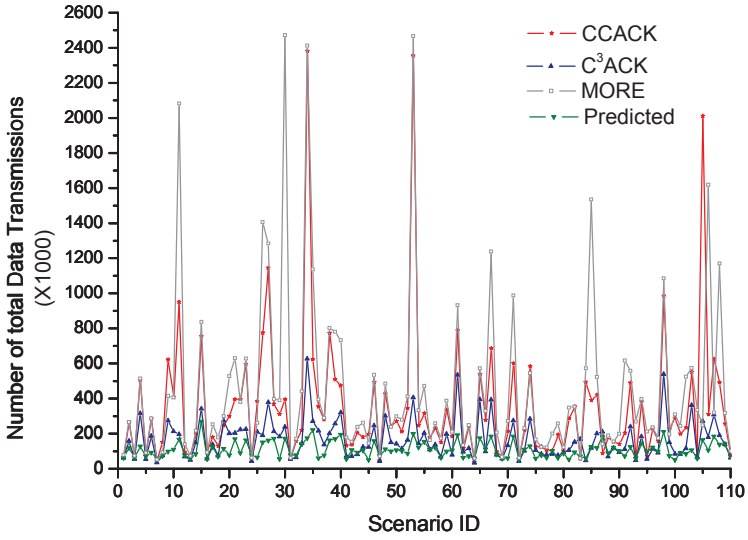

Figure 7: Total number of data transmissions per scenario.

when data can be successfully delivered. The smaller the total number of transmissions, the better the performance. From Fig. 7, we can observe that nodes running MORE and CCACK intend to transmit more packets than the predicted number in most topologies. The actual number is often twice more than the predicted number, and even up to 7-8x and 6-7x the predicted number concerning MORE and CCACK, respectively, in some scenarios.

The much higher volume of packet transmissions associated with CCACK verifies our analysis concerning the false-negative issue of CCACK (refer to Section 2.2). As CCACK has a non-negligible false-negative probability, it will cause a large number of unnecessary packet transmissions, which in turn greatly increases the total number of packet transmissions associated with CCACK. In addition, nodes running CCACK cannot stop transmitting in time when the destination node has already received enough packets, due to intense channel competition, which delays the arrival of ACK and wastes partial wireless bandwidth. Moreover, the source running CCACK cannot stop transmitting packets in a timely manner. With long paths, this may result in a large number of unnecessary transmissions, as the ACK travels towards the source.

As for MORE, compared with the predicted number, the rather higher volume of packet transmissions is mainly due to the prediction inaccuracy of the offline ETX measurement, which can hardly reflect the randomness of the online channel status. Even in the case of the absence of background traffic, based on the

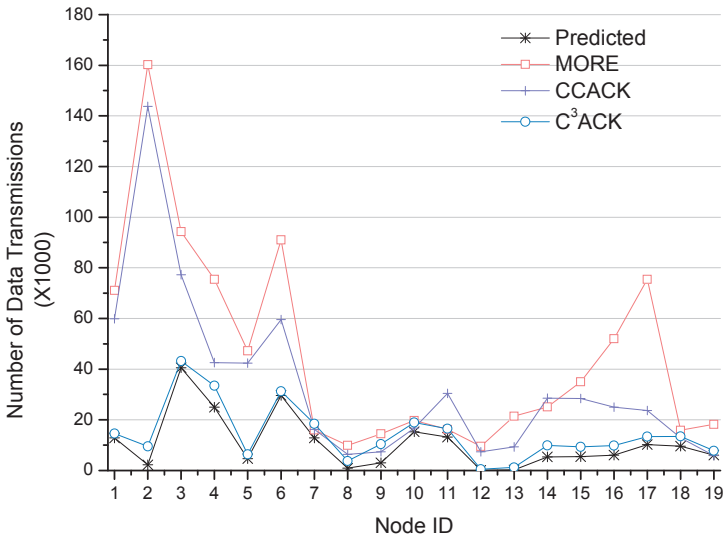

Figure 8: Total number of packet transmissions per node for one scenario.

offline ETX measurements, MORE usually incurs twice the predicted number, and even up to 7-8x the predicted number in some scenarios. Moreover, the source in MORE keeps transmitting packets until it receives an ACK from the destination. In the topologies of long paths, this may result in a large number of unnecessary transmissions during the period of the ACK traveling from the destination to the source.

As a comparison, the number of data transmissions associated with C^{3}ACK is much lower than that of MORE and CCACK in all scenarios. In most scenarios, it is close to the predicted number, and even lower in certain scenarios. Bandwidth is typically the scarcest resource in a wireless network. Thus, the natural approach to increasing wireless throughput is to decrease the number of transmissions necessary to deliver a packet from the source to the destination [9, 10, 13]. Compared with MORE and CCACK, C^{3}ACK can greatly reduce the number of redundant packet transmissions and increase the effective throughput accordingly. The fundamental reason is that the double coding-coefficient feedback scheme adopted by C^{3}ACK improves the accuracy of knowledge space coverage, which can greatly reduce the redundant transmissions. Therefore, C^{3}ACK outperforms both MORE and CCACK in terms of throughput.

To further illustrate the advantage of C^{3}ACK, we compare it with CCACK and MORE in a finer granularity, as shown in Fig.8, which compares the number of packet transmissions of individual nodes associated with MORE, CCACK,

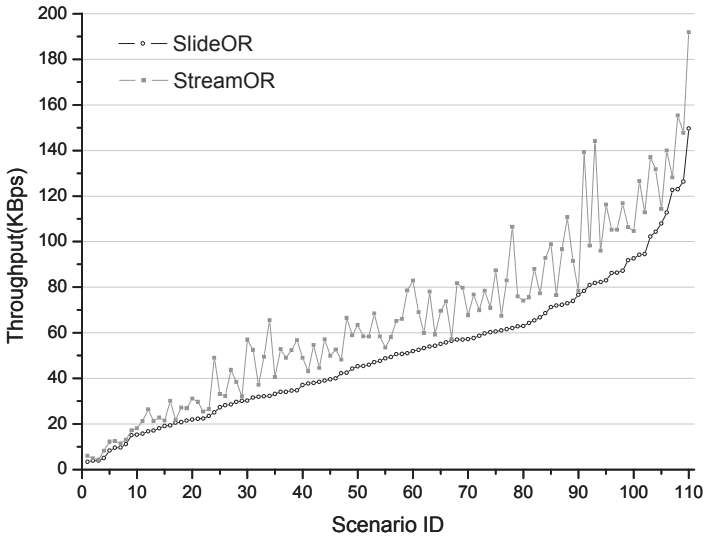

Figure 9: StreamOR Throughput.

and C^{3}ACK, respectively, in a certain scenario. Nodes are sorted with respect to their ETX distance to the destination, i.e., node 1 is the source and node 19 is the forwarding node closest to the destination. Overall, C^{3}ACK still produces significantly fewer transmissions than those of MORE and CCACK, respectively.

To evaluate our stream-based coded OR protocol, StreamOR, we compare it to the most cited stream-based OR protocol, SlideOR [23]. Fig. 9 and Fig. 10 plot the throughputs, and the throughput CDFs of the 110 topologies achieve with StreamOR and SlideOR, respectively. The simulation results illustrate that StreamOR outperforms SlideOR, as the expected throughputs of StreamOR and SlideOR are 66.81kb/s and 51.43kb/s, respectively, with average throughput improvement 32.64%.

5.3 C^{3}ACK' overhead

Finally, we would like to estimate C^{3}ACK's overhead compared to CCACK. Similar to [19], we discuss three types of overhead: coding calculation, buffer requirement, and packet header overhead.

(a) Coding Cost of Calculation: Apparently, CCACK's coding calculation is much more complicated than C^{3}ACK's, because the construction of orthogonal vector in CCACK is far more complicated than the construction of coding vector in C^{3}ACK. To verify it, we measured the calculation cost for various operations needed during the processing of one packet delivered from source to destination on 110 different topologies. Table 1 provides the average value and the standard

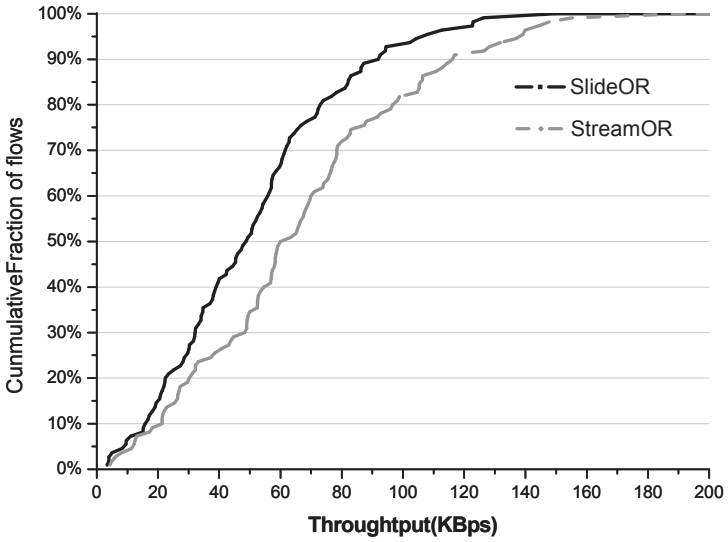

Figure 10: StreamOR Throughput CDF.

deviation of calculation cost for each operation. The costs are estimated in terms of GF(2^8) multiplications, which are the most expensive operations involved in coding/decoding [7].

From Table 1, we can observe that the most significant difference between CCACK and C^{3}ACK is the different costs to construct an orthogonal vector and a coding vector. The construction of an orthogonal vector in CCACK requires 11,464 multiplications on average, while construction of a coding coefficient vector in C^{3}ACK requires only 598 multiplications. Thus, the overhead on packet transmission of C^{3}ACK is significantly less than that of CCACK.

(b) Buffer size Requirement: Compared to CCACK, nodes in C^{3}ACK need to maintain two additional buffers for coding-coefficients. One buffer is named M^i_{ox}, which stores overheard coding coefficients from downstream nodes. The other buffer is M^i_{op}, which is an operational buffer used to store coding coefficients from both forward and backward traffic (i.e. $M^i_{rx} \cup M^i_{ox}$). In our implementation, the total size of M^i_{ox} and M^i_{op} is $2 \times 32 \times 32 = 2$KB.

(c) Header Overhead: Both CCACK and C^{3}ACK add a K-byte ACK vector in packet header, where N ($N = 32$ in our experiments) is the batch size. Thus, C^{3}ACK's header is the same as CCACK's in our implementation.

Table 1: The coding overhead of C^{3}ACK and CCACK in terms of GF(2^8) multiplication. Operations in packet transmission and reception are marked with $*$ and $\divideontimes$ respectively.

Operation	**Avg.** (CCACK/C^{3}ACK)	**Std. Dev** (CCACK/C^{3}ACK)
Coded pkt construction on src*	45824/45824	0/0
Coded pkt construction on FNs*	29119/26752	16924/15146
Orthogonal vector construction*	11464/- -	3381/- -
ACK vector construction*	- -/598	- -/338
Independence check$^{\divideontimes}$	401/407	286/273
H_tests$^{\divideontimes}$	435/- -	267/- -
Rank of H pkts in $M^i_{rx} \cup M^i_{tx}{}^{\divideontimes}$	314/- -	269/- -
Rank of $M^i_{ox}{}^{\divideontimes}$	- -/345	- -/257
Rank of $M^i_{op}{}^{\divideontimes}$	- -/362	- -/264

6 IMPLEMENTATION AND TESTBED EVALUATIONS

Since the NS2 network simulator cannot reflect the computational complexity, in order to verify the performance of C^{3}ACK in a real wireless network, we build up a testbed of wireless mesh networks, named CSU-Mesh, which is described as follows.

6.1 Testbed description

Our CSU-Mesh consists of 10 wireless mesh routers deployed on the second floor of the Computer Building at Central South University, as shown in Fig. 11. Each router in CSU-Mesh is a DELL PC equipped with a wireless network card (Atheros 5212) attached to an omni-directional antenna and operating in 802.11b ad hoc mode. Routers run Ubuntu 9.04 (Linux kernel version 2.6.28) and the Click toolkit [18]. Wireless cards are enabled by the open-source Madwifi driver. CSU-Mesh executes OR protocols as user daemon programs. All coded OR implementations in CSU-Mesh are based on MORE's implementation [2], which implements MORE at layer-2.5, called OR layer (ORL) in this work, as a shim between the IP and the MAC layer.

6.2 Implementation details

We implement three batch-based coded OR protocols, MORE, CCACK, which is a typical orthogonal-vector based coded OR, and C^{3}ACK, which imbeds our coded feedback scheme into MORE. All three OR protocols implemented in CSU-Mesh handle only synthetic traffic, i.e., data packets are generated within Click, similarly

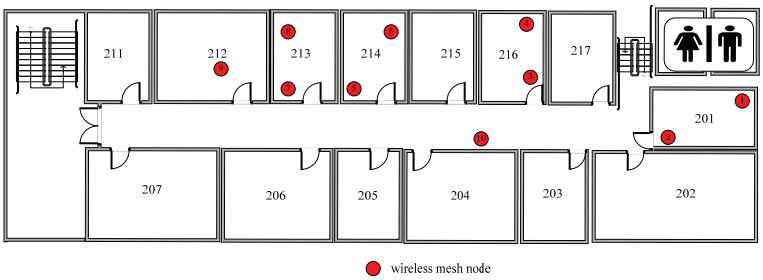

Figure 11: CSU-Mesh testbed deployment.

MAC Header	MORE/CCACK/C^{3}ACK Header	IP Header	Data	MAC Tail

Figure 12: MORE, CCACK and C^{3}ACK header.

to MORE's implementation [7]. The ORL header of MORE , CCACK or C^{3}ACK is added between the IP header and the MAC header, as shown in Fig. 12. The implementation of the packet format, the credit counter, the pre-code packet, and the end-to-end ACK are based on MORE's implementation. The implementation of the pre-code packet for C^{3}ACK is slightly different, as C^{3}ACK needs to handle not only the innovative coded packets in the forward coding traffic, but also the innovative backward coding vectors in the backward coding traffic. To keep the backward coding vectors up-to-date, the pre-code backward coding vectors need to be updated by multiplying the newly received and overheard coding vectors from the forward and backward coding traffics, respectively.

In MORE's implementation, ACK reliability is achieved through the shortest ETX path and the adoption of 802.11 unicast mode, which provides a reliability mechanism through acknowledgments and retransmissions. Unfortunately, there is an upper limit to the number of times a packet can be retransmitted at the MAC layer. For the Atheros wireless cards used in CSU-Mesh, the retransmission upper-bound is 11, which may not always be sufficient to deliver the packet to the next hop in our experiments, especially under heavy traffic. Since Atheros wireless cards do not allow changing this upper-bound through iwconfig, we implemented an additional ACK-retransmission scheme at the ORL. Since we have no control over a packet once it leaves the ORL, we have to guarantee that an ACK packet will never be dropped if an Atheros wireless card's queue is full of data packets.

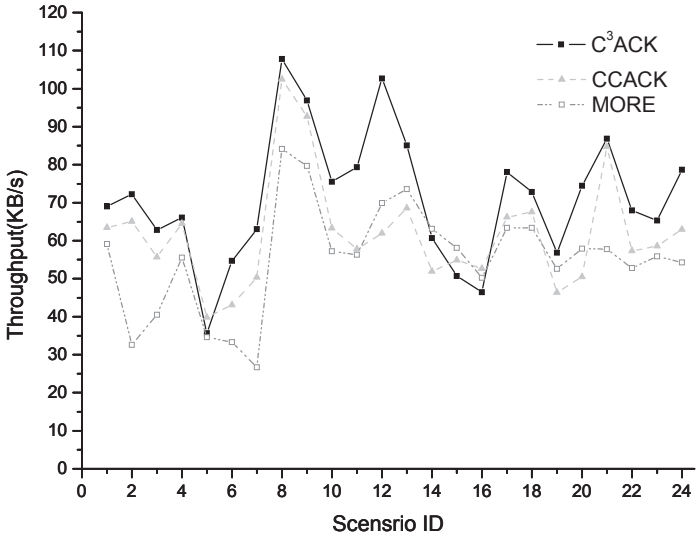

Figure 13: Throughput gain of C^{3}ACK over MORE and CCACK in CSU-Mesh.

6.3 Experimental setup

In all the experiments, the batch size is set to 32 packets, the size of the Galois field used to generate coding coefficients is set to 2^8, the bit rate is set to 2 Mb/s, and the transmission power is set to 18 dBm. RTS/CTS is disabled for unicast frames, as most operational networks are. With these settings, the length of the shortest ETX paths between different nodes is 1-4 hops in length, and the loss rates of the links vary from 0% to 86.9%, with an average value of 28.5%. We experimented with 24 single-flow scenarios (i.e., randomly selected source-destination pairs). For each scenario, the ETX module is executed first to collect the pairwise loss rates and calculate the corresponding ETX metrics accordingly. Then the three coded OR protocols, MORE, CCACK and C^{3}ACK, are executed in sequence. The source sends a 10-MB file consisting of 1500-B packets to the destination, using these three protocols. For each topology, the experiments repeat five times.

6.4 Experimental results

The experiments mainly evaluate the throughput gain of C^{3}ACK over MORE and CCACK. Fig. 13 plots the expected throughputs of MORE, CCACK and C^{3}ACK in each topology. We can observe that C^{3}ACK outperforms both MORE and CCACK in most scenarios. Specificially, the average throughputs of MORE, CCACK and C^{3}ACK of all 24 topologies are 55.529KBps, 61.775KBps and 71.233KBp, re-

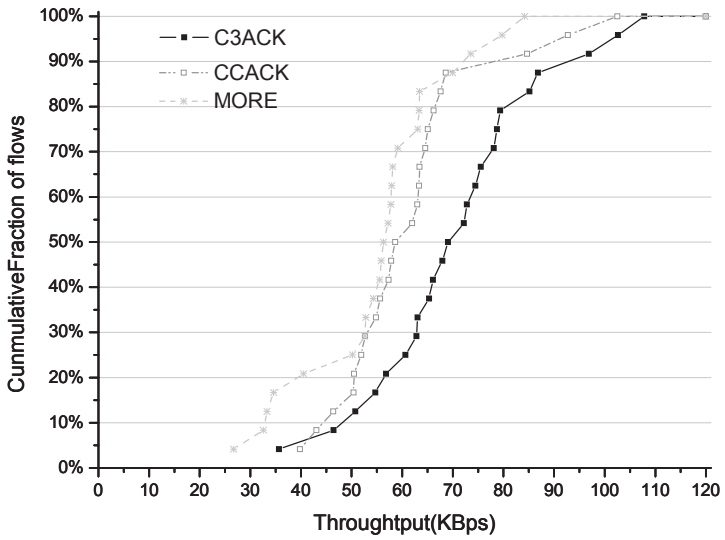

Figure 14: Throughput gain CDF of C^{3}ACK over MORE and CCACK in CSU-Mesh.

spectively. Fig. 14 plots the CDF of the throughput, which also shows that C^{3}ACK outperforms both CCACK and MORE. For example, with MORE, CCACK, and C^{3}ACK, about 30%, 50%, and 75% of topologies have throughput greater than 60KBps, respectively.

Fig. 15 plots the relative throughput improvement of C^{3}ACK over MORE and CCACK in all 24 topologies. Most throughput improvements in 24 topologies of C^{3}ACK over CCACK are at about 15%. However, in three topologies, the throughput improvements are negative. The reason is that these three topologies are all of the single path scenario, with a few hops from the source to the destination. The false-negative problem does not exist in the above simple scenario, and the false-positive problem is not significant, with a limited number of hops. Moreover, as an orthogonal vector contains more information than an ordinary coding vector, CCACK will outperform C^{3}ACK in the single path scenarios.

The relative throughput improvements of C^{3}ACK over MORE in most topologies are about 30%. Similarly, 3 out of 24 topologies show negative improvement, as the three topologies all have single paths with a few hops, which means fast end-to-end ACK for MORE. This, in turn, greatly reduces the redundant-packet transmissions. As for C^{3}ACK, its overhead in the packet header is higher than that of MORE, because it embeds an additional coding vector in each coded packet. Therefore, the performance of MORE is slightly better in these simple topologies.

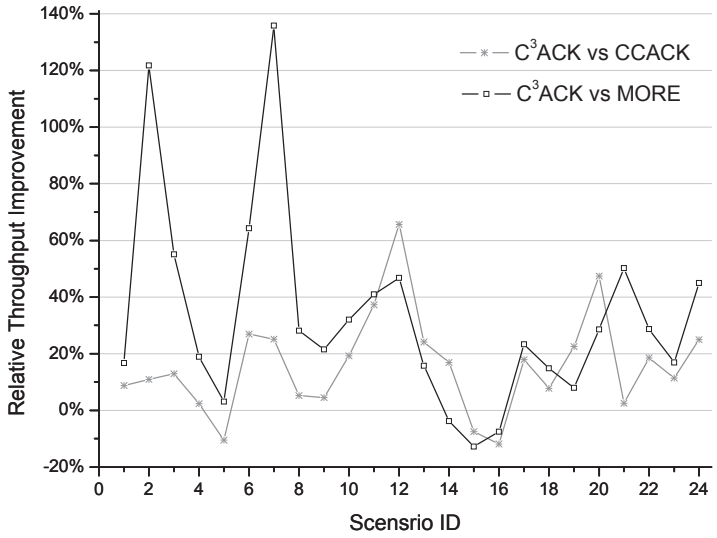

Figure 15: Relative throughput gain of C^{3}ACK over MORE and CCACK in CSU-Mesh.

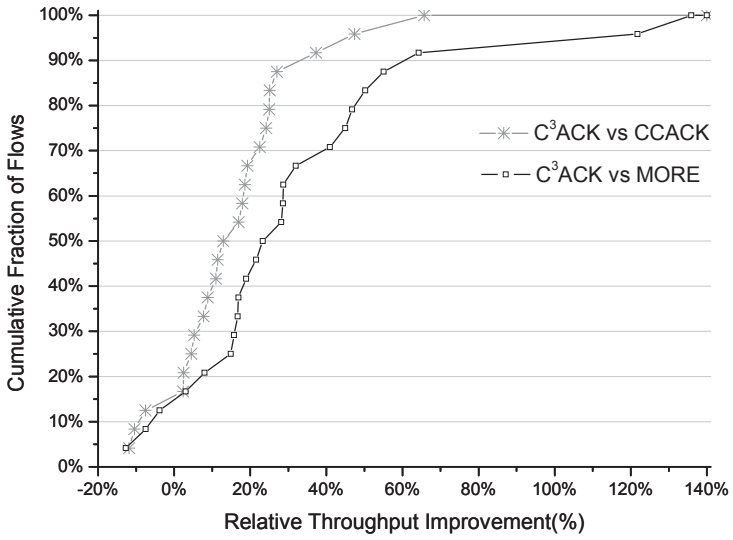

Figure 16: Relative throughput gain CDF of C^{3}ACK over MORE and CCACK in CSU-Mesh.

Fig. 16 plots the CDF of the relative throughput. Out of all 24 topologies, C^{3}ACK outperforms CCACK in 21 topologies, and also outperforms MORE in 21 topologies. The average gain of C^{3}ACK over CCACK and MORE are 16% and 33%, respectively.

7 Related Works

MORE [7] is the first coded OR protocol, which adopts the transmission credit to control the relative transmission frequencies, and thus, reduce the redundant packet transmission. As the transmission credit is calculated according to the ETX metic, which expresses the expected behavior, it cannot efficiently reduce redundant packet transmission due to the randomness of wireless channel. Numerous works [12, 14, 22, 23] have been proposed to deal with the redundant-packet transmission problem in order to enhance the throughput. CodeOR [22] proposed to transmit multiple batches in a pipeline fashion, so as to reduce the end-to-end ACK delay. However, CodeOR also faces the same fundamental difficulty as MORE, since it still adopts the batch-based NC.

SlideOR [23] extends CodeOR to adopt packet as a pipeline unit instead of batch, through combining online NC, called stream-based coding in this work, with TCP Vegas [5]. However, SlideOR only supports end-to-end stream-based coding. SlideOR has been further extended [8, 27]. Chen et al. [8] improved SlideOR by restricting the number of injected innovative packets at the source node. Xia and Chen [27] extended SlideOR by adopting a variable length network coding scheme. However, both of the mentioned extensions of SlideOR still adopt end-to-end stream-based coding. To the best of our knowledge, none of existing coded OR protocols have implemented hop-by-hop stream-based coding.

Inspired by the observation that the error probability of symbols is much lower than that of packets on a wireless link, MIXIT [15] extends MORE by operating NC on symbols rather than on packets. With such a simple modification, MIXIT can utilize correct symbols in a corrupted packet, and therefore attains higher throughput than MORE. The improvement achieved by the symbol-level coding is complementary to the enhancement made through our approach. We will incorporate the symbol-level coding in our future work.

The transmission-credit based redundancy reduction method heavily depends on the accuracy and freshness of the loss rate measurements, which are obtained through periodic probing, and are propagated from all nodes to the source. Apparently, the higher the probing frequency, the higher the accuracy, which, however, also imposes higher overhead. To reduce this overhead, MORE collects the loss rates and calculates the credits only in the beginning of each experiment. How-

ever, studies [3,28] have shown that, although link metrics remain relatively stable for long intervals in a quiet network, they are very sensitive to background traffic. Hence, the transmission-credit based method cannot reflect the online channel and network status.

To reflect the online network status, numerous works propose hop-by-hop feedback schemes [16,19,20,26]. Among them, SOR [20] associates each packet with a packet sequence number (PSN), which combines ACKMap to achieve hop-by-hop feedback. However, coded packet is randomly mixed with many packets, so that the linear relationship of coded packets is hard to infer through PSN. Moreover, the false-negative problem exists in SOR, which results in unnecessary packet transmissions. Orthogonal vector [16, 19, 26] has been utilized as a compressed representation of the knowledge space. However, because an orthogonal vector is a lossy compression, it incurs the false-positive problem, which occurs when a node receives a feedback vector orthogonal to its knowledge space, denoting the coverage of its knowledge space; however, the orthogonality is actually a coincidence. Although the false-positive problem can be mitigated by reducing the probability of coincidental orthogonality [19], it is at the expense of exacerbating the collective-space problem.

8 Conclusion and Future Works

This work first identifies the relevance between forward and backward coding traffics, through both theoretical analysis and simulation evaluation. Based on these oblivious characteristics, a simple but efficient code-pruning scheme, called C^3ACK, is proposed to reduce redundant-coded-packet transmissions. The effectiveness of C^3ACK is illustrated through both theoretical proof and experiment evaluation. Theoretically, C^3ACK is proved to be false-positive free and free from the collective-space problem, which do not hold for the existing orthogonal-vector-based coded feedback schemes. Both testbed and simulation experiments have been conducted to verify that C^3ACK significantly outperforms existing code-pruning schemes in the batch-based coded OR model. C^3ACK is further extended to the stream-based coded OR model, and the designed stream-based OR, called StreamOS, is the first hop-by-hop stream-based OR protocol, which is superior to the existing end-to-end stream-based OR protocols through simulation study.

This work focuses on intra-flow NC in unicast scenarios. In the future, we would like to extend our work to include multicast, broadcast, inter-flow network coding, and XOR coding. We also plan to integrate link and path correlation to further reduce redundant-coded-packet transmissions.

Acknowledgment

This work was partially supported by Project No 61232001, No 61173169, and No 60903222 supported by NSFC.

References

[1] Network simulator -ns-2. http://www.isi.edu/nsnam/ns/.

[2] More source code. http://people.csail. mit.edu/szym/more, 2009.

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

[4] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In *Proceedings of ACM SIGCOMM'05*, pages 133–144, 2005.

[5] L.S. Brakmo and L.L. Peterson. TCP vegas: end to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[6] S. Chachulski. Trading structure for randomness in wireless opportunistic routing. Master's thesis, MIT, 2007.

[7] S. Chachulski, M. Jennings, Sachin K., and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of ACM SIGCOMM'07*, pages 169 – 180, 2007.

[8] C. Chen, C. Dong, F. Wu, H. Wang, L. Peng, and J. Nie. Improving unsegmented network coding for opportunistic routing in wireless mesh network. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1847 –1852, 2012.

[9] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MOBICOM'03*, 2003.

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

[11] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Valuable detours: least-cost anypath routing. *IEEE/ACM Transactions on Networking*, 19(2):333–346, 2011.

[12] Euhanna Ghadimi, Olaf Landsiedel, Pablo Soldati, Simon Duquennoy, and Mikael Johansson. Opportunistic routing in low duty-cycled wireless sensor networks. *ACM Transactions on Sensor Networks*, 10(4), 2014.

[13] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Proc. of IEEE INFOCOM*, 2003.

[14] W. Hu, J. Xie, and Z. Zhang. Practical opportunistic routing in high-speed multi-rate wireless mesh networks. In *Proc.of 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'13)*, 2013.

[15] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level network coding for wireless mesh networks. In *Proc. of ACM SIGCOMM*, 2008.

[16] A. Khreishah, I. M. Khalil, and J. Wu. Distributed network coding based opportunistic routing for multicast. In *Proc.of 13th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'12)*, 2012.

[17] A. Khreishah, I. M. Khalil, and J. Wu. Universal opportunistic routing scheme using network coding. In *Proc. of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2012.

[18] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[19] D. Koutsonikolas, C.-C. Wang, and Y. C. Hu. Efficient network coding based opportunistic routing through cumulative coded acknowledgments. *IEEE/ACM Transactions on Networking*, 19(5):1368 – 1381, 2011.

[20] Patrick P. C. Lee, V. Misra, and D. Rubenstein. On the robustness of wireless opportunistic routing toward inaccurate link-level measurements. In *the Second International Conference on Communication Systems and Networks (COMSNETS)*, 2010.

[21] N. Letor, P. De Cleyn, and C. Blondia. Enabling cross layer design: Adding the madwifi extensions to nsclick. In *Proceedings of the First International Workshop on Network Simulation Tools*, 2007.

[22] Y. Lin, B. Li, and B. Liang. CodeOR: opportunistic routing in wireless mesh networks with segmented network coding. In *Proceedings of IEEE ICNP'08*, pages 13–22, 2008.

[23] Y. Lin, B. Liang, and B. Li. SlideOR: oline opportunistic network coding in wireless mesh networks. In *Proc. of IEEE INFOCOM'10 (MiniConference)*, 2010.

[24] M. Lu and J. Wu. Opportunistic routing algebra and its applications. In *Proceedings of IEEE INFOCOM*, 2009.

[25] M. Neufeld, G. Schelle, and D. Grunwald. Nsclick user manual. Technical report, University of Colorado, 2003.

[26] J. Wu and I. Khalil. Universal network coding-based opportunistic routing for unicast. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints):1, 2014.

[27] Z. Xia and Z. Chen. The research of variable length network coding based on opportunistic routing in streaming meida WMN. *Journal of Computational Information Systems*, 8(4):1723–1731, 2012.

[28] X. Zhang and B. Li. Dice: a game theoretic framework for wireless multipath network coding. In *Proc. of ACM MobiHoc*, 2008.