DCAR: Distributed Coding-Aware Routing in Wireless Networks

Jilin Le, John C.S. Lui
Department of Computer Science and Engineering
Chinese University of Hong Kong
{jlle, cslui}@cse.cuhk.edu.hk

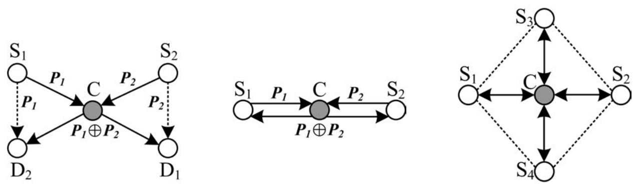
Dah Ming Chiu
Department of Information Engineering
Chinese University of Hong Kong
dmchiu@ie.cuhk.edu.hk

Abstract

The practical network coding system proposed in [1] has two fundamental limitations: 1) the coding opportunity is crucially dependent on the established routes; 2) the coding structure is limited within a two-hop region. To overcome these limitations, we propose DCAR, the first distributed coding-aware routing mechanism which combines (a) the discovery for available paths between a given source and destination, and (b) the detection for potential network coding opportunities. DCAR has the potential to find high throughput paths with coding opportunities while conventional routing fails to do so. In addition, DCAR can detect coding opportunities on the entire path, thus eliminating the "two-hop" coding limitation in [1]. We also propose a novel routing metric called "CRM" (Coding-aware Routing Metric) which facilitates the comparison between coding-possible and coding-impossible paths. We implement the DCAR system in NS-2 and conduct extensive evaluation, which shows that DCAR achieves 7% *to* 20% *throughput gain over the coding system in [1].*

1 Introduction

Network coding has been shown to be able to improve the throughput of wireless networks. In particular, authors of [1] proposed COPE, the first practical wireless network coding system. Fig. 1 shows the basic scenarios where COPE works. In Fig. 1(a), suppose node S_1 needs to send a packet P_1 to node D_1, relayed by node C; and S_2 needs to send a packet P_2 to node D_2 relayed by node C. The dashed arrows $S_1 \dashrightarrow D_2$ and $S_2 \dashrightarrow D_1$ indicate that D_2, D_1 are within the transmission range of S_1, S_2 respectively. Therefore, D_1, D_2 can perform "*opportunistic listening*": when S_1 (S_2) transmits P_1 (P_2) to node C, node D_2 (D_1) can overhear the transmission. When node C forwards the packets, it only needs to broadcast ($P_1 \oplus P_2$) to both D_1 and D_2 because they have already overheard the necessary packets for decoding via the opportunistic listening. In this case, using network coding can save one transmission at node C and enhance the bandwidth efficiency. Another potential coding scenario is illustrated in Fig.1(b), where *no* opportunistic listening is required. In this case, the source node S_1 (S_2) needs to send a packet P_1 (P_2) to its destination node S_2 (S_1). Since each source is also a destination node, it has the necessary packets for decoding upon receiving the encoded packet $P_1 \oplus P_2$. Last but not least, Fig.1(c) shows a *hybrid form* of coding which combines the former two cases, namely, some packets for decoding are obtained via opportunistic listening while other packets are obtained by the fact that the node is the source of that packet. In this case, node C can encode at most four packets from different flows and save $3/8$ of transmission times.

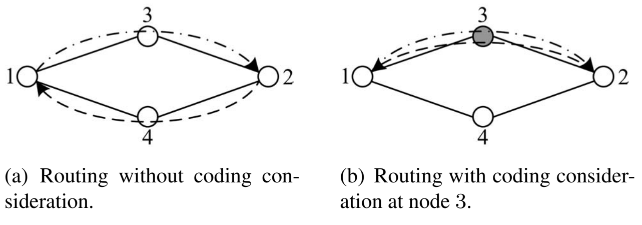


(a) Coding scenario with opportunistic listening.
(b) Coding scenario without opportunistic listening.
(c) Hybrid scenario.

Figure 1. Basic coding scenarios in COPE [1].

Essentially, COPE takes advantage of the "*broadcast nature*" of the wireless channel to perform *opportunistic listening* and *encoded packet broadcasting* so that the number of packet transmissions can be reduced. Nevertheless, there are two fundamental limitations in COPE, as we illustrate as follows.

The first limitation is that *coding opportunity is crucially dependent on traffic pattern*. In other word, network coding is possible only when different flows form certain coding structures. If one uses the shortest-path routing, or some recently proposed ETX-like routing [2], the potential for network coding may be significantly reduced. To illustrate, consider the example in Fig. 2 where there are two flows that need to be routed. Without consideration on potential coding opportunities, the disjoint paths shown in Fig. 2(a) may very likely be chosen. On the other hand, if we use a *coding-aware* routing decision as shown in Fig. 2(b), node 3 has the opportunity to encode packets. Clearly, in this example, coding-aware routing will result in higher end-to-end throughput for both flows.



3
1
2
4
(a) Routing without coding consideration.
(b) Routing with coding consideration at node 3.

Figure 2. Example: effect of routing decision on the potential coding opportunity.

Another limitation of COPE is that *it limits the entire coding scenario within a two-hop region*. Take Fig.1(a) as an example, it is assumed that the transmitters for opportunistic listening (i.e., node S_1, S_2) are the one-hop predecessors of node C, and that the intended decoders (i.e., node D_1, D_2) are the one-hop successors of node C. These assumptions may unnecessarily eliminate coding opportunities in a wireless network with flows that traverse longer than two hops. To illustrate, consider the scenario in Fig. 3 where two flows $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ and $5 \rightarrow 3 \rightarrow 6 \rightarrow 7$ intersect at node 3. Node 3 can encode packets from these two flows and broadcast the encoded packets to both node 4 and 6. Although node 6 cannot perform the necessary opportunistic listening for decoding, it can forward the encoded packet to node 7, where the opportunistic listening and decoding can take place. In general, both the opportunistic listening and decoding can be several hops away from the coding node (i.e., the node that encodes packets). If these generalized coding opportunities can be detected, we can further enhance the bandwidth efficiency and throughput.

The above limitations raise some challenging questions. For example, how to incorporate consideration on potential coding opportunities into the route selection? How can we look beyond two hops to discover more coding opportunities? How to evaluate and compare the performance of a coding-possible path and a coding-impossible path? To answer these questions, we revisit the system design of practical network coding system, and propose a novel wireless routing system: *Distributed Coding-Aware Routing* (DCAR).

The contributions of our work are:

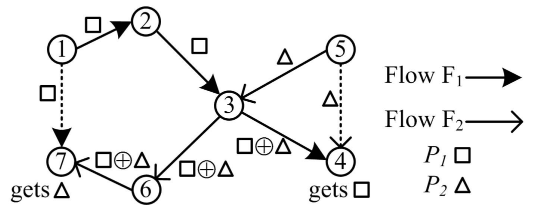


Figure 3. Example: the generalized coding scheme.

- We propose a distributed routing mechanism that can discover the available paths and potential coding opportunities concurrently.
- We formally define the generalized *coding conditions*, in which the practical network coding can occur. These conditions lead us to look beyond two hops and detect more coding opportunities.
- We propose a unified framework, which we called the "*coding-aware routing metric*" (CRM), to evaluate the performance of a path, may it be coding-possible or coding-impossible.
- We implement the DCAR routing system in NS-2 and carry out extensive evaluation showing the performance gain over COPE and conventional routing.

The outline of our paper is as follows. In Section 2, we describe the "Coding+Routing Discovery" which combines the detection processes of available paths and potential coding opportunities. The new discovery mechanism removes the "two-hop" limitation of COPE, and makes possible the coding-aware route selection. In Section 3, we introduce our coding-aware routing metric, which quantifies the potential benefit of "coding-possible" paths, and facilitates the comparison between "coding-possible" and "coding-impossible" paths. In Section 4 we present the simulation results under NS-2. Related work is given in Section 5 and finally, Section 6 concludes.

2 The "Coding+Routing" Discovery

The limitations of COPE, in particular the "coding-oblivious" route selection and the "two-hop" coding scenario, are mainly due to the "*separation*" between its coding discovery process and the routing discovery process. In COPE, each node initiates some active or passive detection for coding opportunities based on the *established routes*, therefore, routes in Fig. 2(a) may be chosen instead of the routes with coding opportunity in Fig. 2(b). On the other hand, because the coding detection is made only based on local information, the coding structure is inevitably limited within a region with short hops from the coding node. This observation leads us to a "coding+routing" combined solution to tackle such limitations.

2.1 Coding Conditions

Before presenting the algorithm, let us state the *necessary* and *sufficient* conditions in which coding can occur. To formally define this concept, we introduce the following notations. Let a denote a node, and let $N(a)$ denote the set of one-hop neighbors of node a. Let F be a flow[1] and we use $a \in F$ to denote that node a is along the flow F. Let $U(a, F)$ denote the set of all *upstream nodes* of node a in flow F, and let $D(a, F)$ denote the set of all *downstream nodes* of node a in flow F. For example, in Fig. 3, we have $U(3, F_1) = \{1, 2\}$, $U(3, F_2) = \{5\}$, $D(3, F_1) = \{4\}$ and $D(3, F_2) = \{6, 7\}$. Generally, when two flows F_1 and F_2 intersect at a node denoted by c, packets of these two flows can be encoded for transmission at node c *if and only if* the following conditions are met:

Coding Conditions for two flows F_1 and F_2 intersecting at node c
1. There exists $d_1 \in D(c, F_1)$, such that $d_1 \in N(s_2), s_2 \in U(c, F_2)$, **OR** $d_1 \in U(c, F_2)$. 2. There exists $d_2 \in D(c, F_2)$, such that $d_2 \in N(s_1), s_1 \in U(c, F_1)$, **OR** $d_2 \in U(c, F_1)$.

Lemma 1: *The above conditions are the necessary and sufficient conditions for any proper coding and decoding to occur*[2].
Proof: Due to lack of space, please refer to our technical report [9].

The importance of the above coding conditions is that each node can *individually* and *distributively* determine whether it can play the role of a coding node or not, if it has the following information:

- The **"path"** information: $U(c, F)$ and $D(c, F)$ for any flow F relayed by node c.
- The **"who-can-overhear"** information: $N(a)$ for each node $a \in U(c, F)$, for any flow F relayed by node c.

In what follows, we present a distributed algorithm to gather the above information and to realize coding and routing discovery concurrently.

2.2 Distributed "Coding+Routing" Discovery

Now we describe how to discover the available path(s) for a new flow initiated into the wireless network, and at the same time, detect the potential coding opportunities of the paths. The detection for coding opportunity is based on the conditions described in Section 2.1. Note that when we detect a path with coding opportunity (or what we call a *coding-possible path*), we do not impose the requirement that the new flow has to take this path as its routing outcome, instead, we have another module which will evaluate the benefit of each path and to make the final path selection. We will discuss this further in Section 3.

For each node a in a wireless network, it maintains a list of all its one-hop neighbors (i.e. $N(a)$) and the *packet loss probabilities* of all its *outgoing* links. These information can be collected by periodically sending probing messages as in [2], or by estimating the loss probability based on previously transmitted traffic. We use $P(a, b)$ to denote the packet loss probability on the link $a \to b$ where $b \in N(a)$.

When a new flow arrives, it activates the *coding+routing discovery process* which has the following steps:
Step 1. The source node s initiates the route discovery by broadcasting the Route Request (RREQ) message. The RREQ contains the following information:

- One-hop neighbors of the source node, which have high overhearing probabilities, i.e. $\{a | a \in N(s), P(s, a) > threshold\}$ where $threshold$ is set to 0.8 in our NS-2 implementation.
- The path that it has traversed, as any source routing does.

Step 2. Upon receiving a RREQ, an intermediate node, say node c, first checks whether the RREQ has already traversed through itself. If so, node c discards the RREQ to prevent loop; otherwise node c performs the following:

- Temporally storing the RREQ, which contains the "who-can-overhear" information for the new path. In other words, node c stores the list of overhearing nodes that can perform "opportunistic listening" when the upstream nodes transmit.
- Updating the "who-can-overhear" information. Node c appends its high quality neighbors into the RREQ, such that the list gradually enlarge when the RREQ travels through the network.
- Re-broadcasting the updated RREQ to discover remaining path to the destination node.

Step 3. When a RREQ reaches the destination node, the destination replies with Route Reply (RREP) message using the reverse path back to the source node. The RREP is a unicast message that contains the "path" information.
Step 4. Upon receiving a RREP, an intermediate node, say node c, compares the upstream path contained in the RREP with the paths in its temporally stored RREQs. If there is a match, then it has obtained both the "*path*" and "*who-can-overhear*" information for the new path. Each node also

[1]In the remaining of this paper, unless we state otherwise, we refer to "paths" and "flows" interchangeably.
[2]Note that we assume perfect channel condition and scheduling for opportunistic hearing here. We will deal with imperfect overhearing in later sections.

maintains the "path" and "who-can-overhear" information for all the existing flows relayed by itself. Given these information, node c can check whether the new flow can be encoded with some existing flow(s) using the coding conditions stated in Section 2.1. If there is coding opportunity, node c marks its link as "coding-possible" in the RREP.

Step 5. When the RREP(s) return to the source node, a routing decision is made based on the potential coding opportunities and the benefit of each available paths (which we will present in Section 3), and the source node begins to send data packets on the selected path.

Step 6. When the first data packet reaches an intermediate node, say node c, it stores the "who-can-overhear" and "path" information for the selected path, while discarding other temporally stored information.

2.3 An Illustrative Example

We use the simple wireless network in Fig. 3 to illustrate how the "coding+routing" discovery works. Suppose the flow $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ (i.e. flow F_1) is an existing flow, and Fig. 4(a) shows the information for the existing flow F_1 stored at node 3. Now we wants to find a path for the new flow $5 \rightarrow 7$, the discovery process goes as follows:

Flow F_1
Path: 1→2→3→4
Who-can-overhear: 7

(a) Information stored at node 3 for the existing flow.

Temporally stored RREQ
Path: 5→3→···→7
Who-can-overhear: 4

(b) Information contained in the temporally stored RREQ at node 3.

Figure 4. An example of the data structures maintained at the coding node.

1. Node 5 initiates the discovery by sending RREQ, and adds its high quality neighbors $3, 4$ into the RREQ.
2. When node 3 receives the RREQ, it temporally stores the "who-can-overhear" information (i.e. node 4 can overhear the transmission of the upstream nodes) and the "upper" path. The data structure is shown in Fig. 4(b): the "upper" path is $5 \rightarrow 3$ and the overhearing node is 4. Node 3 then updates the overhearing information (i.e. adding node $2, 6$ into the list) before rebroadcasting the RREQ.
3. Suppose one RREQ reaches node 7 through the path $5 \rightarrow 3 \rightarrow 6 \rightarrow 7$, node 7 replies with RREP, which contains the complete node list on the entire path.
4. When node 3 receives this RREP, it matches the path $5 \rightarrow 3 \rightarrow 6 \rightarrow 7$ with its temporally stored RREQ information as shown in Fig. 4(b), and discovers that the new path can be encoded with the existing flow $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, thus marking the link $3 \rightarrow 6$ as "coding-possible" in the RREP.
5. The RREP finally returns to the source node 5 with information of potential coding opportunities.

3 Designing Coding-Aware Routing Metric

In the previous section, we presented the discovery process for both available paths and their potential coding opportunities. The remaining question is *how to choose a good path among these available choices*. One important point to note is that we *should not always choose a path with coding opportunity* because a coding-possible path may not provide the best possible performance: it may already be congested, or it may take too many hops to reach the destination and consume more network resource. In other words, there may exist some "coding-impossible" paths with higher throughput or lower delay. The essential issue in path selection is to design a good *routing metric* which can be used to quantify the merit between *coding-possible* and *coding-impossible* paths. In the following, we present CRM, the Coding-aware Routing Metric. Please refer to our technical report [9] for a general review of existing routing metrics.

3.1 Necessary Properties of CRM

First let us consider what routing metric is suitable for the coding-aware route selection. Suppose there are some existing flows in the wireless network, and we want to find a path for a new flow. Some of the potential paths may have coding opportunities while some may not. For proper path evaluation, we impose the following two properties onto the coding-aware routing metric:

- The metric should take into account the benefit of the *coding-possible* paths. If a new path can be encoded with some existing flows, it can "*free-ride*" on the bandwidth used by the existing flows.
- The metric should be *universal* for both coding-possible paths and coding-impossible paths. In other words, the interpretation of "*free-ride*" benefit for coding-possible paths should be *transferable* to the performance measure for coding-impossible paths.

Based on the first property, a coding-aware routing metric must take into account the existing traffic load information in making the evaluation because the saving in the "free-ride" bandwidth is crucially dependent on the existing traffic. The technical difficulty of this requirement is that typically one does not know the actual throughput of the existing flows. For the second property, we also face another technical challenge of how to compare the performance measure of *coding-possible* paths versus *coding-impossible* paths. In the following, we show how to tackle these two technical issues.

3.2 Interpreting the "Free-Ride" Benefit

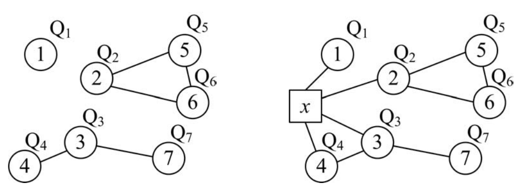
Let us first work out a proper way to interpret the "free-ride" benefit. To achieve this, let us consider the following simple example to gain the intuition: suppose a node has an on-going flow, and it finds out that a new flow traversing through it has a coding opportunity with the on-going flow, then what is the potential benefit on this the coding-possible link? If the current bandwidth consumption of the on-going flow at the node is B_1, then in the best case, the node only needs to use B_1 bandwidth as long as the throughput on the new flow (denoted by B_2) does not exceed B_1, because all the new traffic can "free-ride" on (or be encoded with) the existing traffic. If $B_2 > B_1$, then the node needs to consume B_2 bandwidth to deliver all the traffic.

The above approach of interpreting the "free-ride" benefit is straightforward, however, it inevitably needs the actual throughput of the on-going traffic, which, as we discussed, is difficult to obtain in practice. Now let us consider another approach: *examining the buffered queue length of the node*. Intuitively, the average queue length can be an indicator for how busy the node is (and therefore how much free-time is left) and the delay for incoming traffic. Suppose the queue length of the node is Q_1 before the new flow is initiated. Without network coding, there are Q_1 packets ahead of those packets for the new flow. However, when coding is used, the new flow actually see *zero* packets ahead in the queue, because its packets can always be encoded with the existing ones! In short, if using average queue length as an indicator, the actual calculation of the queue length need to be "*modified*" in case there is a coding opportunity. In what follows, we present how to modify the queue length in the general case.

3.3 Modified Queue Length

For a considered node, we modify the calculation of its queue length according to the coding relationships. For example, if two flows with average queue length Q_1 and Q_2 can be encoded together, then their total contribution in the modified queue length should be $\max\{Q_1, Q_2\}$. To assist the analysis for the general case, we first introduce "*coding graph*" as an analytical tool to represent the coding relationships.

1) Coding Graph. A coding graph is an undirected graph, with each vertex representing a flow relayed by the considered node. For the existing flows, each vertices i is associated with a value Q_i, which is equal to its average number of packets in the queue. An edge between two vertices indicates that these two flows satisfy the coding condition. Consequently, if a subgraph of the coding graph is a *complete* graph, then the vertices (i.e. flows) in this subgraph can be encoded all together.



(a) The *stable* case. (b) The *dynamic* case.

Figure 5. Examples of coding graph for the considered node c.

Fig. 5 shows two examples of the coding graph for a considered node, in the *stable* case and *dynamic* case respectively. In stable case (Fig. 5(a)), there is no new paths (or flows) to be added. In dynamic case (Fig. 5(b)), there is a new path to be examined, which we represent by vertex x.

2) Modified Queue Length in Stable Case. First of all, if there is no edges in the coding graph (i.e. no coding opportunity), then the modified queue length is simply the sum of average queue lengths for all flows. If there exists some edges, we know that for a *complete* subgraph (i.e. a *clique*) of the coding graph, their total contribution in the modified queue length should be the maximal queue length among them. The larger the clique is, the more we can reduce in the modified queue length. Upon each transmission, finding the *maximum clique* in the coding graph can help us encode the maximum number of packets, however, the *maximum clique problem* is NP-complete [7]. To reduce computational cost, we calculate the modified queue length based on a *"round-robin"* encoding scheme. Let the considered node be node c, we use $MQ_s(c)$ to denote the modified queue length of node c in the stable case. The calculation steps are shown in Table 1.

In Table 1, Step 3-8 shows the round-robin coding scheme: randomly pick one flow and encode with as many flows as possible. Because the maximum number of flows that can be encoded with a given flow is bounded by a small number [8], the computational cost in each iteration is insignificant. Using Figure 5(a) as a example, suppose we choose vertex 2 in the first iteration, and choose vertex 3 in the second iteration, the resulting $MQ_s(c)$ will be $\max\{Q_2, Q_5, Q_6\} + \max\{Q_3, Q_4\} + Q_7 + Q_1$ or $\max\{Q_2, Q_5, Q_6\} + \max\{Q_3, Q_7\} + Q_4 + Q_1$.

3) Modified Queue Length in the Dynamic Case.

```
1. Remove all vertices with zero queue length and
   their corresponding edges.
2. Initialize $MQ_s(c) = 0$, vertex set $V = \emptyset$.
3. Randomly pick vertex $i$ among remaining vertices.
4. Find out the maximal complete subgraph that contains $i$.
5. Add vertices of the subgraph into $V$.
6. Add $\max_{i \in V}\{n_i\}$ into $MQ_s(c)$.
7. Remove all vertices in $V$ and their edges.
8. Reset $V = \emptyset$.
9. Repeat Step 3-8, until all vertices are removed.
```

Table 1. Calculation of modified queue length in the *stable* case.

In this case, we examine the the average queue length "seen" by packets from the new flow x. We use $MQ_d(c)$ to denote the modified queue length of node c in the dynamic case. The calculation goes as follows:

```
1. Initialize $MQ_d(c) = 0$.
2. Remove vertex $x$ and all the vertices that are adjacent
   to $x$, and their corresponding edges.
3. For the rest of the graph, go through the same calculation
   as in the stable case.
```

Table 2. Calculation of Modified Queue Length in *Dynamic* Case.

Compared to the stable case, the difference is that we treat all flows that can be encoded with the new flow as "non-existing", because an incoming packet of the new flow can "override" on the transmission for any of these existing flows. For example, in Figure 5(b), we first remove vertex $x, 1, 2, 3, 4$ and all their edges from the graph, and the resulting $MQ_d(c)$ is $\max\{Q_5, Q_6\} + Q_7$.

3.4 MIQ: Modified Interference Queue Length

The modified queue length of a node, however, is not sufficient to estimate its available bandwidth in the wireless network, because a node with very short queue length can still be congested if its interfering nodes have a lot of packets to send. Let $I(c)$ denote the set of node c's interfering nodes. We define $MIQ(c)$, the "Modified Interference Queue" length, as

$$\begin{aligned} MIQ_s(c) &= MQ_s(c) + \sum_{i \in I(c)} MQ_s(i) \\ MIQ_d(c) &= MQ_d(c) + \sum_{i \in I(c)} MQ_s(i) \end{aligned}$$

where MIQ_s, MIQ_d are the MIQ values in stable and dynamic cases respectively. For evaluating a new path, we should use $MIQ_d(c)$.

Essentially, we model the considered node c and its interfering nodes as a big queueing system, with the wireless channel around them as a service center which needs to serve packets for node c and all its interfering nodes. The MIQ value indicates how busy the channel is and the delay for an incoming packet. Furthermore, the MIQ value for a node represents its *private* view of the channel status, which may vary significantly from node to node.

3.5 CRM: Coding-aware Routing Metric

For each link l on a path L, let $MIQ_d(l)$ be the dynamic MIQ value of the transmitter on l, and let P_l denote the packet loss probability on l. The CRM metric of link l is calculated as:

$$CRM_l = \frac{1 + MIQ_d(l)}{1 - P_l}$$

Intuitively, CRM_l corresponds to *expected number of transmissions* for successfully transmitting the existing packets as well as one incoming packet for the new flow[3]. We use the dynamic MIQ value on link l because the path to be evaluated is for the new flow. For the metric of the entire path L, we define the CRM values as $CRM_L = \sum_{l \in L} CRM_l$.

CRM incorporates topology, traffic load and interference information together in a unified manner. By using the "*modified interference queue length* (MIQ)" as the indicator for channel status, CRM does not require the wireless card to report "channel busy time" to higher layers, and also does not need to be aware of the actual throughput of existing flows. More importantly, CRM provides a *unified* measure for both coding-possible and coding-impossible paths.

The message overhead of CRM lies in that it requires neighboring nodes to communicate "modified queue length" with each other to compute the MIQ value, and it also needs the packet loss probability on each link. In our NS-2 implementation, we let each node broadcast HELLO message periodically within its one-hop neighbors, and piggyback its modified queue length into the HELLO message. Therefore, the only message overhead is the HELLO exchange, which is already used by many wireless routing protocols like AODV. For implementation details of the DCAR system, please refer to our technical report [9].

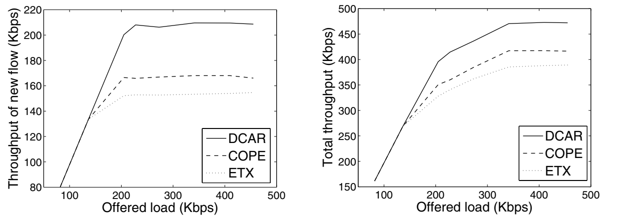
[3]Note that this is an approximation, because the packet loss probabilities for different outgoing links may vary.

4 Simulation Results

We now present the simulation results. We implement the DCAR and COPE [1] system under NS-2. There are two main differences between DCAR and COPE: 1) COPE uses ETX [2] as the routing metric while DCAR uses CRM; 2) COPE limits the coding structure within two hops while DCAR eliminate such limitation. The goals of our simulation are to evaluate the effectiveness of CRM in finding high-throughput path with coding opportunities, and to quantify the benefit of breaking the "two-hop" limitation. Throughout the experiments, we use 802.11b and UDP traffic sources. The transmission range is set to 250 and the carrier sensing range is set to 550. When a new flow is to be added, we allow 3 seconds for the "coding+routing" discovery to find the available paths. Once a flow decides on one path, it uses the path towards the end of the simulation time.

4.1 Results from Illustrative Scenarios

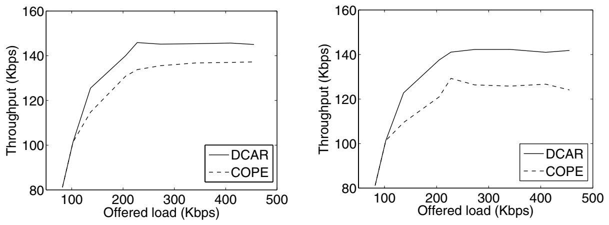
Experiment 1. Bidirectional flows (Fig. 2): We first study the simple scenario shown in Fig. 2. We start a flow from node 1 to node 2, and then add the new flow 2 to 1. The flows are given the same traffic load. We vary the offered load and plot the resulting end-to-end throughput in Fig. 6. Three types of system are considered: DCAR, COPE, and ETX routing without network coding. We observe that DCAR always chooses the intersecting paths for both flows, while the routes chosen by COPE (and ETX) vary between the disjoint and intersecting patterns. The throughput gain of DCAR tends to be more significant when the offered load increase, resulting in a 20% gain for the new flow and a 12% gain for the total throughput over COPE.



(a) Throughput of the new flow from 2 to 1. (b) Total throughput.

Figure 6. Results from the topology in Fig. 2.

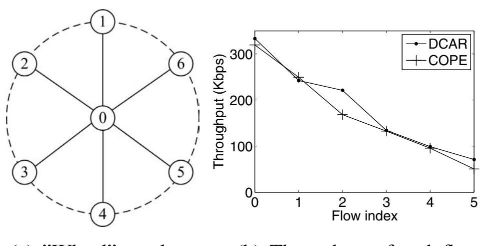
Experiment 2. Generalized coding (Fig. 3): We compare the performance of DCAR and COPE using the topology shown in Fig. 3. In this case, the routes chosen by DCAR and COPE are the same, however, COPE can not detect the potential coding opportunity at node 3, because it misses the fact that node 7 can perform opportunistic listening. For each offered load, we repeat the experiment 10 times, varying the arrival orders of flow $5 \rightarrow 7$ and flow $1 \rightarrow 4$. The resulting average throughput of both flows is plotted in Fig. 7. The throughput gain by the generalized coding scheme ranges from 7% to 16% in this scenario.



(a) Flow from 5 to 7. (b) Flow from 1 to 4.

Figure 7. Results from the topology in Fig. 3.

Experiment 3. "Wheel" topology: It is interesting to study how DCAR works in a "wheel" topology as shown in Fig. 8(a), where a central node (0) is surrounded by six nodes (1 to 6) evenly distributed along the cycle. Each node along the cycle can reach everyone else except for the node on the opposite end of the diameter (e.g. node 1 can reach everyone else except for 4, vice versa). We let each node along the cycle starts a flow to the node at the opposite end of the diameter. There are plenty of coding opportunities in this scenario, not only at the central node 0 but also at other nodes. For example, if two flows $1 \rightarrow 4$ and $2 \rightarrow 5$ use the paths $1-3-4$ and $2-3-5$ respectively, then node 3 can also encode packets. In the experiment, we vary the traffic load and arrival order of each flow, and plot the average throughput in Fig. 8(b). We can see that DCAR typically offers higher throughput than COPE, but the gain is not significant. The underlying reason is that even if the paths are randomly chosen between available shortest paths, there are still many coding opportunities at the surrounding nodes as we discussed.

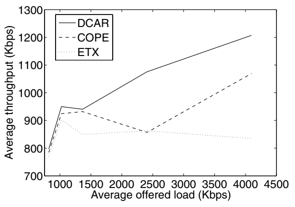


(a) "Wheel" topology. (b) Throughput of each flow.

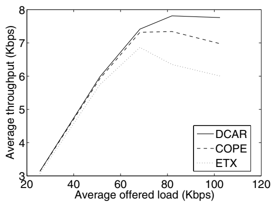
Figure 8. Results from a "wheel" topology.

4.2 Results from Ad Hoc Networks

Experiment 4. Grid topology: We construct a 4 by 4 grid topology where each node can only reach its northern, southern, eastern and western nodes. The experiment is of 10 rounds. At each round, we randomly add 5 flows into the network and repeat the process for 3 times. We plot the average end-to-end throughput achieved by DCAR, COPE and ETX respectively in Fig. 9(a). Not surprisingly, the gain by DCAR tends to be larger with higher offered load. In Fig. 9(b), we make the network even more congested by adding 10 flows in each round, the results also reveal the potential of offering higher throughput by DCAR.



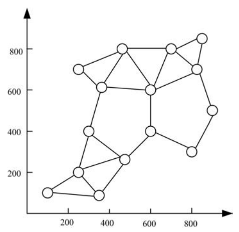
(a) Results by adding 5 flows.

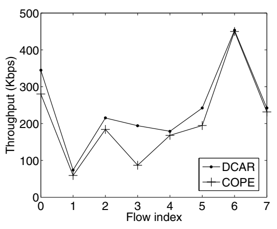


(b) Results by adding 10 flows.

Figure 9. Results from a grid topology.

Experiment 5. Random topology: Finally we compare DCAR and COPE in a 15-node random topology as shown in Fig. 10(a). The average node degree is 3.2. We randomly pick 8 flows and vary their arrival orders and loads in each round. The average throughput for each flow is plotted in Fig. 10(b). Because there is a rich set of coding opportunities and available paths, DCAR achieves substantial gains over COPE.


(a) 15-node random topology.



(b) Throughput of each flow.

Figure 10. Results from a 15-node random topology.

5 Related Work

The concept of network coding is first proposed in [3]. For wireless networks, authors of [1] propose COPE, the first practical XOR coding system and demonstrate the throughput gain via implementation and measurement. COPE uses ETX [2] as its routing function. Based on COPE, authors of [4, 5] introduce the concept of coding-aware routing and formulate the max flow LP with coding considerations, however, their work is a centralized approach and assumes perfect scheduling. Authors of [6] propose a complex optimization framework for adaptive coding and scheduling. Authors of [8] study limitations of COPE under practical physical layer and link-scheduling algorithms, propose the concept of coding-efficient link-scheduling for practical network coding. Compared with former works, DCAR is the first practical distributed coding-aware routing system, and adopts a more generalized coding scheme by eliminating the “two-hop” limitation in COPE.

6 Conclusion

We propose DCAR, the first distributed coding-aware routing system for wireless networks. DCAR incorporates potential coding opportunities into route selection using the “Coding+Routing Discovery” and “CRM” (Coding-aware Routing Metric). DCAR also adopts a more generalized coding scheme by eliminating the “two-hop” limitation in COPE [1]. Extensive evaluation under NS-2 reveals substantial throughput gain over COPE achieved by DCAR.

References

[1] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard and J. Crowcroft. XORs in the Air: Practical Wireless Network Coding. *Proceedings of ACM SIGCOMM 2006.*

[2] D. Couto, D. Aguayo, J. Bicket and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. *Proceedings of ACM MOBICOM 2003.*

[3] R. Ahlswede, N. Cai, S. Li and R. Yeung. Network Informaion Flow. *IEEE Trans. on Informaion Theory*, 46(4), pp. 1204-1216, July 2000.

[4] B. Ni, N. Santhapuri, Z. Zhong and S. Nelakuditi. Routing with Opportunistically Coded Exchanges in Wireless Mesh Networks. *Poster session of SECON 2006.*

[5] S. Sengupta, S. Rayanchu and S. Banerjee. An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing. *Proceedings of IEEE INFOCOM 2007.* 2007.

[6] P. Chaporkar and A. Proutiere. Adaptive Network Coding and Scheduling for Maximizing Throughput in Wireless Networks. *Proceedings of ACM MOBICOM 2007.*

[7] R.M. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations.* New York: Plenum, 85-103.

[8] J. Le, JCS Lui and DM Chiu. How Many Packets Can We Encode? - An Analysis of Practical Wireless Network Coding. *Proceedings of IEEE INFOCOM 2008.*

[9] J. Le, JCS Lui and DM Chiu. DCAR: Distributed Coding-Aware Routing in Wireless Networks. *CUHK CSE Technical Report 2008-01.* Available at `http://www.cse.cuhk.edu.hk/~jlle/icdcs-report.pdf`.