DCAR: Distributed Coding-Aware Routing in Wireless Networks

Jilin Le* John C.S. Lui* Dah-Ming Chiu+
*Computer Science & Engineering Department
+Information Engineering Department
The Chinese University of Hong Kong
Email: {jlle,cslui}@cse.cuhk.edu.hk, dmchiu@ie.cuhk.edu.hk

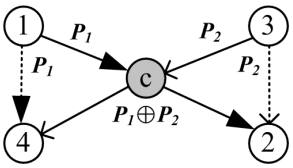
***Abstract*—Recently, there has been a growing interest of using network coding to improve the performance of wireless networks, for example, authors of [1] proposed the practical wireless network coding system called COPE, which demonstrated the throughput gain achieved by network coding. However, COPE has two fundamental limitations: (a) the coding opportunity is crucially dependent on the established routes; (b) the coding structure in COPE is limited within a two-hop region only. The aim of this paper is to overcome these limitations. In particular, we propose DCAR, the Distributed Coding-Aware Routing mechanism which enables (1) the discovery for available paths between a given source and destination, and (2) the detection for potential network coding opportunities over much wider network region. On interesting result is that DCAR has the capability to discover high throughput paths with coding opportunities while conventional wireless network routing protocols fail to do so. In addition, DCAR can detect coding opportunities on the entire path, thus eliminating the "two-hop" coding limitation in COPE. We also propose a novel routing metric called Coding-aware Routing Metric (CRM) which facilitates the performance comparison between "coding-possible" and "coding-impossible" paths. We implement the DCAR system in ns-2 and carry out extensive evaluation. We show that, when comparing to the coding mechanism in [1], DCAR can achieve much higher throughput gain.**

Keywords: network coding, wireless networks, routing.

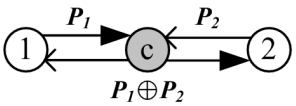
I. Introduction

In the past few years, network coding is becoming an emerging communication paradigm that can provide performance improvement in throughput and energy efficiency. Network coding was originally proposed for wired networks, and the throughput gain was illustrated by the well-known example of "butterfly" network [7]. Recently, there is a growing interest to apply network coding onto wireless networks since the broadcast nature of wireless channel makes network coding particularly advantageous in terms of bandwidth efficiency and enables opportunistic encoding and decoding.

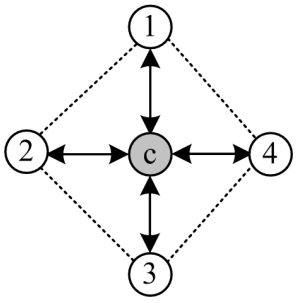
In [1], the authors proposed COPE, the first practical network coding system for multi-hop wireless networks. Figure 1 shows the basic scenarios of how COPE works. In Figure 1(a), there are five wireless nodes. Suppose node 1 wants to send a packet P_1 to node 2 and this packet needs to be relayed by node C; and node 3 wants to send another packet P_2 to node 4 wherein node C also needs to relay this packet. The dashed arrows $1 \dashrightarrow 4$ and $3 \dashrightarrow 2$ indicate that $4, 2$ are within the transmission ranges of $1, 3$ respectively. Under this scenario, nodes 4 and 2 can perform "*opportunistic overhearing*": when 1 (3) transmits P_1 (P_2) to node C, node 4 (2) can overhear the transmission. When node C forwards the packets, it only needs to broadcast one packet, $(P_1 \oplus P_2)$, to both 4 and 2. Since 4 and 2 have already overheard the necessary packets, they can carry out the decoding by performing $P_2 \oplus (P_1 \oplus P_2)$ or $P_1 \oplus (P_1 \oplus P_2)$ respectively, thereby obtaining the intended packet. In this case, it is easy to see that there is a reduction in bandwidth consumption because node C can use network coding to reduce one transmission.



(a) Coding scenario with opportunistic overhearing.



(b) Coding scenario without opportunistic overhearing.



(c) Hybrid scenario.

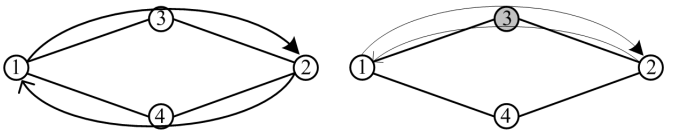
Fig. 1. Basic coding scenarios in COPE [1].

It is interesting to point out that network coding can also be used when there is no opportunistic overhearing, and this scenario is illustrated in Figure 1(b). In this case, the source node 1 (2) needs to send a packet P_1 (P_2) to its destination node 2 (1). Since each source is also a destination node, it has the necessary packets for decoding upon receiving the encoded packet $P_1 \oplus P_2$. Again, instead of four transmissions when network coding is not used, one only needs three transmissions and thereby reducing the bandwidth consumption. Last but not least, Figure 1(c) shows a *hybrid form* of coding which combines the former two cases, namely, some packets for decoding are obtained via opportunistic overhearing while

other packets are obtained by the fact that the node is the source of that packet. Under this scenario, instead of requiring eight packet transmissions, networking coding can reduce it to five transmissions: four for transmitting a packet to node C, and one for node C to encode four packets and to transmit the encoded packet.

In essence, COPE takes advantage of the "broadcast nature" of the wireless channel to perform "*opportunistic overhearing*" and "*encoded broadcast*", so that the number of necessary transmissions can be reduced. However, COPE has two fundamental limitations which we illustrate as follows. Let us elaborate on these further.

The first limitation is that whether network coding is possible (or we called the "*coding opportunity*") is crucially dependent on traffic pattern. In other words, network coding is possible only when there exists certain "coding structure" that is similar to the ones shown in Figure 1. In COPE, network coding functions as a separate layer from the MAC and network layers. If one uses the shortest-path routing, or some recently proposed ETX-like routing [2], [3], the potential coding opportunity may be significantly reduced. To illustrate, consider the example in Figure 2 where there are two flows to be routed. Without consideration on potential coding opportunities, the disjoint paths shown in Figure 2(a) may very likely be chosen. On the other hand, if we use a *coding-aware* routing decision as shown in Figure 2(b), node 3 has the opportunity to perform network coding. In this example, coding-aware routing will result in a higher end-to-end throughput for both flows if we assume a two-hop interference model, i.e., the interference range is about twice the transmission range under the 802.11.



(a) Routing without coding consideration.

(b) Routing with coding consideration at node 3.

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

The second limitation of COPE is that it *limits* the entire coding structure within a *two-hop region*. To illustrate, consider the example as depicted in Figure 1(a). COPE assumes that the transmitters for opportunistic overhearing (i.e., node $1, 3$) are the one-hop predecessors of node C, and that the intended receivers (i.e., node $4, 2$) 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 Figure 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 overhearing for decoding, it can forward the encoded packet to node 7, where the opportunistic overhearing and decoding can take place. The important point is, both the opportunistic overhearing 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.

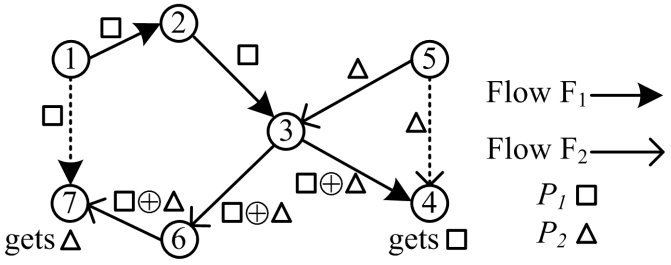


Fig. 3. Example: the generalized coding scheme.

The above limitations raise some challenging and interesting questions, for example, is it possible to incorporate consideration on potential coding opportunities into the route selection? Can a routing scheme examines beyond two hops to discover more coding opportunities? How to evaluate and compare the performance of a coding-possible path (we refer to a "coding-possible path" as a path where certain encoding and decoding nodes exist) 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:

- We propose a distributed routing mechanism that can concurrently discover the available paths and potential coding opportunities.
- We formally define the generalized *coding conditions*, in which the practical network coding can occur. These conditions help us to design algorithms which an 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 a coding-possible path or coding-impossible path.
- 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 II, we describe the "Coding+Routing Discovery" which combines the detection processes of available paths and potential coding opportunities. This new discovery mechanism removes the "two-hop" limitation of COPE, and makes it possible to perform coding-aware route selection. In Section III, we formally introduce the coding-aware routing metric, which quantifies the potential benefit of "coding-possible" paths, and facilitates the comparison between "coding-possible" and "coding-impossible" paths. The overall system design of DCAR is presented in Section IV. Simulation results are presented in Section V Related work is given in Section VI and finally, conclusion is given in Section VII.

II. The "Coding+Routing" Discovery

It is important to point out that 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 route*, therefore, routes in Figure 2(a) may be chosen instead of the routes with coding opportunity in Figure 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 combine solution of "coding+routing" to overcome the above discussed limitations.

A. Assumptions

We first state the underlying assumptions we use in this paper. We refer to a "*coding node*" as a node which encodes packets, e.g., node C in Figure 1 or node 3 in Figure 3. A "coding structure" is a collection of nodes and flows including the necessary transmitters for opportunistic overhearing, the coding node, the intended receivers which decode packets, and the necessary relaying nodes connecting the flows. The structures shown in Figure 1 and Figure 3 are all examples of coding structures. We consider coding structures as the basic building blocks for general networks which use the network coding paradigm.

Throughout this paper, we focus on the inter-flow coding fashion similar to the ones used in COPE [1]. The philosophy is to make sure every encoded packet must be decoded by the intended receiver, as opposed to proposals for randomized and intra-flow coding [13]–[15]. By far the inter-flow coding in COPE is the most practical and realizable application for network coding. In the rest of this paper, unless we state otherwise, we consider a stationary multi-hop wireless network.

B. General Coding Conditions

In order to discover paths with potential coding opportunity, we need to first state the *necessary* and *sufficient* conditions in which network 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 Figure 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, say node c, packets of these two flows can be encoded for transmission at node c *if and only if* the *coding conditions* are met. The definition of coding conditions is specified as follows:

[1]In the remaining of this paper, unless we state otherwise, we refer to "paths" and "flows" interchangeably.

Definition 1: *Coding conditions for two flows, say F_1 and F_2, which intersect at node c, are:*

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: *Assuming perfect channel condition and scheduling (e.g., no packet loss or collision), the above conditions are the necessary and sufficient conditions for any proper coding and decoding to occur.*

Proof: To ensure that the destinations of both flows get their respective "native" packets, there must exist some downstream nodes (i.e. $d_1 \in D(c, F_1)$ and $d_2 \in D(c, F_2)$) which can extract the "native" packets included in the encoded one. For example, in Figure 3, we have $4 \in D(3, F_1)$ such that $4 \in N(5)$ and $4 \in U(3, F_2)$. Without loss of generality, let us consider d_1. It must have P_2 before it receives the encoded $P_1 \oplus P_2$, and there are only two ways for d_1 to obtain P_2: either by the fact that d_1 can overhear the transmission of P_2 by some node in $U(c, F_2)$ (i.e., $d_1 \in N(s_2), s_2 \in U(c, F_2)$), or by the fact that d_1 itself has transmitted P_2 (i.e., $d_1 \in U(c, F_2)$). Without these, d_1 cannot extract P_1 out of the encoded packet. This proves the necessity of the coding conditions. On the other hand, assuming perfect channel condition and scheduling, either way can let d_1 receives P_2 before it receives the encoded packet, therefore, this proves the sufficiency of the coding conditions. ∎

Remark: Note that we assume perfect channel condition and scheduling in the above coding conditions. In practice some opportunistic overhearing may be *unsuccessful*, either due to channel fading or due to packet collision at the link layer. To cope with such effect in practice, we only select those neighboring nodes which have a high probability of overhearing (say greater than 0.8) in the coding judgement. The details will be presented in the following sections. For more than two intersecting flows, the common node in these intersecting flows needs to check that the above conditions hold for *any two* of the intersecting flows in order to determine whether it can encode packets of these flows all together.

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 when it has the following information:

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

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

C. Distributed "Coding+Routing" Discovery

Let us now 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 II-B. Note that when we detect a path with coding opportunity (and we call this the *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. In Section III, we will present this in full detail.

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 $a \rightarrow b$ where $b \in N(a)$.

When a new flow arrives to the wireless network, the source node of this new flow 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. $\{a|a \in N(s), P(s, a) > threshold\}$. The threshold value can be predefined by the network designers or operators. We believe a threshold value greater than 0.7 will be sufficient. Unless we state otherwise, in our ns-2 implementation, we set the $threshold$ to 0.8.
- 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 overhearing" 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 the 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 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 II-B. 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 III), and the source node begins to send data packets on the selected path.

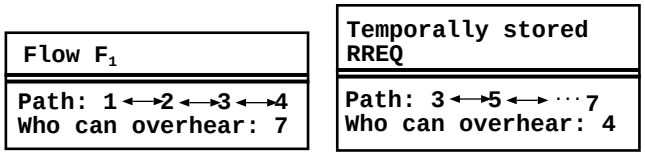
Step 6. When the first data packet reaches an intermediate node, say node c, it stores the "who-can-overhear" and "path" information for the selected path, while discarding other temporally stored information.

In summary, the key differences from conventional DSR path discovery include:

- RREQ contains one-hop neighbors and link qualities. This is to inform intermediate nodes the overhearing information along the path.
- Each node temporally stores RREQs during the discovery phase. This is to facilitate the matching with RREPs received later.
- Each node maintains overhearing and path information for all flows passing it. This piece of information is used to decide whether a new flow can be encoded with existing ones.

D. An Illustrative Example

We use the simple wireless network in Figure 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 Figure 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:



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

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

Fig. 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 Figure 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 Figure 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.

E. Overheads of Distributed Coding+Routing Discovery

Before we proceed to the next section, let us quantify the complexity and overheads of this distributed coding+routing discovery process. In general, we exploit on the flooding of RREQ messages to help intermediate nodes to collect the "*who-can-overhear*" information, and use RREP messages to inform the wireless nodes of the "*path*" information. Therefore:

- *Overhead due to flooding of* RREQ: this overhead is associated with any on-demand routing like DSR [16], [17] or AODV [18]. There have been several optimizations for reducing the flooding overhead [17]–[19], however, the reduction in flooding overhead comes with a cost of a reduced set of available paths. In DCAR, there is a clear tradeoff between the flooding overhead and the amount of potential coding opportunities discovered. In particular, one can pre-define a constant message overhead by setting a proper TTL value for the RREQ.

- *Storage overhead at intermediate nodes*: as required by the discovery process, each node (denoted by a) needs to store its one-hop neighbors $N(a)$ and corresponding packet loss probabilities $P(a,b)$ for all $b \in N(a)$. Each intermediate node along an existing flow also needs to remember the "path" information and "who-can-overhear" information for this flow. Suppose each node's ID or packet loss probability takes 4 bytes of storage, let DEG^* denote the maximum node degree of the wireless network, and let NF^* denote the maximum number of flows traversing a node, then the constant storage overhead (in bytes) at one node will not exceed

$$8 \times DEG^* + 4 \times NF^* \times [(TTL-1) \times DEG^* + TTL], \quad (1)$$

  where TTL is the pre-defined time-to-live value in RREQ.

  Let us provide some explanation to Equation (1): the first component refers to the storage overhead for one-hop neighbors and corresponding packet loss probabilities, while the second component refers to the storage overhead for existing flows. Since TTL is an upper bound on the path length, therefore the number of upper-stream nodes along a flow is at most $TTL-1$, and the number of overhearing nodes is at most $(TTL-1) \times DEG^*$.

  It is important to note that each node also needs to temporally store the "path" and "who-can-overhear" information for a newly found path. Because we use a "soft-state" [20] approach to handle such storage, they do not contribute to the constant overhead.

- *Overhead for the extra length in the* RREQ *packet*: this overhead corresponds to the number of overhearing nodes of previous hops. Using the above notations, such overhead (in bytes) will not exceed

$$4 \times (TTL-1) \times DEG^*. \quad (2)$$

Note that these communication and storage overheads are included in our ns-2 implementation. The tradeoff is of course whether we can improve the system performance (e.g., increase throughput or reduce bandwidth consumption) with these overheads. We will answer this important question in Section V.

III. Defining Coding-Aware Routing Metric

In the previous section, we presented the distributed algorithm to discover both available paths as well as their potential coding opportunities. Another challenging question is *how to choose a good path among these available choices*. Note that one 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 first review some existing routing metrics, and then present the proposed Coding-aware Routing Metric (CRM).

A. Brief Review of Current Routing Metrics

For wireless networks, there are basically two types of routing metrics proposed: the *topology-based* metrics and *load-based* metrics. We briefly review some representative metrics and their associated algorithms here:

(1) Hop-count based routing: The minimum hop count routing is probably the most often used topology-based routing algorithm due to its simplicity and ease of implementation. However, in wireless networks, hop-count based routing cannot guarantee to find a high-throughput path since it does not take link interference, fading channel and traffic load into consideration.

(2) ETX-based routing: ETX (Expected Transmission Count) routing [2] chooses the path with the minimal number of expected successful transmissions. It is a topology-based routing algorithm and is most effective when there is significant packet loss due to channel fading. Let L denote the path, let $l \in L$ denote a link on the path, and let P_l denote the packet loss probability on link l, then the routing metric $\mathcal{M}_{ETX}(L)$ for link l is computed as

$$\mathcal{M}_{ETX}(l) = \frac{1}{(1-P_l)}.$$

which accounts for the expected number of transmissions for a successful packet transmission. ETX for the path L is simply the sum of ETX for all of its links:

$$\mathcal{M}_{ETX}(L) = \sum_{l \in L} \mathcal{M}_{ETX}(l)$$

and the path L^* is selected when its measure $\mathcal{M}_{ETX}(L^*)$ is the minimum among all available paths. ETT [3] (Expected Transmission Time) extends ETX by taking into account the packet size and data rate on the link. We do not further discuss this metric because we focus on single-rate networks.

IRU (Interference-aware Resource Usage) [4] further takes into account the number of interfering neighbors of each link. In a single-rate network, the performance metric $\mathcal{M}_{IRU}(L)$ of the path L is computed as

$$\mathcal{M}_{IRU}(L) = \sum_{l \in L} \mathcal{M}_{ETX}(l) \times N_l,$$

where N_l is the number of interfering neighbors of link l.

(3) Load-Based Routing [5], [6]: These routing mechanisms take into account the current interfering traffic when making a routing decision. To make a proper decision, the required information typically includes the "channel busy time" and packet loss probability sensed by each node on the path. Despite of the increased accuracy in estimating the potential throughput, such approach may also bring substantial overhead since nodes in the same interference region may sense *different* channel states, and the fact that links on the new path may interfere with each other, i.e., *self-interference*. Additionally, such approach usually requires nodes to be aware of the throughput of existing flows, and requires the wireless card to report the "channel busy time" to higher protocol layers, which may not be feasible in practice.

B. Desirable Properties of Coding-aware Routing Metric

Let us first 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 desirable properties onto the coding-aware routing metric:

1) The metric should take into account the "free-ride" benefit of the *coding-possible* paths: if a new flow can be encoded with some existing flows, it can "*free-ride*" on the bandwidth used by the existing flows.
2) The metric should be *general* in quantifying the merits 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.

For the first desirable 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, a node does not know the actual throughput of on-going flows. For the second desirable property, we face another technical challenge on 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.

C. Assumptions on Encoded Transmission

First, it is important for us to clarify our assumptions on the encoded transmission. Using the coding conditions we specified in Section II-B, a node can decide for each flow, which other flow(s) to encode with. The local coding relationship can be complicated. For example, flow 1 may be able to encode with flow 2, while flow 2 can encode with 3, but flow $1, 3$ cannot be encoded with each other. The question is to choose which flows to encode, upon each channel access opportunity. One approach is to encode as many packets as possible to maximize the bandwidth efficiency, however, the problem of finding the maximum number of flows to encode can be reduced to the *Maximum Clique Problem* [10], which is NP-complete (we will present the detail in Section III-F). Instead, we use a *round-robin* encoding scheme: whenever having an opportunity to transmit, the node randomly picks one flow, and encode as many flows as possible with the chosen packet. Although finding the maximum flows to encode with a given flow is itself NP-complete, it is established in [12] that the number of flows that can encode with one given flow is bounded by a small number (ranging from 4 to 7), thus the computational overhead is insignificant.

Once the node has decided on the set of packets to encode, it has to perform the *encoded broadcasting*. The encoded packet is supposed to be broadcasted to multiple receivers, however, since there is no ACK for broadcast packet in the 802.11 standard, we use the "pseudo-broadcast" technique [1]—the encoded packet is unicasted to one of the receivers while this transmission can be overheard by other intended receivers. The chosen unicast receiver is responsible for sending ACK back to the coding node. In the round-robin coding scheme, we randomly pick the unicast receiver among all the intended ones.

D. Interpreting the "Network Coding" Benefit

Let us illustrate the benefit of network coding To achieve this, consider the following simple example so as 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 interpretation the network coding benefit is intuitive, however, one needs to know the throughput of the on-going traffic to determine the benefit, and as we discussed previously, this is difficult to obtain in practice. Now let

us consider another approach to quantify the benefit: *by 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 so that we can quantify the benefit of network coding on an existing encoding path.

E. Feasibility of Using Queue Length as Routing Metric

The above consideration leads us to investigate the feasibility of using queue length as the coding-aware routing metric. One immediate question would be the stability issue: with dynamic traffic the queue length may change too frequently to be a good measure of available bandwidth. To visualize to the queue size, we set up a 15-node random wireless network, and add a few random UDP flows. In Fig. 5 we plot its queue length statistics over 60 seconds. The curves shown include the instantaneous queue length every one second, the average queue length in the last 5 instant values and 10 instant values respectively. Clearly, the instantaneous queue length varies significantly over time (with a variance of 19.48 in this case). However, after averaging the last few instant values the variance drops dramatically (variance 1.23 for 5 average and 0.32 for 10 average). This implies that the *Average Queue Length* can be a much more stable and accurate measure of bandwidth availability. Therefore, we use the *Average Queue Length at Each Node* as the starting point, and *modify* the queue length to account for various benefits of using network coding.

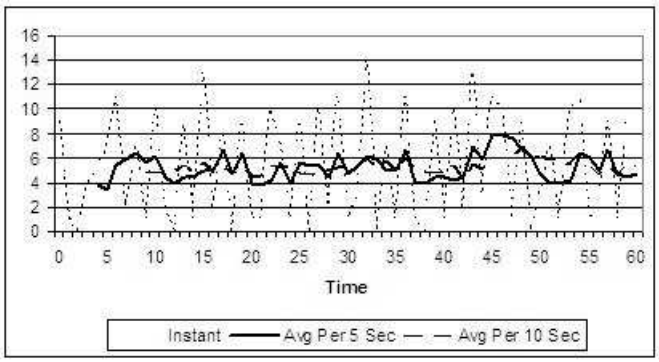
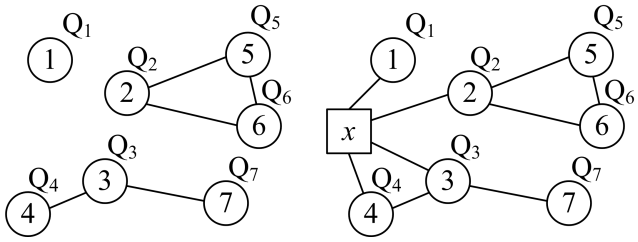


Fig. 5. Queue length statistics in a sample wireless node.

F. 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 $\max\{Q_1, Q_2\}$. However, when there are more flows intersecting at one node, the modification becomes less obvious. 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.

Fig. 6. Examples of coding graph for the considered node c.

Note that we differentiate between *Stable* case and *Dynamic* case in the above graph. A *Stable* coding graph is used to represent the local coding relationship between *existing* flows. It is effectively *average number of packets a node sees in its own queue based on current traffic*, and can be used to calculate the modified queue length based on *existing* flows.

A *dynamic* coding graph is more focused on the coding relationship between a *potential* new flow and existing flows. It takes into account the "free-ride" benefit as the effective queue lengths "seen" by the new flow can be reduced if it can be encoded with some existing flow(s). With the dynamic coding graph we are trying to examine the average number of packets (in the queue) that *can not* be encoded with the new flow.

Figure 6 shows two examples of the coding graph for a considered node, in the *stable* case and *dynamic* case respectively. In stable case (Figure 6(a)), there is no new paths (or flows) to be added. In dynamic case (Figure 6(b)), there is a new path[2] 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

[2]Here by "new path" we mean one of the possible new flows that could be formed from source and destination.

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 [10]. To reduce computational cost, we can calculate the modified queue length based on the *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 Fig. 7.

Calculation of modified queue length (stable 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.

Fig. 7. Calculation of modified queue length under the stable case.

Base on Fig. 7, one can observe that Step 3-8 mimics 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 [12], the computational cost in each iteration is insignificant. Using Figure 6(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. 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 of the modified queue length for the dynamic case is depicted in Figure 7.

Calculation of modified queue length (dynamic case):
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.

Fig. 8. Calculation of Modified Queue Length under the 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 6(b), we first remove vertex $x, 1, 2, 3, 4$ and all their edges from the graph, and the resulting $MQ_d(c)$ is $\max\{Q_5, Q_6\} + Q_7$.

G. 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, with

$$MIQ_s(c) = MQ_s(c) + \sum_{i \in I(c)} MQ_s(i), \quad (3)$$

$$MIQ_d(c) = MQ_d(c) + \sum_{i \in I(c)} MQ_s(i), \quad (4)$$

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 virtual 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.

H. 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:

$$CRM_l = \frac{1 + MIQ_d(l)}{1 - P_l}. \quad (5)$$

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:

$$CRM_L = \sum_{l \in L} CRM_l. \quad (6)$$

Compared to routing metrics we reviewed in Section III-A, 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.

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

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. Periodical HELLO exchange has been widely adopted by most of the routing protocols [2]–[6] reviewed in Section III-A. In this sense, DCAR does not impose more message overhead on these existing protocols.

IV. Implementation Details

We now present the implementation details of DCAR in ns-2. We modified the DSR routing agent [22] in ns-2 to include the "coding+routing" discovery and path selection functions. We also modified the Interface Queue to include encoding and decoding functions. The overall architecture of DCAR is shown in Figure 9.

The DCAR routing agent maintains a list of one-hop neighboring nodes and the corresponding link qualities (i.e. packet loss probabilities) by periodically broadcasting HELLO messages (the HELLO interval is set to 0.5 second in our ns-2 implementation). When sending the HELLO, each node piggybacks its "*Modified Queue (MQ)*" length as well as its one-hop neighbors and their MQ lengths. In this way, each node can obtain the queue length information of its *two-hop neighbors*. Because the carrier sensing range is approximately two times of the transmission range in 802.11, we define the "interfering nodes" ($I(c)$) to be the two-hop neighbors of a node (c). In the HELLO message, each node also piggybacks the number of HELLO messages it receives from its neighbor in the last 5 seconds to let its neighbors examine the reverse links.

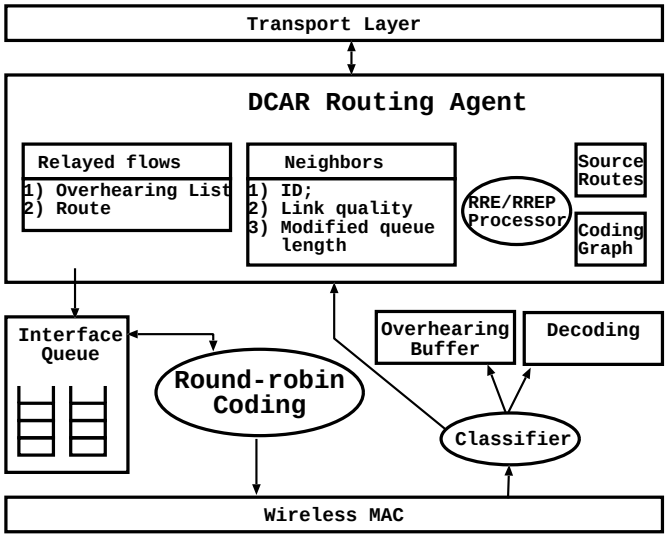


Fig. 9. DCAR architecture.

The RREQ/RREP processing is essential for the "coding+routing" discovery. We have already presented how to find potential coding opportunities in Section II, and the calculation of CRM (Coding-aware Routing Metric) is done on a link by link basis when the RREP travels back to the source node. In particular, when an intermediate node receives a RREP (in Step 4 in Section II-C) and finds the potential coding opportunity, it calculates the CRM and records it into the RREP. When the RREP(s) return back to the source, the source node chooses the path with the minimum CRM value and starts to transmit data packets along the path. Once an intermediate node receives the data packet, it records the "overhearing" and "path" information of the new flow into the list of "Relayed Flows". Based on the list of relayed flows, the DCAR agent updates the "Coding Graph", which represents the local coding relationship among the relayed flows. For proper decoding, the system also maintains a circular buffer for the overheard packets and packets it has transmitted.

In the interface queue, we maintain a separate queue for each flow. The advantages of such queueing structure are : 1) the congestion of one flow will not cause buffer overflow for other flows; 2) it facilitates the round-robin coding. For the queue of each flow, we update the average queue length every second. We also add the encoding and decoding functions into the interface queue. Whenever there is an transmission opportunity, a *round-robin encoding* takes place and several packets may be cleared out of the queue. For the encoded packets, we use similar packet formate as in COPE [1]. Whenever we receive a packet, we first check whether it is a *native* or *encoded* packet using the "Classifier", and then forward it up to upper layers or decode it accordingly. For all the opportunistically overheard packets, we store them in a circular "Overhearing Buffer" for future decoding usage.

V. Performance Evaluation

We now present the simulation results. We implement the DCAR and COPE [1] system under ns-2. There are three main differences between DCAR and COPE: 1) DCAR takes potential coding into consideration of route selection, while COPE separates routing with potential coding; 2) COPE uses ETX [2] as the routing metric[4] while DCAR uses CRM; 3) COPE limits the coding structure within two hops while DCAR eliminates 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 DCAR over COPE. Throughout the study, we use 802.11b and UDP traffic sources. The transmission range of each wireless node 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. The TTL value of RREQ is set to 5.

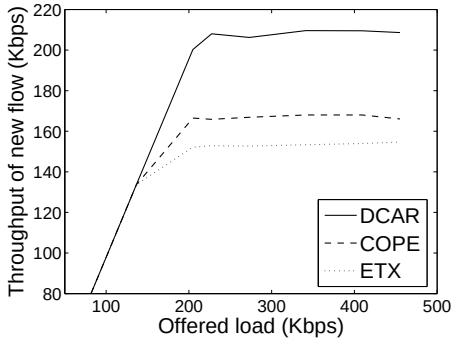
A. Results from Illustrative Scenarios

Simulation 1. Bidirectional flows: We first study the simple scenario shown in Figure 2. We start a flow from node 1 to

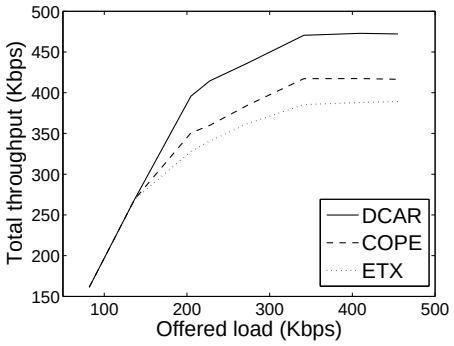
[4]Our COPE implementation uses DSR with ETX as routing metric.

node 2, and then add the new flow 2 to 1. The flows are given the same traffic load. Whether a coding structure can be formed depends on the route selection for flow $2\ to 1$. Under COPE or ETX, the path $2-3-1$ and $2-4-1$ are almost of same quality. However, under DCAR, a reverse path for the previously added flow will be more favorable because the intersecting node can have a much lower *modified queue length* after accounting for the "free-ride" benefit.

We vary the offered load and plot the resulting end-to-end throughput in Figure 10. 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.

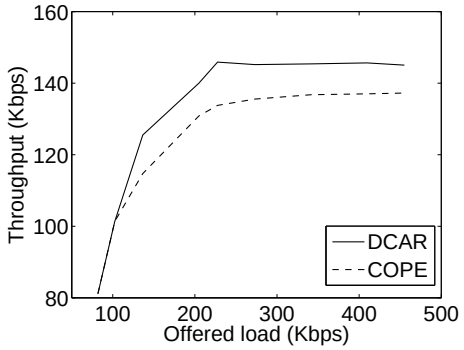


(b) Total throughput.

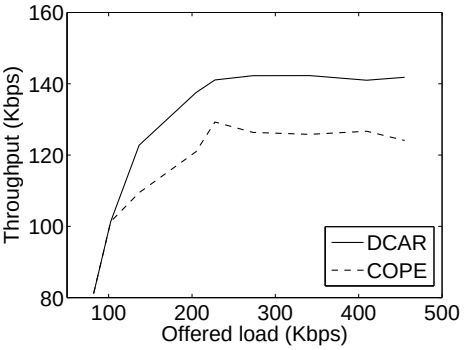
Fig. 10. Results from the topology in Figure 2.

Simulation 2. Generalized coding: Here we observe the effectiveness of DCAR in overcoming the "two-hop limitation". We compare the performance of DCAR and COPE using the topology shown in Figure 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 overhearing and decoding. Therefore node 3 becomes a bottleneck in COPE.

For each offered load, we repeat the simulation 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 Figure 11. 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.

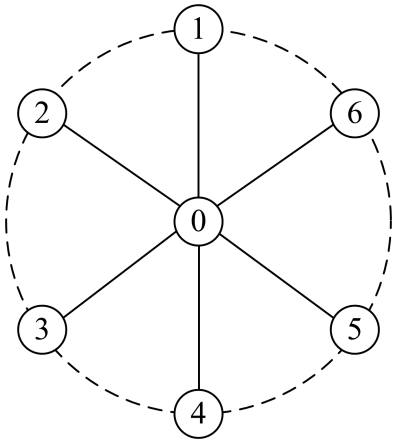
Fig. 11. Results from the topology in Figure 3.

Simulation 3. "Wheel" topology: It is interesting to study how DCAR works in a "wheel" topology as shown in Figure 12(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. The "wheel" structure is a generalized model for any coding structure in COPE, and has been well studied in [12]. 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.

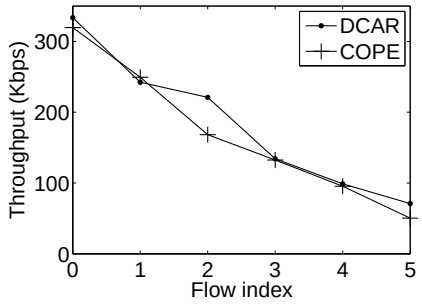
In the simulation, we vary the traffic load and arrival order of each flow, and plot the average throughput in Figure 12(b). We can see that DCAR typically offers higher throughput than COPE, but the gain is not as significant as the previous scenarios. The underlying reason is that even if the paths are randomly chosen between available shortest paths, there are still many coding opportunities at the surrounding nodes as we discussed.

B. Results from Mesh Networks

Simulation 4. Grid topology: Now we consider larger-scale networks. We construct a 4 by 4 grid topology where each

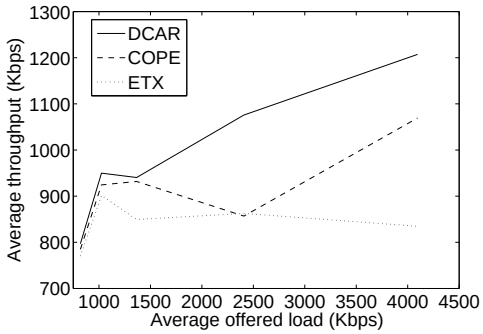


(a) "Wheel" topology.

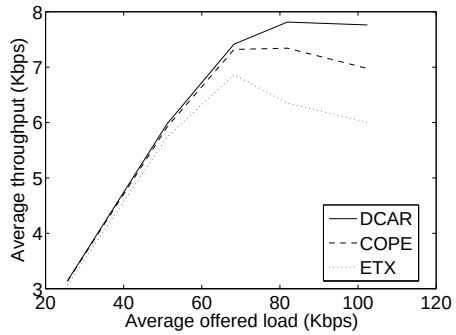


(b) Throughput of each flow.

Fig. 12. Results from a "wheel" topology.



(a) Results by adding 5 flows.



(b) Results by adding 10 flows.

Fig. 13. Results from a grid topology.

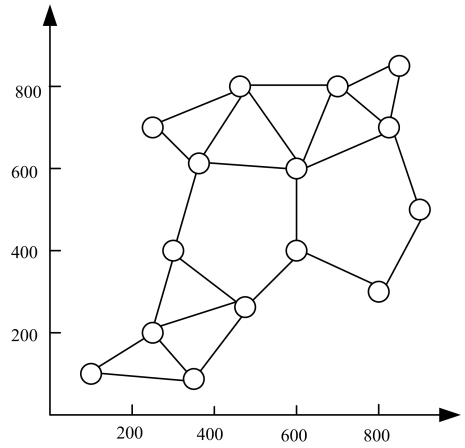
node can only reach its northern, southern, eastern and western nodes. There is a rich set of spatial reuse as well as coding opportunities in this example. The simulation is of 10 rounds. At each round, we randomly add 5 flows (each with 2 to 5 hops) into the network and repeat the process for 3 times. We plot the average end-to-end throughput achieved by DCAR, COPE and ETX respectively in Figure 13(a). Not surprisingly, the gain by DCAR tends to be larger with higher offered load. In Figure 13(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.

Simulation 5. Random topology: We compare DCAR and COPE in a 15-node random topology as shown in Figure 14(a). The average node degree is 3.2. We randomly pick 8 flows (each with 2 to 5 hops) and vary their arrival orders and loads in each round. The average throughput for each flow is plotted in Figure 14(b). Because there is a rich set of coding opportunities and available paths, DCAR achieves substantial throughput gains over COPE.

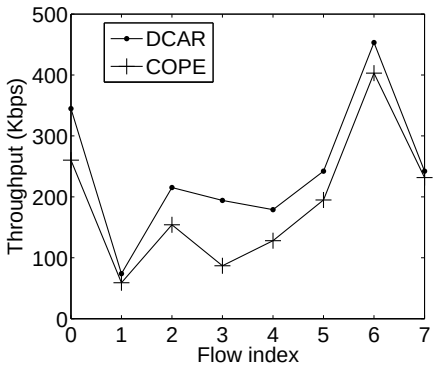
Simulation 6. Fraction of Encoded Traffic: It is interesting to examine how many traffic are actually encoded in DCAR and COPE. In this simulation, we randomly add flows into both the grid and the 15-node topology, and count for the total data packets transmitted in all links accordingly. For a native packet, we consider it as one unit of traffic; while for an encoded packet with n native packets, we consider it as n units of traffic. The fractions of encoded traffic (in grid and random topology) are shown in Figure 15. We can see that the random topology usually has more coding opportunities, mainly because of the increased chances for opportunistic overhearing.

Simulation 7. Routing Overhead: In here, we quantify the overheads of DCAR. As discussed in previous section, the overhead of DCAR includes flooding of RREQ messages and the periodic exchange of HELLO messages. However, both of these are also needed for ETX or any other recently proposed link-state routing mechanisms, e.g., [4]–[6]. Therefore, they should have similar routing overhead in terms of packet counts. However, because DCAR piggybacks extra information into the routing control messages, it has a higher overhead compared to ETX in terms of total bytes of routing messages. To quantify these overheads, we carry out simulation study and in Figure 16, we plot the normalized routing overhead both in number of packets and in bytes. The way we compute the overhead is to sum up all the RREQ, RREP and HELLO messages (in bytes or in packet counts) transmitted in all links during whole simulation time. It is interesting to note that these results confirmed our intuition: DCAR needs about $20 \sim 30\%$ more bytes, but similar number of routing messages compared to ETX. Although DCAR requires more bytes in communication, the gain we have is in the improvement of end-to-end throughput and reduction in bandwidth consumption.

Simulation 8. Fraction of Different Types of Coding Structures: It is interesting to observe how many coding structures



(a) 15-node random topology.



(b) Throughput of each flow.

Fig. 14. Results from a 15-node random topology.

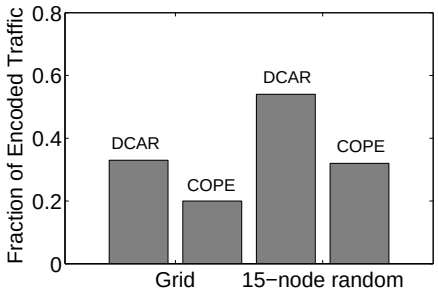
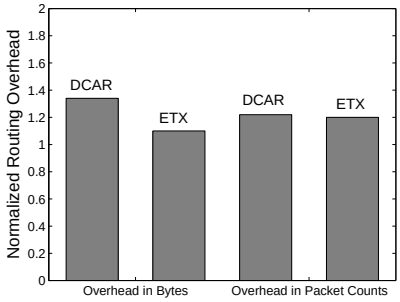
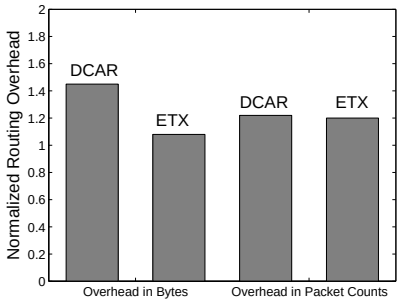


Fig. 15. Contribution of encoded traffic in different settings.



(a) Grid.



(b) 15-node random.

Fig. 16. Routing overhead.

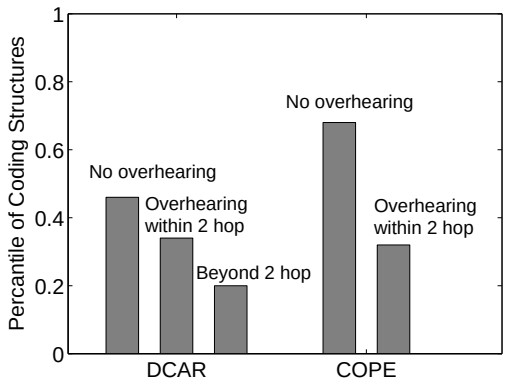


Fig. 17. Percentile of different types of coding structures (in 15-node random topology).

are formed within two-hop (as in COPE), and how many are beyond two-hop (allowed by DCAR), and furthermore, how many are due to opportunistic overhearing. During the simulation with the random topology, we analyze each formed coding structure and differentiate them into three categories: 1) without opportunistic overhearing; 2) with overhearing and within two-hop; 3) with overhearing and beyond two-hop. The fraction of these three types of coding structures are plotted in Figure 17.

C. **Remarks on the Results**

In simulation 4 and 5, we have been using flows with 2 to 5 hops, as we found that the with more than 5 hops the flow throughput become very bad and there is no visible difference with or without network coding. This is due to scalability problem of 802.11 in multi-hop environment and has been studied extensively in the literature [24]. DCAR proposed in this paper is MAC-independent, but does rely on the MAC layer to provide low-collision connections. With more scalable MAC protocol in future, we expect DCAR to offer performance gains in bigger network with longer flows.

We have applied several empirical values in the simulations, e.g., the "good link" assumption of above 0.8 overhearing rate, the 3 seconds waiting time in discovery phase, etc. These are the values a network designer can tweak based on traffic and channel conditions. There are clearly trade-offs behind these numbers: with lower overhearing threshold, one can

find more links and potentially more coding opportunities, but these come with higher chance of transmission failure. With longer waiting time in discovery phase, one can find more potential paths, but these come with increased delay and reduced timeliness of flow status. How to adjust these values online remains an open question.

VI. **Related Work**

The concept of network coding is first proposed in [7]. Since then, there are various studies concerning theoretical limitations and practical applications of network coding. Authors in [25] show the practical and security issues of batch content distribtion via network coding in wired networks. For multiple unicast sessions in wireless networks, authors of [23] show that the throughput gain is upper bounded by $\frac{1+\Delta}{1+\Delta/2}$ in 1D random networks, and upper bounded by $2c\sqrt{\pi}\frac{1+\Delta}{\Delta}$ in 2D random networks, where Δ is a parameter characterizing the intensity of interference, and $c = \max\{2, \sqrt{\Delta^2 + 2\Delta}\}$. It is conjectured in [23] that the throughput gain is also upper bounded by 2 in 2D random networks.

Recently, authors of [1] propose COPE, the first practical XOR coding system and demonstrate the throughput gain via implementation and measurement. COPE utilizes *inter-flow* coding and uses ETX [2] as its routing layer. Based on COPE, authors of [8], [9] 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 link-scheduling. Authors of [11] propose a complex optimization framework for adaptive coding and scheduling. Authors of [12] study limitations of COPE under practical physical layer and link-scheduling algorithms, propose the concept of coding-efficient link-scheduling for practical network coding. In [13], the authors implement a *intra-flow* random coding system and demonstrate the throughput gains for lossy wireless networks.

Compared with former works, DCAR falls into the category of *inter-flow* coding. It is the first practical coding-aware routing system, and adopts a more generalized coding scheme by eliminating the "two-hop" limitation in COPE.

VII. **Conclusion**

We propose DCAR, the first distributed coding-aware routing system for wireless networks. DCAR is an on-demand and link-state routing protocol, it 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. In our work, we proposed to use average queue length as an estimator. Our current work includes the optimal control on queue length averaging and responsiveness of traffic changes in routing decisions. One possible future direction of this work is how to provide resiliency and to guarantee network coding opportunity in the face of link/node failure.

Acknowledgement: we like to thank the anonymous referees for providing useful and insightful comments. This reseach is supported by RGC Grant 415708.

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, pp. 243-254, 2006.*

[2] D. Couto, D. Aguayo, J. Bicket and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. *Wireless Networks, 11(4), pp. 419-434, 2005*.

[3] R. Draves, J. Padhye and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. *Proceedings of ACM MOBICOM, pp. 114-128, 2004.*

[4] Y. Yang, J. Wang and R. Kravets. Designing Routing Metrics for Mesh Networks. *Proceedings of WiMesh 2005*.

[5] Y. Yang and R. Kravets. Contention-Aware Admission Control for Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 4(1), pp. 363-377, 2005.

[6] T. Salonidis, M. Garetto, A. Saha and E. Knightly. Identifying High Throughput Paths in 802.11 Mesh Networks: a Model-based Approach. *Proceedings of ICNP, pp. 21-30, 2007*.

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

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

[9] S. Sengupta, S. Rayanchu and S. Banerjee. An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing. *Proceedings of INFOCOM, pp. 1028-1036, 2007*

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

[11] P. Chaporkar and A. Proutiere. Adaptive Network Coding and Scheduling for Maximizing Throughput in Wireless Networks. *Proceedings of ACM MOBICOM, pp. 135-146, 2007*.

[12] J. Le, JCS Lui and DM Chiu. How Many Packets Can We Encode? - An Analysis of Practical Wireless Network Coding. *Proceedings of INFOCOM, pp. 371-379, 2008* (under submission to IEEE Transactions on Mobile Computing).

[13] S. Chachulski, M. Jennings, S. Katti and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. *Proceedings of SIGCOMM, pp. 169-180, 2007.*

[14] T. Ho, R. Koetter, M. Médard, D.R. Karger and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. *IEEE International Symposium on Information Theory (ISIT) 2003.*

[15] P. A. Chou, Y. Wu and K. Jain. Practical network coding. *Proceedings of Allerton Conference on Communication, Control, and Computing (Allterton) 2003.*

[16] D. Johnson, D. Maltz and J. Broch. DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. *Ad Hoc Networking*. Chapter 5, pp. 139-172, Addison-Wesley, 2001.

[17] D. Johnson, Y. Hu and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobiel Ad Hoc Networks for IPv4. http://www.ietf.org/rfc/rfc4728.txt, *RFC 4728*.

[18] C. Perkins, E. Belding-Royer and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. http://www.faqs.org/rfcs/rfc3561.html, *RFC 3561*.

[19] S. Das, C. Perkins and E. Royer. Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks. *Proceedings of IEEE INFOCOM 2000.*

[20] D.D. Clark. The Design Philosophy of the DARPA Internet Protocols. *Proceedings of ACM SIGCOMM, pp. 106-114, 1988.*

[21] The Network Simulator, NS-2. http://www.isi.edu/nsnam/ns/.

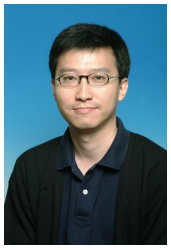
[22] Bryan's NS-2 DSR FAQ. http://www.geocities.com/b_j_hogan/.

[23] J. Liu, D. Goeckel and D. Towsley. Bounds on the Gain of Network Coding and Broadcasting in Wireless Networks. *Proceedings of IEEE INFOCOM, pp. 1658-1666, 2007.*

[24] S. Xu and T. Saadawi. Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?. *IEEE Communications Magazine*, 39(6), pp. 130-137, June 2001.

[25] Qiming Li, D.M. Chiu, JCS Lui. On the Practical and Security Issues of Batch Content Distribution Via Network Coding. *International Conference on Network Protocols (ICNP), pp. 158-167, 2006.*

Jilin Le received the B.Eng degree in Electrical Engineering from Peking University, and the M.Phil degree in Computer Science from the Chinese University of Hong Kong. His research interests include wireless networks, network protocols and applications.



John C.S. Lui received his Ph.D. in Computer Science from UCLA. He is currently the chairman of the Computer Science & Engineering Department at the Chinese University of Hong Kong. His research interests span both in systems as well as in theory/mathematics with the emphasis on the robustness, scalability, and security issues on the Internet. John received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award, as well as the co-recipient of the Best Student Paper Awards in the IFIP WG 7.3 Performance 2005 and the IEEE/IFIP Network Operations and Management (NOMS) Conference. He is an associate editor in the Performance Evaluation Journal, IEEE-TC, IEEE-TPDS and IEEE/ACM Transactions on Networking. John was the TPC co-chair of ACM Sigmetrics 2005 and the General Co-chair for ICNP 2007.

Dah-Ming Chiu received the B.Sc degree in Electrical Engineering from Imperial Collage, University of London, and the Ph.D degree from Harvard University, in 1975 and 1980 respectively.

He was a Member of Technical Staff with Bell Labs from 1979 to 1980. From 1980 to 1996, he was a Principal Engineer, and later a Consulting Engineer at Digital Equipment Corporation. From 1996 to 2002, he was with Sun Microsystems Research Labs. Currently, he is a professor in the Department of Information Engineering in The Chinese University of Hong Kong. He is known for his contribution in studying network congestion control as a resource allocation problem, the fairness index, and analyzing a distributed algorithm (AIMD) that became the basis for the congestion control algorithm in the Internet. His current research interests include economic issues in networking, P2P networks, network traffic monitoring and analysis, and resource allocation and congestion control for the Internet with expanding services. Two recent papers he co-authored with students have won best student paper awards from the IFIP Performance Conference and the IEEE NOMS Conference. Recently, Dr Chiu has served on the TPC of IEEE Infocom, IWQoS and various other conferences. He is a member of the editorial board of the IEEE/ACM Transactions on Networking, and the International Journal of Communication Systems (Wiley).