FlexONC: Joint Cooperative Forwarding and Network Coding With Precise Encoding Conditions

Somayeh Kafaie, Yuanzhu Chen, Mohamed Hossam Ahmed, and Octavia A. Dobre, *Senior Member, IEEE*

***Abstract*—In recent years, network coding has emerged as an innovative method that helps a wireless network approach its maximum capacity by combining multiple unicasts in one broadcast. However, the majority of research conducted in this area is yet to fully utilize the broadcasting nature of wireless networks and still assumes a fixed route between the source and destination that every packet should travel through. This assumption not only limits coding opportunities but can also cause buffer overflow in some specific intermediate nodes. Although some studies considered scattering of the flows dynamically in the network, they still face some limitations. This paper explains pros and cons of some prominent research in network coding and proposes a Flexible and Opportunistic Network Coding Scheme (FlexONC) as a solution to such issues. Furthermore, this research discovers that the conditions used in previous studies to combine packets of different flows are overly optimistic and would affect the network performance adversely. Therefore, we provide a more accurate set of rules for packet encoding. The experimental results show that FlexONC outperforms previous methods especially in networks with high bit error rate, by better utilizing redundant packets spread in the network.**

***Index Terms*—Coding conditions, network coding, opportunistic forwarding, wireless mesh networks.**

I. INTRODUCTION

IN RECENT years, a significant amount of research has been conducted to explore the effect of network coding in different scenarios and improve the network performance. To exploit network coding, related research mostly focuses on either inter-flow or intraflow network coding.

One of the most popular examples showing the gain behind inter-flow network coding is the X-topology in Fig. 1(a), where S_1 sends packet a to D_1, and S_2 sends packet b to D_2 through an intermediate node N. Since D_1 and D_2 are able to overhear the packets of the other flow from its source, the relay node N mixes packets of two flows and sends their combination to the network. Doing so, network coding decreases the number of required transmissions to deliver packets to their final destination and improves the performance.

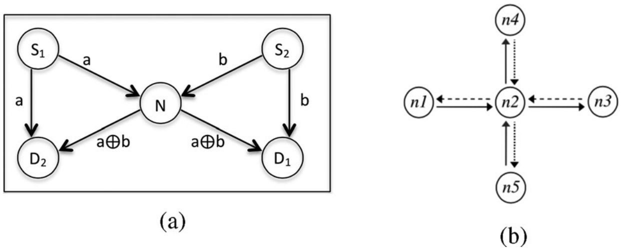


Fig. 1. Some topologies utilizing network coding. (a) X-topology. (b) Cross topology [1].

COPE [1] is one of the first methods that realize this idea in practical scenarios. Whenever an intermediate node receives packets from different flows, it encodes them if it is likely that the next-hops of the native packets combined in the coded packet are able to decode this packet and retrieve the original content. However, coding opportunities in COPE are restricted only to joint nodes that receive packets from multiple flows. Therefore, to provide more coding opportunities, COPE needs more packets to arrive at the same node. However, this traffic concentration may overload intermediate nodes, and cause longer delay, buffer overflow, and channel contention.

As a solution to this problem, BEND [2] applies network coding while trying to avoid traffic concentration. By taking advantage of the broadcasting nature of wireless networks, BEND allows all receivers of the packet, in addition to the intended next-hop specified by the routing protocol, to help in mixing and forwarding the packet if they believe they can be helpful. However, these *nonintended forwarders* (i.e., the receivers of the packet which are not specified as the next-hop on the route defined by the routing protocol, and can help in forwarding) are allowed to assist the *intended forwarder* only in forwarding received native packets. In fact, if they receive a coded packet, they just discard it, even if they were able to decode the received packet. This restriction not only limits the number of coding opportunities in the network but increases the number of retransmissions as well. The terms *intended* and *nonintended forwarders* as well as some other terms used in this research are summarized in Table I.

Furthermore,almost all inter-flow network coding methods, which mix packets within a two-hop region, follow a similar set of coding conditions to encode packets. We call this set “common coding conditions.” Based on these coding conditions,

Manuscript received April 7, 2016; revised September 15, 2016, November 15, 2016, and January 9, 2017; accepted January 19, 2017. Date of publication January 26, 2017; date of current version August 11, 2017. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Discovery program. The review of this paper was coordinated by Dr. C. Yuen.

S. Kafaie, M. H. Ahmed, and O. A. Dobre are with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John’s, NL A1B 3X5, Canada (e-mail: somayeh.kafaie@mun.ca; mhahmed@mun.ca; odobre@mun.ca).

Y. Chen is with the Department of Computer Science, Memorial University of Newfoundland, St. John’s, NL A1B 3X5, Canada (e-mail: yzchen@mun.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2017.2659539

TABLE I
DEFINITION OF SOME TERMS USED IN THIS PAPER

Term	Definition
native packet	a packet that is not combined with any other packet
coded packet	XORed of more than one native packet
intended forwarder	the designated next-hop by the routing protocol
nonintended forwarder	the neighbors of the next-hop which can help in forwarding
coding node	a node in which coded packets are generated
eligible forwarder	a node which is the neighbor of both the next-hop and the second next-hop of a packet
decoded-native packet	a native packet which was received coded and has been decoded
coding partner	each native packet encoded with other packets
common coding conditions	the conditions used by previous methods (e.g., COPE and BEND) to combine packets

given a high delivery probability between nodes, two packets are combined if the next-hop of each packet is the previous hop of the other packet or one of the neighbors of the previous hop. However, in some scenarios as shown in this research, the common coding conditions may decide incorrectly to mix some packets that cannot be decoded at the next-hops. This wrong encoding causes failures in decoding, increases the number of required retransmissions to deliver the packets and consequently decreases the network throughput.

To better utilize the broadcasting nature of wireless networks, we introduce Flexible and Opportunistic Network Coding (FlexONC), which provides more flexibility to previous methods like COPE and BEND by adding opportunistic forwarding, and allowing nonintended forwarders to help in decoding in addition to encoding and forwarding. Moreover, FlexONC proposes an additional coding condition to find coding opportunities more accurately, and designs a mechanism to merge it with the common coding conditions.

The main contributions of FlexONC are as follows:

1) More diffusion gain since more packets (i.e., coded and native packets) can be forwarded by a node other than their intended forwarder.
2) Faster packet delivery to the final destination because even if the intended forwarder does not receive the packet or cannot decode the received coded packet, some nonintended forwarders can still help.
3) More coding opportunities as nonintended forwarders are eligible to receive and probably decode coded packets and consider them as candidates to be mixed with other packets.
4) More intelligent and comprehensive encoding decisions to avoid transmitting undecodable packets in the network.

The rest of the paper is organized as follows. Related research on network coding, especially COPE and BEND, is discussed in Section II. Section III provides two examples to show the effectiveness of FlexONC. Section IV describes the objectives and challenges of FlexONC, and introduces its implementation details. Section V presents performance evaluation results and compares FlexONC with a noncoding scheme as well as other inter-flow network coding methods. In Section VI, some intrinsic features of FlexONC are discussed further. Finally, Section VII concludes the paper and provides ideas to extend FlexONC in future research.

II. BACKGROUND AND RELATED WORK

Network coding represents an innovative idea introduced by Ahlswede *et al.* [3] in 2000 to increase the transmission capacity of the network, as well as its robustness. In general, two different types of network coding can be applied, namely intraflow and inter-flow network coding. While in the former, nodes mix packets of the same flow to increase the robustness [4]–[6], in the latter packets of different flows are mixed to reach the maximum capacity of the network [1], [2], [7]. Xie *et al.* provide a survey on inter-flow network coding under both reliable links and lossy links [8].

In inter-flow network coding, an intermediate node combines two packets if the next-hop of each packet has already received the other coding partner. To keep track of the packets received by each node, two types of information are used: deterministic information and probabilistic information. Deterministic information are provided by exchanging "reception reports" among nodes, where each node's reception report contains the packets that recently have been received or overheard by the node [1]. These reception reports are usually piggybacked on data packets or broadcasted periodically.

In the absence of deterministic information (e.g., when a node does not transmit any data packet and only relies on periodic updates), probabilistic information is used to decide on encoding. In this case, if the delivery probability between nodes is greater than a threshold, two packets are combined if the next-hop of each packet is the previous-hop of the other coding partner or one of the neighbors of the previous-hop. In this research, we present scenarios where encoding decisions made based on the probabilistic information through the common coding conditions are not accurate enough and cause a significant number of decoding failures.

COPE is one of the prominent examples of inter-flow network coding. In COPE, a node combines the packets, P_1, P_2,..., P_n, with different next-hops, NH_1, NH_2,..., NH_n, when in the combined packet 1) for each next-hop there is at most one packet, and 2) for each packet P_i, all the next-hops have already received the packet except for its corresponding next-hop, NH_i. For example, let us assume that in the cross topology depicted in Fig. 1(b), for each node all nodes are in its transmission range except for the diametrically opposed node, and n_1, n_3, n_4, and n_5 are the sources of four flows intersecting at n_2. Then, n_2 can mix four packets received from all sources because each next-hop contains all other coding partners except for its intended packet. However, the improvement of throughput in COPE depends on the traffic pattern. In fact, it limits coding opportunities because coding can be accomplished only at joint nodes. As an example, if in Fig. 1(b) the sources choose a different intermediate node than n_2, all flows cannot intersect at the same node and less coding opportunities are provided by COPE.

A variety of improvements over COPE have been put forward, especially by adding opportunistic forwarding [9]. In CORMEN [10], as a network coding scheme enhanced with

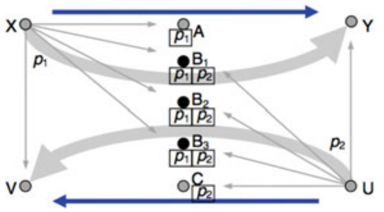


Fig. 2. Diffusion gain in BEND [2].

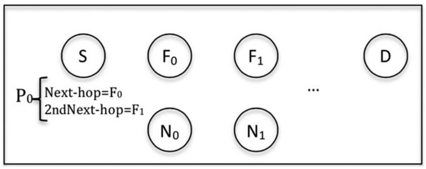


Fig. 3. In BEND, nonintended forwarders drop coded packets.

opportunistic routing, the nodes in the forwarder set are neighbors of the nodes in the shortest path to avoid diverging the path and unnecessary duplicate packets. However, similar to source routing protocols, the packet header should contain not only the forwarder set but the nodes in the shortest path as well. In addition, since the packet may not follow the shortest path, the forwarders need to keep updating it. Also, end-to-end acknowledgments are sent instead of hop-by-hop ones.

CORE [11] is also one of the first research that integrates interflow network coding with opportunistic forwarding to increase the coding opportunities in the network. In each transmission, among all neighbors of the last forwarder which are closer than it to the destination, CORE selects the node with more coding gain as the next forwarder. To prioritize the nodes with different coding opportunities, forwarding timers are used so that the node with more coding opportunities forwards its packet earlier. In addition, in CORE the packets are broadcasted without any acknowledgment and retransmission mechanism. While CORE defines a coding gain function in each node only in terms of the number of neighbors that are able to decode a coded packet, CoAOR [12] takes into account the number of flows coded in a packet and the link quality as well.

CAR [13] is another coding-aware opportunistic routing scheme that aims to maximize the number of native packets coded together in a single transmission by dynamically selecting the route based on real-time coding opportunities. In some described works, the closeness to the destination (i.e., to find the forwarding set) is calculated in terms of the geographical distance, which does not necessarily represent the quality of the path. In addition, in most of the research in this area, the maximum coding opportunities is the only factor taken into account to select the next forwarder, even if the path traveled by the node is excessively longer than the shortest path.

BEND, as another advancement of COPE, introduces a type of gain, referred to as the *diffusion gain*, which is the benefit of being able to scatter flows through multiple forwarders dynamically. In BEND, each node has three queues: Q_1 for intended native packets, Q_2 for overheard native packets, and *mixing-Q* for coded packets. A node can combine two packets if the next-hop of the first packet is the previous hop of the second packet or one of its neighbors, and vice versa.

To avoid traffic concentration in BEND, a nonintended forwarder may receive a native packet and mix and forward it on behalf of the intended forwarder. For example in Fig. 2, where A and C are the intended forwarders of the flows from X and U to Y and V, respectively, COPE cannot find any coding opportunity. On the other hand, BEND allows nonintended forwarders which can overhear packets of both flows (e.g., B_1, B_2, and B_3) to combine and forward the packets on behalf of the intended forwarders. To do so, a *second-next-hop* field is included in native packets. As such, when a nonintended forwarder receives a native packet, it can find the address of the next-hop in the *second-next-hop* field.

However for coded packets, the *second-next-hop* field does not present the correct address in a way that the packets still travel near the original route. Therefore, nonintended forwarders must drop coded packets since they do not know the address of the next-hop from the intended forwarder to the destination. To illustrate the idea, let us assume in Fig. 3 that the source S sends a packet P_0 to D. Based on the information provided by the routing protocol, it fills the *next-hop* and *second-next-hop* fields with F_0 and F_1, respectively. We assume that F_0 fails to receive the packet, and N_0 overhears it. In addition, N_0 can mix P_0 with a packet P_1 in its buffer and forward it. Based on P_0's header, N_0 sets the new *next-hop* field with the current *second-next-hop* field, F_1. However, N_0 cannot set the *second-next-hop* field in P_0 because N_0 does not know the *second-next-hop* from the intended forwarder's point of view (i.e., the *second-next-hop* from F_0). Now, if F_1 receives and decodes P_0 successfully, it can consult the routing module and find the next-hop because F_1 is the designated intended forwarder. However, if nonintended forwarder N_1 receives the coded packet, since *second-next-hop* was not set as well as N_1 was not specified in the route, it may not be able to find the correct next-hop. Thus, N_1 as a nonintended forwarder must drop coded packets.

A preliminary version of FlexONC [14] moves one step further for more diffusion gain than BEND, and allows nonintended forwarders to cooperate in receiving and forwarding not only native packets but coded packets as well. In fact, it provides the next-hop information of decoded packets to nonintended forwarders so that they are able to forward the packet to the correct next-hop toward the destination. As we explained in the previous section, by doing so, FlexONC provides more *diffusion gain* and more coding opportunities, which lead to a higher throughput in comparison to previous methods.

In this paper, we discover and address the problem related to the *common coding conditions*, and we augment the implementation of FlexONC to incorporate our solution for this problem. In addition, we further discuss FlexONC as a media access control (MAC) layer solution that not only increases the coding opportunities in the network, but also allows us to control effectively how far packets stray away from a designated shortest path. We conduct more experiments to show the efficiency of our solution by comparing FlexONC with other

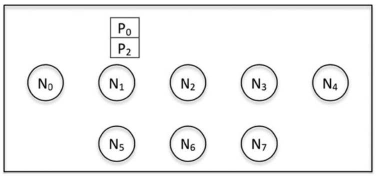


Fig. 4. Nonintended forwarders can help decoding.

schemes from different aspects such as throughput, end-to-end delay, the number of duplicate packets, the number of coding opportunities, and overall overhead and complexity.

III. FlexONC Motivating Examples

A. More Diffusion Gain

Fig. 4 presents an eight-node topology where there exist two flows from N_0 to N_4, and vice versa. In all topologies used in this research, we assume each node can receive packets only from nodes immediately next to it horizontally, vertically, or diagonally. As shown in this figure, N_1's queue contains two native packets P_0 and P_2 with different next-hops N_0 and N_2, respectively. Let us assume P_0's next-hop is P_2's previous forwarder or one of its neighbors, and vice versa. So, N_1 decides to mix these packets together, hoping that N_2 (N_0) has already received P_0 (P_2) and it can decode P_2 (P_0). Therefore, N_1 sends a coded packet $P = P_0 \oplus P_2$ to N_0 and N_2 (i.e., next-hop list in the packet header contains N_0 and N_2) while we assume N_6 overhears the packet.

In the previous methods like COPE and BEND, N_6 discards the packet immediately because either it is not the next-hop (as in COPE) or the packet is not a native packet (as in BEND). Here, we assume that N_2 does not receive the coded packet or P_0, so it cannot decode P_2, and that N_6 receives it successfully, and also can decode the packet. In such a scenario, in previous methods, after a time-out N_1, which has not heard any ACK from N_2, retransmits the packet. However, FlexONC avoids such unnecessary retransmissions, and N_6 forwards the packet to its next-hop on behalf of N_2.

In fact, FlexONC allows nonintended forwarders like N_6 to decode a received coded packet if they can, and forward it toward the final destination as long as the intended forwarder fails to do so. By doing so, since N_2 is not the only node in charge of forwarding packets, the traffic is spread in the network. That is if N_2 fails to receive or decode a packet, its role is immediately covered by N_6. This idea not only can accelerate packet delivery by removing some retransmissions but can provide more coding opportunities as well. For example, let us further assume N_6 is going to forward P_2 on behalf of N_2. If P_2 is eligible to be mixed with some packets queued at N_6, by allowing N_6 to decode and forward it, we capture more coding opportunities in N_6. However as will be described later, we provide some strategies to ensure that the nodes do not stray far away from the original route, and also to limit the number of duplicate packets in the network.

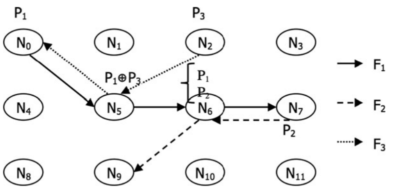


Fig. 5. Common coding conditions are not sufficient.

B. Right Coding Opportunities

Let us assume that in the grid topology provided in Fig. 5, our focus is on three specific flows: 1) F_1 with packets like P_1 from N_0 to N_7, 2) F_2 with packets like P_2 from N_7 to N_9, and 3) F_3 with packets like P_3 from N_2 to N_0. Let us further assume that N_5 transmits a coded packet from flows F_1 and F_3, $P_1 \oplus P_3$. We assume N_6, as the intended forwarder of P_1 can decode the packet successfully, but N_9 cannot decode it as N_9 cannot overhear P_3. Let us call a packet like P_1, which has been received coded by the node and then it is decoded, a *decoded-native* packet. Now, the question is that under what conditions a node (e.g., N_6) can combine a decoded-native packet (e.g., P_1) with other packets? For example, can N_6 combine packets received from N_5 and N_7? Are the common coding conditions enough to decide on encoding such packets?

Based on the common coding conditions, the combination of P_1 and P_2 at N_6 seems a valid encoding strategy because the next-hop of P_1 (i.e., N_7) is the previous hop of P_2, and the next-hop of P_2 (i.e., N_9) is one of the neighbors of the previous hop of P_1 (i.e., N_5). However, one may notice that if N_9 receives the coded packet $P_1 \oplus P_2$, it cannot decode P_2 correctly as it has only overheard $P_1 \oplus P_3$ and neither P_1 nor P_3. In fact, the problem happens because the previous hop of P_1 (i.e., N_5) sends it as a coded packet; therefore, its neighbors (e.g., N_9) do not receive P_1 natively. As a result, if N_6 encodes this decoded-native packet, N_9 cannot decode the received coded packet $P_1 \oplus P_2$.

Note that although COPE uses reception reports, in such a scenario COPE could not rely on them for encoding. Since N_9 does not send any packet, it has to send the reception reports periodically, which reduces the probability that its neighbors receive a fresh report on time. Therefore, most of the time the neighbors do not have deterministic information required for encoding and would need to guess based on the delivery probability between nodes. Hence, if the delivery probability between different nodes is high, in COPE, N_6 will encode P_1 and P_2. To show the severity of the issue, we ran simulations, using a simulation version of COPE in ns-2, to decide on encoding of the packets in the topology depicted in Fig. 5.

Fig. 6 presents the number of coded packets received by N_6 (i.e., coded@6), the number of coded packets received by N_9 (i.e., coded@9), as well as the number of coded packets that N_9 cannot decode (i.e., failure@9) because of the explained issue. As shown in this figure, by decreasing the interarrival time (i.e., increasing the arrival rate), the length of the transmission queue as well as the coding opportunities at nodes increase. Therefore, the probability that an encoded packet received and decoded by N_6 (i.e., a decoded-native packet) can be encoded again

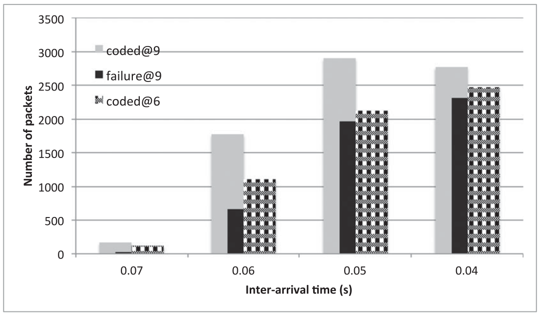


Fig. 6. Decoding failure of COPE by applying the common coding conditions.

increases, which in this scenario causes the explained issue and consequently increases decoding failures at N_9.

This example and simulation results show that the common coding conditions are not enough, and more restrictive coding conditions are required to address the issue stated here. Therefore, we not only provide cooperative forwarding for native and coded packets, but also address this issue by proposing an additional rule to restrict the common coding conditions.

IV. DESIGN DETAILS

As described earlier, the idea behind FlexONC is to have backup nodes to decode and forward a packet in case where the intended forwarder fails, either due to unsuccessful reception of the packet or lack of required packets in the buffer to decode the original packet. In addition, FlexONC provides more comprehensive coding conditions and a mechanism to detect right coding opportunities and avoid undecodable encodings. In this section, we first discuss some of the challenges that FlexONC addresses. Then, we describe in detail the responsibilities of the sender and receiver of a coded packet to realize these ideas, and address these challenges.

A. Objective and Challenges

FlexONC should avoid unnecessary changes to the standard MAC protocols, and be as simple as possible to be feasible in real scenarios. Moreover, it should be compatible with different routing protocols despite few modifications. To realize such compatibility, while having more flexibility and accuracy in forwarding and coding, FlexONC should address the following questions:

1) How to select the nodes that can help the intended forwarder to forward packets: In other words, how should we decide which nodes are eligible for packet forwarding? For example, in Fig. 4, when N_1 sends the packet, N_5, N_2, and N_6 may receive it, but are they good candidates to forward the packet?
2) How to limit the number of duplicate packets: Since more nodes cooperate to move packets toward the destination, their imperfect collaboration may cause a significant number of duplicate packets travelling in the network leading to unnecessary contention and collision. Some mechanisms are required to control duplicate packets.
3) How to provide flexible forwarding but not too far from the specified route: Although in FlexONC, like BEND, packets may not follow the exact route specified by the routing protocol, we need to keep them around the determined route. To do so, BEND uses the *second-next-hop* field in native packets. However, as we described earlier, it is not applicable to coded packets at nonintended forwarders. For example, in Fig. 4 when N_6 receives the coded packet, even if it can decode P_2, it does not know the address of the next-hop from N_2 toward the destination. Thus in FlexONC, we need a new approach for nonintended forwarders to find the correct address of the next-hop.
4) How to propose a complete set of rules to combine packets: As illustrated in Section III-B, the common coding conditions used in other inter-flow network coding methods are not accurate enough to recognize right coding opportunities in some scenarios, and may lead to decoding failures. The question is how to establish a complete set of rules to correctly decide on mixing the packets of flows which are decodable at the next-hop?

We address all these aspects in the next sections.

B. Decoding and Forwarding Strategy

In FlexONC, nodes in the network are in promiscuous mode, and store all received and overheard packets in a buffer, called *coding buffer*. Each packet is kept there for a period of time, long enough that the node can use these packets to decode the received coded packets. In case of successful decoding, the receiver sends an ACK while a NACK (i.e., negative acknowledgement) signals failure in decoding. In terms of forwarding, native packets are only sent by intended forwarders. A nonintended forwarder may forward a packet on behalf of an intended forwarder if the nonintended forwarder can provide more coding opportunities.

In FlexONC, although packets may not follow the exact route specified by the routing protocol, they travel near it and do not stray too far away. Thus, when a nonintended forwarder forwards the packet on behalf of the intended forwarder, it should send it to the next-hop toward the destination from the intended forwarder's point of view. For example in Fig. 4, when N_1 sends the coded packet $P = P_0 \oplus P_2$, N_0, N_5, N_2, and N_6 may receive the packet. If N_2, which is the intended next-hop for P_2, fails to receive the packet successfully, and if one of the nonintended forwarders (e.g., N_5, N_0, N_6) wants to forward it, they need to know the address of the next-hop from N_2 toward the destination (not from themselves), which is N_3 in this example.

Since the *second-next-hop* field in BEND cannot solve this problem, instead of adding this field to the packet header, in FlexONC, the routing protocol is enhanced such that each node also maintains forwarding tables of all its neighbors. As such, when for example N_6 forwards P_2 on behalf of N_2, it knows the address of the next-hop from N_2 toward the destination, and simply sends the packet to it.

C. Receivers in FlexONC

Since every node in the vicinity of the sender can receive the packet, we classify the receivers of a packet in two groups, intended forwarders and nonintended forwarders. As summarized in Table I, an intended forwarder is a node whose address has been specified in the packet header as the next-hop of the packet by the routing protocol. On the other hand, nonintended forwarders are the nodes that are in the neighborhood of the next-hop and can help it in forwarding packets.

When a sender transmits a coded packet, all of its neighbors may receive it. However, every node that receives the packet is not necessarily eligible to forward it. In addition, if all eligible nodes were to forward the same packet, that would be a waste of the network bandwidth as well as a source of collision. We need a method to choose and prioritize eligible forwarders.

A node is an *eligible* nonintended forwarder if it is not only the neighbor of the sender, but also a neighbor of both next-hop and the second next-hop of a coding partner. Following this rule ensures that a packet would travel correctly toward its final destination, even if it is forwarded by a different node than its next-hop. In the rest of this paper, we use the term "nonintended forwarder" to refer to "eligible nonintended forwarders."

If an intended forwarder (e.g., N_2 in Fig. 4) receives a coded packet and can decode the packet, it simply replies with an ACK. However, if it cannot decode the packet, it sends a NACK instead. In FlexONC, ACKs and NACKs contain the address of their sender (i.e., the transmitter of ACK/NACK) instead of the receiver, the same as in BEND. If nonintended forwarders (e.g., N_6) hear the ACK, they realize that the intended forwarder has decoded the packet successfully and does not need their help.

In FlexONC, when a node like N_6 in Fig. 4 receives a coded packet, it first looks for its address in the next-hop list. If it cannot find its address, clearly it is not the intended forwarder for any coding partner in the coded packet. Therefore, N_6 searches for a native packet in the coded packet that 1) its intended forwarder (e.g., N_2 for P_2 in Fig. 4) is N_6's neighbor, 2) its next-hop from the intended forwarder (e.g., N_3 for P_2 in Fig. 4) is N_6's neighbor, and 3) it is decodable by N_6. Based on these criteria, in Fig. 4, although when N_1 sends the coded packet P, N_0, N_5, and N_6 as well as N_2 may receive the packet, N_0 is not eligible to forward P_2 due to the first criterion. Furthermore, N_5 is not qualified for the second criterion, and therefore N_6 is the only nonintended forwarder which can send P_2 on behalf of N_2 if it can decode it.

However, a nonintended forwarder should not forward a packet immediately after decoding it because the intended forwarder may forward the packet itself and would not need the nonintended forwarders' help. In addition, if there are more than one eligible nonintended forwarder, an ordering among them is required to avoid the transmission of more than one ACK to the packet sender. Due to this reason, in FlexONC the sender adds the index of all eligible nonintended forwarders to the packet header.[1] Specifically, when a nonintended forwarder receives a coded packet, it sorts the list of indexes (i.e., all nonintended forwarders), gives the first priority to the intended forwarder of the decoded packet, and considers its index in the sorted list as its rank. Then, it sets a timer and waits for an ACK from any node with a higher rank. If it does not hear any ACK after time-out, it is likely that none of the nodes with a higher rank has received and can forward the packet, so it is its turn to send the ACK back to the sender, mixes possibly the decoded packet with other packets in the queue, and forwards it. Fig. 7 presents the flowchart for receivers of a coded packet in FlexONC.

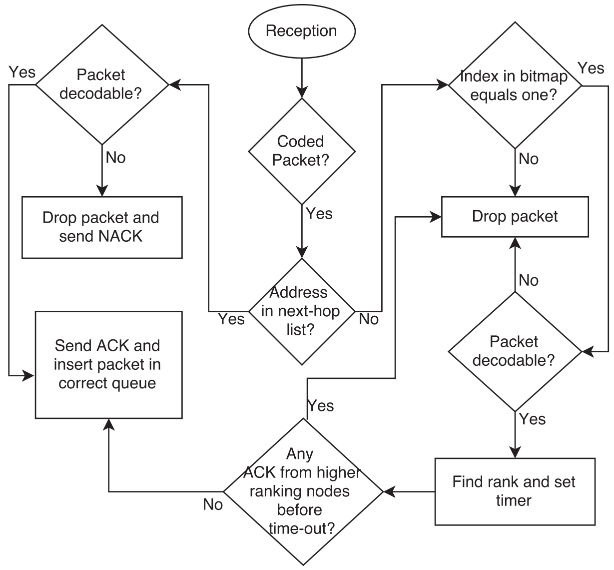


Fig. 7. Flowchart for receivers of coded packets in FlexONC.

D. Senders in FlexONC

When a node sends a coded packet, it adds the list of the next-hops of all coding partners to the packet header. Thus, when each next-hop receives the packet, it does not send the acknowledgement (either ACK or NACK) immediately, but after some time proportional to its position in the next-hop list as well as the transmission and propagation time of the acknowledgement. For example, if a node transmits the combination of three packets with the next-hops N_1, N_2, and N_3, after receiving the coded packet, N_3 waits for a certain amount of time to ensure that N_1 and N_2 have sent their packet acknowledgements, and then N_3 sends back ACK/NACK.

Furthermore, the sender detects all eligible nonintended forwarders of a coded packet, and adds a bitmap to the packet header where each bit represents one of the nodes in the network (as discussed in Section VI-G, the overhead introduced by adding this bitmap is less than a few bytes). If the node is an eligible forwarder, the corresponding bit is set to 1, otherwise the bit keeps the default value which is 0. We assume that each node is represented with a unique index known by all other nodes, and each node ranks eligible nonintended forwarders based on their indexes.

In FlexONC, the fields in the packet header of native packets do not change. However, the MAC-layer header of coded packets includes some additional information, such as the number of

---

[1] We assume that all nodes in the network agree on the same numbering system which represents each of them with a unique index known by all other nodes.

frame control	duration	code-len	MAC-dest [code-len]	MAC-source	pkt-id [code-len]	bitmap

Fig. 8. MAC header for coded packets.

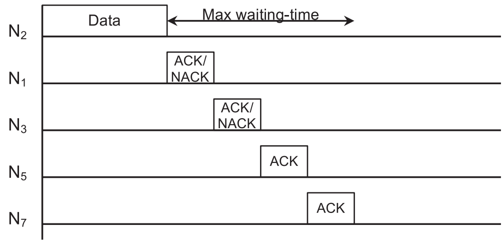


Fig. 9. Time-window dedicated to different nodes to send back the acknowledgment, where in the topology depicted in Fig. 4, N_2 transmits a coded packet to the next-hops N_1 and N_3, and N_5 and N_7 are nonintended forwarders.

coding partners, the bitmap, and the address of the next hop and the packet-id of all coding partners as presented in Fig. 8. Note that we keep the original format of the upper layers' headers, and the *XOR* of the coding partners is added to the MAC dataframe as payload.

Since the sender stores the forwarding table of its neighbors, it can check which neighbors are eligible nonintended forwarders. Doing so, the sender can calculate its maximum waiting time for receiving an ACK which is proportional to the number of the next-hops (i.e., intended forwarders) and eligible nonintended forwarders of coding partners. It is obvious that when a sender sends a combination of n packets, it should wait to receive n ACKs. Thus, its waiting time before time-out is more than when it transmits a native packet. In FlexONC, because more nodes can help in decoding and forwarding a packet, if the sender does not hear an ACK from the intended forwarder, there is still a chance that it receives the ACK from a nonintended forwarder. Therefore, the sender should wait a little longer before it retransmits the packet. As such, in FlexONC the waiting time of the sender for coded packets is calculated in terms of the number of both coding partners and eligible nonintended forwarders.

To illustrate the idea in more details, let us assume that in Fig. 4, N_2 mixes two native packets and forwards the coded packet to the next-hops N_1 and N_3 (i.e., N_1 and N_3 are the intended forwarders of these two packets), while N_5 and N_7 are eligible nonintended forwarders specified in the bitmap. Fig. 9 shows the maximum waiting time at the sender N_2 after transmitting the data packet and the time-window dedicated to the intended and nonintended forwarders to reply if they need. Note that the intended forwarders reply by an ACK after successful decoding and send a NACK after decoding failure. In addition, a nonintended forwarder replies by an ACK only if decoding is successful and no ACK was heard from neither the corresponding intended forwarder nor higher ranking nonintended forwarders.

When the sender receives an ACK for a packet, it removes the packet from its transmission queue; it may still keep it in the coding buffer for decoding purposes. On the other hand, when the sender receives a NACK for the sent packet, it keeps waiting until either time-out or receiving an ACK for the same packet. In the case of time-out for native packets, the sender retransmits the same packet if the number of transmissions does not exceed the maximum retransmission count. However, for coded packets, if the node receives ACKs or NACKs for none of the coding partners, it retransmits the same coded packet. Otherwise, it inserts the coding partners which are not ACKed in the transmission queue.

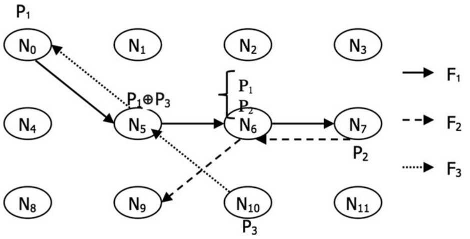


Fig. 10. RecodingRule, sufficient but not necessary.

E. Encoding Strategy

As explained earlier to decide on encoding packets, the majority of encoding methods, within a two-hop region, use a similar coding structure called *two-hop coding structure* [8] with the same coding conditions [1], [2], [15]–[18]. Based on these *common coding conditions*, node N can combine two packets P_1 and P_2 if:

1) The next-hop of P_1 is the previous hop of P_2 or one of its neighbors.
2) The next-hop of P_2 is the previous hop of P_1 or one of its neighbors.

However, as illustrated in Section III-B in some scenarios such as Fig. 5, these coding conditions are not sufficient. In fact, the issue happens because in the common coding conditions, it is assumed that all the neighbors of the previous hop (e.g., N_5) are able to decode the coded packet sent by it (e.g., $P_1 \oplus P_3$). However, this is not necessarily a valid assumption as some of these neighbors (e.g., N_9) may not be able to do so. To address this issue, we add an additional condition to the common coding conditions as follows.

RecodingRule—To combine a decoded-native packet (i.e., a packet received as a coded packet from its previous hop and has been decoded) with other packets (i.e., recode the packet), the node does not check the neighborhood of the previous hop of the packet. In fact, if P_1 is a decoded-native packet the common coding conditions should be modified as follows:

1) The next-hop of P_1 is the previous hop of P_2 or one of its neighbors.
2) The next-hop of P_2 is the previous hop of P_1.

RecodingRule is sufficient but may not always be necessary. That is, although it avoids misleading coding opportunities and decoding failures in the scenario depicted in Fig. 5, in some other scenarios it limits the number of right coding opportunities in the network. As an example, let us describe the effect of our *RecodingRule* on the scenario presented in Fig. 10. In this figure, the route of flow F_3, in comparison to Fig. 5, has changed so that N_9 can overhear the packets of this flow. Now, N_9 overhears P_3

from N_{10}, and $P_1 \oplus P_3$ from N_5. As a result, we do not need to apply *RecodingRule*, and N_9 can decode $P_1 \oplus P_2$ received from N_6 successfully.

Therefore, *RecodingRule* should be intelligently used only in cases that the interaction between flows is so that the common coding conditions may provide misleading coding opportunities. This type of encoded packets cannot be decoded in the next-hop, and the sender will receive a NACK for it. Thus, we propose a solution called *SwitchRule* to decide properly on applying *RecodingRule* on different flows at different nodes. In fact, *SwitchRule*, based on the received NACKs for each flow at each node, decides to switch back and forth to use and not to use *RecodingRule*. Note that *SwitchRule* only needs to be applied at the flow-granularity, not the packet-granularity.

At the beginning, every node uses the common coding conditions to encode packets. However, when each node combines a decoded-native packet P_1, with another packet P_2, if the next-hop of P_2 is not the previous hop of P_1 but one of its neighbors, P_1 is tagged as a *suspect* packet. This means we are suspicious that decoding failure may happen because the next-hop of P_1's partner (i.e., P_2) may have not overheard the suspect packet P_1. Each node keeps track of the number of NACKs received for the partners of suspect packets of each flow. If the number of NACKS for a flow is greater than a threshold, the node applies *RecodingRule* for the rest of the packets of that particular flow. This means the node will not combine a decoded-native packet of that flow with any other partner if the next-hop of the partner is not the previous-hop of the decoded-native packet.

Furthermore, a node will switch back to not using *RecodingRule* whenever it hears packets of a new flow or it does not hear any packet from a flow anymore. To implement the latter case in *SwitchRule*, each node set a timer for each flow. If the timer of a flow times-out before receiving a new packet of that flow, the node switches back to the common coding conditions for all flows. The waiting time before the time-out is several times of the estimated interarrival time of the packets of the flow. The interarrival time of each flow is estimated using a weighted-average over the previous average and the latest measured interarrival time. Fig. 11 presents the pseudocode of the *SwitchRule*'s mechanism.

```
Initialization:
for each flow $F_i$
    NACK[$F_i$]=0
    RecodingRule[$F_i$]=false
To encode a packet:
if P is decoded-native
    if RecodingRule[F(P)]
        apply RecodingRule
    else
        apply common coding conditions
        if P is combined with P'
            if NH(P') ∈ ng(PH(P))
                tag P as suspect
After receiving an ACK/NACK:
if a NACK is received for P'
    if its partner P was tagged as suspect
        NACK[F(P)]= NACK[F(P)]+1
        if NACK[F(P)] > NACK_th
            RecodingRule[F(P)]=true

if packet P of flow F is sensed
    if the node is a neighbour of NH(P)
        MIAT[F]=0.5×MIAT[F] + 0.5×IAT[F]
        Set flow's timer for $\alpha$×MIAT[F]
if a flow's timer times-out or a new flow is sensed
    for each flow $F_i$
        NACK[$F_i$]=0
        RecodingRule[$F_i$]=false
```

Fig. 11. Pseudocode of *SwitchRule*. The number of NACKs received for flow F is stored in NACK[F]. NH(P), PH(P), and F(P) denote the next-hop, the previous-hop, and the flow of P, respectively. ng(N) represents the set of neighbors of node N. IAT[F] and MIAT[F] denote the interarrival time and the mean interarrival time of flow F. The timer for flow F is set to α times of MIAT[F], where $\alpha > 1$.

TABLE II
INFORMATION AVAILABLE AT NODES IN DIFFERENT SCHEMES

Information	Non coding	COPE	CORE	BEND	FlexONC
next-hop	√	√		√	√
second next-hop				√	
neighbors' forwarding info					√
forwarder set			√		√
node's geo-position			√		

F. How to Limit the Number of Duplicate Packets?

Although FlexONC aims to eliminate duplicate packets by prioritizing nonintended forwarders and making the sender wait for their ACKs, duplicate packets may still exist in the network, due to various reasons such as lack of perfect synchronization. For example, a nonintended forwarder may not hear the ACK sent by the intended forwarder or higher ranking nonintended forwarders, and transmit the packet unnecessarily. Therefore, FlexONC relies on more strategies to control the number of duplicate packets in the network.

First, after receiving an ACK for a given packet-id, if the node finds a packet with the same packet-id in its transmission queue that the sender of the ACK is the next-hop of the packet or one of corresponding eligible nonintended forwarders, the node drops the packet (i.e., the packet has already been received by down stream nodes). Second, in FlexONC each node stores a limited number of received ACKs, and if it receives a packet, it searches this ACK list. If it finds an ACK for the same packet sent by its next-hop or one of its eligible nonintended forwarders, it also drops the packet.

V. PERFORMANCE EVALUATION

We use the Network Simulator (ns-2) to compare the performance of FlexONC, with and without *RecodingRule*, against the noncoding scheme, a simulation version of COPE as a prominent research on network coding, and two opportunisitc forwarding schemes in network coding (i.e., BEND and CORE).[2] Table II

---

[2] Note that in all simulations, IEEE 802.11 [19] is selected as the data link layer signaling method.

summarizes the type of information provided at nodes in different schemes. The rest of this section describes the experiment scenarios as well as the performance results in three different topologies.

A. Settings

To study the performance under different link qualities and packet loss probabilities in our simulation, bit error rate (BER) is added to the physical layer. In fact, even if the signal strength of a received packet is higher than reception threshold, the packet may still be dropped with a probability calculated in terms of BER. BEND and CORE also use a similar physical layer model. The channel propagation used in ns-2 is a two-ray ground reflection model [20], and the maximum transmission range is 250 m. The data rate is fixed to 1 Mb/s. The sources, in our simulation scenarios, send constant bit rate (CBR) data flows with a datagram size of 1000 bytes. In addition, we use destination-sequenced distance-vector (DSDV) [21] as the routing protocol and apply a few minor changes so that each node can obtain forwarding tables from its neighbors.

We compare the performance of FlexONC with other baselines in several scenarios. In the first part, we use scenarios in which common coding conditions are enough to encode the packets in all methods, including FlexONC. Then in the second part, we present the performance of different methods in scenarios where *RecodingRule* is required to avoid erroneous encoding causing decoding failures.

B. Performance Under Common Coding Conditions

To investigate the performance of FlexONC in comparison to BEND, CORE, COPE, and the noncoding scheme, we test them in different scenarios and compare their throughput as well as the throughput gain of FlexONC over the baselines for different BERs in two topologies. First, we compare them using a simple 8-node topology shown in Fig. 4, and then we use a 5 × 5 grid topology as a more general case. In both topologies, different flows have been selected so that in most cases the common coding conditions are enough and we compare all methods using the same coding conditions (i.e., common coding conditions).

1) 8-Node Topology: In the 8-node topology presented in Fig. 4, two flows in opposite directions transmit packets from N_0 to N_4 and vice versa. Since the distance between adjacent nodes in both X and Y axes is 150 m, each node can receive packets only from nodes immediately next to it horizontally, vertically, or diagonally (e.g., N_1 can hear from N_0, N_5, N_2, and N_6). The interarrival time of CBR flows in these scenarios is 0.07 s and its duration is 150 s.

In this topology, for each intended forwarder except for the destination, there exists at least one nonintended forwarder that can help the intended forwarder and forward packets when the intended forwarder fails to do so. Regarding CORE, it means that at least two nodes can be chosen in the forwarder set of each packet. Fig. 12 presents the throughput of BEND, CORE, COPE, noncoding, and FlexONC for three lowest BERs in our experiments.

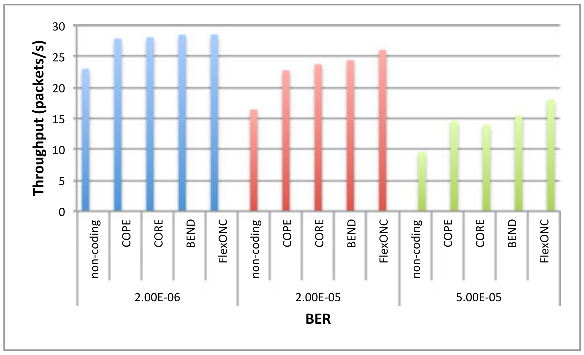


Fig. 12. Throughput of different methods in 8-node topology for different BERs.

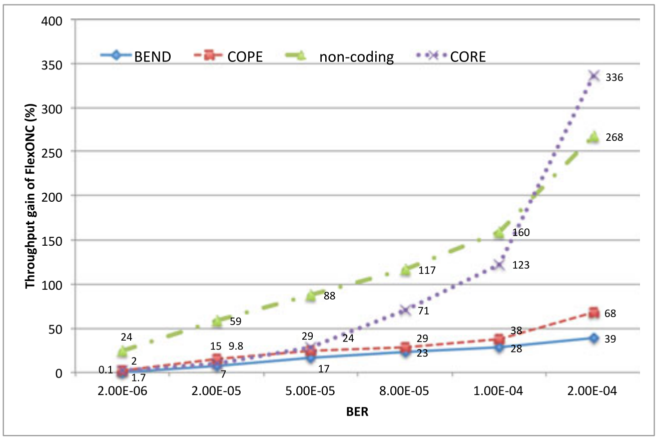


Fig. 13. FlexONC's gain over other methods in 8-node topology.

We observe that when BER $= 2 \times 10^{-6}$ (i.e., the network condition is almost perfect), most transmitted packets are received by the intended forwarders successfully. Therefore, there hardly exists an opportunity for nonintended forwarders to decode and forward a packet on behalf of the intended forwarder. It is obvious that in such a situation, FlexONC does not show its real power and its throughput is close to BEND. However, as the BER increases, more opportunities for nonintended forwarders are provided and FlexONC's gain over other methods increases significantly.

Furthermore, Fig. 13 presents the performance gain of FlexONC over BEND, CORE, COPE, and noncoding for six different BER levels, which corroborates our observation. In particular, by increasing the BER, FlexONC becomes more powerful in comparison to the baselines, and its throughput gain increases. The throughput gain of FlexONC over each baseline is calculated as:

$$\text{throughput gain} = \frac{Tr(\text{FlexONC}) - Tr(\text{baseline})}{Tr(\text{baseline})} \times 100 \quad (1)$$

where $Tr(x)$ denotes the calculated throughput for scheme x.

As shown in these figures, although at lower BER, CORE's performance is very close to FlexONC's, in lossy networks FlexONC outperforms CORE due to the following reasons. First, in this topology with a small forwarder set, at high BERs many packets are lost without being received by any forwarder.

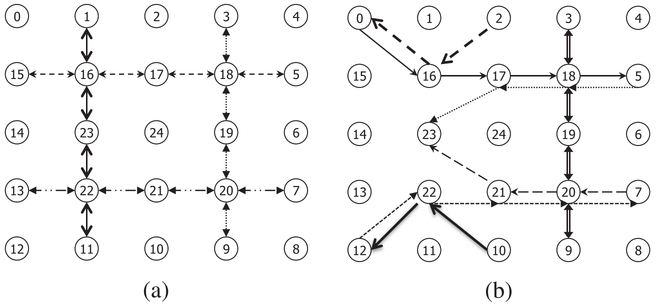


Fig. 14. 5 × 5 grid topology.

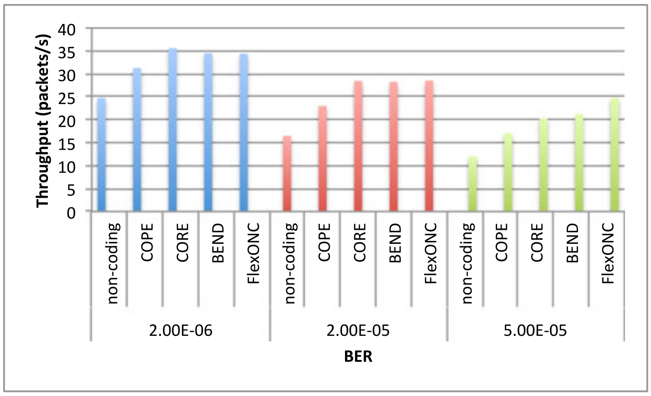


Fig. 15. Throughput of different methods in the grid topology for different BERs.

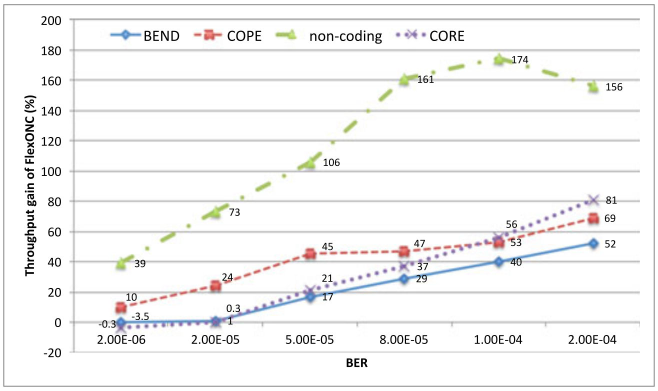


Fig. 16. FlexONC's gain over other methods in the grid topology.

Second, in CORE the packets are broadcasted without any retransmission mechanism to compensate for packet loss.

2) Grid Topology: To investigate the performance of FlexONC in a general topology, we test it in a 5 × 5 grid, where again the distance between two adjacent nodes is 150 m. Eight different flows with an interarrival time of 0.1 s and duration of 150 s transmit packets between Row 2 and Row 4, and also Column 2 and Column 4 of the grid, as shown in Fig. 14(a).

The performance results depicted in Figs. 15 and 16 again show that at nontrivial BER levels, FlexONC almost always outperforms other methods. In perfect network conditions (BER $= 2 \times 10^{-6}$), CORE performs slightly better than FlexONC because there is no intended forwarder in CORE, and it distributes packet transmissions more evenly than FlexONC among possible forwarders. However, as explained earlier, in lossy environments CORE cannot benefit from opportunistic forwarding and network coding as much as FlexONC due to the lack of any retransmission mechanism, especially in such multihop routes (i.e., each node should pass at least four hops to be delivered to the destination).

In addition, one may notice that by increasing the BER, the throughput gain of FlexONC over CORE increases faster in the 8-node topology in comparison to the grid topology. In fact, the larger forwarder set in the grid topology decreases the probability of packet loss in each transmission.

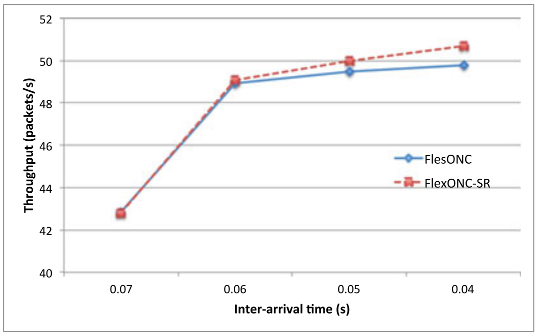


Fig. 17. Effect of SwitchRule on the throughput of FlexONC in the topology depicted in Fig. 5.

C. *Performance Under SwitchRule*

We investigate the effect of *SwitchRule* on the performance of FlexONC in two different scenarios, where at some nodes the common coding conditions may not be sufficient to combine the right packets. First, we compare the throughput of FlexONC in the topology depicted in Fig. 5 with different interarrival times for cases that the *SwitchRule* functionality is OFF (i.e., only common coding conditions are used) and is ON. We call the latter version of FlexONC, which uses *SwitchRule*, *FlexONC-SR*. In this scenario, three flows transmit their packets for 150 s, BER equals 2×10^{-6}, and in FlexONC-SR, the NACK threshold to start applying *RecodingRule* is equal to 5.

As shown in Fig. 17, although at lower packet arrival rates (i.e., longer interarrival time) the performance of FlexONC and FlexONC-SR is close, at higher arrival rates FlexONC-SR can benefit from *SwitchRule* to avoid decoding failures and more retransmissions to deliver packets to the destination. As an evidence, Fig. 18 presents the number of retransmitted packets and the number of received NACKs in both FlexONC and FlexON-SR. As explained in Section III-B, the common coding conditions may wrongly decide to combine the decoded-native packets with other packets, and obviously at higher arrival rates, more decoded-native packets are generated (i.e., the probability that the same packet is encoded at different nodes increases).

We also compare the performance of FlexONC-SR with other baselines in a 5 × 5 mesh network with eight different CBR flows, as depicted in Fig. 14(b), with duration of 150 s. As shown in Fig. 19, although BER is very small ($BER = 2 \times 10^{-6}$), FlexONC outperforms other schemes. Moreover, when the

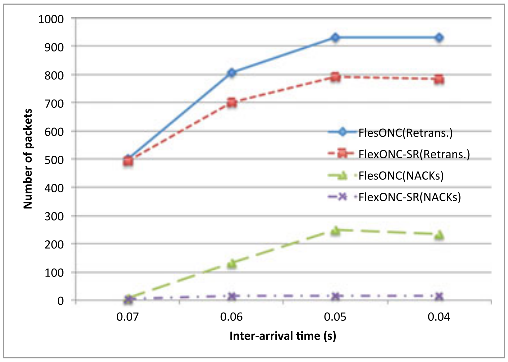


Fig. 18. Number of retransmissions and received NACKs with and without applying SwitchRule in FlexONC.

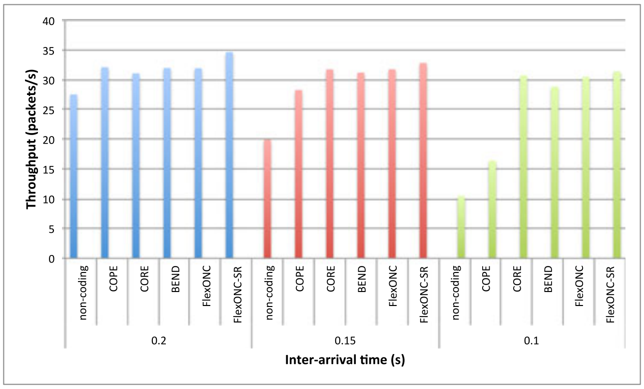


Fig. 19. Throughput of different methods in the topology depicted in Fig. 14(b).

functionality of *SwitchRule* is added to FlexONC (i.e., FlexONC-SR), its throughput is even further boosted.

VI. DISCUSSION

A. Routing Protocol

In our experiments, we selected DSDV as the routing protocol for its well-known behavior. Moreover, it is a distance-vector approach that makes fewer assumptions about the routing information in comparison to source routing protocols. Therefore, if FlexONC works well with DSDV, it will work with source routing protocols as well. As a matter of fact, choosing DSDV as the routing module does not lose generality of our scheme in a stationary mesh network. We believe choosing any other routing protocol would not make a big difference in FlexONC's performance gain, as long as the routing protocol can be modified in a way that each node contains forwarding information for its neighbors.

B. End-to-End Delay

On one hand, FlexONC decreases the delay in forwarding packets and increases the throughput by avoiding packet retransmission when an intended forwarder fails to decode the coded packet, and a nonintended forwarder alternatively passes the packet toward the destination. On the other hand, when more nodes have the responsibility of passing the packet further to the destination, in case of retransmissions, the sender should wait longer for an ACK before it retransmits the packet, and this longer waiting time means longer delay which may lead to a lower throughput.

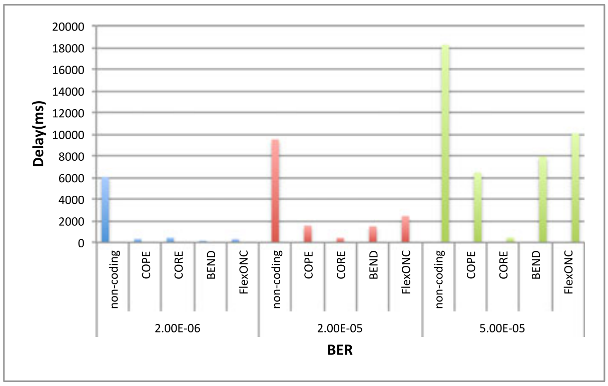


Fig. 20. End-to-end delay of different methods in 8-node topology for different BERs.

Therefore, we face a tradeoff here. While the maximum waiting time of the sender is proportional to the number of eligible forwarders, the gain of FlexONC is also related to the number of neighbors of the sender (i.e., more precisely, eligible nonintended forwarders), as well as the probability of intended forwarder's failure in receiving or decoding a coded packet, which is in turn affected by the packet loss probability and BER in the network. The performance result showed that even for a very low BER when the intended forwarder itself can decode and forward the majority of received coded packets and FlexONC does not have much chance to be applied, its performance is comparable to BEND's performance or even better.

Fig. 20 shows the average end-to-end delay of delivered packets in different methods, for the scenario described in Section V-B1. While the noncoding scheme has the highest average end-to-end delay, the delay in FlexONC is slightly longer than BEND. As explained earlier, the most important reason of this longer delay is that the sender of coded packets in FlexONC waits longer to receive an ACK than in BEND. Therefore, if the packet transmission fails and no ACK is received, BEND's timer, for anticipated ACKs, usually expires earlier than FlexONC's, leading to a faster retransmission in BEND, which can reduce its average end-to-end delay in comparison to FlexONC.

In addition, one may notice that in CORE the end-to-end delay does not vary much over different BERs. While at lower BERs, CORE's delay is longer than that of other coding schemes, at higher BERs its delay is significantly shorter than that of other protocols. The main reason of this shorter and almost constant delay in delivery is the lack of any retransmission mechanism; any packet either is delivered by one transmission or is dropped.

As shown in Fig. 20, the delay in the noncoding scheme is significantly higher than other methods. The main reason is that coding enables free-riding. In other methods, more than one packet can be combined and sent simultaneously, which means that packets can free-ride on other packets. Therefore, the packets are forwarded faster. In addition, this decreases the queue

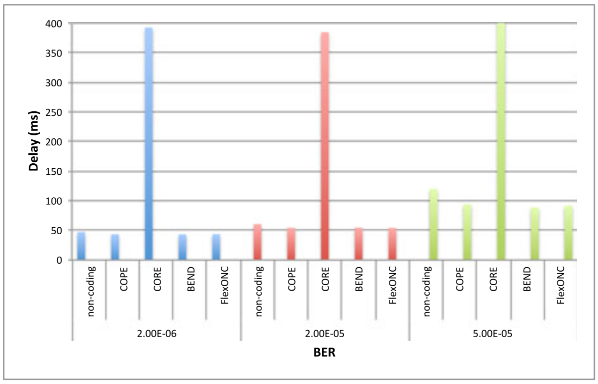


Fig. 21. End-to-end delay of different methods in 8-node topology for different BERs with less CBR traffic.

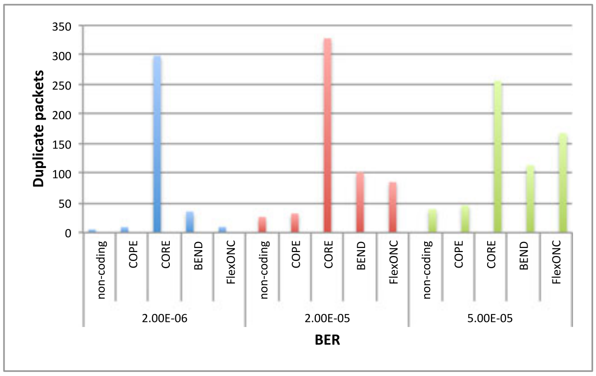


Fig. 22. Duplicate packets of different methods in 8-node topology for different BERs.

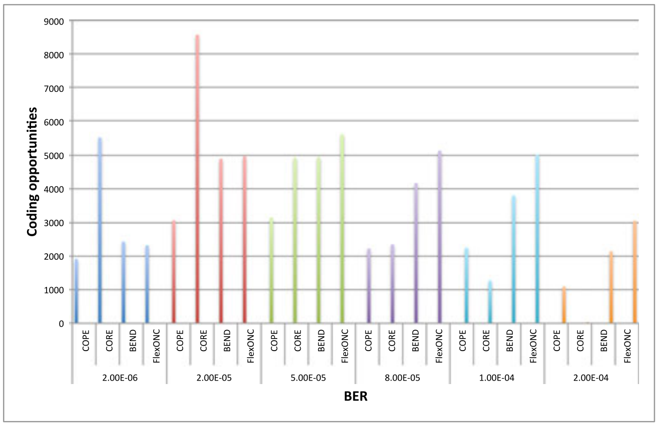


Fig. 23. Coding opportunities in different methods in 8-node topology for different BERs.

length at nodes, causing shorter waiting time and consequently shorter delay.

To verify this explanation, we repeat simulations with less CBR traffic with the interarrival time of 0.15 s (instead of 0.07 s). By increasing the interarrival time, less packets are injected to the network per second, which reduces the probability of having more than one packet in the queues, and in turn, creates less coding opportunities at nodes. The results are shown in Fig. 21, where the delay in noncoding is comparable to the other methods, as the coding schemes provide less free-riding opportunities for the packets.

Furthermore, while this figure justifies the almost constant end-to-end delay in CORE over different BERs, it also shows that the delay in CORE is significantly longer than that of other methods. As mentioned earlier, in this scenario with a small packet arrival rate, the coding opportunities are rare in the network, and most packets are sent natively. To provide higher priority for coded transmissions in CORE, the native packets are delayed before transmission; therefore, forwarding a large number of native packets in this scenario increases the end-to-end delay significantly.

C. Duplicate Packets

As explained in [2], since in BEND more nodes cooperate in forwarding packets toward the final destination, it is prone to generating more duplicate packets in case of imperfect collaboration among nodes. The situation in FlexONC could seem even more severe, as it allows nonintended forwarders to cooperate in more ways (i.e., forwarding of not only received native packets, but also received coded packets). To control duplicate packets in FlexONC, we introduced some mechanisms in Section IV-F.

Fig. 22 shows the number of duplicate packets generated by different methods. As shown in this figure, the largest number of duplicate packets are generated at CORE, as nodes should only rely on overhearing other transmissions to avoid duplicate packets. In addition, while the number of duplicate packets in BEND is higher than noncoding and COPE, FlexONC is able to control the number of duplicate packets, especially at lower BERs. The reason could be related to the additional mechanisms introduced in FlexONC to control the number of duplicate packets. However at higher BER $= 5 \times 10^{-5}$, there are more duplicate packets in FlexONC than in BEND because these mechanisms are highly susceptible to the reception of ACKs and at higher BERs the probability of losing ACKs increases.

D. Coding Opportunities

As shown in Fig. 23, at lower BERs the code opportunities at CORE are more than that of FlexONC. However, at higher BERs, FlexONC provides more coding opportunities than other schemes. One may notice that, by increasing BER, first coding opportunities in all methods increases. The reason is that, due to a greater need for retransmission, packets stay longer in the queue and the chance of combining them with the packets of other nodes increases, leading to more coding opportunities. On the other hand, when BER further increases, the number of retransmissions increases significantly; therefore, the probability of generating new coding opportunities decreases. That is why for BERs higher than 5×10^{-5}, the coding opportunities in the networks drops. In CORE, although there is no retransmission, at higher BERs and in this topology many packets can not go further than one or two hops, which decreases the number of packets in nodes' queues as well as the number of coding opportunities.

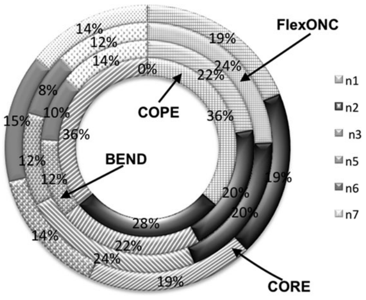


Fig. 24. Distribution of coding opportunities at different nodes in different methods in 8-node topology.

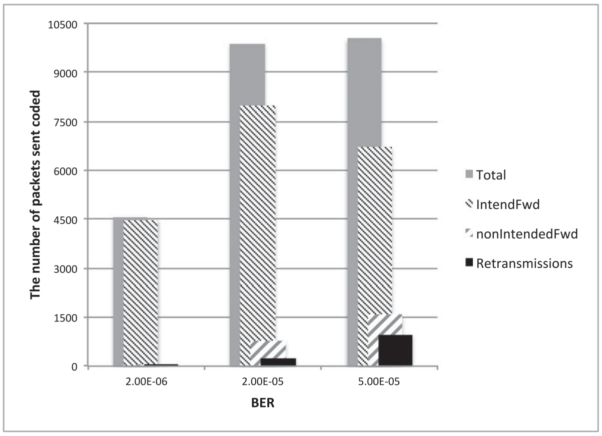


Fig. 25. What happens to coded packets when BER changes.

To show the distribution of coding opportunities at different nodes, we run simulations using the topology depicted in Fig. 4 and the scenario explained in Section V-B1, but the route between N_0 and N_4 is fixed through N_1, N_2, and N_3 for COPE, BEND, and FlexONC (i.e., the intended forwarders are N_1, N_2, and N_3). As shown in Fig. 24, coding opportunities in COPE are restricted to the intended forwarders; however, other coding schemes use nonintended forwarders (i.e., N_5, N_6, and N_7) to accelerate packet forwarding and provide more coding opportunities. In addition, since in CORE there is no intended forwarder, and possible forwarders are prioritized only based on coding opportunities, the coding opportunities are distributed more evenly in CORE than in other coding schemes.

E. What Happens to Coded Packets in FlexONC?

To show why by increasing BER FlexONC outperforms other schemes in throughput, we run simulations using the scenario depicted in Section V-B1, and calculate: 1) the total number of coded packets sent, 2) the number of coded packets received and forwarded by the intended forwarder, 3) the number of coded packets only received and forwarded by one of the nonintended forwarders (i.e., on behalf of the intended forwarder), and 4) the number of coded packets for which the sender does not receive any ACK (or NACK) and retransmits.

As shown in Fig. 25, by increasing BER, intended forwarders receive a smaller percentage of total coded packets sent, and the portion of coded packets which are received only by nonintended forwarders increases. This means that nonintended forwarders can cooperate more effectively in forwarding and be more beneficial. This collaboration among nodes, which increases at higher BER, is the key idea of FlexONC, which leads to increased robustness and higher packet delivery ratio in comparison to the baselines.

F. Packet Delivery Rate

Opportunistic forwarding is utilized to increase the probability of successful delivery of a packet as more nodes can help in forwarding packets. In this section, we investigate the effect of the number of nodes in the forwarder set, and the link quality on the performance of opportunistic forwarding protocols, especially BEND and FlexONC, for both native and coded packets. We focus on the case with no retransmission first, and the case with retransmission is a natural extension, as we see later. In addition, we assume that the nodes in the forwarder set have a perfect coordination mechanism, which means that all nodes in the forwarder set know which one of them forwards the packet.

Let us denote p as the probability of successful transmission at each link, and N as the average number of nodes in the forwarder set. Then, the probability of successful transmission of a native packet to at least one of the nodes in the forwarder set equals: $p_f^n = 1 - (1-p)^N$. If a packet traverses H hops in average to be delivered to the destination, in each transmission $N-1$ nonintended forwarders help the intended forwarder except for the transmission to the destination. Then, the probability of successful delivery to the destination can be calculated as: $p_d^n = (1-(1-p)^N)^{H-1} \times p$. It is worth noting that for $N = 1$ (i.e., only one node in the forwarder set of each transmission) $p_d^n = p^H$, which is basically the probability of successful delivery of a packet in traditional forwarding with H hops. Furthermore, when N increases $p_d^n > p^H$, which shows that by increasing the number of nonintended forwarders (i.e., the nodes in the forwarder set) the packet delivery rate increases.

Regarding coded packets, a received coded packet with m coding partners is decoded successfully if $m-1$ coded partners have already been received. Therefore, the probability of delivery of a coded packet to the next-hop equals p^m. As discussed earlier, in BEND coded packets are only forwarded by the intended forwarder (i.e., no opportunistic forwarding). Therefore, the probability of delivery of a coded packet with m coding partners to the destinations in BEND equals $p_d^c(\text{BEND}) = (1-(1-p)^N)(p^m)^{H-1}$, given that the source always sends native packets. On the other hand, since FlexONC extends opportunistic forwarding to coded packets as well, the probability of delivery of coded packets to the destination in FlexONC equals: $p_d^c(\text{FlexONC}) = (1-(1-p)^N)(1-(1-p^m)^N)^{H-2}p^m$.

To compare the delivery rate in BEND and FlexONC, we focus on the delivery of coded packets, which is different in these two approaches. Assuming that the coding opportunities at both protocols are similar, when the number of nonintended forwarders (i.e., N) increases, $p_d^c(\text{FlexONC})$ increases faster than $p_d^c(\text{BEND})$, which shows that the gain obtained by opportunistic forwarding is greater in FlexONC than in BEND. Furthermore, when the link quality is perfect (i.e., $p = 1$), the packet delivery ratio for both protocols is the same and independent of N, justifying the fact that in perfect network conditions opportunistic forwarding is not beneficial. However, as shown below, in imperfect link qualities (i.e., $p < 1$), FlexONC outperforms BEND

$$\begin{aligned} & 0 < p < 1 \\ & \Rightarrow \quad 0 < p^m < 1 \\ & \overset{N \geq 1}{\Rightarrow} \quad (1 - p^m) > (1 - p^m)^N \\ & \Rightarrow \quad (1 - (1 - p^m)) < (1 - (1 - p^m)^N) \\ & \Rightarrow \quad (p^m)^{H-2} < (1 - (1 - p^m)^N)^{H-2} \\ & \Rightarrow \quad p_d^c(\text{BEND}) < p_d^c(\text{FlexONC}). \end{aligned}$$

In addition, we can prove in a similar fashion that the performance gap between BEND and FlexONC in terms of the packet delivery rate increases as the link quality decreases. Furthermore, when retransmission is enabled, since $p_d^c(\text{FlexONC}) > p_d^c(\text{BEND})$, each coded packet in FlexONC needs less number of retransmissions to be delivered to the destination, which increases the capacity of the network, and consequently improves the performance.

G. Overall Comparison

In this section, we provide an overall comparison of FlexONC with other methods, especially BEND, in terms of required storage, packet overhead, computational complexity, delay, and throughput. FlexONC provides more coding opportunities, and outperforms other schemes in terms of throughput, especially at higher BERs. Even though having a more powerful protocol may imply increased complexity and overhead, this is not the case of FlexONC, and it is able to keep other metrics such as the end-to-end delay and the number of duplicate packets comparable to other methods, particularly BEND.

Regarding the packet header overhead, while BEND adds the *second-next-hop* field to the packet header of native packets (i.e., four bytes), FlexONC does not need this field. Instead, it adds a bitmap to the header of coded packets to specify eligible forwarders, which is the case in CORE as well. Given the total number of nodes N in the network, the array needs N bits in the packet header, which does not exceed a few bytes in average. Furthermore, to find the forwarder set in each node, CORE adds the geographical-position of the sender and the final destination of each packet to its header, which is not required by FlexONC.

On the other hand, COPE needs neither the second-next-hop field nor the bitmap since it does not benefit from opportunistic forwarding. Moreover, in FlexONC as well as all other opportunistic forwarding protocols with network coding (e.g., CORE and BEND), all nodes are in promiscuous mode, and store overheard (in addition to intended) packets. Therefore, this overhead is common in all mentioned baselines except for COPE. In fact, in all experiments over different methods, nodes have the same buffer size.

As explained earlier, in FlexONC, in contrast with COPE, CORE, and BEND, each node stores the forwarding information of its neighbors. This information is used to control the route followed by packets and avoid them from straying too away from the designated shortest path. If K denotes the maximum number of neighbors of a node in the network, and each entry of the forwarding table needs at most 10 bytes, the total memory required to store the forwarding information of the neighbors equals $10 \times K \times N$ bytes. Thus, in a network with about 30 nodes, even if we assume all nodes are connected to each other, the total required storage is less than 9 KB. On the other hand, while in BEND each node only stores its own forwarding table, the size of this forwarding table is greater than a regular forwarding table, as it stores the IP addresses of the *second-next-hops* in addition to the next-hops themselves.

All mentioned schemes need to utilize a routing protocol except for CORE as it broadcasts the packets. However, this broadcasting mechanism and lack of retransmission affects the performance of CORE significantly in lossy networks, as shown in the last section. Having routing information of the neighbors in FlexONC only requires adding one extra field to the route advertisement messages of a proactive routing protocol to include the next hop leading to each destination. However, this very small additional routing overhead is not limited to FlexONC; BEND also adds the same field to the route control packets to update *second-next-hop* field in the forwarding table of each node.

Regarding the computational complexity, the most important processes are encoding and decoding which are almost the same in all coding schemes except for CORE. While in FlexONC and other mentioned coding schemes, nodes encode the packets in advance immediately after reception, in CORE a packet is encoded when it is going to be transmitted. In addition, to increase the coding gain in lossy environments, CORE introduces a more complicated encoding algorithm in which each node checks all possible coding patterns of the first K packets in its queue.

In terms of the average end-to-end delay, as explained in Section VI-B, the delay in FlexONC is slightly longer than that in BEND because of the longer maximum waiting time before triggering retransmission of coded packets. Compared with CORE, at lower arrival rates the delay in CORE is significantly longer than that of FlexONC, since CORE delays native transmissions. On the other hand, at higher arrival rates the delay in FlexONC is longer.

Although the experiments in this paper are conducted in grid topologies, the benefit of having more *diffusion gain* as well as an additional rule in the coding conditions and having a mechanism to turn it ON/OFF dynamically is still present in general scenarios with random node distribution and flow assignments, and we expect the relative performance among these different methods to be similar to what we have shown here.

VII. CONCLUSION AND FUTURE WORK

This paper presented FlexONC, an enhancement over BEND, which provides more flexibility and coding opportunities in the network. By utilizing the broadcasting nature of wireless networks, FlexONC is able to spread different flows better than BEND and enable a higher level of cooperation between intended and nonintended forwarders at the link layer in a multi-hop wireless network. Furthermore, by adding an additional rule to the current conditions used to encode the packets in different methods, FlexONC provides more accurate coding conditions, and utilizes *SwitchRule* to apply these coding conditions appropriately and limit decoding failures.

By applying *SwitchRule*, FlexONC is able to adapt coding conditions in different scenarios, and uses a more complete set of rules for encoding when *common coding conditions* are not sufficient. Furthermore, FlexONC benefits from opportunistic forwarding especially at higher BERs. The performance results show that at higher BERs, when an intended forwarder may fail to receive or decode a coded packet and needs its neighbor's help, FlexONC significantly outperforms previous methods like BEND, CORE, COPE, and noncoding. Even under an ideal network condition, when intended forwarders usually do not need any help and can decode and forward received coded packets, FlexONC outperforms other schemes because of more precise coding conditions.

In future work, we plan to provide an analytical model for the combination of opportunistic forwarding and inter-flow network coding in multihop wireless mesh networks. Furthermore, in recent years a number of publications have been presented that apply both inter- and intraflow network coding, but in some limited scenarios [22]–[24]. We believe that this combination, if realized carefully, could introduce further improvement in the performance, and represents another way to extend FlexONC.

Moreover, to address the coding condition problem described in this paper, *SwitchRule* is proposed which decides on more precise coding conditions (i.e., *RecodingRule*) in certain scenarios. In future, we plan to propose a scheme that provides nodes with more timely deterministic information and also more accurate probabilistic decisions in encoding. In addition, FlexONC can be extended to include a combination of cooperative forwarding with more powerful detection of coding opportunities beyond a two-hop region [25], [26].

REFERENCES

[1] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 497–510, Jun. 2008.

[2] J. Zhang, Y. P. Chen, and I. Marsic, "MAC-layer proactive mixing for network coding in multi-hop wireless networks," *Comput. Netw.*, vol. 54, no. 2, pp. 196–207, Feb. 2010.

[3] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[4] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 169–180, Aug. 2007.

[5] Y. Lin, B. Li, and B. Liang, "Efficient network coded data transmissions in disruption tolerant networks," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 2180–2188.

[6] J. Joy, Y.-T. Yu, M. Gerla, S. Wood, J. Mathewson, and M.-O. Stehr, "Network coding for content-based intermittently connected emergency networks," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 123–126.

[7] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *Proc. ACM SIGCOMM Workshop Delay-Tolerant Netw.*, 2005, pp. 284–291.

[8] L. Xie, P. H. Chong, I. W. Ho, and Y. Guan, "A survey of inter-flow network coding in wireless mesh networks with unicast traffic," *Comput. Netw.*, vol. 91, no. C, pp. 738–751, Nov. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2015.08.044

[9] M. A. Iqbal, B. Dai, B. Huang, A. Hassan, and S. Yu, "Review: Survey of network coding-aware routing protocols in wireless networks," *J. Netw. Comput. Appl.*, vol. 34, no. 6, pp. 1956–1970, Nov. 2011.

[10] J. Islam and D. P. K. Singh, "CORMEN: Coding-aware opportunistic routing in wireless mesh network," *J. Comput.*, vol. 2, no. 6, pp. 71–77, 2010.

[11] Y. Yan, B. Zhang, J. Zheng, and J. Ma, "CORE: A coding-aware opportunistic routing mechanism for wireless mesh networks," *IEEE Wireless Commun.*, vol. 17, no. 3, pp. 96–103, Jun. 2010.

[12] Q. Hu and J. Zheng, "CoAOR: An efficient network coding aware opportunistic routing mechanism for wireless mesh networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2013, pp. 4578–4583.

[13] H. Liu, H. Yang, Y. Wang, B. Wang, and Y. Gu, "CAR: Coding-aware opportunistic routing for unicast traffic in wireless mesh networks," *J. Netw. Syst. Manage.*, vol. 23, no. 4, pp. 1104–1124, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10922-014-9333-5

[14] S. Kafaie, Y. Chen, M. H. Ahmed, and O. A. Dobre, "Network coding with link layer cooperation in wireless mesh networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2015, pp. 5282–5287.

[15] L. Huang and C. W. Sung, "An iterative routing algorithm for energy minimization in coded wireless networks," in *Proc. 22nd IEEE Int. Symp. Personal Indoor Mobile Radio Commun.*, Sep. 2011, pp. 1124–1128.

[16] H. Seferoglu and A. Markopoulou, "Network coding-aware queue management for unicast flows over coded wireless networks," in *Proc. IEEE Int. Symp. Netw. Coding*, Jun. 2010, pp. 1–6.

[17] H. Guo, X. Liu, Z. Shi, and X. Bai, "Image relevance feedback retrieval based on selective cluster ensembles," in *Proc. Chin. Conf. Pattern Recog.*, Oct. 2010, pp. 1–5.

[18] S. Wang, G. Tan, Y. Liu, H. Jiang, and T. He, "Coding opportunity aware backbone metrics for broadcast in wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 275–279.

[19] *IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-2007 (Revision of Std. 802.11-1999), 2007.

[20] T. S. Rappaport, *Wireless Communications: Principles and Practice*. 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, Dec. 2001.

[21] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 234–244, Oct. 1994. [Online]. Available: http://doi.acm.org/10.1145/190809.190336

[22] H. Seferoglu, A. Markopoulou, and K. Ramakrishnan, "I2NC: Intra- and inter-session network coding for unicast flows in wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1035–1043.

[23] J. Hansen, J. Krigslund, D. Lucani, and F. Fitzek, "Bridging inter-flow and intra-flow network coding for video applications: Testbed description and performance evaluation," in *Proc. 18th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw.*, Sep. 2013, pp. 7–12.

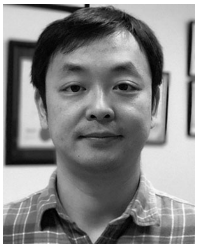
[24] J. Krigslund, J. Hansen, M. Hundeboll, D. Lucani, and F. Fitzek, "CORE: COPE with MORE in wireless meshed networks," in *Proc. IEEE Veh. Technol. Conf.*, Jun. 2013, pp. 1–6.

[25] J. Le, J. Lui, and D. M. Chiu, "DCAR: Distributed coding-aware routing in wireless networks," in *Proc. 28th Int. Conf. Distrib. Comput. Syst.*, Jun. 2008, pp. 462–469.

[26] B. Guo, H. Li, C. Zhou, and Y. Cheng, "Analysis of general network coding conditions and design of a free-ride-oriented routing metric," *IEEE Trans. Veh. Technol.*, vol. 60, no. 4, pp. 1714–1727, May 2011.

Somayeh Kafaie received the B.Sc. degree in computer engineering from Amirkabir University of Technology, Tehran, Iran, in 2007, and the M.Sc. degree in computer engineering from Iran University of Science and Technology, Tehran, Iran, in 2011. She is currently working toward the Ph.D. degree at the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NF, Canada.

Her research interests include wireless mesh networks, network coding, opportunistic routing, complex networks, and graph theory.



Yuanzhu Chen received the B.Sc. degree from Peking University, Beijing, China, in 1999 and the Ph.D. degree from Simon Fraser University, Burnaby, BC, Canada, in 2004.

He is an Associate Professor with the Department of Computer Science, Memorial University of Newfoundland, St. John's, NF, Canada. He was the Deputy Head for Undergraduate Studies in 2012–2015, and the Deputy Head for Graduate Studies in 2016 to present date. Between 2004 and 2005, he was a Postdoctoral Researcher with Simon Fraser University. His research interests include computer networking, mobile computing, graph theory, complex networks, Web information retrieval, and evolutionary computation.

Mohamed Hossam Ahmed received the Ph.D. degree in electrical engineering from Carleton University, Ottawa, ON, Canada, in 2001.

From 2001 to 2003, he was a Senior Research Associate with Carleton University. In 2003, he joined the Faculty of Engineering and Applied Science, Memorial University, where he works currently as a Full Professor. He published more than 135 papers in international journals and conferences. His research interests include radio resource management in wireless networks, multi-hop relaying, cooperative communication, vehicular ad-hoc networks, cognitive radio networks, and wireless sensor networks. His research is sponsored by NSERC, CFI, QNRF, Bell/Aliant and other governmental and industrial agencies.

Dr. Ahmed is a registered Professional Engineer (P.Eng.) in the province of Newfoundland, Canada. He serves as an Editor for IEEE COMMUNICATION SURVEYS AND TUTORIALS and as an Associate Editor for *Wiley International Journal of Communication Systems* and *Wiley Communication and Mobile Computing*. He served as a Guest Editor of a special issue on Fairness of Radio Resource Allocation, EURASIP JWCN in 2009, and as a Guest Editor of a special issue on Radio Resource Management in Wireless Internet, Wiley Wireless and Mobile Computing Journal in 2003. He served as the Cochair of the Signal Processing Track in ISSPIT14 and served as the Cochair of the Transmission Technologies Track in VTC10-Fall, and the multimedia and signal processing symposium in CCECE09. He received the Ontario Graduate Scholarship for Science and Technology in 1997, the Ontario Graduate Scholarship in 1998, 1999, and 2000, and the Communication and Information Technology Ontario Graduate Award in 2000.

Octavia A. Dobre (M'05–SM'07) received the Engineering Diploma and Ph.D. degrees from Politehnica University of Bucharest (formerly Polytechnic Institute of Bucharest), Bucharest, Romania, in 1991 and 2000, respectively.

She received a Royal Society Scholarship at Westminster University, U.K. in 2000, and held a Fulbright Fellowship at Stevens Institute of Technology, Hoboken, NJ, USA, in 2001. Between 2002 and 2005, she was with Politehnica University of Bucharest and New Jersey Institute of Technology, Newark, NJ, USA. In 2005, she joined Memorial University, St. John's, NF, Canada, where she is currently is a Full Professor and Research Chair. She was a Visiting Professor at the Université de Bretagne Occidentale, France, and Massachusetts Institute of Technology, Cambridge, MA, USA, in 2013. Her research interests include 5G technologies, blind signal identification and parameter estimation techniques, cognitive radio systems, and transceiver optimization algorithms for wireless communications, as well as optical and underwater communications. She has coauthored more than 200 journal and conference papers and gave more than 40 invited talks to industry and academia. Her research was supported by the Natural Sciences and Engineering Research Council of Canada, Mathematics of Information Technology and Complex Systems, Canada Foundation for Innovation, Research and Development Corporation, Atlantic Canada Opportunities Agency, Defence and Research Development Canada, Communications Research Centre Canada, Altera Corporation, Allen Vanguard, DTA Systems, EION Wireless, ThinkRF, and Agilent Technologies.

Dr. Dobre serves as the Editor-in-Chief of the IEEE COMMUNICATIONS LETTERS, as well as an Editor for the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS and IEEE SYSTEMS. She was an Editor and a Senior Editor for the IEEE COMMUNICATIONS LETTERS, an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and a Guest Editor for other prestigious journals. She served as the General Chair of CWIT, and the Technical Co-Chair of symposia at numerous conferences, such as IEEE GLOBECOM and ICC. She is the Chair of the IEEE ComSoc Signal Processing for Communications and Electronics Technical Committee, the Chair of the IEEE ComSoc WICE Standing Committee, as well as a member-at-large of the Administrative Committee of the IEEE Instrumentation and Measurement Society.