International Journal on Applications in Information and Communication Engineering
Volume 4: Issue 2: April 2018, pp 12 - 25. www.aetsjournal.com ISSN (Online) : 2394 - 6237

RECOVERY OF PACKET LOSSES FOR REAL TIME APPLICATIONS

R.SUDHARASHANA , K.SUBASRI , S.PAVUNPAPPA , S.SUNGAYA,
ARIVUSUDAR.K , M. MAKURU

Abstract— We consider the scenario of broadcasting for real-time applications, such as multi-player games and video streaming, and loss recovery via instantly decodable network coding. The source has a single time slot or multiple time slots to broadcast (potentially coded) recovery packet(s), and the application does not need to recover all losses. Our goal is to find packet(s) that are instantly decodable and maximize the number of lost packets that the users can recover. First, we show that this problem is equivalent to the unique coverage problem in the general case, and therefore, it is hard to approximate. Then, we consider the practical probabilistic scenario, where users have i.i.d. loss probability and the number of packets is either constant (video streaming), linear (multi-player games), or polynomial in the number of users, and we provide two polynomial-time (in the number of users) algorithms. For the single-slot case, we propose *Max Clique*, an algorithm that provably finds the optimal coded packet w.h.p. For the case where there is a small constant number of slots, we propose *Multi-Slot Max Clique*, an algorithm that provably finds a near-optimal solution w.h.p. when the number of packets is sufficiently large. The proposed algorithms are evaluated using both simulation and real network traces from an Android multi-player game. And they are shown to perform near optimally and to significantly outperform the state-of-the-art baselines.

Keywords— Broadcast, loss recovery, instantly decodable network codes, real-time applications, network coding.

I. INTRODUCTION

Broadcasting data to multiple users is widely used in many wireless applications, ranging from satellite communications to WiFi networks. Wireless transmissions, however, are subject to packet losses due to channel impairments, such as, wireless fading and interference. Effectively recovering these losses could provide tremendous performance improvement to many applications, especially real-time applications, such as, fast-pace local multi-player games and live video streaming.

We have experienced this problem firsthand. In our previous work, we built MicroPlay [1], one of the first networking frameworks for multiplayer games that exploit WiFi broadcast to achieve accurate game rendering and low latency. The rendering of a player is quicker and more accurate because it is done by leveraging directly the command packets generated by the player (that are then broadcast to the other players). When deploying the system, we experienced broadcast losses, and our measurements showed that if we could recover even a small percentage of packet losses (less than 1%) in a timely manner, then our game rendering engine would benfit tremendously, *i.e.*, animation jitters could be completely eliminated.

As another example, CrowdWiFi by Streambolico [2] is one of the first commercial systems that exploit WiFi broadcast to stream live videos to a large number of users, such as, crowds at stadiums, concerts, and conferences. CrowdWiFi is able to achieve 11x bandwidth improvement over traditional unicast solutions [3]. The loss recovery algorithm used by CrowdWiFi is the key contributor to its superior performance: it has helped the system to keep the number of packets that miss their playback deadline low even when the loss rate is high. (The number of packets that miss deadline is kept negligible even when the wireless loss rate is up to 10% [3].)

Motivated by the importance of loss recovery in wireless broadcast for real-time applications, in this work, we aim to find the best coding scheme to recover packet losses.

Real-time applications have two distinct characteristics:

(i) they have strict and urgent deadlines, *i.e.*, a packet is outdated after a short amount of time, and (ii) they can tolerate some losses. For example, a game client can tolerate packet losses by partially moving (rendering) the players using only the received packets, then later on, it corrects the players' positions by syncing its state periodically [1]. Therefore, it is highly desirable to recover packet losses with very low delay and within a very narrow coding window. Consequently, we focus on coding schemes for loss recovery that allows for instantaneous decoding, *i.e.*, with zero delay. These schemes are also known as Instantly Decodable Network Codes (IDNC) [4]–[10].

In contrast to previous IDNC literature, our work does not focus on how to recover *all* packet losses with a minimal number of transmissions. This is because for real-time applications, there may not be enough time for all receivers to recover all the losses. Instead, we address the practical case where the

R.Sudharashana , Final Year Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.
K.Subasri , Final Year Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.
S.Pavunpappa , Final Year Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.
S.Sungaya , Final Year Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.
Arivusudar.K , Final Year Cse, Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.
M. Makuru , Head Of The Department Of Computer Science And Engineering , Meenakshi Ramaswamy Engineering College, Thathanur, Tamilnadu, India.

source, due to real-time constraints, has a limited number of opportunities to send recovery packets, *i.e.*, one time slot or a small number of time slots. To this end, we first investigate the scenario where the source has a single slot to transmit a recovery packet. We then use the result of our analysis for the single-slot scenario to analyze and design a recovery algorithm for the multi-slot scenario.

We formulate the single-slot problem, which we refer to as the *Real-Time IDNC* problem, as follows:

Consider a source that broadcasts a set of packets, , to a set of n users, . Each user, u , wants all packets in and already knows a subset of them, u, *e.g.*, through previous transmissions. The goal is to choose one (potentially coded) packet to broadcast from the source, so as to maximize the number of users who can immediately recover one lost packet. (This is equivalent to maximizing the number of lost packets that the users can recover.)

We first show that Real-Time IDNC is equivalent to the Maximum Clique problem in an IDNC graph (to be precisely defined in Section III) as well as the Unique Coverage problem (introduced by Demaine *et al.* [11]). Therefore, these problems are all hard to approximate [11].

Then, we present the main contribution of this work: the analysis of random instances of the problem, where the probability that a user receives a packet is i.i.d. Bernoulli. This problem, referred to as Random Real-Time IDNC, corresponds to a Maximum Clique problem on an appropriately created random IDNC graph. Surprisingly, we show that when the number of packets is polynomial in the number of users, the Maximum Clique problem can be solved with high probability on this particular family of random graphs, by a polynomial-time (in the number of users) algorithm that we propose, called *Max Clique*. Based on the results of the single-slot scenario, we then extend the analysis to the scenario where there is a small constant number of slots, and we propose *Multi-Slot Max Clique*, a recovery algorithm that provably finds near-optimal solution w.h.p. when the number of packets is sufficiently large (to be precisely defined in Section VI-B).

We implement and compare the proposed coding schemes, Max Clique and Multi-Slot Max Clique, against several baselines: the best repetition scheme, a COPE-like greedy scheme [12], the recovery scheme used by CrowdWiFi [3], the $\Omega(1/\log n)$-approximation algorithm proposed by Demaine *et al.* [11], and the optimal solution (via brute force). Simulation results show that our proposed schemes perform close to the optimal solution and significantly outperform the state-of-the-art ones for the loss rate varying from 1% to 99%. For example, for the 15-user 15-packet 2-slot case, Multi-Slot Max Clique improves by a factor of 1.3 on average over both COPE-like and best repetition code, and it performs up to 1.6 times better than the COPE- like code and up to 2.7 times better than the best repetition code. Finally, we evaluate Max Clique on network traces collected from a real-time multi-player game on Android. The results of this trace-based evaluation confirm the superior performance of Max Clique over the baselines.

The rest of this paper is organized as follows. Section II discusses related work. Section III formulates the Real-Time IDNC problem. Section IV describes the Maximum Clique and the Unique Coverage problems and shows that these two problems are equivalent to Real-Time IDNC. Section V analyzes the probabilistic version (Random Real-Time IDNC) and describes Max Clique, the polynomial-time algorithm to find a maximum clique w.h.p. Section VI analyzes the multi-slot scenario and describes the Multi-Slot Max Clique algorithm. We also discuss the support of packets with different priorities here. Section VII evaluates and compares our coding schemes with existing schemes. Section VIII concludes the paper.

II. RELATED WORK

Real-Time Applications That Use Wireless Broadcast: In our previous work, we designed and implemented MicroPlay, a networking framework for local multi-player games [1]. Games built on top of MicroPlay would have accurate rendering without the need of complex movement prediction and very low game latency. To achieve these desired properties, MicroPlay explicitly exploits the broadcast nature of WiFi: command packets are broadcast by the Access Point and pseudo-broadcast (unicast + overhearing) by the clients; these command packets are then used directly to render movement. In our experiments with MicroPlay, we have found that although broadcasting in our setting has a small loss rate (less than 1%), it demands frequent re-syncing of game state that incurs uncomfortable animation glitches. This motivates us to investigate the best way to recover loss of wireless broadcast in this work.

Recently, Ferreira et al. present a live video streaming system that exploits WiFi broadcast and uses IDNC loss recovery packets [3]. By combining broadcast and coded recovery, the authors show that their system could achieve up to 11x bandwidth improvement compared to traditional unicast streaming solutions. More importantly, the authors have shown that their recovery coding scheme has helped the system to keep the number of packets that miss their playback deadline low (negligible) even when the wireless loss rate is high (up to 10%). The system is currently offered by Streambolico as a commercial product called CrowdWiFi [2]. CrowdWiFi allows a WiFi base station to stream live video at high speed to a large number of users, *e.g.*, crowds at stadiums, concerts, and conferences. CrowdWiFi demonstrates that utilizing wireless broadcast is of high interest in practice. In Section VII, we compare our coding scheme to the heuristic coding scheme used in this system and show that we bring significant improvement.

Instantly Decodable Network Coding: Katti *et al.* [14] proposed COPE, an opportunistic inter-session network coding scheme for wireless networks. Encoded packets are chosen in a heuristic way that aims to maximize the number of receivers that can decode them in the next time slot. Keller *et al.* [12] investigated algorithms that minimize decoding delay, including two algorithms that allow for instantaneous decoding: a COPE-like greedy algorithm and a simple repetition algorithm.

In Section VII, we describe these two algorithms in more detail and use them as baselines for comparison.

In a follow-up work [13], Sadeghi *et al.* improved the opportunistic algorithm previously proposed by Keller *et al.* [12] by giving high priority to packets that are needed by a large number of users. The authors also gave an Integer Linear Program formulation to the problem of finding a packet that can be immediately decoded (*i.e.*, instantly decodable) by the largest number of users. They showed that this problem is NP-Hard based on the *Set Packing* problem. We note that their formulation differs from ours since it requires that a coded packet must be instantly decodable by *all users who still need at least one packet*. This restricts the solution to the one that satisfies all of these users. Thus, this may lead to a suboptimal solution because there may be a coded packet that is only instantly decodable *by some but not all* users but is beneficial to a larger number of users. Our formulation ensures that we find this optimal packet.

Sorour *et al.* have an extensive line of work investigating instantly decodable codes [4]–[10], focusing on minimizing the completion delay. They introduced the term Instantly Decodable Network Coding (IDNC) that we adopt in this work. In one of the earlier work [4], they proposed a construction of IDNC graphs based on feedback from the users and then introduced a transmission scheme based on graph partitioning. We consider the same construction of IDNC graphs as them [4]. Based on a stochastic shortest path formulation, they proposed a heuristic algorithm to minimize the completion delay [5]. In a latter work [6], they introduced the notion of *generalized* IDNC problem, which does not require the transmitted code to be decodable by all users, as opposed to the *strict* version studied previously [4], [5], [12]. Real-Time IDNC considers the generalized version. Furthermore, in a follow-up work [6], they related finding an optimal IDNC code to the Maximum Clique problem in IDNC graphs and suggested that it is NP-Hard; however, no explicit reduction was provided. In later work [7], [8], they extended the previous work [5] to cope with limited or lossy feedback. Recently, they considered the case of multicast instead of broadcast [9], and the case where users could buffer coded packets in addition to plain packets [10].

Li *et al.* [15] adopted IDNC for video streaming and showed that, for independent channels and sufficiently large video file, their proposed IDNC schemes are asymptotically throughput-optimal subject to hard deadline constraints when there are *no more than three users*. In contrast, we consider an arbitrary number of users, and we provide an optimal single transmission for the single-slot case and constant

number of near-optimal transmissions for the multi-slot case. CrowdWiFi also uses a heuristic IDNC recovery scheme [3], which we describe in detail and compare against in Section VII-A.

Index Coding: Our problem setup is relatively similar to that of the Index Coding (IC) problem, introduced by Birk and Kol [16] previously and extensively studied since. An IC problem also considers a source that knows a set of packets, , and a set of users. Each user (x,) demands one particular packet, x , and has side information consisting of a subset of packets. The source broadcasts to all users without errors. The goal is to find an encoding scheme that minimizes the number of transmissions required to deliver the packets to all users.

It has been shown that except for the cases that can be solved with one or two transmissions, other instances of the IC problem are NP-Hard to solve [17]–[19], including a variation of IC where users are pliable and happy to receive any one packet [20], [21]. Furthermore, even finding an approximation to the problem has been shown to be hard [22], previous work [23], [24] provided heuristic algorithms to find such codes.

Despite the similarities, there are two main differences between our problem and IC. First, in our problem, each user wants *all* the packets, not just a single packet. Second, we want to find an instantly decodable packet that maximizes the number of beneficiary users, not the total number of transmissions to satisfy all users.

Data Exchange: The Data Exchange (DX) problem, originally introduced by El Rouayheb *et al.* [25], also has

a similar setup to our problem. There is a set of packets,, and a set of users. Each user, u, knows a subset of packets, $_u$, and wants all packets in . In DX, all users can broadcast messages. The objective is to find an encoding scheme that minimizes the number of transmissions required to deliver all packets in to all users.

To solve the DX problem, a randomized polynomial-time solution was proposed [26], and deterministic polynominal time solutions were also proposed [27], [28]. There is existing work that studied the problem in general network topologies [29]. Variants of the problem where there are helpers and transmission weights were also studied [30], [31]. Various necessary and sufficient conditions that characterize feasible transmission schemes for the problem were also proposed under a different name of *universal recovery* [32]–[34].

Similar to DX, in our setting, all users want all the packets in . However, there are two main differences: (i) in our setting, only the source can broadcast as opposed to having all users capable of broadcasting, and (ii) we are nterested in instantaneous decoding to maximize the number of beneficiary users with one transmission, as opposed to minimizing the total number of transmissions.

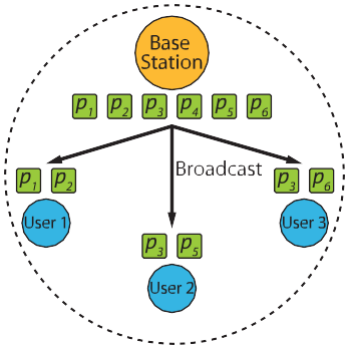

Figure 1. Example 1: A source broadcast 6 packets p_1, , p_6 to 3 users. Due to packet loss, user 1 only received p_1 and p_2; user 2 received p_3 and p_5; user 3 received p_3 and p_6.

Unique Coverage: This problem was originally introduced by Demaine *et al.* [11] and is as follows. Given a universe $U = e_1, , e_n$ of elements and a collection $S = S_1, , S_m$ of subsets of U. Find a subcollection S', S to maximize the number of elements that are uniquely covered, *i.e.*, appear in exactly one set of S'.

The Unique Coverage problem was shown to be hard to approximate within a factor of $\Omega(1/\log^{\sigma} n)$, for some constant σ depending on s, assuming that NP ¢ BPTIME(2^{n^g}) for some $s > 0$ [11] . In this work, we show that Real-Time IDNC is equivalent to Unique Coverage, thus it is also hard toapproximate. The detailed mapping between the two problems is provided in Section IV.

This Work in Perspective: A preliminary version of this work has appeared before [35]. In this paper, we extend the previous work in the following ways. First, by showing that Real-Time IDNC is equivalent to Unique Coverage, we show that Real-Time IDNC is not only NP-Hard [35] but also hard to approximate. Second, we extend our analysis of the single-slot scenario to the multi-slot scenario, and we propose Multi-Slot Max Clique, a recovery algorithm that provably finds near-optimal solutions. Finally, we collect network traces of a real-time application that utilizes broadcast and present a new evaluation based on the traces.

III. PROBLEM FORMULATION

Let

$$U = \{u_1, \ldots, u_n\}$$

denote the set of n users, and $P = \{ P_1 \ldots . P_m \}$ be the set of m packets. We assume that the original m packets were broadcast by a source. Due to packet loss, each of n users missed some of the m packets. (In general, a user may not have missed any packet, in which case, we do not consider him/her as part of the problem formulation.) We denote the set of packets that were successfully received by user i by H_i. Furthermore, let W_i be the set of packets that user i wants, *i.e.*,

$$W_i = P \setminus H_i.$$

Consistently with existing literature [4], [17], we call H_i and W_i the "Has" and "Want" sets of user i, respectively.

After the initial broadcast, the source tries to recover the losses,

$$W = \bigcup_{i \in [1,n]} W_i$$

by sending coded packets and exploiting the side information of the already delivered packets,

$$H = \bigcup_{i \in [1,n]} H_i$$

Let the n m matrix **A** be the identification matrix for the side information of the users, *i.e.*,entry $a_{ij} = 1$ if user u_i wants packet p_j and 0 otherwise. **A** is also called a feedback matrix [4]–[9]. Let us clarify this by an example.

Example 1: Consider a scenario with 3 users and 6 packets. Furthermore, assume that after the initial broadcast, user u_1 successfully received packets p_1 and p_2; user u_2 received p_3 and p_5; and user u_3 received p_3 and p_6. The scenario is depicted in Fig. 1. In this case, the side information matrix is as follows:

$$\mathbf{A} = \begin{matrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{matrix}.$$

To deliver the packets in the of the users, we focus on instantly decodable, lightweight coding schemes that operate over GF(2). For a set of packet M , s the corresponding coded packet, c, is their binary sum, denoted by

$$c = \bigoplus_{p_i \in M} p_i. \qquad (1)$$

Definition 2: A coded packet, c^N , is instantly decodable *with respect to a set of users,* N *, if and only if*

(i) Every user, u_i , can decode c^N immediately upon reception to recover a packet p^i $_i$. That is, each user in Benefits *from c^N by recovering one of the packets from its want set.*

(ii) Every packet in the binary sum of c^N is wanted by at least one user in .

For example, for the scenario of Example 1, the coded packet $c^{\{u}1,u2,u3\}$ $p_1 \oplus p_3$ is instantly decodable with respect to u_1, u_2, u_3 since u_1 can recover p_3, while u_2 and u_3 can get p_1. Meanwhile, $c^{\{u}2,u3\} = p_5 \oplus p_6$ is not instantly decodable with respect to u_1.

Note that without the second condition of the definition, the packet is still instantly decodable. This condition is included to facilitate the analysis: with this condition, every component (plain packet) of a solution (coded packet) contributes to the value of the solution. For instance, we do not consider $c^{\{u}2,u3\} = p_3 \oplus p_5 \oplus p_6$ instantly decodable with respect to u_2, u_3 since although $c^{\{u}2,u3\}$ can be decoded by u_2 and u_3, packet p_3, which is a component of $c^{\{u}2,u3\}$, is not wanted by either u_2 or u_3. In other words, p_3 does not add value to the solution as u_2 and u_3 already knew it. From here on, we will omit the superscript $|N|$ of c^N when there is no ambiguity. We would like the coded packet to be immediately beneficial to as many users as possible. Thus, our notion of optimality is w.r.t. the cardinality of the set of beneficiary users .

The Real-Time IDNC Problem: Given a side information matrix **A**, find a (possibly coded) packet which can be instantly decoded by the largest number of users. We refer to this packet an optimal recovery packet.

IV. MAXIMUM CLIQUES IN IDNC GRAPHS AND THE UNIQUE COVERAGE PROBLEM

Given a side information matrix **A**, we form an Instantly Decodable Network Coding (IDNC) graph corresponding to **A** as in [4]: we create a vertex v_{ij} when user u_i still wants packet p_j. For instance, for matrix **A** in

Example 1, there is a vertex for each entry 1 in the matrix. Given a vertex v_{ij}, we use the term *user index* of v_{ij} to indicate i and *packet index* of v_{ij} to indicate j. There is an edge between two vertices v_{ij} and v_{kA} if one of the below conditions hold:

(i) $j = A$: In this case, both users u_i and u_k want the same packet $p = p_j = p_A$.

(ii) p_j $_k$ and p_A $_i$: In this case, user u_k has packet p_j that user u_i still wants, and vice versa.

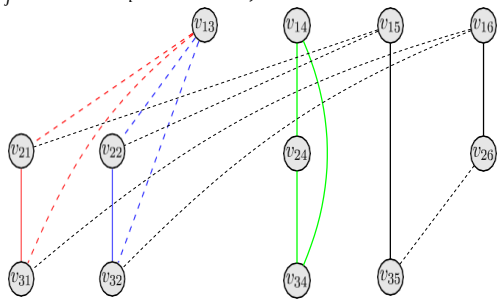

Figure 2. The Instantly Decodable Network Coding (IDNC) graph of Example 1. Solid edges are edges of type (i) and dashed edges are edges of type (ii). There are three maximum cliques: $\{v_{13}, v_{21}, v_{31}\}$, $\{v_{13}, v_{22}, v_{32}\}$, and $\{v_{14}, v_{24}, v_{34}\}$, all of which are of size 3.

Denote the IDNC graph corresponding to a matrix **A** by $tt^{\mathbf{A}} = (\quad E)$. Figure 2 shows the IDNC graph corresponding to the side information matrix given in Example 1.

A. Cliques and Instantly Decodable Packets

Proposition 3: Finding an optimal instantly decodable code given a side information matrix **A** *is equivalent to finding a maximum clique in the corresponding IDNC graph* $tt^{\mathbf{A}}$.

The proof is provided in Appendix A. Intuitively, let us consider the clique involving v_{13}, v_{21}, and v_{31} in Example 1. XORing all packets corresponding to vertices of this clique, *i.e.*, p_1 p_3, forms an instantly decodable packet because

(i) user 1 must have p_1, and users 2 and 3 must have p_3, otherwise there are no edges (v_{13}, v_{21}) and (v_{13}, v_{31}), and

(ii) each component of the coded packet is wanted by the user corresponding to the row of the vertex. Finally, the clique size equals 3, which is the number of beneficiary users.

B. Real-Time IDNC and Unique Coverage

In a previous work [35], we showed that Real-Time IDNC is NP-Hard. Here we show that Real-Time IDNC is in fact equivalent to the Unique Coverage problem [11], thus it is not only NP-Hard but also hard to approximate. Recall that the Unique Coverage problem is formulated follows:

Given a universe $U = \quad e_1, \quad , e_n$ of elements and a collection $S = S_1, , S_m$ of subsets of U. Find a subcollection $S' \subseteq S$ to maximize the number of elements that are uniquely covered, *i.e.*, appear in exactly one set of S'.

Let us call such a subcollection S' a "unique cover". Given a Real-Time IDNC problem, one can construct the corresponding Unique Coverage problem as follow: for each user u_i, create an element e_i, and for each packet p_j, create a subset S_j such that if p_j is wanted by u_i then S_j contains e_i. Given a Unique Coverage problem, one can also construct the corresponding Real-Time IDNC problem by reversing the above mapping. Consequently, based on the approximation result by Demaine *et al.* [11], we have the following result.

Theorem 4: Given a Real-Time IDNC problem with the side information matrix $\mathbf{A}_{n\times m}$ *and its corresponding Unique Coverage problem, finding an optimal instantly decodable packet and an optimal unique cover are equivalent. And theyare all hard to approximate within a factor of* $\Omega(1/\log^{\sigma} n)$, *for some constant* σ *depending on s, assuming that NP* ¢ $BPTIME(2^{n^g})$ *for some s*> 0.

The proof of this theorem is provided in Appendix B.

V. MAXIMUM CLIQUES IN RANDOM IDNC GRAPHS

In this section, we investigate Random Real-Time IDNC. In particular, we assume that each user, u_i, i [1, n], fails to receive a packet, p_j, j[1, m], with the same probability, p (0, 1), independently. For ease of analysis, we assume that m is linear in n: $m = dn$, for some constant $d >$ 0. As we will show subsequently, our results also hold when m is a constant or is polynomial in n. A random IDNC graph, denoted as $tt^{\mathbf{A}}(p)$, is the graph corresponding to a side information matrix **A**; and each entry of **A** equals 1 with probability p and 0 with probability $q = 1 \_ p$ independently from other entries.

In what follows, we will provide a concentration result of the size of the maximum clique, *i.e.*, the clique number, of random IDNC graphs. This result facilitates the design of a polynomial-time algorithm to find the maximum clique w.h.p. In particular, we will show the followings:

(i) For any p (0, 1), the clique number for almost every graph in $tt^{\mathbf{A}}(p)$ equals $\boldsymbol{j}^* pq^{j^*-1} n$, where $\boldsymbol{j}^* = \text{argmax}_{\boldsymbol{j}\, m,\boldsymbol{j}}$, $\mathrm{N} jpq^{j-1}$. And the optimal recovery packet involves combining $\boldsymbol{j}^*$ packets w.h.p.

(ii) The maximum clique can be found in polynomial time in n w.h.p. We provide an explicit algorithm, Max Clique, to find it. Consequently, the optimal recovery packet can be computed in polynomial time in n w.h.p.

Comparison to Erdős-Rényi Random Graphs: Clique numbers of Erdős-Rényi random graphs with n vertices and p = 1/2 are known to be close to $2\log_2 n$ [37]. However, it is widely conjectured that for any constant $s >$ 0, there does not exist a polynomial-time algorithm for finding cliques of size $(1 + s)\log_2 n$ with significant probability [38]. In contrast, for random IDNC graphs with n m vertices, where m is polynomial in n, we show that the clique numbers are linear in n, and the corresponding cliques can be found in polynomial time in n.

Number of Packets and Number of Users: Let us consider two practical real-time applications: live video streaming and multi-player games.

In live video streaming, there is typically a sliding widow where packets expire if they fall off this window. The number of packets sent per window is a function of the widow size (*e.g.*, 1 second [3]), video bit rate (*e.g.*, 2 Mbps), and packet size (*e.g.*, 1500 B). In other words, it does not depend on the number of users. For analysis purposes, for this case, the number of packets could be considered a constant w.r.t. *n*.

In MicroPlay [1], *m* is linear in *n* if there is no overhearing available, *i.e.*, only the Access Point can broadcast (as in a typical WiFi LAN setting). This is because *m* directly relates to the number of command packets a player generates: a player generates command packets; packets are unicast to the Access Point; then the Access Point broadcasts them.

In summary, for the class of practical real-time applications we consider, *m* is typically a constant or linear in *n*, and our results are applicable in these cases. Theoretically, our results hold when *m* is polynomial in *n*.

A. Clique Number of Random IDNC Graphs

First, observe that any *k* 1's that lie in the same column of the side information matrix **A** form a clique of size *k* in the IDNC graph. Since the expected number of 1's in a single column of **A** is *np*, the expected size of this type of cliques that involve a single column is *np*. As a result, we expect the maximum clique size to be linear in *n*.

Fix a set C_j of *j* columns. A row *r* is said to be *good* with respect to C_j if among the *j* columns, it has 1 one and $j-1$ zeros. The probability that a row is good w.r.t. C is

$$f(j) = jpyq^{j-1}. \qquad (2)$$

Let Z_{C_j} be the number of good rows w.r.t C_j. Then Z_{C_j} has a binomial distribution Bin($n, f(j)$).

Let X_{C_j} be the size of the maximum clique that has at least one 1 (corresponding to a vertex) on every column of In other words, the 1's representing the vertices of the clique are on *j* columns of C_j, or the clique touches *j* columns of C_j. Observe that if $j = 1$, then $f(1) = p$, and $X_{C1} = Z_{C1}$, which is the number of 1's in the chosen column. Thus, X_{C1} has a Binomial distribution Bin(n, p). For $j > 1$, $X_{C_j} = Z_{C_j}$ since the set of good rows may not have a 1 in every column in C_j. The following lemma states that, when Z_{C_j} is sufficiently large, $X_{C_j} = Z_{C_j}$ w.h.p.

Lemma 5: For a set C_j of j columns, where j is a constant, there exists a constant $k_j > 0$ such that for all $k \geq k_j$,

$$Pr[Z_{C_j} = X_{C_j} | Z_{C_j} = k] \geq 1 - j\left(\frac{j-1}{j}\right)^k.$$

Proof: For $k \geq j > 0$, let B^j_k denote the number of ways to put *k* 1's into a matrix of size $k \times j$ such that (i) each row has one 1, and (ii) each column has at least one 1. Note that $B^1_k = 1$, and we have the following recurrence:

$$B^j_k = j^k - \binom{j}{1} B^j_k - \binom{j}{2} B^j_k - \cdots - \binom{j}{j-1} B^1_k. \qquad (3)$$

This recurrence states that the number of ways to put *k* 1's into *k* rows (each row has one 1) using exactly *j* columns equals to the number of ways to put *k* 1's into *k* rows without any column restriction except for the cases where there are 1, 2, ,*j* 1 empty columns. It can be shown by induction (details are in Appendix C) that

$$B^j_k = \sum_{i=0}^{j-1} (-1)^i \binom{j}{i} (j-i)^k.$$

Thus, we have that

$$B^j_k = j^k - \binom{j}{j-1} B^j_k + \cdots -$$

The following lemma states that X_{C_j}, the size of the maximum clique that touches all *j* columns, heavily concentrates around $nf(j)$ for large *n*.

Lemma 6: For a set of constant j columns C_j and any constant $c > 1$, let $\mu = nf(j)$ and $\delta = \sqrt{\frac{3c \ln n}{nf}}$. For a large n such that $\mu - \mu\delta \geq k_j$ (k_j is as in Lemma 5), we have

$$Pr[|X_{C_j} - \mu| \geq \mu\delta] \leq \frac{2}{n^c} + 2\mu\delta j\left(1 - \frac{1}{j}\right)^{\mu - \mu\delta}.$$

This probability goes to 0 as n .

The proof is provided in Appendix D. Intuitively, this result follows from $X_C = Z_C$ w.h.p. (Lemma 5), and the fact that the Binomial distributed Z_{C_j}, the number of good rows, concentrates heavily around its mean, $nf(j)$. Note that $\mu\delta$ is $\Theta(\sqrt{n \ln n})$; thus, X_{C_j} is within $\Theta(\sqrt{n \ln n})$ of $nf(j)$ w.h.p.

Next, for a constant *j*, let X_j be the size of the maximum clique that touches *any j* columns. X_j also heavily concentrates around $nf(j)$. Recall that $m = dn$, for some constant $d > 0$. Formally,

Theorem 7: For a constant j and any constant $c > j$, let $\mu = nf(j)$ and $\delta = \sqrt{\frac{3c \ln n}{nf(j)}}$. For a large n such that $\mu - \mu\delta \geq k_j$ (k_j is as in Lemma 5), we have

$$\Pr[|X_j - \mu| \geq \mu\delta] \leq \frac{2d^j}{n^{c-j}} + 2d^j n^{\mu - \mu\delta} \mu\delta j (1 - \frac{j}{j})^{\mu - \mu\delta}.$$

This probability goes to 0 as $n \to \infty$.

Proof: The proof is by using the union bound on the result of Lemma 6:

$$\Pr[|X_j - \mu| \geq \mu\delta] = \Pr[\cup_{C_j} |X_{C_j} - \mu| \geq \mu\delta]$$

$$\leq \sum_{C_j} \Pr[|X_{C_j} - \mu| \geq \mu\delta]$$

$$\leq m^j \frac{c}{n^c} + 2\mu\delta j(1-1/j)^{\mu-\mu\delta}$$

$$\leq \frac{2d^j}{n^{c-j}} + 2d^j n^{\mu-\mu\delta} \mu\delta j(1-1/j).$$

We note that the above concentration result also holds when the number of packets, m, is a constant or polynomial in the number of users.

In particular, in the case that m is a constant, *i.e.*, $m = d$, the concentration is more, *i.e.*, the bounds are tighter:

$$\Pr[|X_j - \mu| \geq \mu\delta] \leq \frac{2d}{n^c} + 2d\mu\delta j(1 - 1/j)^{\mu-\mu\delta}.$$

In the case that m is polynomial in n, *i.e.*, $m = n^d$, for some constant $d > 0$, we need a larger constant c (such that $c > dj$), which means less concentration (as δ is larger):

Let k_j be the minimum positive integer value of k such that $\sum_{i=2}^{j} (-1)^{i} \binom{j}{i} (j-i)^k \geq 0$. Then, for all $k \geq k_j$,

$$\Pr[Z_{C_j} = X_{C_j} | Z_{C_j} = k] = B \geq \frac{\sqrt{j^k - j(j-1)^k}}{j^k}.$$

Apparently, the results do not hold when m is exponential in n. As mentioned, the cases where m is either a constant or linear in n are sufficient for practical real-time applications considered in this work, such as video streaming or multi-player games .

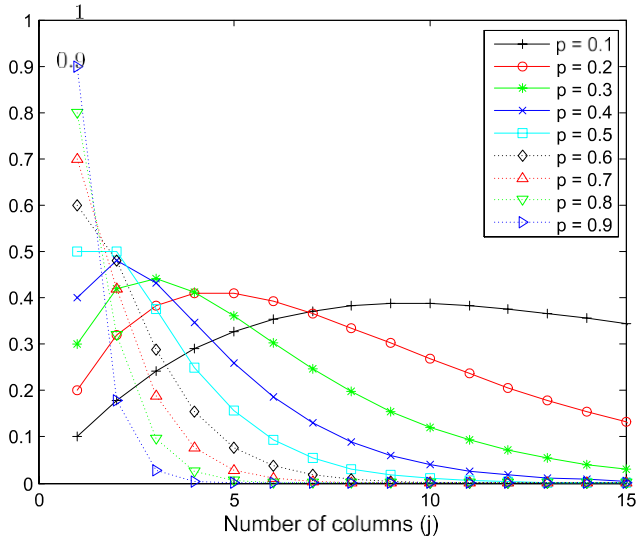

Figure 3. Plot of $f(j) = jp(1-p)^{j-1}$ for different loss rate p.

Now let $j^* = \arg\max_{j \leq m, j \in N} f(j)$ and let $j_{max} = \arg\max_{j \in N} f(j)$. Note that for a constant p, j_{max} is also a constant. If m is large enough, *i.e.*, if $m \geq j_{max}$ then $j^* = j_{max}$. If $m \leq j_{max}$ then $j^* = m$ as $f(j)$ is an increasing function for $j < j_{max}$, as illustrated in Fig. 3.

Corollary 8: For a sufficiently large n, with high probability, the maximum clique touches j^* of columns, where $j^* = \arg\max_{j \leq m, j \in N} f(j)$.

Proof: Intuitively, this follows from the above result that the size of the maximum clique that touches j columns heavily concentrates around $nf(j)$. In detail, for any constant j such that $f(j) < f(j^*)$, let $c > \max(j, j^*)+1$. Theorem 7 implies that w.h.p., the size of the maximum clique that touches any j columns is at most

$$k' = nf(j) + \sqrt{3cf(j)n \ln n},$$

and the size of the maximum clique that touches any j^* column is at least

$$k'' = nf(j^*) - \sqrt{3cf(j^*)n \ln n}.$$

For a large enough n, it is clear that $k' < k''$.

plot shows that (i) for p 0.5, $f(j)$ is a decreasing function, and for p < 0.5, $f(j)$ initially increases then decreases, and

(ii) j^* increases as p decreases, which suggests the following key insight:

The number of packets that should be coded together increases when the loss rate decreases.

Fig. 4 plots the values of $f(j^*)$ and the corresponding values of j^*. An important observation from Fig. 4 is that even when the loss rate is small, the clique size is still high. For instance,

when p = 0.1, we have j^* = 9 and $f(j^*)$ ⊂ 0.38, which means that the optimal packet involves coding 9 plain packets together, and this packet will benefit about 38% of the users. To better understand the implication of this result, let us

discuss two extreme cases: large p and small p. When the loss rate p is large (*e.g.*, greater than 0.9), our analysis shows that $j^* = 1$, thus $f(j^*) = f(1) = p$, indicating that w.h.p,

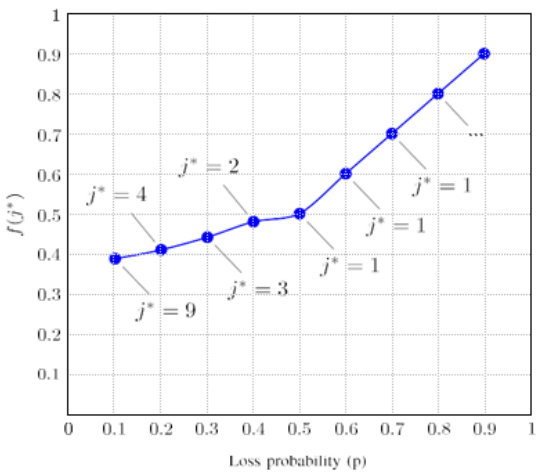

Figure 4. Values of $f(j^*)$ and its corresponding j^*. The clique number heavily concentrates around $f(j^*) \times n$, and j^* is the number of packets should be coded together.

the maximum clique corresponds to some (roughly) pn nodes, all having a 1 under the same packet in the side information matrix. Therefore, the broadcast of that single uncoded packet is most likely optimal. When p is small (*e.g.*, less than 0.1), our analysis shows that $j^* > 9$, *i.e.*, there is a likely a clique of size $j^*(1-p)^{j^*-1}pn > pn$, involving nodes in j^* different columns, and these nodes are connected in the IDNC graph by the edges of the second type. For such cliques, we need to code. We further elaborate on these cases at the end of Section VII-A, where we discuss the simulation results.

B. Finding a Maximum Clique

Based on the analysis in the previous section, we propose Max Clique (Algorithm 1). It examines all cliques that touch j columns, for all j combinations of m columns, where j is within a small constant Δ neighborhood of j^*, where Δ is a small constant, *e.g.*, Δ 3.

Lemma 9: For a sufficiently large n, Algorithm 1 finds a maximum clique of a given random IDNC graph w.h.p.

Proof: Because Max Clique examines all cliques that touch j^* columns, the correctness of the algorithm follows from Corollary 8.

Δ improves the performance of the algorithm in practice, when n may not be sufficiently large. In Section VII, we how that a small value of $\Delta = 3$ is sufficient for Max Clique to achieve top performance for $n = 5, 8, 15$. Note that the larger n is, the smaller Δ has to be.

Complexity: Recall that $j_{\max} = \text{argmax}_j \, Nf(j)$. If $j_{\max} < m$, then $j^* = j_{\max}$. For a constant loss rate p, $j_{\max}$ is a constant. Then for each loop starting at Line 3 runs at most 2Δ m $j\max+\Delta$ times. The for loop starting at Line 5 runs n times. The if condition check at Line 6 examines up to $j_{\max} + \Delta$ entries. The total runtime is at Most .

$$2\Delta \cdot \underbrace{m}_{j_{\max}+\Delta} \cdot n(j_{\max} + \Delta) = O(nm^{j_{\max}+\Delta}).$$

Thus in this case, Max Clique runs in polynomial time in n if m is polynomial in n.

Algorithm 1 *Max Clique*: Finding the Maximum Clique

Input: p: loss probability, n: number of users, m: number of packets, $\mathbf{A}$: side information matrix of size $n \times m$, δ: neighborhood size

Output: $\mathbf{I}^*$: vertices of the maximum clique

1: $j^* \in \text{argmax}_{j \le m, j \in \mathbb{N}} f(j)$
2: $\mathbf{I}^* =$
3: **for each** combination of j columns out of m columns, where $j \in [j^* - \Delta, j^* + \Delta]$
4: $\quad \mathbf{I} =$
5: $\quad$ **for** $r = 1 \in n$ **do**
6: $\quad\quad$ **if** row r has only one 1 at column c **then**
7: $\quad\quad\quad$ Add (r, c) to $\mathbf{I}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **if** $|\mathbf{I}| > |\mathbf{I}^*|$ **then**
11: $\quad\quad \mathbf{I}^* = \mathbf{I}$
12: $\quad$ **end if**
13: **end for each**

If $m \le j_{\max}$, then $j^* = m$. The for each loop starting at starting at Line 5 runs

$$\sum_{m-\Delta}^{\Delta} n(m+\Delta) = O(nm).$$

Thus, in this **e**,Max Clique n times. The if condition check at Line 6 examines up to $m + \Delta$ entries. The total runtime is

$m\Delta+1 = \Delta$

also runs in polynomial time in n if m is polynomial in n.

Optimal Coded Packet: Given the vertices of the maximum clique output by Max ≤ Clique, one can readily compute the optimal instantly decodable packet by XORing the packets whose indices correspond to the packet indices of the output vertices, as indicated in Proposition 3.

VI. THE MULTI-SLOT MAX CLIQUE ALGORITHM

So far, we consider the scenario where the source has a single slot to broadcast a single recovery packet. However, for some real-time applications, for the same feedback matrix, the source may have opportunities to broadcast multiple recovery packets. In this section, we formulate the problem of loss recovery for the scenario where there is a small constant number of slots. We show that this problem is also hard to

approximate. We then extend our analysis in the probabilistic setting for the single-slot scenario to the multi-slot scenario, where the number of packets is sufficiently large. Based on the analysis, we present the Multi-Slot Max Clique algorithm that can find a near-optimal solution w.h.p.

A. Multi-Slot Formulation

The number of transmission opportunities (time slots) for recovery packets depends on many factors. For example, in CrowdWiFi [2], [3], the video base station only broadcasts recovery packets when its incoming buffer does not contain any new packet (new video data) or any critical packet (packets medium, the reception rates at the users, and maybe on other factors as well. We formulate the problem of loss recovery for the multi-slot scenario as follows:

Consider a source that broadcasts a set of packets, to a set of users . Each user, u, already knows a subset of them, $_u$, through the previous broadcasts and wants the rest, $_u = _u$. Given t time slots, where t is a small constant, $0 < t < u \in \mathrm{U}\ u$, the goal is to choose up to t instantly decodable packets to broadcast from the source, so as to maximize the number of lost packets that the users can recover.

The multi-slot problem is at least as hard to approximate as the single-slot problem. This is a direct result of the single-slot analysis. (The reduction from the multi-slot problem to single-slot problem is by setting $t = 1$.) Next, we analyze the multi-slot problem in the probabilistic setting, where the number of packets is sufficiently large (as defined below). We then present the Multi-Slot Max Clique algorithm, that is constructed based on the analysis and can find a near-optimal solution w.h.p.

B. The Multi-Slot Max Clique Algorithm

Analysis: Consider the probabilistic setting of the t-slot scenario with uniform loss rate p. For the analysis in this section, we assume that m is sufficiently large:

$m > tj_{\max}$,

where $j_{\max} = \operatorname{argmax}_{j \ \mathrm{N}} f(j) = \operatorname{argmax}_{j \ \mathrm{N}} jp(1 \quad p)^{j-1}$.

Recall that for the single-slot scenario, the maximum clique concentrates around $nf(j_{\max})$ w.h.p. The key observation we use to extend the single-slot result to the multi-slot scenario is that the size of the maximum clique does not depends on the number of packets but only depends on the number of users and the loss rate.

Let us analyze the following algorithm, called "Multi-Slot Max Clique with Removal" or MSMCR for short, that uses Max Clique consecutively t times: for each time $i = 1, , t$, run Max Clique to find the maximum clique C_i, then remove the $j_{\max}$ columns that are associated with C_i before the next iteration. Finally, the algorithm outputs $C_1, , C_t$ as the resulting cliques. Since these cliques have non-overlapping vertices, the size of the solution, *i.e.*, the number of packets that can be recovered, denoted by MSMCR, is $^{t}\ C_i$. Now, let OPT denote the size of the optimal solution for the t-slot problem. The following lemma states that, in the worst case, MSMCR is $(t\ 1)\ 2\mu\delta$ smaller than OPTw.h.p.

Lemma 10: For a large m such that $m > tj_{\max}$ and for any constant $c > j$, let $\mu = nf(j \quad)$ and $\delta = \frac{3c \ln n}{nf(j_{\max})}$ a large n such that $\mu - \mu\delta \geq k_{j_{\max}}$ ($k_{j_{\max}}$ is as in Lemma 5), we have

$$\Pr[OPT - MSMCR > (t-1)2\mu\delta]$$

already broadcast but missing at some of the users and needed immediately for playback). In general, the number of recovery time slots depends on the video bit rate, the outgoing transmission rate of the station, the loss rate of the wireless

$$\leq 1 - \quad 1 - \left(\frac{2d^{j_{\max}}}{n^{c-j_{\max}}} + 2d^{j_{\max}} n^{j_{\max}} \mu\delta j_{\max}\left(1 - \frac{1}{j_{\max}}\right)^{\mu-\mu\delta}\right)^{t}$$

This probability goes to 0 as n .

The proof of this lemma is provided in Appendix E.

Algorithm 2 *Multi-Slot Max Clique*: Finding Multiple Cliques

Input: t: number of time slots, p: loss probability, n: number of users, m: number of packets, $\mathbf{A}$: side information matrix of size $n \times m$, δ: neighborhood size

Output: $C_1, \cdots, C_t$: t recovery cliques

1: $C_1 = \mathrm{MaxClique}(p, n, m, \mathbf{A})$
2: $C = C_1$
3: $j^* \ \min(m, \operatorname{argmax}_{j \ \mathrm{N}} f(j))$
4: **for** $i = 2$ **do**
5: $_i = C$
6: **for each** combination of j columns out of m columns, where $j \ [j^* - \Delta, j^* + \Delta]$
7: $\mathbf{I} =$
8: **for** $r = 1$ **do**
9: **if** row r has only one 1 at column c **then**
10: Add (r, c) to $\mathbf{I}$
11: **end if**
12: **end for**
13: **if** $|C \ \mathbf{I}| > |C \ C_i|$ **then**
14: $C_i = \mathbf{I}$
15: **end if**
16: **end for each**
17: $C = C \ C_i$
18: **end for**

Algorithm: We construct the Multi-Slot Max Clique algorithm based on the MSMCR algorithm. Algorithm 2 describes Multi-Slot Max Clique. Multi-Slot Max Clique is an improvement of MSMCR as it does not perform the removal step, *i.e.*, it investigates all cliques that MSMCR does and some more. Consequently, the solution of Multi-Slot Max Clique is at least as good as MSMCR.

In particular, in the first slot, it uses the result of Max Clique (Line 1). In subsequent slots, it executes a variant of Max Clique (Line 5–17), where the criterion to choose the current clique is that its union with the previously chosen cliques should have the largest number of vertices (Line 13). In other words, at step $i > 1$, clique C_i is chosen such that the union of all the cliques $C_1, , C_i$ covers the largest number of vertices.

omplexity: The Multi-Slot Max Clique algorithm essentially executes Max Clique t times where t is a small constant. Thus, it is still polynomial in n when m is polynomial in n.

Coded Packets: Given the vertice sets C_1, , C_t of cliques output by Max Clique, one can compute an instantly decodable packet corresponding to each clique as before.

C. Packet Priority

Different packets in a real-time application may have different priorities. For example, in a video streaming application, a packet, p_j, could be more urgent than others if it is needed for immediate video playback at a receiver, u_i. (CrowdWiFi has a similar concept of α-critical packets [3].) In this scenario, a coded packet should be constructed to facilitate the immediate recovery of p_j at user u_i. Finding the optimal coded packet here is equivalent to finding the maximum clique that has vertex v_{ij} in the IDNC graph.

This variation of the problem, called Priority IDNC, is also at least as hard to approximate as Real-Time IDNC. The sketch of the reduction is as follow. Any Real-Time IDNC problem can be mapped into an instance of Priority IDNC by creating a new user u and a new packet p, where u has all existing packets but not p, and all existing users have p. This pair u, p then corresponds to the only new vertex, v, in the new IDNC graph, and this vertex connects to all existing vertices. Thus,is a clique in the original IDNC graph if and only if v is a clique in the new IDNC graph. Therefore, any algorithm that solves Priority IDNC can be used to solve Real-

Time IDNC: the solution to Real-Time IDNC can be obtained by removing v from the solution of Priority IDNC.

In the probabilistic setting, Max Clique can be used to find a good coded packet for Priority IDNC as follow. Given a feedback matrix $\mathbf{A}$ and an urgent vertex $v_{i,j}$, we form a smaller feedback matrix $\mathbf{A}^{\chi}$ by removing all users who do not have packet p_j and all packets that user u_i does not have. Now, if

is a clique in $tt^{\mathbf{A}^{*}}$ then $v_{i,j}$ is a clique in $tt^{\mathbf{A}}$. (Note that the reverse is not true.) This is because every vertex in $tt^{\mathbf{A}^{*}}$ connects to $v_{i,j}$ by an edge of the second type. Thus, we can add $v_{i,j}$ to the solution of Max Clique for $tt^{\mathbf{A}^{*}}$ to form a solution for the original Priority IDNC problem.

Other greedy-based coding schemes, *e.g.*, COPE-Like and CrowdWiFi's regular coding scheme [3], [12], that form a coded packet by consecutively adding plain packets in multiple rounds (described in detail in Section VII-A) can be modified to heuristically address this Priority IDNC problem as well. The modification is by starting with p_j in the first round, and maintaining the decodability of p_j at u_i at every subsequent round. This approach is similar to the Critical Recovery algorithm in CrowdWiFi [3].

VII. PERFORMANCE EVALUATION

A. Numerical Evaluation

In this section, we use simulation to compare the performance of the proposed algorithms: Max Clique (Algorithm 1) and Multi-Slot Max Clique (Algorithm 2) against several baselines: an optimal repetition-based algorithm, a COPE-like algorithm [12], the recovery algorithm used by CrowdWiFi [3], the $\Omega(1/\log n)$-approximation algorithm [11], and the optimal solution (via brute force). We describe these baseline algorithms below.

Best Repetition: This algorithm sorts the plain packets in the decreasing order of the number of users who still wants the packets. Then it rebroadcasts the packets in this order. This is inherently the best repetition strategy.

COPE-Like [12]: For the single-slot case, this algorithm goes through all the packets that are still wanted by at least one user in a random order, and it tries to compute a coded packet, c, that is instantly decodable to *all* users. In particular, it begins by selecting the first packet of a random permutation,

$c = p_1$. It then goes through the rest of the packets one by one in that order. At each step j, $j > 1$, it XORs the next packet with c if the resulting c is still instantly decodable to users; otherwise, it consider the next packet in the line. For the multi-slot scenario, this process is repeated with a smaller coding pool where the packets already used in the previous slots are removed.

Greedy-Exhaustive [3]: This refers to the coding scheme used in the Recovery Algorithm of CrowdWiFi for video streaming [2], [3]. This algorithm is similar to the above COPE-Like algorithm, except for two changes: (i) it starts

with $c = p_1$, where p_1 is wanted by the most number of users, and (ii) at each step $j > 1$, it chooses p_j such that $c = c\ p_j$ is instantly decodable to the *most* number of users, *i.e.*, by *exhaustively* searching among all remaining packets.

It stops when it cannot find a new c that increases the number of benefiting users. As described, this algorithm's runtime is in $O(m!)$, which is exponential in n. For a fair baseline of comparison against our multi-slot algorithm, we extend this coding scheme to support the multi-slot scenario by removing packets already used in the previous slots from the coding pool,[1] similar to the COPE-Like.

$\Omega(1/\log n)$-*Approximation [11]:* We will simply refer to this as *Approximation*. The algorithm was originally given for the generalized Budgeted Unique Coverage problem. To adopt the algorithm for the Unique Coverage problem, we set the profits of all elements to 1, costs of all subsets to 1, and the budget to the number of subsets, which is m. We refer the reader to Section 4.1 of the original work of Demaine *et al.* [11] for the detailed description of the algo- rithm. We also extend this coding scheme to support the multi- slot scenario by removing already used in the previous slots from the coding pool.

Optimal (Brute Force): Given a feedback matrix $\mathbf{A}$ and $0 < d <$ $_{u\in U}$ $_{u}$, the optimal algorithm exhaustively search for d or less cliques that cover the largest number of vertices in the

IDNC graph of **A**. The algorithm runtime is in $O(2^{dm})$ as it enumerates all subsets of size d or less of the set of all possible coded packets. This is clearly not practical for large problem instances and is only used here to assess how far from the optimal our proposed algorithms perform.

Random Repetition: For reference, we also include the Random Repetition algorithm, which chooses a random permutation of the packets that are still wanted by at least one user, then rebroadcasts them in this order.

Simulation Setup: For each loss rate ranging from 1% to 99%, per 1% increment, we randomly generate 100 side information matrices. We then run all the algorithms on these matrices. For the Max Clique and Multi-Slot Max Clique algorithms, we set Δ, the neighborhood around j^*, to 3. Fig. 5 plots the average numbers of packets that can be recovered by the users for three parameter settings $\{n = 15, m = 15, t = 1\}$, $\{n = 8, m = 8, t = 2\}$, and $\{n = 5, m = 5, t = 3\}$. Note that in this set of simulation, to accommodate the expensive computation of the Optimal algorithm, we have to decrease the problem size as we increase the number of slots. Omitting the Optimal solution, Fig. 6 provide plots for larger problem size for the multi-slot scenarios: $\{n = 15, m = 15, t = 2\}$, $\{n = 15, m = 15, t = 3\}$, and $\{n = 15, m = 15, t = 4\}$. We have also performed simulation for larger settings, $\{n = 20, m = 20, t = 1\}$ and $\{n = 20, m = 40, t = 1\}$ [35] that show similar results and we

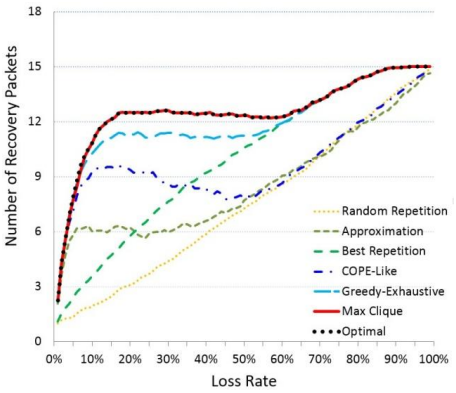

Figure 5 (a)

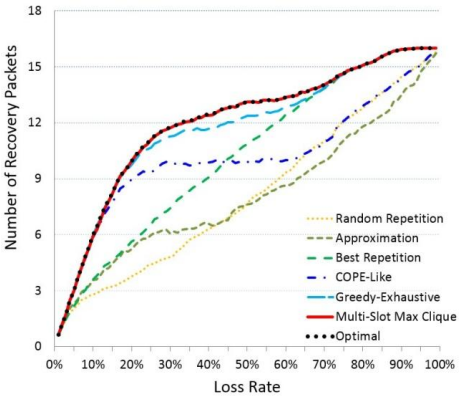

Figure 5 (b)

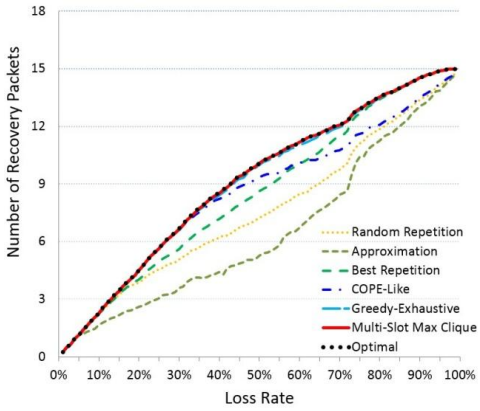

Figure 5 (c) Max Clique and Multi-Slot Max Clique have close to optimal performance while consistently outperforming the baselines when there are 5 users and 5 packets to 15 users and 15 packets. (a) $n = 15$ users, $m = 15$ packets, $t = 1$ slot. (b) $n = 8$ users, $m = 8$ packets, $t = 2$ slots. (c) $n = 5$ users, $m = 5$ packets, $t = 3$ slots.

Figure 6 (a)

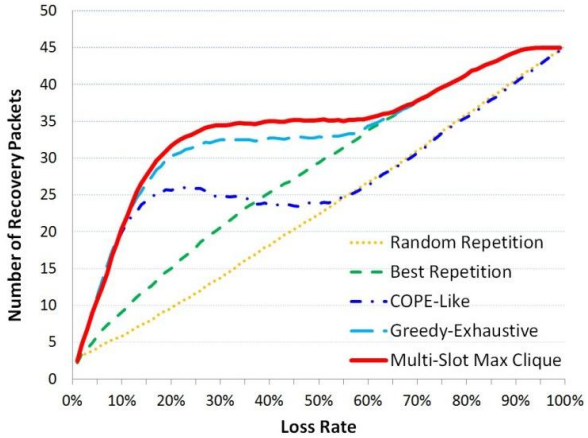

Figure 6 (b)

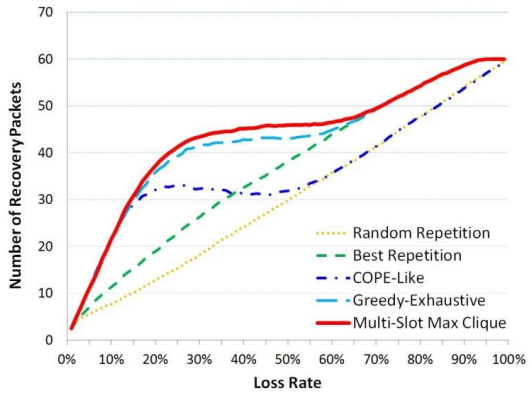

Figure 6 (c) Multi-Slot Max Clique consistently outperforms the baselines when there are 15 users and 15 packets. (a) $n = 15$ users, $m = 15$ packets, $t = 2$ slots. (b) $n = 15$ users, $m = 15$ packets, $t = 3$ slots.(c) $n = 15$ users, $m = 15$ packets, $t = 4$ slots.

omit here due to redundancy. For clarity, we skip plotting the standard deviations, which range from 0 to 3 for all algorithms. *Results:* In Fig. 5, we can observe that both the proposed Max Clique and Multi-Slot Max Clique algorithms perform very close to the Optimal algorithm. In particular, for $t = 1$ in Fig. 5(a), Max Clique has identical performance to the Optimal 91% of the time. For $t = 2, 3$ in Fig. 5(b) and 5(c), Multi-Slot Max Clique has identical performance to the Optimal 50% of the time and 98% of the time within 0.1 of the optimal value. This validates our analysis of the single-slot scenario and demonstrates that the Multi-Slot Max Clique can also find near-optimal solution for the multi-slot scenario.

One can also observe from Fig. 5 that the proposed Max Clique and Multi-Slot Max Clique algorithms consistently and significantly outperform the baselines. In particular, for the case $\{n = 15, m = 15, t = 1\}$, on average, Max Clique performs 1.3 times better than both the Best Repetition and COPE-Like, with the highest improvement over Best Repetition and COPE-Like at 3.3 and 1.6 times, respectively. It also performs up to 1.12 times better than the state-of-the-art Greedy-Exhaustive algorithm, noticeably in the wide range of loss rate from 10% to 60%; note that Greedy-Exhaustive has exponential runtime. We also observe that the Approximation algorithm has poor performance: only slightly better than the Random Repetition in the single slot scenario and worse in the multi-slot scenarios.

In addition, Fig. 6 demonstrates the consistent superior performance of Multi-Slot Max Clique for a larger problem size $\{n = 15, m = 15\}$ and more number of slots $\{t = 2, 3, 4\}$. For instance, for the case $\{n = 15, m = 15, t = 2\}$, , Multi-Slot Max Clique improves by a factor of 1.3 on average over both COPE-Like and Best Repetition, and it performs up to 1.6 times better than the COPE-Like and up to 2.7 times better than Best Repetition.

Two interesting regions can be observed from the curves in both Fig. 5 and Fig. 6, *i.e.*, across all settings:

Repetition Region: When the loss rate is larger than a certain threshold: about 65% for $\{n = 15, m = 15\}$ (and also for $\{n = 20, m = 20\}$ as in [35]), the performance of (Multi-Slot) Max Clique, Best Repetition, Greedy-Exhaustive, and Optimal are the same. This indicates that beyond this point, it is optimal to just send uncoded packet(s). This is because there are very likely uncoded packets that are missed by many users due to high loss rate. Thus, we refer to this region as the *Repetition* region.

Easy Coding Region: When the loss rate is less than another threshold: about 10% for $\{n = 15, m = 15\}$ (or 5% for $\{n = 20, m = 20\}$ [35]), the performance of (Multi-Slot) Max Clique, Greedy-Exhaustive, COPE-Like, and Optimal are very similar. We look into this region carefully in our simulation, and we find that for this low loss rate region, best coded packets typically involve many uncoded packets but there are many opportunities to code. Thus, they can be easily found with a greedy approach. Furthermore, they typically have similar benefit. Consequently, all the coding schemes investigated here have similar performance that is close to the Optimal. We refer to this region as the *Easy Coding* region.

Best Practice: The above observations suggest that it would be beneficial to figure out if an application operates in either of the Repetition or Easy Coding regions, *e.g.*, by applying multiple algorithms described here and monitoring their performance for a short period of time. Then for the best performance and speed, one could use the simple Best Repetition for the Repetition region or COPE-Like for the Easy Coding region as both of these simple algorithms run in just linear time. Outside of these regions, Multi-Slot Max Clique is the best candidate.

Figure 7. MicroPlay networking model: One phone acts as the WiFi access point and as the game server. This phone uses WiFi broadcast to disseminate its game commands.

Commercial systems, like CrowdWiFi [3], could readily get significant improvement in both speed and performance by replacing their existing coding algorithms, *e.g.*, Greedy-Exhaustive, with Multi-Slot Max Clique or a combination of algorithms as described above.

B. Trace-Based Evaluation

In this section, we evaluate the performance of Max Clique in comparison with the baselines, Best Repetition and COPE-Like, using real network traces of an Android application called Racer [1]. Racer is a real-time multi-player racing game implemented on top of a networking framework, called MicroPlay, that we previously developed [1]. MicroPlay exploits wireless broadcast to disseminate input commands from one player to the rest in a timely manner to support accurate game rendering and low latency.

In particular, in Racer, each player's car races around a closed rectangular track and broadcast its movement continuously to the rest of the players. A player uses the broadcast packets to update the positions of the other players' cars. In the context of this work, we examine the packets broadcast by one player, who is acting as the game server and the WiFi access point to the group, depicted in Fig. 7. This scenario we select for evaluation here, in principle, matches the broadcast scenario that we examined earlier in our analysis in Fig. 1.

C. Trace Collection and Description:

We created a Racer game session that has 5 players: 1 server and 4 clients, as shown in Fig. 7. The hardware in use consist of 3 Samsung Captivate and 2 Nexus S phones, all running Android OS

2.3 (Gingerbread). The players are scattered in an on-campus cafeteria, whose area is of sizes approximately 40 x 40 meters. The game session occurs during a busy lunch hour.[2]

Each packet broadcast by the server has a unique ID. We implemented a statistics-collection module within the Racer game client to capture the reception of the packets

broadcast by the server: each client logs the packets it were able to receive and the time it received them. The game session lasted about 15 minutes, and during the game, the server broadcast 19,059 packets, about a packet every 47 ms on average.

The average reception rate of all 4 clients during the game is shown in Fig. 8(a) by the "No Recovery" line. Each point plotted represents the average reception rate of packets broadcast within a 10-second bin. Fig. 8(a) shows that the average reception rate of the clients is high: most of the time above 90%. Nevertheless, there are several instances when the average reception rate drops below 90%, for example, from second 574 to 738. Also, the average reception rate drops as low as 23% at second 811. The reception rates are quite similar across the clients. For this reason, we skip reporting the plots of the individual client rates.

D. Settings:

For each batch of packet of size B, we compute a recovery packet using the Best Repetition, COPE-Like, and Max Clique algorithms with loss rate $p = 12\%$ (average rate we observed previously in the same environment). This recovery packet is to be broadcast at the end of each batch by the server to recover packet losses at the client. For evaluation purposes, we assume that this packet would be successfully received by all the clients. We then compute the new reception rates at the clients for each recovery scheme.

E. Results:

Fig. 8(a) plots the average reception rate when each of the recovery schemes is used for batch size $B = 10$. It could be observed from this figure that Max Clique consistently outperforms the COPE-Like and Best Repetition. In other words, the improvement of the average reception rate is higher when Max Clique is used to compute the recovery packet.

In more details, Fig. 8(b) plots the number of beneficiary users when each of the recovery scheme is used. Each point plotted is the average over multiple recovery packets within a 10-second bin. Fig. 8(b) shows that the recovery packets computed by Max Clique consistently benefit more users: on average, Max Clique helps 16% more users than Best Repetition and 26% more users than COPE-Like. The performance gaps between Max Clique and the baselines are more noticeable when the reception rates are low, *e.g.*, between second 574 and 738, or at second 811, where Max Clique helps 50–250% more users than the others.

We also perform similar evaluation for batches of sizes $B = 5$ and $B = 20$. For $B = 5$, the average performance improvement of Max Clique over Best Repetition is 5% and over COPE-Like is 12%, which are less than those when $B = 10$. This is due to the reduced number of coding opportunities (over just 5 packets). For $B = 20$, the average performance improvement of Max Clique over Best Repetition is 12% and over COPE-Like is 28%, which are similar to those when $B = 10$. This implies that $B = 10$ creates sufficient coding opportunities for the loss rates of this set of traces.

Finally, unlike the numerical results reported in the previous section, Fig. 8 shows that Best Repetition consistently outperforms COPE-Like. This is likely due to the dependency of the packet losses at the clients: a packet lost at a client is likely to be lost at other clients, which implies that re-sending this packet might benefit many clients. This also occurs when $B = 5$ and $B = 20$.

VIII. CONCLUSION

In this paper, we study packet loss recovery in wireless broadcast for real-time applications, namely video streaming and multi-player games.

First, we investigate the scenario where the source has a single time slot to broadcast a single recovery packet. We formulate the Real-Time IDNC problem, which seeks to compute a recovery packet that is immediately beneficial to the maximum number of users. We show that Real-Time IDNC is equivalent to Unique Coverage [11], which is hard to approximate. We then analyze the Random Real-Time IDNC, where each user is assumed to lose every packet with the same probability independently. When the number of packets is polynomial in the number of users, we show that the optimal packet could be computed in polynomial time in the number of users w.h.p. We provide an explicit algorithm, called Max Clique, to find the optimal packet w.h.p.

Second, we consider the scenario where the source has a small constant number of time slots to send multiple instantly decodable recovery packets, which is at least as hard to approximate as Real-Time IDNC. For the probabilistic version of the multi-slot problem, we propose Multi-Slot Max Clique, a polynomial-time algorithm that is developed based on Max Clique and provably finds a near-optimal solution w.h.p. when the number of packets is sufficiently large.

Finally, we evaluate the proposed algorithms numerically (via simulation) as well as experimentally (based on real

network traces). The results demonstrate that (i) the proposed algorithms perform very close to the optimal, and (ii) they consistently and significantly outperform all the state-of-the-art baselines.

IX. ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers who suggested the connection between the Real-Time IDNC and the Unique Coverage problems and provided many thoughtful comments that help them to improve this paper.

References

[1] A. Le, L. Keller, C. Fragouli, and A. Markopoulou, "MicroPlay: A networking framework for local multiplayer games," in *Proc. ACM SIGCOMM Workshop Mobile Gaming*, Helsinki, Finland, 2012, pp. 155–160.

[2] *CrowdWiFi Streaming*, Streambolico, Porto, Portugal, 2016.

[3] D. Ferreira, R. A. Costa, and J. Barros, "Real-time network coding for live streaming in hyper-dense WiFi spaces," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 773–781, Apr. 2014.

[4] S. Sorour and S. Valaee, "Adaptive network coded retransmission scheme for wireless multicast," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seoul, South Korea, Jun./Jul. 2009, pp. 2577–2581.

[5] S. Sorour and S. Valaee, "On minimizing broadcast completion delay for instantly decodable network coding," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Cape Town, South Africa, May 2010, pp. 1–5.

[6] S. Sorour and S. Valaee, "Minimum broadcast decoding delay for generalized instantly decodable network coding," in *Proc. IEEE Global Commun. Conf. (Globecom)*, Miami, FL, USA, Dec. 2010, pp. 1–5.

[7] S. Sorour and S. Valaee, "Completion delay minimization for instantly decodable network coding with limited feedback," in *Proc. IEEE Int. Conf. Commun.*, Kyoto, Japan, Jun. 2011, pp. 1–5.

[8] S. Sorour and S. Valaee, "Completion delay reduction in lossy feedback scenarios for instantly decodable network coding," in *Proc. IEEE Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Toronto, ON, Canada, Sep. 2011, pp. 2025–2029.

[9] S. Sorour and S. Valaee, "On densifying coding opportunities in instantly decodable network coding graphs," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Boston, MA, USA, Jul. 2012, pp. 2456–2460.

[10] N. Aboutorab, S. Sorour, and P. Sadeghi, "O2-GIDNC: Beyond instantly decodable network coding," in *Proc. IEEE Int. Symp. Netw. Coding (NetCod)*, Calgary, AB, Canada, Jun. 2013, pp. 1–6.

[11] E. D. Demaine, M. T. Hajiaghayi, U. Feige, and M. R. Salavatipour, "Combination can be hard: Approximability of the unique coverage problem," in *Proc. ACM-SIAM Symp. Discrete Algorithm (SODA)*, Miami, FL, USA, 2006, pp. 162–171.

[12] L. Keller, E. Drinea, and C. Fragouli, "Online broadcasting with network coding," in *Proc. IEEE Int. Symp. Netw. Coding (NetCod)*, Hong Kong, Jan. 2008, pp. 1–6.

[13] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive network coding for broadcast channels," in *Proc. IEEE Int. Symp. Netw. Coding (NetCod)*, Lausanne, Switzerland, Jun. 2009, pp. 80–85.

[14] S. Katti *et al.*, "XORs in the air: Practical wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 497–510, Jun. 2008.

[15] X. Li, C.-C. Wang, and X. Lin, "On the capacity of immediately-decodable coding schemes for wireless stored-video broadcast with hard deadline constraints," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1094–1105, May 2011.

[16] Y. Birk and T. Kol, "Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2825–2830, Jun. 2006.

[17] S. Y. El Rouayheb, M. A. R. Chaudhry, and A. Sprintson, "On the minimum number of transmissions in single-hop wireless coding networks," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Lake Tahoe, CA, USA, Sep. 2007, pp. 120–125.

[18] H. Maleki, V. Cadambe, and S. Jafar, "Index coding: An interference alignment perspective," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Boston, MA, USA, Jul. 2012, pp. 2236–2240.

[19] A. S. Tehrani and A. G. Dimakis, "Finding three transmissions is hard," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Anaheim, CA, USA, Dec. 2012, pp. 2317–2322.

[20] S. Brahma and C. Fragouli, "Pliable index coding," in *Proc. IEEE Int. Symp. Inf. Theory*, Boston, MA, USA, Jul. 2012, pp. 2251–2255.

[21] S. Brahma and C. Fragouli, "Pliable index coding: The multiple requests case," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 1142–1146.

[22] M. Langberg and A. Sprintson, "On the hardness of approximating the network coding capacity," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Toronto, ON, Canada, Jul. 2008, pp. 315–319.

[23] M. A. R. Chaudhry and A. Sprintson, "Efficient algorithms for index coding," in *Proc. IEEE Int. Conf. Comput. Commun. Workshops (INFOCOM)*, Phoenix, AZ, USA, Apr. 2008, pp. 1–4.

[24] A. S. Tehrani, A. G. Dimakis, and M. J. Neely, "Bipartite index coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Boston, MA, USA, Jul. 2012, pp. 2246–2250.

[25] S. El Rouayheb, A. Sprintson, and P. Sadeghi, "On coding for cooperative data exchange," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Cairo, Egypt, Jan. 2010, pp. 1–5.

[26] A. Sprintson, P. Sadeghi, G. Booker, and S. El Rouayheb, "A randomized algorithm and performance bounds for coded cooperative data exchange," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Austin, TX, USA, Jun. 2010, pp. 1888–1892.

[27] A. Sprintson, P. Sadeghi, G. Booker, and S. El Rouayheb, "Deterministic algorithm for coded cooperative data exchange," in *Proc. ICST Conf. Heterogeneous Netw. Quality, Rel., Secur. Robustness (QShine)*, Houston, TX, USA, 2010, pp. 282–289.

[28] N. Milosavljevic, S. Pawar, S. El Rouayheb, M. Gastpar, and K. Ramchandran, "Deterministic algorithm for the cooperative data exchange problem," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, St. Petersburg, FL, USA, Jul./Aug. 2011, pp. 410–414.

[29] M. Gonen and M. Langberg, "Coded cooperative data exchange problem for general topologies," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Boston, MA, USA, Jul. 2012, pp. 2606–2610.

[30] N. Milosavljevic, S. Pawar, S. El Rouayheb, M. Gastpar, and K. Ramchandran, "Data exchange problem with helpers," in *Proc. IEEE Int. Symp. Inf. Theory*, Boston, MA, USA, Jul. 2012, pp. 2611–2615.

[31] D. Ozgul and A. Sprintson, "An algorithm for cooperative data exchange with cost criterion," in *Proc. Inf. Theory Appl. Workshop (ITA)*, San Diego, CA, USA, Feb. 2011, pp. 1–4.

[32] T. A. Courtade, X. Bike, and R. D. Wesel, "Optimal exchange of packets for universal recovery in broadcast networks," in *Proc. Military Commun. Conf.*, San Jose, CA, USA, 2010, pp. 2250–2255.

[33] T. A. Courtade and R. D. Wesel, "Efficient universal recovery in broadcast networks," in *Proc. Allerton Conf. Commun., Control Comput. (Allerton)*, Champaign, IL, USA, 2010, pp. 1542–1549.

[34] T. A. Courtade and R. D. Wesel, "Weighted universal recovery, practical secrecy, and an efficient algorithm for solving both," in *Proc. Allerton Conf. Commun., Control Comput. (Allerton)*, Champaign, IL, USA, 2011, pp. 1349–1357.

[35] A. Le, A. S. Tehrani, A. G. Dimakis, and A. Markopoulou, "Instantly decodable network codes for real-time applications," in *Proc. IEEE Int. Symp. Netw. Coding*, Calgary, AB, Canada, Jun. 2013, pp. 1–6.

[36] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.

[37] N. Alon and J. H. Spencer, *The Probabilistic Method*. New York, NY, USA: Wiley, 1992.

[38] A. Juels and M. Peinado, "Hiding cliques for cryptographic security," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, San Francisco, CA, USA, 1998, pp. 678–684.

[39] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb. 2009.

[40] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and W. Xu, "Raptor codes for reliable download delivery in wireless broadcast systems," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2006, pp. 192–197.

[41] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. ACM SIGCOMM*, Vancouver, BC, Canada, 1998, pp. 56–67.