The Report Committee for Timothy Austin Mahler
Certifies that this is the approved version of the following report:

Enabling Decentralized Wireless Index Coding in Practice

APPROVED BY
SUPERVISING COMMITTEE:

Sriram Vishwanath, Supervisor

Jeffrey Mahler

Enabling Decentralized Wireless Index Coding in Practice

by

Timothy Austin Mahler

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2018

Acknowledgements

I would like to thank my advisor Dr. Sriram Vishwanath for his advice and guidance over the course of the project as well as his endless support and friendship throughout my time in graduate school. I would also like to thank my mother Cheryl Thalmann, father Kenneth Mahler, step brother Ian Pelkey, and older brother Jeffrey Mahler for their encouragement and support during my graduate studies.

Abstract

Enabling Decentralized Wireless Index Coding in Practice

Timothy Austin Mahler, M.S.E.

The University of Texas at Austin, 2018

Supervisor: Sriram Vishwanath

Index coding is a problem in theoretical computer science and network information theory that studies the optimal coding scheme for transmitting multiple messages across a network to receivers with different side information. The ultimate goal of index coding is to reduce transmission time in a communication network by minimizing the number of messages based on shared information. Index coding theory extends to several key engineering problems in network communication including peer to peer communication, distributed broadcast networks, and interference alignment. Although the theoretical connection between index coding and wireless networks is valuable, we focus on finding index coding strategies for a realistic wireless network. More specifically, we investigate how index coding can be applied to an OFDMA downlink network during the retransmission phase. An orthogonal frequency-division multiple access (OFDMA) downlink network is a network where data is sent downward from a designated higher-level transmitter to a group of receiving nodes. In addition, receivers can often decode the other receivers' physical layer signals on the other sub-channels that can be exploited as

side information. If this side information is sent back to the transmitter, it can then be coded to cancel the interference in subsequent retransmission phases resulting in fewer retransmission messages. In this report, we explain the coding model and characterize the benefits of index coding for retransmissions within an OFDMA downlink network. In addition, we demonstrate the results of applying this index coding scheme in such network in both simulation and in an active wireless mesh network.

Table of Contents

List of Tables

List of Figures

INTRODUCTION

Motivation

Using index coding with side information was first introduced by Birk and Kol using their solution called Informed-Source Coding-On-Demand (ISCOD) to efficiently supply non-identical data from a central server to multiple clients through a broadcast channel [1, 2]. It was inspired by applications like broadcasting a daily newspaper where a sending server has to deliver multiple sets of data, audio, and/or video packets to a set of receiving clients each requesting a different data set. Prior to transmission, each client has some files in their possession already from a previous transmission. Using a slow backward channel, the clients return the information back to the sender regarding which data they already have and what data they still need. ISCOD uses the joint exploitation of the cached data on each client, the server's knowledge of what already has been sent, and what the client is requesting in order to reduce the number of remaining transmissions to each client via index coding. More specifically, their two-phase ISCOD algorithm assembles ad-hoc error-correction sets based on each client's cached information and the items requested to create error-correction codes to construct the data packet to be transmitted. Each client uses the cached data they already have combined with the coded data packet to derive the data it originally requested.

Building upon this general coding scheme, other previous works have applied index coding in the wireless domain. First off, the authors H. Maleki et al. in "Index coding: An interference alignment perspective" [3] and S. Jafar in "Topological interference management through index coding" [4] demonstrate a direct link between index coding and interference alignment. They argue

that the index coding problem is the simplest multiuser capacity problem because it can optimize a communication network that has only one link with finite capacity. Consider the example of a wireless bottleneck network show in Figure 1 compromised of source nodes on left and destination nodes on the right. Index coding normalizes the bottleneck link capacity to unity so that all transmission rates are measured as multiples of the bottleneck. Furthermore, index coding normalizes the number of signal dimensions available to the bottleneck receiver where all degrees of freedom are measured as multiples of the bottleneck degrees of freedom resulting in an optimal capacity. The authors show that the solution using index coding to solve a single receiver bottleneck network also applies to a wide class of interference alignment problems where an optimal capacity vector coding solution can be translated to the wireless medium [3].

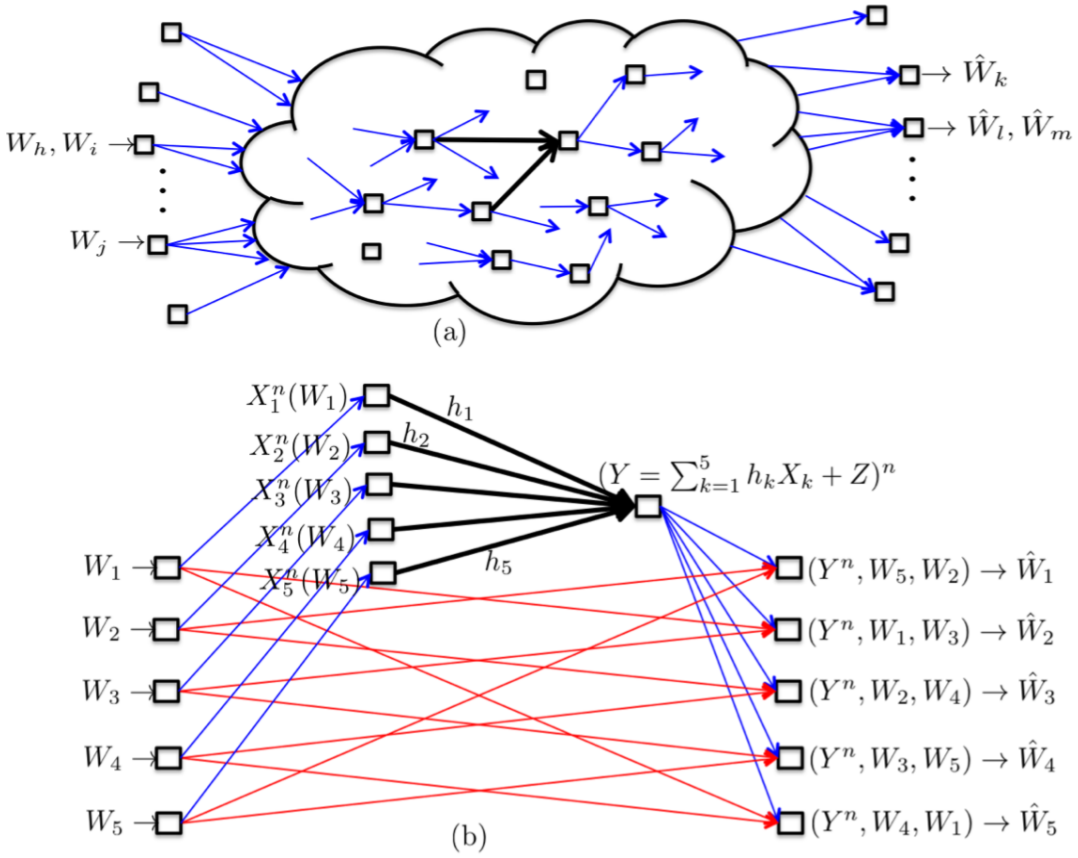


Figure 1: (a) Wireless network: If only one receiver (shown with incoming signals in black) in the intermediate network has non-zero (unit) variance, and all the other receivers have zero noise (infinite capacity) (b) Wireless index coding setting [3].

In addition, there have been several examples of index coding applied directly to downlink networks. The authors J. I. Tamir et al. in the paper "Wireless index coding through rank minimization" set out to achieve efficient communication over a multi-way interchange wireless network as seen in Figure 2 [5]. The network consists of a primary access point connected to multiple wireless terminals. Their algorithm shows the benefits of structured combinations of data using index coding versus previous works of using random network coding (RNC). In RNC,

network coding is performed in terms of choosing random coefficients for linear combinations of data packets that are subsequently transmitted. Instead, the authors use a non-static linear combination coding of packets as a transmission strategy. More specifically, they use a greedy rank minimization framework that exploits knowledge of those packets that are successfully received by destinations within the network when transmissions may be corrupted and therefore erased. They show that this strategy provides up to twice the throughput of RNC.

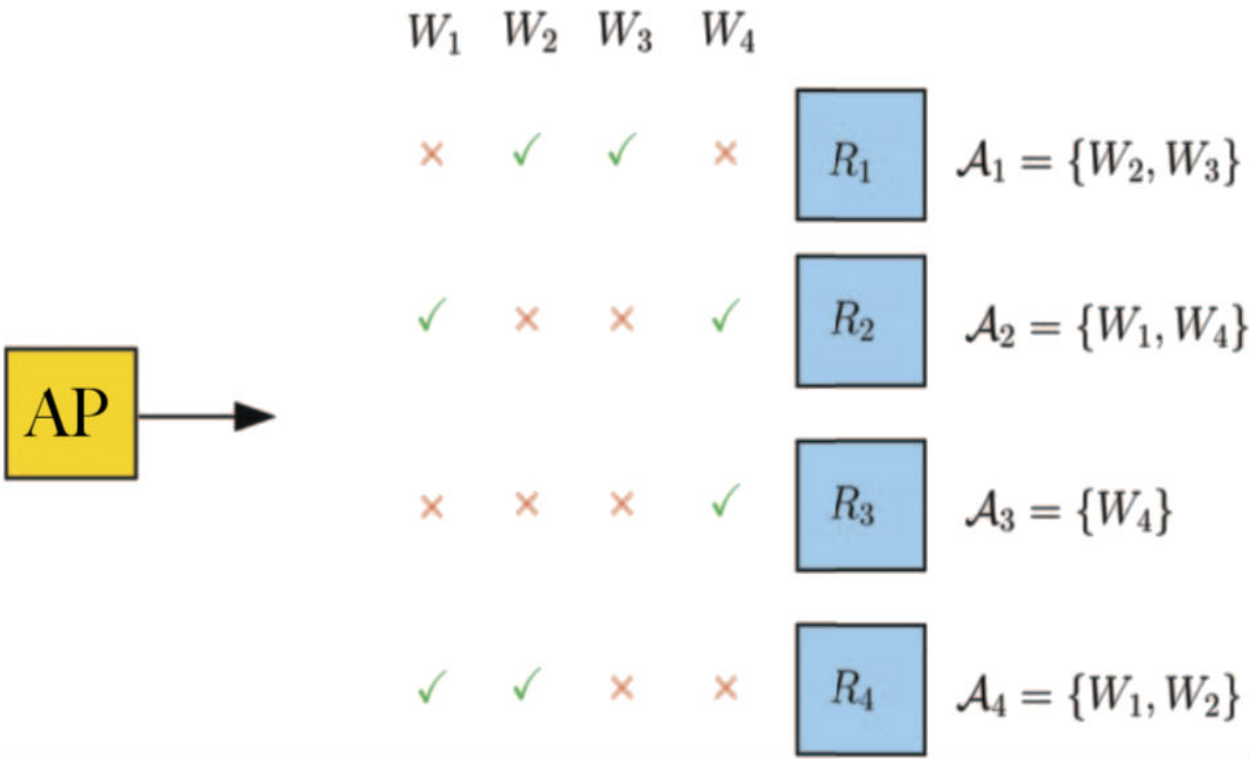


Figure 2: Access point to K=4 terminals using time-division multiplexing schemes [5]. Check means receiver r received message W_i and 'X' indicates an erasure.

Contribution

All of these related works provide the basis for the following algorithm described and implemented in this report. More specifically, the paper "Index-Coded Retransmission for OFDMA Downlink" by M. Kim et al. [6], which builds upon [5] by using a more complex network topology using side information at each node, describes the algorithm used by our active wireless mesh network we describe in this report. Essentially, this report is an extension of the work done in [6].

INDEX CODING ALGORITHM

System Model

The system model considered by the algorithm consists of an OFDMA downlink transmission network where a base access point transmitter sends signals to K receivers on L parallel frequency subchannels. Each K active receiver is scheduled on a frequency subchannel. In an OFDMA downlink network receivers can decode the other receivers' physical layer signals. The main idea is that these signals from the other receivers can be used as side information to cancel the interference in subsequent retransmission cycles. The transmitter sends $x = [x_1, x_2, x_3, \dots, x_L]^T$. At receiver k, the received signal is [6]:

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x} + \mathbf{z}_k \quad (1)$$

The diagonal matrix $H = \text{diag}(h_{k,1}, h_{k,2}, h_{k,3}, \dots, h_{k,L})$ where $h_{k,l}$ is the set of all $h_{k,L}$ for $k = 1, 2, 3, \dots, K$ and $l = 1, 2, 3, \dots, L$. The matrix H_k is realized once in each time frame and the diagonal elements of H_k are independent and identically distributed (i.i.d). The vector $z_k = [z_{k,1}, z_{k,2}, z_{k,3}, \dots, z_{k,L}]^T$ where $z_{k,l}$ represents the noise as an independent and identically distributed circularly symmetric complex Gaussian, $\mathcal{CN}(0,1)$. Each subchannel is sent at a different time slot scheduled for a receiver. One subchannel and one-time frame at time t is the basic unit for multi-user scheduling where $t \epsilon \{1, 2, 3, \dots, n\}$. Each transmit signal is subject to average power constraints $\sum_{t=1}^{n} |x_l[t]^2| = nP_l$ and sum-power constraint $\sum_{l=1}^{L} P_l = P$ [6].

We assume perfect channel state information (CSI) is available at the receivers, but not at the transmitter. Suppose receiver k with code rate $R_k$is scheduled on subchannel l with power

allocation P_l. The erasure probability, or probability that transmitted bits are dropped without the receiver knowing, at receiver k on subchannel l with erasure random variable $\epsilon_{k,l} = \{0,1\}$, is [6]:

$$\epsilon_{k,l} = \mathbb{P}\{\varepsilon_{k,l} = 1\} = \mathbb{P}\left\{|h_{k,l}| < \sqrt{\frac{2^{R_k} - 1}{P_l}}\right\}. \qquad (2)$$

The effected spectral efficiency for receiver k on subchannel l is defined as R_k weighted by success probability [6]:

$$\rho_{k,l} = R_k(1 - \epsilon_{k,l}) = R_k \exp\left(-\frac{2^{R_k} - 1}{\text{SNR}_{k,l}}\right) \qquad (3)$$

where $p_{k,l}$ is bits per channel use where the Signal-to-Noise Ratio $SNR_{k,l} = g_{k,l} P_l$ is determined by the power allocation $(P_1, P_2, P_3, \dots, P_L)$.

Index-Coded Retransmission

The following retransmission algorithm and index coding sections follow M. Kim et al. verbatim [6]. We consider a binary message vector for receiver k as $w_k \in \{0,1\}^{nR_k}$ (nR_k is an integer). This message $W = [w_1, w_2, w_3, \dots, w_K]^T$ is sent for each receiver for each subchannel at a given timeslot during transmission. Next, we will describe the retransmission algorithm, index coding scheme, and the reconstruction phase at each receiver.

Retransmission Algorithm

The retransmission algorithm follows the same design presented in the multi-way interchange wireless network rank minimization algorithm in [5]. For the time frame m the receivers are in the group $\mathcal{U}_{m-1} \subseteq \{1, 2, \dots, K\}$ if the desired messages of the receivers were not received (erased) in time step $m-1$ and are scheduled for retransmission in time step m. In the simplest approach, one would simply resend all of the dropped messages on each separate subchannel for a total of $g = |\mathcal{U}_{m-1}|$ subchannels. However, resending each dropped message is far from optimal. Our approach is to use index coding to send a linear combination of messages, called equations, given the side information stored at each receiver on $r < q$ subchannels. We only consider $q \geq 2$ given that $q = 1$ is trivial.

If the message for receiver k is dropped on subchannel $l = k$ in the time frame $m-1$ then the receiver looks at the other subchannels $l \neq k$ and collects the other messages as side information. Receiver k then sends a packet back to the transmitter containing the list of side information it received $\mathcal{A}_{k,m-1} \subseteq \{1, 2, 3, \dots, K\} \setminus \{k\}$. This list contains the ids of the other receivers' messages it received, not the actual messages themselves. This vector is best expressed as a matrix $\mathcal{A}_k$ where each row is a message received with a 1 in the k^{th} position of the id of the receiver whose message it is. Next, we will describe two basic examples that demonstrate the algorithm.

Consider the basic example (Example #1) on a cellular downlink network seen in Figure 3 with 1 primary transmitter and 3 receivers with $\mathcal{U}_{m-1} = \{1, 2, 3\}$ where all 3 receivers do not

receiver their own messages. Instead, receiver 1 sees receiver 2's message, receiver 2 sees both receiver 1 and receiver 3's messages, and receiver 3 sees receiver 1's message. In other words:

- Receiver 1: $\mathcal{A}_{1,m-1} = \{2\}$ or $\mathcal{A}_1 = [0 \quad 1 \quad 0]$
- Receiver 2: $\mathcal{A}_{2,m-1} = \{1,3\}$ or $\mathcal{A}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Receiver 3: $\mathcal{A}_{3,m-1} = \{1\}$ or $\mathcal{A}_3 = [1 \quad 0 \quad 0]$

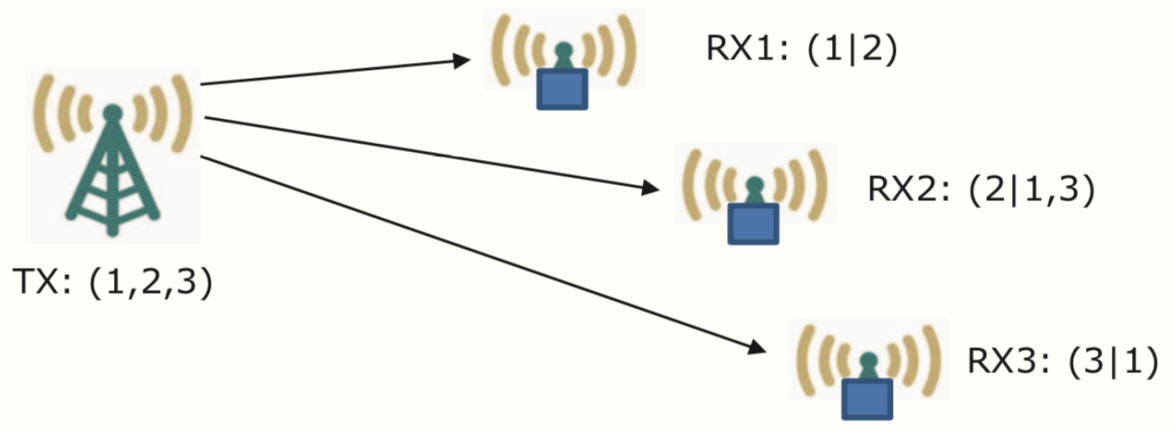


Figure 3: Cellular downlink network with 1 transmitter and 3 receivers [6].

The receiver side channel graph can be seen in Figure 4. Given the side information for each receiver, the transmitter can then construct a $K \; x \; K$ dashboard matrix F. The dashboard matrix F is constructed as follows:

- The (k, k) position is set to 1 if receiver k decoded their own message.
- The $(k, l \neq k)$ position is set to $(*)$ if receiver k decoded the message w_l successfully. Meaning, receiver k received the message successfully for receiver l.
- The (k, l) position is set to 0 if receiver k did not decode the message w_l.

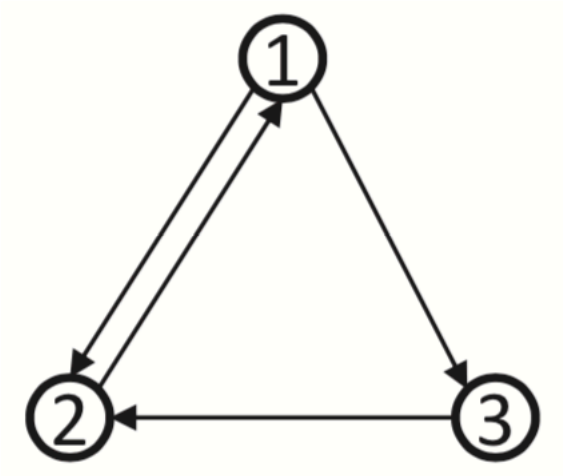


Figure 4: Side information graph for cellular downlink network above [6].

Given the example above, the transmitter would construct the dashboard matrix F given the side information as follows:

$$F = \begin{bmatrix} 0 & * & 0 \\ * & 0 & * \\ * & 0 & 0 \end{bmatrix}$$

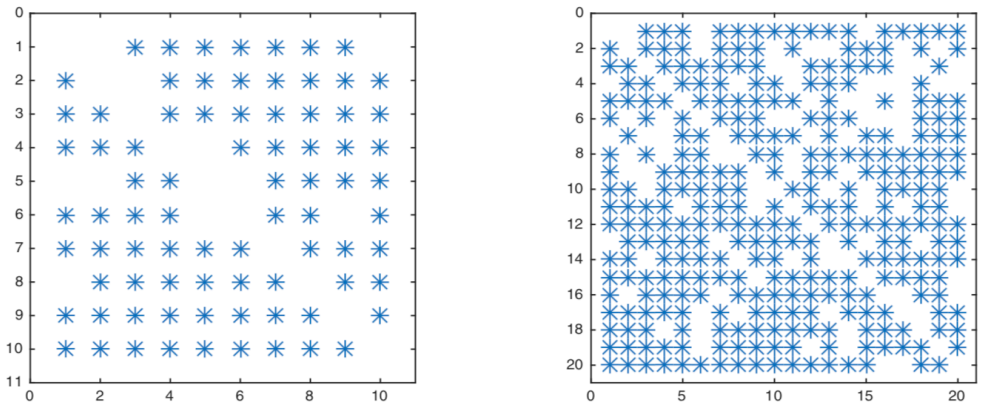
The next example (Example #2) shows how the constructed dashboard matrix F can be reduced to a sub matrix $F_{\mathcal{U}_{m-1}}$ by removing the k^{th} row when receiver k receives its own message (position (k, k) is set to 1). Given the dashboard matrix:

$$F = \begin{bmatrix} 1 & 0 & 0 & * & 0 \\ 0 & 0 & * & 0 & 0 \\ 0 & * & 0 & 0 & * \\ 0 & 0 & * & 1 & * \\ * & * & 0 & 0 & 0 \end{bmatrix}$$

From matrix F above we can see that both receiver 1 and receiver 4 received their desired message given the k^{th} row is from receiver k. We can construct a $q\ x\ q$ submatrix $F_{\mathcal{U}_{m-1}}$ by removing the k^{th} row and k^{th} column if position $(k,k) = 1$. Basically, since receiver 1 and receiver 4 received their desired message, we can remove them completely from the dashboard matrix by removing row 1, column 1, row 4, and column 4. Thus, the new matrix $F_{\mathcal{U}_{m-1}}$ is:

$$F_{\mathcal{U}_{m-1}} = \begin{bmatrix} 0 & * & 0 \\ * & 0 & * \\ * & 0 & 0 \end{bmatrix}$$

More examples of generated dashboard matrices can be seen below in Figure 5 with randomly generated $F_{\mathcal{U}_{m-1}}$ where $K = 100$ and the probability of erasure $\epsilon_{k,l} = \epsilon\ \forall\ k, l$ [6]. The expected dimension $q\ x\ q$ of $F_{\mathcal{U}_{m-1}}$ is thus $\mathbb{E}[q] = \epsilon K$. By observing the matrices in Figure 5 we can acquire a lot of information about ϵ and can see how the dimension size q reflects the erasure probability.



(a) (b)

Figure 5: Randomly generated dashboard matrices $F_{\mathcal{U}_{m-1}}$ where $K = 100$. (a) $\epsilon = 0.1$ and thus $q = 10$. (b) $\epsilon = 0.2$ and thus $q = 20$ [6].

Index Coding

Given the constructed dashboard matrix $F_{\mathcal{U}_{m-1}}$ by the transmitter from the side information in time step m as described above, the next step is to generate index codes that represent linear combinations of messages that will be sent in the subsequent retransmission phase [6]. Without index coding, we would need $q = |\mathcal{U}|$ subchannels to retransmit q messages. The ultimate goal is to reduce the number of retransmission messages from q to the r, where r is the minimum rank. Thus, the first step is to find a solution to the rank minimization problem using the side information dashboard matrix $F_{\mathcal{U}_{m-1}}$. Again, the index coding algorithm follows [6] verbatim.

First, we consider the side information graph $\mathcal{G}$ for $\mathcal{U} = \{1, 2, \dots, q\}$, defined by $\mathcal{G} = \{(1|\mathcal{A}_1), (2|\mathcal{A}_2), \dots, (q|\mathcal{A}_q)\}$ which is equivalent to $F_{\mathcal{U}_{m-1}}$. The input of the rank minimization algorithm is $\mathcal{A}_{\mathcal{G}} = \mathrm{I} + F_{\mathcal{U}_{m-1}}$ which is $q\ x\ q$ and outputs a matrix $\mathcal{B}_{\mathcal{G}}$ also of size $q\ x\ q$ with $(*)$ positions filled in. More specifically, the rank minimization algorithm is [6]:

$$\mathrm{minrk}_2(\mathcal{G}) = \min_{\mathcal{B} \in \{0,1\}^{qxq}} \{\, \mathrm{rank}(\mathcal{B}) : [\mathcal{B}]_{ij} = [\mathcal{A}_{\mathcal{G}}]_{i,j}\ if\ [\mathcal{A}_{\mathcal{G}}]_{i,j} \neq * \,\}$$

The solution of this optimization problem results in a set of optimal scalar binary linear index codes. More specifically, these index codes are the resulting rows of matrix $\mathcal{B}$ after the rank minimization algorithm concludes. We further reduce matrix $\mathcal{B}$ to matrix **G** by taking the r linearly independent rows of $\mathcal{B}$. We then use matrix **G** to create a linear combination of messages with the original message vector $W = [w_1,\ w_2,\ w_3, \dots, w_K]^T$, which we call equation matrix **S**. Lastly, we use **S** to create retransmission matrix **X**, $\mathbf{X} = f(\mathbf{S})$, where $f_i(\cdot)$ are the encoding functions for error correction coding. We send matrix **X** along with coefficient matrices

$\mathcal{B}$ and **G**. These matrices become negligible if messages w_k are sufficiently large. In time frame m we use r subchannels for retransmission.

To demonstrate this index coding step, we continue with Example #1 describe above. The input to our rank minimization algorithm is $\mathcal{A}_{\mathcal{G}} = \mathrm{I} + F_{\mathcal{U}_{m-1}}$ with $F_{\mathcal{U}_{m-1}} = F$. The resulting matrix $\mathcal{B}$ from the minimization rank algorithm results in:

$$\mathcal{B} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

with $r = \mathrm{rank}(\mathcal{B}) = \mathrm{minrk}_2(\mathcal{G}) = 2$. Taking only the $r = 2$ linearly independent rows we get matrix **G**:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Next, we calculate matrix **S**, the equation matrix, by taking the linear combination $\mathbf{S} = \mathbf{GW}$:

$$\mathbf{S} = \begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix} = \mathbf{G} \begin{bmatrix} w_1^T \\ w_2^T \\ w_3^T \end{bmatrix} = \mathbf{GW} = \begin{bmatrix} w_1^T + w_2^T \\ w_2^T + w_3^T \end{bmatrix}$$

Lastly, we create matrix **X** by encoding matrix **S** for error correction coding:

$$\mathbf{X} = \begin{bmatrix} f_1(s_1)^T \\ f_2(s_2)^T \\ f_3(s_3)^T \end{bmatrix}$$

Matrix **X** is sent along with matrices $\mathcal{B}$ and **G**. Receiver k in $\mathcal{U}$ first decodes all the equations $\{s_i\}_{i \in \mathcal{M}_k}$ that involves its desired message w_k and then solves the equations for the message ($\mathcal{M}_k$ is the set of retransmission subchannels). The equation s_i can be recovered if $f(s_i)$ is decoded without error. The receiver can thus calculate **BW** from **GW** by combining $[\mathbf{BW}]_k$ and $\mathcal{A}_k\mathbf{W}$ where $[\cdot]_k$ is the k^{th} row of the matrix. Thus, the receivers solve as follows:

- At receiver 1:

$$[\mathbf{BW}]_1 + \mathcal{A}_1\mathbf{W} = (\mathrm{w}_1^T + \mathrm{w}_2^T) + \mathrm{w}_2^T = \mathrm{w}_1^T$$

- At receiver 2:

$$[\mathbf{BW}]_2 + [\mathcal{A}_2\mathbf{W}]_2 = (\mathrm{w}_2^T + \mathrm{w}_3^T) + \mathrm{w}_3^T = \mathrm{w}_2^T$$

- At receiver 3:

$$[\mathbf{BW}]_3 + \mathcal{A}_3\mathbf{W} = (\mathrm{w}_1^T + \mathrm{w}_3^T) + \mathrm{w}_1^T = \mathrm{w}_3^T$$

This final step can be done using Gaussian Elimination [7] using matrices **X,** $\mathcal{B}$, and **G**. Thus, at the end of this step all receivers k have received their desired messages w_k. In the following section, we discuss the implementation of this index coding algorithm during the retransmission phase of a real OFDMA downlink network which is the novelty of this report.

IMPLEMENTATION

Active Mesh Network

Device Setup

Our wireless mesh network consists of six Android devices, one transmitter and five receivers each labeled accordingly (see Figure 6 below). More specifically, we use the LG Nexus 5 running Android Marshmallow 6.0.1. Each device runs our developed Android application that sets up the connection between all devices in the network and configures / runs each index coding test. In order to communicate between devices in our mesh network, the Android application leverages the M87 Proximity Software Development Kit (SDK) [8]. We chose six total devices due to the limitations of the M87 Proximity SDK on a single subchannel.

Figure 6: Our wireless mesh network

M87 Proximity Software Development Kit

The M87 Proximity SDK allows us to setup a mesh network using Wi-Fi Direct. Wi-Fi Direct enables devices to communicate with each other over the standard Wi-Fi channel without a wireless access point. Thus, Wi-Fi Direct is inherently a single hop communication (Ad-Hoc) network. However, the M87 Proximity SDK allows each device (node) to act as a router and retransmit messages on behalf of any device in the network creating a mesh network [8]. Wi-Fi Direct is advantageous because it can send data almost 10 times faster and at distance almost 3 times further than Bluetooth [9].

The M87 Proximity SDK leverages Wi-Fi Direct to enable message broadcasting across a network. More specifically, you can create several subchannels that devices can 'subscribe' to which allows them to 'publish' messages that will be heard by all devices within the subchannel. For simplicity sake, in our mesh network all devices subscribe to the same subchannel 'sam' as seen in Figure 7. As a result, all receivers can communicate with the transmitter, can communicate with each other, and can hear all messages sent over the network. This allows receivers to collect side information in order for us to run our index coding algorithm. Once a device subscribes to the channel 'sam', it must publish a basic message so that all other nodes know it exists in the network (Figure 8). Because nodes can belong to multiple subchannels, each message must contain 'sam' in the metadata. In the following section, we will describe our Android application in more detail and how we can leverage the M87 Proximity SDK to test our index coding algorithm.

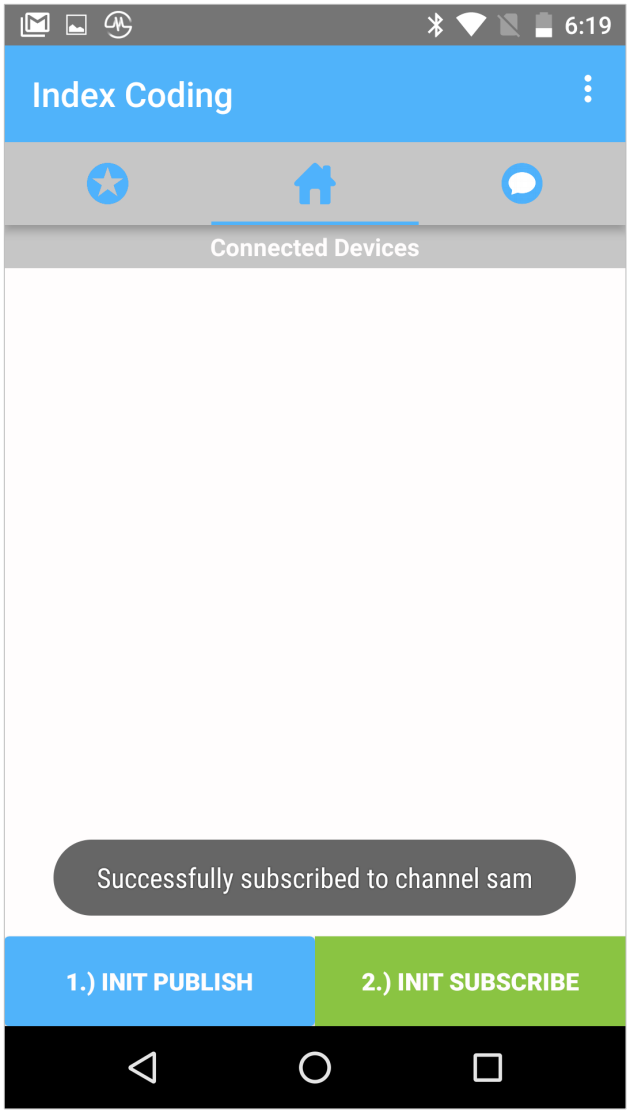


Figure 7: Subscribing to our mesh network: receiving node has subscribed to subchannel 'sam' allowing it to decode all messages sent across the subchannel.

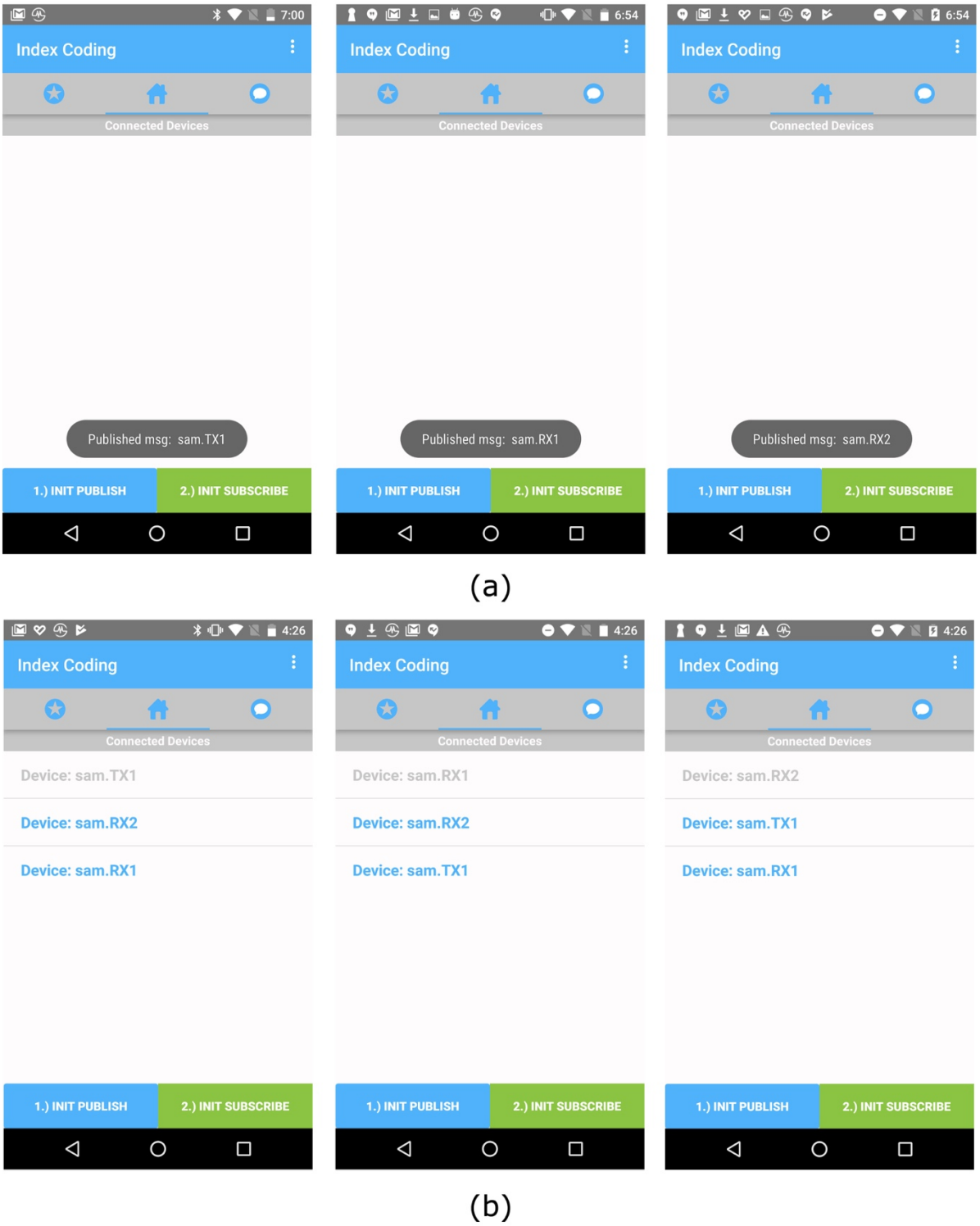


Figure 8: Initialize Mesh Network Connection: 1 transmitter 2 receivers example. (a) all nodes publish an initialization message to the network in order to be discovered. (b) resulting mesh network with devices seeing all each other in network.

Message Construction Scheme:

Once our mesh network has been initialized where all nodes are visible to each other on the same subchannel, we construct our own scheme for structuring messages in order for nodes to efficiently communication allowing our index coding algorithm to work.

First, the M87 Proximity SDK, written in Java, provides classes with core parameters in the *sendMessage()* and *onReceiveMessage()* function handlers. More specifically, the *M87ProximityDevice* and *ProximityMessage* classes include the key fields: ‘deviceId’, ‘transactionId’, ‘messageStatus’, and ‘message’. These fields are essential as they allow us to know which node the message came from as well as map initialization messages to the device label. For example, when a node receives the initialization message:

```
{
   message: 'sam.RX1',
   deviceId: 062a220f200511a45
}
```

the node can map the device ‘062a220f200511a45’ to receiver 1 (RX1) which allows each node to know the role of every other device in the algorithm.

Given these base functions and classes of the M87 Proximity SDK, we designed our own scheme for the ‘message’ field which is a string of maximum length 1000 bytes. Each message contains several fields separated by a ‘.’ that allow our index coding algorithm to work. Below is the list of parameters in order of occurrence:

- source: designates who sent the message. Example ‘RX1’ resembles receiver 1.
- messageType:

    - 0 if an initialization message sent during network setup.
    - 1 if a message that is a part of the index coding test.
- numDevices: number of receiver nodes in the test.
- roundType: round in the transmission cycle.
    - 0 if first transmission of new message **W**.
    - 1 if retransmission message from receiver containing side information.
    - 2 if retransmission message containing index codes from transmitter.
    - 3 if end of test.
- destination: who the message is meant for.
- messageSize: size of the message in bytes.
- messageString: actual bytes of message.
    - The message vector **W** explained in the algorithm if roundType is 0.
    - Side information of node ids if roundType is 1.
    - Index coded matrix **X** if roundType is 2 with rows separated by commas.
- retransmissionMetadata: only sent with a retransmission message from the transmitter. This contains the matrices $\mathcal{B}$ and **G** separated by a '.'. Each row of a matrix is separated by a comma in the message. Lastly, this field contains the row index belonging to the destination receiving node. For example, in Example #2 when the dashboard matrix dimension $q < K$, meaning some receiver nodes successfully decoded their own message in the first round of transmission, the row indices of the resulting index coded matrices no longer represent the device id (row 1 might correspond to receiver node #3).

An example of a message sent by the transmitter in the first round of transmission can be seen in Figure 9 below. The next section describes the rank minimization algorithm used by our Android application in order to construct index codes.

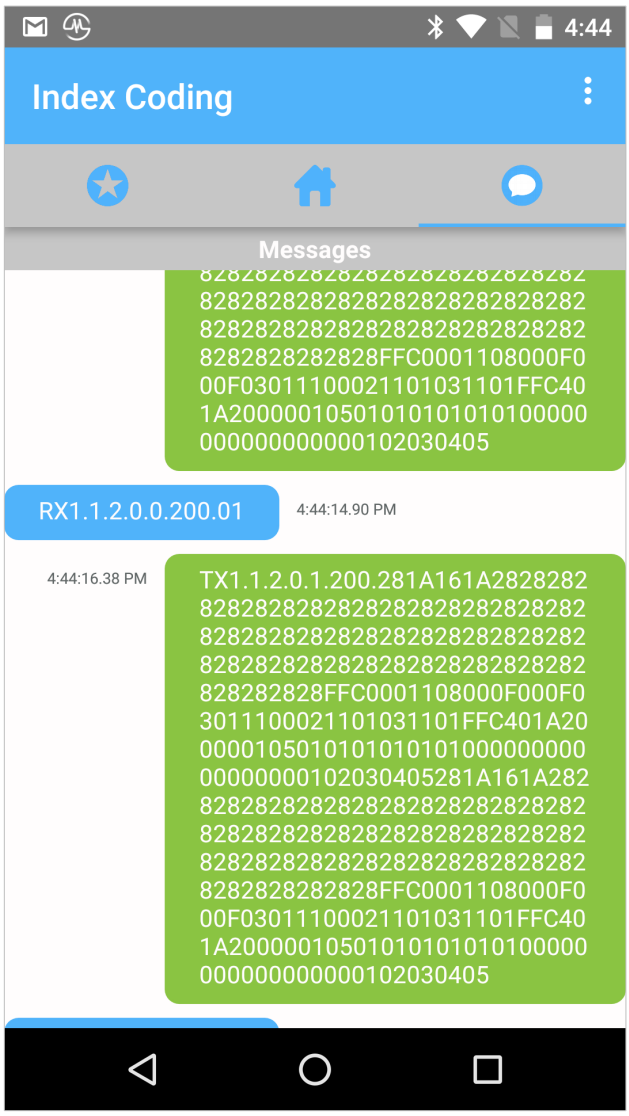


Figure 9: Message Construction Scheme. Above is an example of a message sent by the transmitter (TX1) (green message on bottom right). Given the scheme explained above, the message starting with "TX1.1.2…" is the first round of transmission meant for receiving node 1 containing w_1 of size 200 bytes.

Rank Minimization Algorithm

Once the transmitter receives all acknowledge messages from the receivers containing the side information after the first round of transmission, the transmitter constructs matrix $\mathcal{A}_{\mathcal{G}}$ from the dashboard matrix $F_{\mathcal{U}_{m-1}}$ as described in the algorithm section above and solves the rank minimization optimization algorithm which outputs the matrix $\mathcal{B}$ containing the index codes representing the linear combination of messages to be sent in the retransmission phase. In our Android application, we use a Greedy Coloring algorithm by looping through and finding the maximal clique in the directed side information graph $\mathcal{G}$ and then reducing the graph $\mathcal{G}$ to only remaining nodes not in the maximal clique until no nodes remain. To find the maximal cliques of $\mathcal{G}$ in each iteration we use the Bron-Kerbosch algorithm which is a recursive backtracking algorithm for finding all maximal cliques in a graph [10]. Once the maximal clique is found, a row is appended to resulting matrix $\mathcal{B}$ with a 1 in position k if node k is included in the maximal clique. Once the rank minimization algorithm finishes with output matrix $\mathcal{B}$, we create matrices **G**, **S** and **X** as described in the index coding algorithm above. Note that matrix **X** is the same as matrix **S** given that we do not do any error correction coding.

Test Setup

Our Android application consists of one primary test setup with a few modifiable parameters. Each test is initiated from the transmitter on the main screen of the application as seen in Figure 10 with the ability to change the number of sequential tests done in succession for simulation purposes, which test to choose from between a basic predetermined test or the

primary test used in simulation, whether or not index coding should be used to compare results against a baseline, and whether or not the results should be stored.

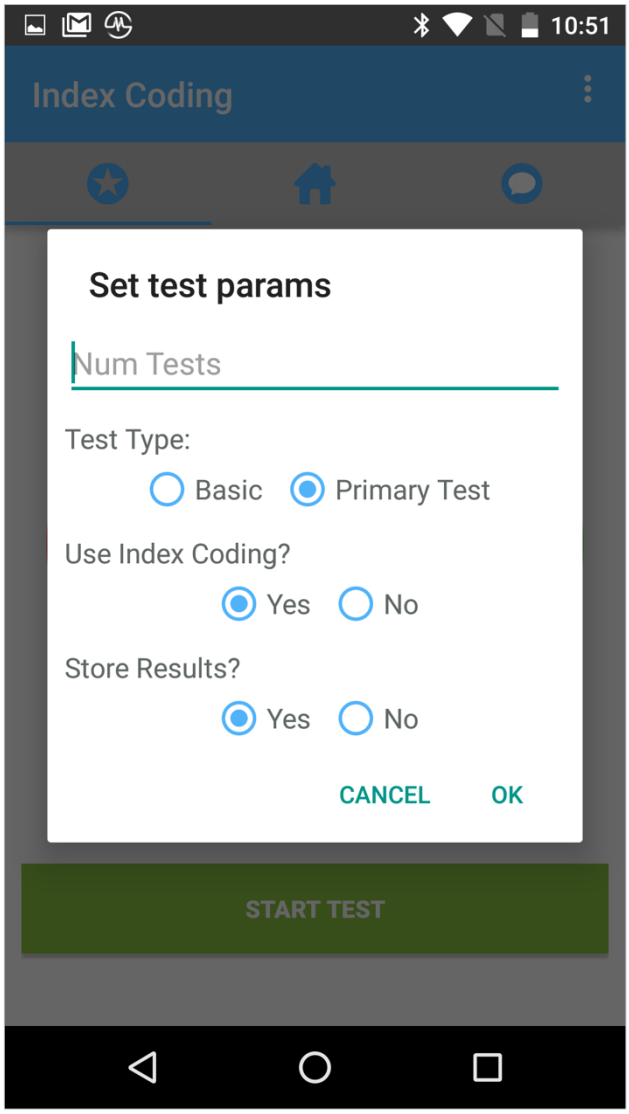


Figure 10: Primary Test Configuration.

In addition, each device has its own configurable erasure probability ϵ that can be set prior to each test (Figure 11). This erasure probability is necessary for simulation given that the M87 Proximity SDK doesn't drop messages on its own. When collecting results, we modified

this erasure probability for different tests, but kept it consistent across all receivers. In addition, this erasure probability only applies to the first round of transmission. Meaning, all messages in the retransmission phase are received.

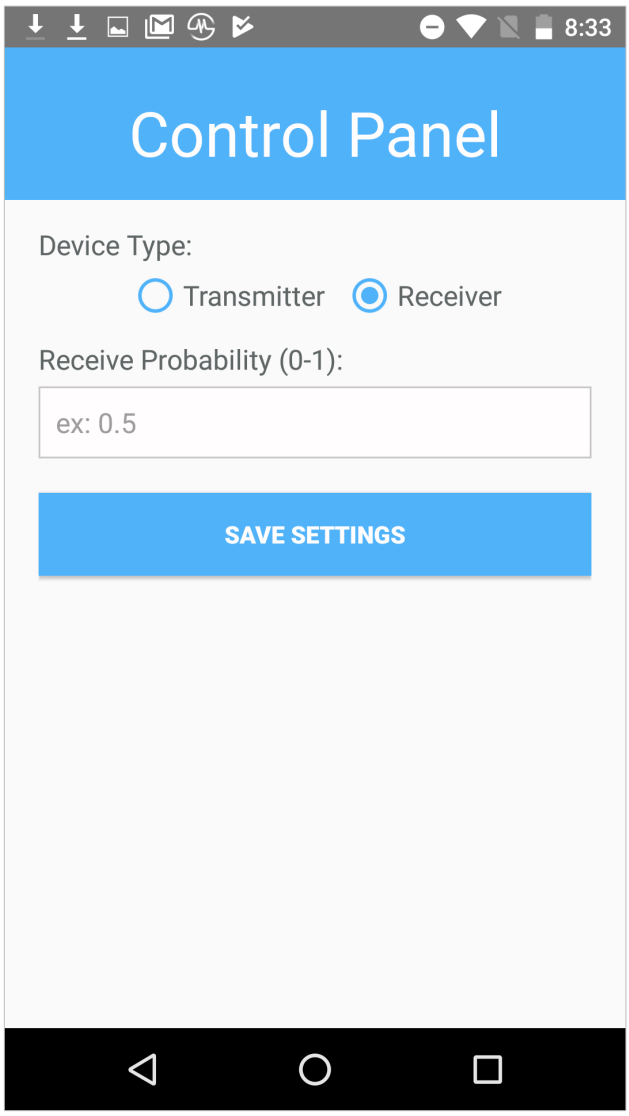


Figure 11: Erasure Probability Configuration.

The primary test used for collecting results involves transmitting a unique 4 KB (15 x 15 resolution) image to each receiver in the network. The test sends 200 bytes of each image every

transmission round for the ‘messageString’ parameter in the message structure for a total of 20 rounds. At the end of the test, each receiver displays the image it received (Figure 12). Lastly, if a test is configured to store results, the transmitter sends the results to an AWS EC2 server where it saves the results. The results of our mesh network demonstrating the effects of our index coding algorithm compared to the simulation results of [6] can be seen in the following section.

Figure 12: Primary Test Conclusion. Above shows receiver 1 displaying the image it received from the transmitter at the end of the test.

RESULTS

Performance Criteria

The following performance criteria is defined verbatim by M. Kim et al. [6]. The index coding gain η is defined as the ratio of spectral efficiency with using index coding ρ_{RTX} versus without (baseline) ρ_0:

$$\eta = \frac{\rho_{RTX}}{\rho_0}$$

The equation for spectral efficiency is given in equation 3 on page 7. Since $\epsilon_{k,l}$ are independent, the spectral efficiency for retransmission for the set of subchannels $\mathcal{M}_k$ is given by:

$$R_k \prod_{l \in \mathcal{M}_k} (1 - \epsilon_{k,l})$$

Given that q messages are to be retransmitted to r subchannels, the expected spectral efficiency with index coding is:

$$\rho_{RTX} = \frac{1}{r} \sum_{k=1}^{q} R_k \prod_{l \in \mathcal{M}_k} (1 - \epsilon_{k,l})$$

The baseline spectral efficiency given retransmission of q subchannels is given by:

$$\rho_0 = \frac{1}{q} \sum_{k=1}^{q} R_k \, (1 - \epsilon_{k,l})$$

Basically, q is the number of receivers that did not decode their own message and r is the rank of the index coded matrix $\mathcal{B}$.

In the paper, M. Kim et al. use the lower bound of the performance gain η_{LB} to measure performance [6]. More specifically, they prove that the lower bound of the spectral efficiency using index coding ρ_{LB} is given by:

$$\rho_{RTX} \geq \rho_{LB} = \frac{1}{r} \sum_{k=1}^{q} R_k \prod_{l=1}^{r} (1 - \epsilon_{k,l})$$

Thus, the lower bound of the performance gain η_{LB} is:

$$\eta_{LB} = \frac{\rho_{LB}}{\rho_0} = \frac{q}{r} \cdot \frac{\sum_{k=1}^{q} R_k \prod_{l=1}^{r} (1 - \epsilon_{k,l})}{\sum_{k=1}^{q} R_k (1 - \epsilon_{k,l})} = \frac{q}{r} (1 - \epsilon)^{r-1}$$

It is worthy to note that $R_k = R$ and $SNR_{l,k} = SNR$ if we meet the transmit power constraint $\sum_{l=1}^{L} P_l = P$ (example/proof given in the paper) [6]. As a result, $\epsilon_{k,l} = \epsilon \; \forall \; k, l$. Below are a few remarks about the performance criteria:

- If $r = q$, then we should just retransmit the individual messages w_i as $\eta_{LB} = (1 - \epsilon)^{r-1} < 1$ given that $0 < \epsilon < 1$.
- If $r = 1$ ($\text{rank}(\mathcal{B}) = 1$), then we only need to retransmit 1 index coded message and are at the maximum possible gain with $\eta_{LB} = q$.

Figure 13 below shows the plot M. Kim et al. generated of η_{LB} over r with $\epsilon = 0.1$ and $q = 10$ [6]. The plot demonstrates that $r \leq 5$ achieves $\eta_{LB} \geq 1$. The next section will describe the simulation results by M. Kim et al.

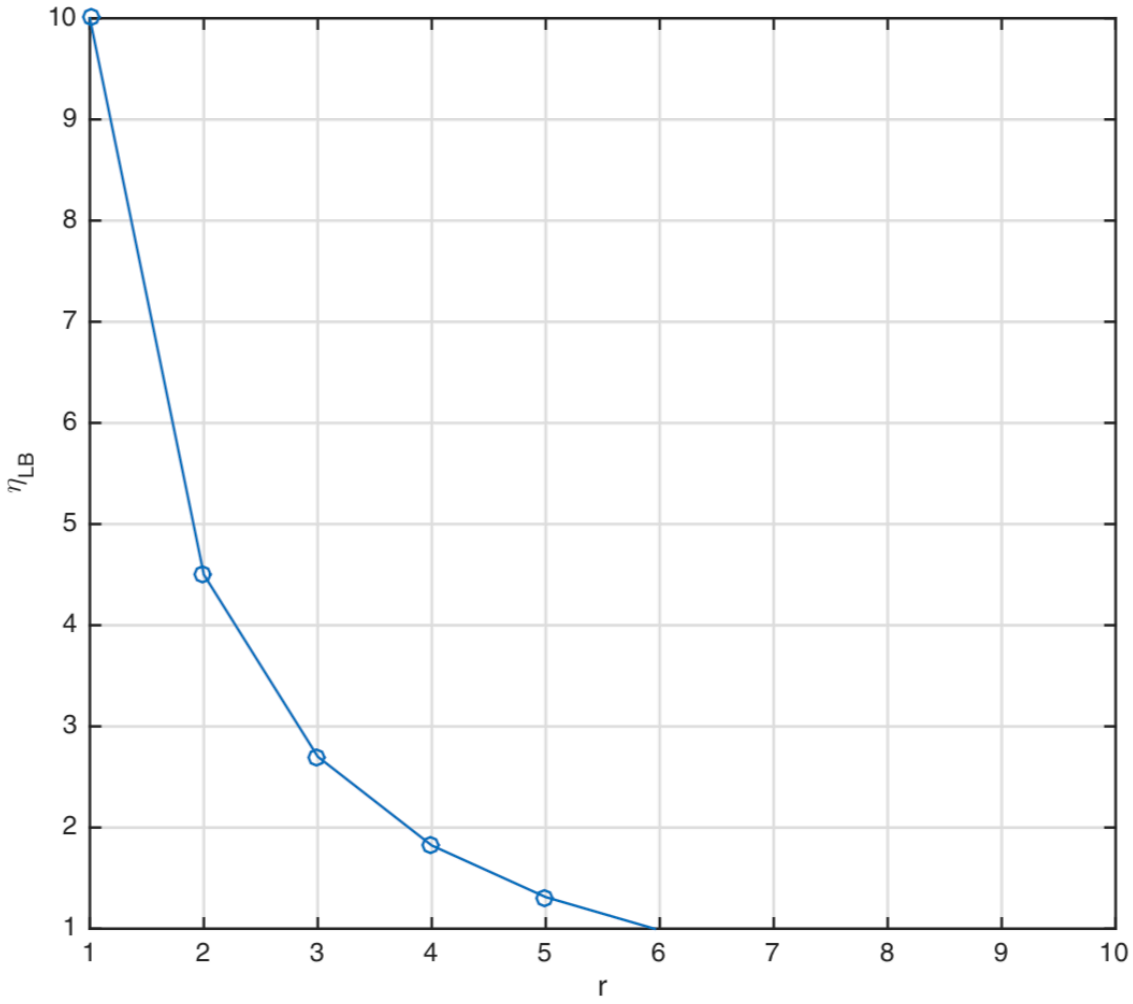


Figure 13: Index code gain η_{LB} as a function of r using $\epsilon = 0.1$ and $q = 10$ [6].

Simulation Results

In the paper by M. Kim et al., the authors randomly generate a dashboard matrix F with $K = 100$ and find $\mathcal{A}_{\mathcal{G}}$, rank($\mathcal{B}$), and calculate η_{LB} [6]. In their first experiment, they use the first 1000 randomly generated dashboard matrices with $q = 10$ given $\epsilon = 0.1$ and similarly for $q = 20$ given $\epsilon = 0.2$. Then, they run a clique covering algorithm to determine the resulting rank r of matrix $\mathcal{B}$ and calculate η_{LB}. Their results are shown in Table 1 with the percentage field resembling the percentage of the 1000 matrices resulting with the respective rank r [6]. Table 1a shows that there are 0 cases where $\eta_{LB} < 1$ when $q = 10$. Thus, showing that index coding improved performance in every case. Table 1b shows that $\eta_{LB} \geq 1$ 97.9% of the time.

	$r = 2$	$r = 3$	$r = 4$	$r = 5$
Percentage	0.321	0.623	0.055	0.001
η_{LB}	4.5	2.7	1.82	1.31

(a) $\epsilon = 0.1$ and $q = 10$ case

	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$
Percentage	0.085	0.583	0.311	0.020	0.001
η_{LB}	2.56	1.63	1.09	0.74	0.52

(b) $\epsilon = 0.2$ and $q = 20$ case

Table 1: Index Coding performance gain η_{LB} on randomly generated dashboard matrices with (a) $q = 10$ and (b) $q = 20$ [6].

Lastly, in Figures 14 and 15 the authors show the empirical averages of q and r for 1000 randomly generated dashboard matrices of size $K = 100$. These figures show that η_{LB} increases when $\epsilon \leq 0.08$ and decreases afterward as ϵ increases.

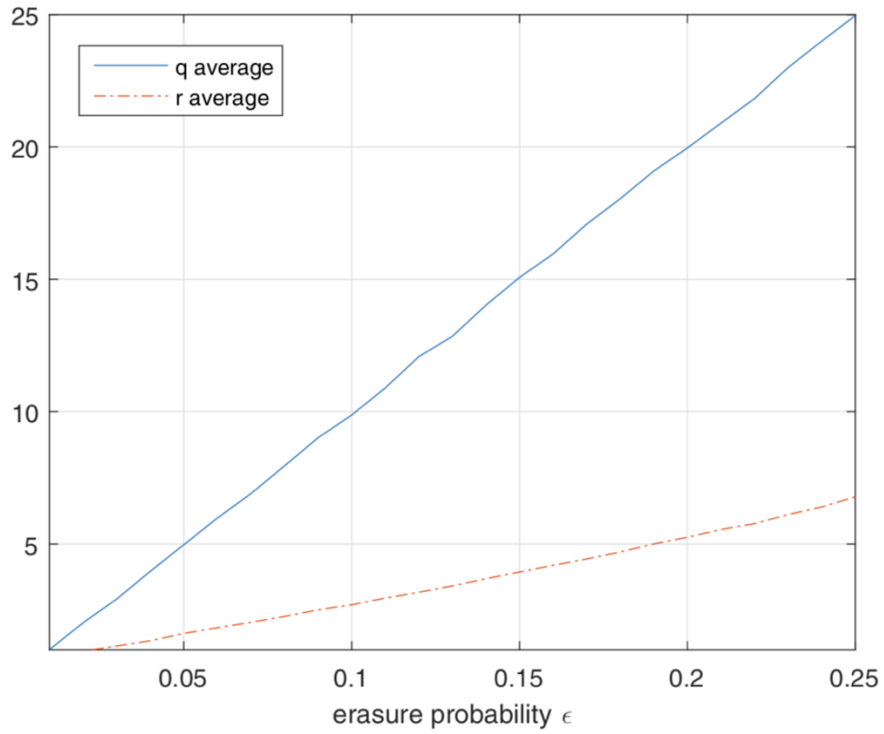


Figure 14: Empirical average of q and r for each ϵ [6].

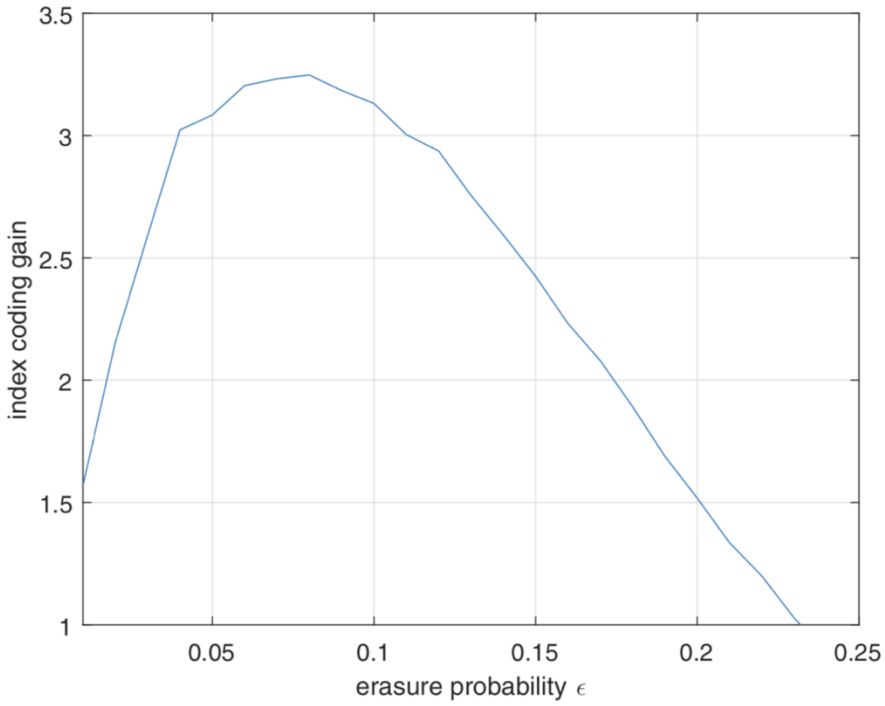


Figure 15: Empirical average for η_{LB} with $K = 100$ for each ϵ [6].

Mesh Network Results

For our active mesh network, we wanted to see if we could produce similar results to the simulation results found by M. Kim et al. in [6]. We ran our primary test with 1 transmitter and 5 receivers ($K = 5$) over 100 times each for erasure probabilities $\epsilon \in [0.1, 0.25, 0.33, 0.5, 0.75]$. Given that each test sends 4 KB images to each receiver by sending 200 bytes per transmission round, there are a total of 20 transmission rounds per test. In addition, we also ran the same tests using the baseline retransmission strategy without using index coding by simply resending each dropped message. The average time to send one message packet is roughly 1.2 seconds.

Figure 16 shows the same comparrison of η_{LB} versus erasure probability ϵ as Figure 15, but using the data collected by our mesh network. The index coding gain is on average a lot lower than the simulations run by M. Kim et al. due to the fact that our network only contains 5 receivers, which affects the benefits of index coding due to smaller side information graphs. As a result, when the erasure probability $\epsilon = 0.1$, the expected number of dropped messages X in the first round of transmission is $\mathbb{E}[X] = \epsilon K = (0.1)(5) = 0.5 < 1$. Given that the expected value is less than 1, index coding is not necessary and thus the gain is close to 1. However, in the improbable case when 2 receivers drop their message, $q = 2$, when $\epsilon = 0.1$ the rank of the resulting index-coded matrix $\mathcal{B}$ is usually 1 given that the receivers have a high percentage of overlapping side information because they collected more side information. The reverse scenario occurs when $\epsilon = 0.75$. In this scenario, the expected value of dropped messages in the first round of transmission is $\mathbb{E}[X] = \epsilon K = (0.75)(5) = 3.75$. However, the difference between number of receivers who dropped their message (q) and resulting rank of matrix $\mathcal{B}$ (r) is low due to the fact that each receiver also can't decode as many other receivers' messages. Thus, there is little shared side information between receivers in matrix **G** resulting in a higher rank of matrix $\mathcal{B}$ and thus more index codes need to be sent in the retransmission phase. According to our results, index coding is the most effective when the erasure probability $\epsilon = 0.5$.

Figure 17 and 18 show similar results. Figure 17 shows the average total time to run the experiment from start to end. The total time increases as the erasure probability increases due to the fact that more retransmission messages must be sent every transmission round because there are more dropped messages. Lastly, Figure 18 shows the total time saved percentage using index coding compared to the baseline. The trend in Figure 18 follows the same pattern as Figure 16,

that our index coding algorithm performed best when the erasure probability $\epsilon = 0.5$. The exact value of erasure probability ϵ for maximum gain is dependent on the size of the network because it affects the size of the side information graph. The maximum total time saved as a percentage of the baseline is roughly 7% that occurs at $\epsilon = 0.5$. In other words, our experiment shows that index coding saves us up to a maximum of 7% of the total time as compared to not using index coding.

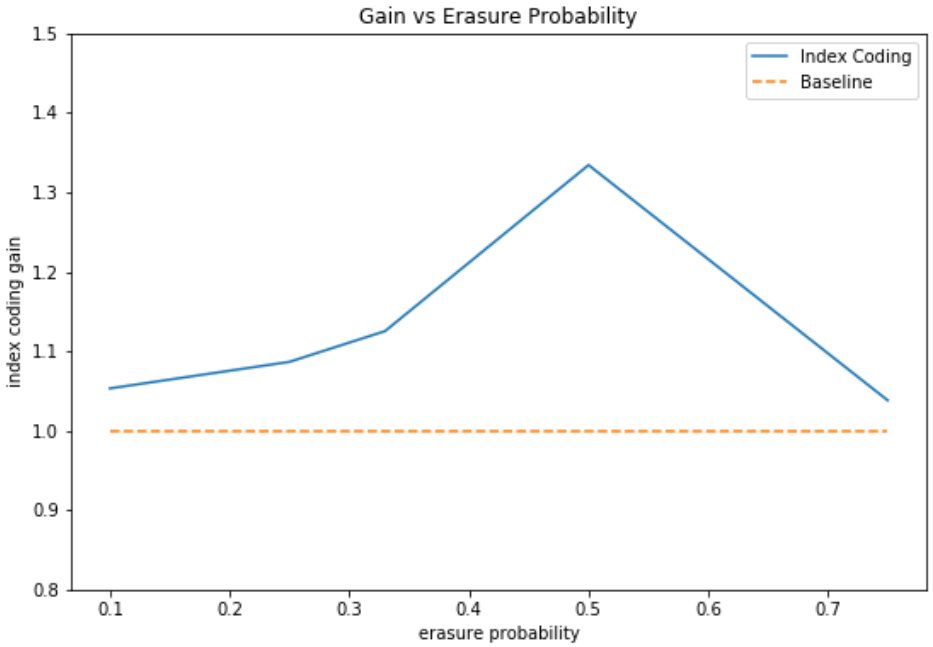


Figure 16: Mesh network average η_{LB} with $K = 5$ for each ϵ.

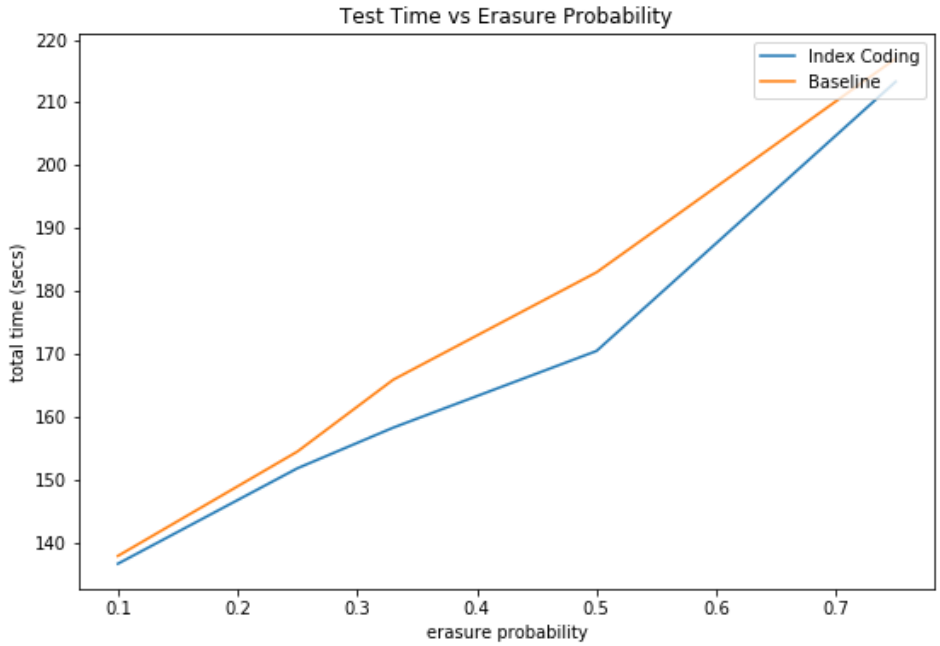


Figure 17: Total test time vs. erasure probability.

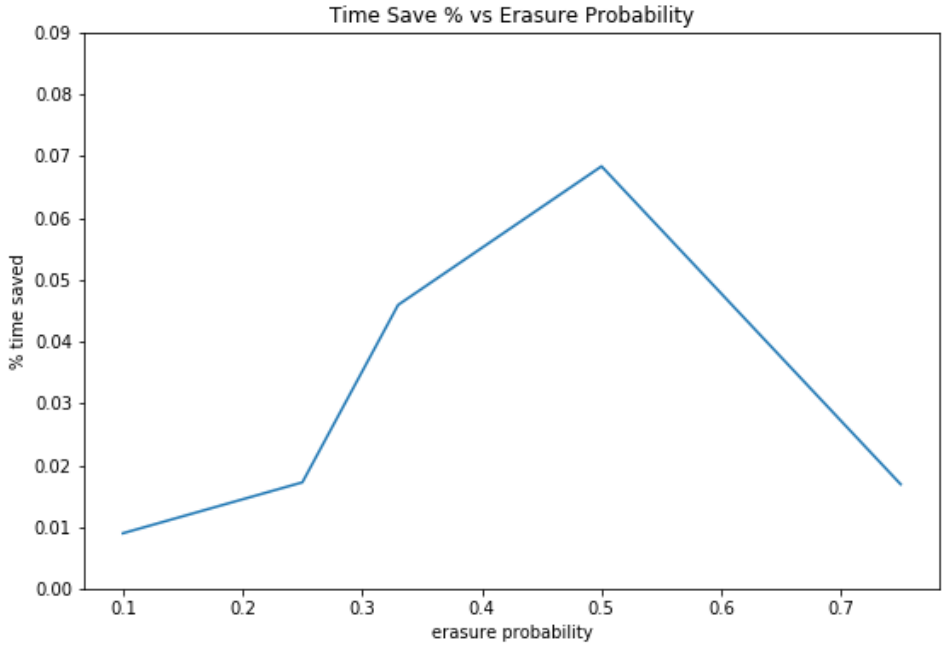


Figure 18: Total time saved as percentage compared to baseline vs. erasure probability.

CONCLUSION

In conclusion, index coding studies the optimal coding scheme for transmitting messages across a network where nodes share side information. In this report we developed an active OFDMA mesh network using Android phones leveraging the M87 Proximity SDK that demonstrates the benefits of using index coding during the retransmission phase as described in the algorithm proposed by M. Kim et al. in [6]. Using index coding in our network saved us up to 7% of the total time to execute our experiment of transmitting 4 KB images to 5 different receivers in the network. In addition, we tested the relationship between the performance gain of using index coding in our network compared to simply retransmitting the original message (the baseline) across multiple erasure probabilities with the maximum gain η_{LB} of 1.45 occurring at the erasure probability $\epsilon = 0.5$. Furthermore, our results had similar trends to the results produced by M. Kim et al. in simulation, but different optimal points due to the size of the network affecting the amount of side information that could be coded. Future research would be to test the effectiveness of index coding in our network using different rank minimization algorithms other than a clique covering algorithm. In addition, with the advancements of the M87 Proximity SDK the next logical step would be to increase the network size as the small network of $K = 5$ does not show the full potential performance gain of index coding. Lastly, it would be interesting to explore the benefits of index coding in our network where there are multiple transmitters and multiple side channels.

REFERENCES

[1] Birk, Yitzhak and Tomer Kol. “Informed-Source Coding-on-Demand (ISCOD) over Broadcast Channels.” INFOCOM (1998).

[2] 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.

[3] H. Maleki, V. Cadambe, and S. Jafar, “Index coding: An interference alignment perspective,” in Proc. IEEE International Symposium on Information Theory (ISIT). IEEE, 2012, pp. 2236–2240.

[4] S. Jafar, “Topological interference management through indexcoding,” IEEE Trans. Inform. Theory, vol. 60, no. 1, pp. 529–568, Jan. 2014.

[5] J. I. Tamir, E. R. Elenberg, A. Banerjee, and S. Vishwanath, “Wireless index coding through rank minimization,” in Proc. IEEE ICC, 2014.

[6] M. Kim, Y. Chen, M. Borokhovich and S. Vishwanath, "Index-Coded Retransmission for OFDMA Downlink," 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 2016, pp. 1-6.

[7] Gaussian elimination. https://en.wikipedia.org/wiki/Gaussian_elimination

[8] M87 Proximity Software Development Kit. http://www.m-87.com/sdk/

[9] D. Camps-Mur, A. Garcia-Saavedra and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," in IEEE Wireless Communications, vol. 20, no. 3, pp. 96-104, June 2013.

[10] C. Bron and J. Kerbosch, "Algorithm 457 – finding all cliques of an undirected graph", Comm. ACM 16: 575–577, 1973.