Redundancy Minimizing Techniques for Robust Transmission in Wireless Networks

Anna Kacewicz and Stephen B. Wicker

Abstract: **In this paper, we consider a wireless multiple path network in which a transmitting node would like to send a message to the receiving node with a certain probability of success. These two nodes are separated by N erasure paths, and we devise two algorithms to determine minimum redundancy and optimal symbol allocation for this setup. We discuss the case with $N = 3$ and then extend the case to an arbitrary number of paths. One of the algorithms minimum redundancy algorithm in exponential time is shown to be optimal in several cases, but has exponential running time. The other algorithm, minimum redundancy algorithm in polynomial time, is sub-optimal but has polynomial worst-case running time. These algorithms are based off the theory of maximum-distance separable codes. We apply the MRAET algorithm on maximum-distance separable, Luby transform, and Raptor codes and compare their performance.**

Index Terms: **Error correction coding, multiple path channels, networks, redundancy, robustness.**

I. INTRODUCTION AND RELATED WORK

Wireless networks are being increasingly used to download/transmit a large amount of data. This data is subject to deterioration because it is sent through the air rather than a more reliable medium. Not only are wireless networks susceptible to noise, but are also more vulnerable to malicious or uncooperating nodes. The messages sent across a wireless network should be received without error in a short amount of time regardless of the increased disturbance levels. Hence, it is useful to devise coding methods and algorithms to ensure message integrity across wireless channels. While redundancy increases network robustness, it also reduces network efficiency. This suggests the importance of minimally increasing the message length as to guarantee an aspired degree of network reliability.

In this paper, we consider a multiple path wireless network through which a source-destination pair would like to communicate. Moreover, the presence of adversaries on some paths may cause information to be lost or corrupted. We model the presence of a malicious node as an erasure channel. The erasure channel is a type of wireless channel in which packets forwarded through that channel are either fully received or erased with a certain probability. For security purposes, we assume that the message is encrypted by a standard encryption technique. The erasure channel assumption is feasible since the encryption allows the destination node to validate data it receives, meaning that if the node receives corrupted data then that data will be thrown out and treated as an erasure. Malicious nodes may also steal messages, causing the receiving node to observe an erasure. The path erasure probabilities are associated with the degree to which they can be trusted.

In [1], the authors suggest a protocol for secure message transmission in a multipath channel. They generate a method to assess the"trustworthiness" of each path based on previous behavior, and the paths which are trusted above a certain threshold are put in the active path set (APS). The "rustworthiness" level of a path is directly linked to the probability of successful transmission. The paths in the APS are then used to transport messages. We assume a similar scenario and then use the path security levels to strategically transmit data.

Wireless networks such as mobile ad hoc networks or sensor networks have frequent changes in topology due to link failures, physical obstructions, network intrusions, etc. Dependability on these sort of networks requires dynamic algorithms which quickly determine routing, redundancy, etc. for deviations in the network. In [2], we introduce algorithms to dynamically determine the redundancy and message dispersion among the path.

There is a vast array of work done on routing in multipath channels. In [3], the authors introduce a method to find maximally disjoint paths. We assume that the independent paths in our setup were found in a similar fashion. In their paper they transmit information down these paths assuming that they have the same security level. In [4], the authors discuss a multipath routing technique over equally reliable links. They devise a method to optimally allocate channel coded packets down the paths by maximizing the success probability. The problem with this method is the low network efficiency. In another paper [5], the authors remove the assumption that the paths have the same performance. They determine an approximation of the success probability for the network and then allocate packets down each path as to maximize this function. They do not consider the question of how much redundancy to add to the original message, and just assume that it is a pre-determined number. In this paper we discuss algorithms which determine the minimum redundancy and optimal symbol allocation to achieve a target probability of success.

Redundancy is vital in erasure channels since it allows perfect decoding even with some erasures. Maximum distance separable (MDS) codes are important examples of erasure codes since they have the property that only a set the size of the input symbols is required to perfectly decode the message. Our algorithms are based on the structure of MDS codes. In [1], the authors also use redundancy in the form of an erasure code which is based off of Rabin's algorithm [6]. A widely used MDS code is the Reed-Solomon code [7].

Our main contribution in this paper include the design of two

Manuscript received April 29, 2009.

The authors are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, email: ak387@cornell.edu, wicker@ece.cornell.edu.

algorithms minimum redundancy algorithm in exponential time (MRAET) and minimum redundancy algorithm in polynomial time (MRAPT) to determine minimum redundancy and optimal symbol allocation to attain a probability of success. We compare the performance of the algorithms with respect to each other and the desired success level. We test and compare these algorithms on three different error-correction codes, namely minimum-distance separable (MDS), Luby transform (LT), and Raptor codes. Also, we design the MDS, LT, and Raptor code parameters to be compatible with the algorithms. The performance of the codes is evaluated using the MRAET algorithm.

This paper is organized as follows. In Section II, we discuss necessary background information on the error-correction codes that we use and our system model. Section III discusses the basic case where there are $N = 3$ paths, and next in Section IV, we move on to the algorithms for the case with an arbitrary number of paths. Further, the parameters for the codes we used are discussed in Section V and their performance is simulated in Section VI. Finally, in Section VII, we conclude the paper.

II. PROBLEM SETUP AND BACKGROUND INFORMATION

We assume that we have a source node who has to convey information to a destination node. These two nodes are separated by multiple wireless paths, each acting like an independent erasure channel. An erasure on a particular path could be caused by a malicious node which is stealing or tampering with data. The data is encrypted with a standard encryption algorithm which allows the destination to determine if an adversary has corrupted a path. Hence each path is associated with a predetermined erasure probability which represents its "trustworthiness" level. Let path i have a probability of success p_i. Without loss of generality, we assume that $p_1 \geq p_2 \geq \cdots \geq p_N$ given that there are N paths. The transmitting node knows the statistics of the channels between itself and the receiving node, and hopes the message can be decoded within probability of success p^*. The source node needs an algorithm that increases the length of the message and allocates symbols down the paths so that p^* is achieved. Since the security level on the paths changes over time, this algorithm also needs to be dynamic.

Erasure channels are discrete memoryless channels in which individual packets are received without an error or not received at all. Typical network protocols often use a feedback channel to mitigate the effects of an erasure channel, through which the receiver notifies the sender of any lost packets. Classic Shannon theory has shown that the capacity of a discrete memoryless channel with feedback is equivalent to the same channel without feedback, implying that feedback channels are wasteful and unnecessary for this class of channels. Digital Fountain Codes are a class of sparse-graph codes designed for erasure channels. Unlike typical codes used for erasure channels, such as Reed-Solomon (RS) codes, fountain codes are *rateless* codes meaning that the symbols can be determined on the fly. For example using an (n, k) RS code, one must determine the code rate $r = k/n$ based on the probability of erasure prior to transmitting. Since Reed-Solomon codes are MDS codes, this implies that any k symbols out of n can be used to reconstruct the original message. We will elaborate on MDS codes in the next section. If the estimated success probability of the channel is too high causing the destination to receive less than k symbols, then the message cannot be decoded. For erasure channels one wants a simple way to expand the code leading to a lower rate code. In 1998, Luby created digital fountain codes called LT codes. In 2002 he created the first company, Digital Fountain, which dealt with sparse graph codes. We go into the details and intuition of LT codes in a later section [8].

A. Maximum-Distance Separable (MDS) Codes

MDS codes have a beautiful property which makes them particularly amenable for erasure channels. Next we mention the background information to understand this special class of codes [9].

Definition 1: Hamming Distance
The *hamming distance* between two codewords $\mathbf{u}, \mathbf{v}$ of length n is the number of positions in which they differ or,

$$d_{\text{Hamming}}(\mathbf{u}, \mathbf{v}) = d(\mathbf{u}, \mathbf{v}) = |\{i | u_i \neq v_i,\ i = 0, 1, \ldots, n-1\}|.$$

Definition 2: Minimum Distance of a Code
The *minimum distance* $d_{\min}$ of a code is the minimum Hamming distance between all distinct codewords in the codebook.
An (n, k) code is one which starts with a message of length k and encodes it to a codeword of length n, or adds $n-k$ redundant symbols. We call the ratio $\frac{n}{k} = \gamma$.

Theorem 1: Singleton Bound
The minimum distance $d_{\min}$ for an (n, k) code is bounded by

$$d_{\min} \leq n - k + 1.$$

Definition 3: Maximum-Distance Separable Code
Maximum-distance separable (MDS) codes meet the Singleton Bound with equality, or:

$$d_{\min} = n - k + 1.$$

From [9] it is known that for an (n, k) MDS code in systematic form, any combination of k out of the n codeword symbols allows for perfect recovery of the original message. This attribute serves well in erasure channel scenarios because, with complete certainty, a message can be decoded with up to $n - k$ erasures.

An important and well known MDS code is the RS code. RS codes are thus excellent candidates for channels with bursts of error such as CD's, DVD's, and the newer blu-ray discs. The reason for this is the fact that these data discs get errors when the disc gets scratched or dirty in a specific area resulting in error bursts. In particular, the CD uses a form of RS code called cross-interleaved RS codes or CIRC. CIRC is composed of the concatenation of two layers of an RS code separated by an interleaver.

RS codes are extremely useful since their codewords are maximally spaced apart, though there is a trade-off between this property and running time complexity. The encoding and decoding times for an RS code are quadratic with the codeword size. This implies that these codes are optimal for small message and codeword size. The problem is that most applications require fairly large source sizes. In the next section, LT codes are introduced, which are almost MDS codes and have reduced running times. For further theory on the RS code we encourage the reader to go to [7].

B. Luby Transform (LT) Codes

A simple analogy to LT codes is holding a bucket and catching drops until a sufficient amount of drops are caught. The codes can be thought of as the balls and bins problem, where the bins represent the input symbols and the balls being the encoded symbols. The question is, how many balls does one have to throw so that with probability $1-\delta$ each bin has at least one ball. Given that there are k bins, the number of balls should be around $k\log(k/\delta)$[8]. This yields small encoding and decodings times both on the order of $k\log(k/\delta)$. We will first introduce the encoding and decoding algorithms for the LT codes.

Suppose we have a source message $m_1 m_2 \dots m_k$, and we wish to form an output symbol c_i. The encoding of the message goes as follows:

1. Pick d_n from a degree distribution $\rho(d)$. The specifics of $\rho(d)$ will be revealed below.
2. Choose d_n distinct input packets uniformly at random and call the set of their indices I_{d_n}. Then, using modulo 2 arithmetic, $c_n = \sum_{j:j\in I_{d_n}} m_j$.

These codes are sparse because the constructed degree distribution results in a mean degree that is considerably smaller than k. There are many different ways to relay the degree and edge connectivity of the graph to the decoder such as synchronized clock, sending header containing a key, etc.

Successful decoding depends on the degree distribution used and the edges in the graph G. The decoding is quite simple due to the encoding structure and is a simplified version of the belief propagation algorithm, also known as the sum-product algorithm [10]. Using terminology as in [11], the *ripple* represents the set of output symbols of degree one. If the decoding algorithm reaches a point where the ripple is empty prior to decoding all the input symbols, then the receiver fails to obtain the source message. Hence, it is vital to design a degree distribution which assures with high probability that the ripple is always nonempty before all the input symbols are found and also that all the input symbols are connected to at least one output symbol. In theory, the ideal degree distribution is called the *ideal soliton distribution*,

$$\begin{aligned}\rho(1) &= \quad 1/k \\ \rho(d) &= \frac{1}{d(d-1)} \text{ for } d = 2, 3, \cdots, k\end{aligned}$$

which behaves as it should in expectation. The expected degree of this distribution is about $\log k$. The problem with this distribution is that even the smallest oscillation around the expected degree results in the failure of the decoding algorithm because there is no degree one check node. To take care of this issue, Luby introduced a slightly altered degree distribution which he calls the *robust soliton distribution* [8]. This distribution avoids the problem of not having an output node with degree one by guaranteeing that the expected number of degree one outputs is approximately $R = c\log(k/\delta)\sqrt{k}$ $(c > 0)$. The robust soliton distribution is:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z}$$

where $Z = \sum_d \rho(d) + \tau(d)$ and $\tau(d)$ is defined as follows,

$$\tau(d) = \begin{cases} \frac{R}{kd}, & \text{for } d = 1, 2, \cdots, \frac{k}{R} - 1 \\ \frac{R}{k}\log(\frac{R}{\delta}), & \text{for } d = \frac{k}{R} \\ 0, & \text{for } d > \frac{k}{R}. \end{cases}$$

Using this distribution there need to be at least $n = kR$ encoded symbols so that with probability $1-\delta$ we can successfully decode the message [8]. An extension of LT codes include Raptor codes, which have a linear encoding and decoding time [12].

C. Raptor Codes

Raptor codes are a class of fountain codes which have the property that encoding and decoding have a constant cost (in the per symbol sense). LT codes have a per symbol decoding on the order of $O(\log(k))$ since the decoding graph must have roughly $k\log(k)$ edges to ensure that the input symbols are all covered with high probability. Raptor codes diminish this condition to having a certain fraction of recoverable input nodes, which results in constant decoding cost [12]. Since the goal is to be able to recover all the input symbols, Raptor codes first apply a classical erasure code to the input symbols, followed by the LT code. A Raptor code is of the form $(k, C, \Omega(x))$, where C is the first layer code and $\Omega(x)$ is the output symbol distribution. The encoding goes as follows:

1. Using the code C, encode the message (m) of length k symbols into a codeword (c) of n symbols.
2. Apply the LT code algorithm to the codeword c resulting in another codeword c' which is slightly larger than c. The LT algorithm should use the specified output distribution $\Omega(x)$.

The choice of the code C effects the encoding and decoding costs, and also the decoding algorithm. Choices for C include Tornado codes, LDPC codes, extended Hamming codes, etc [13]. In our paper we use a regular Gallager LDPC code for C.

D. System Model

Each path i out of the N paths acts as an erasure channel with erasure probability $1-p_i$. An erasure channel with erasure probability p is one in which each symbol is erased with probability p [14]. In our model we assume that the destination node either receives all the symbols down a particular path or receives nothing. This assumption is feasible since data cannot be trusted if it has been tampered with by an adversary.

A codeword of length n is dispersed among the N paths, with path i receiving f_i symbols forming a vector $\mathbf{f} = [f_1, f_2, \dots, f_N]$. This means that $\sum_{i=1}^{N} f_i = n$. We can form a vector $\mathbf{s}$ of length N composed of '1's and '0's with a '1' in spot i representing a non-erasure on path i and a '0' representing an erasure. If we use an MDS code, then with complete certainty the message can be decoded if k out of n symbols are received. Thus, if we construct a matrix S composed of all possible combinations of '0's and '1's our probability of successful decoding for a specified f becomes:

$$P_{\text{success}}(f) = \sum_{s\in S} \prod_{i=1}^{N} p_i^{s_i}(1-p_i)^{1-s_i} u(s\cdot f - k) \qquad (1)$$

where $u(\cdot)$ represents the unit step function. The matrix S is filled with all possible vectors $\mathbf{s}$ implying that there are 2^N rows. Hence if we run through all the rows in S to calculate P_{success}, this results in an exponential running time with relation to the number of paths. If the number of paths is not too large and one wishes for extreme precision, this calculation is not too arduous. Otherwise, it is necessary to find a close alternative which we will discuss next.

Each path has a Bernoulli distribution since it receives the exported symbols (1) with probability p_i or an erasure (0) with probability $1-p_i$. This implies that the sum of multiple transmissions across path i has a Binomial distribution. For large sample sizes, the Binomial distribution can be approximated using the Gaussian distribution, and the authors in [5] suggest this approximation to calculate the probability of success.

We consider a random variable that represents the average number of successful transmission attempts out of f_i on path i. Using the observation mentioned above, this random variable is distributed binomially. We approximate this random variable using a Gaussian distribution and it is known that the distribution of the sum of independent Gaussian random variables is also Gaussian. This indicates that the distribution of the sum of the paths is also Gaussian. Hence, the approximation on each path is Gaussian distribution $\sim \mathcal{N}(f_i p_i, f_i^2 p_i(1-p_i)$, and the distribution of the sum of the paths becomes $\sim \mathcal{N}\left(\sum_{i=1}^N f_i p_i, \sum_{i=1}^N f_i^2 p_i(1-p_i)\right)$. Then, integrating this distribution over the scenario that k or more symbols are received in total yields a probability of success function:

$$P_{\text{success}}(f) \approx \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{\sum_{i=1}^N f_i p_i - k + \frac{1}{2}}{\sqrt{2\sum_{i=1}^N f_i^2 p_i(1-p_i)}}\right) \qquad (2)$$

where $\text{erf}(x) = \frac{2}{\pi}\int_0^x e^{-y^2}dy$. It can be seen that computation of this success probability approximation is significantly simpler than that of the true success probability.

The source node wants to assure that his message is received intact with probability p^* at the destination. The success probability functions along with the security level of each path can be used to determine message redundancy and symbol allocation. The original data is k symbols long, and if an MDS code is used to extend the k to n symbols, there are a few observations we can make. As mentioned earlier, any of the k of the n symbols can decode the original message. Let $\gamma = n/k$ represent the redundancy ratio. Below are some initial observations:

- $p_1 \geq p_2 \geq \cdots \geq p_N$ implies that $f_1 \geq f_2 \geq \cdots \geq f_N$.
- If $\gamma \geq N$ then an optimal approach is to send $f_1, f_2, \cdots, f_N \geq k$.
- It is not optimal to send more than k symbols down any path.
- If $p_1 \geq p^*$, then k symbols should be sent down path 1. In this case $\gamma = 1$.

These observations have led to developing two optimal redundancy and symbol allocation algorithms described in [2], and presented in the next section.

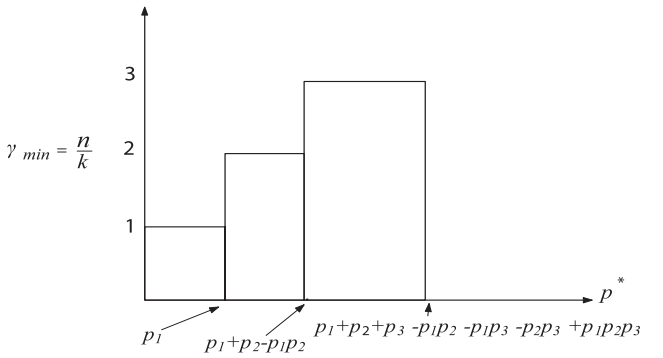


Fig. 1. Minimum redundancy for $N = 3$ when $p_1 \geq p_1p_2+p_2p_3+p_1p_3-2p_1p_2p_3$.

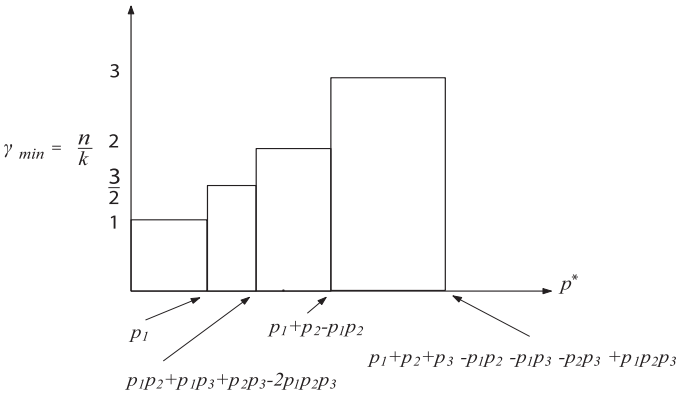


Fig. 2. Minimum redundancy for $N = 3$ when $p_1 < p_1p_2+p_2p_3+p_1p_3-2p_1p_2p_3$.

III. OPTIMAL SYMBOL ALLOCATION AND MINIMUM REDUNDANCY FOR $N = 3$

To simplify our analysis, in [2], we began with the situation with $N = 3$ paths between the source and destination. A brute force method can be used to find optimal symbol allocation and minimum redundancy. Assume that the desired success probability is p^* and that paths $1, 2, 3$ have erasure probabilities $1-p_1, 1-p_2, 1-p_3$. The optimal symbol allocation vector $\mathbf{f}$ and redundancy ratio γ are as follows:

- If $p_1 + p_2 - p_1p_2 < p^* \leq p_1 + p_2 + p_3 - p_1p_2 - p_2p_3 - p_1p_3 + p_1p_2p_3$,
  $\Rightarrow \gamma_{\min} = 3$ and $f_1, f_2, f_3 = k$.
- If $\max\{p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3, p_1\} < p^* \leq p_1 + p_2 - p_1p_2$,
  $\Rightarrow \gamma_{\min} = 2$ and $f_1, f_2 = k, f_3 = 0$.
- If $p_1 < p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$ and $p_1 < p^* \leq p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$,
  $\Rightarrow \gamma_{\min} = \frac{3}{2}$ and $f_1, f_2, f_3 = \frac{k}{2}$.
- If $0 < p^* \leq p_1$,
  $\Rightarrow \gamma_{\min} = 1$ and $f_1 = k, f_2, f_3 = 0$.

Fig. 1 shows a plot of minimum redundancy versus the target success probability for $p_1 \geq p_1p_2 + p_2p_3 + p_1p_3 - 2p_1p_2p_3$. Fig. 2 shows the case where $p_1 < p_1p_2+p_2p_3+p_1p_3-2p_1p_2p_3$. Results in Figs. 1 and 2 represent all possible combinations for the symbols across the paths. Ordering the paths simplifies the analysis so assume that $p_1 \geq p_2 \geq \cdots \geq p_N$. We know that the redundancy γ is such that $1 \leq \gamma \leq 3$. Considering

$\gamma = 1$ then there are three different possible symbol allocations: $\mathbf{f} = [k, 0, 0]$, $\mathbf{f} = [\frac{k}{2}, \frac{k}{2}, 0]$, and $\mathbf{f} = [\frac{k}{3}, \frac{k}{3}, \frac{k}{3}]$. It is known that a success occurs if the sum of the received symbols across all the paths is greater than or equal to k and since $p_1 \geq p_2 \geq p_3$, this implies that we only need to look at three situations: 1) When only the first path is needed for success, 2) when the first and second path are needed for success, and 3) when all three paths are needed for success. The reason for having only three symbol allocations is based on the following observation. Consider the case when only two paths are required and assume that $k/2$ symbols are sent down each path, then it can be seen that probability of success is exactly the same as when $(i-1)k/i$ symbols were sent down the first path and k/i symbols sent down the second path. Note that p_1, p_1p_2, and $p_1p_2p_3$ are probabilities of success for scenarios 1, 2, and 3, respectively. So clearly in this scenario, $\mathbf{f} = k, 0, 0$ yields the highest success probability. Continuing in this manner for different possible values of γ we obtain the results shown above. Unfortunately, there are not always going to be $N = 3$ paths, so next we devise a method to take care of the instance where there are an arbitrary amount of paths.

IV. ALGORITHMS FOR ARBITRARY NUMBER OF PATHS

Following [2], the results can be extended to an arbitrary number of paths. Since the APS set size varies with path security, it is important to design an algorithm that dynamically determines all parameters. A closed form expression for the network's probability of success is difficult to obtain, thus, we developed heuristic algorithms which determine parameters. One of these algorithms has exponential running time and is called minimum redundancy algorithm in exponential time or MRAET. Though MRAET has an exponential running time, we prove that in several special cases it's optimal. In cases when exponential running time is unacceptable, we introduce an algorithm with polynomial running time. We call it minimum redundancy algorithm in polynomial time (MRAPT).

Both MRAET and MRAPT algorithms consist of two steps. Part 1 is to reduce dimensionality of the space due to the fact that the search redundancy/symbol dispersion space is extremely large. This reduction results in a shorter search time required to determine the desired parameters. Given p^*, part 1 first assigns the symbol allocation vector $\mathbf{f} = [k, 0, \cdots, 0]$. $\mathbf{f}$ is then plugged into probability expression (1) and (2) in MRAET and MRAPT, respectively) and compared with p^*. If the value is greater than or equal to p^* the algorithm moves on to part 2. Otherwise it sets $\mathbf{f} = [k, k, 0, \cdots, 0]$ and repeats the procedure. Part 1 is exited when $\mathbf{f}$ is found that generates a probability greater than or equal to p^*. Part 2 picks off where part 1 left off, and it checks possible symbol allocations which have a smaller redundancy than that found in part 1 and also result in a success probability greater than p^*. The algorithm terminates when there are no options left. Taking the value $j = \gamma_{\min}$ from the first part of the algorithm, the second part of the algorithm starts with the case were the first $j-2$ paths have k symbols assigned to them. The algorithm then steps through different combinations for the rest of the $N-(j-2)$ paths to see if there is a combination which results in a lower redundancy and also meets the target success probability requirement. An example of a combination which part two of the algorithm will attempt would be $f_1, f_2, \cdots, f_{j-2} = k, f_{j-1}, f_j, f_{j+1} = \frac{k}{2}, f_{j+2}, \cdots, f_N = 0$. Some new notation will be mentioned before introducing both parts of the algorithm

$$P^A_{\text{success}} = \sum_{\mathbf{s}\in\mathbf{A}} \prod_{i=1}^{N} p_i^{s_i}(1-p_i)^{1-s_i}$$

Minimum Redundancy Algorithm in Exponential Time

Part 1:

Step 1: Assign $j = 1$ and go to *step 2*.

Step 2: Let $A = S_{((2^{N-j+1:2^N}),(1:N))}$ and go to *step 3*.

Step 3: Calculate P^A_{success}

If $P^A_{\text{success}} \geq p^*$

$f_1, \ldots, f_j = k, f_{j+1}, \ldots, f_N = 0$

$\gamma_{\min} = j$, $P_{\text{temp}} = P^A_{\text{success}}$

Go to *Part 2* of the algorithm

else let $j = j + 1$

if $j > N$ move on to *Part 2*

else return to *Step 2*

If $j < 2$ then we have an optimal allocation and we are done. Otherwise:

Part 2:

Let $i = 2$ and $j = \gamma_{\min}$

Step 1: Let $i = i + 1$

if $i > N$ or $j - 2 + i > N$ then terminate Part 2

else go to *Step 2*

Step 2: Let $s = 2$ and go to *step 3*

Step 3:

if $j - 2 + \frac{i}{s} \leq j$

Let A denote the subset of matrix S composed of rows whose indices are in $Z^j_{(i,s)}$ (where $Z^j_{(i,s)}$ is the set defined above) followed with rows $(2^{N-(j-2)} + 1 : 2^N)$ of the matrix S, or

$$A = \begin{bmatrix} S_{\left((Z^j_{(i,s)}),(1:N)\right)} \\ S_{((2^{N-(j-2)}+1:2^N),(1:N))} \end{bmatrix}$$

Go to *step 4*

else Go to *step 6*

Step 4: Calculate P^A_{success}

if ($P^A_{\text{success}} \geq p^*$ with $j - 2 + \frac{i}{s} < j$) or ($j - 2 + \frac{i}{s} = j$ and $P_{\text{temp}} < P^A_{\text{success}}$)

Go to *step 5*

else Go to *step 6*

Step 5: Let $P_{\text{temp}} = P^A_{\text{success}}$, $\gamma_{\min} = j - 2 + \frac{i}{s}$, and

$f_1, \ldots, f_{j-2} = k$

$f_{j-1}, \ldots f_{j-2+i} = \frac{k}{s}$

$f_{j-1+i}, \ldots, f_N = 0$

Go to *step 6*

Step 6: Let $s = s + 1$

if $s > i$ Go to *step 1*

else Go to *step 3*

This algorithm is optimal for several of cases. One case is when we have $N = 3$ paths.

where A is some submatrix of S (S is a matrix filled with all possible length N binary vectors, with the first row being the all zero vector and the last row being the all one vector). $\sum_{\mathbf{s}\in\mathbf{A}}$ represents a sum which begins with the first row vector of A and terminates with the vector that is the last row of A. Let

$$Z^j_{(i,s)} = \{z \in \{1, \cdots, 2^{N-(j-2)}\} \mid \sum_{l=j-1}^{j-2+i} S_{z,l} \geq s\}$$

for some integers s, i, j. Where $S_{z,l}$ represents the element of S in the zth row and lth column. Let $S_{((i:j),(1:l))}$ ($i \geq j$ and $l \geq 1$) represent a submatrix of S composed of all the rows $i, i+1, \cdots, j$ of S up to the column l.

Theorem 2: MRAET is optimal when $N = 3$.

Proof: After Part 1 of the algorithm, we have 3 options for j, $j = 1, 2, 3$.

If $j = 1$, then the algorithm terminates after part 1 since $j < 2$. We are left with $\gamma_{\min} = 1 \Rightarrow n = k$.

Thus, $f_1 = k, f_2 = f_3 = 0$ which is optimal.

If $j = 2$, then

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

and $P_{\text{temp}} = p_1 + p_2 - p_1 p_2$.

The algorithm then steps into part 2. It starts and ends with the scenario $i = 3, s = 2$ since $N = 3$. It first checks is $j - 2 + \frac{i}{s} \leq \gamma_{\min} = j$. If so, then it searches through

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

rows $(1, \cdots, 2^{N-(j-2)}) = (1, \cdots, 8)$ such that the columns $(j-1, \cdots, j-2+i) = (1, \cdots, 3)$ sum to greater than or equal to $s = 2$. Then,

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

Then since $j - 2 + \frac{i}{s} = \frac{3}{2} < j = 2$, MRAET checks if $P_{\text{success}} = \sum_{\mathbf{s}\in\mathbf{A}} \prod_{l=1}^{3} p_i^{s_i}(1-p_i)^{1-s_i} = p_1 p_2 + p_2 p_1 + p_1 p_3 - 2 p_1 p_2 p_3 \geq p^*$. If so, then $\gamma_{\min} = \frac{3}{2}$ and $f_1, f_2, f_3 = \frac{k}{s} = \frac{k}{2}$, otherwise $\gamma_{\min} = 2$ and $f_1 = f_2 = k, f_3 = 0$.

Lastly, if $j = 3$ the algorithm stops before part 2 because $j - 2 + 3 > 3$. Thus, $f_1 = f_2 = f_3 = k$, which is the same allocation as we had above. □

Next, we prove a theorem to help us show another optimal case.

Theorem 3: Suppose we have k symbols allocated to paths $1, \cdots, i-1$ and 0 symbols allocated to the remaining of the N paths, resulting with probability of success $\hat{p}_{i-1}$. Then, if we let path i have k symbols, the probability of success is

$$\hat{p}_i = \hat{p}_{i-1} + p_i - \hat{p}_{i-1} p_i \tag{3}$$

Proof: By induction on the integer i.

Base Case: $i = 2$

We have $\hat{p}_{i-1} = \hat{p}_1 = p_1$, since we only have success if path 1 succeeds. If we let path 2 have k symbols, then we have a success solely if path 1 succeeds, if path 2 is the only successful one, or if they both succeed. This is equivalent to:

$$\hat{p}_2 = \hat{p}_1 p_2 + \hat{p}_1(1-p_2) + (1-\hat{p}_1)p_2 = \hat{p}_1 + p_2 - \hat{p}_1 p_2.$$

Inductive Hypothesis: Suppose (3) holds $\forall i \leq m-1$.

Inductive Step: Let $i = m$. Then by inductive hypothesis we know that $\hat{p}_{m-1}$ is the probability of success for the first $m-1$ paths having k symbols and the rest having 0. We can think of $\hat{p}_{m-1}$ as being the probability of success for one super path. Thus, if we let the mth path have k symbols, then we have success if only the super path is successful, the mth path is the only successful one, or if they are both successful. That is:

$$\begin{aligned} \hat{p}_m &= \hat{p}_{m-1} p_m + \hat{p}_{m-1}(1-p_m) + (1-\hat{p}_{m-1})p_m \\ &= \hat{p}_{m-1} + p_m - \hat{p}_{m-1} p_m. \end{aligned}$$

Hence the result holds $\forall i \in 2, \ldots N$. □

Theorem 4: MRAET is optimal when $j = N$.

Proof: If $j = N$, then we know that A is equal to S, excluding the all zero first row, $P^A_{\text{success}} \geq p^*$.

By Thm. 3 we know that the probability of success for the first $N-2$ paths having k symbols and the rest having 0 is:

$$\hat{p}_{N-2} = \hat{p}_{N-3} + p_{N-2} - \hat{p}_{N-3} p_{N-2} < p^*.$$

Thus, if we treat the first $N-2$ paths as one super path with probability $p_{\text{super}} = \hat{p}_{N-2}$, then the current problem can be mapped to the case where $N = j = 3$ since super path is path 1, $N-1$ is path 2, and N is our third path. □

Corollary 1: MRAET is optimal when $j = N - 1$

This results follows from the theorem above, since $j = N-1$ can be mapped to the case where $N = 3$ and $j = 2$.

For MRAPT we proceed similarly to MRAET but we use the success probability approximation, (2). This means that there is no need to search through the matrix S and further spending exponential running time by calculating the true probability of error. Hence, Part 2 for the MRAPT algorithm excludes the S matrix search, and when it terminates a vector $\mathbf{f}$ is returned. Next we will analyze the running time of these algorithms.

A. Running Time Analysis

We begin by analyzing the MRAET algorithm. We assume that S is computer offline. The matrix S has 2^N rows and the worst case scenario is if we have to search through the entire

matrix. Using a common logarithmic search algorithm like the binary search mentioned in [15], the running time of this process becomes $O(\log_2(2^N)) = O(N)$. The worse case running time ($A = S$) calculating the probability of success is $O(2^N)$. Thus, inside the loop running time becomes $O(N + 2^N)$. It can be seen that the outer loop takes $< N$ iterations, hence the total worst case running time of part 1 is $O(N(N + 2^N))$. Part 2 has one more outside loop of N iterations but the analysis is pretty much identical. This implies that MRAET's total worst scenario running time is $O(N^2(N + 2^N))$. Without a doubt MRAET is an exponential time algorithm.

The running time of MRAPT is significantly reduced since it is not necessary to traverse matrix S or to calculate the true success probability. Using (2) as the probability of success function, it is necessary to calculate the mean and variance of the Gaussian distribution $\sim \mathcal{N}\left(\sum_{i=1}^{N} f_i p_i, \sum_{i=1}^{N} f_i^2 p_i(1-p_i)\right)$. Assuming that the values $p_i(1-p_i)$ are saved, the mean and variance calculations each take $O(2N)$ iterations which is the worst case running time. Similar to the analysis for MRAET, the second part overbears the running time with its two loops resulting in $O(N^2(4N)) = O(N^3)$ running time for MRAPT. MRAPT thus runs in polynomial time with respect to the number of paths.

V. APPLICATION TO DIFFERENT CODES

Although these algorithms were developed for MDS codes, we can also apply them to LT and Raptor codes. These codes are almost MDS codes implying that instead of needing to receive exactly k symbols to decode the message, around $k(1+\varepsilon)$ are needed for decoding. Although this increase in overhead is not desirable, these codes have several advantages over the classic MDS codes. As mentioned above, both LT and Raptor codes have smaller encoding/decoding time than MDS codes. There are many instances where the network efficiency is less important than the cost of the encoder/decoder. Another advantage of these codes is that they are fountain codes, meaning that they are rateless. This is extremely important because in the situation where the decoder does not receive enough output symbols to decipher the transmitted message, the encoder can very simply send some more data independently of the output symbols which were received. An RS code, for example, would either require knowledge of the exact locations of missing symbols or would have to re-encode and retransmit the original message.

A. LT Code Implementation

Theoretically in an LT code, with probability of $1-\delta$, all k input symbols can be recovered from a set of $k(1+\varepsilon)$, where overhead ε depends on δ. To implement LT codes, we first decide on the value of δ which is used to determine the parameter $R = c\log(k/\delta)\sqrt{k}$. The next step is to establish the overhead ε. We use the Robust Solition distribution mentioned in an earlier section for the output symbol degree distribution. In [8], the authors show that using this distribution and to ensure a probability $1-\delta$ of successful decoding, the total output symbol size should be on the order of $K = k + O(\sqrt{(k)}\ln^2(k/\delta))$. In particular, $K = k + \sum_{i=1}^{k/R-1} \frac{R}{i} + R\ln(R/\delta)$ implying that $\varepsilon = \frac{1}{k}\left(\sum_{i=1}^{k/R-1} \frac{R}{i} + R\ln(R/\delta)\right)$. Our algorithm is used to determine total redundancy needed to ensure that the probability of success is above a certain threshold. This means that our algorithm passes $1-\delta$ and p^* as parameters and determines the symbol allocation/redundancy so that the calculated success probability is at least as large as p^*.

B. Raptor Code Implementation

We implement a raptor code using a regular LDPC code combined with an LT code. The first step is to determine the overhead of both codes. Based on the fraction of input symbols, δ, which we would like to decode using the inner LT code, we determine ε_{LT} which specifies the LT redundancy. Then, following the method used in [16], we let $\varepsilon_{\text{raptor}} = 2\varepsilon_{\text{LT}}$. Since $(1+\varepsilon_{\text{raptor}}) = (1+\varepsilon_{\text{LDPC}})(1+\varepsilon_{\text{LT}})$, $\varepsilon_{\text{LDPC}}$ becomes

$$\varepsilon_{\text{LDPC}} = \frac{\varepsilon_{\text{raptor}}}{2+\varepsilon_{\text{raptor}}}.$$

Next, using the raptor overhead, or $k(1+\varepsilon_{\text{raptor}})$, the total redundancy and symbol allocation are determined using either MRAET or MRAPT. Once the redundancy and the symbol allocation is specified, it is necessary to determine the amount of redundant check nodes of each code. Since the previous overheads represented the k out of n symbols similar to MDS codes, these redundancies no longer match the total code size. So the initial overhead used to determine the total amount of check nodes and symbol allocation was:

$$k' = k(1+\varepsilon_{\text{LDPC}})(1+\varepsilon_{\text{LT}}). \tag{4}$$

In order to determine the new redundancies, we keep in mind that we would like the ratio of the original redundant nodes of each code to be the same. We introduce two new constants which we would like to solve for: γ_{LT} and γ_{LDPC}. These constants represent the increase in the overhead for the LDPC and LT code to reach the size of the final codeword n. We have,

$$n = k(1+\gamma_{\text{LT}}\varepsilon_{\text{LT}})(1+\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}). \tag{5}$$

Initially there are $r_{\text{iLDPC}} = k\varepsilon_{\text{LDPC}}$ and $r_{\text{iLT}} = k(\varepsilon_{\text{LT}}+\varepsilon_{\text{LT}}\varepsilon_{\text{LDPC}})$ redundant nodes for the LDPC and LT code respectively. After finding the total codeword size n from the algorithm, the LDPC code has $r_{\text{LDPC}} = k\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}$ redundant symbols and the LT code has $r_{\text{LT}} = k(\gamma_{\text{LDPC}}\varepsilon_{\text{LT}} + \gamma_{\text{LT}}\gamma_{\text{LDPC}}\varepsilon_{\text{LT}}\varepsilon_{\text{LDPC}})$ redundant symbols. We solve for $\gamma_{\text{LT}}, \gamma_{\text{LDPC}}$ by setting the ratio of the final redundant symbols equal to that of the initial ones, or:

$$\begin{aligned}\frac{r_{\text{LT}}}{r_{\text{LDPC}}} = \frac{r_{\text{iLT}}}{r_{\text{iLDPC}}} &= \frac{\gamma_{\text{LDPC}}\varepsilon_{\text{LT}} + \gamma_{\text{LT}}\gamma_{\text{LDPC}}\varepsilon_{\text{LT}}\varepsilon_{\text{LDPC}}}{\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}} \\ &= \frac{\epsilon_{\text{LT}} + \epsilon_{\text{LT}}\epsilon_{\text{LDPC}}}{\epsilon_{\text{LDPC}}}.\end{aligned} \tag{6}$$

Using (5) we obtain:

$$\gamma_{\text{LT}} = \frac{\frac{n}{k(1+\gamma_{\text{LDPC}}\varepsilon_{\text{LT}})} - 1}{\varepsilon_{\text{LT}}}. \tag{7}$$

From (6) and (7), we acquire:

$$\gamma_{\text{LDPC}} = \frac{\frac{n}{k} - 1}{(\varepsilon_{\text{LDPC}} + \varepsilon_{\text{LT}} + \varepsilon_{\text{LDPC}}\varepsilon_{\text{LT}})}. \tag{8}$$

Hence the LDPC code has $k\gamma_{\text{LDPC}}\varepsilon_{\text{LDPC}}$ redundant nodes. We use a regular Gallagher code, meaning that the $n-k\times n$ parity check matrix has three '1's per column. The belief propagation algorithm is used for decoding.

To determine the degree distribution of the LT code, we use an idea discussed in [12] and [17]. In [12], the authors discuss the design of the LT degree distribution for finite length raptor codes. Based on keeping the expected ripple size of the LT code at $c\sqrt{k(1-x)}$, they determine that the LT degree distribution should meet this inequality:

$$\Omega'(x) \geq \frac{-\ln\left(1-c\sqrt{\frac{1-x}{k}}\right)}{1+\varepsilon_{\text{raptor}}} \tag{9}$$

with $x\in[0,1-\delta]$ and $\delta > c/\sqrt{(k)}$. The degree distribution can be solved as in [12], by discretizing x in the interval $[0,1-\delta]$ and forming a linear program that must meet the constraints in 9. In particular, the linear program is a minimization problem where the expected degree, $\Omega'(1)$, is minimized, or:

$$\begin{aligned} &\text{minimize} \quad \Omega'(1) \\ &\text{subject to} \quad \Omega'(x) \geq \frac{-\ln\left(1-c\sqrt{\frac{1-x}{k}}\right)}{1+\varepsilon_{\text{raptor}}} \\ &\qquad \sum_{i=1}^{D}\Omega_i = 1 \\ &\qquad x\in[0,1-\delta] \end{aligned} \tag{10}$$

where D represents the maximum degree. Using the overhead determined above for the LT code, we use this degree distribution in the final step of the encoding process to add redundancy to the LDPC code and forming the final codeword.

VI. SIMULATIONS AND COMPARISONS

All of our simulations are run over numerous Monte Carlo runs to accurately depict performance. We measure the performance of our algorithms by first comparing the success probability of MRAET and MRAPT in Fig. 3. Fig. 3 also shows the desired probability level as well as the success probability approximation to allow for a full comparison. The parameters we use for this plot are N=7, k=4, $p=[0.8000, 0.5901, 0.5338, 0.5261, 0.5203, 0.5107, 0.5000]^T$. MRAET algorithm performs very well and achieves a success probability that is higher or equal to p^*. MRAPT on the other hand seems to oscillate around p^*, but typically stays above p^*. This makes MRAPT particularly practical since it determines symbol allocation and redundancy in polynomial time. Comparing MRAPT to the approximation we used for the probability of success, it can be seen that this approximation seems to be off by a constant offset from the true performance.

Using the same parameters as used for 3, Fig. 4 shows a comparison of the redundancy ratios of MRAET and MRAPT vs. p^*. Due to the sub-optimality of MRAPT, the redundancy ratio of MRAPT is slightly higher than that of MRAET. The gap between the redundancy ratios is fairly small until p^* gets to be around 0.98 and there appears to be a large jump for the redundancy of MRAPT.

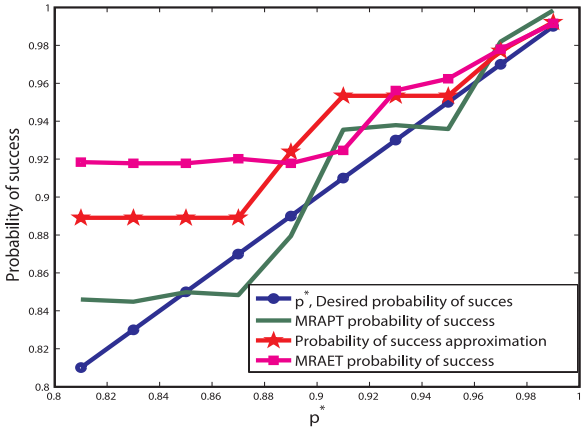


Fig. 3. Probability of success of MRAPT and MRAET.

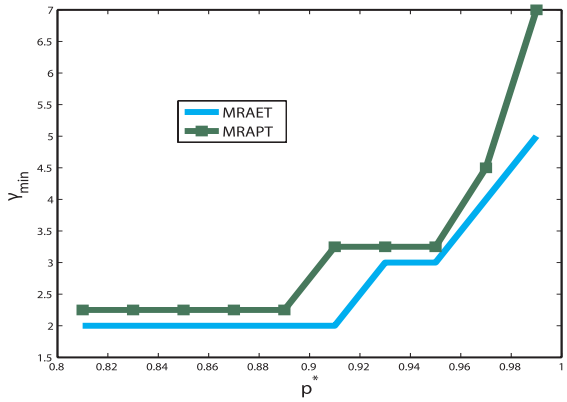


Fig. 4. Redundancy ratio for MRAPT and MRAET.

Figs. 5, 6, and 7 show the performance of MDS, LT, and Raptor codes using the MRAET algorithm for k=50. In the remaining figures we change the number of input symbols to k=50, and we evaluate the performance of MDS, LT, and Raptor codes using the MRAET algorithm. Fig. 5 shows the actual probability of successful decoding for MDS, LT, and Raptor codes as compared to the target probability of success. All of these codes have a successful decoding probability that is equal to or above p^*, making them excellent candidates for multipath channels. On average, the MDS code seems to have lower success probability than the other codes, though in Fig. 6 it can be seen that the MDS codes do have the minimum redundancy. The Raptor code has higher decoding success probability than the LT code for lower values of p^* but as p^* grows the LT code outperforms the Raptor code. Though in Fig. 6 LT code has significantly bigger codeword size than both Raptor and MDS codes.

The Raptor code seems to have a constant amount of extra output symbols than the MDS code. The code has more redundancy added but it outperforms the MDS code significantly as shown in Fig. 7. Fig. 7 depicts the average bit error probability for these three codes. The Raptor code has consistently lower bit error rate than both the other codes, and the LT code has lower bit error rate than the MDS code. The bit error rate for the Raptor code is extremely low and the LT error rate is higher by a relatively minute amount.

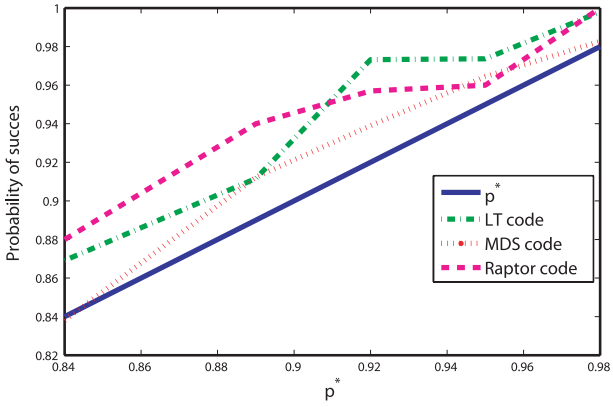


Fig. 5. Probability of success over different codes using MRAET.

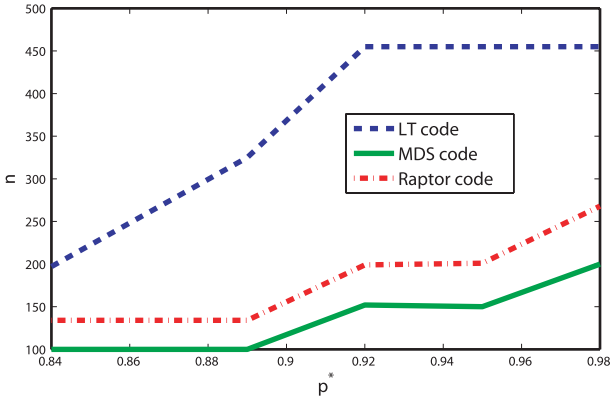


Fig. 6. Total codeword size for different codes using MRAET.

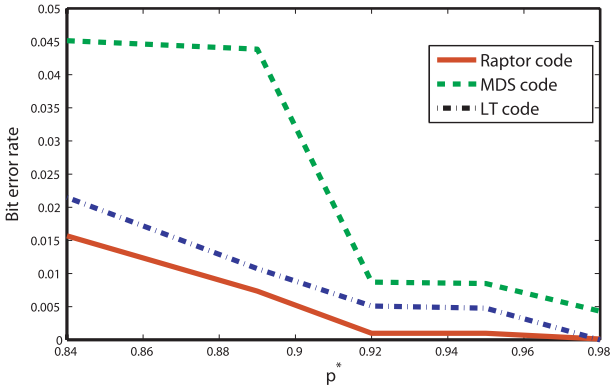


Fig. 7. Bit error rate over different codes using MRAET.

VII. CONCLUSION

We consider a network setup with a source and destination node and N independent paths separating them. These paths are erasure paths and they have a pre-determined "trustworthiness" level. We determined the minimum code length and message dispersal down the paths as to achieve a target success probability. Due to the inability to express the true success probability in closed form, we introduced two heuristic algorithms to determine the mentioned parameters. MRAET uses the actual success probability and results in an exponential running time. MRAPT on the other hand uses an approximation of the probability of success, which causes sub-optimal performance, though runs in efficient polynomial time. Both these algorithms are developed for the specific class of codes called MDS codes.

We apply MRAET to three codes, MDS, LT, and Raptor codes. The simulations showed that the Raptor code appears to be the best candidate for our algorithm in a multipath channel. It has a high success probability, along with very low bit error rate, and the code does not have a huge amount of redundancy difference over the MDS code. If one wishes to have higher network efficiency MDS codes would be the ideal choice since the redundancy is lower than the other codes. The problem with MDS codes is that the higher encoding/decoding time result in more complex and expensive encoders/decoders. Also, if an MDS code is used, and the necessary output symbols are not received then it is much more difficult to retransmit the missing information. Raptor and LT codes are rateless meaning it is very simple to re-encode and transmit missing output symbols. Another advantage of these codes is the fact that they have cheap encoding/decoding costs and result in low bit-error rate.

REFERENCES

[1] P. Papadimitratos and Z.J. Haas, "Secure message transmission in mobile ad hoc networks," *Ad Hoc Netw.*, pp.193–209, 2003.

[2] A. Kacewicz and S.B. Wicker, "Optimizing redundancy using MDS codes and dynamic symbol allocation in mobile ad hoc networks," in *Proc. Conference on Information Sciences and Systems*, Princeton University, Mar. 2008.

[3] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *Proc. IEEE ICC*, 2001, pp. 3201–3205.

[4] A. Tsirigos and Z. J. Haas, "Analysis of multipath routing, part 1: The dffect on the packet delivery ratio," *IEEE Trans. Wireless Commun.*, vol. 3, no. 1, Jan. 2004.

[5] A. Tsirigos and Z. J. Haas, "Analysis of multipath routing, part 2: Mitigation of the effects of frequently changing network topologies," *IEEE Trans. Wireless Commun.*, vol. 3, no. 2, Mar. 2004.

[6] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, pp. 335–348, 1989.

[7] S. B. Wicker and V. K. Bhargava, *Reed Solomon Codes and Their Applications*, Piscataway: IEEE Press, 1994.

[8] M. Luby, "LT-codes," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[9] S. B. Wicker. *Error Control Systems for Digital Communication and Storage*. Upper Saddle River, NJ: Prentice Hall, 1995.

[10] David J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[11] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proc. IEEE ISIT*, Chicago, June 2004.

[12] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[13] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT codes," in *Proc. IEEE ISIT*, Seattle, WA, July 2006.

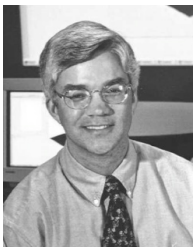
[14] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, John-Wiley and Sons, Inc. 2nd ed., 2006.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and L. Stein, *Introduction to Algorithms*. 2nd ed., MIT Press, 2001.

[16] P. Cataldi, M. P. Shatarski, M. Grangetto, and E. Magli, "Implementation and performance evaluation of LT and Raptor codes for multimedia applications," *Intelligent Information Hiding and Multimedia Signal Processing*, 2006.

[17] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp.569–584, Feb. 2001.

Anna Kacewicz is currently working on her Ph.D. in Electrical and Computer Engineering at Cornell University. She received her B.S. in Electrical and Computer Engineering from University of Texas at Austin and her M.S. degree from Cornell University. Her research interests include coding and information theory, information security and reliability, and wireless network security.



Stephen B. Wicker is a Professor of Electrical and Computer Engineering at Cornell University, and a member of the graduate fields of Computer Science and Applied Mathematics. He was awarded the 1988 Cornell College of Engineering Michael Tien Teaching Award, the 2000 Cornell School of Electrical and Computer Engineering Teaching Award, and the 2009 Cornell College of Engineering Douglas Whitney Teaching Award. As of 2009, he has supervised thirty-five doctoral dissertations. He is the author of Codes, Graphs, and Iterative Decoding (Kluwer, 2002), Turbo Coding (Kluwer, 1999), Error Control Systems for Digital Communication and Storage (Prentice Hall, 1995), and Reed-Solomon Codes and Their Applications (IEEE Press, 1994). He has served as Associate Editor for Coding Theory and Techniques for the IEEE Transactions on Communications, and is currently Associate Editor for the ACM Transactions on Sensor Networks. He has served two terms as a member of the Board of Governors of the IEEE Information Theory Society, and chaired the Technical Program Committee for the Fifth International Conference on Information Processing in Sensor Networks (IPSN 2006). He teaches and conducts research in wireless information networks and sensing systems. His research has focused on the development and application of advanced technologies for adaptive networks. He is also conducting joint research with the Berkeley School of Law on privacy policy and the impact of the deployment of sensor networks in public spaces. Current interests include the application of artificial intelligence and game theory to self-configuring wireless sensor networks. Such networks are being developed for disaster recovery, infrastructure monitoring, and national security. Other interests include network security, the behavior of complex systems, and the evolution of the nation's telecommunications networks. He is the Cornell Principal Investigator for the TRUST Science and Technology Center a National Science Foundation center dedicated to the development of technologies for securing the national critical infrastructure.