ENHANCED RATELESS CODING AND COMPRESSIVE SENSING FOR EFFICIENT DATA/MULTIMEDIA TRANSMISSION AND STORAGE IN AD-HOC AND SENSOR NETWORKS

By

ALI TALARI

Bachelor of Science in Electrical Engineering
University of Kashan
Kashan, Iran
2003

Master of Science in Electrical Engineering
Sharif University of Technology
Tehran, Iran
2007

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2012

ENHANCED RATELESS CODING AND COMPRESSIVE SENSING FOR EFFICIENT DATA/MULTIMEDIA TRANSMISSION AND STORAGE IN AD-HOC AND SENSOR NETWORKS

Dissertation Approved:

Dr. Nazanin Rahnavard

Dissertation Advisor

Professor Qi Cheng

Professor Damon Chandler

Professor Johnson P Thomas

Outside member

ACKNOWLEDGMENTS

There are so many people in these five years who inspired me throughout my journey in pursuing my PhD. First of all, I would like to thank my advisor, Dr. Nazanin Rahnavard, whose never ending support, encouragements, and endless hours of help and devotion made this achievement possible. Further, I would like to thank my dissertation committee Professor Qi Cheng, Professor Damon Chandler, and Professor Johnson P Thomas for their support as I moved from an idea to a completed study. In addition, I thank Professor Sunil Kumar from San Diego State University for extensive contributions on the study reported in Chapter 7 both on the novel video encoding schemes and performing extensive simulations. I would also like to thank my colleague and my close friend Behzad Shahrasbi for hours of insightful discussions in my progress towards preparation of this study. I also especially thank my beloved parents and sister whose never ending support made it possible to withstand all the challenges throughout my PhD studies.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

When data, movie, voice, etc. are transmitted between a server and a client, e.g., a cell phone receiving live video stream from the Internet, an stream of bits are transmitted over wired and wireless channels. Communication channels are known to be contaminated with noise coming from various sources. For instance, white noise naturally exists in all channels and all electrical components, and interference may leak from other systems working in close frequency bands.

To compete with the inevitable noise of the channel and protect the transmitted bits many *forward error correction* (FEC) codes have been designed. FEC coding schemes receive series of bits as input and intelligently append redundant bits to the bit stream such that the incorrectly received bits can be *detected* and partially *corrected* at the receiver. FEC codes are broadly employed in communication systems such as hard disks, home internet services, cell phone communications, satellite radio, and almost anywhere information is sent over unreliable channels.

Originally designed FEC codes such as *convolutional codes* are designed to protect *bits* from noise, hence they are referred to by *bit-level* FEC codes. The bit-level FEC codes are suitable for lower *layers* of network, which are responsible with the delivery of bits over a single hop (point-to-point), namely *link layer* (LL) and *physical layer* (PL). The *code rate* $R = \frac{k}{n}$ defines the number of redundancy bits added by FEC code, where k and n are the number of input bits and encoded bits, respectively. Since in these codes the number of redundancy bits is fixed by the structure of the code, they are called *fixed-rate* codes.

Recently, with the advent of the Internet and modern digital communications a new class of *application layer* (AL) FEC codes, referred to by *rateless* or *fountain* codes, have been proposed [1, 2]. Rateless codes may encode large *symbols* rather than bits, where each symbol may contain one to thousands of bits. Therefore, these codes are referred to as *packet-level* FEC codes.

When FEC coding is present in the lower layers (LL and PL), AL only observes *success* or *failure* in the delivery its transmitted symbols. Such a channel is modeled by *erasure channel*, the term which was first introduced by Peter Elias of MIT in 1954. In erasure channels, each symbol is either lost (hence its value will be unknown) with probability ε during the transmission or is delivered correctly with probability $1 - \varepsilon$.

The idea behind rateless coding is that the encoder can potentially generate *unlimited* number of *output symbols* (encoded data) from k *input symbols* (source data). The encoder continues until $\gamma_{succ}k$ output symbols are collected at the receiver for which a full decoding of k input symbols is possible with high probability. Therefore, a rateless codes can adapt to any erasure channel with varying or unknown ε since only the number of correctly delivered symbols is important at decoder. γ_{succ} is called the coding overhead and is slightly larger than 1. Therefore, no coding rate can be defined for these FEC codes, thus the term *rateless* is employed.

LT codes [1] were the first practical implementation of rateless codes. Later, *raptor* codes [2] complemented LT codes by providing linear time rateless encoding and decoding algorithms. Ratelessness of these codes makes them very suitable for multimedia content delivery for wireless devices. Recently, raptor codes have been standardized for 3GPP Multimedia Broadcast/Multicast Services [3].

Despite the flexibility and advantages of rateless codes, when the received overhead is less than γ_{succ}, i.e., the transmission is still in progress and the received overhead γ is $\gamma < \gamma_{succ}$, almost no input symbols may be recovered from the set of already received output symbols. In other words, rateless codes have weak *intermediate* performance

[4]. Therefore, in Chapter 3 we investigate methods that improve the intermediate performance of rateless codes. First, we assume no information about ε is available at the encoder and design *LT-like* rateless codes that are concurrently optimal in selected overheads in intermediate region $0 \leq \gamma \leq \gamma_{succ}$ [5, 6]. Next, we assume an estimate of the channel erasure rate is available at the encoder. Therefore, the encoder can generate encoded symbols ahead of transmission and *reorder* them to improve the intermediate recovery of the input symbols. We propose a greedy algorithm that reorders the encoded symbols based on the dependencies of encoded symbols in decoding procedure [6, 7]. We will see that these two methods greatly improve the performance of these codes in intermediate region. As a practical application example, we employ our designed codes to improve the intermediate data delivery in *Delay Tolerant Networks* (DTN) [8].

DTNs are *sparse ad-hoc* networks with *mobile* nodes that meet opportunistically. Therefore, regular ad-hoc routing algorithms with routing tables may not be employed in these networks. The effective routing algorithms in these networks are called store-carry-and-forward, where nodes receive a packet from the source or intermediate nodes and *carry* it until a forwarding opportunity to a node with higher probability of meeting the destination arises. Since rateless codes can potentially generate unlimited number of *independent* output symbols, they are very suitable for data delivery in DTNs because the decoding can be performed as soon as $\gamma_{succ}k$ output symbols are collected.

Besides DTNs, rateless codes can improve the performance of data collection in *wireless sensor networks* (WSN). In WSNs, small energy constrained sensing devices are spread in an area to capture an event or to monitor a natural phenomenon. Conventional rateless codes are designed for point-to-point data delivery and are extensively employed in WSNs routing algorithms. However, in WSNs the data collection may be *distributed*, where nodes send their information to the *base* through

a *relay* node. Therefore, conventional point-to-point rateless codes may be suboptimal in these scenarios. Further, the collected data from various sources may have *unequal importance*. Consequently, we investigate the *distributed* rateless codes with the capability of providing *unequal error protection* (UEP) in Chapter 4

We define the construction (encoding and decoding) of these codes and design distributed UEP-rateless (DU-rateless) codes for two sources [9, 10]. We will analyze these codes and investigate their successful decoding probability. We will show that these codes surpass regular LT codes for distritbuted data collection.

Moreover, we investigate the performance of LT codes in the presence of feedback. The original LT codes only required a single-bit feedback to inform the transmitter from the reception of $\gamma_{succ}k$ output symbols. However, in many cases a low bandwidth and weak feedback channel exists, which remains *unused* when regular LT codes are employed. Consequently, in Section 5 we design LT codes with smart feedback (LT-SF codes) that can greatly take advantage of the feedback channel and considerably decrease γ_{succ} [11].

Rateless codes can easily provide UEP for various parts of k input symbols by slightly modifying the encoding procedure. The UEP property can extensively improve the data delivery performance when various parts of data have unequal importance. The best examples of such data are MPEG and H.264 video streams. These video streams are constructed from I, P, and B frames, where I frames have the highest importance and B frames have the lowest importance. Therefore, in Chapter 6 we take advantage of UEP property of rateless codes and investigate how these codes may improve MPEG video delivery [12]. Next, we show that UEP-rateless codes may effectively reduce the initial startup delay (the buffering time) in *Video-on-Demand* (VOD) services [13].

Although rateless codes may be efficiently employed to design efficient video transmission schemes, PL FEC codes may be also employed to perform the video encoding.

Therefore, we investigate an H.264 video transmission scheme with fixed FEC coding at PL and rateless coding at AL. We investigate the interaction of these two codes and their optimal setup since both layers are capable of providing UEP. Further, both FEC codes share the same bandwidth to add their redundancy. Therefore, we need to investigate the optimal allocation of the available bandwidth to these two codes. Consequently, in Chapter 7, we study various setups of FEC coding at AL and PL for efficient video transmission in H.264 video delivery.

Recently, *compressive sensing* (CS) techniques, which have close connection to FEC codes, have shown that a *compressible* signal may be recovered from its undersampled *random projections* (also called *measurements*). A signal is said to be compressible when its coefficients have a form of correlation; hence, the signal may be represented with a *sparse* signal in an *appropriate* transform domain. Recently, CS algorithms have been employed in *data storage* algorithms in WSNs. In data storage algorithms in WSNs, we are interested in increasing the *persistency* of nodes data by disseminating them at all nodes such that they all can be recovered by sampling a small subset of nodes. To store the samples of readings at node both rateless coding and CS techniques have been employed. The goal of the problem is to minimize the total number of transmissions to realize the storage.

In Chapter 8, we design a novel data storage algorithms using CS for WSNs *without* routing tables [14]. We exploit the *broadcast* property of wireless channels and employ the well-known *probabilistic broadcasting* (PB) algorithm to disseminate nodes readings [14]. We will show that our proposed algorithm based on PB considerably reduce the total number of transmissions. PB has a parameter, called *forwarding probability*, that should be tuned based on network parameters to realize minimum number of transmissions. Therefore, the performance of PB is affected by network changes and incorrectly selecting the forwarding probability. Consequently, we design a *parameterless* data dissemination algorithm referred to by *alternating branches*

(AB). AB automatically adapts to network changes and performs almost equally for a wide range of network parameters. Further, we will show that the total number of transmissions will be even further reduced using AB.

Finally, Chapter 9 summarizes the dissertation and describes potential extensions and future research directions of this work.

CHAPTER 2

BACKGROUND

In this section, we provide a brief background to original rateless codes and UEP-rateless codes. Next, we describe the *And-Or tree* analysis which is a mathematical tool to analyze *asymptotic* behavior of rateless decoding. Further, we describe MPEG video stream structure and describe its need for UEP. Moreover, we provide a brief introduction to video-on-demand systems, which may also benefit from UEP-rateless coding as described later.

Since we will investigate a video transmission scheme with PL fixed-rate coding and AL rateless coding, we provide a brief review on rate-compatible convolutional codes, which are bit-level fixed-rate codes. Furthermore, we provide a brief overview on emerging compressive sensing techniques and basics of probabilistic broadcasting. Next, we briefly review delay tolerant networks. Finally, we describe the basics of single and multi objective *genetic algorithms* optimization algorithms.

2.1 Rateless Encoding/Decoding

Rateless codes have a simple encoding and decoding procedures over erasure channels. In this dissertation, without loss of generality and for simplicity we assume that the input and output symbols are binary symbols, while they may contain several thousands of bits.

Rateless encoding: In rateless encoding of k input symbols, first an output symbol degree, d, is chosen from a degree distribution $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$, where Ω_i is the probability that $d = i$. The probability distribution is also shown by its generator

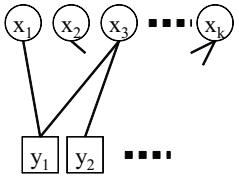


Figure 2.1 The rateless encoding of two output symbols.

polynomial $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$. Next, d input symbols are chosen *uniformly at random* from k input symbols and are *XOR*ed together to generate an output symbol. We refer to the d contributing input symbols in forming an output symbol as its *neighbors*. We can see that potentially unlimited number of *independent* output symbols can be generated in rateless coding.

The input and output symbols of a rateless code can be viewed as vertices of a bipartite graph T, where the input symbols are the *variable nodes* and the output symbols are the *check nodes* [15, 16]. In Figure 2.1, we have shown the encoding of two output symbols, where *variable nodes* (circles) and *check nodes* (squares) represent input and output symbols, respectively. We can see that y_1 has degree $d = 2$ and is formed by XORing x_1 and x_3, and y_2 has degree $d = 1$ and only contains x_3. In this dissertation, we interchangeably employ the terms input and output symbols with variable and check nodes, respectively.

The degree distribution $\Omega(x)$ is usually finely tuned such that $\gamma_{succ}k$ output symbols can recover k input symbols with high probability. The degree distribution employed in LT codes is called the *Robust-Soliton* (RS) distribution ($\mathrm{RS}(x) = \sum_{i=1}^{k} \mathrm{RS}_i x^i$), which is obtained by combining *ideal-Soliton* (IS) distribution $\mathrm{RS}^I(x)$ and distribution $\mathrm{RS}^1(x)$. $\mathrm{RS}^I(x)$ and $\mathrm{RS}^1(x)$ are given by

$$\mathrm{RS}_i^I = \begin{cases} \frac{1}{k} & i = 1, \\ \frac{1}{i(i-1)} & i = 2, \ldots, k, \end{cases} \tag{2.1}$$

and

$$\mathrm{RS}_i^1 = \begin{cases} \frac{L}{ik} & i = 1, \ldots, \frac{k}{L} - 1, \\ \frac{L}{k}\ln(\frac{R}{\delta}) & i = \frac{k}{L}, \\ 0 & i = \frac{k}{L} + 1, \ldots, k, \end{cases}$$

respectively, where $L = c\ln(\frac{k}{\delta})\sqrt{k}$, and δ and c are two *tuneable* parameters [1]. It is easy to show that the average degree of output symbols with IS distribution is $\sum_{i=1}^{k} i\mathrm{RS}_i^I = \mathrm{RS}'^I(1) = H(k) \approx \ln k$ [1], where $\mathrm{RS}'^I(x)$ is the first derivative of $\mathrm{RS}^I(x)$ with respect to its variable x, and $H(k)$ is the k^{th} Harmonic number [1]. Finally, $\mathrm{RS}(x)$ degree distribution is obtained by

$$\mathrm{RS}_i = \frac{\mathrm{RS}_i^I + \mathrm{RS}_i^1}{\sum_{j=1}^{k} \mathrm{RS}_j^I + \mathrm{RS}_j^1}, i = 1, \ldots, k. \tag{2.2}$$

The average degree of output symbols generated by $\mathrm{RS}(x)$ increases as k increases. Therefore, the decoding complexity of LT codes is not linear in time [2]. Shokrollahi [2] proposed raptor codes by concatenating LT codes with a *pre-coding* phase with a conventional fixed-rate code with code rate R close to 1. In this way, the average degree of LT coding phase does not increase with k and the decoding complexity becomes linear in time ($O(1)$ operations per output symbol). Shokrollahi designed degree distributions for various k's in [2]. For instance, the optimal degree distribution for raptor codes of length $k = 65536$ is as follows

$$\begin{aligned} \Omega_{shok}(x) =& 0.007969x + 0.493570x^2 + 0.166220x^3 + 0.072646x^4 + 0.082558x^5 \\ &+ 0.056058x^8 + 0.037229x^9 + 0.055590x^{19} + 0.025023x^{65} + 0.003135x^{66}. \end{aligned} \tag{2.3}$$

Rateless decoding: The decoding procedure of rateless codes is performed *iteratively*. At each iteration, an output symbol is found such that the value of all but one of its neighboring input symbols is known. The value of the unknown input symbol is computed by a simple XOR. This step is applied iteratively until no more such output symbols can be found. Assume y_1 and y_2 of Figure 2.1 have been correctly

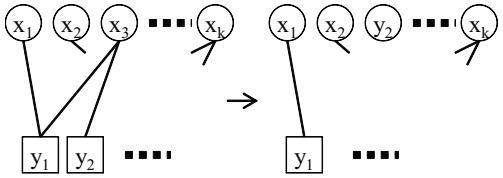


Figure 2.2 The iterative rateless decoding of two output symbols.

received at the receiver after transmission over an erasure channel. To perform the encoding, first y_2 (and any other degree 1 output symbol) decode the value of their only neighbor. Next, several output symbols similar to y_1, as shown in Figure 2.2, are reduced to degree $d = 1$ and become decodable.

It has been shown that when RS(x) is employed for $\gamma_{succ} = 1$ the decoding is asymptotically successful (all k input symbols are decoded) with high probability. Note that the set of output symbols reduced to degree-one is called the *ripple*. If the ripple becomes empty the decoding stops and the decoder needs to wait for new output symbols to join the ripple to proceed the decoding. In addition, when an output symbol in the ripple decodes an input symbol, its degree reduces to zero and is removed from the decoding process.

Although distribution RS(x) is asymptotically capacity achieving, i.e., $\gamma_{succ} \to 1$ as $k \to \infty$ [1], when k is finite γ_{succ} becomes significantly larger than 1 [1, 17, 18], which may result in an inefficient FEC coding. As we later show, a feedback channel can be used to obtain a much smaller γ_{succ} for a finite k.

2.2 Unequal Error Protection Rateless Codes

Most existing FEC codes provided *equal error protection* (EEP) of the k input symbols. However, in many applications, e.g., multimedia content coding, various parts of k input symbols have unequal importance; hence, they need *non-uniform* protection. In [19, 20], authors showed that rateless codes may easily provide UEP by changing the source symbol selection from uniform to *non-uniform*.

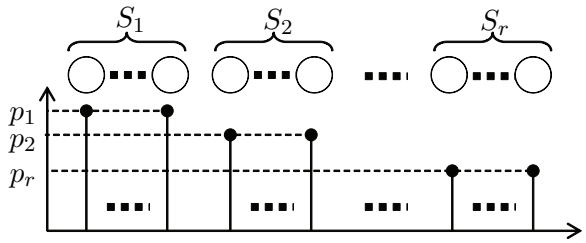


Figure 2.3 Input node selection with non-uniform probability in UEP-rateless encoding, where nodes with higher importance are selected with a higher probability.

In UEP-rateless codes, k input symbols are partitioned into r sets, $S_1, S_2, \ldots, S_r$ of sizes $s_1k, s_2k, \ldots, s_rk$, such that $\sum_{j=1}^{r} s_j = 1$. Let p_i be the probability that an input symbol from set S_i is chosen to form an output symbol as shown in Figure 2.3. Clearly, $p_i = \frac{1}{k}$ provides the simple EEP coding. Further, let us define the *protection level* of S_i as $k_i = p_ik$, where $\sum_{j=1}^{r} k_j s_j = 1$, and $k_i = 1$ provides the EEP.

With this setup, if we set $k_i > 1$ for a particular S_i, the input symbols from this section are included in larger number of output symbols. In this way, input symbols from S_i will have a higher probability of being decoded compared to case when $k_i = 1$. This probability is discussed in detail in the following section.

2.3 And-Or Tree Analysis of Rateless Codes

Let us briefly review And-Or tree analysis technique [21], which has been employed to analyze iterative rateless decoding on erasure channels for asymptotic cases (large k). An And-Or tree T_l is a tree of depth $2l$ with a root at depth 0, and nodes at depth i have children at depth $i + 1$. Further, nodes at *even* and *odd* depths are *OR-nodes* and *AND-nodes*, respectively, and evaluate logical OR and AND operations on the value of their children. Note that the OR-nodes at depth 2 in T_l are the roots for independent And-Or tree T_{l-1}.

Assume OR-nodes have i children with probability $\delta_i, i \in \{0, 1, \ldots, A\}$ and AND-nodes have j children with probability $\beta_j, j \in \{0, 1, \ldots, B\}$. Each OR-nodes at depth

$2l$ is independently assigned a value of 0 or 1, with y_0 being the probability that it is 0. Further, OR-nodes and AND-nodes with no children have the values 0 and 1, respectively. With this setup, y_l, the probability that the root node evaluates to 0, can be obtained from the following lemma [21].

Lemma 2.1 *Let y_l and y_{l-1} be the probabilities that the root nodes of And-Or trees T_l and T_{l-1} evaluate to 0, respectively. We have $y_l = f(y_{l-1})$, where*

$$f(x) = \delta(1 - \beta(1 - x)), \delta(x) = \sum_{i=0}^{A} \delta_i x^i, \beta(x) = \sum_{i=0}^{B} \beta_i x^i. \tag{2.4}$$

Following [2, 22], we can rephrase the iterative rateless decoding algorithm as following. At every iteration of the algorithm, messages (0 or 1) are sent along the edges from check nodes to variable nodes, and then from variable nodes to check nodes. A variable node sends 0 to an adjacent check node if and only if its value is not recovered yet. Similarly, a check node sends 0 to an adjacent variable node if and only if it is not able to recover the value of the variable node. In other words, a variable node sends 1 to a neighboring check node only if it has received at least one message with value 1 from its other neighboring check nodes. Also a check node sends 0 to a neighboring variable node only if it has received at least one message with value 0 from its other neighboring variable nodes. Therefore, we see that variable nodes indeed do the logical OR operation, and the check nodes do the logical AND operation. We can use the results of Lemma 2.1 on a subgraph T_l of T (T being the rateless encoding graph as shown in Figure 2.1) to find the probability that a variable node is not recovered after l decoding iterations (its value evaluates to zero). We choose T_l as following. Choose an edge (v, w) uniformly at random from all edges in T. Call the variable node v the root of T_l. Subgraph T_l is the graph induced by v and all neighbors of v within distance $2l$ after removing the edge (v, w). It can be shown that T_l is a tree asymptotically [21]. We can map each check node to an AND-

node and each variable node to an OR-node, and set $\beta_i = \frac{(i+1)\Omega_{i+1}}{\Omega'(1)}$ and consequently $\beta(x) = \frac{\Omega'(x)}{\Omega'(1)}$ and set $\delta(x) = \exp(n\Omega'(1)\gamma(x-1))$ [19, 20]. Recovery of each variable node can be mapped to evaluating the root of the tree T_l to zero. Thus have

$$y_l = \exp(\Omega'(1)\gamma\beta(1-y_{l-1})), l \geq 1 \tag{2.5}$$

in which $\beta(x) = \Omega'(x)/\Omega'(1)$ and $y_0 = 1$.

Similarly, the And-Or tree can be mapped to UEP-rateless codes [19, 20]. Let $y_{l,j}$ be the probability that a source symbol in s_j is not recovered after l rateless decoding iterations at the receiver. For $j = 1, \ldots, r$ we have [19, 20]

$$y_{l,j} = \delta_j(1 - \beta(1 - \sum_{m=1}^{r} p_m s_m k y_{l-1,m})), l \geq 1 \tag{2.6}$$

where $y_{0,j} = 1$ and $\delta_j(x) = \exp(kp_j\Omega'(1)\gamma(x-1))$.

It can be shown that sequences $\{y_{l,j}\}_l, \forall j$ converge to a fixed point y_j [19, 20], where y_j is the final decoding error rate of symbols in set $j \in \{1, 2, \ldots r\}$ for a UEP-rateless code with the parameters $\{\Omega(x), \gamma, S_1, S_2, \ldots, S_r, p_1, p_2, \ldots, p_r\}$.

Let $G_{l,i,j} \triangleq \frac{y_{l,i}}{y_{l,j}} = \exp(n(p_j - p_i)\Omega'(1)\gamma\beta(1 - \sum_{m=1}^{r} p_m s_m k y_{l-1,m}))$, which compares the recovery probabilities of nodes in S_i and S_j. A larger $G_{l,i,j}$ maps to higher recovery probability of the nodes in S_j in comparison to S_i. Therefore, for $p_j > p_i$ we have $G_{l,i,j} > 1$, which confirms that a higher selection probability from a particular set of input symbols to form output symbols results in higher recovery probability of nodes in that set. Therefore, desired UEP may be realized using UEP-rateless codes by appropriately setting $p_i, i \in \{1, 2, \ldots, r\}$.

In a simple case of UEP-rateless codes, the source data is divided into two parts ($r = 2$) with higher and lower priorities [19, 20]. Let αk with ($0 < \alpha < 1$) be the number of *more important symbols* (MIS), and $(1-\alpha)k$ be the number of *less important symbols* (LIS). We set the importance levels $k_M = p_1 k$ and $k_L = p_2 k$ for $0 < k_L < 1$ and $k_M = \frac{1-(1-\alpha)k_L}{\alpha}$. Let $y_{l,M}$ and $y_{l,L}$ denote the decoding error rate of

MIS and LIS at the l^{th} decoding iteration, respectively. We have [19, 20]

$$y_{l,M} = \exp\left(-k_M \Omega'(1)\gamma\beta(1-(1-\alpha)k_L y_{l-1,L} - \alpha k_M y_{l-1,M})\right), \tag{2.7}$$

$$y_{l,L} = \exp\left(-k_L \Omega'(1)\gamma\beta(1-(1-\alpha)k_L y_{l-1,L} - \alpha k_M y_{l-1,M})\right), \tag{2.8}$$

with $y_{0,L} = y_{0,M} = 1$.

For a given overhead γ, we have $y_M < y_L$, where y_M and y_L are the fixed point for convergent sequences $\{y_{l,M}\}_l$ and $\{y_{l,L}\}_l$, respectively. We could also fix the target decoding error rates of MIS and LIS and compare γ_{MIS} and γ_{LIS}, which are the overheads needed for MIS and LIS to reach the target error rate, respectively. We have $\gamma_{MIS} < \gamma_{LIS}$. This means that decoding error rate of MIS reaches a target error rate faster (smaller overhead) than the error rate of LIS. Therefore, the *unequal recovery time* (URT) property is also provided by UEP-rateless codes. Later, we take advantage of URT provided by UEP-rateless codes in video transmission schemes.

2.4 Introduction to MPEG and H.264 Video Format

MPEG video encoders, including MPEG-I, MPEG-II, and H.264 (MPEG-IV) encoders, generate three types of frames I, P, and B from the source raw video stream. The encoded frames are then grouped into batches of frames, called *group of pictures* (GOP), which contain one I-frame and several P- and B-frames. The structure of a GOP is determined by two numbers, M and N. M refers to the distance between P-frames, which is also the distance between an I-frame and the first P-frame, and N defines the distance between two I-frames, which is also the length of the GOP. For instance, the structure of a GOP defined by $M = 3$ and $N = 12$ is IBBPBBPBBPBB.

In MPEG video decoders, first the I-frame is recovered independently. Next, P-frames are recovered using the previous I-frame and P-frames. Finally, the B-frames are constructed using preceding and succeeding I- or P-frames. The last $(M-1)$ B-frames in each GOP are decoded using the previous I-frame and P-frames and

also the next GOP's I-frame. The structure of a GOP and frame dependencies are depicted in Figure 2.4.

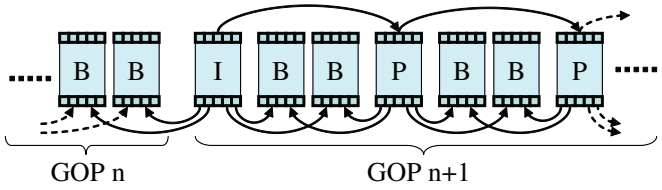


Figure 2.4 Different frame types in a GOP of an MPEG video stream. Arrows show frame dependencies for reconsecration.

One can see that if an error occurs in an I-frame, it propagates throughout the GOP, and if a P-frame is defected the error propagates until the next I-frame. On the other hand, a defected B-frame, in the worst case, causes only one frame drop. Therefore, the I-frames have the highest level of importance, and in contrast, B-frames have the lowest importance. Therefore, I-frames have the highest importance, while B-frames have the lowest importance. As we later show, by having more protection on more important frames the quality of the delivered video will considerably increase.

The quality of a video stream can be measured with various quality metrics such as *peak signal-to-noise ratio* (PSNR), *mean opinion score* (MOS), *symbol delivery ratio*, *decodable frame rate* (Q), etc. From an error control coding view, we choose decodable frame rate, Q, to measure the quality of the decoded video. The reason is that Q can mathematically be formulated, and it closely reflects the behavior of PSNR [23–29]. The value of Q varies between 0 and 1, where a larger value shows a higher frame recovery rate. Q is defined as

$$Q = \frac{E[\text{Number of decoded frames}]}{\text{Total number of transmitted frames}} = \frac{N_{dec_I} + N_{dec_P} + N_{dec_B}}{N_{total}}, \tag{2.9}$$

where N_{dec_I}, N_{dec_P}, and N_{dec_B} are the expected number of decodable frames of each type, and N_{total} is the total number of frames in the video.

Before the frames can be encoded using rateless codes, they need to be split into smaller transmittable symbols. Due to error concealment techniques employed in video source decoding techniques, a frame is decodable when ν fraction of its symbols is delivered, where ν is called the *decodable threshold*. For instance, $\nu = 0.8$ means that the decoder can tolerate 20% symbol loss. ν is determined based on application and the type of video decoder.

All MPEG streams MPEG-I, MPEG-II, and H.264 videos have the same frame dependencies as shown in Figure 2.4. To provide more decoding flexibility compared to MPEG-I and MPEG-II video streams, in H.264 AVC (MPEG-IV) each frames is broken into much smaller square shaped *macroblocks*. The macroblock are grouped together and *slices* are formed, which have fixed sizes. Slices may be decoded independently and partially recover a part of frame. Therefore, we will have *micro*-dependencies compared to frames. Note that still slices from I-frames have the highest importance. However, the slices from the same I frames can also have unequal importance. Therefore, in this case non-uniform protection is more grained and is defined in slice level.

H.264 slices can be prioritized based on their *distortion* contribution to the received video quality [30–35]. We employ the *cumulative mean square error* (CMSE) metric to measure the total distortion caused by loss of a slice, which takes into consideration the error propagation within the entire GOP [31]. Let the original uncompressed video frame at time t be $f(t)$, the decoded frame without the slice loss be $\hat{f}(t)$, and the decoded frame with the slice loss be $\tilde{f}(t)$. Assuming that each frame consists of $N \times M$ pixels, the MSE introduced by the loss of a slice in the video frame is computed by

$$\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[(\text{pixel-value}_{i,j})_{\hat{f}(t)} - (\text{pixel-value}_{i,j})_{\tilde{f}(t)}\right]^2.$$

The loss of a slice in a reference frame can also introduce error propagation in the current and subsequent frames until the end of GOP. The CMSE contributed by the

loss of the slice is thus computed as the sum of MSE over all frames in the GOP.

2.5 Video-On-Demand Broadcasting

In *video-on-demand* (VOD) broadcasting systems, different clients can choose a video from a list of videos at different times and they all should be able to watch the video from the beginning after a short *delay*. Clearly, transmitting a different video stream for each client may not be feasible in practice due to huge bandwidth burdens and high imposed computational complexity. However, by *partitioning* the video stream into *segments* and concurrently transmitting all the segments, viewers may employ the same streams to watch the video at different time instances at the cost of a short startup delay.

These algorithms are called *periodic broadcasting* algorithms [36], which break a video into segments and broadcast each segment periodically in an individual channel. Major periodic broadcasting protocols can be grouped in three categories as follows:

1. Increasing segments size and equal transmission bandwidths (e.g, [37–41]),
2. Equal segments size and different transmission bandwidths (e.g, [42–44]), and
3. Unequal segments size and transmission bandwidths (e.g, [45–47]).

Previously, several contributions have employed FEC coding in VOD systems. Authors in [48] have proposed to employ EEP-rateless codes along existing periodic broadcasting protocols to implement an efficient and loss resilience VOD broadcasting protocol.

Author in [49] has proposed to employ Reed-Solomon codes and provide UEP for frames that are required earlier to provide URT. However, since a large amount of overhead is assigned to beginning frames, the proposed algorithm may perform sub-optimally. Further, as described in the previous section, Reed-Solomon coding/decoding has a higher complexity compared to rateless codes. Authors in [50]

have proposed to use *harmonic broadcasting* (which belong second category of periodic VOD broadcasting protocols) along with EEP-rateless coding. Authors have mostly concentrated on providing flexible video stream delivery in *mobile datacast* channels. Later, we will employ UEP-rateless codes and design a novel VOD algorithm, and show that it obtains a shorter startup delay compared to exiting algorithms.

2.6 Rate-Compatible Convolutional Codes

Convolutional codes [51] are one the most widely used fixed-rate codes. These codes receive a stream of bits and perform series of XOR operations using a *memory* for previously received bits to generate output bits. The original convolutional codes had fixed coding rate. Later, it was shown that by removing certain output bits from the generate encoded bits, which is referred to by *puncturing*, *low rate* convolutional mother code can achieve a wide range of coding rates [52]. These codes are called rate-compatible convolutional (RCPC) codes.

The RCPC decoder employs a *Viterbi* decoder. The bit error rate P_b of the Viterbi decoder is upper bounded by [52]

$$P_b \leq \frac{1}{P} \sum_{d=d_f}^{\infty} c_d P_d, \tag{2.10}$$

where d_f is the free distance of the convolutional code, P is the puncturing period, and c_d is the total number of error bits produced by the incorrect paths and is known as the *distance spectrum* [52]. Finally, P_d is the probability of selecting a wrong path in Viterbi decoding with Hamming distance d, which depends on the modulation and channel characteristics. For an RCPC code with rate R, using the AWGN channel, BPSK modulation and the symbol to noise power ratio $\frac{E_S}{N_0} = R\frac{E_b}{N_0}$, the value of P_d (using soft Viterbi decoding) is given by

$$P_d = \frac{1}{2}\text{erfc}\sqrt{d\frac{E_S}{N_0}} = Q\sqrt{2d\frac{E_S}{N_0}} \tag{2.11}$$

where $Q(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{\lambda}^{\infty} e^{-\frac{a^2}{2}} da$.

The decoding bit error rate of RCPC codes for a sample code with $R = \frac{1}{3}$ and memory $M = 6$ [52] is depicted in Figure 2.5.

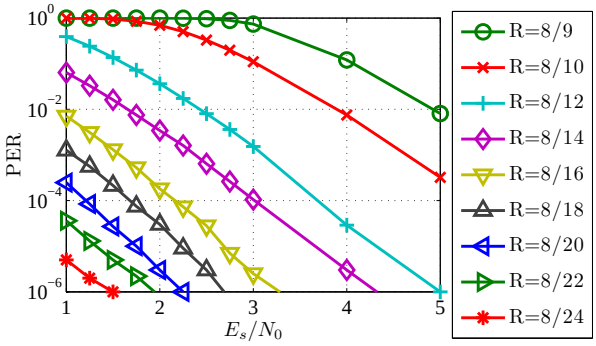


Figure 2.5 The decoding bit error rate of sample RCPC code from [52] versus $\frac{E_S}{N_0}$.

2.7 Compressive Sensing

Emerging *compressive sensing* (CS) techniques [53, 54] provide means to recover a *compressible* signal from its undersampled *random projections* also called *measurements*.

Let $\underline{\theta} = [\theta_1\theta_2 \dots \theta_N]^T$ be the transform of signal $\underline{x} = [x_1x_2 \dots x_N]^T$ in transform domain $\Psi \in \mathbb{R}^{N\times N}$, i.e., $\underline{x} = \Psi\underline{\theta}$. $\underline{x}$ is said to be compressible in Ψ if $\underline{\theta}$ has only K significant coefficients (the rest $N - K$ coefficients can be set to zero). Such a signal is referred to by K-sparse signal. Formally, the signal $\underline{x}$ is compressible if the magnitude of its sorted transform coefficients, i.e., $|\theta|_{(1)} \geq |\theta|_{(2)} \geq \dots \geq |\theta|_{(N)}$, decay faster than $C_1 i^{-\frac{1}{C_2}}$, where $0 < C_2 \leq 1$ and C_1 is a constant [53, 55–58]. A larger C_2 shows a higher compressibility.

The idea behind CS is that when $\underline{x}$ is K-sparse in Ψ, only $M \ll N$ ($M \geq O(K \log N)$) measurements $\underline{y} = [y_1y_2 \dots y_M]^T$ of $\underline{x}$ can reproduce an estimate $\underline{\hat{x}}$ using CS reconstruction with a comparable error to the best approximation error using K largest transform coefficients [53, 57, 58]. CS is composed of the two following key components.

Encoding: The measurements are generated by $\underline{y} = \Phi\underline{x}$, where Φ is a well-chosen $M \times N$ random matrix called *projection matrix*.

Decoding: Signal reconstruction can be performed by finding the estimate $\hat{\underline{\theta}}$ (and consequently $\hat{\underline{x}} = \Psi\hat{\underline{\theta}}$) via solving

$$\hat{\underline{\theta}} = \text{argmin}\|\underline{\theta}\|_1, \text{ s.t. } \underline{y} = \Phi\Psi\underline{\theta}, \tag{2.12}$$

where $\|\underline{\theta}\|_1 = \sum_{i=1}^{N} |\theta_i|$. The problem (2.12) is an underdetermined system of equations and various techniques have been proposed to obtain $\underline{\theta}$ knowing that it is sparse. In this thesis, we employ the CS reconstruction algorithms based on linear-programming referred to by *basis pursuit* (BP) [59]. There are also numerous *iterative* reconstruction algorithms [60, 61] that offer a lower reconstruction complexity at the cost of lower reconstruction accuracy.

Initially studied Φ's, were *dense* matrices where entries of Φ were randomly selected from $\{-1, +1\}$ or Gaussian distribution [53, 57]. Later, it was shown that when Ψ is *dense* and *orthonormal*, e.g., Fourier transform basis, a sparse Φ with *at least one non-zero* placed independently and randomly per row satisfies CS requirements on Φ [55, 58]. Later, we employ this interesting property of random Φ matrices in a WSN to reduce the total number of transmissions.

The selection of Ψ depends on the nature of the signal. For instance, temperature signals are shown to be sparse in *discrete cosine transform* (DCT) basis [56]. Therefore, without loss of generality in the rest of this dissertation we assume that Ψ is the DCT transform basis, while we could chosen have any other dense and orthonormal basis.

2.8 Probabilistic Broadcasting

Applications of WSNs is becoming more prevalent as their deployment cost decreases and network nodes provide new functionalities. WSNs are formed by tens or thou-

sands of power constrained nodes. Therefore, the nodes are unreliable and prone to failure, which may result in loss of data and topology changes over time. Therefore, one of the most important design criteria for data dissemination or data collection algorithms is their ability to perform the desired task with the minimum number of transmissions. Further, WSN nodes usually have limited computational power; hence, the data processing load also needs to be minimized on nodes.

When the WSN is small, actual or relative nodes location can be found to form efficient routing tables. However, these tables need to be updated on regular basis due to possible topology changes. However, in large scale wireless sensor networks such routing tables cannot be obtained and maintained since forming and keeping these tables up-to-date imposes a huge number of transmissions. Therefore, in such networks *stateless* routing tables are employed, which may only need local neighbors information. For instance, in data dissemination using *random walks*, a piece of data is forwarded node by node. Each node selects its next neighbor randomly or based on their number of neighbors [62]. After many steps, the data will travel required number of nodes. We can see that such a routing algorithm does not need the overall nodes location information and nodes may perform the routing independently. Therefore, random walks is scalable and flexible to network changes. Simple Flooding [63] is also one of the initial stateless algorithms, where all nodes unconditionally *broadcast* any piece of information they receive for the first time. The application of Flooding is limited in WSN since Flooding is known to cause broadcast storm problem [64]. Later, variations of Flooding were proposed to reduce the number of transmission by avoiding redundant transmissions.

Probabilistic broadcasting (PB) is a form of flooding where nodes perform the broadcasting with a certain *forwarding probability* p. It has been shown that PB can greatly reduce the number of transmissions compared to flooding while almost all nodes still receive the transmitted data [65]. Consider a WSN with N nodes with

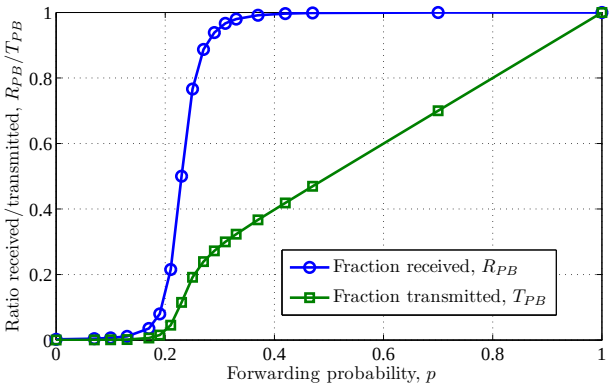


Figure 2.6 The fraction of nodes receiving a transmission $R_{PB}(p)$ and fraction of nodes the perform the transmissions $T_{PB}(p)$ versus forwarding probability p in PB.

identical transmission range of r_t deployed uniformly and randomly in an area A. Such a network is asymptotically connected with probability one for

$$\frac{\pi r_t^2}{A} = \frac{\ln n + \omega(n)}{n}, \tag{2.13}$$

if and only if $\omega(n) \to \infty$.

Assume node i, n_i, has a piece of information x_i that needs to disseminate in the network. Similar to Flooding n_i broadcasts x_i. Every node in the network that receives x_i for the *first time* rebroadcasts x_i with probability p. The fraction of nodes that receive a particular transmission $R_{PB}(p)$ and the fraction of nodes that perform the transmission $T_{PB}(p)$ are depicted in Figure 2.6 for $N = 10^4$ and $r_t = 0.025$.

Figure 2.6 shows that at $p \approx 0.24$ a large fraction of nodes receive x_i. Moreover, it has been shown that although increasing p beyond $p \approx 0.24$ does not improve the delivery of the reading, it considerably increases the number of transmissions. Therefore, a well-chosen small forwarding probability $p^* = 0.24$ would be sufficient to ensure that a large fraction of nodes in a network has received a transmission. It has been shown that p^* is close to the probability that a *giant component* appears in the network p^G, where asymptotically $p^G \approx \frac{1.44}{Nr_t^2}$ [66, 67]. For our given network topology with $N = 10^4$ and $r_t = 0.025$ we have $p^G = 0.23$. Therefore, p^* can be approximated

with p^G when N is large enough.

Later, we employ PB along with CS to implement an efficient data *storage algorithm* for WSNs. In our proposed algorithm, the total number of transmission is reduced compared to existing algorithms. Further, by employing CS the burden of computational complexity is transferred to the data collector rather than network nodes, which can further improve the life time of a WSN.

2.9 Delay Tolerant Networks

Delay tolerant networks (DTNs), also called *disruptive tolerant networks*, are networks with N *intermittently* connected mobile nodes, where no routes exist between a source and destination the receiver at any given time. Therefore, existing DTN routing algorithms employ *carry-and-forward* packet forwarding schemes to deliver packets to a destination. In carry-and-forward algorithms, when a node comes into the vicinity of the source it receives one packet. The packet is stored in the node's buffer and carried around in the network area until the node opportunistically meets the receiver and delivers the packet.

To cope with the packet loss (due to nodes' buffer overflow, nodes life-time, lack of contact with the receiver, etc.) contributions [68–71] have proposed to encode k source symbols employing FEC codes and transmit the encoded packets instead. Contributions [70, 71] in particular have proposed to employ the rateless codes as an efficient and flexible FEC coding schemes. Later, we employed out designed rateless codes with high ISRR in a DTN and compare its performance with [70].

Another important factor in DTNs is the nodes mobility pattern, which reflects nodes movements' characteristic. Mobility patterns can be divided into two groups. The first group of mobility patterns [72–80] is obtained from *real life networks* such as city busses network, walking people network, etc. Because of the inflexibility of real-life mobility patterns, *mobility models* [70, 81–87] have been proposed and are

employed instead of real-life mobility patterns in theoretical simulations (see [88–90] for more detailed reviews).

Existing DTN routing protocols can be divided into three groups based on the way they exploit nodes' meeting history. The packet forwarding algorithms in the first group [68, 91–93] do not use any meeting history, and they are suitable for large-scale and random DTNs where no history can be collected. The second group of protocols [76, 85] requires comprehensive information about the network such as complete nodes meeting history, buffers status, etc., thus they may not be applicable in all networks. The third group of algorithms [86, 94–97] requires limited information from the network such as nodes meeting history; hence their implementation is more prevalent and divers. We refer interested readers to [93, 98] for more detailed studies on DTN routing protocols.

2.10 Single Objective Genetic Algorithms Optimization

John Holland's in [99] shows how the evolutionary process can be applied to solve a wide variety of problems using a parallel technique that is now called the genetic algorithms [100]. Non-linear and complicated optimization problems which cannot be solved employing conventional optimization algorithms such as *linear programming* can be effectively solved using genetic algorithms. Let W and $\overline{w} = \{w_1, w_2, \ldots, w_k\}$ denote the decision space and k decision variables, respectively. Let $F(\overline{w})$ denote the objective function that we need to optimize (minimize/maximize). In conventional genetic algorithm methods, each w_i was translated to a binary format. The steps to find the optimum answer are as follows.

1. Generate a random initial population of size i each including k members $\overline{w_j}, j = \{1, 2, \ldots, k\}$.

2. Translate the generated population from real numbers to binary format consid-

ering desired precision.

3. Concatenate the translated version of k decision variables together to generate i binary population members.

4. Evaluate i fitnesses $F(\overline{w_j}, j \in \{1, 2, \ldots, i\})$ of the current population.

5. Select two parents randomly, assigning higher probability of selection to the parents with a better fitness value.

6. Perform *crossover* and *mutation* [99] on the parents to generate two offsprings. For crossover, cut two parents from a random location and exchange second parts to generate offsprings. For mutation, with a small probability flip a random bit in the offsprings' bit streams.

7. Go to 5 until $i - 2$ offsprings are generated.

8. Keep two parents with the best fitnesses and replace the rest $i-2$ with the new offsprings.

9. If maximum iterations is not reached go to 4, otherwise translate the member of population with the best fitness from binary to real format and report it as the final answer.

The above algorithm is an overall view of conventional genetic algorithms. However, many variations have been proposed since genetic algorithms were first introduced. For instance, the translation from real to binary and vice-versa is no more performed and the algorithm and the crossover and mutation are performed all in real numbers. More detailed explanation of genetic algorithms is out of the scope of this dissertation. We refer the interested readers for performance evaluations of genetic algorithm methods to [100] and the numerous available surveys on genetic algorithms.

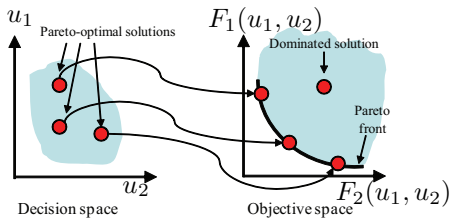


Figure 2.7 Pareto optimality, pareto front, and domination for a two-objective minimization problem with two decision variables, u_1 and u_2.

2.11 Multi Objective Genetic Algorithms Optimization

Multi objective genetic algorithms are basically different from single objective version. Let U and $\underline{u}$ denote the decision space and a decision vector, respectively, of an optimization problem. Let $F_1(\underline{u}), F_2(\underline{u}), \ldots, F_n(\underline{u})$ denote the conflicting objective functions. The problem is to find decision vectors $\underline{u}$ that *concurrently* minimize/maximize *all* objective functions. In a simple case with a single objective function, the problem boils down to a conventional minimization/maximization problem.

For a minimization problem, $\underline{u}_1 \in U$ is said to be dominated by $\underline{u}_2 \in U$, or $\underline{u}_1 \prec \underline{u}_2$, if $\forall\ i \in \{1, \ldots, n\},\ F_i(\underline{u}_1) \geq F_i(\underline{u}_2)$ and for at least one i, $F_i(\underline{u}_1) > F_i(\underline{u}_2)$. A non-dominated *pareto front* vector, $\underline{u}^*$, is a decision vector that no other decision vector can dominate. In other words, in a minimization problem no other decision vector exists such that it would decrease some objective functions without deteriorating at least one other objective function compared to $\underline{u}^*$.

The set of all dominant solution vectors form *pareto optimal set*. The plot of objective functions of pareto optimal members in the objective space builds the *pareto front*. In Figure 2.7, the pareto optimality and pareto front for a simple problem with two decision variables u_1 and u_2, and two-objective functions $F_1(u_1, u_2)$ and $F_2(u_1, u_2)$ is illustrated.

Figure 2.7 shows that no member can dominate pareto front members, and pareto front members do not dominate each other. Multi-objective optimization methods

search to find decision variables that result in pareto front members that are *well spread* and *equally spaced* to cover the whole pareto front. NSGA-II [101] is one of the many such algorithms with an outstanding performance that we employ in our design. Note that although genetic-algorithms have very high complexity, the optimization can be performed in an off-line mode and stored and the appropriate codes can be later selected based on the system requirements.

CHAPTER 3

ON THE INTERMEDIATE SYMBOL RECOVERY RATE OF RATELESS CODES

In this section, we investigate the performance of rateless codes in *intermediate range* when the number of received output symbols is less than the required number for full decoding. We design codes for various problem setups.

3.1 Introduction

Although traditional rateless codes are capacity-achieving, in intermediate range, $0 \leq \gamma \leq \gamma_{succ}$, where the number of received output symbols is less than the minimum required for full decoding of k input symbols, i.e. $\gamma_{succ}k$, few input symbols can be decoded. because most of the received output symbols are *buffered* for a later decoding [4, 102–104]. Therefore, designing new rateless codes with high *intermediate symbol recovery rates* (ISRR) is of interest for many applications such as multimedia transmission.

It has been shown that the intermediate range has three regions and the optimal rateless codes at each region have different characteristics. To design rateless codes with high ISRR, we select one overhead from each region and design rateless codes that concurrently have almost optimal ISRR at these three overheads employing *multi-objective genetic algorithms* assuming ε is not known to the source [5, 6].

In the next step, we assume that an estimate of ε is available at the source and propose *rateless coded symbol sorting* (RCSS), which further improves the ISRR of the codes we design in the first step. RCSS employs the history of the previously

transmitted output symbols and their dependencies for decoding to *reorder* their transmission such that each transmitted symbol has the highest probability of decoding an input symbol at decoder (if correctly delivered) among the remaining ones. Next, we discuss the advantages and capabilities of RCSS [6, 7].

Let $z \in [0, 1]$ denote the fraction of decoded input symbols at a decoder; hence, due to low ISRR of rateless codes we have $z \approx 0$ in $0 \leq \gamma \leq 1$. In [4], author shows that the intermediate range of rateless codes can be divided into three regions. The three intermediate regions for $0 \leq z < 1$ are $z \in [0, \frac{1}{2}]$, $z \in [\frac{1}{2}, \frac{2}{3}]$, and $z \in (\frac{2}{3}, 1)$, which approximately give the equivalent regions of $\gamma \in [0, 0.693]$, $\gamma \in [0.693, 0.824]$, and $\gamma \in [0.824, 1]$. Further, author designs optimal degree distributions that achieve the upper bound on ISRR of all rateless codes in these regions. However, the codes designed in [4] are asymptotically optimal and may not be employed when k is finite. Further, the proposed degree distributions are only optimal in one intermediate region.

In [102, 104] authors propose to employ feedbacks from the receiver to keep the encoder aware of z. They propose to gradually increase the degree of output symbols such that the instantaneous recovery probability of each arriving output symbol is maximized. The codes designed in [102, 104] require feedbacks, hence their application is not always feasible.

Authors in [103] propose to transmit output symbols in the order of their *ascending degree*. Although this would increase the ISRR, we will see that RCSS always outperforms this technique.

In this chapter, We first employ *multi-objective genetic algorithms* to design degree distributions that have *almost* optimal ISRR *throughout* $0 \leq \gamma \leq 1$. We employ the term "almost optimal" because genetic algorithms are known to find solutions that are not necessarily global-optimum but are rather very close to the global-optimum solution. Therefore, throughout our code design process the term optimal implies almost optimal. In the next step, we assume that an estimate of the channel erasure

rate $\varepsilon \in [0, 1)$ is available at the encoder and propose *rateless coded symbol sorting* (RCSS), which rearranges the transmission order of output symbols to further improve the ISRR.

3.2 Rateless Code Design with High ISRR

In this section, we design degree distributions for rateless coding with various k's employing multi-objective genetic algorithms.

3.2.1 Intermediate Overhead Selection and Optimization

To obtain high ISRRs in all three intermediate regions, we need to tune the degree distribution $\Omega(x)$ considering all three intermediate regions of $0 \leq \gamma \leq 1$. We choose three overheads $\gamma = 0.5$, $\gamma = 0.75$, and $\gamma = 1$ (one from each intermediate region) and define the respective value of z at these γ's as our objective functions. Let $z_{0.5,\Omega(x)}$, $z_{0.75,\Omega(x)}$, and $z_{1,\Omega(x)}$ denote the value of z at three selected γ's representing three objective functions that we aim to *concurrently maximize* and realize a high ISRR. With this setup, we have three *conflicting* objective functions meaning that improving z at one point may decrease z at one or both other γ's. As a result, we employ *multi-objective optimization methods* to design our desired distributions.

Clearly, in our optimization problem the decision variables are entries of $\Omega(x)$. Codes that are designed to realize a high ISRR have $\Omega(x)$'s with much smaller *maximum* degree compared to codes designed for full input symbol recovery [1, 2, 22]. For instance, codes that optimally perform in the first and second intermediate regions have maximum degrees of only 1 and 2 [4], respectively. Consequently, we consider degree distributions with maximum degree of 50. Thus, we have fifty *decision variables* $\{\Omega_1, \Omega_2, \ldots, \Omega_{50}\}$ that take values in $[0, 1]$ such that $\sum_{i=1}^{50} \Omega_i = 1$. Later, we see that the optimum $\Omega(x)$'s have much smaller maximum degree than 50.

We need to take different approaches to find $z_{\gamma,\Omega(x)}$ for asymptotic and finite length

setups. For asymptotic case, the expression providing the rateless decoding error rate is given by (2.5). On the other hand, the expression for the error rate of rateless decoding for finite k has been analyzed in [105, 106]. However, the high complexity of these expressions makes their application in genetic-algorithm implementation almost impossible. Therefore, to find z for finite k we employ Monte-Carlo method by averaging z for a large enough number of decoding simulation experiments for $k \in \{10^2, 10^3, 10^4\}$. Similar to asymptotic case, our objective functions are $z_{0.5,\Omega(x)}$, $z_{0.75,\Omega(x)}$, and $z_{1,\Omega(x)}$, which in this case are found by numerical simulations.

3.2.2 Optimized Rateless Codes for High ISRR

We employ NSGA-II multi-objective optimization algorithm [101] to find the distributions that have optimal z at three selected γ's (see Section 2.11 and refer to [101] for more information on NSGA-II). The results of our optimizations are four *databases* of degree distributions optimized for $k \in \{10^2, 10^3, 10^4, \infty\}$. Due to huge size of the four databases they may not be reported in the dissertation and are made available online at [107]. In the next section, we investigate the performance of several designed distributions.

3.2.3 Performance Evaluation of the Designed Codes

Based on the desired ISRR at each intermediate region an appropriate $\Omega(x)$ needs to be selected among the *many* optimum degree distributions in our databases. To facilitate the distribution selection from our databases we propose a *weighted* function $F(\Omega(x))$ defined by

$$F(\Omega(x)) = W_{0.5}[Z_{0.5} - z_{0.5,\Omega(x)}] + W_{0.75}[Z_{0.75} - z_{0.75,\Omega(x)}] + W_1[Z_1 - z_{1,,\Omega(x)}], \quad (3.1)$$

where Z_γ is the highest possible z (upper bound on z) at γ for all rateless codes and W_γ is a *tunable weight*. From [4], we have $Z_{0.5} = 0.3934$, $Z_{0.75} = 0.5828$ and $Z_1 = 1$.

For future references, we define $\mathbf{W} = (W_{0.5}, W_{0.75}, W_1)$. We can find $\Omega(x)$ of interest by setting the appropriate weights and selecting the $\Omega(x)$ that *minimizes* $F(\Omega(x))$. However, we emphasize that one may replace (3.1) with any desired linear or non-linear weighted function. Table 3.1 shows the optimum degree distributions for the selected arbitrary weights. Note that the degree distributions reported in Table 3.1 are *only samples* of many degree distributions we have made available at [107].

Table 3.1 Optimum degree distributions for different weights $\mathbf{W} = (W_{0.5}, W_{0.75}, W_1)$.

k	$\mathbf{W}$	Optimum degree distribution $\Omega(y)$
10^2	$(1,1,1)$	$0.348y + 0.652y^2$
	$(0,1,0)$	$0.1911y + 0.8082y^2 + 0.0003y^4$
	$(0,0,1)$	$0.116y + 0.467y^2 + 0.417y^3$
	$(1,4,1)$	$0.346y + 0.652y^2$
	$(1,1,4)$	$0.1515y + 0.7903y^2 + 0.0581y^3$
10^3	$(1,1,1)$	$0.3131y + 0.6869y^2$
	$(0,1,0)$	$0.0139y + 0.9861y^2$
	$(0,0,1)$	$0.0624y + 0.5407y^2 + 0.2232y^4 + 0.1737y^5$
	$(1,4,1)$	$0.1448y + 0.8552y^2$
	$(1,1,4)$	$0.0624y + 0.9315y^2$
10^4	$(1,1,1)$	$0.2474y + 0.7526y^2$
	$(0,1,0)$	$0.011y + 0.989y^2$
	$(0,0,1)$	$0.0312y + 0.4069y^2 + 0.3716y^3 + 0.0024y^6$ $+0.0264y^7 + 0.1519y^{10} + 0.0096y^{14}$
	$(1,4,1)$	$0.1452y + 0.8548y^2$
	$(1,1,4)$	$0.16y + 0.3524y^2 + 0.1318y^3 + 0.3553y^5$ $+0.0001y^7 + 0.0003y^{10} + 0.0001y^{14}$
∞	$(1,1,1)$	$0.29599y + 0.70401y^2$
	$(0,1,0)$	$0.00003y + 0.99997y^2$
	$(0,0,1)$	$0.00536y + 0.50088y^2 + 0.12547y^3$ $+0.17492y^4 + 0.03797y^5 + 0.00583y^6$ $+0.00011y^7 + 0.00013y^8 + 0.00001y^{10}$ $+0.00209y^{11} + 0.06425y^{13} + 0.08297y^{14}$
	$(1,4,1)$	$0.12469y + 0.87531y^2$
	$(1,1,4)$	$0.11003y + 0.24932y^2 + 0.34144y^3$ $+0.14488y^4 + 0.02164y^5 + 0.00123y^6$ $+0.00014y^{11} + 0.05257y^{13} + 0.07862y^{14}$ $+0.00012y^{17}$
All	$(1,0,0)$	y
	$(4,1,1)$	

One may choose an optimal distribution based the desired weights from the databases provided at [107]. From Table 3.1, we can see that the optimal degree distributions for finite length slightly differ from the distributions proposed in [4]. For instance for $\mathbf{W} = (0, 1, 0)$ the distribution is non-zero for Ω_1, which allows the rateless decoding to start. Moreover, from our databases we observe that the maximum degree of all designed degree distributions is 19, which is much smaller than 50. Further, we can see that as k decreases, large degrees are also eliminated. We compare the performance of our designed degree distributions with the upper bound found in [4] in Figures 3.1 and 3.2.

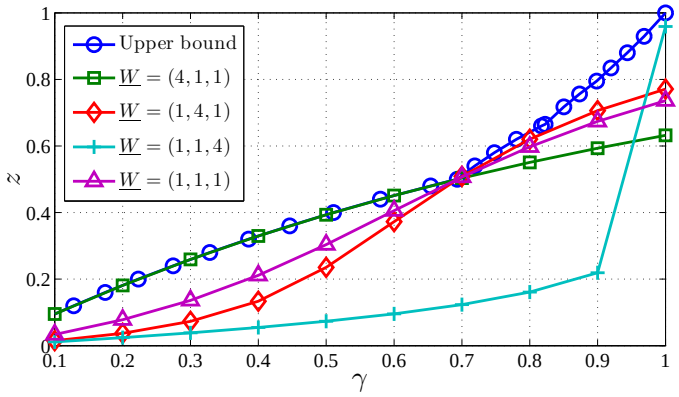


Figure 3.1 ISRR of selected designed codes and the ISRR upper bound for asymptotic setup.

The ISRR of the codes designed for $\mathbf{W} = (1, 1, 1)$ as shown in Figures 3.1 and 3.2 are optimal at three selected γ's. In other words, there is *no other degree distribution* that can go closer to the upper bound at *one* γ *without decreasing* z for at least one other γ compared to our designed degree distributions. Moreover, from Figures 3.1 and 3.2 we can see that by setting the desired weights the selected distribution performs better at the region with the higher weight. Further, we can see that as k increases the difference of ISRR with the upper bound decreases because the upper bound is derived for asymptotic setup. In the next section, we show how ISRR of our designed codes may be increased even more.

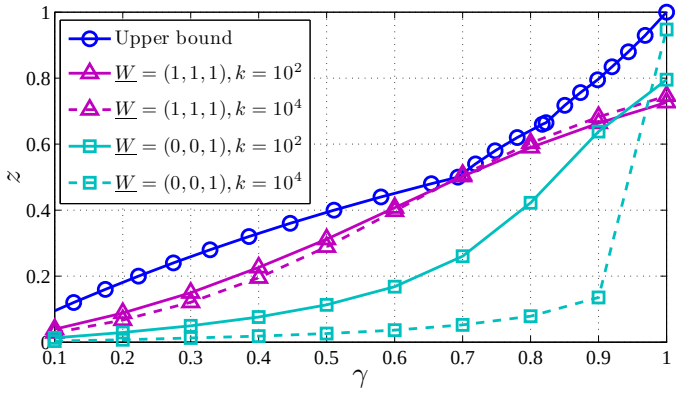


Figure 3.2 ISRR of selected designed codes and the ISRR upper bound for $k = 10^2$ and $k = 10^4$.

3.3 *RCSS*: Rateless Coded Symbol Sorting

In practice an estimate of the channel erasure rate ε may be available at the encoder [108]. The value of ε may be exploited as a side information to further improve the ISRR of rateless codes.

3.3.1 RCSS: Rateless Symbol Sorting Algorithm

When the encoder has an estimate of ε, it is aware that in total $m = \frac{k\gamma_{succ}}{1-\varepsilon}$ output symbols should be transmitted so that the receiver obtains $k\gamma_{succ}$ output symbols. The main idea in designing RCSS is that the encoder can generate m output symbols ahead of transmission. Therefore, it can *rearrange* the order of m output symbols such that each delivered symbol has the highest probability of decoding an input symbol at the receiver. This results in a considerable improvement of ISRR since fewer output symbols are buffered for a later decoding at the receiver. We should note that RCSS is merely implemented at the encoder and the decoder remains intact. Therefore, in contrast to [102, 103] we assume that the receiver generates no feedback and RCSS can only employ the information available at the encoder.

The reordering of m output symbols in RCSS is performed as follows. The encoder

maintains a probability vector $\boldsymbol{\rho} = [\rho_1 \rho_2 \dots \rho_k]$, in which ρ_j represents the probability that x_j is still *not recovered at* the receiver. Clearly, the encoder initializes $\boldsymbol{\rho}$ to an all-one vector when the transmission has not started yet. At each transmission the encoder finds an output symbol c_i that has the highest probability of recovering an input symbol at the receiver based on $\boldsymbol{\rho}$ (as described later). Next, the encoder transmits c_i and updates $\rho_j, j \in \mathcal{N}(c_i)$, where $\mathcal{N}(c_i) \subset \{1, 2, \dots, k\}$ is a set containing index of input symbols that are neighboring to c_i. The encoder continues until all m output symbols are transmitted.

From the rateless decoding procedure, we can see that an output symbol c_i with degree d, i.e., $|\mathcal{N}(c_i)| = d$, where $|.|$ represents the cardinality of a set, can recover an input symbol x_j iff all $x_w, w \in \{\mathcal{N}(c_i) - j\}$ have already been recovered. Let $p_i^{dec}, i \in \{1, 2, \dots, m\}$ denote the probability that c_i can recover an input symbol at the receiver.

Since at the beginning of transmission no input symbol is still recovered, we have $p_i^{dec} = 0$ if $|\mathcal{N}(c_i)| > 1$, i.e., output symbols with degrees larger than one cannot decode any input symbol at the receiver. Besides, for $|\mathcal{N}(c_i)| = 1$ we have $p_i^{dec} = (1 - \varepsilon)$, i.e. only degree-one output symbols that are not erased on the channel (with probability $1 - \varepsilon$) can recover an input symbol. Therefore, at the beginning of transmission degree-one output symbols have the highest probability of decoding an input symbol at the receiver. Consequently, the encoder transmits degree-one c_i's with $\mathcal{N}(c_i) = \{j\}$ and updates $\rho_j = \varepsilon \rho_{j,old}$, where $\rho_{j,old}$ is the value of ρ_j before c_i was transmitted.

Next, we consider a degree-two output symbol c_i with $\mathcal{N}(c_i) = \{j, l\}$. In this case, c_i can recover x_j with probability $(1 - \varepsilon)(1 - \rho_l)\rho_j$, which is the probability that c_i is not dropped on channel, x_j has not been recovered previously, and x_l has already been recovered. Similarly, c_i can recover x_l with probability $(1 - \varepsilon)(1 - \rho_j)\rho_l$. Consequently, $p_i^{dec} = (1 - \varepsilon)[(1 - \rho_l)\rho_j + (1 - \rho_j)\rho_l]$. Assume $\forall w \neq i, p_i^{dec} > p_w^{dec}$, i.e. c_i has the highest probability of decoding an input symbol at the receiver among

the remaining output symbols. Therefore, the encoder transmits c_i next and sets $\rho_j = \rho_{j,old}(1-(1-\varepsilon)(1-\rho_{l,old}))$ and $\rho_l = \rho_{l,old}(1-(1-\varepsilon)(1-\rho_{j,old}))$.

Further, we consider an output symbol c_i with $|\mathcal{N}(c_i)| = d$. Such a c_i can decode an $x_j, j \in |\mathcal{N}(c_i)|$ with probability $(1-\varepsilon)\rho_j \prod_{v\in\mathcal{N}(c_i),v\neq j}(1-\rho_v)$. Therefore, $p_i^{dec} = (1-\varepsilon)\sum_{l\in\mathcal{N}(c_i)}[\rho_l \prod_{v\in\mathcal{N}(c_i),v\neq l}(1-\rho_v)]$. If $\forall w \neq i, p_i^{dec} > p_w^{dec}$, the encoder transmits c_i and updates $\rho_j = \rho_{j,old}[1-(1-\varepsilon)\prod_{v\in\mathcal{N}(c_i),v\neq j}(1-\rho_{v,old})], j \in \mathcal{N}(c_i)$. We summarize RCSS in Algorithm 1. The output of Algorithm 1 is a suitable rearranged transmission order $\boldsymbol{\pi}$ of output symbols that substantially improves ISRR.

Algorithm 1 RCSS: proposed output symbol sorting algorithm

Initialize: $\boldsymbol{\pi} = [\,]$, $\boldsymbol{\rho} = [1]_{1\times k}$

for counter = 1 to m **do**

for $j = 1$ to $m, j \notin \boldsymbol{\pi}$ **do**

$p_j^{dec} = (1-\varepsilon)\sum_{l\in\mathcal{N}(c_i)}[\rho_l \prod_{v\in\mathcal{N}(c_i),v\neq l}(1-\rho_v)]$

end for

$i^* = \underset{i}{\operatorname{argmax}}(p_i^{dec})$

$\boldsymbol{\pi} = [i^*, \boldsymbol{\pi}]$

for $j \in \mathcal{N}(c_{i^*})$ **do**

$\rho_j = \rho_{j,old}[1-(1-\varepsilon)\prod_{v\in\mathcal{N}(c_{i^*}),v\neq j}(1-\rho_{v,old})]$

end for

end for

Suppose two (or more) output symbols c_j and c_l have equal probability of decoding of an input symbol, i.e., $p_j^{dec} = p_l^{dec}$. In addition, assume this probability is the largest probability of decoding an input symbol compared to that of other remaining output symbols. In this case, $\underset{i}{\operatorname{argmax}}(p_i^{dec})$ returns the index of c_j or c_l whichever has a *lower degree*.

3.3.2 RCSS Lower and Upper Performance Bounds

We investigate the upper and the lower bounds on the performance of RCSS in the following lemmas.

Lemma 3.1 *The performance of RCSS is upper bounded by $z = \gamma$ for $\varepsilon \to 0$.*

Proof. Clearly, we have

$$\lim_{\varepsilon \to 0} p_i^{dec} = \lim_{\varepsilon \to 0} (1-\varepsilon) \sum_{l \in \mathcal{N}(c_i)} [\rho_l \prod_{v \in \mathcal{N}(c_i), v \neq l} (1-\rho_v)] \in \{0,1\}, \forall i. \tag{3.2}$$

This means that since each packet is delivered with high probability the recovery of input symbols is no longer *probabilistic*. Therefore, we have

$$\lim_{\varepsilon \to 0} \rho_j = \lim_{\varepsilon \to 0} \rho_{j,old} [1 - (1-\varepsilon) \prod_{v \in \mathcal{N}(c_i), v \neq j} (1-\rho_{v,old})] \in \{0,1\}, \forall j, \tag{3.3}$$

showing that the recovery of each input symbol is similarly deterministic and is exactly known to the encoder. Therefore, the encoder can determine which output symbols can decode an input symbol with probability 1 in the next step. Consequently, as long as output symbols c_i with $p_i^{dec} = 1$ are available $z = \gamma$ is obtained. However, since the codes that we designed in Section 3.2 may not be capacity-achieving $z = \gamma$ is not necessarily realized. Therefore, the performance of RCSS is indeed upper bounded by $z = \gamma$. ■

We note that if the employed distribution is capacity achieving, i.e., $\gamma_{succ} = 1$, $z = \gamma$ can be obtained.

Lemma 3.2 *The performance of RCSS is lower bounded by the performance of [103] (where symbols are only sorted based on their degree) for $\varepsilon \to 1$.*

Proof. We have $\lim_{\varepsilon \to 1} p_i^{dec} = 0, \forall i$. Further, since initially we set $\boldsymbol{\rho} = [1]_{1\times k}$ then $\lim_{\varepsilon \to 1} \rho_j = 1, \forall j$. In other words, the encoder cannot make a meaningful estimate about the recovery of input symbols at the receiver. Since for $p_i^{dec} = p, \forall i$, $\underset{i}{\text{argmax}}(p_i^{dec})$ returns the index of output symbols with the lowest degree, for $\varepsilon \to 1$ Algorithm 1 boils down to an algorithm that only sorts output symbols based on their degree similar to [103]. Therefore, the performance of RCSS is lower bounded by the performance of the scheme proposed in [103]. ■

3.3.3 Complexity and Delay Incurred by RCSS

It is worth noting that in RCSS all output symbols need to be generated and sorted before the transmission can start in contrast to the *conventional* rateless coding where each c_i can be independently transmitted upon generation. This would result in some delays in transmission when RCSS is employed. However, this delay can be easily eliminated with the following procedure.

Clearly, the order of sorted output symbols is independent of the contents of input symbols and only depends on $\mathcal{N}(c_i), i \in \{1, 2, \ldots, m\}$. Therefore, before the transmission starts, the encoder generates c_i's from a *dummy* $\mathbf{x}$ and obtains and saves an *off-line* version of $\boldsymbol{\pi}_{\text{off-line}}$ and $\mathcal{N}_{\text{off-line}}(c_i)$. When the actual encoding starts, $\mathbf{x}$ of interest replaces the dummy $\mathbf{x}$, and the encoder generates c_i, $i \in \boldsymbol{\pi}_{\text{off-line}}$ by XORing $x_j, j \in \mathcal{N}_{\text{off-line}}(c_i)$. In this way, each c_i can be transmitted upon generation and no delay occurs. However, we need to note that the described procedure to eliminate the delay increases the memory requirements and necessitates data storage in contrast to conventional setup.

In addition, when RCSS is employed the overall complexity of rateless coding increases from $O(k)$ [2] in conventional rateless coding to $O(k^2)$ since Algorithm 1 has the complexity of $O(m^2) = O(k^2)$.

3.3.4 Performance Evaluation of RCSS

We implement RCSS for the rateless codes we designed for $\mathbf{W} = (0, 0, 1)$ with $k = 10^2$ with the distribution $\Omega_1(y) = 0.116y + 0.467y^2 + 0.417y^3$ and plot its ISRR along with its upper and lower bounds (for $\varepsilon \to 1$ and $\varepsilon = 0$, respectively) in Figure 3.3. Figure 3.3 shows that when an estimate of ε is available at the encoder, RCSS can substantially improve the ISRR of the codes designed in the Section 3.2. For instance at $\gamma = 0.5$ for $\varepsilon = 0.1$, we can see that z has increased from 0.1131 to 0.4003.

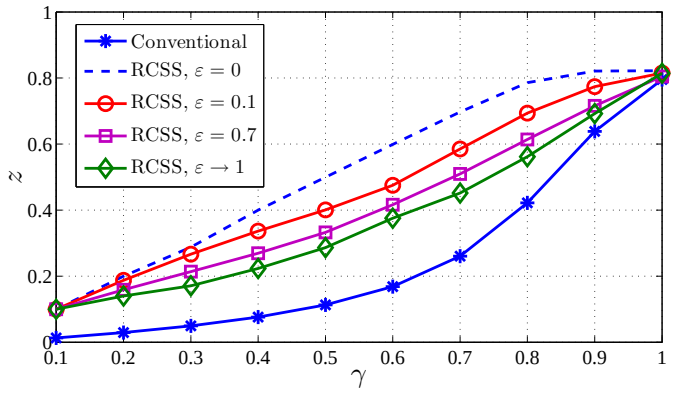


Figure 3.3 ISRR of codes designed for $k = 10^2$ with degree distribution $\Omega_1(x)$.

3.3.5 Employing RCSS with Capacity-Achieving Codes

Since RCSS only reorders the transmission of output symbols, it can be employed along with capacity-achieving rateless codes such as LT codes [1] while preserving their capacity-achieving property. We choose an LT code with parameters $c = 0.05$, $\delta = 0.01$, and $k = 10^3$ (c and δ are LT codes' distribution parameters [1]) and evaluate its ISRR improvement by RCSS in Figure 3.4. Figure 3.4 confirms that the ISRR of the employed LT code has considerably improved while its performance at $\gamma_{succ} = 1.4$ has remained intact.

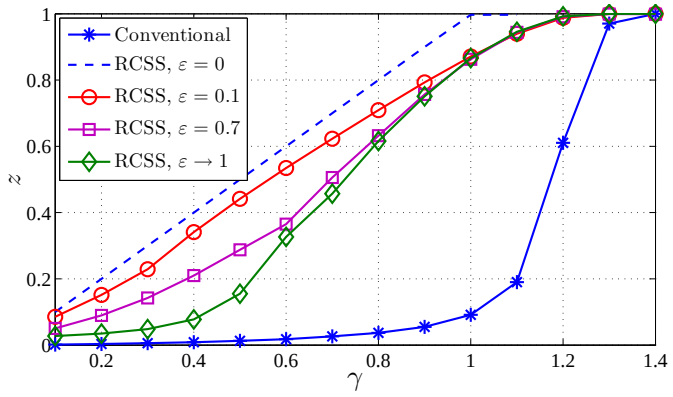


Figure 3.4 ISRR of LT codes employing RCSS, and the respective upper and lower bounds.

3.3.6 RCSS for Varying ε

Assume that the encoder has generated m output symbols considering ε and has sorted them employing RCSS. Further, assume that the erasure rate of the channel changes to ε_{new} when $\frac{k\gamma_c}{1-\varepsilon}$ symbols have already been transmitted and $\frac{k(\gamma_{succ}-\gamma_c)}{1-\varepsilon}$ output symbols are still remaining to be transmitted. If $\varepsilon_{new} > \varepsilon$, less than $k\gamma_{succ}$ output symbols would be collected by the receiver, making the full decoding impossible. In this case, the encoder generates $t = (\frac{1}{1-\varepsilon_{new}} - \frac{1}{1-\varepsilon})k(\gamma_{succ} - \gamma_c)$ *new* output symbols and adds them to the queue of output symbols to be transmitted to ensure the delivery of $k\gamma_{succ}$ output symbols to the receiver. Next, the encoder *rearranges* all output symbols employing RCSS and continues the transmission. On the other hand, if $\varepsilon_{new} < \varepsilon$ then the encoder randomly drops $1 - \frac{1-\varepsilon}{1-\varepsilon_{new}}$ fraction of remaining output symbols from the transmission queue. Further, if ε varies multiple times the same procedures are followed after each change.

Assume that the encoder has generated m output symbols employing distribution $\Omega_1(x)$ given in Section 5.3 for $\gamma_{succ} = 1$, $\varepsilon = 0.3$, and $k = 10^2$. Further, assume that ε increases to $\varepsilon_{new} = 0.5$ at $\gamma_c = 0.5$. Therefore, the encoder adds $t = \lceil 0.5714k(\gamma_{succ} - \gamma_c) \rceil$ new output symbols and runs RCSS again. The ISRR of this code has been shown in Figure 3.5 where the jump in ε_{new} occurs at $\gamma = \gamma_c = 0.5$. Figure 3.5 shows

that a large jump of 66.6% in the ε is well compensated by RCSS and the same z is achieved at $\gamma_{succ} = 1$. However, due to disturbance in the ordering caused by the newly added symbols a slight performance loss is observed.

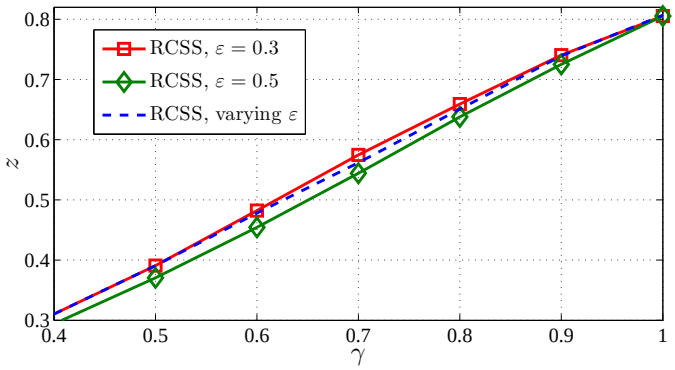


Figure 3.5 The resulting ISRR employing RCSS for the case where ε increases from 0.3 to 0.5 at $\gamma_c = 0.5$.

3.4 Application Example of Rateless Codes with High ISRR

As discussed earlier, in DTNs the delay in the delivery of data is usually very large, while the receiver in such a networks may benefit from *partial* recovery of input symbols from the incomplete set of output symbols. Previously, rateless codes have been employed to improve the overall system performance and data delivery flexibility [70, 71]. We take a step further, and show how rateless codes with high ISRR can improve the intermediate recovery of DTNs [8].

From the rateless decoding procedure, we can observe that decoding of output symbols with lower degrees depends on the recovery of a smaller subset of input symbols [103]. Therefore, at the beginning of the transmission where many input symbols are unrecovered, low degree output symbols have *higher* probability of decoding an input symbols at the receiver. Consequently, it is of interest to *deliver* encoded packets to *destination* in *ascending* order of output symbol *degree*.

We adopt a two-hop routing algorithm [109] from the encoder to the receiver. Assume all nodes including the encoder and the receiver have equal transmission range r_t, and the encoder and the receiver can communicate with nodes in their transmission range. Further, we assume all nodes, $n_i, i \in \{1, 2, \ldots, N\}$ have one buffer space to carry a single c_i. In addition, we assume that when a node comes into the transmission range of the encoder and the receiver the contact duration is long enough to transfer a *single* packet. Further, we assume network nodes are moving based on *localized random walk* [81] in our simulations. We deploy a network with the parameters given in table 3.2.

Table 3.2 DTN simulation parameters

Parameter name	Value
Network size	200×200 [distance unit]
r_t	15 [distance unit]
Nodes speed	randomly selected from $[0.1, 0.5]$ $\frac{\text{[distance unit]}}{\text{[time unit]}}$
Encoder and receiver locations	$(5, 5)$, $(195, 195)$

If the average aggregate packet loss of the network is ε, the encoder generates $m = k\gamma_S$ rateless coded c_i's employing degree distribution $\Omega_I(x) = 0.348x + 0.652x^2$ (which is optimized for $k = 100$ and $\mathbf{W} = (1, 1, 1)$) with $\gamma_S \geq \frac{\gamma_{succ}}{1-\varepsilon}$. Next, the encoder sorts m generated c_i's based on their *ascending* degrees. When a node n_i comes into the transmission range of the encoder, if it has an empty buffer and has previously met the receiver, the encoder dispatches an output symbol with lowest degree, and remove the output symbol from its buffer. While the encoder is giving out packets to incoming nodes, the receiver obtains packets from nodes coming to its transmission range with a full buffer.

We compare the ISRR when $\Omega_I(x)$ is used versus where distribution RS(x) with parameters $c = 0.05$ and $\delta = 0.01$ is employed [70] for $\gamma_S = 2.2$ in Figure 3.6.

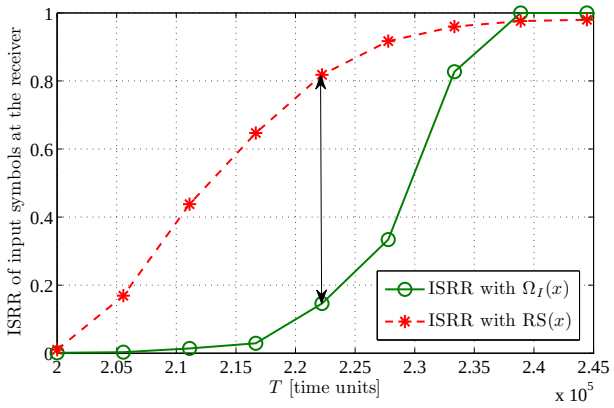


Figure 3.6 ISRR of input symbols at the receiver employing designed degree distribution $\Omega_I(x)$ and RS(x) versus T.

Figure 3.6 shows that $\Omega_I(x)$ results in a considerable improvement in ISRR compared to existing work. For instance at $T = 2.22 \times 10^5$ we can see that ISRR has improved from 0.146 for RS(x) to 0.818 for $\Omega_I(x)$. The receiver in a DTN may greatly benefit from this high ISRR. However, we can observe that since $\Omega_I(x)$ is not capacity approaching ISRR of $\Omega_I(x)$ never reaches 1 in contrast to ISRR of RS(x) distribution. Therefore, $\Omega_I(x)$ may be employed in applications where full recovery of input symbols is not necessary, such as multimedia content delivery.

3.5 Conclusion

Previously, it was shown that the *intermediate* range of rateless codes is comprised of *three* regions and for each region a rateless coding *distribution* that achieves optimal *intermediate symbol recovery rate* (ISRR) has been designed. In this chapter, we selected a point from each region and designed degree distributions that have optimal performance at *all* three selected points employing *multi-objective genetic algorithms*. Next, we assumed that an estimate of the channel erasure rate ε is available at encoder

and proposed *RCSS* that exploits ε and rearranges the transmission order of output symbols to *further* improve the ISRR of rateless codes. Finally, we employed one of the designed codes for data delivery in DTNs and showed that the ISRR can be greatly improved, which may be beneficial to the receiver.

CHAPTER 4

DISTRIBUTED UNEQUAL-ERROR-PROTECTION RATELESS CODES OVER ERASURE CHANNELS

In this chapter, we investigate and design rateless codes for *distributed* data collection.

4.1 Introduction

In *distributed rateless* coding, multiple disjoint sources need to deliver their *rateless* coded output symbols to a *common* destination via a single *relay*. For instance, r nodes within a *cluster* in a WSN that transmit their rateless coded data to a *base station* via a *cluster-head* form such a distributed data collection. Note that r data sources may have different data block lengths and different data *importance levels*.

Conventional LT codes did not target *distributed* data collection; hence, they may suboptimally perform in distributed data collection [110]. Consequently, we propose and design novel *distributed UEP-rateless* (DU-rateless) codes that can provide UEP for disjoint sources with *unequal* data lengths on erasure channels [9, 10].

To design DU-rateless codes, we tune the coding parameters at each source and propose to *smartly* combine the encoded symbols at the relay. We analyze DU-rateless codes employing And-Or tree analysis technique and leverage our analysis to design several sets of codes for various setups employing multi-objective genetic algorithms. We evaluate the performance of the designed codes using numerical simulations and discuss their advantages. As a first step in the design of DU-rateless codes, we consider $r = 2$ to design DU-rateless codes for erasure channels. We should note that DU-

rateless codes are inspired by UEP-rateless codes [19, 20].

In DU-rateless codes, the size of input symbols can be arbitrary from one-bit (binary) symbol to hundreds or thousands of bits similar to LT codes. The problem in DU-rateless codes is to tune a degree distribution for each source and to design relaying parameters to achieve (almost) minimal decoding error rates with a certain ratio referred to by *UEP gain*. Similar to LT codes, DU-rateless codes are also *universal* [1] meaning that they are simultaneously near optimal for every erasure channel. We employ And-Or tree analysis technique to study decoding of DU-rateless codes. Next, we utilize our analytical results to design *jointly* optimize DU-rateless codes parameters and obtain several *close to* optimal DU-rateless codes for various setups employing NSGA-II [101]. Finally, we report the designed codes and evaluate their performance. The comparable scheme to DU-rateless codes is employing an independent LT codes at each source.

Authors in [110] have designed *distributed LT* (DLT) codes. In DLT coding, Robust-Soliton distribution is decomposed into r identical distributions to encode input symbols at r sources. Next, the encoded symbols are selectively combined or forwarded with certain probabilities to the destination such that the delivered coded symbols follow Robust-Soliton degree distribution (which is known to be capacity-achieving).

Authors in [111], considered rateless coding at r sources with an identical degree distribution. In [111], the number of combined encoded symbols (regardless of their degree) at the relay is determined by a second independent degree distribution. Authors have analyzed their codes and designed a few distributed rateless codes. In [112] authors considered a network with two sources $r = 2$ and designed a simple forwarding from the relay such that the degree distribution of the delivered symbols to destination follows a *Soliton-like* distribution (SLRC). Authors have shown that SLRC codes outperform DLT codes. Further, SLRC codes reduce to LT codes when

a source leaves.

Authors in [108] propose an *online* encoding ensemble of LT codes such that the i^{th} output symbol is strictly comprised of the first i input symbols. They design their encoding and relaying scheme such that delivered symbols to destination maintain Robust-Soliton distribution. The scheme proposed in [108] may not be distributively implemented in contrast to DU-rateless codes. Authors in [19, 20], proposed *UEP* rateless codes. Although codes designed in [19, 20] are capable of providing UEP, they may not be distributively implemented.

4.2 Distributed Unequal-Error-Protection Rateless Codes

In this section, we describe DU-rateless coding/decoding.

4.2.1 Proposed Coding and Decoding

Consider a distributed data collection with two sources s_1 and s_2 with data block of lengths ρk and k *input* symbols, respectively, where $0 < \rho \leq 1$. Let S_1 and S_2 denote the set of s_1 and s_2 input symbols, respectively. In DU-rateless coding, s_1 employs $\Omega(x)$ to encode its data block S_1 (in the same way that input symbols are encoded by Robust-Soliton distribution in LT coding). Similarly, s_2 employs $\varphi(x)$ to encode S_2. Next, s_1 and s_2 transmit their output symbols to a common relay R, which based on the following two rules generates three types of output symbols and forwards them to a destination D.

1. With probabilities p_1 and p_2 it directly forwards s_1 and s_2's output symbols to D, respectively.

2. With probability $p_3 = 1 - p_1 - p_2$ it forwards the XOR of two incoming coded symbols to D.

The decoding process of LT and DU-rateless codes are identical and is performed iteratively as follows. Find an output symbol such that the value of all but one neighboring input symbol is known. Recover the value of the unknown input symbol by bitwise XOR operations. Repeat this process until no such an output symbol exists. As we later show, iterative decoding of rateless codes is a form of *belief propagation* decoding. The DU-rateless decoding succeeds with a high probability when $(1+\rho)\gamma_{succ}k$ output symbols are received at D. For a received coding overhead of $0 \leq \gamma \leq \gamma_{succ}$, the proposed DU-rateless code ensemble is specified by parameters $(\rho k, k, \Omega(x), \varphi(x), p_1, p_2, p_3, \gamma)$.

Let ε_1, ε_2, and ε_3 denote the erasure rates of $s_1 - R$, $s_2 - R$, and $R - D$ channels, respectively. Further, assume packet transmission at s_1 and s_2 is not synchronized. With this setup, we need to set the *symbol transmission rates* of s_1 and s_2 such that no huge symbol buffering or dropping is required at R. It is not hard to show that s_2 needs to generate $\frac{(1-p_1)(1-\varepsilon_2)}{(1-p_2)(1-\varepsilon_1)}$ output symbols per one output symbol generated at s_1 so that in expectation no symbols are buffered. We should note that due to random losses of s_1 and s_2 symbols and their asynchronous transmissions, R may need to buffer only a few symbols for a short period time. For example, assume R decides to combine s_1 and s_2 symbols. However, due to random losses on the channel several symbols from s_1 arrive while no symbols from s_2 arrives. In such a case, R needs to buffer a few symbols from s_1 until symbols from s_2 arrive. Therefore, the transmission rate of $\frac{(1-p_1)(1-\varepsilon_2)}{(1-p_2)(1-\varepsilon_1)}$ symbol at s_2 guarantees that R may have to buffer only a few symbols for a short period of time.

4.2.2 And-Or Tree Analysis of the Proposed Codes

To investigate the recovery probability of an input symbol in DU-rateless decoding on erasure channels, we extend the And-Or tree analysis [21, 22] technique described in Section 2.3 to fit the decoding process of DU-rateless codes. In DU-rateless coding,

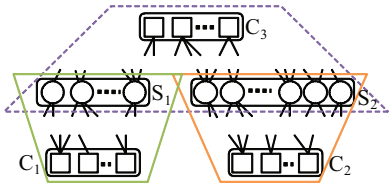


Figure 4.1 The bipartite graph T representing the input symbols S_1 and S_2 and the output symbols C_1, C_2, and C_3 resulting from a DU-rateless coding with two sources.

the bipartite graph T representing the input and output symbols has two types of variable nodes (mapped to S_1 and S_2), and three types of check nodes generated by R. Let C_1 and C_2 denote the set of output symbols directly forwarded from R, and C_3 denote the set of combined output symbols as shown in Figure 4.1.

Clearly, C_1 symbols are generated based on $\Omega(x)$ and are only connected to S_1. Similarly, C_2 symbols are generated based on $\varphi(x)$ and are only connected to S_2. Finally, output symbols of C_3 are generated using both S_1 and S_2 with a degree distribution equal to $\Omega(x) \times \varphi(x)$ [110]. It is worth noting that the ratio of the number of symbols in C_1, C_2, and C_3 is equal to p_1, p_2, and p_3, respectively.

Let us choose $T_{l,1}$ a subgraph of T as following. Choose an edge (v, w) uniformly at random from all edges in T with one end among S_1 symbols. Call the input symbol v connected to edge (v, w) the root of $T_{l,1}$, which is assumed to be at depth 0. $T_{l,1}$ is a graph induced by v and all neighbors of v within distance $2l$ after removing the edge (v, w). It can be shown that $T_{l,1}$ is a *tree* asymptotically [19–21]. Similarly, we define $T_{l,2}$ such that the root of $T_{l,2}$ resides in S_2 symbols.

In addition, in the iterative belief propagation LT decoding process on binary-erasure-channel (BEC) we can assume that messages (0 or 1) are sent along the edges from output symbols to input symbols, and then vice-versa [2, 19, 20, 22]. An input symbol sends 0 to an adjacent output symbol if and only if its value is not recovered yet. Similarly, an output symbol sends 0 to an adjacent input symbol if and only if it is not able to recover the value of the input symbol. In other words, an input symbol

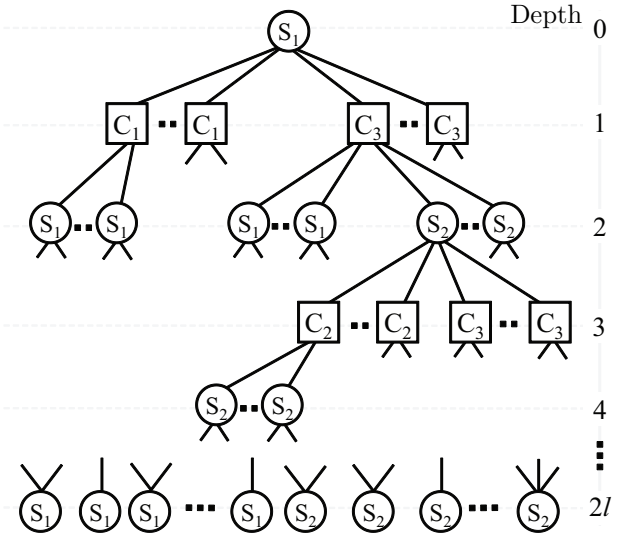


Figure 4.2 $T_{l,1}$ And-Or tree with two types of OR-nodes (S_1 and S_2 input symbols) and three types of AND-node (C_1, C_2, and C_3), with a root among S_1.

sends 1 to a neighboring output symbol if and only if it has received at least one message with value 1 from other neighboring output symbol, hence it is performing the logical OR operation. Also an input symbol sends 0 to a neighboring output symbol if only if it has received at least one message with value 0 from its other neighboring input symbols, which is a logical AND operation. Therefore, $T_{l,1}$ and $T_{l,2}$ are And-Or trees with OR and AND nodes on even and odd depths, respectively. Note that we denote the symbols at depth $i+1$ as the children of symbols at depth i. $T_{l,1}$ and $T_{l,2}$ have been shown in Figures 4.2 and 4.3.

Let $\delta_{i,1}, i \in \{0, \ldots, A_1\}$ be the probability that an input symbol in S_1 has i children in C_1 or C_3. Further, let $\delta_{i,2}$ be the probability that a S_2 symbol has $i \in \{0, \ldots, A_2\}$ children in C_2 or C_3. Moreover, let C_1 symbols choose to have $i \in \{0, \ldots, B_1 - 1\}$ children from S_1 with probability $\beta_{i,1}$, and C_2 choose to have $i \in \{0, \ldots, B_2 - 1\}$ children from S_2 with probability $\beta_{i,2}$.

Moreover, in $T_{l,1}$ C_3 symbols choose $i \in \{0, \ldots, B_1 - 1\}$ and $j \in \{1, \ldots, B_2\}$ children from S_1 and S_2 symbols with probabilities $\beta_{i,1}$ and $\beta_{j,3}$, respectively. Further,

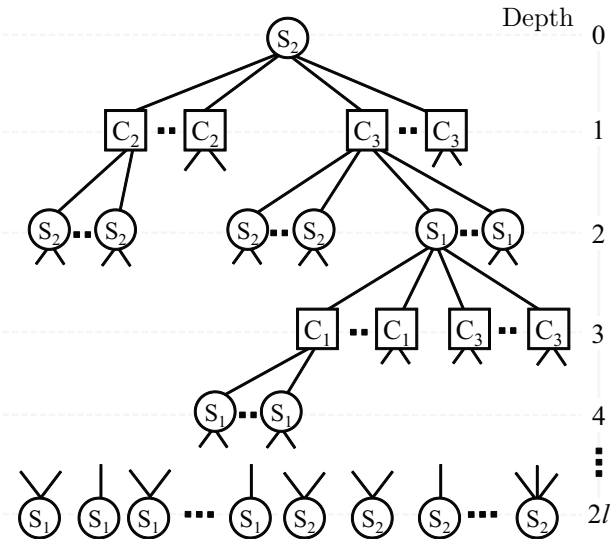


Figure 4.3 $T_{l,2}$ And-Or tree with two types of OR-nodes (S_1 and S_2 input symbols) and three types of AND-node (C_1, C_2, and C_3), with a root among S_2.

in $T_{l,2}$, C_3 symbols can choose $i \in \{0, \ldots, B_2 - 1\}$ and $j \in \{1, \ldots, B_1\}$ children from S_2 and S_1 symbols with probabilities $\beta_{i,2}$ and $\beta_{j,4}$, respectively. The probabilities that the root input symbol of And-Or trees $T_{l,1}$ and $T_{l,2}$ evaluate to 0 is given in the following Theorem.

Theorem 4.1 *Let $y_{l,1}$ and $y_{l,2}$ be the probabilities that the roots of the And-Or trees $T_{l,1}$ and $T_{l,2}$ evaluate to* 0*, respectively. Then we have*

$$y_{l,1} = \delta_1\Big(1 - p_1'\beta_1(1 - y_{l-1,1}) - p_3' \sum_{i=1}^{B_1+B_2-1} \sum_{j=0}^{i-1} \Big[\beta_{j,1}(1 - y_{l-1,1})^j \beta_{i-j,3}(1 - y_{l-1,2})^{i-j}\Big]\Big), \tag{4.1}$$

$$y_{l,2} = \delta_2\Big(1 - p_2'\beta_2(1 - y_{l-1,2}) - p_4' \sum_{i=1}^{B_1+B_2-1} \sum_{j=0}^{i-1} \Big[\beta_{j,2}(1 - y_{l-1,2})^j \beta_{i-j,4}(1 - y_{l-1,1})^{i-j}\Big]\Big), \tag{4.2}$$

with $y_{0,1} = y_{0,2} = 0$, $\delta_1(x) = \sum_{i=0}^{A_1} \delta_{i,1} x^i$, $\delta_2(x) = \sum_{i=0}^{A_2} \delta_{i,2} x^i$, $\beta_1(x) = \sum_{i=0}^{B_1-1} \beta_{i,1} x^i$, $\beta_2(x) = \sum_{i=0}^{B_2-1} \beta_{i,2} x^i$, $p'_1 = \frac{p_1}{1-p_2}$, $p'_3 = \frac{1-p_1-p_2}{1-p_2} = \frac{p_3}{1-p_2}$, $p'_2 = \frac{p_2}{1-p_1}$ *and* $p'_4 = \frac{1-p_1-p_2}{1-p_1} = \frac{p_3}{1-p_1}$.

Proof. Consider output symbols of depth 1 in $T_{l,1}$ (which are of type C_1 and C_3). A C_1 symbol has children in S_1 symbols of depth 2* and evaluates to 1 with probability $\sum_{i=0}^{B_1-1} \beta_{i,1}(1 - y_{l-1,1})^i$. A C_3 symbol may have between 0 to $B_1 - 1$ children from S_1 symbols and between 1 to B_2 children from S_2 symbols. Hence, the probability that such an input symbol evaluates to 0 is $\sum_{i=1}^{B_1+B_2-1} \sum_{j=0}^{i-1} [\beta_{j,1}(1 - y_{l-1,1})^j \beta_{i-j,3}(1 - y_{l-1,2})^{i-j}]$.

From the children of the *root* of $T_{l,1}$ at depth 0, p'_1 fraction are C_1 symbols and the rest p'_3 fraction are C_3 symbols. Hence, the probability that an output symbol that is a child of $T_{l,1}$'s root evaluates to 0 is $\Big(1 - p'_1 \sum_{i=0}^{B_1-1} \beta_{i,1}(1 - y_{l-1,1})^i - p'_3 \sum_{i=1}^{B_1+B_2-1} \sum_{j=0}^{i-1} \Big[\beta_{j,1}(1 - y_{l-1,1})^j \beta_{i-j,3}(1 - y_{l-1,2})^{i-j}\Big]\Big)$.

Therefore, the probability that the root of $T_{l,1}$ evaluates to 0, $y_{l,1}$, is given by (4.1). Note that $y_{l,2}$ can be analyzed in a similar way to obtain (4.2).

■

To complete DU-rateless codes analysis, we only need to compute the probabilities $\beta_{i,1}$, $\beta_{i,2}$, $\beta_{i,3}$, $\beta_{i,4}$, and functions $\delta_1(x) = \sum_i \delta_{i,1} x^i$ and $\delta_2(x) = \sum_i \delta_{i,2} x^i$. First, we need to investigate the degree distribution of input symbols in S_1 and S_2. In the following lemma, we show that the degree (the number of edges connected to) of each input symbol in the proposed ensemble of DU-rateless code with parameters $(\rho k, k, \Omega(x), \varphi(x), p_1, p_2, p_3, \gamma)$ is Poisson-distributed asymptotically.

*Note that S_1 and S_2 symbols at depth 2 in $T_{l,1}$ (as well as in $T_{l,2}$) are the roots for independent And-Or tree $T_{l-1,1}$ and $T_{l-1,2}$, respectively.

Lemma 4.1 *Consider two sources s_1 and s_2 employing a $(\rho k, k, \Omega(x), \varphi(x), p_1, p_2, p_3, \gamma)$ DU-rateless code. Asymptotically, for a total received overhead of γ the degree of S_1 and S_2 input symbols in the corresponding bipartite graph T follow Poisson distributions with means $\lambda_1 = \Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}$ and $\lambda_2 = \varphi'(1)\gamma(1-p_1)(1+\rho)$, respectively.*

Proof. The average degrees of $\Omega(x)$ and $\varphi(x)$ are given by $\sum_i i\Omega_i = \Omega'(1)$ and $\sum_i i\varphi_i = \varphi'(1)$, respectively. S_1 symbols are chosen based on $\Omega(x)$ and are included in a fraction $(1-p_2)$ of $(1+\rho)\gamma k$ total output symbols. Therefore, $\Omega'(1)(1+\rho)\gamma k(1-p_2)$ edges are connected uniformly at random to S_1 symbols. Consequently, a S_1 symbols has degree d with probability

$$\tau_{d,1} = \binom{(1-p_2)\Omega'(1)\gamma k(1+\rho)}{d} \times \left(\frac{1}{\rho k}\right)^d \left(1-\frac{1}{\rho k}\right)^{(1-p_2)\Omega'(1)\gamma k(1+\rho)-d}. \tag{4.3}$$

Similarly, $(1-p_1)\varphi'(1)k(1+\rho)\gamma$ edges are connected uniformly at random to S_2 symbols. As a result, a S_2 symbol has degree d with probability

$$\tau_{d,2} = \binom{(1-p_1)\varphi'(1)\gamma k(1+\rho)}{d} \times \left(\frac{1}{k}\right)^d \left(1-\frac{1}{k}\right)^{(1-p_1)\varphi'(1)\gamma k(1+\rho)-d}. \tag{4.4}$$

Asymptotically, (4.3) and (4.4) approach to

$$\tau_{d,1} = \frac{e^{-(1-p_2)\Omega'(1)\gamma\frac{(1+\rho)}{\rho}}\left[\Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}\right]^d}{d!}, \tag{4.5}$$

and

$$\tau_{d,2} = \frac{e^{-(1-p_1)\varphi'(1)\gamma(1+\rho)}\left[\varphi'(1)\gamma(1-p_1)(1+\rho)\right]^d}{d!}, \tag{4.6}$$

respectively, which are Poisson distributions with the means $\lambda_1 = \Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}$ and $\lambda_2 = \varphi'(1)\gamma(1-p_1)(1+\rho)$.

■

Next, employing Lemma 4.1 we find $\beta_{i,1}$, $\beta_{i,2}$, $\beta_{i,3}$, $\beta_{i,4}$, $\delta_1(x) = \sum_i \delta_{i,1}x^i$, and $\delta_2(x) = \sum_i \delta_{i,2}x^i$ as a function of a DU-rateless code parameters in the following lemma.

Lemma 4.2 *The probabilities $\beta_{i,1}$, $\beta_{i,2}$, $\beta_{i,3}$, $\beta_{i,4}$, and functions $\delta_1(x)$ and $\delta_2(x)$ for a $(\rho k, k, \Omega(x), \varphi(x), p_1, p_2, p_3, \gamma)$ DU-rateless code are given as*

$$\delta_1(x) = e^{(1-p_2)\Omega'(1)\gamma\frac{(1+\rho)}{\rho}(x-1)}, \delta_2(x) = e^{(1-p_1)\varphi'(1)\gamma(1+\rho)(x-1)},$$
$$\beta_{i,1} = \frac{(i+1)\Omega_{i+1}}{\Omega'(1)}, \text{ hence } \beta_1(x) = \frac{\Omega'(x)}{\Omega'(1)},$$
$$\beta_{i,2} = \frac{(i+1)\varphi_{i+1}}{\varphi'(1)}, \text{ hence } \beta_2(x) = \frac{\varphi'(x)}{\varphi'(1)},$$
$$\beta_{i,3} = \varphi_i, \text{ and } \beta_{i,4} = \Omega_i.$$

Proof. We have $\beta_{i,1}$ is the probability that a randomly chosen edge with one end in S_1 is connected to a C_1 or C_3 symbol with i children in S_1. Therefore, $\beta_{i,1}$ is the probability that a randomly selected edge with one end connected to a S_1 symbol has the other end connected to an output symbol in C_1 or C_3 with $(i+1)$ children in S_1. Therefore, we have $\beta_{i,1} = \frac{(i+1)\Omega_{i+1}}{\Omega'(1)}$ or equivalently $\beta_1(x) = \frac{\Omega'(x)}{\Omega'(1)}$, which is edge degree distribution from C_1 perspective. Similarly, we have $\beta_{i,2} = \frac{(i+1)\varphi_{i+1}}{\varphi'(1)}$, which gives $\beta_2(x) = \frac{\varphi'(x)}{\varphi'(1)}$, which is edge degree distribution from C_2 perspective.

Moreover, $\beta_{i,3}$ is the probability that a randomly chosen edge with one end in S_1 is connected to a C_3 symbol with i children in S_2. Therefore, $\beta_{i,3}$ is the probability

that a randomly selected edge connected to a S_1 symbol in the graph T is connected to a C_3 output symbol with i children in S_2. This simply gives $\beta_{i,3} = \varphi_i$. In the same way, $\beta_{i,4} = \Omega_i$.

Further, we have $\delta_{i,1}$ is the probability that the input symbol connected to a randomly selected edge has degree $i+1$ given that the input symbol belongs to S_1. Therefore, $\delta_{i,1} = \frac{(i+1)\lambda_{i+1,1}}{\sum_i i\lambda_{i,1}}$, where $\lambda_{i,1}$ is given in Lemma 4.1. Using Lemma 4.1, we have

$$\begin{aligned}
\delta_{i,1} &= \frac{(i+1)\lambda_{i+1,1}}{\Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}}, \\
&= \frac{(i+1)e^{-(1-p_2)\Omega'(1)\gamma\frac{(1+\rho)}{\rho}}\left[\Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}\right]^{i+1}}{\Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}(i+1)!}, \\
&= \frac{e^{-(1-p_2)\Omega'(1)\gamma\frac{(1+\rho)}{\rho}}\left[\Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}\right]^{i}}{i!}.
\end{aligned}$$

After substitution, we have

$$\begin{aligned}
\delta_1(x) &= \sum_i \delta_{i,1}x^i, \\
&= \sum_i \frac{e^{-(1-p_2)\Omega'(1)\gamma\frac{(1+\rho)}{\rho}}\left[\Omega'(1)\gamma(1-p_2)\frac{(1+\rho)}{\rho}x\right]^{i}}{i!}, \\
&= e^{(1-p_2)\Omega'(1)\gamma\frac{(1+\rho)}{\rho}(x-1)}.
\end{aligned}$$

Similarly, we have $\delta_2(x) = e^{(1-p_1)\varphi'(1)\gamma(1+\rho)(x-1)}$.

■

Similar to [19, Lemma 4], we can show that the sequences $\{y_{l,1}\}_l$ and $\{y_{l,2}\}_l$ are monotone decreasing and are bounded in $[0,1]$, and they converge to fixed points. Let BER_1 and BER_2 denote the corresponding fixed points. These fixed points are the probabilities that S_1 and S_2 symbols are not recovered after l decoding iter-

ations. In other words, these fixed points are the final decoding error rates of a $(\rho k, k, \Omega(x), \varphi(x), p_1, p_2, p_3, \gamma)$ DU-rateless code. To realize almost minimal BER_1 and BER_2, we will design DU-rateless codes with parameters that are *jointly* optimized for a given γ_{succ} in the next section.

4.3 Distributed Unequal-Error-Protection Rateless Codes Design

For an ensemble of DU-rateless code with parameters $(\rho k, k, \Omega(x), \varphi(x), p_1, p_2, p_3, \gamma)$, we define the UEP gain $\eta \triangleq \frac{\text{BER}_2}{\text{BER}_1}$, where BER_1 and BER_2 can be computed from (4.1) and (4.2), respectively, for a large enough l. A larger η shows a higher recovery rate of S_1 input symbols at D or equivalently a higher level of protection compared to S_2. It is worth noting that $\eta = 1$ corresponds to *equal-error-protection* (EEP) case where S_1 and S_2 are equally protected. The question that arises is that what are the appropriate parameters $\Omega(x)$, $\varphi(x)$, p_1, p_2, and p_3 that would result in a *desired* η and minimal BER_1 and BER_2. It is not hard to show that BER_1 and BER_2 are two *conflicting* objective functions by investigating (4.1) and (4.2) (improving one may deteriorate the other one). Therefore, we have a *multi-objective optimization* problem.

4.3.1 Proposed Codes Design Employing NSGA-II

We fix $\gamma_{succ} = 1.05$ and employ NSGA-II [101] (see Section 2.11 and refer to [101] for more information on NSGA-II) to find the optimum $\Omega(x)$, $\varphi(x)$, p_1, p_2, and p_3 that *concurrently minimize* BER_1 and BER_2 for various values of $\eta = \frac{\text{BER}_2}{\text{BER}_1}$ and $\rho \in \{0.3, 0.5, 1\}$. In other words, we have a problem including two objective functions given by (4.1) and (4.2) (BER_1 and BER_2), with 202 independent decision variables, i.e. $\underline{u} = \{\Omega_1, \Omega_2, \ldots, \Omega_{10^2}, \varphi_1, \varphi_2, \ldots, \varphi_{10^2}, p_1, p_2\}$.

The output of our optimization are 3 databases of *close to optimal* degree dis-

tributions for $\rho \in \{0.3, 0.5, 1\}$, each embracing a large number of DU-rateless codes parameters that realize various η's made available online at [107]. We emphasis that our results are close to optimal since genetic algorithms are known to find solutions that are not necessarily global-optimal but are rather *very close* to global-optimal solutions. In addition, confining the largest degree to $B_1 = B_2 = 10^2$ limits the degree distribution search space and results in the design of the codes that are suboptimal. Therefore, the performance of our designed DU-rateless codes is close to optimal. We plot the pareto fronts obtained from our optimizations in Figure 4.4(a). Similarly, we set $\gamma_{succ} = 1.02$ and $\rho = 1$ and find the set of optimal DU-rateless codes for this setup with the pareto front illustrated in Figure 4.4(b).

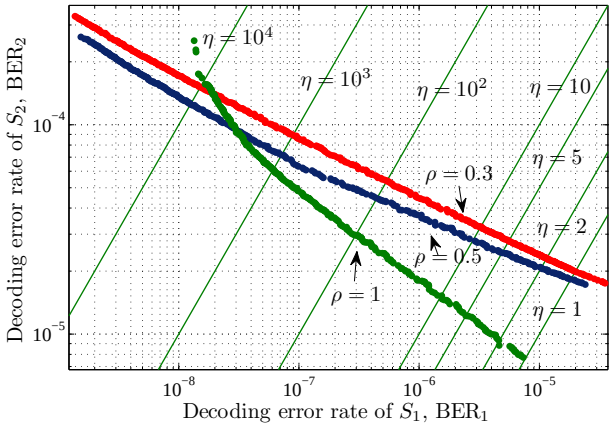
In Figure 4.4 each point corresponds to two degree distributions and three relaying parameters $\Omega(x)$, $\varphi(x)$, p_1, p_2, and p_3. Figure 4.4(a) shows that our designed DU-rateless codes are well spread with respect to η. One should choose an appropriate point according to a desired η and employ the corresponding DU-rateless code. From Figure 4.4(b) we can see that due to much smaller γ_{succ} the minimum achievable error rates have increased, which shows an interesting trade-off between the achievable error-floor and the decoding overhead γ_{succ}. However, the UEP gain can be obtained for a wide range of η's.

4.4 Performance Evaluation of the Designed Codes

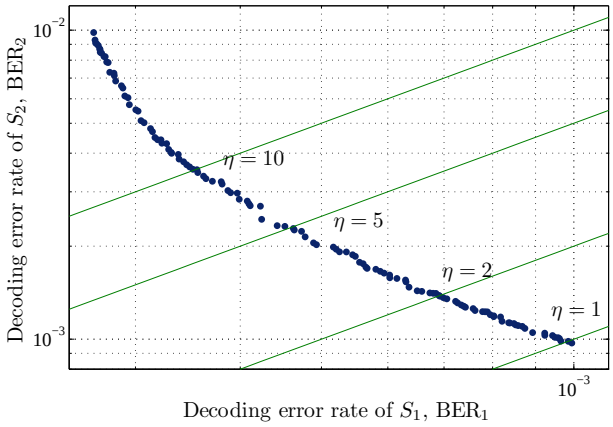
This section report the performance evaluation of our designed codes.

4.4.1 Asymptotic Performance Evaluation of the Designed Codes

From the sets of our optimized DU-rateless codes available at [107], we choose two DU-rateless codes for $\eta \in \{10, 10^2\}$, $\rho = 1$, and $\gamma_{succ} = 1.05$ and evaluate their performance in Figure 4.5(a) for $k \to \infty$ given by (4.1) and (4.2). For comparison,



(a) The resulting pareto fronts for DU-rateless codes design with $\gamma_{succ} = 1.05$ and $\rho \in \{0.3, 0.5, 1\}$.



(b) The resulting pareto fronts for DU-rateless codes design with $\gamma_{succ} = 1.02$ and $\rho = 1$.

Figure 4.4 The resulting pareto fronts for various DU-rateless codes setups

we have also plotted the BER_1 and BER_2 for EEP case ($\eta = 1$). Similarly, we choose an optimal DU-rateless codes with parameters $\gamma_{succ} = 1.02$ and $\rho = 1$ for $\eta = 10$ and evaluate its performance as shown in Figure 4.5(b).

Figure 4.5(a) shows that the expected UEP gain is fulfilled for $\gamma_{succ} = 1.05$ with the minimal values of BER_1 and BER_2. In addition, Figure 4.5(b) shows that the expected UEP gain $\eta = 10$ is achieved although the error floors are higher due to smaller γ_{succ}. The parameters of a DU-rateless code for $\rho = 1$, $\eta = 10$, and $\gamma_{succ} = 1.05$ with performance illustrated in Figure 4.5(a) is given as follows.

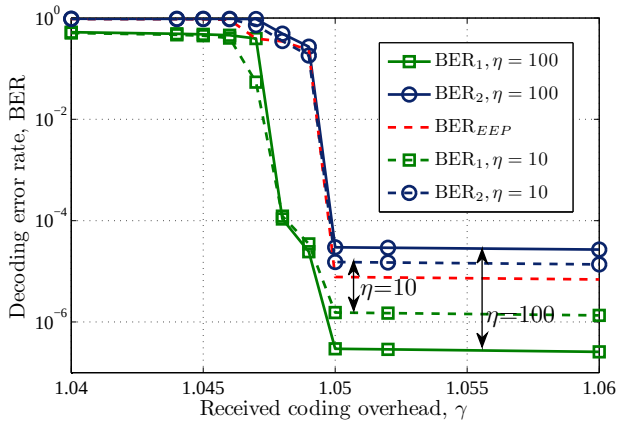
$$\begin{aligned}\Omega(x) =&0.039x^1 + 0.492x^2 + 0.094x^3 + 0.09x^4 + 0.096x^5 + 0.002x^6 \\ &+ 0.055x^7 + 0.019x^8 + 0.033x^9 + 0.014x^{10} + 0.004x^{20} \\ &+ 0.005x^{27} + 0.001x^{28} + 0.004x^{31} + 0.001x^{39} + 0.005x^{43} \\ &+ 0.004x^{78} + 0.001x^{79} + 0.005x^{86} + 0.01x^{95} + 0.004x^{96} \\ &+ 0.001x^{99} + 0.006x^{100},\end{aligned} \tag{4.7}$$

$$\begin{aligned}\varphi(x) =&0.072x^1 + 0.48x^2 + 0.055x^3 + 0.051x^4 + 0.063x^5 + 0.059x^6 \\ &+ 0.037x^7 + 0.026x^8 + 0.025x^9 + 0.036x^{10} + 0.005x^{15} \\ &+ 0.001x^{25} + 0.002x^{28} + 0.005x^{37} + 0.002x^{44} + 0.001x^{67} \\ &+ 0.001x^{70} + 0.001x^{76} + 0.001x^{77} + 0.002x^{83} + 0.001x^{84} \\ &+ 0.001x^{88} + 0.003x^{93} + 0.052x^{95} + 0.002x^{97},\end{aligned} \tag{4.8}$$

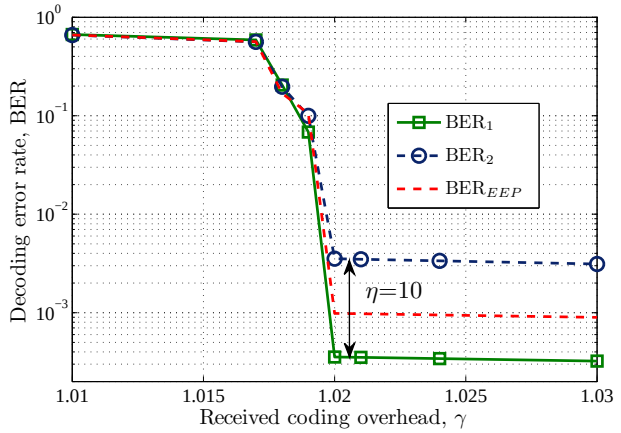
with $p_1 = 0.4822$, and $p_2 = 0.1173$, which gives $p_3 = 0.4005$. We can see that to achieve an optimum distributed coding 40.05% of the generated output symbols at s_1 and s_2 should be combined at the relay.

4.4.2 Performance Evaluation for Finite-length

Our designed DU-rateless codes are optimized based on our analytical results derived in Section 4.2 for asymptotic case, i.e., $k \to \infty$. However, in practice k is finite.



(a) The resulting BERs with optimized sets of parameters for $\eta \in \{10, 10^2\}$, $\gamma_{succ} = 1.05$, and $\rho = 1$.



(b) The resulting BERs with optimized sets of parameters for $\eta = 10$, $\gamma_{succ} = 1.02$, and $\rho = 1$.

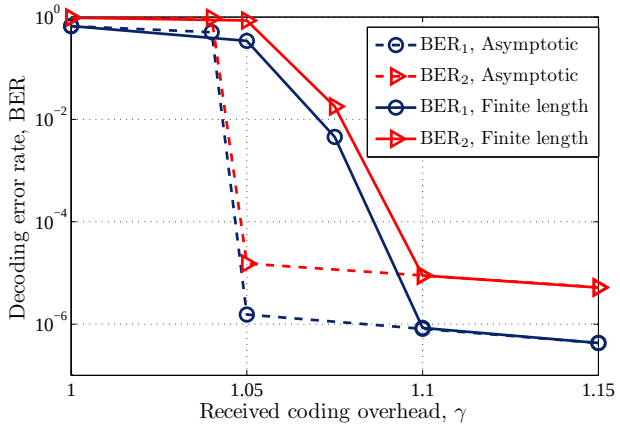
Figure 4.5 Asymptotic performance evaluation of the designed DU-rateless codes

Therefore, we set the parameters $\rho = 1$ and $\eta = 10$ for two cases of $\gamma_{succ} = 1.05$ and $\gamma_{succ} = 1.02$ and evaluate the performance of DU-rateless code for $k = 10^4$ using numerical encoding and decoding versus anosmatic setup as shown in Figure 4.6. To find BER_1 and BER_2 in the finite length case, we take average over decoding error rates of 10^5 numerical decoding iterations. Figure 4.6 shows that the expected UEP gain ($\eta = 10$) and minimal error rates are realized at slightly larger γ_{succ}'s. Therefore, our designed DU-rateless codes can indeed be employed for finite k cases as well for a larger γ_{succ}.

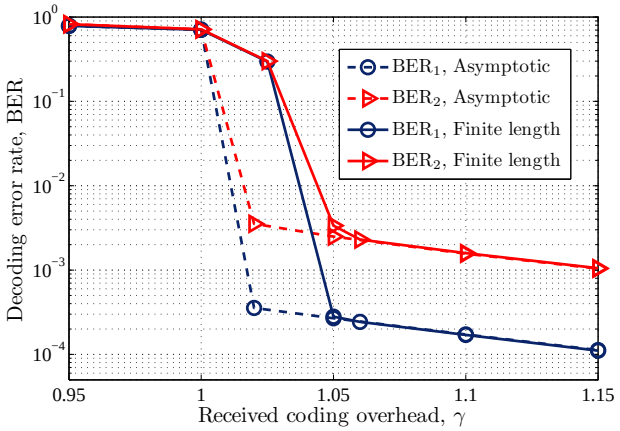
4.4.3 Performance Comparison with LT and DLT Codes

In this section, we compare the performance of DU-rateless codes with the case where s_1 and s_2 independently employ two LT codes $\mathcal{C}_1$ and $\mathcal{C}_2$ to generate C_1 and C_2, and R directly and intermittently forwards them to D. To perform the comparison, we set the parameters $k = 10^4$, $\rho = 1$, and $\eta = 10$. The DU-rateless code optimized for this setup has degree distributions given by (4.7) and (4.8) with $p_1 = 0.4822$, $p_2 = 0.1173$, and $p_3 = 0.4005$, which achieves $\text{BER}_1 \approx 5 \times 10^{-7}$ and $\text{BER}_2 \approx 5 \times 10^{-6}$ at $\gamma = 1.15$. This DU-rateless code results in output symbols with average degree of $\mu_{DU} \approx 11.38$.

To perform a *fair* comparison, we need to have equivalent *decoding complexities* in both setups. Since the decoding complexity of LT decoding is determined by the average output symbols degree [1], we need to maintain the same average coded degree when two LT codes replace this DU-rateless code. Let $\mathcal{C}_1(c_1, \nu_1)$ and $\mathcal{C}_2(c_2, \nu_2)$ denote the desired LT codes, where c_1, ν_1, c_2, and ν_2 are the respective Robust-Soliton degree distributions parameters [1]. Further, assume that $\mathcal{C}_1(c_1, \nu_1)$ and $\mathcal{C}_2(c_2, \nu_2)$ have average output symbol degrees of $\mu_{\mathcal{C}_1}$ and $\mu_{\mathcal{C}_2}$ and realize the desired BER's at $\gamma_{\mathcal{C}_1}$ and $\gamma_{\mathcal{C}_2}$ in rateless decoding, respectively. Consequently, to have equal decoding complexities in both setups we need to find $\mathcal{C}_1(c_1, \nu_1)$ and $\mathcal{C}_2(c_2, \nu_2)$ such that $\frac{\gamma_{\mathcal{C}_1}\mu_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}\mu_{\mathcal{C}_1}}{\gamma_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}} = \mu_{DU}$. On the other hand, we should select c_1, ν_1, c_2, and ν_2 such



(a) BER's for DU-rateless codes optimized for $\gamma_{succ} = 1.05$.



(b) BER's for DU-rateless codes optimized for $\gamma_{succ} = 1.02$.

Figure 4.6 The resulting BERs for asymptotic case and finite length case ($k = 10^4$) for DU-rateless codes optimized for $\gamma_{succ} = 1.05$ and $\gamma_{succ} = 1.02$ with parameters $\eta = 10$ and $\rho = 1$.

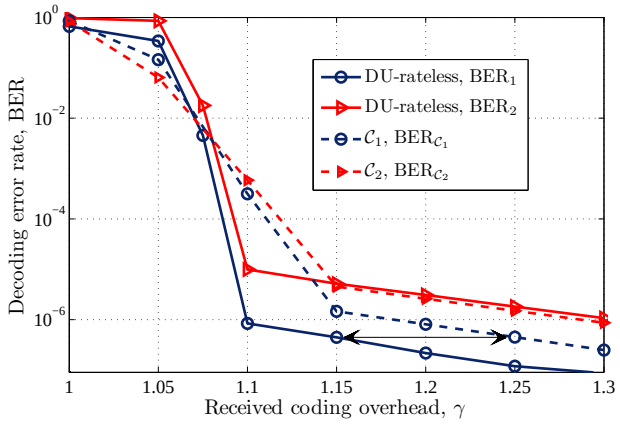


Figure 4.7 Performance comparison of the employed DU-rateless code and the equivalent optimal separate LT codes. As shown, the overhead for achieving $\text{BER}_1 = 5 \times 10^{-7}$ reduces from 1.25 to 1.15 if we employ a DU-rateless code instead of two separate LT codes.

that $\mathcal{C}_1(c_1, \nu_1)$ and $\mathcal{C}_2(c_2, \nu_2)$ can realize the desired BER's at minimum possible total overhead $\gamma_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}$. Therefore, to find $\mathcal{C}_1(c_1, \nu_1)$ and $\mathcal{C}_2(c_2, \nu_2)$ we solve the following minimization problem:

$$\begin{aligned} &\underset{c_1,\nu_1,c_2,\nu_2}{\text{argmin}} \ (\gamma_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}) = [c_1^*, \nu_1^*, c_2^*, \nu_2^*], \\ &\text{s.t.} \ \frac{\gamma_{\mathcal{C}_1}\mu_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}\mu_{\mathcal{C}_2}}{\gamma_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}} = \mu_{DU}, \\ &\text{BER}_1 \leq 5 \times 10^{-7}, \text{and BER}_2 \leq 5 \times 10^{-6}. \end{aligned} \tag{4.9}$$

We search the whole decision space of c_1, ν_1, c_2, and ν_2 to find the global minimum of $\gamma_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2}$. The optimal $\mathcal{C}_1$ has parameters $c_1 = 0.1, \nu_1 = 40, \gamma_{\mathcal{C}_1} = 1.15$, and $\mu_{\mathcal{C}_1} = 10.74$. Further, the optimal $\mathcal{C}_2$ has parameters $c_1 = 0.1, \nu_1 = 15, \gamma_{\mathcal{C}_1} = 1.25$, and $\mu_{\mathcal{C}_1} = 11.98$. We have compared the performance of the setup with two separate LT codes $\mathcal{C}_1(c_1, \nu_1)$ and $\mathcal{C}_2(c_2, \nu_2)$ along with the equivalent DU-rateless code in Figure 4.7.

Figure 4.7 shows that the total amount of required overhead has decreased from $\gamma_{\mathcal{C}_1} + \gamma_{\mathcal{C}_2} = 2.4$ in separate coding setup to $(1 + \rho)\gamma_{succ} = 2.3$ in the setup employing

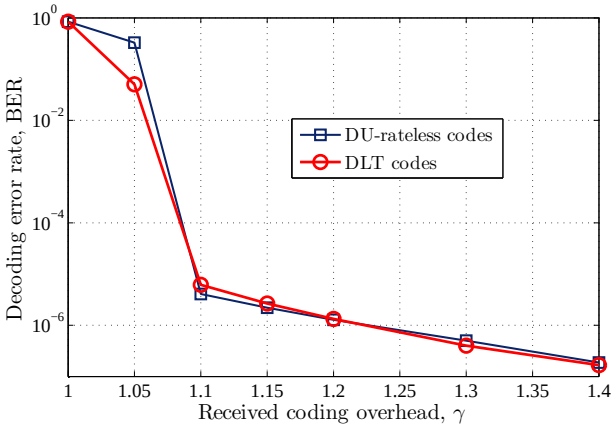


Figure 4.8 Performance comparison of the DU-rateless codes for designed for $\rho = 1$, $\eta = 1$, $\gamma = 1.05$ and the DLT codes with average output degree of 11.03 for $k = 10^4$.

DU-rateless codes. This shows that when DU-rateless codes are employed 10^3 fewer symbols need to be delivered to receiver. Therefore, in our example fDU-rateless codes can make 25% reduction in the number of required redundant received output symbol for successful decoding compared to two separate LT codes. This improvement is realized by increasing data block length, which is obtained by combining output symbols at the relay.

To compare DU-rateless codes with DLT codes [110], we have to select a DU-rateless code with $\rho = 1$ and $\eta = 1$ since DLT codes can only encode data blocks of equal size and may only provide EEP. This DU-rateless code results in the generation of output symbols with average degree of 11.03. Similar to comparison with regular LT codes, we find a Robust-Soliton distribution for DLT coding with average degree 11.03 and compare its performance to the selected DU-rateless code in Figure 4.8 for $k = 10^4$. Figure 4.8 interestingly shows that for $\rho = 1$ and $\eta = 1$ DLT and DU-rateless codes have almost the same performance and achieve the same error floor. However, we should note that DU-rateless codes are capable of providing UEP and also support sources with unequal block sizes.

4.5 Conclusion

In this chapter, we proposed *DU-rateless codes*, which are distributed rateless codes with unequal-error-protection property for two data sources with *unequal* data block lengths over erasure channels. First, we analyzed DU-rateless codes employing And-Or tree analysis technique, and then we designed several close to optimum sets of DU-rateless codes using multi-objective genetic algorithms. Performance comparison of the designed DU-rateless codes showed that they fulfilled the expected UEP property with almost minimal error rates. We also showed that although DU-rateless codes are designed for large message lengths, they can be employed for finite message lengths as well. Finally, we showed that DU-rateless codes surpass the performance of exiting LT codes in distributed rateless coding.

CHAPTER 5

LT-SF CODES: LT CODES WITH SMART FEEDBACK

In this section, we design LT codes with *smart feedback* that improves the performance of LT codes for short data blocks.

5.1 Introduction

LT codes [1] require only *one* feedback that is issued by the decoder (receiver) to inform the encoder (transmitter) of a successful LT decoding. Although requiring a single feedback is an outstanding advantage of LT codes, the available feedback channel remains *unused* during the transmission. In addition, as the data-block length decreases the performance of LT codes significantly deteriorates [1, 17, 18]. Therefore, in [102, 113–116] it has been proposed to employ the feedback channel during the transmission as well to keep the encoder aware of the decoders's status. In this way, the performance of LT codes for short data-block lengths considerably increases. However, we should note that feedback channels are usually resource constrained and have lower data transmission capability compared to forward channels. Therefore, the design of a feedback scheme should be cleverly devised to consider these scarce resources.

In this chapter, we propose *LT-SF* codes, which are LT codes with *smart* feedback [11]. The main idea to design LT-SF codes is that the decoder may issue *two types* of feedback according to its *needs*. The existing LT codes with feedback (as are extensively explored later) are designed such that the decoder informs the encoder

with the number of successfully decoded input symbols [102, 113, 114], a suitable input symbol for decoding [115], or the index of some recovered input symbols [116].

In contrast, in LT-SF codes we do not confine the information content type of the feedbacks. Hence, the decoder may alternatively issue feedbacks to inform the encoder with the number of successfully decoded input symbols or request a specific input symbol that makes the largest progress in the decoding of the data-block. To generate the latter type of feedback, we propose three novel algorithms (with a trade-off in their algorithm complexity and performance) that describe how to analyze the decoder's status and select suitable input symbols to request. We show that LT-SF codes considerably surpass existing algorithms in the number of required *output symbols* (LT coded packets) for full decoding and the total number of required feedbacks.

Further, we consider a realistic feedback channel with unknown or varying erasure rate $\varepsilon_{fb} \in [0, 1)$ in contrast to previous work [102, 113–116], which assumed $\varepsilon_{fb} = 0$. We design LT-SF codes such that high feedback loss rates does not considerably degrade the recovery error rate of data block at the decoder, and we refer to this property by having a *high resiliency* against feedback channel loss. To detect feedback losses by decoder we propose a novel idea to employ the encoded symbols of degree one (as fully described later) as ACK to the reception of a feedback at the encoder. Therefore, all feedback losses will be discovered by decoder and feedback retransmissions will be performed until the encoder receives the feedback. This novel capability of LT-SF codes considerably distinguishes them with exiting work on LT codes with feedback.

Authors in [113] proposed *shifted LT* (SLT) codes to exploit the available feedback channel. They have shown that when n input symbols have been recovered at the decoder, the degree of each arriving output symbol decreases by an expected $\frac{k-n}{k}$ fraction (due to earlier recovery of their neighboring input symbol). Therefore, they

propose to *shift* the RS distribution (as discussed in Section 2.1) such that its average degree $\mathrm{RS}'(1)$ is increase by $\frac{k}{k-n}$, where $\mathrm{RS}'(x)$ is the first derivative of $\mathrm{RS}(x)$ with respect to its variable x. With this setup, arriving output symbols at decoder *always* maintain an RS degree distribution regardless of the value of n. SLT codes considerably improve the performance of LT codes. Therefore, we make some changes to the idea of *distribution shifting* proposed in SLT codes and employ it in the design of LT-SF codes, while showing that LT-SF codes considerably outperform SLT codes.

In contributions [102] and [114], *Growth codes* and *RT-oblivious codes* have been proposed, respectively, which have basically the same structure. In these algorithms, as n increases and reaches to certain thresholds a feedback indicating that decoder has achieved the corresponding threshold is initiated. Therefore, the encoder gradually increases the degree of output symbols on-the-fly based on the feedbacks such that the *instantaneous* decoding probability of each delivered output symbol is maximized. Since Growth and RT-oblivious codes only consider the instantaneous recovery probability of each output symbol upon reception, they do not have a good performance compared to SLT and LT-SF codes.

Authors in [115] propose to employ IS degree distribution $\mathrm{RS}^I(x)$ for LT coding. They have proposed to start decoding when an overhead of $\gamma = 1$ has been delivered to the decoder. When the decoding halts during the decoding process and some input symbols are remaining unrecovered, a randomly selected input symbol that is a neighbor of an output symbol of degree two is requested from the encoder. This algorithm is performed iteratively to decoding completion. Despite the advantages of algorithm proposed in [115], in this scheme many feedbacks are issued back-to-back as soon as γ exceeds 1. Further, during the iterative request process all degree-two output symbols may be *consumed* (decoded), while more input symbols are remaining uncovered. Therefore, this scheme may results in high *error-floors* due to remaining undecoded input symbols.

Let $\Omega_{k,n}(x)$ denote the degree distribution of LT-SF codes for a data-block of length k when n input symbols are already recovered at decoder. We adopt the idea of SLT codes [113], and propose to *shift* $\Omega_{k,n}(x)$ based on n. Therefore, we allow the decoder to issue the first type of feedback referred to by fb_1, which is used to keep the encoder updated with the current value of n.

Although IS distribution (see Section 2.1) is solely designed for the theoretical analysis of RS distribution, we slightly modify it and employ it in the encoding phase of LT-SF codes in combination with two types of feedback. The IS distribution is tuned for $\gamma = 1$ such that at each decoding iteration in expectation exactly one input symbol is recovered and only one output symbols is reduced to degree 1 and is added to the ripple. The single output symbol in the ripple with degree-one can decode one input symbol in the next iteration. Since on average only a single degree-one output symbol is generated for k output symbols (note that $\mathrm{RS}_1^I = \frac{1}{k}$, see Section 2.1), the IS distribution would realize an optimal coding/decoding, i.e., complete recovery of k input symbols from k output symbols and $\gamma_{succ} = 1$.

However, due to inherent randomness and uncertainties in the output symbol generation there is a high probability that an output symbol does not reduce to degree one when an input symbol is recovered. Consequently, the ripple becomes empty and the decoding stops although undecoded output and unrecovered input symbols are remaining. Therefore, while the IS distribution shows an ideal behavior in terms of the expected number of encoding symbols needed to recover the data, it is quite fragile and in fact so much so that it is *useless* in practice [1]. Despite this, we can easily see that if we exploit the feedback channel and *request a suitable* input symbol (which is an output symbol of degree 1), the decoding may continue and we may employ IS distribution. Therefore, we allow the encoder to request desired input symbols employing the first type of feedback referred to by fb_2.

Moreover, to design $\Omega_{k,n}(x)$ we propose to modify the IS distribution such that the encoder does not generate *any* degree-one output symbol. With this setup, we may exploit the degree-one output symbols as *acknowledgments* from the encoder to the reception of feedbacks. Therefore, the encoder generates a degree-one output symbol if and only if it has received a fb_1 or fb_2. Consequently, the lack of the arrival of an output symbol at the decoder with degree-one after issuing a fb_1 or fb_2 clearly indicates a feedback loss. Consequently, all feedback packet losses are identified by the decoder and a feedback retransmission is performed.

We should note that a degree-one output symbol contains a *randomly* selected input symbol after a fb_1 and the requested input symbol after a fb_2. In this way, the decoding recovery rate of LT-SF codes does not considerably degrade at *high* feedback channel loss rates $\varepsilon_{fb} \in [0, 1)$ in contrast to existing work [102, 113–116].

Let $\Omega_{k,n}(x) = \sum_{i=1}^{k} \Omega_{k,n,i} x^i$, where $\Omega_{k,n,d}$ is the probability of selecting degree d to generate an LT-SF output symbol. Since we do not allow the encoder to generate any degree-one symbol, we set $\Omega_{k,n,1} = 0$. Further, let $\mathrm{RS}_k^I(x) = \sum_{i=1}^{k} \mathrm{RS}_{k,i}^I x^i$ be the IS distribution for data block of length k. Employing the distribution shifting idea from [113] we define $\Omega_{k,n,d}$ as follows.

$$\Omega_{k,n,d} = \begin{cases} 0 & d = 1, \\ \frac{k}{k-1}\mathrm{RS}_{k-n,i}^I & d = 2, 3, \ldots, k, \lceil \frac{i}{1-\frac{n}{k}} \rfloor = d, \end{cases} \tag{5.1}$$

where $\lceil . \rfloor$ returns the closest integer to its argument and $\frac{k}{k-1}$ is the normalizing factor to have $\sum_d \Omega_{k,n,d} = 1$.

Lemma 5.1 *The average degree of a check node generated employing $\Omega_{k,n}(x)$ distribution is*

$$\sum_i i\Omega(i) = \Omega'_{k,n}(1) \approx \frac{k^2 \ln(k-n)}{(k-n)(k-1)}, \tag{5.2}$$

where $\Omega'_{k,n}(x)$ is the first derivative of $\Omega_{k,n}(x)$ with respect to its variable x.

Proof. The average degree of IS distribution $\mathrm{RS}^I(x)$ is $\ln k$ (see Section 2.1). Therefore, it is easy to see that $\Omega'_{k,0}(1) \approx \frac{k}{k-1} \ln k$, since for $n = 0$ no shift occurs in IS distribution and degree-one check nodes are not generated. Generalization for $\Omega'_{k,n}(1)$ is straightforward. Considering that the average degree of IS degree distribution for a data-block length of $k-n$ is $\ln(k-n)$ and the shift of degree distribution increases the average degree by a factor of $\frac{k}{k-n}$, (5.2) is obtained.

■

5.2.1 Generating fb_1

Obviously, the decoder is not always aware of n unless its knowledge about n is updated by a fb_1. Initially, the encoder assumes $n = 0$ and employs the degree distribution $\Omega_{k,0}(x)$ to generate output symbols. Let n_r denote the most recent reported value of n employing a fb_1. Similar to [113], we propose the encoder to generate a fb_1 when $\Omega'_{k,n}(1) - \Omega'_{k,n_r}(1) \geq \sqrt{\ln k}$, i.e., average degree of $\Omega_{k,n}(x)$ increases by at least $\sqrt{\ln k}$. Let n_i be the *threshold* that for $n \geq n_i$ the i^{th} fb_1 is generated. In the following lemma we give the expression for n_i.

Lemma 5.2 *In LT-SF codes with data-block length k, n_i the threshold of n for which i^{th} fb_1 is issued is recursively obtained as follows.*

$$\begin{aligned} n_0 &= 0 \\ n_i &= \left\lceil k + \frac{W_{-1}\left(-A_i(k)\right)}{A_i(k)} \right\rceil, i > 0, \end{aligned} \tag{5.3}$$

where $A_i(k) = \frac{1}{k}\left(\frac{k-1}{k}\sqrt{\ln k} + \frac{k}{k-n_{i-1}} \ln\left(k - n_{i-1}\right)\right)$ and $W_m(.)$ is the m^{th} root of Lam-

bert W-Function *(the Lambert W-Function is defined as the inverse function of* $f(x) = x \exp x$ *[117]).*

Proof. Let us first analyze n_1 the value of n that initiates the first fb_1. Since before the first fb_1 no distribution shifting occurs we have $\Omega'_{k,n_0}(1) = \Omega'_{k,0}(1) = \frac{k}{k-1} \ln k$. Therefore, the first fb_1 is issued for a value of n_1 that $\Omega'_{k,n_1}(1) - \Omega'_{k,0}(1) = \sqrt{\ln k}$. Using Lemma 5.1 we have

$$\frac{k}{k-n_1}\frac{k}{k-1}\ln(k-n_1) - \frac{k}{k-n_0}\frac{k}{k-1}\ln(k-n_0) = \sqrt{\ln k}, \tag{5.4}$$

which gives

$$\frac{\ln(k-n_1)}{k-n_1} = \frac{1}{k}\left(\frac{k-1}{k}\sqrt{\ln k} + \frac{k}{k-n_0}\ln(k-n_0)\right). \tag{5.5}$$

Next, let $A_i(k) = \frac{1}{k}\left(\frac{k-1}{k}\sqrt{\ln k} + \frac{k}{k-n_{i-1}}\ln(k-n_{i-1})\right)$. Since $A_i(k) > -\pi, \forall k, i$ employing Lambert's W function, we have

$$k - n_1 = -\frac{W_{-1}(-A_1(k))}{A_1(k)}, \tag{5.6}$$

which gives

$$n_1 = \left\lceil k + \frac{W_{-1}(-A_1(k))}{A_1(k)} \right\rceil. \tag{5.7}$$

Further, we can easily see that n_2 can be obtained from $\Omega'_{k,n_2}(1) - \Omega'_{k,n_1}(1) = \sqrt{\ln k}$ that in the same way gives

$$n_2 = \left\lceil k + \frac{W_{-1}(-A_2(k))}{A_2(k)} \right\rceil. \tag{5.8}$$

Finally, we have $\Omega'_{k,n_i}(1) - \Omega'_{k,n_{i-1}}(1) = \sqrt{\ln k}$ that proves the lemma.

■

Lemma 5.2 gives the value of n for which fb_1's are generated. In Figure 5.1, we have depicted $\frac{n_i}{k}, i \in \{1, 2, \ldots, 5\}$ versus k. From Figure 5.1, we see can that $\frac{n_i}{k}$ decreases as k increases. As an example, we can see that at $k = 10^2$ the first and the second fb_1's are issued at $n \geq 39$ and $n \geq 58$, respectively. Further, for $k = 10^4$ the first and the second fb_1's are issued at $n \geq 2740$ and $n \geq 4346$, respectively.

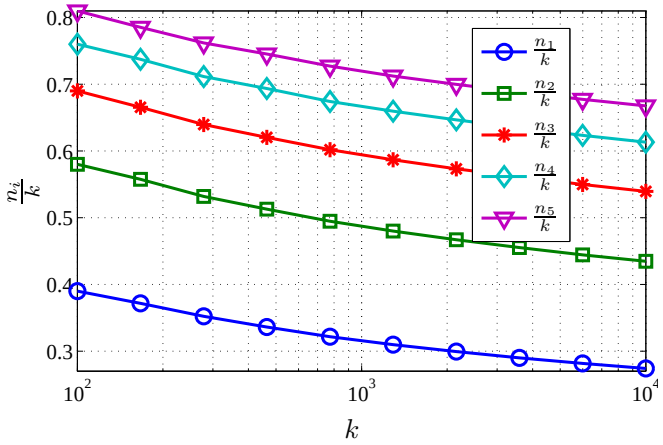


Figure 5.1 Values of $\frac{n_i}{k}, i \in \{1, 2, \ldots, 5\}$ versus k.

5.2.2 Generating fb_2

Since in LT-SF coding no degree-one output symbol is generated, no decoding is performed and we have $n = 0$ until some degree-one output symbols are requested employing fb_2's. The idea to generate fb_2 is to *smartly* and *greedily* choose and request an input symbol that makes the *largest* progress toward decoding completion.

It is well-known that LT codes have *all-or-nothing* decoding property (also called waterfall phenomenon) [1], where an abrupt jump in the ratio of decoded input symbols occurs at a γ close to $\gamma_{succ} > 1$. Therefore, transmission of fb_2's before $\gamma = 1$ does not considerably contribute to decoding progress. Therefore, we propose to generate fb_2's only when γ surpasses 1. Note that authors in [115] have employed the same idea to determine the start point of their single type of feedback.

To have uniformly distributed fb_2's and to avoid feedback channel congestion, a LT-SF decoder issues a fb_2 on the reception of every $(\ln k)^{th}$ output symbol (starting from k^{th} received output symbols, or equivalently $\gamma = 1$). Therefore, in LT-SF codes feedbacks start with a fb_2 at $\gamma = 1$. It is worth mentioning that we have experimentally found a good distance between two fb_2's equal to $\ln k$; hence, we make no claim about its optimality. However, from our experiments we have observed that this distance should not be far from its optimal value. Further, since the selection of input symbols to request is greedily performed LT-SF codes do not necessarily obtain the optimal decoding performance when feedback channel is available. However, these codes significantly improve the performance of existing LT codes with feedback.

Let us consider a bipartite graph G representing the input and output symbols of LT-SF codes. During data transmission some variable nodes $v_i, i \in \{1, \dots k\}$ are decoded and some check nodes $c_j, j \in \{1, 2, \dots, \gamma k\}$ are reduced to degree zero and are both removed from the decoding graph G. Let us refer to the set of remaining *undecoded variable* nodes by V_{un} and the set of *buffered check* nodes with a degree higher than one by C_{buff}. We remind that the check nodes with degree 1 are called the ripple. Figure 5.2 illustrates such a graph G at a decoder at $\gamma = 1$ for $k = 7$.

It is important to note that the design of fb_2 is to greedily decode as many as possible input symbols so that decoding succeeds at a smaller γ_{succ}. However, as discussed earlier as n increases closer to the end of decoding the average degree of check nodes should be increased to decrease the probability that they become

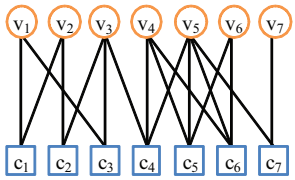


Figure 5.2 The bipartite graph representing the input and the output symbols of a LT-SF code at the buffer of a decoder.

redundant due to earlier recovery of all their neighboring variable nodes. This is the rationale to employ the distribution shifting and fb_1 along with fb_2. In the next sections, we devise three algorithms to analyze the graph G at decoder and greedily select a suitable variable nodes to generate fb_2's.

Generating fb_2 Based on Variable Node with Maximum Degree (VMD)

One insight in choosing a suitable variable node is requesting the variable node $v_i \in V_{un}$ with the *maximum degree*. Such a selection greedily removes the highest number of edges in the first step of decoding after the delivery of the respective input symbol. Based on this idea we propose an algorithm called "*Variable Node with Maximum Degree*" (VMD), where the decoder requests the variable node with the highest degree in its current decoding graph to issue a fb_2. For instance, in Figure 5.2 VMD would choose and request v_5. On the arrival of c_8 containing only v_5, the decoding graph reduces to the state shown in Figure 5.3, where the dashed nodes and edges are removed from graph G. We can see that c_7 is added to ripple, which recovers v_7 in the next decoding iteration. Note that at this step the ripple becomes empty and decoding stalls; hence we have $C_{buff} = \{c_1, c_2, \ldots, c_6\}$ and $V_{un} = \{v_1, v_2, v_3, v_4, v_6\}$. We can see that VMD greedily removes the largest possible number of edges from G and decreases the degree of many check nodes.

Let us investigate the expected maximum degree of input symbols when the first fb_2 is being generated. Since LT-SF decoding does not occur in $\gamma \in (0, 1]$ (due

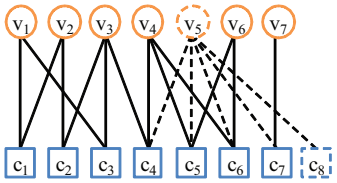


Figure 5.3 The decoding bipartite graph G after the reception of the requested variable node v_5 employing VMD. Dashed nodes and edges have been removed from the decoding graph G.

to lack of degree-one check nodes), the degree of all check nodes at the receiver follows distribution $\Omega_{k,0}(x)$. Thus, at $\gamma = 1$ for the first fb_2 there are on average $e_{tot} = k\Omega'_{k,0}(1)\gamma = \frac{k^2}{k-1}\ln(k)$ edges in the decoding graph G. Since e_{tot} are connected to variable nodes uniformly at random the degree of variable nodes for finite values of k follow binomial distribution with success probability $\frac{1}{k}$.

Let X_i, p_{X_i} and F_{X_i} denote the random variable representing the degree of v_i, *probability mass function* (pmf) of X_i, and the *cumulative density function* (cdf) of X_i, respectively. Clearly, $Pr[X_i = j] = \binom{e_{tot}}{j}\frac{1}{k}^j(1-\frac{1}{k})^{e_{tot}-j}$. Further, let us define the random variable $X = \max(X_i), i \in \{1, 2, \ldots, k\}$ with pmf p_X and cdf F_X, which denotes the degree of the variable node with maximum degree. Since $X_i, \forall i \in \{1, 2, \ldots, k\}$ are i.i.d., we have $F_X = F_{X_i}^k$ [118]; hence p_X and consequently $d_{max} = E[X]$ may be easily obtained from F_X. For instance, we can see that for $k = 10^4$ with IS distribution we have $d_{max} = 17.80$. This analysis of X is suitable when k is small.

For asymptotic setup ($k \to \infty$), the degree of variable nodes which has Binomial distribution can be approximated with Poisson distribution with mean $\lambda = \Omega'_{k,0}(1)\gamma$ [19], i.e., $Pr[X_i = j] = \frac{e^{-\lambda}\lambda^j}{j!}$. Clearly, we have $\lambda = \frac{k}{k-1}\ln(k)$ at $\gamma = 1$. To find the distribution of X (for $X = \max(X_i), i \in \{1, 2, \ldots, k\}$) in this case we employ the results of [119][1]. The following lemma shows the interesting asymptotic behavior of

X.

Lemma 5.3 *At $\gamma = 1$, we asymptotically have $Pr(X \in (I, I+1)) \rightarrow 1$, where I is an integer. In other words, X asymptotically takes the value of one of two consecutive integers I or $I+1$ w.h.p. A close estimate of I within one unit is obtain as follows:*

$$I \approx x_0 + \frac{\ln \lambda - \lambda - \frac{\ln 2\pi}{2} - \frac{3 \ln x_0}{2}}{\ln x_0 - \ln \lambda}, x_0 = \frac{\ln k}{W_1\left(\frac{\ln k}{e\lambda}\right)}, \tag{5.9}$$

where $W_m(.)$ is defined in Lemma 5.2.

Proof. For proof refer to [119]. ■

Therefore, the maximum degree of variable nodes in G can be obtained from LT-SF code's parameters for finite and infinite message lengths.

In regular LT decoding, the number of operations required to decode each check node is equal to the average check node degree [1], i.e., for k input symbols $\mathrm{RS}'(1) = O(\ln k)$. Since the decoding procedure of LT-SF and LT codes are identical the number of operations required to decode each LT-ST check node at $\gamma = 1$ is $O(\frac{k^2}{(k-n)(k-1)} \ln (k-n))$, which is the average degree of LT-SF degree distribution, i.e., $\Omega_{k,0}(1)'$. Therefore, the number of operations required to decode each output symbol increases as n decreases since average degree of check nodes gradually increases in LT-SF codes based on fb_1. Further, for large values of n the number of operations is way higher than $O(\ln k)$ operations required per output symbol for regular LT codes. Due to varying complexity of LT-SF decoding, we postpone the complexity comparison of the overall coding/decoding to numerical simulations. Although VMD seems naïve, we will later see that it greatly improves the performance of LT codes with feedback.

[1] In [119] the distribution of a random variable that is defined as the maximum of several random variables with the same Poisson distribution has been asymptotically studied.

Generating fb_2 Based on Longest Degree-Two Chain (LDC)

Although VMD's complexity is suitably low, it aims for the recovery of as many as possible input symbols by removing the largest number of edges from the decoding graph only in the first step of iterative decoding. However, removing the largest number of edges in the first decoding iteration does not guarantee decoding of the highest number of variable nodes. Therefore, we propose a second algorithm "*Longest Degree-2 Chain*" (LDC), that considers the subsequent decoding iterations as well.

From LT-SF distribution, we observe that at $\gamma = 1$ no decoding can been performed; hence $C_{buff} = \{c_1, c_2, \ldots, c_k\}$ and $V_{un} = \{v_1, v_2, \ldots, v_k\}$. In such a decoding graph, on average more than 50% of the check nodes are of degree-two since $\Omega_{k,0}(2) = \frac{k}{2(k-1)} > 0.5$. Consider a decoding graph G_2, which is formed only by check nodes of degree-two and their respective neighbors, i.e., $G_2 = \{(v_i, c_j) \big| c_j \in C_{buff}, |\mathcal{N}(c_j)| = 2, v_i \in V_{un}, v_i \in \mathcal{N}(c_j)\}$. We call G_2 the decoding graph *induced* by degree-two check nodes.

By investigating the decoding graph G_2 we observe that some check nodes along with $n_v > 1$ variable nodes form structures that the delivery of the *any* of n_v variable nodes results in the decoding of all other $n_v - 1$ variable nodes. We call such a structure decoding *chain* of length n_v. For instance, a single degree 2 check node forms a chain of length $n_v = 2$ since knowing the value of either of its neighboring variable node results in the decoding of the other one. Figure 5.4 shows two chains of length $n_v = 4$ with different structures. The graph G_2 obtained from G in Figure 5.2 has a chain of length $n_v = 3$ including v_1, v_2, and v_3 and a chain of length $n_v = 2$ including v_5 and v_7. We emphasize that this rule only holds for check nodes of degree-two as we discuss further in the next section.

We can see that the degree of variable nodes does not affect the length of chains, and the chains extend as far as the variable nodes are connected to degree-two check nodes. Based on our discussion, the decoder finds all chains of of degree 2 and ran-

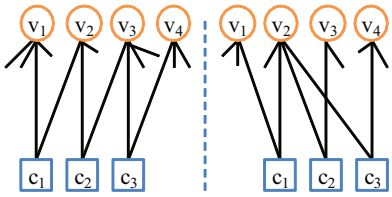


Figure 5.4 Two decoding chains with $n_v = 4$.

domly selects a variable node from the *longest* chain. Next, this variable is requested employing a fb_2.

We are interested in the expected value of n_v for the longest chain for the first fb_2 (at $\gamma = 1$). This is the number of variable nodes recovered in G_2 on the arrival of the requested variable node. If we consider the check nodes in G_2 as *edges* connected to variable nodes as vertices [120], our bipartite graph G_2 would be mapped to a *random graph* including k vertices and an average number of $k\gamma\Omega(2) = \frac{k}{2(k-1)}$ edges (at $\gamma = 1$, $\frac{k}{2(k-1)}$ check nodes of degree two are available). Figure 5.5 shows how this mapping is accomplished for a simple graph.

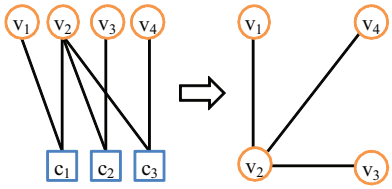


Figure 5.5 Mapping a bipartite graph with degree-two check nodes to a random graph.

Therefore, we may employ the extensively studied properties of random graphs on the size of longest chain n_v. It has been shown that in a random graph with k vertices a *giant component* exist w.h.p. if and only if the average degree of vertices is larger than 1 [120]. A giant component is a connected set with a size linear in the number of graph vertices, i.e., $O(k)$ [120]. On the other hand, if the average vertices degree is less than one the connected vertices in the random graph have the size of $O(\log k)$. As described earlier, the mapping of G_2 to a random graph gives $\frac{k}{2(k-1)}$

edges, hence variable nodes have average degree of $\frac{k}{k-1} > 1$. Therefore, in LDC for the first fb_2 we have $n_v = O(k)$. To give an example, for $k = 10^4$ and $\gamma = 1$ we empirically find $n_v \approx 250$. Hence, on average the first fb_2 generated employing LDC decodes 250 variable nodes out of $k = 10^4$. Later, we will see that LDC has slightly a higher complexity compared to VMD while surpassing its performance.

Generating fb_2 Based on Full Variable Node Decoding (FVD)

LDC is designed considering the graph induced by degree-two check nodes only. However, higher degree check nodes are also present in the decoding graph, which may form more complex decoding chains. When higher degree check nodes are also considered, the main rule of the decoding chains is violated. That is, if recovery of a particular variable node v_i results in the recovery of a set of variable nodes $v_j \in V_i$, the delivery of any of $v_j \in V_i$ does not necessarily guarantee the decoding of v_i. Therefore, no decoding chain can be defined in this case. Consequently, for all $v_i \in V_{un}$ we need to find V_i the set of variable nodes that are decoded as a result of v_i's delivery.

In Figure 5.6, we have illustrated a part of a decoding graph of a LT-SF code. We can observe that the delivery of v_2 results in the decoding of v_1 and v_3, while delivery of v_1 does not decode v_2 and v_3. This is clearly due to considering c_2 that is a degree-three check node in the decoding chain.

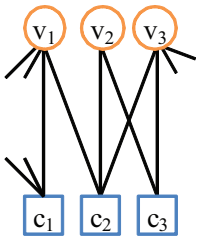


Figure 5.6 Chain of decoding considering check nodes with degrees higher than two. Delivery of v_1 does not necessarily decode v_2 and v_3 while delivery of v_2 results in decoding of v_1 and v_3.

Therefore, we propose "*Full Variable Node Decoding*" (FVD) that considers all check nodes with any degree, and provides the *optimal* selection of variable nodes to issue fb_2's. FVD is performed once when a fb_2 is to be issued as follows.

1. For all $v_i \in V_{un}$ find V_i by running *dummy* decodings.
2. Find $i^* = \text{argmax}_i |V_i|$ and generate a fb_2 containing i^*.

FVD finds the variable node v_{i^*} for fb_2 that results in the *highest* number of decodings considering the full graph G. However, we later see that FVD has a much higher complexity than LDC and VMD.

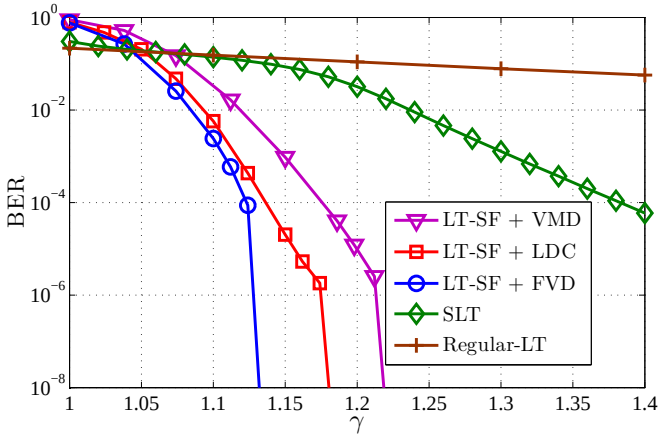
5.3 Performance Evaluation

In this section, we evaluate the performance of LT-SF codes employing numerical simulations. Our results are obtained employing Monte-Carlo method by averaging over the results of at least 10^7 numerical simulations.

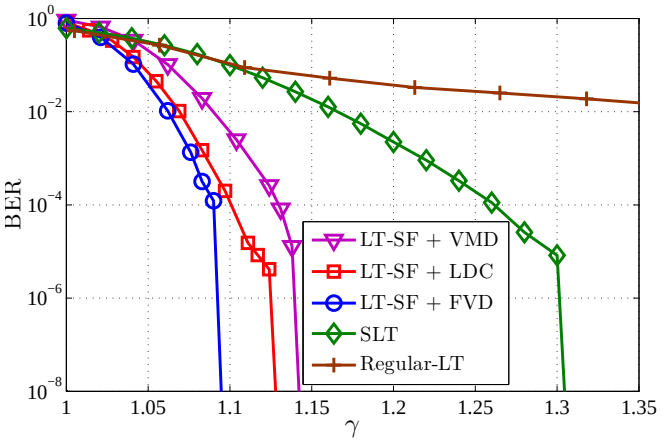
5.3.1 LT-SF Decoding Error Rate and Runtime

Since we are interested to see the performance of LT-SF codes for short data-block lengths we run our performance evaluations for $k = 500$ and $k = 1000$. We plot the decoding *bit-error-rate* (BER) (average ratio of unrecovered input symbols to total number of input symbols $1 - E[\frac{n}{k}]$) and the *ratio of successful decodings* versus received overhead γ in Figures 5.7 and 5.8, respectively. Note that we set $c = 0.9$ and $\delta = 0.1$ for SLT codes as proposed in [113].

Figures 5.7 and 5.8 show that LT-SF codes significantly surpass SLT codes. We can see that the required coding overhead γ_{succ} (to achieve BER$\leq 10^{-8}$) for $k = 1000$ has decreased by 0.223, 0.189, and 0.175 when LT-SF decoder employs FVD, LDC, and VMD algorithms, respectively. This is respectively equivalent to **69.7%**, **59.1%**,



(a) Performance comparison for $k = 500$.

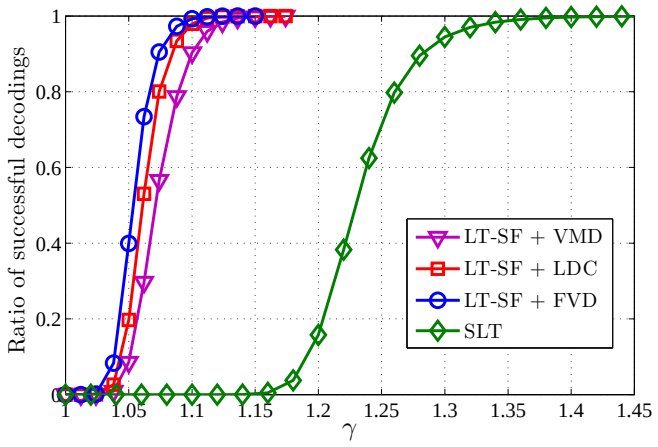


(b) Performance comparison for $k = 1000$.

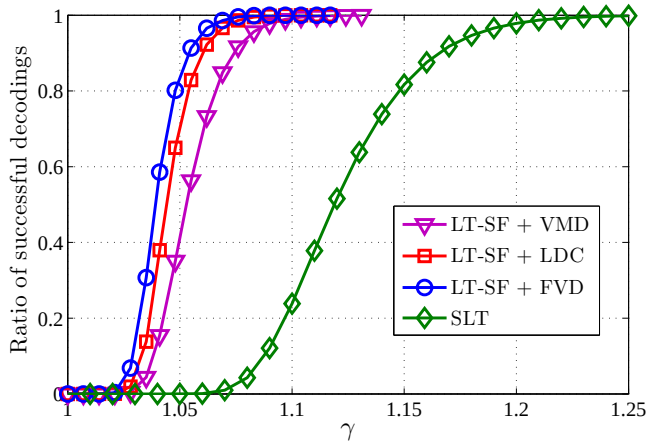
Figure 5.7 The BER of SLT codes and various setups of LT-SF codes versus received overhead γ for $k = 500$ and $k = 1000$.

and **54.7%** reduction in the number of required redundant output symbols (codings overhead) for full decoding compared to SLT codes. More interestingly, we can see that for SLT codes with $k = 500$ full decoding is not obtained even for $\gamma = 1.4$, while LT-SF codes can obtain a full decoding at much smaller γ's.

It is worth mentioning that the straight line at the end of BER curves shows that we did not observe any decoding error at these overheads in 10^7 iterations of numerical



(a) Performance comparison for $K = 500$.



(b) Performance comparison for $K = 1000$.

Figure 5.8 The ratio of successful decodings for SLT codes and various setups of LT-SF codes versus received overhead γ .

simulations and all decodings were successful. Therefore, the error rates are indeed less than 10^{-8}. Next, we have summarized the runtime comparison of LT-SF codes employing FVD, LDC, and VMD with SLT codes in Table 5.1.

Table 5.1 shows that the complexity of FVD is way higher than the other two proposed algorithms. However, we can see that LDC and VMD have close complexities. Therefore, LDC may be the *best* option that provides a low complexity besides

Table 5.1 Runtime comparison of LT-SF and SLT codes on the same platform in seconds.

Algorithm	$N = 500$	$N = 1000$
LT-SF+VMD	0.122	0.683
LT-SF+LDC	0.181	1.050
LT-SF+FVD	11.000	122.270
SLT	0.535	1.550

improved coding performance. Further, we can see that LT-SF codes employing VMD and LDC have lower complexities compared to SLT codes. The reason for this lower complexity is that in LT-SF codes for $\gamma < 1$ no decoding is performed and no feedback is generated, and when the decoding starts the full recovery is obtained at a smaller γ resulting in less number of decoding iterations. Therefore, LT-SF codes employing VMD and LDC outperform SLT codes both in the number of required output symbols and complexity.

5.3.2 Number of Feedbacks

In this section, we compare the total number of feedbacks issued by LT-SF codes and compare it to that of SLT codes for $k = 500$ and $k = 1000$. We emphasize that other proposed LT codes with feedback cannot achieve the performance of SLT and LT-SF codes. The expected number of feedbacks for LT-SF and SLT codes are summarized in Table 5.2 for $k = 500$ and $k = 1000$. From Table 5.2, we can interestingly observe that not only LT-SF codes decrease the required coding overhead for a successful decoding γ_{succ}, but also they need slightly smaller number of feedbacks compared to SLT codes.

Table 5.2 The average number of feedbacks issued in LT-SF and SLT codes for full decoding of data block.

Algorithm	$N = 500$			$N = 1000$		
	fb_1	fb_2	total	fb_1	fb_2	total
LT-SF+VMD	2.68	7.37	10.05	2.68	9.29	11.97
LT-SF+LDC	3.15	6.49	9.64	3.90	8.00	11.90
LT-SF+FVD	2.75	6.01	8.76	3.58	6.92	10.5
SLT	-	-	10.43	-	-	12.27

5.3.3 Robustness to Erasure in Feedback Channel

We mentioned that LT-SF codes are designed to be resilient to loss in feedback channel in contrast to all existing work [102, 113–116], and their decoding recovery rate does not considerably deteriorate for $\varepsilon_{fb} \in [0, 1)$. We evaluate the effect of feedback loss on the performance of LT-SF codes and SLT codes. Assume that the loss rate of the feedback channel is $\varepsilon_{fb} = 0.9$ (which is not known to encoder and decoder), hence 90% of the feedbacks are lost in transmission. Note, that in a lossy forward channel the degree-one acknowledgements may also be dropped while fb_1 or fb_2 may have already been delivered. In the case of fb_2 loss, the retransmission compensates this loss. However, in case of fb_1 loss, the encoder shifts the degree distribution accordingly while the decoder remains unaware of this shift. In this case, feedback retransmission is not even required since the degree distribution shift has already occurred. Therefore, we consider the worst case in our simulations and assume that if an acknowledgement is lost the distribution shifting does not occur as well. Figure

5.9 shows the performance of LT-SF codes and SLT codes for $k = 1000$ and $\varepsilon_{fb} = 0.9$.

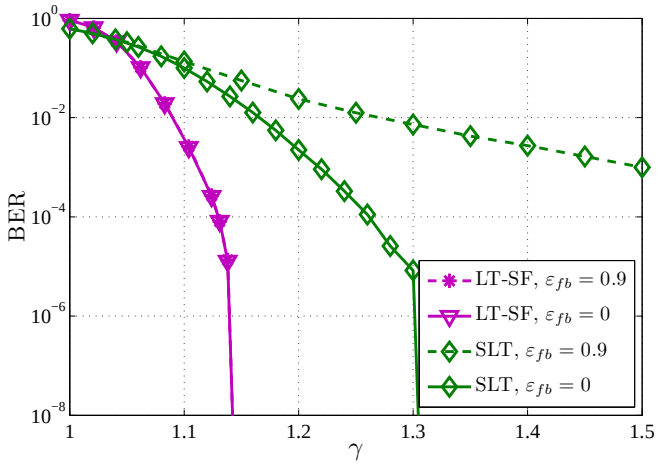


Figure 5.9 Effect of 90% feedback loss on the performance of SLT and LT-SF codes employing VMD.

Figure 5.9 shows the excellent resilience of LT-SF codes to feedback loss in contrast to SLT codes. In practice, the performance of SLT codes approach that of regular LT codes as the feedback loss ratio increases. To the best of our knowledge robustness against feedback loss had not been considered in any existing work and this significantly distinguishes LT-SF codes.

5.4 Conclusion

In this chapter, we proposed LT-SF codes that are LT codes with *smart* feedback, which alleviate the low performance of LT codes for short data-block lengths. We proposed to employ two types of feedbacks according to the status and needs of the decoder. In LT-SF codes, the decoder may inform the encoder with the total number of decoded input symbols by the first type of feedback or request a certain input symbol from the encoder employing second type of feedback. We designed three algorithms for LT-SF codes that described how to analyze the decoder's buffer and request a suitable input symbol. In addition, employing a novel idea we made LT-SF

code resilient against high loss rates in the feedback channel. We analyzed LT-SF codes and discussed its advantages.

We showed that our contribution in the design of LT-SF codes compared to existing work is fourfold. LT-SF codes reduce the coding overhead for a successful decoding and decrease the total number of feedbacks. Further, we observed that overall runtime required for a complete LT-SF decoding is lower than that of existing work. Finally and most importantly, LT-SF codes' performance does not considerably degrade at large loss rates in the feedback channel.

CHAPTER 6

UNEQUAL ERROR PROTECTION RATELESS CODING IN VIDEO TRANSMISSION

So far, we have investigated various aspects of rateless codes and their advantages. In this Chapter, we demonstrate how UEP-rateless codes can be employed to increase video transmission efficiency compared to the case where conventional EEP-rateless codes are employed. First, we employ UEP-rateless codes to provide more protection for more important frames in a video stream, namely I- and P-frames. This increases the received video quality or equivalently decreases the amount of transmitted data to reach a certain video quality. Next, we utilize UEP-rateless codes to design a novel *periodic broadcasting video-on-demand* protocol with reduced startup delay. We discuss the advantages of our proposed algorithms and evaluate their performances employing numerical simulations.

6.1 UEP-Rateless Codes in MPEG Video Transmission

In this section, we propose a coding scheme that employs UEP-rateless codes to provide more protection for video frames with higher influence on the quality of the displayed video. Previously, several work have addressed this problem. Authors in [121–124] propose to employ different *Reed-Solomon* codes [125] to separately encode each frame/layer of the video according to its importance. By assigning a larger coding overhead to more important video frames/layers they have shown that a higher video quality can be achieved. However, since these algorithms have employed fixed-

rate Reed-Solomon codes, the transmitter needs to have an estimate of the channel erasure rate to set the appropriate coding rates to obtain an efficient video transmission scheme. This information is not always available at transmitter. Further, due to high complexity of Reed-Solomon codes implementation of these algorithms may not be feasible in applications with constrained resources.

Authors in [126] propose to have a higher protection on GOP header and *motion vectors* instead of I- and P-frames employing *LDPC codes* [127]. Similar to Reed-Solomon codes, LDPC codes impose a fixed coding rate, which may not be of interest in some applications. Authors in [128] propose an MPEG video transmission scheme which provides more protection for I-frames only by transmitting multiple copies of I-frames instead of using UEP codes. This video transmission scheme may be suboptimal since a large amount of redundant packets are transmitted from I-frames and the higher importance of P-frames compared to B-frames is not considered.

In contrast to previous studies, our proposed algorithm employs UEP-rateless codes. Thus it does not need to have any knowledge about the channel's erasure rate. Further, we propose to encode all frames of one GOP with a single UEP-rateless code instead of multiple EEP codes. This idea considerably reduces the coding/decoding overhead and complexity.

6.1.1 Proposed MPEG Video Coding Using UEP-Rateless Codes

To increase the video transmission efficiency, i.e., increasing Q and consequently the PSNR of the video, we propose to protect frames *unequally* according to their *importance* employing UEP-rateless codes. We encode each GOP by applying one independent UEP-rateless code over the entire GOP. Therefore, the number of input symbols k denotes the total number of symbols of all frames in one GOP.

There is one I-frame in each GOP with the highest level of importance. Let the importance level of an I-frame be $k_I = p_I k$, where p_I shows the probability of

choosing a source symbol from the I-frame to generate an outgoing encoded symbol. Furthermore, we have several B-frames with equal and the lowest level of importance among all frame types. Let us show the importance level of B-frames by $k_B = p_B k$. Finally, according to the length of the GOP, there are $n_P = \frac{N}{M} - 1$ P-frames in each GOP. We denote different P-frames with P_j, $j \in \{1, 2, \ldots, n_P\}$ and their importance levels by $k_{P_j} = p_{P_j} k$.

For the sake of simplicity in illustration, we assume that the I-frame is situated at the beginning of the GOP followed by series of all P-frames, and afterwards B-frames are transmitted. Let the series $\alpha_I k$, $\alpha_{P_1} k$, $\alpha_{P_2} k$, ..., $\alpha_{P_{n_P}} k$, $\alpha_B k$, where $\alpha_I + \alpha_B + \sum_{i=1}^{n_P} \alpha_{P_i} = 1$, denote the segmentation sizes as shown in Figure 6.1. The corresponding source symbol selection probabilities for different frames are also depicted in Figure 6.1. Naturally, I-frames have the largest size due to their lowest compression level and independency, and B-frames have the smallest frame size.

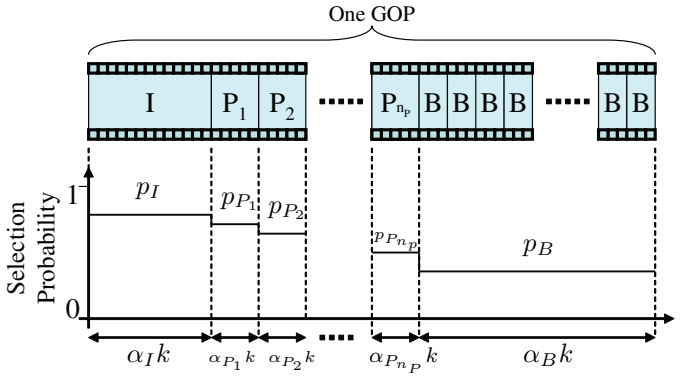


Figure 6.1 Input symbol selection probabilities for various sections of a GOP and partitions with unequal importance and their relative sizes.

The UEP-rateless coding is performed with various protection levels on each GOP. The rateless encoded symbols are transmitted over a lossy channel and are recovered at the receiver side with different decoding error rates. Let ϵ_I, ϵ_B, and ϵ_{P_j}, $j \in \{1, 2, \ldots, n_P\}$ be the decoding symbol receovry rates of I-, B-, and P_j-frames, respectively. ϵ_I, ϵ_B, and ϵ_{P_j} can be found from (2.6) by setting the UEP-rateless coding

parameters to $\alpha_i = \{\alpha_I, \alpha_{P_1}, \alpha_{P_2}, \ldots, \alpha_{P_{n_P}}, \alpha_B\}$ and $p_i = \{p_I, p_{P_1}, p_{P_2}, \ldots, p_{P_{n_P}}, p_B\}$. Further, let c_I, c_P, and c_B represent the number of symbols in each type of frame.

To find the analytical expression for Q given by (2.9), we formulate N_{dec_I}, N_{dec_P} , and N_{dec_B} as follows. The I-frame can be decoded independently if fraction ν of c_I symbols are delivered. The number of delivered symbols to destination has binomial distribution with success probability $1-\epsilon_I$. Therefore the probability that an I-frame is decodable is

$$\lambda_I = \sum_{j=\nu c_I}^{c_I} \binom{c_I}{j} (1-\epsilon_I)^j (\epsilon_I)^{c_I - j}.$$

Therefore, the expected number of decodable I-frames for all GOPs is given by

$$N_{dec_I} = \lambda_I \tau, \tag{6.1}$$

where τ is the number of GOPs in the video. Therefore, we have $N_{total} = \tau N$.

The P-frames are decoded if the preceding I- and P-frames are recovered and a portion ν out of c_P symbols belonging to the corresponding P-frames are received with no defects. Each P-frame has a different decoding error rate, ϵ_{P_j}, $j \in \{1, 2, \ldots, n_P\}$, according to its protection level, and a different probability of successful decoding, λ_{P_j}, given by

$$\begin{aligned}
\lambda_{P_1} &= \lambda_I \sum_{j=\nu c_P}^{c_P} \binom{c_P}{j} (1-\epsilon_{P_1})^j (\epsilon_{P_1})^{c_P - j}, \\
\lambda_{P_2} &= \lambda_{P_1} \sum_{j=\nu c_P}^{c_P} \binom{c_P}{j} (1-\epsilon_{P_2})^j (\epsilon_{P_2})^{c_P - j}, \\
&\vdots \\
\lambda_{P_{n_P}} &= \lambda_{P_{n_P-1}} \sum_{j=\nu c_P}^{c_P} \binom{c_P}{j} (1-\epsilon_{P_{n_P}})^j (\epsilon_{P_{n_P}})^{c_P - j},
\end{aligned}$$

Consequently, the expected number of decodable P-frames is given by

$$\begin{aligned}
N_{dec_P} &= \tau \sum_{q=1}^{n_P} \lambda_{P_q} \\
&= \tau \lambda_I \sum_{q=1}^{n_P} \prod_{l=1}^{q} \sum_{j=\nu c_P}^{c_P} \binom{c_P}{j} (1-\epsilon_{P_l})^j (\epsilon_{P_l})^{c_P - j}.
\end{aligned} \tag{6.2}$$

Finally, each $(M-1)$ B-frames, enveloped between two consecutive P-frames, have the same probability of successful decoding. Let B_j denote the j^{th} group of B-frames, which includes $(M-1)$ B-frames. It can be easily seen that based on the structure of a GOP, there are n_P+1 groups of $(M-1)$ consecutive B-frames in total. A B-frame in any group of B-frames (except the last group) are decodable if the preceding P- and I-frames are decodable and all symbols belonging to this specific B-frame are correctly received. Any B-frame belonging to the last group of B-frames can be decoded if the I-frame of the succeeding GOP is also decodable. Consequently, each group of $(M-1)$ B-frames has the same decoding probability given by

$$\begin{aligned}
\lambda_{B_1} &= \lambda_{P_1} \sum_{j=\nu c_B}^{c_B} \binom{c_B}{j} (1-\epsilon_B)^j (\epsilon_B)^{c_B}, \\
\lambda_{B_2} &= \lambda_{P_2} \sum_{j=\nu c_B}^{c_B} \binom{c_B}{j} (1-\epsilon_B)^j (\epsilon_B)^{c_B}, \\
&\vdots \\
\lambda_{B_{n_P}} &= \lambda_{P_{n_P}} \sum_{j=\nu c_B}^{c_B} \binom{c_B}{j} (1-\epsilon_B)^j (\epsilon_B)^{c_B}, \\
\lambda_{B_{n_P+1}} &= \lambda_I \lambda_{B_{n_P}}
\end{aligned}$$

Consequently, the expected number of decodable B-frames is given by

$$N_{dec_B} = \tau(M-1) \sum_{j=1}^{n_P+1} \lambda_{B_j} \tag{6.3}$$

Now, we substitute (6.1), (6.2), and (6.3) into (2.9) to find the final expression for the decoding error rates with UEP-rateless video coding. The values assigned to protection levels affect the decoding error rates and consequently change the resulting Q. We can find the maximum Q by optimizing the values assigned to protection levels. Since the number of P-frames and their protection levels change by varying the length of the GOP, the optimum protection level values depend on values of M and N.

Similar to UEP-rateless coding, Q is given by (2.9) for EEP-rateless coding. However, when EEP-rateless codes are employed instead of UEP-rateless codes, all frames in a GOP are protected equally. Thus they are recovered with equal error rate at

decoder, i.e., $\epsilon_I = \epsilon_B = \epsilon_{P_1} = \ldots = \epsilon_{P_{n_P}}$, which boils (6.1), (6.2), and (6.3) down to simpler expressions.

Here, we provide two optimization examples for a GOP with $M = 3$, $N = 15$ for two cases of MPEG-I and MPEG-II. This GOP format is a common GOP size used in practice and is shown by IBBPBBPBBPBBPBB. Table 6.1 summarizes the *average* frame sizes and the number of symbols in each frame type, i.e. c_I, c_P, and c_B, when each transmitted symbol conveys $0.5KB$ of data.

Table 6.1 Frame sizes and the number of symbols in each frame type for typical MPEG-I and MPEG-II video streams

	I	P	B	c_I	c_P	c_B
MPEG-I	$75KB$	$25KB$	$10KB$	150	50	20
MPEG-II	$200KB$	$100KB$	$40KB$	400	200	80

Based on the frame sizes provided in Table 6.1, we find the values of $\{\alpha_I, \alpha_{P_1}, \alpha_{P_2}, \alpha_{P_3}, \alpha_{P_4}, \alpha_B\}$ as reported in Table 6.2.

Table 6.2 The sizes of each importance portion for MPEG-I and MPEG-II video streams

	α_I	α_{P_1}	α_{P_2}	α_{P_3}	α_{P_4}	α_{P_B}
MPEG-I	0.2727	0.0909	0.0909	0.0909	0.0909	0.3637
MPEG-II	0.2	0.1	0.1	0.1	0.1	0.4

A GOP with $M = 3$ and $N = 15$ has six protection levels shown by k_I, k_B, k_{P_1}, k_{P_2}, k_{P_3}, and k_{P_4}. According to [19, 20] we have $k_I\alpha_I + k_B\alpha_B + \sum_{j=1}^{4} k_{P_j}\alpha_{P_j} = 1$, which shows that one of the importance level values is dependant to other protection levels;

therefore, we have only five independent protection levels to optimize. We remind that our objective function is Q, which we try to maximize by finding optimum protection levels. For the sake of simplicity in our simulations, we consider $\nu = 1$, and we employ the rateless coding degree distribution $\Omega_{shok}(x)$ given by (2.3) throughout the simulations.

By searching the whole decision space, we find the global optimum values of the protection levels as reported in Table 6.3 for MPEG-I and in Table 6.4 for MPEG-II for different values of received overhead, γ. As we expected and it is confirmed by the optimization results given in Tables 6.3 and 6.4, the highest protection levels are assigned to I-frames, and the lowest protection levels are assigned to B-frames. P-frames protection levels, which are larger than B-frames protection levels and lower than I-frames protection levels, are set to decreasing values according to their position in the GOP. Based on the desired final decoding error rate, one should choose appropriate overhead and the corresponding optimum protection level values from Tables 6.3 and 6.4 to acquire the highest performance. In the next section, we evaluate our proposed scheme's performance.

6.1.2 Performance Evaluation

We set the protection levels to the optimum values given in Tables 6.3 and 6.4 for $\gamma = 1.2$ and find Q based on the derived formulas in this section as our performance metric. We compare the performance of video coding employing UEP-rateless codes with the case where video is coded using conventional rateless codes with EEP property. Simulation results are shown in Figures 6.2(a) and 6.2(b) for MPEG-I and MPEG-II, respectively

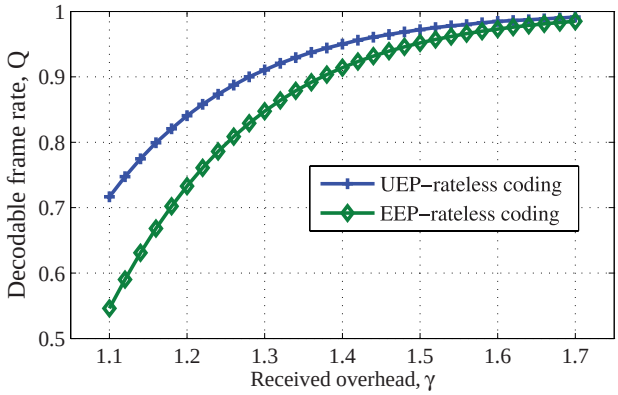
Figure 6.2 can be described in two ways. First, let us assume we have a fixed amount of data and channel bandwidth. We can see that by using UEP-rateless codes there is an increase in the number of decoded frames at the receiver, or equivalently,

Table 6.3 Optimum values of MPEG-I video stream protection levels for different values of γ.

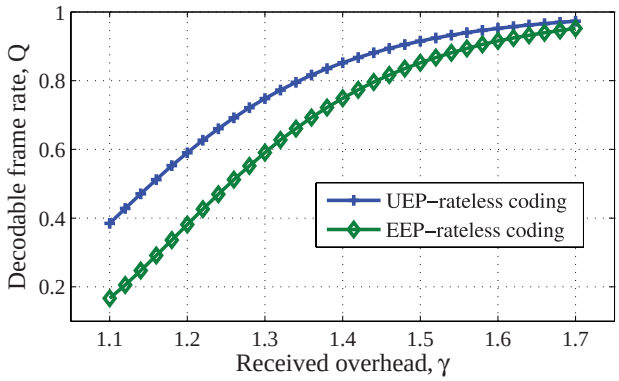
γ	k_I	k_{P_1}	k_{P_2}	k_{P_3}	k_{P_4}	k_B
1.1	1.2	1.17	1.13	1.08	1.01	0.75
1.2	1.18	1.15	1.12	1.07	1.01	0.77
1.3	1.17	1.14	1.11	1.07	1	0.79
1.4	1.15	1.13	1.1	1.06	1.01	0.81
1.5	1.14	1.12	1.1	1.06	1.01	0.82
1.6	1.14	1.11	1.09	1.05	1	0.83

Table 6.4 Optimum values of MPEG-II video stream protection levels for different values of γ.

γ	k_I	k_{P_1}	k_{P_2}	k_{P_3}	k_{P_4}	k_B
1.1	1.24	1.21	1.16	1.09	0.99	0.76
1.2	1.21	1.18	1.14	1.09	1.01	0.79
1.3	1.19	1.16	1.13	1.08	1.01	0.81
1.4	1.17	1.15	1.12	1.08	1.02	0.82
1.5	1.16	1.14	1.11	1.07	1.02	0.83
1.6	1.15	1.13	1.1	1.07	1.02	0.84



(a) Q for MPEG-I video stream with UEP- and EEP-rateless coding.



(b) Q for MPEG-II video stream with UEP- and EEP-rateless coding.

Figure 6.2 Decodable frame rate Q for MPEG-I and MPEG-II video streams employing proposed coding scheme.

the receiver will perceive the video with a higher PSNR. For instance, at $\gamma = 1.3$, for MPEG-II video the number of decoded frames increases from 58% to 74% employing UEP-rateless codes. Second, for a fixed decodable frame rate Q, UEP-rateless coding requires a smaller overhead, γ , than EEP-rateless coding. For example, instead of using EEP-rateless codes with overhead $\gamma = 1.37$ for 90% frame recovery in MPEG-I video, we can encode the video with UEP-rateless codes and transmit only an

overhead of $\gamma = 1.28$.

It should also be noted that since the frames of MPEG-II video have larger sizes compared to frames of MPEG-I video, they have higher probabilities of being dropped in transmission. This is the rationale behind the lower Q for MPEG-II video compared to MPEG-I video. Besides gaining an efficient transmission scheme, an important advantage of our proposed protocol is having much lower complexity compared to previous coding schemes. Rateless codes [2] have linear time coding and decoding complexity of $O(k)$, while Reed-Solomon codes [125] as employed in [121], have coding complexity of $O(k^2)$. As a result, besides having higher efficiency, our proposed scheme suits wireless applications where the computational power is limited.

Furthermore, as mentioned earlier fixed-rate codes such as Reed-Solomon codes cannot adapt to varying loss rates. Hence, they may not be employed on wireless links, which have dynamic characteristics. However, rateless codes are universal erasure channel error-correction codes that can adapt to any loss rate and can still exhibit low coding/decoding complexity. Moreover, rateless codes are the perfect coding choice for multicast video content delivery over wireless channels, since the number of output symbols can be limitless in contrast to fixed-rate codes.

Our results so far have been based on asymptotic formulas for decoding of UEP-rateless codes [19, 20], i.e., Equation (2.6), which assumes that k is large. However, in practice k is limited. Consequently, to evaluate the performance of our proposed scheme when k is finite, we run real EEP- and UEP-rateless coding/decoding simulations. Figure 6.3 shows the resulting values of Q for an MPEG-II video stream with the parameters from Tables 6.1 and 6.2, and with GOPs with $N = 15$ and $M = 3$. Note that according to the number of symbols in different frame types, each GOP would have $k = 2000$ source symbols in total.

Figure 6.3 shows and confirms that Q has increased considerably employing UEP-rateless codes for finite number of source symbols.

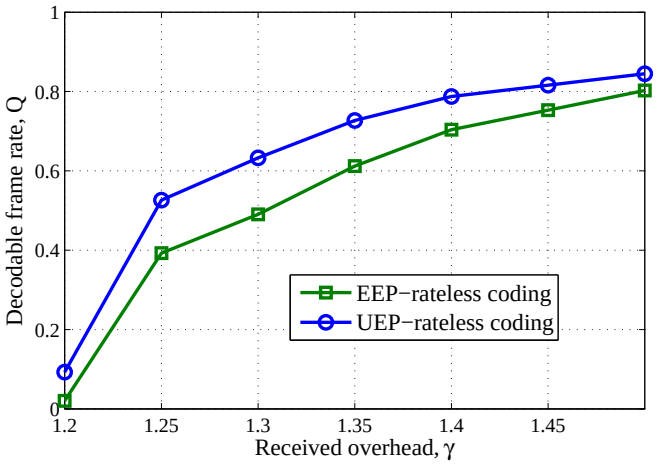


Figure 6.3 Comparison of Q for an MPEG-II video stream with EEP- and UEP-rateless coding for finite length, $k = 2000$.

6.2 UEP-Rateless Codes for Video-On-Demand

In this section, we propose a new efficient VOD broadcasting protocol employing UEP-rateless codes. In this section, we propose a new VOD periodic broadcasting protocol that belongs to the first category of VOD schemes (see Section 2.5). We exploit the URT property provided by UEP-rateless codes [19, 20], to design a periodic broadcasting protocol with reduced startup delay compare to [48–50].

6.2.1 VOD Protocol Design Using UEP-Rateless Codes

In periodic broadcasting protocols, a video with *playout* rate B_0 is first divided into several segments S_j of increasing length and each segment is broadcast on a separate channel with transmission bandwidth B as depicted in Figure 6.4. In Figure 6.4, the upper shape demonstrates the video stream that is *displayed* to client. Further, w is the startup delay incurred due to time required to buffer the first segment. We can see that the video has been partitioned into three segments of increasing size, and each segment is repetitively broadcast on a separate channel.

As can be seen from Figure 6.4, it is necessary to completely deliver each segment

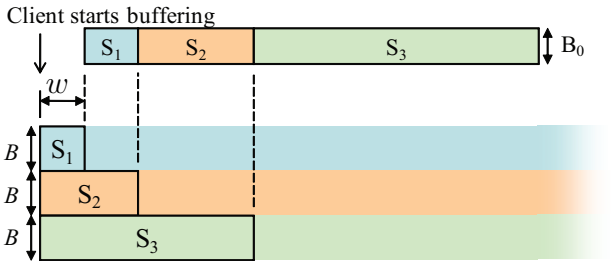


Figure 6.4 Periodic VOD protocols segmentation. The upper video stream shows the actual displayed video timing for client.

to the client before its *playout* time so that the video is shown continuously with no interruptions. Based on this constraint and the type of coding we determine each segment's size. Without loss of generality, we consider the first segment of the partitioned video S_1 as the reference segment with playout duration of one unit time. Therefore, if this segment is transmitted without encoding its download time would be $\frac{B_0}{B}$ unit times. Clearly, if the transmission bandwidth is equal to video playout bandwidth, i.e., $B = B_0$, the time required to receive the first segment is one unit time. However, when the reference segment is encoded employing a rateless code, we also need to consider the coding overhead, γ, in download time.

First, consider the case that the reference segment is encoded with an EEP-rateless code. The download time of this segment is $\gamma_{EEP}\frac{B_0}{B}$ unit times, where γ_{EEP} is the overhead at which the frame loss rate of the decoded segment reduces to 10^{-3}[1].

.

In our proposed protocol, we partition the reference segment into two parts with the fraction sizes α and $1-\alpha$, and protect the first part with a higher priority, k_M, and the rest of the segment with a lower priority $k_L < k_M$ employing a UEP-rateless code [19, 20]. The first part (MIS) acquires the error rate of 10^{-3} at γ_{MIS}, and is

[1]In this section, we assume that video frames are encoded independently as coding symbols. Further, it has been shown that at *frame loss rate* of 10^{-3} clients do not perceive any degradation in the video quality [129].

displayed to the client, while more encoded symbols from the same segment are being received. Meanwhile, the second part (LIS) is also recovered at γ_{LIS}. Other segments are similarly divided into two parts with relative sizes α and $1-\alpha$ as shown in Figure 6.5.

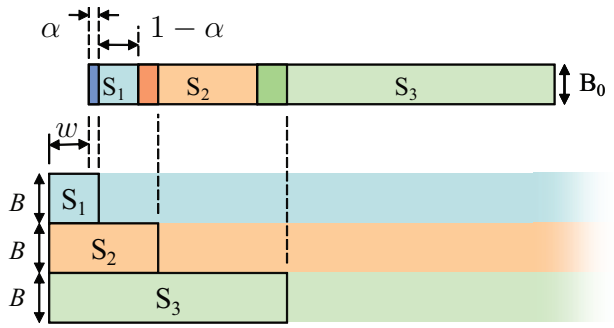


Figure 6.5 Proposed VOD protocol segmentation.

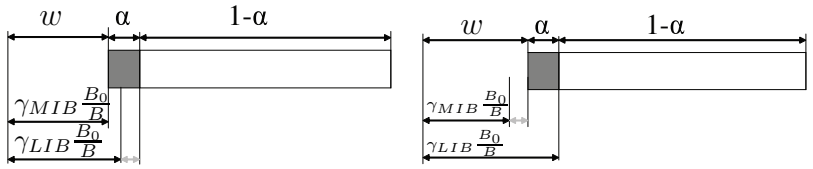
In Figure 6.5, we can see that the first segment can start being displayed to client before S_1 is completely delivered. This actually decreases the start up delay compared to the simple case displayed in Figure 6.4. With this setup, besides the on-time recovery of the whole segment, the on-time recovery constraint of the first and the second part of the segment is added to the requirements. Therefore, we need to choose the UEP parameters k_M, k_L, α, and segment sizes to meet these timing constraints.

The startup delay of the first segment with unit time duration is determined based on its first and second part's recovery times, i.e., $\gamma_{MIS}\frac{B_0}{B}$ and $\gamma_{LIS}\frac{B_0}{B}$. According to the length of these two time durations, two cases may occur. First, when the beginning part's recovery time plus the playout time of this portion is greater than the recovery time of the second part as depicted in Figure 6.6(a). In this case, the startup delay is dominated and determined by the first part's recovery time. The video starts playing upon recovery of the first portion. The second part is recovered on-time, during the first part's playout, and before the time it needs to be displayed.

In the second case, the second part is not recovered during the first segment's

playout, thus it will not be ready on-time as shown in Figure 6.6(b). In this case we need to extend the startup delay so that the second part is recovered on-time and the video can be played without any interruptions. This is equal to decoding and storing the first portion in a buffer, and starting the video display when we are confident about second part's on-time recovery. In this case, the startup delay is controlled by the second part's recovery time.

We remind that γ_{MIS} and γ_{LIS} are determined by plugging the values of k_M, k_L, and α into (2.7) and (2.8), and finding the values of γ_{MIS} and γ_{LIS} for which decoding error rates of MIS and LIS (y_M and y_L) are equal to 10^{-3}.



(a) First part (MIS) recovery-time dominates.

(b) Second part (LIS) recovery-time dominates.

Figure 6.6 First segment and startup delay, w, which is a function of γ_{MIS}, γ_{LIS}, B_0, B, and α.

According to Figure 6.6 and the discussion provided, we can formulate the startup delay of each segment in terms of the recovery overheads of its partitions as

$$w = \max(\gamma_{MIS}\frac{B_0}{B}, \gamma_{LIS}\frac{B_0}{B} - \alpha). \tag{6.4}$$

For the EEP-rateless coding case, α would be equal to zero and (6.4) reduces to

$$w = \gamma_{EEP}\frac{B_0}{B}. \tag{6.5}$$

Our goal is to choose k_M and α in (2.7) and (2.8) such that the startup delay of the VOD protocol w given by (6.4) is minimized. Figure 6.7 depicts w for different

values of k_M and α when $B = 3B_0$. As is shown, w has a global minimum value at $k_M = 1.56$ and $\alpha = 0.1$. Table 6.5 summarizes the optimal values of k_M, α, and the corresponding γ_{MIS}, γ_{LIS}, and minimized startup delays for $B = 2B_0$ to $B = 5B_0$. We use these optimum values in our simulations and protocol design.

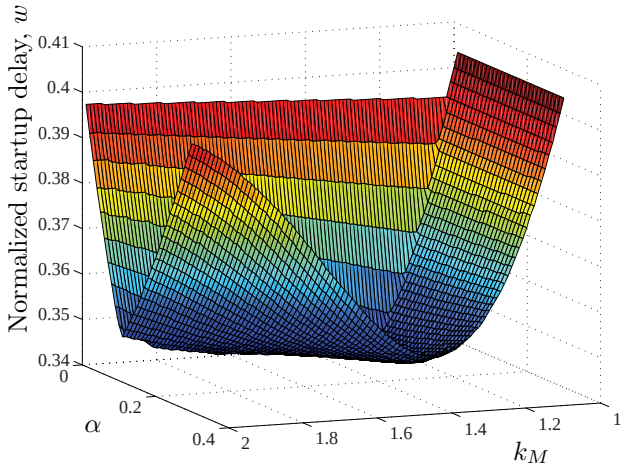


Figure 6.7 Normalize startup delay, w, vs. k_M and α for $B = 3B_0$.

Equivalently, the normalized startup delays for $B = 2B_0$ to $B = 5B_0$, with EEP-rateless coding are given in Table 6.6. Note that for all the cases we have $\gamma_{EEP} = 1.211$.

Now, we can formulate the sizes of the segments in our proposed VOD protocol. We determine the size of segment S_2 considering the criteria that the summation of the startup delay, w, and the duration of the first segment playout time (one unit time), is the *startup delay* for the second segment. The division of this time duration to w, gives the second segment's length as multiples of the first segment duration. This rule applies to other segments as well. The general segmentation formula is given by

$$S_i = \begin{cases} 1 & i = 1, \\ \frac{w + \sum_{j=1}^{i-1} S_j}{w} & i \geq 2, \end{cases} \tag{6.6}$$

Table 6.5 Optimum values of k_M, α, and the corresponding γ_{MIS}, γ_{LIS}, and minimized startup delays, w, with UEP-rateless coding.

B	$2B_0$	$3B_0$	$4B_0$	$5B_0$
α	0.14	0.10	0.06	0.05
k_M	1.43	1.56	1.75	1.83
γ_{MIS}	1.032	1.031	1.030	1.030
γ_{LIS}	1.296	1.287	1.270	1.264
w	0.516	0.343	0.257	0.206

Table 6.6 Normalized startup delay, w, with EEP-rateless coding.

B	$2B_0$	$3B_0$	$4B_0$	$5B_0$
w	0.605	0.403	0.302	0.242

where S_i is the normalized i^{th} segment size. By equating $\sum_i S_i$ to the total duration of the movie the duration of each segment can be determined. For UEP-rateless and EEP-rateless coding, w is given by (6.4) and (6.5), respectively. The same two-priority-level UEP-rateless code is applied on all segments, therefore, all segments include a portion, which is recovered in advance and is displayed to the client before the second part of the segment is recovered.

Note that the number of segments N_S is chosen according to the system requirements and the available system resources. Although a larger N_S results in a shorter startup delay, it requires a higher total video broadcast bandwidth, $B \times N_S$, and also causes heavier computational complexity.

6.2.2 Modified Low-Bandwidth Protocol

In our proposed VOD protocol, we have assumed that server and clients have the same bandwidth equal to $B \times N_S$. However, in some cases clients may have limited amount of bandwidth due to the technical limitations or the bandwidth cost. Our proposed protocol can be easily modified such that each client in the worst case has to receive only *two* segments in parallel. This reduces clients' required bandwidth from $B \times N_S$ to $2B$. However, this leads to a slight increase in the startup delay due to new video segmentation. We still encode the segments with the same UEP-rateless code, and compare the resulting startup delay with EEP-rateless encoding. In Figure 6.8, the segmentation of the modified protocol for lower client bandwidth is illustrated.

The startup delay is still given by (6.4) and (6.5), and the segmentation size of the modified version of the protocol, which satisfies a continuous playout, is given by

$$S_i = \begin{cases} 1 & i = 1, \\ \frac{w+1}{w} & i = 2, \\ \frac{S_{i-1}+S_{i-2}}{w} & i \geq 3. \end{cases} \tag{6.7}$$

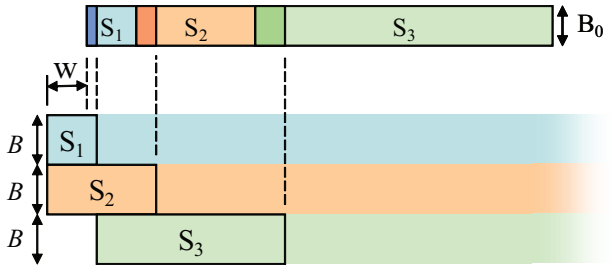


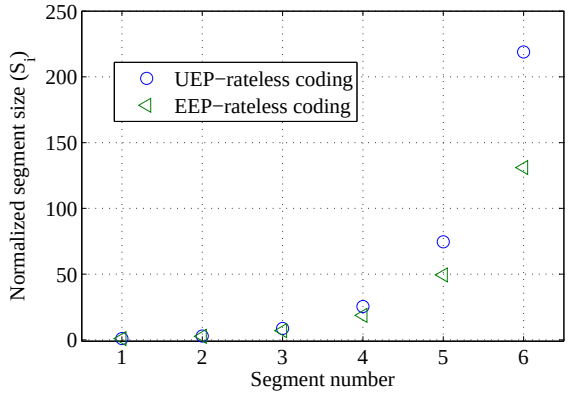
Figure 6.8 Segmentation of our modified protocol. Users only need to receive from two channels in parallel at any time instant.

The segmentation sizes for both proposed protocol schemes, using optimum values from Table 6.5 and 6.6, with EEP- and UEP-rateless encoding, is depicted in Figure 6.9.

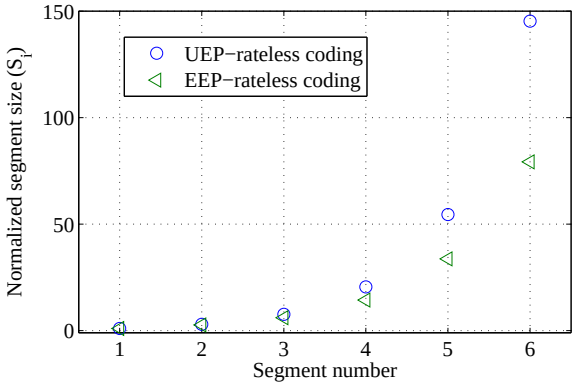
6.2.3 Performance Evaluation of Proposed VOD Protocols

In order to evaluate our proposed VOD protocols, we compare the startup delay of our scheme with UEP-rateless coding to startup delay of the case where video is encoded with an EEP-rateless code. We compare the two protocols for different bandwidths allocated per individual channel, for $B = 2B_0$ to $B = 5B_0$, and when number of segments varies from 2 to 8 segments. In Figure 6.10 the percentage of the reduction made in the startup delay of our proposed protocol is illustrated.

From Figure 6.10, we can see a significant improvement in the systems performance for the same bandwidth. For example, if UEP-rateless codes are used instead of EEP-rateless codes, when the video is partitioned into six segments and $B = 5B_0$, the startup delay decreases about 55%. Similarly, we compare the modified VOD protocol with the segmentation given by (6.7) in two cases of EEP-rateless and UEP-rateless coding. Figure 6.11 depicts the percentage by which the startup delay declines for different bandwidths and different number of segments, when a UEP-rateless code is employed instead of an EEP-rateless code.



(a) Proposed VOD protocol segment sizes, when the bandwidth of server and the bandwidth of clients are equal to $B \times N_S$.



(b) Modified VOD protocol segment sizes, when the bandwidth of server is equal to $B \times N_S$ and the bandwidth of clients is equal to $2B$.

Figure 6.9 Segmentation sizes for the proposed VOD protocols.

From Figures 6.10 and 6.11, it can be seen that our proposed VOD protocol and its modified version that are based on UEP-rateless coding outperform the case where EEP-rateless coding is employed. We also note that as the transmission bandwidth B allocated to each stream increases, the efficiency of the proposed protocols also increases. An increase in the performance can also be observed when the number of

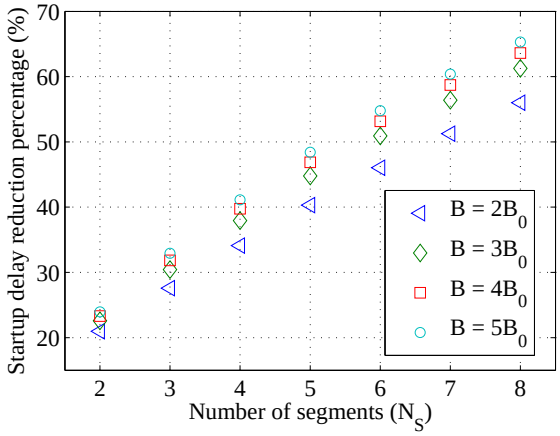


Figure 6.10 Percentage of reduction made in the startup delay of our original proposed VOD protocol with UEP-rateless coding compared to the case where EEP-rateless coding is employed.

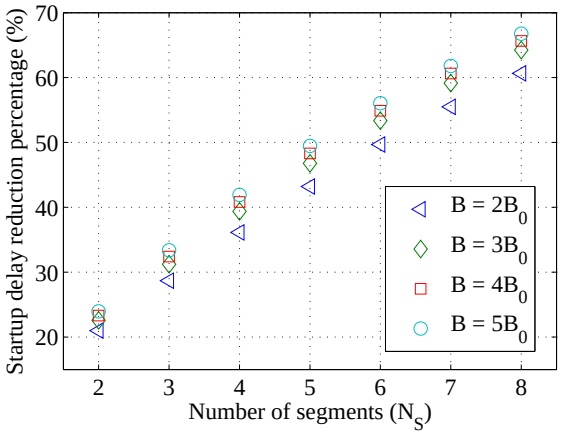


Figure 6.11 Percentage of reduction made in the startup delay of our modified proposed VOD protocol with UEP-rateless coding compared to the case where EEP-rateless coding is employed.

segments increases.

6.3 Conclusion

In this chapter, we studied application-layer UEP-rateless codes for two different important video transmission problems. In our proposed schemes we have exploited

two properties of video streams and showed that UEP-rateless codes can well be applied to improve the efficiency of the systems. The first property of a video is that not all the frames have equal importance. Some frames such as I-frames are more important than other frames (such as P- and B-frames). The second property is that video streams, as opposed to bulk data, should be recovered in sequence.

In our first problem, we proposed to protect three types of MPEG video frames, i.e., I-, P-, and B-frames unequally according to their importance employing UEP-rateless codes. We derived the analytical expression based on the frames dependencies and found the optimum values of UEP-rateless codes parameters that results in an efficient video transmission. Initially, we evaluated the performance of our algorithm for asymptotic cases (large number of frames), and next we showed that similar gains can be achieved when the number of frames is limited.

In the second problem, we proposed a novel periodic VOD broadcasting protocol with unique features of error resiliency and low-startup delay employing UEP-rateless codes. These features were acquired by dividing the video segments into two partitions, and encoding each segment with a separate UEP-rateless code. We also showed that our proposed VOD scheme can easily be modified for the case that clients have a lower bandwidth than the server. Simulation results showed that our VOD broadcasting protocols with UEP-rateless coding can decrease the startup delay considerably compared to the case where EEP-rateless coding is employed, and it can provide loss resiliency in contrast to some existing VOD protocols.

CHAPTER 7

OPTIMIZED CROSS-LAYER FORWARD ERROR CORRECTION CODING FOR H.264 AVC VIDEO TRANSMISSION

In this section, we investigate the design of UEP-rateless codes when fixed-rate FEC coding is also present at physical layer.

7.1 Introduction

In video transmission systems, UEP FEC codes may be employed both at the AL and PL. Recently, some schemes [130–132] have considered the precise tuning of EEP FEC schemes at AL and PL. However, to the best of our knowledge, exiting schemes have not investigated the cross-layer design of UEP FEC codes at AL and PL for prioritized video transmission. Employing FEC codes at both layers introduces two interesting tradeoffs that we investigate in this chapter. *First*, both FEC codes share a common channel bandwidth to add their redundancy and the optimal ratio of overhead added by each needs to be determined for a given channel SNR and bandwidth. *Second*, since UEP can be provided at both layers, we need to find the optimal UEP/EEP FEC setup to maximize the video PSNR. To tackle these tradeoffs we *concurrently* tune the parameters of two FEC codes at both layers.

We use UEP-rateless codes [19, 20] at AL and *rate-compatible punctured convolutional* (RCPC) codes [52] at PL. Next, we carry out a cross-layer optimization to find the optimal parameters of both FEC codes by considering the relative priorities of video packets. For known channel SNR (i.e., $\frac{E_s}{N_0}$), we address the problem of as-

signing optimal FEC code rate at the AL and PL layers to the individual priority slices within the channel bit-rate limitations. The information about the channel conditions can be obtained from the receiver in the form of channel side information (CSI) [130, 133–136].

The scheme provides higher transmission reliability to the high priority slices at the expense of the higher loss rates for low priority slices, and whenever necessary also discards some low priority slices to meet the channel bit-rate limitations. We show that adapting the FEC code rates to the slice priority reduces the overall expected video distortion at the receiver. Our scheme does not assume retransmission of lost slices.

LT codes have recently become popular in video transmission schemes due to their low complexity [1]. Kushwaha et al. [137] used LT codes to encode *group-of-pictures* (GOP) of each layer of H.264 SVC video for transmission over cognitive radio wireless networks. Ahmad et al. [135] took advantage of the ratelessness of LT codes and proposed an adaptive FEC scheme for video transmission over Internet by employing feedback from receivers in the form of acknowledgement. Cataldi et al. [136] proposed a novel LT code, called sliding-window raptor codes, with a higher efficiency than regular LT codes. They used these codes to provide UEP for a two-layer H.264 SVC scalable video. LT codes were also used in [138–143] to design the streaming schemes with lower complexity.

Stockhammer et al. [130] defined the protocol stack, including the FEC coding at AL and PL, for the *multimedia broadcast multicast service* (MBMS) download and streaming in UMTS. In [130], a raptor code [2] is used at AL and the turbo code at PL. Gomez and Bria [131] suggested employing the raptor codes as AL FEC in DVB-H systems for mobile terminals and demonstrated its advantages over conventional *multi-protocol encapsulation* (MPE) FEC. Conventional MPE FEC employs the Reed-Solomon codes to encode the video stream; hence, it lacks the flexibility of LT coding

at AL. Courtade and Wesel [132] considered a setup with LT coding at AL and FEC coding at PL, and showed that the available channel bandwidth should be optimally split between AL and PL FEC codes to improve the system performance.

7.2 Cross-Layer FEC Coding for H.264 Video Bitstream

In this section, we discuss a priority assignment scheme for H.264 AVC video slices, design of LT and RCPC codes, and our proposed cross-layer FEC scheme. We consider a unicast video transmission from a source node (at the transmitter) to a destination node (at the receiver) in a single hop wireless network, and ignore the intermediate network layers, i.e., transport layer (TL), network layer (NL), and link layer (LL). This allows our algorithm to be employed with different exiting network protocols stacks.

7.2.1 Priority Assignment for H.264 Video Slices

In H.264 AVC, the video frames are grouped into GOPs and each GOP is encoded as a unit. For the sake of simplicity, we use a GOP length of 30 frames which corresponds to a duration of one second. We encode each GOP independently by employing FEC codes. We have used a fixed slice size configuration where macroblocks of a frame are aggregated to form a fixed slice size. Let N_s be the *average* number of slices in one second of the video. More details of the video encoding parameters are given in Section 7.4.

We use the CMSE metric to determine the slice priority. All slices in a GOP are distributed into $r = 4$ priority classes of *equal size* based on their CMSE value. The priority 1 slices induce the highest distortion whereas the priority 4 slices induce the least distortion to received video quality. Note that using more than four slice priorities would result in a more accurate and flexible UEP coding at the cost of

much higher complexity due to a larger number of design parameters. In fact, using N_s priority levels would achieve the best performance where each slice is separately protected based on its CMSE. On the other hand, using fewer than four priority levels would limit the flexibility of our scheme and hence decrease its performance.

Let CMSE_i denote the *average* CMSE of all slices in a priority class i. Therefore, we have $\text{CMSE}_1 > \text{CMSE}_2 > \text{CMSE}_3 > \text{CMSE}_4$. Since CMSE_i may vary considerably for various videos depending on their content, we use the *normalized* CMSE_i, $\overline{\text{CMSE}}_i = \frac{\text{CMSE}_i}{\sum_{j=1}^{4} \text{CMSE}_j}$ to represent the relative importance of a priority class. We show $\overline{\text{CMSE}}_i$ for six H.264 test video sequences in Table 7.1. These video sequences have widely different spatial and temporal content.

Table 7.1 Normalized CMSE, $\overline{\text{CMSE}}_i$, for slices in different priorities of sample videos

Sequence	$\overline{\text{CMSE}}_1$	$\overline{\text{CMSE}}_2$	$\overline{\text{CMSE}}_3$	$\overline{\text{CMSE}}_4$
Coastguard	0.61	0.22	0.12	0.05
Foreman	0.63	0.21	0.11	0.05
Bus	0.64	0.21	0.10	0.04
Football	0.65	0.21	0.10	0.04
Silent	0.68	0.2	0.09	0.03
Akiyo	0.85	0.12	0.03	0.01

Table 7.1 shows that the first five videos, which have very different characteristics (such as slow, moderate, and high motion), have almost similar $\overline{\text{CMSE}}_i$ values. We also observed similar $\overline{\text{CMSE}}_i$ values for other video sequences, such as Table Tennis and Mother Daughter. However, Akiyo, which is a static sequence, has different

$\overline{\text{CMSE}}_i$ values than other sequences. The $\overline{\text{CMSE}}_i$ values changed only slightly when these videos were encoded at different bit rates (i.e., 512Kbps and 1Mbps) and slices sizes (150bytes to 900bytes). When these videos are encoded at 840Kbps with 150byte slices, we get $N_s \approx 700$.

We choose the $\overline{\text{CMSE}}_i$ values of Bus, which are similar to most other videos discussed above, to tune our proposed cross-layer scheme for all videos in Section V. Since the $\overline{\text{CMSE}}_i$ values of Akiyo are different, we also study the performance of the proposed cross-layer FEC scheme for Akiyo by using its own $\overline{\text{CMSE}}_i$ values, and compare it to the performance of the scheme designed using the $\overline{\text{CMSE}}_i$ values of Bus in Section VI.

7.2.2 Design of LT Codes at AL

The video slices may be either directly passed to PL or encoded using an EEP/UEP LT code before passing to PL. Therefore, the AL-frames contain either uncoded or LT coded video slices. When no LT coding is performed at AL, each video slice forms an AL frame and the N_s AL-frames are given to the lower network layers. When the LT coding is performed at AL, $\gamma_t N_s$ AL-frames, containing LT coded output symbols, are generated from N_s video slices, where $\gamma_t \geq 1$ denotes the LT coding overhead at *transmitter*. Note that the size of each LT coded AL-frame is still 150bytes, i.e., same as input video slice size, whereas the number of AL-frames increases to $\gamma_t N_s$ from N_s. We emphasize that the transmitted LT overhead γ_t should not be confused with the received LT coding overhead γ_r. Generally, $\gamma_r \neq \gamma_t$ since some AL-frames may not be correctly delivered to the receiver due to channel induced losses.

The parameters of the UEP LT code at AL are $k_i, i \in \{1, \ldots, 4\}$ and γ_t, which need to be optimized while considering the FEC at PL in the cross-layer setup. Since all $r = 4$ priority levels have equal size, we have $\tau_1 = \tau_2 = \tau_3 = \tau_4 = \frac{1}{4}$ (see Section 2.2). For EEP/UEP LT coding, we use the standard degree distribution $\Omega_{shok}(x)$.

The $\gamma_t N_s$ LT coded symbols are randomly and uniformly generated; thus, they are statistically independent and have equal importance. Therefore, only the EEP FEC coding can be performed at PL when AL FEC coding is performed. On the other hand, when video slices are passed to the lower layers without AL FEC coding, the UEP FEC coding can be performed at PL based on the slices priority.

7.2.3 Design of RCPC Codes at PL

At PL, the *cyclic redundancy check* (CRC) bits are added to each AL-frame to detect any RCPC decoding errors. We use the industry-standard CRC-8 defined by the polynomial $1 + x^2 + x^4 + x^6 + x^7 + x^8$ [144]. Next, each AL-frame is encoded using a UEP/EEP RCPC code. As mentioned earlier, we employ an RCPC code designed in [52] with the mother code rate of $R = \frac{1}{3}$ and memory $M = 6$. Based on the AL-frame priority level, the RCPC codes may be punctured to get appropriate higher rates. For four priority groups of AL-frames, we have $R_1 \leq R_2 \leq R_3 \leq R_4$ and $R_i \in \left\{\frac{8}{8}, \frac{8}{9}, \frac{8}{10}, \frac{8}{12}, \frac{8}{14}, \frac{8}{16}, \frac{8}{18}, \frac{8}{20}, \frac{8}{22}, \frac{8}{24}\right\}$, where R_i represents the RCPC code rate of priority i AL-frames. Therefore, the parameters that need to be tuned at PL are R_1 through R_4. For EEP RCPC codes, we have $R_1 = R_2 = R_3 = R_4$. We refer to a frame encoded by the RCPC code as a PL-frame.

For the sake of simplicity and without the loss of generality, we assume that each transmitted packet contains one PL-frame. Note that the number of PL-frames in a packet does not affect the optimal cross-layer setup of FEC codes in our scheme. We have used a conventional BPSK modulation and a simple AWGN channel. Our model can be easily extended to the more complex channel models by using an appropriate P_d in (2.11) from [52].

7.2.4 System Model at Transmitter

Based on our discussions so far, we can use four combinations of cross-layer FEC coding schemes at AL and PL (summarized in Table 7.2). Note that the FEC coding is necessary at PL but optional at AL. We illustrate the layout of cross-layer FEC schemes in Figure 7.1(a) for S-I and S-II schemes and in Figure 7.1(b) for S-III and S-IV schemes. The cross-layer optimization of these FEC-schemes is discussed in Section 7.3.

Table 7.2 Various combinations of cross-layer FEC coding schemes

Model	S-I	S-II	S-III	S-IV
AL FEC	No FEC	No FEC	EEP	UEP
PL FEC	EEP	UEP	EEP	EEP

In S-I and S-II, the FEC coding is applied only at PL. In S-I, the equal protection (i.e., EEP RCPC coding) is provided to all frames regardless of their importance. In S-II, the video slices are protected at PL with various protection levels based on their priority by using the UEP RCPC coding. We expect this scheme to have a considerably improved performance compared to S-I. Note that the priority of each AL-frame is conveyed to PL by using the cross-layer communication. This setup represents the schemes proposed in [34, 122, 123, 145–149].

In S-III and S-IV, FEC coding is applied at both AL and PL in a cross-layer fashion. In S-III scheme, we add the FEC coding at AL by using regular EEP LT codes to the base S-I setup. As we will see later, S-III cannot outperform S-I for all channel conditions since LT codes require extra coding overhead. However, this scheme has the ratelessness property, meaning that it can tolerate loss of the AL-frames and still recover the original video slices after LT decoding. This is in contrast

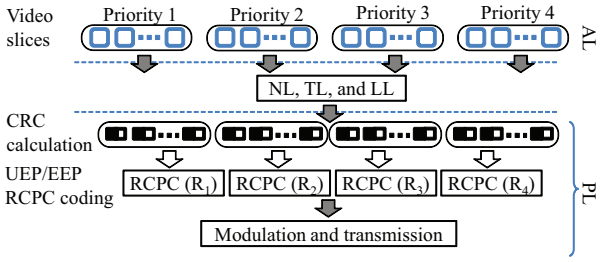
to S-I and S-II where the corrupted frames are considered lost. This setup represents the cross-layer FEC schemes proposed in [130–132, 150–155].

In the proposed S-IV scheme, we apply the UEP LT codes where different slices are protected according to their priority. This scheme benefits both from ratelessness and UEP property. We expect this scheme to achieve the best performance. When LT coding is applied at AL, the rateless coded symbols are uniformly generated and all the encoded AL-frames have equal importance. As a result, using UEP FEC coding at PL would not be beneficial. This is why we have used EEP FEC coding at PL in the cross-layer S-III and S-IV schemes.

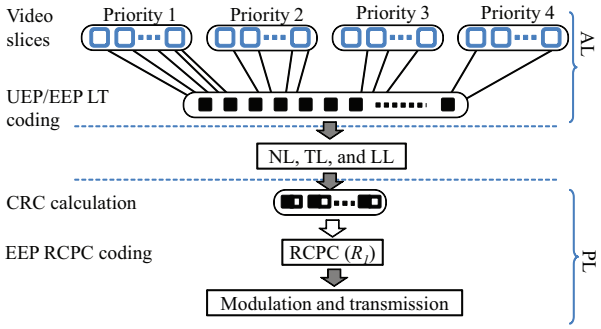
7.2.5 Decoding at Receiver

Let PER_i denote the packet error rate of AL-frames of priority i at the receiver after RCPC decoding and before LT decoding at AL. PER_i can be computed using (2.10).

In S-I and S-II schemes, each AL-frame consists of an uncoded video slice (i.e., LT coding is not performed at AL). Therefore, the *video slice loss rate* (VSLR) of slices in priority i is $\text{VSLR}_i = \text{PER}_i$. In S-III and S-IV schemes, on the other hand, the LT decoding should also be performed and the decoding error rate of LT codes should be considered in VSLR_i. In S-III and S-IV schemes, the EEP RCPC code is used at PL, hence we have PER_1=PER_2=PER_3=PER_4=PER. In this case, we employ (2.6) with $\gamma_r = \gamma_t N_s(1 - \text{PER})$, degree distribution $\Omega_{shok}(x)$ (2.3), and a given set of $k_i, i \in \{1, \ldots, 4\}$ to find the final LT decoding symbol error rates $y_i, i \in \{1, \ldots, 4\}$ for each priority at the receiver (see Section 2.2). If the symbol decoding error rate of priority i is y_i, then $\text{VSLR}_i = y_i$.



(a) The proposed S-I and S-II cross-layer FEC schemes. In these schemes, the video slices are prioritized at AL and UEP/EEP FEC coding is performed only at PL. In S-I we have $R_1 = R_2 = R_3 = R_4$. Here, TL, NL, and LL represent the transport, network, and link layers, respectively.



(b) The proposed S-III and S-IV cross-layer FEC schemes. In these schemes, the video slices are prioritized at AL and two layers of FEC coding at AL and PL are performed. We perform UEP/EEP LT coding at AL and EEP RCPC coding at PL. In S-III, we have $k_1 = k_2 = k_3 = k_4 = 1$ for EEP LT coding.

Figure 7.1 Setups of four cross-layer FEC schemes.

7.3 Cross-Layer Optimization of the Proposed FEC Schemes

In our cross-layer FEC schemes, we consider the following issues. *First*, the AL and PL FEC codes share the same available channel bandwidth to add their coding

redundancy. As the channel $\frac{E_s}{N_0}$ increases, the RCPC code rate at PL can be decreased. Thus, more channel bandwidth becomes available for improving the LT coding at AL. For low values of $\frac{E_s}{N_0}$, assigning a higher portion of the available redundancy to LT codes at AL may not improve the delivered video quality since almost all PL-frames would be corrupted during transmission. Therefore, a stronger RCPC code rate should be used at PL. This consumes a larger portion of the channel bandwidth allowing only a weaker LT code at AL. *Second*, UEP FEC may be used either at AL or PL. We study how using UEP relates to varying $\frac{E_s}{N_0}$ and the bandwidth portions assigned to each FEC code. *Third*, the optimal FEC code rates for one scheme in Table 7.2 may be substantially different from another scheme.

To find the optimal parameters for both the FEC schemes and the portion of channel bandwidth they share, we discuss below the cross-layer optimization for the four schemes given in Table 7.2.

7.3.1 Formulation of Optimization Problem

The goal of cross-layer optimization in our scheme is to deliver a video with the highest possible PSNR for a given channel bandwidth C and SNR. Since computing the video PSNR requires decoding the video at the receiver, it is not feasible to use PSNR directly as the optimization metric due to its heavy computational complexity. Therefore, we use a *substitute* function F to *mathematically capture* the behavior of PSNR.

The PSNR of a video stream depends on the percentage of lost slices and their CMSE values [31, 33]. However, the slice loss may not be linearly correlated to the decrease in PSNR. Therefore, we define a function "*normalized F*", denoted by $\overline{F}$, to capture the behavior of PSNR based on the slice loss rates and their CMSE as

follows:

$$\overline{F} = \sum_{i=1}^{r} \overline{\text{CMSE}}_i^{\alpha} \text{VSLR}_i. \tag{7.1}$$

In (7.1) we use a parameter $\alpha \geq 0$ that needs to be *tuned* so that $\overline{F}$ can correctly capture the behavior of PSNR. Here, α adjusts the *weight* assigned to slices of each priority level such that minimizing F results in maximizing the PSNR. Selecting the optimal α is discussed in the next section.

To minimize $\overline{F}$, we tune the parameters of the FEC codes at AL and PL. In S-I scheme, the optimization function finds the optimal RCPC code rate R for a given channel data rate C as

$$\begin{aligned} &\underset{\{R\}}{\text{argmin}}\, \overline{F} = \{R^*\} \\ &\text{s.t. } N_s(S+1)R^{-1} \leq C, \end{aligned} \tag{7.2}$$

where $S+1$ is the slice size $S = 150$bytes plus one byte CRC.

In S-II, the optimization parameters are R_1 through R_4, such that $R_1 \leq R_2 \leq R_3 \leq R_4$. For this scheme, the optimization function can be written as

$$\begin{aligned} &\underset{\{R_1,R_2,R_3,R_4\}}{\text{argmin}}\, \overline{F} = \{R_1^*, R_2^*, R_3^*, R_4^*\} \\ &\text{s.t. } N_s(S+1)\sum_{i=1}^{4} \frac{R_i^{-1}}{4} \leq C. \end{aligned} \tag{7.3}$$

The optimization parameters for S-III are γ_t and R. In S-III, we have $k_1 = k_2 = k_3 = k_4 = 1$ since EEP LT coding is used at AL. The channel data rate is shared among the two FEC codes and needs to be tuned by selecting an appropriate γ_t. The optimization function is

$$\begin{aligned} &\underset{\{\gamma_t,R\}}{\text{argmin}}\, \overline{F} = \{\gamma_t^*, R^*\} \\ &\text{s.t. } \gamma_t N_s(S+1)R^{-1} \leq C. \end{aligned} \tag{7.4}$$

In S-IV, the UEP LT codes are used and optimization parameters are k_1 through k_3, along with γ_t and R. Here, the value of k_4 can be determined based on k_1 through

k_3 since $\sum_{j=1}^{r} k_j \tau_j = 1$ (see Section 2.2). As a result, the optimization function is

$$\begin{aligned} \underset{\{k_1,k_2,k_3,\gamma_t,R\}}{\operatorname{argmin}} \ & \overline{F} = \{k_1^*, k_2^*, k_3^*, \gamma_t^*, R^*\} \\ \text{s.t. } & \gamma_t N_s (S+1) R^{-1} \leq C. \end{aligned} \tag{7.5}$$

The optimization of LT code's parameters involves employing equation (2.6) for various priority levels. Since (2.6) has a recursive form, it may not be represented by a linear function. Furthermore, the concatenation of two FEC codes presents a nonlinear optimization problem, which cannot be solved using *linear programming* techniques. Therefore, we use the *genetic algorithms* (GA) to perform optimizations [99, 100]. Although GA are computationally complex, they can give solutions which are close to the global optimum [99, 100, 156]. There are numerous implementations of GA. We used the GA toolbox available in Matlab [157]. For performance evaluation of GA methods, we refer the interested readers to [100, 101].

7.3.2 Optimal Value of α

In Table 7.1, the normalized CMSE values ($\overline{\text{CMSE}}_i$) of the video sequences, except Akiyo, were similar. Therefore, the optimal parameters computed for Bus video would be *almost* optimal for the other four video sequences generated by the same encoding parameters. We therefore use the $\overline{\text{CMSE}}_i$ of the Bus video with data rate of 840Kbps to perform our optimizations, followed by the Akiyo sequence. We implement our cross-layer FEC setup including LT coding at AL and RCPC coding at PL for S-I through S-IV (see Table 7.2) in Matlab environment.

In the first step, we find the optimal value of α such that minimizing $\overline{F}$ maximizes PSNR of the decoded video. For this, we perform the optimization to minimize $\overline{F}$ for various values of α and also compute the corresponding video PSNR. Note that the value of α has no effect on a cross-layer scheme with EEP FEC code since all VSLR_i's are equal in this case. Therefore, we perform our optimization for S-II, which is the

simplest UEP FEC scheme. Note that using UEP LT coding at AL (in S-IV) does not affect the optimal α. Table 7.3 reports the PSNR of the Bus video for various values of α and $\frac{E_S}{N_0}$ for $C = 1.4$Mbps when $\overline{F}$ is minimized in S-II.

Table 7.3 PSNR of Bus video sequence for various values of α with optimized F for S-II.

$\frac{E_S}{N_0}$	1dB		2dB		3dB		4dB	
α	$1, 2$	3	$1, 2$	3	1	$2, 3$	1	$2, 3$
PSNR	**18.2**	16.85	**22.3**	19.8	**25.8**	20.6	**29.69**	29

The value of α that concurrently maximizes the PSNR of the video for all values of $\frac{E_S}{N_0}$ is $\alpha = 1$. Although not shown in Table 7.3, the non-integer values of α and $\alpha < 1$ were also considered in optimization. $\alpha = 1$ also gave the best results for Akiyo.

7.3.3 Discussion of Cross-Layer Optimization Results

We report the cross-layer optimization results, including the FEC parameters (e.g., R_i, γ_t, and k_i), $VSLR_i$, normalized $\overline{F}$, and non-normalized F for the $\overline{\text{CMSE}}_i$ values of Bus video. Note that F is calculated by replacing the $\overline{\text{CMSE}}_i$ by the actual average CMSE_i for the video sequence under consideration. The results of all four FEC schemes for three video sequences (Bus, Foreman and Coastguard) are reported in Tables 7.4 through 7.7 for channel bit rate $C = 1.4$Mbps, and in Tables 7.8 through 7.11 for channel bit rate $C = 1.8$Mbps. The results for Akiyo are discussed in Section 7.4.

From Tables 7.4 and 7.5 (for 1.4Mbps channel bit rate), we observe that the use of UEP RCPC coding at PL in S-II scheme achieves much better performance (i.e., lower F_{Bus}) than the EEP RCPC coding in S-I scheme. Both schemes do not use

Table 7.4 Optimal cross-layer parameters for S-I scheme with $C = 1.4$Mbps

E_s/N_0	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB	2.75dB	3dB	4dB	5dB
$\overline{F}$	0.998	0.988	0.949	0.852	0.694	0.503	0.328	0.197	0.11	0.008	0
F_{Bus}	443.4	438.9	421.6	378.5	307.9	223.5	145.7	87.5	48.9	3.1	0
$F_{Forem.}$	214.7	212.5	204.1	183.3	149.1	108.2	70.6	42.4	23.7	1.5	0
$F_{Coast.}$	179.8	178.0	171.0	153.5	124.9	90.6	59.1	35.5	19.8	1.3	0
R	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$
VSLR$_i, \forall i$	0.998	0.988	0.949	0.852	0.693	0.503	0.328	0.197	0.11	0.007	0

FEC coding at AL. Similar performance is also observed in Tables 7.8 and 7.9 for 1.8Mbps channel bit rate.

Since the RCPC code rate of $\frac{8}{12}$ at PL is not strong enough for $\frac{Es}{No} \leq 2$dB, the value of F_{Bus} in S-I scheme is high ($F_{Bus} > 300$ in Table 7.4) because many packet are corrupted due to high channel errors. For a successful decoding in LT, the number of error-free packets received should be above a threshold. As a result, S-III scheme (which also uses RCPC with the same code rate as in S-I) achieves a lower performance (higher value of F_{Bus}) than S-I for $\frac{Es}{No} \leq 2$dB (see Tables 7.4 and 7.6). However, S-III scheme achieves much better performance ($F_{Bus} < 10$) than S-I for $\frac{Es}{No} \geq 2.5$dB because fewer packets are now corrupted at PL and the LT coding becomes effective. Table 7.10 shows a similar behavior at $\frac{Es}{No} \geq 1$dB for channel bit rate of $C = 1.8$Mbps.

From Tables 7.6 and 7.7, we observe that the proposed S-IV scheme achieves much lower values of F_{Bus} than S-III at all values of $\frac{Es}{No}$ for $C = 1.4$Mbps channel bit rate. A similar behavior is also observed from Tables 7.10 and 7.11 for $C = 1.8$Mbps channel bit rate. This demonstrates that using UEP LT codes at AL along with EEP RCPC codes at PL gives a far superior performance than using the EEP codes at both layers.

From Table 7.7 for S-IV scheme, we observe an interesting tradeoff between the code rates assigned to FEC codes at AL and PL. For lower values of $\frac{Es}{No}$, a larger

Table 7.5 Optimal cross-layer parameters for S-II scheme with $C = 1.4$Mbps

E_s/N_0	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB	2.75dB	3dB	4dB	5dB
$\overline{F}$	0.172	0.163	0.158	0.111	0.077	0.059	0.05	0.046	0.041	0.003	0
F_{Bus}	76.1	72.2	70.1	49.3	34.0	25.9	22.1	20.4	17.9	1.1	0
$F_{Forem.}$	30.2	28.4	27.4	21.8	14.3	10.3	8.4	7.6	7.7	0.5	0
$F_{Coast.}$	30.7	29.1	28.2	20.5	14.3	11.1	9.5	8.8	7.4	0.5	0
R_1	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$
R_2	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$
R_3	$\frac{8}{9}$	$\frac{8}{9}$	$\frac{8}{9}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$
R_4	1	1	1	1	1	1	1	1	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$
VSLR_1	0.007	0.003	0.001	0.072	0.036	0.017	0.008	0.004	0.001	0	0
VSLR_2	0.063	0.033	0.0162	0.072	0.036	0.017	0.008	0.004	0.11	0.007	0
VSLR_3	1	1	1	0.072	0.036	0.017	0.008	0.004	0.11	0.007	0
VSLR_4	1	1	1	1	1	1	1	1	0.11	0.007	0

portion of the bit budget is assigned to RCPC codes at PL rather than LT codes at AL because the LT coding cannot be effective when large number of packets are corrupted due to channel errors. Furthermore, a stronger UEP (i.e., higher value of k_i to higher priority video slices) is provided at AL. For higher values of $\frac{Es}{No}$, the RCPC code rate is relatively high and more protection is provided to LT codes at AL. Also, the UEP (i.e., value of k_i) at AL is relatively less strong now. We observe a similar behavior from Table 7.11 for $C = 1.8$Mbps channel bit rate.

Overall, the proposed S-IV scheme achieves the best performance at different channel SNRs, followed by S-II scheme for $\frac{Es}{No} \leq 2.5$dB (for $C = 1.4$Mbps) and 1dB (for $C = 1.8$Mbps). S-III outperforms S-II for other higher channel SNRs. We observe similar results for Foreman and Coastguard videos. Therefore, we can generally conclude that it is optimal to provide UEP at AL and EEP at PL using a cross-layer design.

Table 7.6 Optimal cross-layer parameters for S-III scheme with $C = 1.4$Mbps

E_s/N_0	1.75dB	2dB	2.25dB	2.5dB	2.75dB	3dB	4dB	5dB
$\overline{F}$	1	0.972	0.268	0.022	0.021	0.017	0.007	0.006
F_{Bus}	444.3	431.9	119.2	9.8	9.3	5.3	2.1	0.8
$F_{Forem.}$	215.1	209.1	57.7	4.7	4.5	2.6	1.0	0.4
$F_{Coast.}$	180.2	175.2	48.3	4.0	3.8	2.1	0.8	0.3
R	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{10}$	$\frac{8}{10}$	$\frac{8}{9}$
γ_t	1.05	1.05	1.05	1.05	1.05	1.25	1.25	1.4
$\text{VSLR}_i, \forall i$	1	0.972	0.268	0.022	0.021	0.012	0.005	0.002

Note that the optimization is performed only once for a given set of $\overline{\text{CMSE}}_i$ values, a GOP structure, and a set of channel SNRs, and need not be run separately for each GOP. The same set of optimized parameters can be used for any video stream with similar properties.

7.4 Performance Evaluation of FEC Schemes For Test Videos

In this section, we evaluate the performance of our optimized cross-layer FEC schemes for four CIF (352×288 pixels) video sequences, Bus, Foreman, Coastguard, and Akiyo. These sequences were encoded using H.264/AVC JM 14.2 reference software [158] at 840Kbps and 150bytes slice size, for a GOP length of 30 frames with GOP structure $IDR\ B\ P\ B \ldots P\ B$ at 30 frames/sec. The slices were formed using dispersed mode FMO with two slice groups per frame. Two reference frames were used for predicting the P and B frames, with error concealment enabled using temporal concealment and spatial interpolation. We have used two channel transmission rates of $C = 1.4$ and $C = 1.8$Mbps to study the performance over AWGN channels.

We used the slice loss rates reported in Tables 7.4 through 7.7 to evaluate the average PSNR of three video sequences (Bus, Foreman, and Coastguard) in Figures

Table 7.7 Optimal cross-layer parameters for S-IV with schemes $C = 1.4$Mbps

E_s/N_0	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB	2.75dB	3dB	4dB	5dB
$\overline{F}$	0.157	0.058	0.047	0.045	0.044	0.026	0.017	0.016	0.013	0.005	0.004
F_{Bus}	69.7	25.6	20.9	19.9	19.6	11.4	7.6	7.2	5.8	2.1	2.0
$F_{Forem.}$	27.3	10.1	7.8	7.3	7.2	5.1	3.4	3.2	2.6	0.9	0.9
$F_{Coast.}$	28.0	10.9	9.0	8.6	8.5	4.7	3.1	2.9	2.4	0.9	0.8
R	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{10}$	$\frac{8}{10}$	$\frac{8}{10}$
γ_t	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.2	1.2	1.2
k_1	2	1.4	1.4	1.4	1.4	1.2	1.2	1.2	1.2	1.2	1.2
k_2	2	1.3	1.3	1.3	1.3	1.1	1	1	1	1	1
k_3	0	1.3	1.3	1.3	1.3	0.9	0.9	0.9	0.9	1	1
k_4	0	0	0	0	0	0.8	0.9	0.9	0.9	0.8	0.8
VSLR_1	0.004	0.014	0.004	0.002	0.002	0.015	0.008	0.008	0.006	0.002	0.002
VSLR_2	0.004	0.021	0.007	0.004	0.003	0.024	0.025	0.024	0.019	0.007	0.007
VSLR_3	1	0.021	0.007	0.004	0.003	0.064	0.043	0.041	0.034	0.007	0.007
VSLR_4	1	1	1	1	1	0.107	0.043	0.041	0.034	0.028	0.026

7.2 through 7.4 for $C = 1.4$Mbps channel bit rate. Similarly, the slice loss rates reported in Tables 7.8 through 7.11 were used to evaluate the average PSNR of these video sequences in Figures 7.5 through 7.7 for $C = 1.8$Mbps channel bit rate. Figures 7.2 through 7.7 confirm that our proposed cross-layer S-IV scheme, with UEP FEC coding at AL and EEP FEC coding at PL, achieves considerable improvement in average video PSNR, especially at low values of $\frac{E_s}{N_0}$. It outperforms S-II scheme, which uses UEP RCPC code at PL, by about $2 \sim 7$dB for $\frac{E_s}{N_0} \leq 3.5$dB (at $C = 1.4$Mbps) and $\frac{E_s}{N_0} \leq 1.5$dB (at $C = 1.8$Mbps). Only S-III has a comparable performance at $\frac{E_s}{N_0} \geq 2.5$dB for $C = 1.4$Mbps and 1dB for $C = 1.8$Mbps. However, at low values of $\frac{E_s}{N_0}$ the S-IV scheme considerably outperforms S-III.

Although our cross-layer FEC parameters were optimized for Bus sequences, the

Table 7.8 Optimal cross-layer parameters for S-I scheme with $C = 1.8$Mbps

E_s/N_0	0dB	0.25dB	0.5dB	0.75dB	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB
$\overline{F}$	0.514	0.343	0.211	0.119	0.064	0.033	0.016	0.008	0.003	0.002	0.001
F_{Bus}	228.4	152.4	93.7	52.9	28.4	14.7	7.1	3.6	1.3	0.9	0.4
$F_{Forem.}$	110.6	73.8	45.4	25.6	13.8	7.1	3.4	1.7	0.6	0.4	0.2
$F_{Coast.}$	92.6	61.8	38.0	21.4	11.5	5.9	2.9	1.4	0.5	0.4	0.2
R	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$
VSLR$_i, \forall i$	0.514	0.343	0.211	0.119	0.064	0.033	0.016	0.008	0.003	0.002	0.001

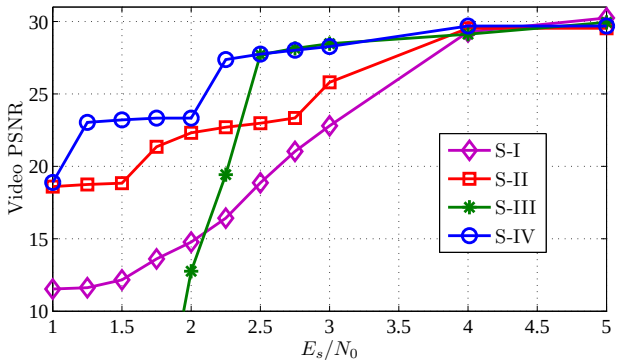


Figure 7.2 Average PSNR of Bus video for different channel SNRs at $C = 1.4$Mbps. The PSNR for error free channel is 30.26dB.

average PSNR performance is similar for the other two test video sequences, i.e., Foreman and Coastguard. As mentioned earlier, both these sequences have different characteristics than the Bus sequence.

Since Akiyo has considerably different values of $\overline{\text{CMSE}}_i$, the proposed S-IV scheme designed by using Bus video's $\overline{\text{CMSE}}_i$ values would be suboptimal for Akiyo. In order to study the effect of these CMSE variations, we also designed the S-IV scheme by using the $\overline{\text{CMSE}}_i$ values of Akiyo and compare its performance with its suboptimal version. The optimization results are reported in Table 7.12 for $C = 1.4$Mbps. In this table, we also included the suboptimal values of F_{sub} and PSNR$_{sub}$, which were

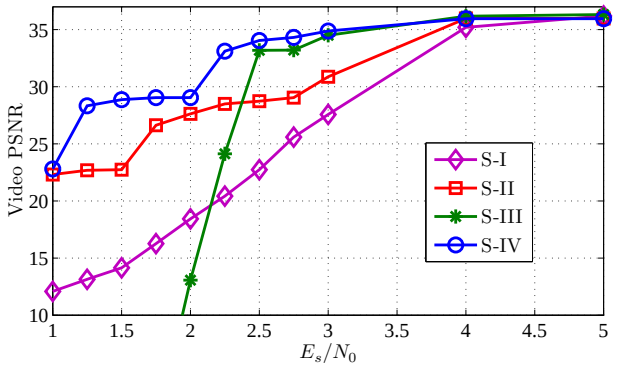


Figure 7.3 Average PSNR of Foreman video for different channel SNRs at $C = 1.4$Mbps. The PSNR for error free channel is 36.9dB.

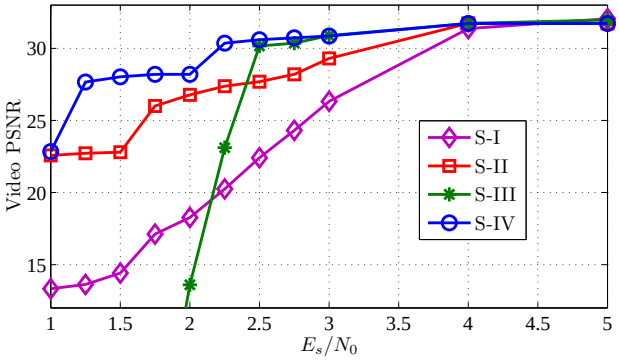


Figure 7.4 Average PSNR of Coastguard video for different channel SNRs at $C = 1.4$Mbps. The PSNR for error free channel is 32.1dB.

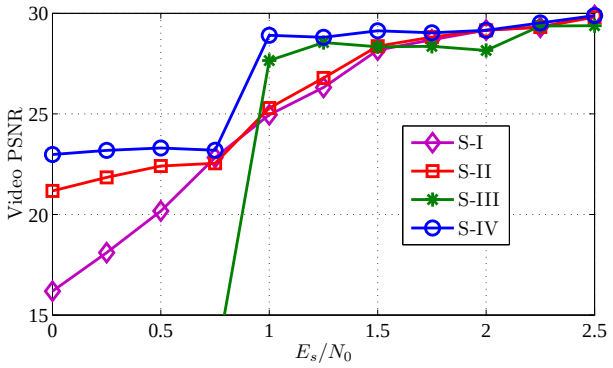


Figure 7.5 Average PSNR of Bus video for different channel SNRs at $C = 1.8$Mbps. The PSNR for error free channel is 30.26dB.

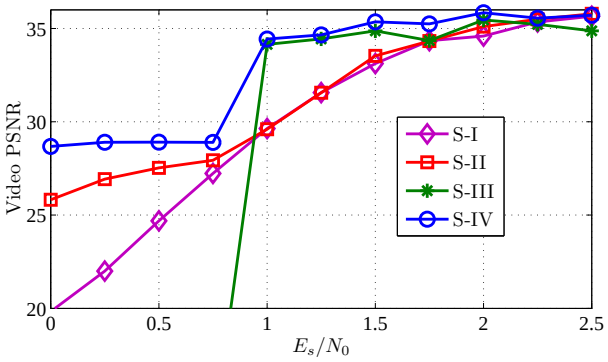


Figure 7.6 Average PSNR of Foreman video for different channel SNRs at $C = 1.8$Mbps. The PSNR for error free channel is 36.9dB.

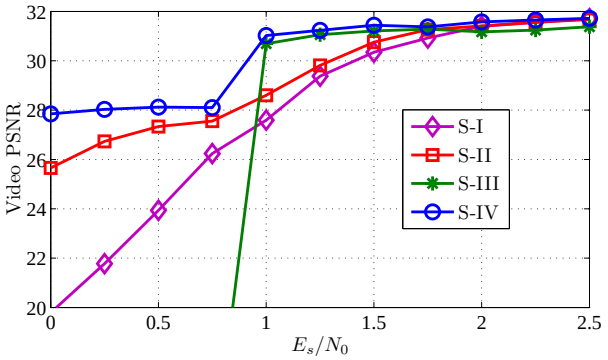


Figure 7.7 Average PSNR of Coastguard video for different channel SNRs at $C = 1.8$Mbps. The PSNR for error free channel is 32.1dB.

Table 7.9 Optimal cross-layer parameters for S-II scheme with $C = 1.8$Mbps

E_s/N_0	0dB	0.25dB	0.5dB	0.75dB	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB
$\overline{F}$	0.099	0.071	0.057	0.049	0.041	0.023	0.012	0.006	0.003	0.001	0.001
F_{Bus}	43.9	31.2	25.4	21.7	18.4	10.0	5.1	2.8	1.1	0.6	0.3
$F_{Forem.}$	18.5	12.5	9.8	8.1	7.6	4.1	2.1	1.1	0.4	0.2	0.1
$F_{Coast.}$	18.4	13.2	10.9	9.4	7.7	4.2	2.2	1.2	0.5	0.3	0.1
R_1	$\frac{8}{20}$	$\frac{8}{20}$	$\frac{8}{20}$	$\frac{8}{20}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$
R_2	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$
R_3	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{18}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$
R_4	$\frac{8}{9}$	$\frac{8}{9}$	$\frac{8}{9}$	$\frac{8}{9}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$
VSLR_1	0.03	0.013	0.007	0.003	0.007	0.003	0.001	0.001	0	0	0
VSLR_2	0.118	0.062	0.033	0.015	0.064	0.033	0.016	0.008	0.003	0.002	0.001
VSLR_3	0.118	0.062	0.033	0.015	0.064	0.033	0.016	0.008	0.003	0.002	0.001
VSLR_4	1	1	1	1	0.393	0.241	0.136	0.072	0.036	0.017	0.008

obtained by using the optimized parameters of Bus video from Table 7.7. The values of PSNR_{opt} and PSNR_{sub} are also shown in Figure 7.8.

In Table 7.12 (for optimal schemes) and Table 7.7 (for suboptimal scheme), the LT code overhead (i.e., γ_t) and RCPC code strength (R) are the same for both schemes, whereas the values of LT code protection level k_i for each priority class vary slightly (e.g., k_1 is higher for optimal scheme compared to the suboptimal scheme). Similarly, the values of VSLR_i for higher priority slices (which have the most impact on F and PSNR) are similar in both tables, except for channel SNRs of 2.25, 2.5dB and 2.75dB in the decreasing order of the difference in values. The maximum PSNR degradation of the suboptimal scheme compared to the optimal scheme is 1.7dB at the channel SNR of 2.25, with only about 0.1 to 0.3dB PSNR degradation at other channel SNRs. We can, therefore, conclude that the performance of the proposed cross-layer FEC

Table 7.10 Optimal cross-layer parameters for S-III scheme with $C = 1.8$Mbps

E_s/N_0	0.5dB	0.75dB	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB
$\overline{F}$	1	0.961	0.016	0.012	0.01	0.01	0.009	0.007	0.007
F_{Bus}	444.3	427.0	7.1	5.3	4.4	4.0	4.0	3.1	3.1
$F_{Forem.}$	215.1	206.7	3.4	2.6	2.2	1.9	1.9	1.5	1.5
$F_{Coast.}$	180.2	173.2	2.9	2.2	1.8	1.6	1.6	1.3	1.3
R	$\frac{8}{16}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$
γ_t	1	1.15	1.15	1.15	1.15	1.15	1.2	1.2	1.2
$\text{VSLR}_i, \forall i$	1	0.961	0.016	0.012	0.01	0.009	0.009	0.007	0.007

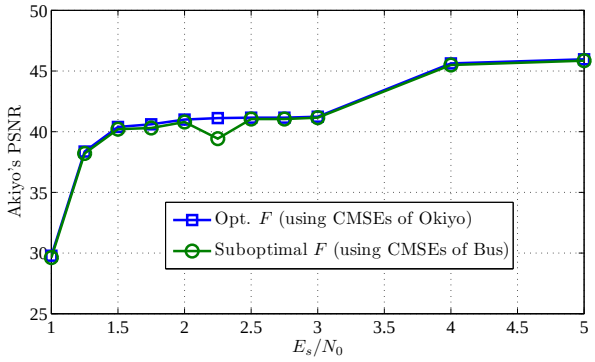


Figure 7.8 Average PSNR performance of the optimal and sub-optimal cross-layer FEC scheme for the Akiyo video sequence.

scheme is not very sensitive to the precise values of normalized CMSE.

7.5 Conclusion

Previously, EEP and UEP FEC coding schemes have been used for video transmission over lossy channels. However, the joint optimization of cross-layer UEP FEC codes at the AL and PL for robust video transmission has never been considered. In this chapter, we used the UEP LT coding at AL and RCPC coding at PL for robust H.264 video transmission over wireless channels. H.264 video slices were prioritized based

Table 7.11 Optimal cross-layer parameters for S-IV scheme with $C = 1.8$Mbps

E_s/N_0	0dB	0.25dB	0.5dB	0.75dB	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB
$\overline{F}$	0.048	0.046	0.045	0.044	0.012	0.009	0.007	0.007	0.006	0.005	0.005
F_{Bus}	21.4	20.3	19.8	19.7	5.1	4.0	3.2	3.1	2.8	2.2	2.1
$F_{Forem.}$	8.0	7.5	7.3	7.2	2.3	1.8	1.4	1.4	1.2	1.0	0.9
$F_{Coast.}$	9.2	8.8	8.6	8.5	2.1	1.6	1.3	1.3	1.1	0.9	0.9
R	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{16}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{14}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$
γ_t	1	1	1	1.15	1.15	1.15	1.15	1.15	1.2	1.2	1.2
k_1	1.4	1.4	1.4	1.4	1.2	1.2	1.2	1.2	1.2	1.2	1.2
k_2	1.3	1.3	1.3	1.3	1	1	1	1	1	1	1
k_3	1.3	1.3	1.3	1.3	0.9	0.9	1	1	1	1	1
k_4	0	0	0	0	0.9	0.9	0.8	0.8	0.8	0.8	0.8
VSLR_1	0.005	0.003	0.002	0.002	0.005	0.004	0.003	0.003	0.003	0.002	0.002
VSLR_2	0.009	0.005	0.004	0.003	0.017	0.013	0.011	0.011	0.009	0.008	0.007
VSLR_3	0.009	0.005	0.004	0.003	0.031	0.024	0.011	0.011	0.009	0.008	0.007
VSLR_4	1	1	1	1	0.031	0.024	0.04	0.037	0.034	0.029	0.028

on their contribution to video quality. We performed the cross-layer optimization to concurrently tune the FEC code parameters at both layers, to minimize the video distortion and maximize the peak signal-to-noise ratio (PSNR). We observed that our cross-layer FEC scheme outperformed other FEC schemes that either use the UEP coding at PL alone or EEP FEC schemes at AL as well as PL. Further, we showed that our optimization works well for different H.264 encoded video sequences, which have widely different characteristics.

Table 7.12 Optimal cross-layer parameters for S-IV at $C = 1.4$Mbps for Akiyo video sequence

E_s/N_0	1dB	1.25dB	1.5dB	1.75dB	2dB	2.25dB	2.5dB	2.75dB	3dB	4dB	5dB
F_{opt}	1.111	0.600	0.287	0.243	0.229	0.223	0.221	0.219	0.215	0.066	0.062
F_{sub}	1.141	0.600	0.317	0.259	0.239	0.494	0.325	0.306	0.240	0.079	0.074
PSNR_{opt}	29.78	38.36	40.39	40.6	41.0	41.12	41.15	41.15	41.23	45.62	45.96
PSNR_{sub}	29.62	38.20	40.2	40.3	40.8	39.42	41.04	41.05	41.15	45.49	45.85
R	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{12}$	$\frac{8}{10}$	$\frac{8}{10}$	$\frac{8}{10}$
γ_t	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.2	1.2	1.2
k_1	2.3	1.4	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.3	1.3
k_2	1.7	1.3	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1	1
k_3	0	1.3	1	1	1	1	1	1	1	0.9	0.9
k_4	0	0	0	0	0	0	0	0	0	0.8	0.8
VSLR_1	0.001	0.014	0.001	0	0	0	0	0	0	0.001	0.001
VSLR_2	0.012	0.021	0.014	0.008	0.006	0.005	0.005	0.004	0.004	0.008	0.007
VSLR_3	1	0.021	0.039	0.024	0.018	0.016	0.015	0.015	0.013	0.015	0.014
VSLR_4	1	1	1	1	1	1	1	1	1	0.028	0.027

CHAPTER 8

DECENTRALIZED COMPRESSIVE DATA STORAGE IN WIRELESS SENSOR NETWORKS

To increase the data persistence in *wireless sensor networks* (WSNs) with N nodes, distributed *data storage* algorithms have been proposed to disseminate sensors readings *throughout* the network so that a data collector can query an *arbitrary small subset* of nodes to obtain all N readings [159, 160].

Recently, *compressive sensing* (CS) techniques [53, 57] have shown that a *compressible* signal with length N can be reconstructed from only $M \ll N$ *random projections* of the signal (also called measurements or compressed samples). Since natural signals are known to be compressible due to strong *spatial* correlation of sensor readings [56, 161, 162], CS can be exploited to design efficient data storage algorithms.

Consequently, we design a decentralized *compressive data storage* algorithm (CStorage) that exploits the *spatial correlation* of the nodes reading and CS to considerably reduce the total number of required transmissions for data storage. In CStorage, we propose to form a CS *measurement* at each node by disseminating *enough* number of readings throughout the network.

To disseminate the network readings, we take advantage of the *broadcast* property of wireless channels to reduce the required number of transmissions. First, we employ the well-known *probabilistic broadcasting* (PB) for data dissemination and propose *CStorage-P*. In PB, no neighbor information or routing table is required for data

dissemination. Nevertheless, PB has a parameter called *forwarding probability* that needs to be optimized as the network changes and nodes need to be informed. In a different approach for data dissemination, we assume nodes can obtain two-hop neighbor information and design a *parameterless* and efficient data dissemination algorithm referred to by *alternating branching* (AB), and design *CStorage-B*. Since AB has no parameter to tune, CStorage-B is *fully scalable* and can automatically adapt to drastic network topology changes. We will show both CStorage-P and CStorage-B reduce the total number of transmissions compared to existing algorithms for data storage in WSNs, while CStorage-B surpasses CStorage-P in the number of transmissions.

To evaluate our proposed algorithms, we employ actual readings from a real WSN [163] to evaluate the performance of our proposed schemes. Moreover, we investigate how real readings should be vectorized (ordered) based on their spatial correlation to obtain a compressible signal. We will show that vectorization problem based on spatial correlation is equivalent to solving the well-known *traveling salesman* problem (TSP).

Previously, Wang et. al. in [58] showed that *sparse* Φ matrices can satisfy CS requirements and designed a data storage algorithm based on sparse Φ matrices. In their work, some random nodes send their readings to randomly selected destinations (Alg. I) or request readings from randomly selected sources (Alg. II) to form measurements at network nodes. Authors in [56, 162, 164–167] proposed *centralized* data collection algorithms where measurements are formed enroute and are collected at a sink. These algorithms require routing tables and full topology knowledge, which may not be always feasible in practical cases.

Further, Rabbat et. al. in [168] proposed to employ *gossiping* to disseminate all the reading in the network. In gossiping, each node iteratively exchanges the value they are maintaining with a random neighbor. After many iterations all network

nodes would obtain the value of the reading and a measurements is formed at nodes.

Finally, authors in [62, 160, 169] proposed data storage algorithms for sensor networks based on *error correction* codes. Although these algorithms are efficiently designed, they have not exploited the compressibility of the signals in a sensor network to reduce the number of transmissions.

8.1 Compressive Data Storage in WSNs Employing PB

In CStorage node $n_j, j \in \{1, 2, \ldots, N\}$, maintains a CS measurement y_j, where $y_j = \phi_j \underline{\mathbf{x}}$ and ϕ_j is an N-dimensional row vector and $\underline{\mathbf{x}} = [x_1 x_2 \ldots x_N]^T$ is sensors readings. Let $\Phi_{tot} = [\phi_1^T \phi_2^T \ldots \phi_N^T]^T$ and $\underline{\mathbf{y}}_{tot} = [y_1 y_2 \ldots y_N]^T$. Further, let $\varphi_{j,i}$ be the element at the j^{th} row and the i^{th} column of Φ_{tot}. The matrix Φ_{tot} is formed when nodes receive various readings employing an underlying data dissemination algorithm. We will propose two dissemination algorithms for this purpose. We first employ *probabilistic broadcasting* and refer to the compressed storage scheme as *CStorage-P*. We also propose another dissemination scheme called *alternating branching*, and refer to the corresponding compressed storage scheme as *CStorage-B*.

When the transmissions are over, Φ_{tot} is formed distributively (as described in detail later) in the network. The data collector queries M nodes for their measurements y_j and the corresponding ϕ_j maintained at each node, and forms $\underline{\mathbf{y}}$ and $\Phi \in \mathbb{R}^{M \times N}$. Next, the data collector obtains $\hat{\underline{\mathbf{x}}}$ an estimate of $\underline{\mathbf{x}}$ employing basis pursuit by solving (2.12).

8.1.1 CStorage-P Design

The CStorage-P algorithm is described in the following:

1. All nodes choose $\varphi_{j,j}$ from $\mathfrak{N}(0, 1)$ and initialize their measurement to $y_j = \varphi_{j,j} x_j$, where $\mathfrak{N}(0, 1)$ is the zero mean and unit variance Gaussian distribution.

2. N_s nodes randomly select themselves as a source node and broadcast their reading to their neighbors.

3. Upon reception of x_i for the *first time* by node l, n_l, performs the following:

   (a) Chooses $\varphi_{l,i}$ from $\mathfrak{N}(0,1)$ and adds $\varphi_{l,i}x_i$ to y_l.

   (b) *Broadcasts* x_i with probability p (PB).

Based on this scheme, Φ_{tot} will be formed. We note that column j of Φ_{tot} corresponds to dissemination of x_j, sensor reading of node j, and row i of Φ_{tot} corresponds to the measurements formed at node i.

To describe CStorage-P, let us consider a small network with $N = 5$ nodes as shown in Figure 8.1 and investigate one PB of CStorage-P. At the beginning, we have $\varphi_{i,j} = 0$ for all $i \neq j, i, j \in \{1, \ldots, 5\}$. Assume n_1 broadcasts x_1. Since n_2 and n_3 are in the transmission range of n_1, they would receive x_1. Nodes n_2 and n_3 multiply x_1 by $\varphi_{2,1}$ and $\varphi_{3,1}$ and add them to y_2 and y_3, respectively. Next, n_2 and n_3 independently decide whether to broadcast x_1 with probability p or not. Assume that n_2 decides to broadcast x_1. Node n_4 would receive x_1 and adds $\varphi_{4,1}x_1$ to y_4. However, we assume that n_3 and n_4 decide not to rebroadcast x_1. Thus, the PB of x_1 is over and the matrix Φ_{tot} obtains the form of (8.1). As we can also read from Φ_{tot}, x_1 (corresponds to the 1^{th} column of Φ_{tot}) contributes to CS measurements y_1, y_2, y_3, and y_4. Note that in Figure 8.1, we have shown the transmitting nodes with a dark color, while the rest of the nodes are shown by white color. The same procedure is performed for N_s source nodes selected *uniformly at random* to form Φ_{tot}.

$$\Phi_{tot} = \begin{pmatrix} \varphi_{1,1} & 0 & 0 & 0 & 0 \\ \varphi_{2,1} & \varphi_{2,2} & 0 & 0 & 0 \\ \varphi_{3,1} & 0 & \varphi_{3,3} & 0 & 0 \\ \varphi_{4,1} & 0 & 0 & \varphi_{4,4} & 0 \\ 0 & 0 & 0 & 0 & \varphi_{5,5} \end{pmatrix} \tag{8.1}$$

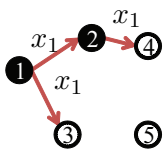


Figure 8.1 Network with $N = 5$ and n_1 transmitting x_1 employing PB.

8.1.2 Suitable Values of N_s and p

As shown in [55, 58], if a sparse Φ matrix has *at least* one non-zero placed *randomly* in each row and *independently* from other rows then reconstruction of a signal $\underline{\mathbf{x}}$ with Ψ of DCT requires the same *order* of number of measurements as a dense ideal Φ. In other words, the matrix Φ should be *full rank* (it should have M linearly independent rows). We need to find the suitable values of N_s and p such that the collected M rows of Φ_{tot} form a sparse Φ with the aforementioned properties while N_{tot}, the total number of transmissions for the N_s disseminations, is minimized. If $T_{PB}(p)$ denote the fraction of network nodes that perform the retransmission in a PB with forwarding probability p (see section 2.8), each PB transmission requires $T_{PB}(p)N$ transmissions. Therefore, we have $N_{tot} = T_{PB}(p)NN_s$.

Since N_s nodes select themselves uniformly at random, the placement of non-zeros in each row is also random. Therefore, we need to investigate the independence of non-zeros in rows of Φ. The entries $\varphi_{j,j}$ (representing sensors own readings) are not independent across rows since their location depends on where the data collector

gathers the M measurements. For instance, if the data collector queries M nodes with close proximity, n_j through n_{j+M}, we know exactly that $\varphi_{1,j}$ to $\varphi_{M,j+M}$ are non-zero. Therefore, these entries cannot satisfy the required placement independence of non-zeros in the rows of φ. As a result, let us assume each node does not add its own reading to its measurements unless it is a source node, and let the Φ' denote the resulting collected measurement matrix. Therefore, in Φ' the N_s disseminated readings (resulting in N_s almost dense columns) should assure that there are M independent rows in Φ'. Note that Φ has always greater or equal number independent rows than Φ' due to having non-zero entries for sensors own readings. Therefore, if Φ' satisfies the required condition of sparse measurement matrices, Φ indeed meets these criteria. In the following theorem we investigate the number of independent rows in Φ' as a function of N_s and p.

Theorem 8.1 *Let an $M \times N$ matrix Φ' be the measurement matrix obtained from Φ_{tot} in CStorage-P when sensors do not add their own reading to their measurement unless they are a source. Further, let $R_{PB}(p)$ be the fraction of nodes that receive a transmission using PB with forwarding probability p (see Section 2.8). $r(j)$, the* expected *number of independent rows of Φ' after the j^{th} transmission (out of N_s transmissions), is given by the following:*

$$\begin{aligned} r(0) &= 0, \\ r(j) &= 1 - (1 - R_{PB}(p))^{M - r(j-1)} \\ &\quad + r(j-1), j \in \{1, 2, \ldots, N_s\}, \end{aligned} \tag{8.2}$$

Proof. Clearly, if the network nodes receive a reading using PB uniformly then $R_{PB}(p)$ would also be the probability that a particular node receives the reading. However,

generally the dissemination is not uniform, e.g., nodes on the border and corners of network receive fewer readings. Let $R_{N_s}(p)$ denote the probability that a node receives all N_s transmissions. Clearly, in a uniform distribution $R_{N_s}(p) = R_{PB}(p)^{N_s}$. In [66, 67], authors have shown that for *non-uniform* dissemination we have

$$R_{PB}(p)^{N_s} \leq R_{N_s}(p) \leq R_{PB}(p),$$

which shows that it is more probable that a node receives all N_s transmissions when disseminations is non-uniform. Therefore, we can assume all nodes uniformly receive each PB with probability $R_{PB}(p)$.

Let $t(j)$ denote the probability that at least one independent row is added to Φ' after j^{th} PB of CStorage-P. Further, let $r(j)$ be the *expected* number of independent rows in Φ'. At the beginning, Φ' has no independent row; hence, we have $r(0) = 0$. When the first broadcast is performed, if at least one node out of M nodes of interest receives this broadcast one independent row is added to Φ'. Therefore, we have $t(1) = 1 - (1 - R_{PB}(p))^M$; hence, $r(1) = 1 \times t(1) = t(1)$. If at least one node has received the first transmission, the next broadcast should be received by one node out of $M-1$ nodes so that the number of independent rows increases to 2. Consequently, the second transmission should be received by at least one of the $M - r(1)$ nodes in expectation so that a new independent row is added to Φ'. As a result, we have $t(2) = 1-(1-R_{PB}(p))^{M-r(1)}$ and $r(2) = r(1)+1\times t(2)$. Similarly, $r(j)$ can be found recursively as $r(j) = r(j-1) + 1 \times t(j)$ with $t(j) = 1 - (1 - R_{PB}(p))^{M-r(j-1)}$.

■

As an example, we set $N = 10^4$ and $M = 700$ and employ Theorem 8.1 along with the values of $R_{PB}(p)$ given in Figure 2.6 to find the number of independent rows of Φ' versus p and N_s as shown in Figure 8.2. Note that the expected number of independent rows in Φ' is a lower bound on the number of independent rows in Φ.

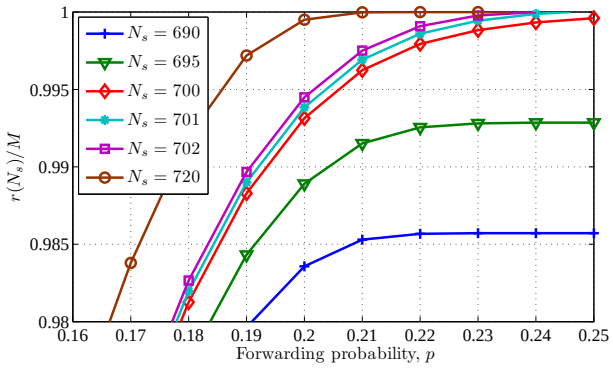


Figure 8.2 Normalized number of independent rows, $\frac{r(N_s)}{M}$, versus p and N_s.

Figure 8.2 shows that for $N_s \geq M$ the number of independent rows of Φ' approaches M for a large enough p and Φ' becomes full rank. More importantly, it shows that as N_s increases a suitable matrix can be generate with a *smaller* value of p. Consequently, we see an interesting *trade-off* since increasing N_s increases $N_{tot} = T_{PB}(p)NN_s$, while it reduces the required p and consequently $T_{PB}(p)$.

Using the results reported in Figure 8.2, we find N_{tot} for all values of p and N_s such that $\frac{r(N_s)}{M} \geq 0.9999$ as shown in Figure 8.3 (we have plotted N_{tot} versus $N_s - M + 1$ to have a better view on the values close to $N_s = M$ on a log-scale axis). Note that we have not fixed p to obtain the curve in Figure 8.3, but have rather relaxed the value of p and searched for the minimum N_{tot}.

The aforementioned trade-off between N_s and N_{tot} can be observed in Figure 8.3, and we can see that the number of transmissions is minimized when N_s is *slightly larger* than M. Figure 8.3 shows that N_{tot} is minimized for $N_s = 702$, which is obtained for $p = 0.24 = p^*$.

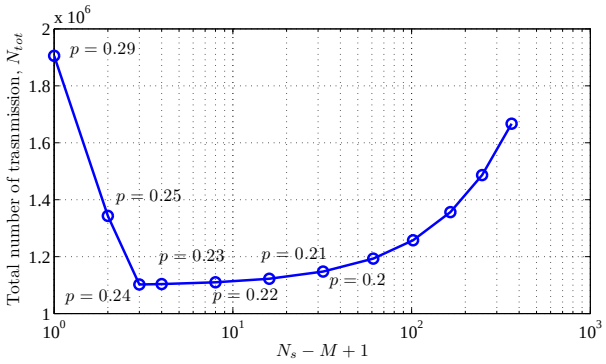


Figure 8.3 The total number of transmissions N_{tot} required to generate a suitable Φ' with $\frac{r(N_s)}{M} \geq 0.9999$ versus $N_s - M + 1$.

8.2 Compressive Data Storage in WSNs Employing CStorage-B

In this section, we propose a novel data dissemination algorithm referred to by *alternating branching* (AB) that is independent of network topology (has no parameter to tune). We will then employ AB for data dissemination in CStorage an propose CStorage-B in Section 8.2.6.

8.2.1 Issues with PB Algorithm

Consider the nodes in Figure 8.4, where n_t is about to rebroadcast a reading x_i (for instance using PB). Let $n_{t,p}$ be the *parent* of n_t, from which n_t has received x_i. Clearly, all nodes in $\mathcal{N}(n_{t,p})$ have received x_i, where $\mathcal{N}(n_{t,p})$ denotes the set of one-hop neighbors of $n_{t,p}$. When n_t performs the transmission, nodes in the gray shaded area receive x_i.

Clearly, to greedily minimize the total number of transmissions, the distance of n_t to $n_{t,p}$ (hence the size of gray area) should be maximized [170]. However, since in PB n_t *blindly* makes the forwarding decision regardless of its distance to $n_{t,p}$, it may be positioned close to $n_{t,p}$ and its transmission may be redundant. This is the first issue in PB that results in redundant transmissions.

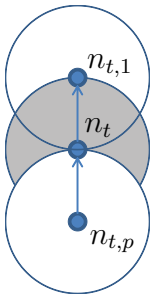


Figure 8.4 Structure of AB algorithm, where the current transmitter, n_t, selects one next transmitter, $n_{t,1}$.

Authors in [170] proposed to employ the location of nodes obtained by GPS to find a node n_t that has the maximum distance with $n_{t,p}$. However, GPS may be unavailable in many WSNs, while two hop-neighbor information can be easily obtained at nodes. Therefore, we design *alternating branching* dissemination algorithm that takes advantage of the two-hop neighbor information to find nodes that are *possibly the farthest* from the current transmitter exploiting their neighbor information. This resolves the first issue of PB.

The second issue with PB is that the *local density* of neighbors is not included in the calculation of p. Therefore, nodes in network corners, close to borders, and in sparse regions of network receive less number of transmissions. Although, there have been several work that propose to locally tune p, they still have a parameter that needs tuning based on network wide information. Authors in [171] propose *SmartGossip*, which has several parameters ($\gamma, T, \mu_1, \mu_2, \sigma$, and δ) that are tuned based on network parameters [171].

Authors in [172] proposed *Smart Gossip*, where nodes start forwarding disseminations with flooding (forwarding probability $p = 1$). As the time passes, each node computes the forwarding probability of its *parents* based on the number of its receptions. Next, all nodes inform they parents by suggested forwarding probability. Next, each node sets its forwarding probability as the maximum value suggested by

its children. Such an algorithm, needs several disseminations from a single source so that the forwarding probabilities are correctly tuned, while in CStorage each node disseminates its reading only once. Further, nodes in Smart Gossip [172] need to be aware of *network diameter*, which is varying and not always known to nodes.

Authors in [173] propose each parent to calculate the forwarding probability of its children based on the number of their parents obtained from *two-hop* neighbor information at nodes. When a node computes the forwarding probability of each one of its children, it requires the network diameter, which again is network dependent and not always known.

In [174], authors propose to locally select the number of next transmitters c. They shows that there is an an optimal value of c^* for which the dissemination becomes reliable (identical to p^* in PB). Therefore, scheme propose in [175] also has a parameter that needs to be tuned and is network dependent similar to p^*.

Therefore, these algorithms require a network wide information to locally tune the forwarding probability, which is not always possible. Consequently, we propose each transmitter to *select a fixed* number of *next* transmitter(s) (regardless of any network parameter) in AB To have a uniform dissemination throughout the network. This ensures that there are enough number of transmitters even in sparse areas of network, and results in *uniform* dissemination of x_i regardless of the density of nodes as we later see.

8.2.2 The Alternating Branching Design

Although, in large scale WSNs global routing tables may not be obtained, retrieving one-hop and two-hop neighbor information is simple and requires a small number of transmissions. If all nodes broadcast a *hello* message, every node obtains one-hop neighbor information. If all nodes broadcast the list of their neighbors following all hello messages of the first round, every node obtains two-hop neighbor information.

Clearly, this results in $2N$ transmissions in total.

Based on our discussions, in AB we propose a node n_t that is retransmitting x_i to be *responsible* to choose the next transmitter(s). Thus, n_t has been selected to be a transmitter by $n_{t,p}$. Assume only one next transmitter $n_{t,1}$ is chosen by n_t. Because all nodes in $\mathcal{N}(n_{t,p})$ have already received x_i, the next transmitter of n_t is selected from $\mathcal{N}(n_t)\backslash\mathcal{N}(n_{t,p})$ (nodes in the gray area of Figure 8.4).

The question here is how n_t can find the (possibly) farthest node using only two-hop neighbor information. Clearly, a neighbor of n_t that has the minimum number of common neighbors with $n_{t,p}$ is probably (and not necessarily) farthest node whose transmission results potentially in the largest new covered area. We emphasize that this is the best n_t can do to find the farthest node when only two-hop neighbor information is available. Consequently, n_t chooses the next transmitter $n_{t,1}$ such that

$$n_{t,1} = \underset{n_{t,l}}{\operatorname{argmin}} |\mathcal{N}(n_{t,l}) \bigcap \mathcal{N}(n_{t,p})|, \tag{8.3}$$

where $\backslash$ and $\bigcap$ denote the subtraction and intersection of two sets. We note that 8.3 greedily maximizes the distance of n_t with $n_{t,1}$ using the only available information at n_t, which is two-hop neighbor information. Therefore, there may be other nodes in the gray area of Figure 8.4 that have larger distance to n_t compared to $n_{t,1}$. However, $n_{t,1}$ is the farthest node n_t could find using the information available to it.

Ideally, $n_{t,1}$ is placed on the transmission border of n_t and on the straight line connecting n_t and $n_{t,p}$. Further in ideal case $n_{t,1}$ has only one common neighbor with $n_{t,p}$, which is n_t. We emphasize that we have shown the ideal setup for the sake of simplicity and in our actual implementation next hop is not necessarily on the edge of transmission range nor is on a straight line with $n_{t,p}$ (it is selected based on Equation (8.3)).

Consider a source node n_i that initiates the broadcast of x_i and assume all its neighbors rebroadcast x_i. If we allow these nodes to choose only one next transmitter,

and those transmitters to choose one transmitter again an so on, they will (ideally) form *straight* lines of transmitters that emanate from n_i and travel toward *borders*. Clearly, such a dissemination will be incomplete in the network. Therefore, some nodes should choose *more than* one next transmitter so that the transmitters *branch* and *multiply* (as the branches of a tree multiply) and x_i is well disseminated by an increase in the number of transmitters.

Consider selecting two next transmitters by the current transmitter n'_t, as depicted in Figure 8.5. We can see that as the number of next transmitters increases, the overlapping area of their coverage also increases, hence their transmissions becomes less efficient. Consequently, we propose to choose only two next transmitters when branching occurs. Let $n'_{t,1}, n'_{t,2} \in \mathcal{N}(n'_t) \backslash \mathcal{N}(n_{t,p})$ denote the two next transmitters. Similar to choosing one next transmitter $n_{t,1}$, we can possibly provide the largest new covered area by the transmission of $n'_{t,1}$, $n'_{t,2}$ when they have minimum number of common neighbors with each other and with $n_{t,p}$. Therefore, $n'_{t,1}$, $n'_{t,2}$ are selected such that

$$n'_{t,1}, n'_{t,2} = \underset{n_{t,1}, n_{t,2}}{\operatorname{argmin}} |\mathcal{N}(n_{t,1}) \bigcap \mathcal{N}(n_{t,2}) \bigcap \mathcal{N}(n_{t,p})|, \tag{8.4}$$

as shown in Figure 8.5. Note that using (8.4), n'_t chooses two neighbors that are potentially far, while they are not guaranteed to have the maximum distance to n'_t. Again, this is the best n'_t can do employing two-hop neighbor information to select the farthest nodes.

As discussed in Section 2.8, we have $r_t = O(\frac{\log N}{N})$; hence, a node has $O(\log N)$ neighbors from which the next transmitter(s) are selected. Therefore, finding a single next transmitter based on (8.3) is finding the maximum entry among $O(\log N)$ entries, which has complexity of $O(\log N)$. However, when (8.4) is employed to find two next transmitters, each node searches among $O(\log^2 N)$ combinations of its neighbors. Finding common neighbors between three particualr nodes each having $O(\log N)$

neighbors, requires $O(\log^2 N)$ operations. Therefore, selection of the next transmitters based on (8.4) has the total complexity of $O(\log^4 N)$. Consequently, AB has the total complexity of $O(\log^4 N)$. We emphasize that the processing complexity is usually not a concern in WSN nodes since the processing unit power consumption is much smaller than power consumption by radio transceivers and actuators [176]. Therefore, the main goal in designing CStorage is reducing the total number of transmissions.

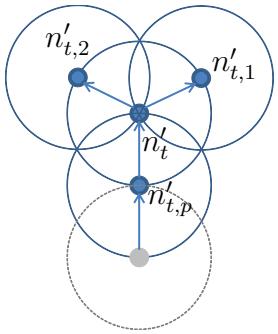


Figure 8.5 Structure of AB algorithm, where the current transmitter, n_t, selects two next transmitters, $n_{t,1}$ and $n_{t,2}$.

The branching should occur frequently in *random* networks to ensure enough new branches are produced to explore new uncovered areas especially when nodes are sparse. Therefore, we propose to branch at every other transmitter. To control the branching, we propose to include a *single-bit binary counter* as *branching flag* along with x_i. When n_t wants to broadcast, it first checks the branching flag. If the flag is 0, n_t chooses one next transmitter and two otherwise. Next, it *flips* the flag, and rebroadcasts x_i along with the ID of the next transmitter(s) and the branching flag. Clearly, if a node is selected as the next transmitter of x_i but it has received it before, the branch has been chosen from an area where x_i has already been disseminated. Therefore, this transmission is redundant and is ignored.

In Figure 8.4, n_t receives x_i with flag 0, and chooses $n_{t,1}$. n_t flips the flag to 1 and performs the transmission. Hence, $n_{t,1}$ now becomes the new transmitter n'_t. Since the flag is 1, it performs the branching and chooses two next transmitters $n'_{t,1}$ and

$n'_{t,2}$. n'_t flips the flag back to 0 and rebroadcasts x_i along with the ID's of $n'_{t,1}$ and $n'_{t,2}$. Since the branching flag is 0 now, $n'_{t,1}$ and $n'_{t,2}$ choose only one next transmitter and so on. Therefore, we refer to our algorithm by *alternating branching* (AB). n_i initiates the broadcast of x_i with branching flag of 0. In Figure 8.6, we have shown the dissemination of one reading using AB with the source node located in the center of a $A = 1 \times 1$ network with $N = 10^4$ nodes at four different progressive time snaps until AB is completed.

In Figure 8.6, we can see that branches emanate from the source and are spread towards borders. However, due to random placement of nodes the may not move straightly toward edges. Further, we can see that branches may arrive at the same node after a few steps and terminate. Further, we can see that the nodes that have not received the transmission are well distributed throughout the network. In the next section, we provide analysis of AB on a grid network and evaluate the fraction of nodes that perform the transmission and receive the transmission.

8.2.3 Analysis of AB on Grids

Let us first investigate AB in an *ideal grid* setup. If we repeat the ideal pattern of transmitting nodes shown in Figures 8.4 and 8.5, they form an *isometric* grid network shown in Figure 8.7. We should note that isometric grids have been previously considered in WSNs [177]. It is easy to see that the transmitters form *hexagon cells*.

Let $r_g \in \{1, 2, \ldots\}$ denote the transmission range of nodes on the isometric grid as multiples of grid-size (in Figure 8.7, we have $r_g = 1$ and $r_g = 2$ on left and right, respectively). Following the solid black nodes in Figure 8.7, we can see that two branches arrive at the same node (shown by a cross) after several steps and they merge into one branch. Further, the nodes that have been shown by hollow circles receive the reading but are not selected as next hop forwarders. Finally, the center nodes that have been shown by squares do not receive the transmission from any

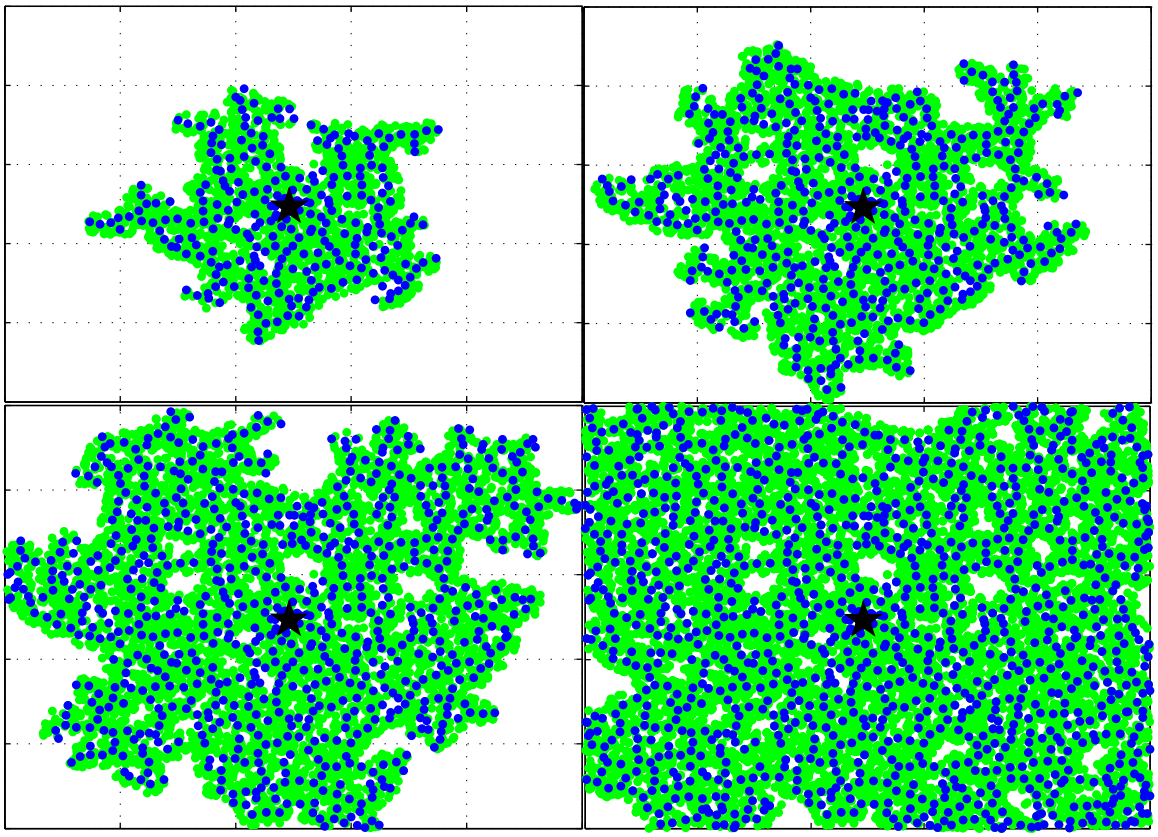


Figure 8.6 Dissemination of a reading from the source node at the center (shown by a star) using AB. The dark colored nodes are the transmitters forming branches, the light colored nodes are the nodes that receive the reading, the white areas are the nodes that do not receive the transmission. Figures belong to the same dissemination in progressive snap times from left to right and up to down, until the dissemination is complete.

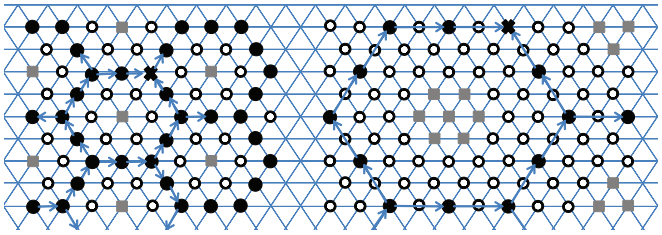


Figure 8.7 Ideal implementation of AB that results in isometric grid. The transmitters are shown with filled black circles, nodes that receive the transmission but do not retransmit are shown by hollow circle, the nodes that do not receive the transmission are shown by gray square (in the center of hexagons formed by transmitters), and arrows show the progress direction of the branch. Clearly, transmitters form hexagon shaped cells. The left and the right figures show the grid when the transmission range is one and two grid size, respectively.

node.

We may simply formulate the fraction of receivers and transmitters in ideal AB on isometric grid. Since the whole network has the same hexagon shaped cells, the fraction of nodes that transmit and receive are equal for a cell and the whole network. The transmitters around a hexagon also belong to its neighboring hexagons too, while the nodes inside a hexagon only belong to one cell. Using Figure 8.7 and the discussion provided, the number of nodes that solely belong to one hexagon N_H and the number of nodes that do *not* receive a transmission in one hexagon N_{NR} are given by

$$N_H = 1 - 6r_g + 6\sum_{i=1}^{2r_g} i \text{ and } N_{NR} = 1 + 6\sum_{i=1}^{r_g-1} i. \tag{8.5}$$

Using (8.5), we find the fraction of nodes that receive a transmission, R_g, and the fraction of nodes that perform the transmission, T_g, in a grid network in the following lemma.

Lemma 8.1 *In ideal AB on an* isometric grid*, for transmission range* $r_g \in \{1, 2, \ldots\}$*, the fraction of nodes that receive the transmission,* R_g*, and the fraction of nodes that*

perform the transmission, T_g, is the same for one hexagon and the whole network. Therefore, we have

$$R_g = \frac{N_H - N_R}{N_H} \text{ and } T_g = \frac{6}{N_H}. \tag{8.6}$$

We also employ Monte-Carlo numerical simulations to find the average fraction of receivers, R_r, and transmitters, T_r, when the deployment of nodes is random with $N = 10^4$. Further, to perform a comparison with existing work, we assume nodes in the random network are equipped with GPS [170], and also propose a second implementation of alternating branching algorithm referred to by AB_{GPS}. In AB_{GPS}, when the flag is 0 one next transmitter is selected from $\mathcal{N}(n_t)\backslash\mathcal{N}(n_{t,p})$ such that it is the farthest node to $n_{t,p}$. Moreover, when the flag is 1 two next transmitters from $\mathcal{N}(n_t)\backslash\mathcal{N}(n_{t,p})$ are selected such that the total pairwise distance of $n_{t,1}$, $n_{t,2}$, and $n_{t,p}$ is maximized. Such a selection also formes the structures shown in Figures 8.4 and 8.5. We denote the average fraction of receivers and transmitters in AB_{GPS} by R_{GPS} and T_{GPS}, respectively.

We increase the transmission range r_t from its minimum value 12, i.e., threshold of r_t for which network becomes disconnected (as discussed in Section 2.8), to large values where nodes are *densely* connected. The number of neighbors in isometric grid cannot take all values in contrast to random networks and is given by $6\sum_{i=1}^{r_g} i \in \{6, 18, 36, \ldots\}$. Figure 8.8, compares R_r, R_g, T_r, and T_g. R_r and T_r are plotted versus the average number of neighbors since AB has no parameter to tune.

Figure 8.8 shows that AB provides almost *constant* fraction of receivers and transmitters despite drastic changes in network topology. Therefore, if the network changes over time AB *automatically* adapts to changes. This is in contrast to PB where R_{PB} and T_{PB} are greatly affected by p. In addition, from Figure 8.8 we can observe that although AB performs very close to AB_{GPS} (ideal setup), while it eliminates the need for GPS information.

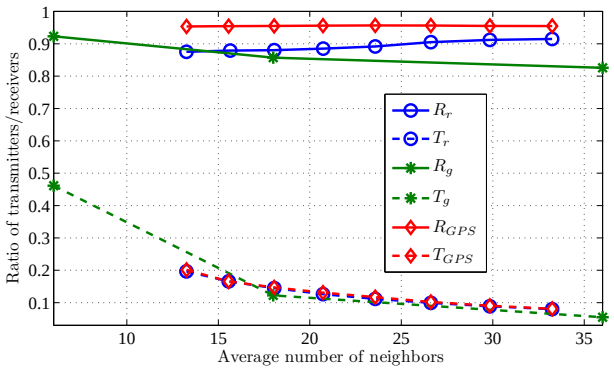


Figure 8.8 The average fraction of nodes receiving and transmitting in a dissemination for AB. R_r, R_g, and R_{GPS} denote the fraction of receivers in AB, AB on isometric grid, and AB_{GPS}, respectively. Further, T_r, T_g, and T_{GPS} denote the fraction of transmitters in AB, AB on isometric grid, and AB_{GPS}, respectively.

We can also observe that as the transmission range increases, AB becomes more efficient and T_r reduces while R_r increases, and its performance approaches that of AB_{GPS}. However, in ideal grid network the fraction of receivers drops as the networks becomes dense. In addition, Figure 8.8 shows that isometric grid analysis of AB can provide close estimates for R_r and T_r. This shows that AB performs close to grid model on random networks although the neat hexagon shaped cells may not appear due to random placement of nodes.

8.2.4 Distance Between Transmitters in Random Networks

In the grid model of AB, we considered next hops to be placed on the border of transmission range of n_t, which is not the case in random networks. We are interested in the expected distance of n_t with the next hops in random networks. Let $\bar{d}$ denote the *expected* distance between n_t and the next forwarders, which we are interested to be the farthest neighbor. Clearly, we are interested in having $\bar{d} = r_t$, i.e., next transmitter is as far as possible and located on the border of transmission.

As mentioned earlier, authors in [170] employ the exact location of all neighbors to select the farthest node as next transmitter using GPS. However, AB only uses the two-hop neighbor information (which is a very limited information compared to exact Euclidian location obtained from GPS) to perform a similar task. We are interested to know how close AB performs to the ideal case where full nodes Euclidian location is available. Therefore, in Figure 8.9 we compare $\bar{d}$ in AB and the ideal case employing extensive numerical simulations.

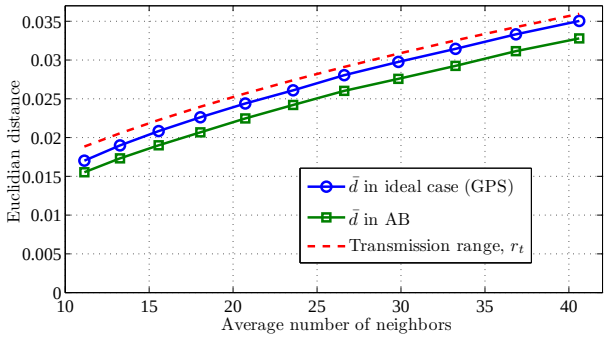


Figure 8.9 The expected farthest node distance $\bar{d}$ to n_t in AB and ideal case using full GPS information, shown along with the transmission range r_t.

Figure 8.9 confirms that AB can perform very close to ideal case with full nodes location knowledge. Therefore, we may assume AB is maximizing the distance of the next transmitter to n_t. Using this result, we can analytically find $\bar{d}$. For the sake of simplicity, let us assume the transmission range of n_t is unit, i.e., $r_t = 1$, and find $\bar{d}$. Let X_i be a random variable indicating the Euclidian distance of n_t with a neighbor in $\mathcal{N}(n_t)$. The pdf and cdf of X_i are given by $f_{X_i}(d) = 2d, 0 \leq d \leq 1$ and $F_{X_i}(d) = d^2$ [170]. The following lemma gives the pdf of a random variable defined as maximum of several random variables.

Lemma 8.2 *Let $X_i, i = \{1, 2, \ldots, k\}$ be i.i.d. random variables with the same cdf $F_X(d)$, and let the random variable $X_{max} = \max\{X_1, \ldots, X_k\}$. $F_{X_{max}}(d)$ the cdf of*

X_{max} is $F_{X_{max}}(d) = F_X^k(d)$.

Proof. $F_{X_{max}}(d) = P(X_{max} \leq d) = P(X_1 \leq d, \ldots, X_k \leq d) = P(X_1 \leq d) \ldots P(X_k \leq d) = F_X^k(d)$. ■

Node n_t maximizes the distance of the next forwarders from the set $\mathcal{N}(n_t) \backslash \mathcal{N}(n_{t,p})$ located in the gray area in Figure 8.4. The size of the shaded region is $A_{sel} = \frac{\bar{d}}{2}\sqrt{4-\bar{d}^2} + 2\arcsin\frac{\bar{d}}{2}$ [170].

The number of nodes in the shaded area is given by $N_{sel} = \frac{N}{A}A_{sel} = \rho A_{sel}$, where $\rho = \frac{N}{A}$ is the density of nodes. Let $X_{max} = \max\{X_1, \ldots, X_k\}$ be the random variable denoting the distance of next transmitters to n_t. Using Lemma 8.2, we have $F_{X_{max}}(d) \approx d^{2N_{sel}}$. Consequently, the expected distance $\bar{d}$ is simply obtained by $\bar{d} = E[X_{max}]$, where $E[.]$ denotes the expected value of a random variable. The expected value of a random variable Z can be calculated from its cdf $F_Z(x)$ by $E[Z] = \int_0^\infty (1 - F_Z(x))dx - \int_{-\infty}^0 F_Z(x)dx$. This gives

$$\begin{aligned}
\bar{d} = E[X_{max}] &= \int_0^1 (1 - z^{2N_{sel}})dz, \\
&= 1 - \frac{1}{2N_{sel}+1} \\
&= 1 - \frac{1}{2\rho\left[\frac{\bar{d}}{2}\sqrt{4-\bar{d}^2} + 2\arcsin\frac{\bar{d}}{2}\right] + 1}
\end{aligned}$$

After a few simple mathematical operations, we obtain

$$\rho = \frac{\bar{d}}{(1-\bar{d})(\bar{d}\sqrt{4-\bar{d}^2} + 4\arcsin\frac{\bar{d}}{2})}. \tag{8.7}$$

The value of $\bar{d}$ may be obtained from (8.7) for any ρ. For instance, at average number of neighbors equal to 22 we have $\bar{d} = 0.963$, and in the worst case for almost disconnected network (average neighbor number of 12), we have $\bar{d} = 0.93$. Therefore, the assumption that next forwarders are placed on the transmission range border of

n_t in grid networks is not far from reality in random networks. Therefore, R_g and T_g may provide close estimates of R_r and T_r as shown in 8.8.

8.2.5 Dissemination Uniformity

As mentioned earlier, one of the main shortcomings of PB is that nodes that are close to border and especially nodes located in the corners do not receive disseminations as uniformly as nodes located in the center of the network due to having less number of neighbors. This results in *nonuniform* data dissemination in PB. However, as we observed in Figure 8.8, in AB the fraction of receivers and transmitter remains almost constant independently from the number of neighbors.

First, assume the data collector queries the M nodes located in the center of the network to obtain M measurements. These nodes will experience the best disseminations due to their centrality in the network. Let R_{cen} denote the average fraction of these M nodes that receive a particular transmission. Next, assume the data collector gathers M measurements from M nodes in a network corner, and let R_{cor} denote the fraction of these M nodes that receive the same transmission.

To evaluate the dissemination uniformity of AB and PB, let us define *dissemination uniformity* $\mu = E[R_{cen} - R_{cor}]$. Clearly, we are interested in a uniform dissemination, which results in $\mu \approx 0$, i.e., nodes in the corner receive the disseminations with the same probability as the nodes in the center of the network. We find μ for PB and AB using extensive numerical simulations in Figure 8.10 for a network with $N = 10^4$ nodes and $M = 700$. We increase the transmission range from the $r_t = 0.020$ (almost disconnected) to $r_t = 0.034$ (heavily connected) and find the average number of neighbors. In PB, for each transmission range we set $p = p^*$ for $R_{PB} \approx 0.7$, i.e., 70% of the network node receive the transmission. The values of p^* for various r_t's in PB are given in Table 8.1. To perform a comparison between these two algorithms, we have also depicted T_r and T_{PB}, the fraction of nodes that perform the transmission

in AB and PB, respectively.

Table 8.1 Transmission range r_t, and the corresponding average number of neighbors and p^*.

r_t	0.021	0.022	0.024	0.026	0.027	0.029	0.031	0.033	0.034
No of neigh.	13	16	18	21	24	27	30	33	37
p^*	0.38	0.32	0.28	0.25	0.22	0.19	0.17	0.16	0.14

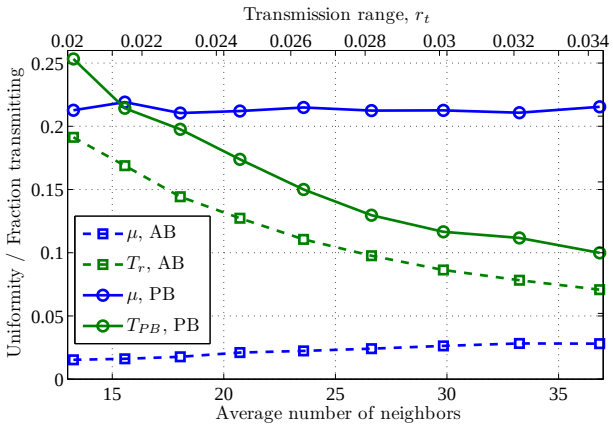


Figure 8.10 Dissemination uniformity, μ, and the fraction of nodes that transmit in PB, T_{PB}, and in AB, T_r, versus r_t and average number of neighbors in a random network.

Figure 8.10 confirm that the dissemination of AB is well uniform and almost the same at the corners compared to the center of the network. In contrast, PB may not provide a uniform dissemination at the corners. We can see that a uniform dissemination is obtained only for very large values of p where the number of transmission is too high. The initial increase in μ for PB is due to lack of dissemination of readings where readings are neither disseminated in corners nor in the center. However, as analyzed earlier we expect PB to disseminate readings effectively for $p \approx p^*$ where

μ is too large compared to AB. The higher dissemination uniformity along with independence from network parameters as shown in Figure 8.8, makes AB a suitable dissemination algorithms in WSNs not limited to CStorage.

8.2.6 CStorage-B Design

Similar to CStorage-P, in CStorage-B node $n_j, j \in \{1, 2, \ldots, N\}$, maintains a CS measurement y_j and after dissemination $\Phi_{tot}^{N\times N}$ is formed in the network, except that AB replaces PB for data dissemination purpose. Consequently, the steps of CStorage-B are as follows.

1. All nodes choose $\varphi_{j,j}$ from $\mathfrak{N}(0, 1)$ and initialize their measurement to $y_j = \varphi_{j,j}x_j$.

2. N_s nodes randomly select themselves as a source node and broadcast their reading to their neighbors with the single-bit flag set to 0.

3. Upon the reception of x_i for the *first time* by node l, n_l, it performs the following:

   (a) Chooses $\varphi_{l,i}$ from $\mathfrak{N}(0, 1)$ and adds $\varphi_{l,i}x_i$ to y_l.

   (b) Checks to see if it has been selected as a next forwarder or is a direct neighbor of source node, n_i. If either of aforementioned conditions are met, it checks x_i's single-bit flag. If the flag is 0, it chooses one next transmitter using (8.3), or otherwise chooses two next transmitters using (8.4). Finally, it flips the single-bit flag and *rebroadcasts* x_i along with the flag and the ID of the next forwarder(s) (AB).

After the transmissions are finished, N_s readings will be disseminated throughout the network. Similar to CStorage-P, a data collector queries M measurements $\underline{\mathbf{y}}$ and the corresponding ϕ_j's from an arbitrary set of M nodes and obtains the measurement matrix Φ (which is subset of Φ_{tot}) and obtains $\underline{\hat{\mathbf{x}}}$. We may rewrite Theorem 8.1 for

CStorage-B to find the expected number of independent rows in Theorem 8.2 for N_s disseminations. Similar to CStorage-P, let Φ' be the measurement matrix formed in CStorage-B when nodes add their own reading if and only if they are a source.

Theorem 8.2 .

Let an $M \times N$ matrix Φ' be the measurement matrix obtained from Φ_{tot} in CStorage-B when sensors do not add their own reading to their measurement unless they are a source. Further, let R_r be the fraction of nodes that receive a transmission using AB on a random network. $r(j)$, the expected *number of independent rows of Φ' after the j^{th} transmission (out of N_s transmissions), is given by the following:*

$$\begin{aligned} r(0) &= 0, \\ r(j) &= 1 - (1 - R_r)^{M - r(j-1)} \\ &\quad + r(j-1), j \in \{1, 2, \ldots, N_s\}, \end{aligned} \tag{8.8}$$

Proof. Proof is similar to proof of Theorem 8.1, except that the fraction of nodes that receive a dissemination is given by R_r. ■

Employing Theorem 8.2 along with R_r values reported in Figure 8.8 we plot the normalized number of independent rows after N_s disseminations ($\frac{r(N_s)}{M}$) in Figure 8.11. Figure 8.11, shows that (similar to CStorage-P) N_s needs to be slightly larger than M to form a measurement matrix Φ with M independent rows (becomes full rank).

In the following section, we evaluate the performance of CStorage-P CStorage-B, and compare them with existing schemes.

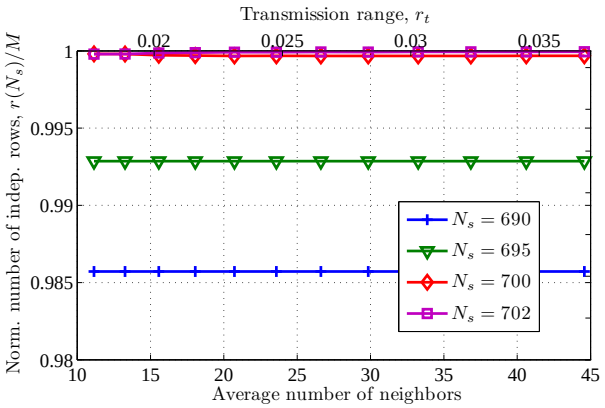


Figure 8.11 Normalized number of independent rows, $\frac{r(N_s)}{M}$, in CStorage-B versus N_s and the average number of neighbors.

8.3 Performance Evaluation

In this section, first we discuss the signal coefficients reordering to form a compressible signal. Next, we evaluate the performance of CStorage-P and CStorage-B using simulations and show that they can both reduce the total number of transmissions for data storage compared to existing algorithms.

To perform the numerical simulations we employ the real temperature readings data sets from EPFL's SensorScope project, LUCE deployment [163]. We capture a snapshot of the network temperature on 5/1/2007 at 12:1 as shown in Figure 8.12. We will have $N = 10^4$ nodes randomly deployed $A = 1 \times 1$. In PB, we set $r_t = 0.025$, and in AB we vary r_t from 0.02 (almost disconnected) to 0.038 (heavily connected).

We employ the normalized reconstruction error defined by $e = \frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_2}{\|\hat{\mathbf{x}}\|_2}$ to evaluate the reconstruction accuracy, where $\|.\|_2$ denotes the *norm* 2 of the signal. The selection of M depends on the target reconstruction error of the signal $\underline{\mathbf{x}}$. Clearly, $e = 0$ denotes perfect recovery. Without loss of generality, we set the target error to $e_t = 0.09$ (while any other target e may be chosen). Employing dense Φ matrices, we observe that $M = 2 \times 10^3$ results in average reconstruction error of $e \approx 0.085$. Therefore, we fix the number of measurements to $M = 2 \times 10^3$. Clearly, a smaller e_t

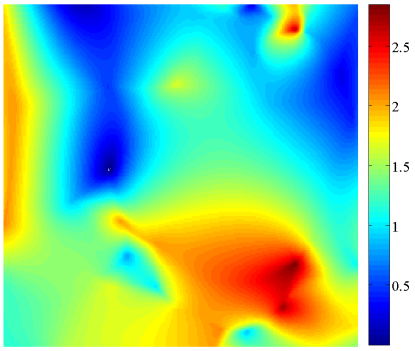


Figure 8.12 The captured snapshot of sensors temperature readings in EPFL's SensorScope project, LUCE deployment [163] on 5/1/2007 at 12:1.

necessitates choosing a larger M.

8.3.1 Signal Reordering

In most of the practical settings, N sensor nodes are deployed randomly in an area A, e.g., they are thrown from an airplane. Hence, the nodes ID will be irrelevant to their location. For instance, node n_1 may be located close to n_N. Clearly, the coefficients of a signal obtained by putting these readings together according to the ID of their collecting node is a *random* signal and is not compressible due to lack of *spatial correlation* between coefficients. Clearly, to take advantage of spatial correlation of nodes reading, the coefficients that are placed together in $\underline{\mathbf{x}}$, should also represent the readings from nodes that are located in *close physical proximity* in the network. Despite its high importance in WSNs data collection, only few existing work have addressed this issue and most work have assumed that the obtained signal is previously reordered and compressible. Note that signal reordering is basically finding a mapping from N nodes' ID to N signal coefficient indices.

Authors in [178] proposed to take advantage of the spatial correlation of readings so that the readings of a node over time can be reconstructed using fewer measurements exploiting neighboring nodes readings. This scheme considers the temporal

correlation rather than spatial correlation; hence, it may not be applied to signal obtained at a certain time instance.

Authors in [179], proposed a reordering algorithm referred to by *SOPerm* that finds a permutation of a *given* signal coefficients such that it has a sparse representation in desired sparsifying basis Ψ. Authors propose a greedy algorithm to perform the reordering. In SOPerm, signal $\underline{\mathbf{x}}$ is assumed to be *known*, while in CStorage measurements are known and the signal $\underline{\mathbf{x}}$ is to be found.

The naïve method that one would come up with first is splitting the network area into very narrow *strips* of the same width as proposed in [167] (and is referred to by *Horz-diff*). The strips contain nodes that have correlated readings when looked along the strip. All the strips are put back to back and the two dimensional area A is transformed into a one dimensional signal [167]. However, in this algorithm the width of the strips needs to be tuned based on signal properties and the density of nodes. For example, if the readings of the nodes in A are very slowly varying, wide strips may be the best choice. Clearly, such a reordering scheme requires the signal information, which is unknown to data collector.

As mentioned earlier, a signal $\underline{\mathbf{x}} = [x_1 x_2 \dots x_N]^T$ becomes more compressible if coefficients that are placed together in the signal are captured from nodes that are also physically placed in close proximity so that the coefficients have high *spatial correlation* [167]. In other words, the readings from nodes with close physical location should have as close as possible coefficient indices number.

The simplest solution to this problem is to start from a random node and iteratively add the index of the next *closest* node that has not been previously included in $\underline{\mathbf{x}}$. However, when many nodes have been *greedily* added, the reminding nodes may be far apart in the network; hence, the last coefficients will not be correlated. This results in a less compressive signal. To solve this problem, we need an intelligent algorithm that avoids the far nodes to be added together and at the same time consecutive

coefficients are selected so that their respective nodes are in close proximity.

A closer look at this problem reveals that this is a well-known problem in graph theory called *traveling salesman problem* (TSP), where a salesman needs to travel through N cities starting from a random one such that each city is visited once, close cities are visited together, and the total distance traveled is minimized. Therefore, there is a one-to-one mapping from our reordering problem to TSP, where cities are network nodes and the selected route is the suitable reordering of the signal coefficients. The reordering of a WSN reading based on TSP will be independent of Ψ and will be only based on spatial correlation of readings.

Therefore, we may employ TSP solvers to find a suitable reordering by mapping N nodes to N cities. Finding the optimal solution of TSP is an NP-complete problem [180]. However, there are numerous heuristics and greedy algorithms that approximately solve TSP problem in linear time. We will employ Lin-Kernighan (LK) heuristic [181] that finds close to optimal routes for TSP and offers a low complexity. The complexity of LK heuristic has been shown to be $O(N^{2.2})$ [182]. It is important to note that this algorithm need to run only once at the data collector when the network topology changes; hence, no computational burdens is imposed on network nodes.

We pictorially compare Horz-diff [167], greedy ordering, and ordering based on TSP using LK heuristic for a small random network with $N = 100$ nodes in Figure 8.13. We divide the area into 10 strips in Horz-diff. Figure 8.13, shows that LK can find a better reordering so that reading of nodes with close proximity are mapped to closer coefficients in $\underline{\mathbf{x}}$, and we can see huge jumps in the greedy algorithm that results in less compressible signal. Note that in all these algorithms the data collector should be aware of nodes location to perform the reordering and achieve a spatially correlated signal. We emphasize that knowledge of nodes' location of nodes is the basic assumption in CStorage since otherwise it is purposeless to obtain a readings

of an unknown location.

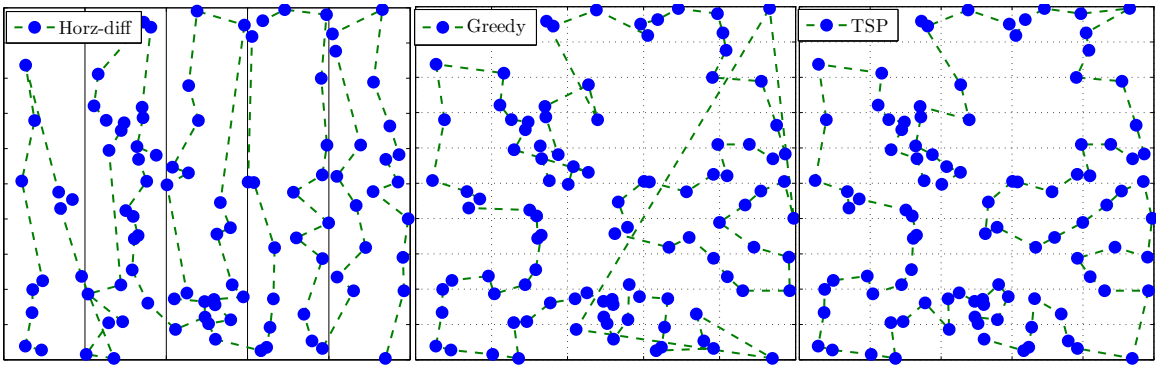


Figure 8.13 Various signal reordering algorithms to realize spatially correlated signals. From left to right, Horz-diff, greedy ordering, and TSP ordering using LK heuristic [181]. The dashed line shows the order of nodes in the signal $\underline{\mathbf{x}}$.

To compare the efficiency of these reordering algorithms, we employ them to reorder the real signal from LUCE Deployment with $N = 10^4$. We employ a dense $\Phi^{M \times N}$ measurements matrix with $M = 2 \times 10^3$. Among the reordering algorithms, Horz-diff is the only one that has a parameter, number of strips, to tune. Therefore, we plot e for all algorithms versus number of strips in Figure 8.14.

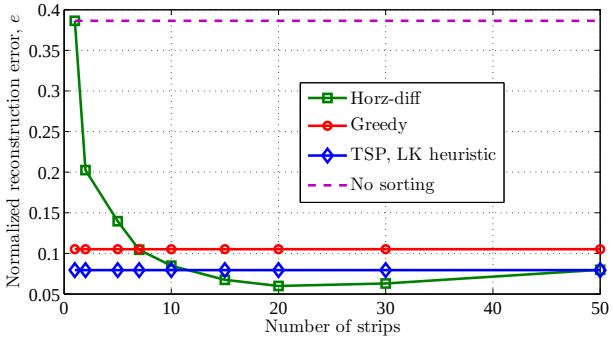


Figure 8.14 The reconstruction error of various reordering algorithms, Horz-diff, greedy ordering, and TSP ordering using LK heuristic [181]. The dashed line shows e when no reordering is performed and coefficients are sorted based on their respective node ID.

Figure 8.14 shows that Horz-diff can achieve the best reconstruction accuracy,

which shows a better reordering of the signal. However, as mentioned earlier the optimal number of strips is signal and area dependent and is cannot be globally optimized for all signal and fields. Nevertheless, it may represent the lower bound on reconstruction error of our signal, e, based on its spatial correlation. In addition, we can observe that the reordering of our signal based on TSP and LK heuristic provides the next best reconstruction, which is close to the minimum achievable e using Horz-diff. We remind that sorting based on TSP is independent from signal characteristics and only depends on nodes location. Therefore, reordering of the signal based on TSP using LK heuristic is an efficient reordering algorithm that is independent of the signal properties. Hence, we employ TSP to perform the reordering of the signal in our simulations in the next section.

8.3.2 Performance Evaluation of CStorage-P and CStorage-B

Based on our design for CStorage-P and CStorage-B, we should emphasis that the dissemination phase (employing PB and AB) forms non-zero entries in the columns of Φ corresponding to the N_s source nodes. Therefore the properties of the Φ matrix is determined by the dissemination phase. In CStorage-P, the forwarding probability p determines the fraction of nodes that receive a reading and determines the total number of transmissions. Based on our discussion in Section 2.8, p should be accurately tuned so that a high ratio of nodes receive the transmission while the minimum number of transmissions is incurred. In our simulations, we set $p = p^*$ and investigate CStorage-P in a randomly deployed network to see the suitable value of N_s for which the desired Φ is constructed and e_t is achieved.

Similarly, in CStorage-B the columns of Φ are formed by disseminating the N_s readings using AB. We will investigate CStorage-B in the same random network as CStorage-P to find the appropriate value of N_s for which the desired Φ is constructed. As N_s increases, the number of columns containing non-zeros (other than the diagonal

entry) increases; hence, $N_s = N$ results in generating a dense Φ. Nevertheless, such a density is not required as discussed earlier and shown by our simulations.

We implement CStorage-P and CStorage-B, and find the reconstruction error, e, by running a large number of iterations of data dissemination on randomly deployed networks. We also run these algorithms on various random network deployments; thus, we perform signal reordering based on TSP when the network changes. To run the simulations for CStorage-P, first we set $r_t = 0.025$ and plot e and N_{tot} versus various values of p as shown in Figures 8.15 and 8.16. Next, we vary r_t and set $p = p^*$ based on Table 8.1 and plot e and N_{tot} versus average number of neighbors and r_t in Figures 8.17 and 8.18. Similarly, Figures 8.19 and 8.20 show e and N_{tot} in CStorage-B versus average number of neighbors. Throughout the simulations, we consider the *worst case* and assume that the data collector is collecting the M readings from one of the corners in the field.

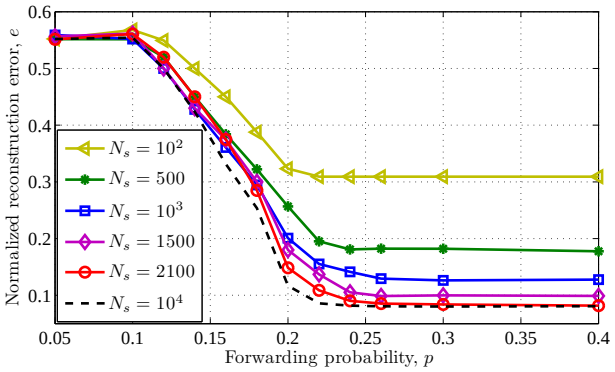


Figure 8.15 The reconstruction error e versus p in CStorage-P.

Figure 8.15 confirms that with N_s slightly larger than $M = 2 \times 10^3$, the desired e_t is achieved. Increasing N_s further does not improve the signal reconstruction accuracy while it considerably increases the number of transmission as shown in Figure 8.16. Further, Figure 8.15 shows that regardless of the value of N_s full reconstruction is impossible for $p < 0.24 \approx p^G$. This shows that for $p < 0.24$ readings are not disseminated in the network. Figure 8.18, shows that if the value of p is set to p^* for

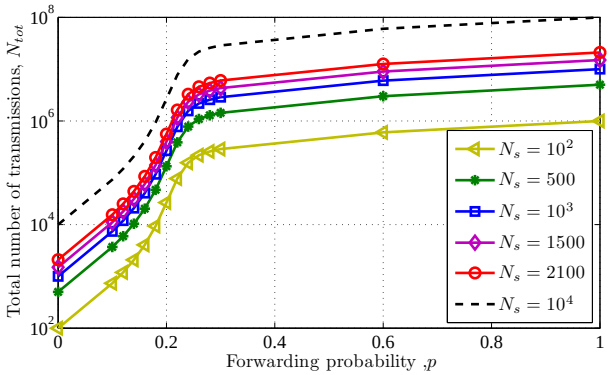


Figure 8.16 The total number of transmissions N_{tot} versus versus p in CStorage-P.

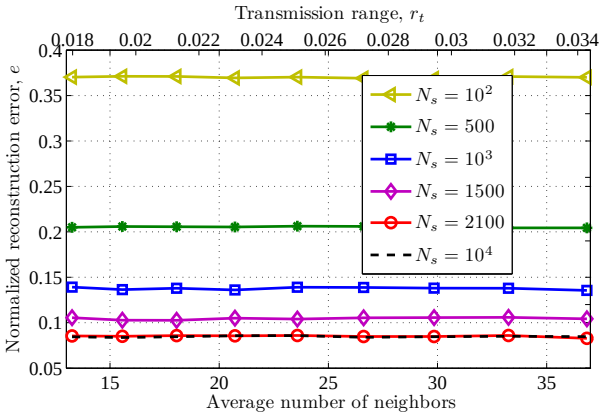


Figure 8.17 The reconstruction error e versus average number of neighbors and r_t in CStorage-P.

each network setup, CStorage-P can achieve the desired e_t for various network setups.

Similarly in Figure 8.19, we can see that for $N_s = 2100$ target reconstruction error $e_t = 0.09$ has been achieved. Clearly, similar to CStorage-P increasing N_s further does not contribute to the reconstruction quality of the signal. This figure confirms that AB is a general and parameterless dissemination algorithm and its performance is independent of the network parameters. This confirms our theoretical results from Theorems 8.1 and 8.2 that for N_s slightly larger than M a suitable measurement matrix Φ in both CStorage-P and CStorage-B.

We can see that in CStorage-P with $N_s = 2100$ and $M = 2 \times 10^3$, for average

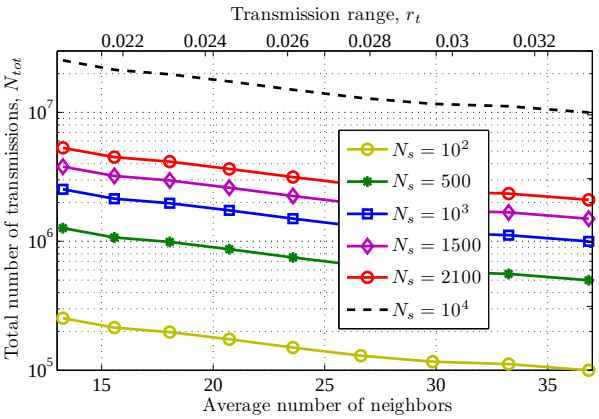


Figure 8.18 The total number of transmissions N_{tot} versus average number of neighbors and r_t in CStorage-P.

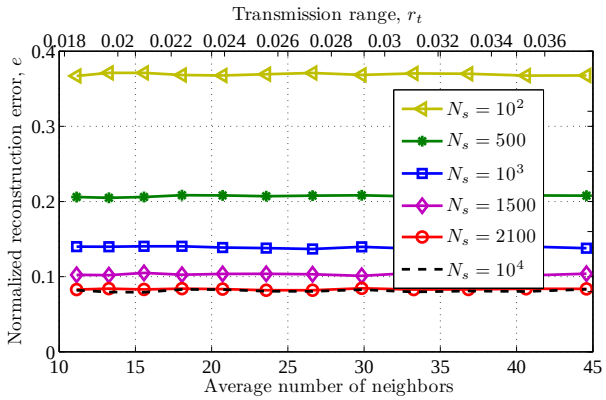


Figure 8.19 The reconstruction error e average number of neighbors and r_t in CStorage-B.

number of neighbor of 13 (minimum number for connectivity) and 37 (densely connected), we have $N_{tot} = 5.31 \times 10^6$ and $N_{tot} = 2.1 \times 10^6$, respectively, to achieve the desired e_t. For the same network structures CStorage-B requires $N_{tot} = 4.68 \times 10^6$ and $N_{tot} = 1.19 \times 10^6$, respectively. AB requires $2N = 2 \times 10^4$ transmission for hello messages to obtain two-hop neighbor information. This increases N_{tot} to $N_{tot} = 4.7 \times 10^6$ and $N_{tot} = 1.21 \times 10^6$. Therefore, CStorage-B decreases N_{tot} by *at least* 11.8%, while it can *automatically* match to network changes.

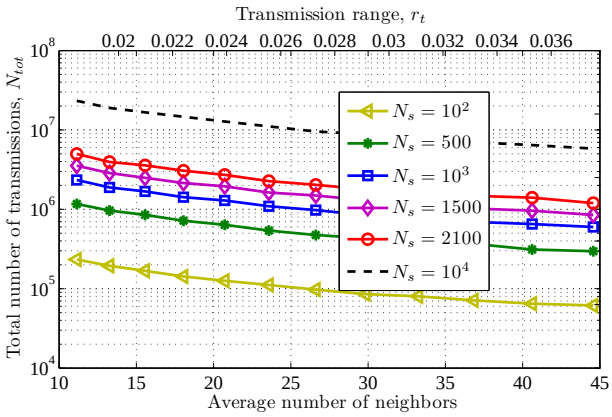


Figure 8.20 The total number of transmissions N_{tot} versus average number of neighbors and r_t in CStorage-B.

8.3.3 Comparison with Existing Algorithms

CStorage-P and CStorage-B may be compared to existing algorithms that do not need routing tables and are distributed. To the best of our knowledge, there are three such stateless data dissemination algorithms for large scale WSNs simple Flooding [63], dissemination using *random walks* [160], and dissemination using *gossiping* [168]. We compare the performance of these algorithms in Table 8.2 for $N = 10^4$, $M = 2 \times 10^3$, $r_t = 0.025$, and $e_t = 0.09$ with Flooding and defer the comparison with Gossiping and dissemination with random walks to future.

The simplest dissemination algorithm is the simple Flooding [63], which results in $N_{tot} = N_s N = 2.1 \times 10^7$ transmissions when used along with CS. Clearly, if CS is not employed all N readings must be stored in all N nodes resulting in $N_{tot} = N^2 = 10^8$ transmissions. Therefore, employing CS reduces the number of transmissions from 10^8 to 2.1×10^7, and CStorage-P and CStorage-B further reduces N_{tot} to 3.27×10^6 and 2.83×10^6, respectively.

Table 8.2 Comparison of N_{tot} in CStorage with existing algorithms for $e_t = 0.09$.

Protocol	N_{tot}	Notes
CStorage-P	3.27×10^6	
CStorage-B	2.81×10^6	
Flooding w. CS	2.1×10^7	$N_{tot} = N_s N$
Flooding w.o. CS	10^8	$N_{tot} = N^2$

8.4 Conclusion

In this chapter, we proposed two distributed data storage algorithms using *compressive sensing* (CS) referred to by CStorage-P and CStorage-B. These algorithms are distributed and are suitable for WSNs where no routing tables may be obtained. In CStorage-P, the readings of randomly selected network nodes are disseminated throughout the networks using *probabilistic broadcasting* (PB) to form CS measurements at nodes. After the dissemination phase, a data collector may query a small arbitrary set of nodes to recover all readings.

CStorage-P has a parameter that needs to be tuned based on network parameters. Hence, it may not be scalable and flexible to network changes. Therefore, we designed a novel *parameterless* data dissemination algorithms referred to by *alternating branching* (AB) that requires two-hop neighbor information at nodes. AB can automatically tune to network changes and requires less number of transmissions compared to PB. We discussed the advantages of CStorage-P and CStorage-B and showed that they can greatly decrease the total number of transmission for data storage compared to Flooding.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

In this dissertation, we investigated the theory and applications of the novel class of FEC codes called *rateless* or *fountain* codes in video transmission and *wireless sensor networks* (WSN). We designed rateless codes for distributed data collection [9, 10], rateless codes for high intermediate data delivery [5–7], and rateless codes with feedback [11]. Next, we investigate the applications of *unequal error protection* (UEP) rateless codes in video transmission systems. Further, we investigated the properties of UEP-rateless codes when conventional FEC codes are considered in physical layer (PL) in a video transmission system [12, 13]. Moreover, we reviewed the emerging *compressive sensing* (CS) techniques that have close connections to FEC coding theory, and designed an efficient *data storage* algorithm for WSNs employing CS [14]. We summarize our contributions and our suggested future research in what follows.

9.1 On The Intermediate Symbol Recovery Rate Of Rateless Codes

Although rateless codes are capacity achieving over erasure channels, in *intermediate* range where the number of received output symbols is less than the minimum required for full decoding of input symbols, few input symbols can be decoded. Previously, it has been shown that the intermediate range of rateless codes is comprised of three regions and for each region a rateless coding distribution that achieves optimal *intermediate symbol recovery rate* (ISRR) has been designed. However, the previously

designed codes are optimal only in one region only.

Therefore, to design rateless codes with high ISRR in all three regions, we selected one overhead from each region and designed rateless codes degree distributions that have optimal ISRR at these three overheads employing *multi-objective genetic algorithms* assuming channel erasure rate ε is not known to the encoder [5, 6]. We designed numerous codes with (almost) optimal ISRR in all regions, by covering a three dimensional pareto front (see Chapter 2.11).

Next, we assumed that an estimate of ε is available at the source and proposed *rateless coded symbol sorting* (RCSS), which further improves the ISRR of the codes we designed in the first step [6, 7]. RCSS employs the history of the previously transmitted output symbols and their dependencies for decoding to *reorder* their transmission such that each transmitted symbol has the highest probability of decoding an input symbol at decoder (if correctly delivered) among the remaining ones. Further, we modified RCSS to support varying ε and found the lower and upper bound on the ISRR. Finally, we employed one of the designed codes for data delivery in DTNs and showed that the ISRR can be greatly improved [8].

9.2 Distributed Unequal-Error-Protection Rateless Codes Over Erasure Channels

We considered a *distributed* data collection using rateless codes for two sources where disjoint sources need to deliver their rateless coded output symbols to a *common* destination through a single *relay*. Data from sources may have different data block lengths and different data *importance levels*. Consequently, we designed *distributed UEP-rateless* (DU-rateless) codes that can provide UEP with *unequal* data lengths [9, 10].

In DU-rateless codes, we optimized the coding parameters at each source and

proposed to *smartly* combine the encoded symbols at the relay. The problem in DU-rateless codes is to tune a degree distribution for each source and to design relaying parameters to achieve (almost) minimal decoding error rates with a certain ratio referred to by *UEP gain*. Similar to LT codes, DU-rateless codes are also *universal* meaning that they are simultaneously near optimal for every erasure channel.

We employed And-Or tree analysis technique to study decoding of DU-rateless codes. Next, we utilized our analytical results to design *jointly* optimize DU-rateless codes parameters and obtained several *close to* optimal DU-rateless codes for various setups. Performance comparison of the designed DU-rateless codes showed that they fulfilled the expected UEP property with almost minimal error rates. We also showed that although DU-rateless codes are designed for large message lengths, they can be employed for finite message lengths as well. Finally, we showed that DU-rateless codes surpass the performance of exiting LT codes in distributed rateless coding.

9.3 LT-SF Codes: LT Codes With Smart Feedback

We proposed *LT-SF* codes that are LT codes with *smart* feedback, which alleviate the low performance of LT codes for short data-block lengths [11]. LT codes require only a single feedback from decoder to inform the encoder (transmitter) of a successful LT decoding. Although requiring a single feedback is an outstanding advantage of LT codes, the available feedback channel remains *unused* during the transmission. Therefore, we proposed to *smartly* employ the resource constrained and weak (low data rate) feedback channel to inform the encoder from the status of decoder to considerably increases the performance of LT codes for short data-block lengths.

In LT-S codes, the decoder may alternatively issue *two* types of feedback to inform the encoder with the number of successfully decoded input symbols or request a specific input symbol that makes the largest progress in the decoding of the data-block. To generate the latter type of feedback, we proposed three novel algorithms

(with a trade-off in their algorithm complexity and performance) that describe how to analyze the decoder's status and select suitable input symbols to request. We showed that LT-SF codes considerably surpass existing algorithms in the number of required *output symbols* (LT coded packets) for full decoding and the total number of required feedbacks. Further, in contrast to previous work we considered a realistic feedback channel with unknown or varying erasure rate and designed LT-SF codes with *high resiliency* against feedback channel loss.

9.4 Unequal Error Protection Rateless Coding In Video Coding

We proposed to employ UEP-rateless codes to provide a higher perceived video quality for MEPG video transmission by providing more protection for video frames with higher influence on the quality of the displayed video [12]. Namely, we provided the highest and the lowest protection for I-frames and B-frames respectively. P-frames receive decreasingly protection (lower than I-frames and higher than B-frames) according to the frames dependencies in MPEG video stream structure. We derived the analytical expression based on the frames dependencies and found the optimum values of UEP-rateless codes parameters that results in an efficient video transmission. Initially, we evaluated the performance of our algorithm for asymptotic cases (large number of frames), and next we showed that similar gains can be achieved when the number of frames is limited.

Next, we proposed a novel periodic *video-on-demand* (VOD) broadcasting protocol with unique features of error resiliency and low-startup delay employing UEP-rateless codes [13]. These features were acquired by dividing the video segments into two partitions, and encoding each segment with a separate UEP-rateless code. We also showed that our proposed VOD scheme can easily be modified for the case that clients have a lower bandwidth than the server. Simulation results showed that our VOD broadcasting protocols with UEP-rateless coding can decrease the startup delay

considerably compared to the case where EEP rateless coding is employed.

9.5 Optimized Cross-Layer Forward Error Correction Coding For H.264 Avc Video Transmission

In video transmission systems, UEP FEC codes may be employed both at the AL and PL, while the cross-layer design of UEP FEC codes at AL and PL has not been investigated. The two FEC codes (rateless codes at AL and RCPC codes at PL) share a common channel bandwidth to add their redundancy and the optimal ratio of overhead added by each needs to be determined for a given channel SNR and bandwidth. Further, since UEP can be provided at both layers, we need to find the optimal UEP/EEP FEC setup to maximize the video PSNR.

Therefore, investigate the cross-layer design of two codes and *concurrently* tuned their parameters. We showed that our optimized schemes provide adapting the FEC code rates to the slice priority reduces the overall expected video distortion at the receiver. In our scheme, we provided higher transmission reliability to the high priority slices at the expense of the higher loss rates for low priority slices. We observed that our cross-layer FEC scheme outperformed other FEC schemes that either use the UEP coding at PL alone or EEP FEC schemes at AL as well as PL. Further, we showed that our optimization works well for different H.264 encoded video sequences, which have widely different characteristics.

9.6 Decentralized Compressive Data Storage In Wireless Sensor Networks

We investigated the data persistence problem in WSNs and designed a new *distributed data storage* algorithm referred to by *CStorage*, where nodes reading are disseminated in the network nodes such that data collector can query an *arbitrary small subset* of

nodes to obtain all readings [14]. We showed that *compressive sensing* (CS) can be employed in CStorage since natural signals are *compressible*, and proposed to form a CS *measurements* at each node by disseminating *enough* number of readings throughout the network.

To disseminate the network readings, we employed the well-known *probabilistic broadcasting* (PB) for data dissemination and proposed *CStorage-P*. In PB, no neighbor information or routing table is required for data dissemination; nevertheless, PB has a parameter called *forwarding probability* that needs to be optimized as the network changes and nodes need to be informed. Therefore, we assume nodes can obtain two-hop neighbor information and design a *parameterless* and efficient data dissemination algorithm referred to by *alternating branches* (AB), and design *CStorage-B*. Since AB has no parameter to tune, CStorage-B is *fully scalable* and can automatically adapt to drastic network topology changes. We showed both CStorage-P and CStorage-B reduce the total number of transmissions compared to case Flooding is employed for data storage in WSNs.

9.7 Suggestion for Future Research

In this dissertation, we investigated several new research areas in rateless coding and its connections with compressive sensing, wireless sensor networks, and multimedia content transmission. In the following, we provide potential future research directions.

- To design rateless codes with high ISRR, we chose three overheads, one from each intermediate region. Our selection is heuristic and the designed codes are only guaranteed to optimally perform in the selected overheads. First, it should be investigates if such a selection can also guarantee optimality throughout the intermediate range. Further, as a next step the number and location of optimization overheads should be studied.

- The DU-rateless codes have been designed for two distributed sources for asymptotic setup. Extension of these codes for multiple distributed sources and their design for finite length scenario are two interesting future research suggestions.

- Although LT-SF codes considerably improve the performance of LT codes in finite length, they are not capacity achieving and are not necessarily optimal. Therefore, optimal LT codes with feedback can be potentially studied and investigate.

- UEP-rateless codes have interesting application in audio and data transmission schemes especially over the Internet, which needs to be further investigated.

- CStorage was designed for static WSNs. However, it may be simply extended to mobile ad-hoc networks and DTNs as a next step.

BIBLIOGRAPHY

[1] M. Luby, "LT codes," *The 43rd Annual IEEE Symposium on Foundations of Computer Science,*, pp. 271–280, 2002.

[2] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551–2567, June 2006.

[3] ""3GPP TS 26.346 V7.1.0" Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs," June.

[4] S. Sanghavi, "Intermediate performance of rateless codes," *Information Theory Workshop, 2007. ITW '07. IEEE*, pp. 478–482, Sept. 2007.

[5] A. Talari and N. Rahnavard, "Rateless codes with optimum intermediate performance," *IEEE GLOBECOM, Honolulu, Hawaii*, Dec. 2009.

[6] A. Talari and N. Rahnavard, "On the intermediate symbol recovery rate of rateless codes," *IEEE Transactions on Communications*, vol. 60, pp. 1237 – 1242, may 2012.

[7] A. Talari, B. Shahrasbi, and N. Rahnavard, "Efficient symbol sorting for high intermediate recovery rate of LT codes," *IEEE International Symposium on Information Theory Proceedings (ISIT), 2010*, pp. 2443 –2447, june 2010.

[8] A. Talari and N. Rahnavard, "Enhanced intermediate packet delivery in delay tolerant networks," *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, pp. 575 –580, Nov 2010.

[9] A. Talari and N. Rahnavard, "Distributed rateless codes with uep property," *IEEE International Symposium on Information Theory Proceedings (ISIT), 2010*, pp. 2453 –2457, june 2010.

[10] A. Talari and N. Rahnavard, "Distributed unequal error protection rateless codes over erasure channels: A two-source scenario," *IEEE Transactions on Communications*, vol. 60, pp. 2084 –2090, August 2012.

[11] A. Talari and N. Rahnavard, "LT-SF Codes: LT Codes with Smart Feedback," *Submitted to Elsevier Computer Communications*.

[12] A. Talari and N. Rahnavard, "Unequal error protection rateless coding for efficient MPEG video transmission," *Military Communications Conference (MILCOM), Boston*, Oct. 2009.

[13] A. Talari and N. Rahnavard, "A low-latency and error-resilient video-on-demand broadcasting protocol using UEP-rateless codes," *46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 991–995, Sept. 2008.

[14] A. Talari and N. Rahnavard, "Cstorage: Distributed data storage in wireless sensor networks employing compressive sensing," *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1–5, 2011.

[15] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," *Proceedings of the 29th annual ACM symposium on Theory of computing*, pp. 150–159, 1997.

[16] N. Rahnavard and F. Fekri, "Finite-length unequal error protection rateless codes: design and analysis," *IEEE Global Telecommunications Conference, GLOBECOM*, vol. 3, p. 5 pp., November-December 2005.

[17] E. Bodine and M. Cheng, "Characterization of Luby transform codes with small message size for low-latency decoding," *IEEE International Conference on Communications, ICC*, pp. 1195 –1199, may 2008.

[18] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," *26th IEEE International Conference on Computer Communications, INFOCOM*, pp. 2576 –2580, may 2007.

[19] N. Rahnavard, B. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Transactions on Information Theory*, vol. 53, pp. 1521–1532, April 2007.

[20] N. Rahnavard and F. Fekri, "Generalization of rateless codes for unequal error protection and recovery time: Asymptotic analysis," *IEEE International Symposium on Information Theory*, pp. 523–527, July 2006.

[21] M. Luby, M. Mitzenmacher, and M. Shokrollahi, "Analysis of random processes via And-Or tree evaluation," *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pp. 364–373, 1998.

[22] P. Maymounkov, "Online codes," *NYU Tech. Rep. TR2003-883*, 2002.

[23] P. Acelas, P. Arce, and J. Guerri, "Effect of the Multiple Description Coding over a Hybrid Fixed-AdHoc Video Distribution Network," p. 176, Springer, 2009.

[24] A. Ziviani, B. E. Wolfinger, J. F. Rezende, O. C. Duarte, and S. Fdida, "Joint adoption of QoS schemes for MPEG streams," *Multimedia Tools Appl.*, vol. 26, no. 1, pp. 59–80, 2005.

[25] C.-Y. Yu, C.-H. Ke, C.-K. Shieh, and N. Chilamkurti, "MyEvalvid-NT - a

simulation tool-set for video transmission and quality evaluation," *IEEE Region 10 Conference TENCON 2006.*, pp. 1–4, Nov. 2006.

[26] C.-H. Lin, C.-H. Ke, C.-K. Shieh, and N. Chilamkurti, "The packet loss effect on MPEG video transmission in wireless networks," *20th International Conference on Advanced Information Networking and Applications, 2006. AINA 2006.*, vol. 1, pp. 565–572, April 2006.

[27] A. Khan, L. Sun, and E. Ifeachor, "Impact of video content on video quality for video over wireless networks," *Fifth International Conference on Autonomic and Autonomous Systems, 2009. ICAS '09.*, pp. 277–282, April 2009.

[28] A. Khan, L. Sun, and E. Ifeachor, "An ANFIS-based hybrid video quality prediction model for video streaming over wireless networks," *The Second International Conference on Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08.*, pp. 357–362, Sept. 2008.

[29] C. L. Ke and C. Shieh., "Evaluation of streaming MPEG video over wireless channels," *Journal of mobile multimedia*, vol. 3, no. 1, pp. 047–064, 2007.

[30] N. Thomos, S. Argyropoulos, N. Boulgouris, and M. Strintzis, "Robust transmission of H. 264/AVC streams using adaptive group slicing and unequal error protection," *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 120–120, 2006.

[31] K. Kambhatla, S. Kumar, and P. Cosman, "Wireless H.264 Video Quality Enhancement through Optimal Prioritized Packet Fragmentation," *IEEE Trans. Multimedia (to appear)*, Oct 2012.

[32] S. Kumar, A. Janarthanan, M. M. Shakeel, S. Maroo, J. D. Matyjas, and M. Medley, "Robust H.264/AVC Video Coding with Priority Classification,

Adaptive NALU Size and Fragmentation," *IEEE MILCOM, 2009, Boston, USA.*, Oct 2009.

[33] S. Paluri, K. Kambhatla, S. Kumar, B. Bailey, P. Cosman, and J. D. Matyjas, "Predicting Slice Loss Distortion in H.264/AVC Video for Low Complexity Data Prioritization," *IEEE Int. Conf. Image Processing (ICIP 2012), Orlando, FL*, Sep-Oct 2012.

[34] E. Baccaglini, T. Tillo, and G. Olmo, "Slice Sorting for Unequal Loss Protection of Video Streams," *IEEE Signal Processing Letters*, vol. 15, pp. 581–584, 2008.

[35] S. Argyropoulos, A. Tan, N. Thomos, E. Arikan, and M. Strintzis, "Robust Transmission of Multi-View Video Streams using Flexible Macroblock Ordering and Systematic LT Codes," *3DTV Conference, 2007*, pp. 1 –4, may 2007.

[36] A. Hu, "Video-on-demand broadcasting protocols: a comprehensive study," *in Proc. of IEEE INFOCOM*, vol. 1, pp. 508–517, Apr. 2001.

[37] S. Viswanathan and T. Imielinski, "Pyramid broadcasting for video on demand service," in *In Proceedings of ST/SPIE Conference on Multimedia Computing and Networking (MMCN)*, pp. 66–77, IEEE Press, 1995.

[38] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Syst.*, vol. 4, no. 4, pp. 197–208, 1996.

[39] C. Aggarwal, J. Wolf, and P. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems, 1996.*, pp. 118–126, Jun 1996.

[40] K. A. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme

for metropolitan video-on-demand systems," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 89–100, 1997.

[41] A. Hu, I. Nikolaidis, and P. V. Beek, "On the design of efficient video-on-demand broadcast schedules," in *MASCOTS '99: Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, (Washington, DC, USA), p. 262, IEEE Computer Society, 1999.

[42] L.-S. Juhn and L.-M. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Transactions on Broadcasting*, vol. 43, pp. 268–271, Sep 1997.

[43] J.-F. Paris, S. Carter, and D. Long, "Efficient broadcasting protocols for video on demand," in *Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1998. Proceedings.*, pp. 127–132, Jul 1998.

[44] J. Paris, S. Carter, and D. Long, "A low bandwidth broadcasting protocol for video on demand," *in Proceedings of the 7 th International Conference on Computer Communications and Networks (IC3N'98)*, pp. 690–697, Oct. 1998.

[45] J. cois, P. aris, S. Carter, and D. Long, "A hybrid broadcasting protocol for video on demand," *In Proceedings of the 1999 Multimedia Computing and Networking Conference*, pp. 317–326, Jan. 1998.

[46] J.-F. Paris, "A simple low-bandwidth broadcasting protocol for video-on-demand," pp. 118–123, 1999.

[47] J.-F. Paris, "A fixed-delay broadcasting protocol for video-on-demand," pp. 418–423, 2001.

[48] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery," *in Proc. of SIGCOMM' 2001*, p. 12, 2001.

[49] L. Xu, "Efficient and scalable on-demand data streaming using uep codes," in *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, (New York, NY, USA), pp. 70–78, ACM, 2001.

[50] T. Stockhammer, T. Gasiba, W. A. Samad, T. Schierl, H. Jenkac, T. Wiegand, and W. Xu, "Nested harmonic broadcasting for scalable video over mobile datacast channels: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 7, no. 2, pp. 235–256, 2007.

[51] P. Elias, "Coding for noisy channels," *IRE Conv. Rec*, vol. 3, no. pt 4, pp. 37–46, 1955.

[52] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, pp. 389–400, April 1988.

[53] D. Donoho, "Compressed sensing," *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[54] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[55] E. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. on information theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[56] C. Chou, R. Rana, and W. Hu, "Energy efficient information collection in wireless sensor networks using adaptive compressive sensing," *IEEE 34th Conference on Local Computer Networks (LCN 2009)*, 2009.

[57] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Information Theory*, vol. 52, pp. 489–509, Feb. 2006.

[58] W. Wang, M. Garofalakis, and K. Ramchandran, "Distributed sparse random projections for refinable approximation," *International conference on Information processing in sensor networks*, pp. 331–339, 2007.

[59] S. S. Chen, D. L. Donoho, Michael, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[60] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, p. 4655, 2007.

[61] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[62] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pp. 1658–1666, 2007.

[63] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks," *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 64–71, 1999.

[64] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 151–162, 1999.

[65] Z. Haas, J. Halpern, and L. Li, "Gossip-based ad hoc routing," *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, INFOCOM 2002.*, vol. 3, pp. 1707–1716, 2002.

[66] N. Rahnavard, B. Vellambi, and F. Fekri, "CRBcast: a reliable and energy-efficient broadcast scheme for wireless sensor networks using rateless codes," *IEEE Trans. on Wireless Communications*, vol. 7, no. 12 Part 2, pp. 5390–5400, 2008.

[67] N. Rahnavard and F. Fekri, "CRBcast: a collaborative rateless scheme for reliable and energy-efficient broadcasting in wireless sensor networks," *International conference on Information processing in sensor networks*, p. 283, 2006.

[68] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 229–236, 2005.

[69] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 109–120, 2005.

[70] B. N. Vellambi, R. Subramanian, F. Fekri, and M. Ammar, "Reliable and efficient message delivery in delay tolerant networks using rateless codes," *MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pp. 91–98, 2007.

[71] E. Altman and F. De Pellegrini, "Forward correction and fountain codes in delay tolerant networks," *IEEE INFOCOM 2009*, pp. 1899 –1907, 19-25 2009.

[72] D. Kotz, T. Henderson, and I. Abyzov, "CRAWDAD data set dartmouth/campus (v. 2004-12-18)." Downloaded from http://www.crawdad.org/dartmouth/campus, dec 2004.

[73] D. Mohney, "Soaring condor relieves headaches," *Mobile Radio Technology*.

[74] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: rethinking connectivity in developing nations," *Computer*, vol. 37, pp. 78–83, Jan. 2004.

[75] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, 2002.

[76] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–11, April 2006.

[77] J. Kopena, E. Sultanik, G. Naik, I. Howley, M. Peysakhov, V. A. Cicirello, M. Kam, and W. Regli, "Service-based computing on manets: Enabling dynamic interoperability of first responders," *IEEE Intelligent Systems*, vol. 20, no. 5, pp. 17–25, 2005.

[78] N. Eagle and A. S. Pentland, "Reality mining: sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, no. 4, pp. 255–268, 2006.

[79] "Cabspotting project," *http://cabspotting.org/*.

[80] P. Luo, H. Huang, W. Shu, M. Li, and M.-Y. Wu, "Performance evaluation of vehicular DTN routing under realistic mobility models," *IEEE Wireless Com-*

munications and Networking Conference, 2008. WCNC 2008., pp. 2206–2211, 31 2008-April 3 2008.

[81] B. Walker, T. Clancy, and J. Glenn, "Using localized random walks to model delay-tolerant networks," *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1 –7, 16-19 2008.

[82] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of random direction models," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, pp. 1897–1907 vol. 3, March 2005.

[83] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model," *Wirel. Netw.*, vol. 10, no. 5, pp. 555–567, 2004.

[84] K.-H. Chiang and N. Shenoy, "A 2-D random-walk mobility model for location-management studies in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 53, pp. 413–424, March 2004.

[85] C. Liu and J. Wu, "Adaptive routing in dynamic Ad Hoc networks," *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pp. 2603–2608, 31 2008-April 3 2008.

[86] Y. Liao, K. Tan, Z. Zhang, and L. Gao, "Estimation based erasure-coding routing in delay tolerant networks," *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pp. 557–562, 2006.

[87] M. McNett and G. M. Voelker, "Access and mobility of wireless PDA users," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 2, pp. 40–55, 2005.

[88] M. Abdulla and R. Simon, "Characteristics of common mobility models for opportunistic networks," *PM2HW2N '07: Proceedings of the 2nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pp. 105–109, 2007.

[89] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for Ad Hoc network research," *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483–502, 2002.

[90] M. Abdulla and R. Simon, "A simulation study of common mobility models for opportunistic networks," *41st Annual Simulation Symposium, 2008. ANSS 2008.*, pp. 43–50, April 2008.

[91] A. Vahdat and D. Becker, "Epidemic routing for partially-connected Ad Hoc networks," *Technical Report CS-200006, Duke University*, April 2000.

[92] M. Grossglauser and D. Tse, "Mobility increases the capacity of Ad-Hoc wireless networks," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 477–486, Aug 2002.

[93] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 145–158, 2004.

[94] A. Lindgren, A. Doria, and O. Schel, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[95] C.-S. Lin, W.-S. Chang, L.-J. Chen, and C.-F. Chou, "Performance study of routing schemes in delay tolerant networks," *AINAW '08: Proceedings of the*

22nd International Conference on Advanced Information Networking and Applications - Workshops, pp. 1702–1707, 2008.

[96] L. Yin, H. mei Lu, and Y. da Cao, "Probabilistic delay routing for delay tolerant networks," *10th International Conference on Advanced Communication Technology, ICACT*, vol. 1, pp. 191–195, Feb. 2008.

[97] Z. Li and H. Shen, "Probabilistic routing with multi-copies in delay tolerant networks," *ICDCSW '08: Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems Workshops*, pp. 471–476, 2008.

[98] A. Islam and M. Waldvogel, "Reality-check for DTN routing algorithms," *28th International Conference on Distributed Computing Systems Workshops, 2008. ICDCS '08.*, pp. 204–209, June 2008.

[99] J. Holland, *Adaptation in natural and artificial systems*. MIT press Cambridge, MA, 1992.

[100] J. R. Koza, "Survey of genetic algorithms and genetic programming," *WESCON/'95. Conference record*, p. 589, November 1995.

[101] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, Apr 2002.

[102] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: Maximizing sensor network data persistence," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 255–266, 2006.

[103] S. Kim and S. Lee, "Improved intermediate performance of rateless codes," *11th International Conference on Advanced Communication Technology, ICACT*, vol. 03, pp. 1682–1686, Feb. 2009.

[104] A. Beimel, S. Dolev, and N. Singer, "RT oblivious erasure correcting," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1321–1332, 2007.

[105] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," *International Symposium on Information Theory Proceedings, ISIT*, p. 39, June-July 2004.

[106] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," *International Symposium on Information Theory Proceedings, ISIT*, pp. 2677–2679, 2006.

[107] *http://cwnlab.ece.okstate.edu/research.htm*.

[108] R. Gummadi and R. Sreenivas, "Relaying a fountain code across multiple nodes," *IEEE Information Theory Workshop, ITW*, pp. 149–153, May 2008.

[109] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad-hoc wireless networks," *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1360 –1369 vol.3, 2001.

[110] S. Puducheri, J. Kliewer, and T. Fuja, "The design and performance of distributed LT codes," *IEEE Transactions on Information Theory*, vol. 53, pp. 3740–3754, October 2007.

[111] D. Sejdinovic, R. Piechocki, and A. Doufexi, "AND-OR tree analysis of distributed LT codes," *IEEE Information Theory Workshop on Networking and Information Theory, ITW*, pp. 261–265, June 2009.

[112] A. Liau, S. Yousefi, and I. Kim, "Binary soliton-like rateless coding for the y-network," *IEEE Transactions on Communications*, vol. PP, no. 99, pp. 1 –6, 2011.

[113] A. Hagedorn, S. Agarwal, D. Starobinski, and A. Trachtenberg, "Rateless coding with feedback," *INFOCOM 2009*, pp. 1791 –1799, 2009.

[114] A. Beimel, S. Dolev, and N. Singer, "RT oblivious erasure correcting," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 1321 –1332, dec. 2007.

[115] S. Kokalj-Filipovic, P. Spasojevic, E. Soljanin, and R. Yates, "ARQ with doped fountain decoding," *IEEE 10th International Symposium on Spread Spectrum Techniques and Applications, ISSSTA*, pp. 780 –784, aug. 2008.

[116] J. Sørensen, P. Popovski, and J. Østergaard, "On the Role of Feedback in LT Codes," *Arxiv preprint arXiv:1012.2673*, 2010.

[117] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the lambert W function," *Advances in Computational mathematics*, vol. 5, no. 1, pp. 329–359, 1996.

[118] T. Anderson and S. Ghurye, "Identification of parameters by the distribution of a maximum random variable," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 337–342, 1977.

[119] K. Briggs, L. Song, and T. Prellberg, "A note on the distribution of the maximum of a set of Poisson random variables," *Arxiv preprint arXiv:0903.4373*, 2009.

[120] A. Shokrollahi, "LDPC codes: An introduction," *Digital Fountain, Inc., Fremont, CA, Tech. Rep*, 2003.

[121] T. Fang and L.-P. Chau, "GOP-based channel rate allocation using genetic algorithm for scalable video streaming over error-prone networks," *IEEE Transactions on Image Processing*, vol. 15, pp. 1323–1330, June 2006.

[122] E. Maani and A. Katsaggelos, "Unequal error protection for robust streaming of scalable video over packet lossy networks," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 20, pp. 407 –416, march 2010.

[123] W. Xiang, C. Zhu, C. K. Siew, Y. Xu, and M. Liu, "Forward error correction-based 2-d layered multiple description coding for error-resilient h.264 svc video transmission," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 19, pp. 1730 –1738, dec. 2009.

[124] A. Bouabdallah and J. Laca, "Dependency-aware unequal erasure protection codes," *Journal of Zhejiang University - Science A*, vol. 7, pp. 27–33, Jan 2006.

[125] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[126] R. Congchong, Z. Haifeng, Y. Liuguo, L. Jianhua, and C. Changwene, "A new uep scheme for robust video transmission in ldpc coded mimo-ofdm system," *Journal of Electronics (China)*, vol. 24, May 2007.

[127] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory,*, vol. 8, no. 1, pp. 21–28, 1962.

[128] A. Abd El Al, T. Saadawi, and M. Lee, "Unequal error protection for real-time video in mobile ad hoc networks via multi-path transport," *Comput. Commun.*, vol. 30, no. 17, pp. 3293–3306, 2007.

[129] D. Wijesekera, J. Srivastava, A. Nerode, and M. Foresti, "Experimental evaluation of loss perception in continuous media," *Multimedia Systems*, vol. 7, no. 6, pp. 486–499, 1999.

[130] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, and T. Gasiba, "Application layer forward error correction for mobile multimedia broadcasting," *Handbook of Mobile Broadcasting: DVB-H, DMB, ISDB-T and Media FLO*, pp. 239–280, 2008.

[131] Gomez-Barquero, D., and A. Bria, "Application layer FEC for improved mobile reception of DVB-H streaming services," *IEEE 64th Vehicular Technology Conference, VTC-2006 Fall*, pp. 1–5, 2006.

[132] T. Courtade and R. Wesel, "A cross-layer perspective on rateless coding for wireless channels," *IEEE International Conference on Communications, ICC*, pp. 1–6, 2009.

[133] M. VanDer-Schaar and S. Shankar, "Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms," *IEEE Wireless Communications*, vol. 12, pp. 50–58, August 2005.

[134] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," *IEEE Wireless Communications*, vol. 12, pp. 59–65, August 2005.

[135] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Adaptive Unicast Video Streaming With Rateless Codes and Feedback," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, pp. 275 –285, Febuary 2010.

[136] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-Window Raptor Codes for Efficient Scalable Wireless Video Broadcasting With Unequal Loss Protection," *IEEE Transactions on Image Processing*, vol. 19, pp. 1491–1503, June 2010.

[137] H. Kushwaha, Y. Xing, R. Chandramouli, and H. Heffes, "Reliable Multimedia

Transmission Over Cognitive Radio Networks Using Fountain Codes," *Proceedings of the IEEE*, vol. 96, pp. 155 –165, January 2008.

[138] C. Hellge, T. Schierl, and T. Wiegand, "Receiver driven layered multicast with layer-aware forward error correction," *15th IEEE International Conference on Image Processing, ICIP*, pp. 2304–2307, October 2008.

[139] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Expanding Window Fountain codes for scalable video multicast," *IEEE International Conference on Multimedia and Expo*, pp. 77–80, April 2008.

[140] A. S. Tan, A. Aksay, C. Bilen, G. B. Akar, and E. Arikan, "Rate-distortion optimized layered stereoscopic video streaming with raptor codes," *Packet Video*, pp. 98–104, November 2007.

[141] H. Jenkac and T. Stockhammer, "Asynchronous Media Streaming Over Wireless Broadcast Channels," *IEEE International Conference on Multimedia and Expo*, pp. 1318–1321, July 2005.

[142] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Robust live unicast video streaming with rateless codes," *Packet Video*, pp. 78–84, November 2007.

[143] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable Video Multicast Using Expanding Window Fountain Codes," *IEEE Transactions on Multimedia*, vol. 11, pp. 1094–1104, October 2009.

[144] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," *International Conference on Dependable Systems and Networks*, pp. 145–154, July 2004.

[145] A. Bouabdallah and J. Lacan, "Dependency-aware unequal erasure protection

codes," *Journal of Zhejiang University - Science A*, vol. 7, pp. 27–33, January 2006.

[146] H. Ha and C. Yim, "Layer-weighted unequal error protection for scalable video coding extension of H.264/AVC," *IEEE Transactions on Consumer Electronics*, vol. 54, pp. 736–744, May 2008.

[147] Y. Liu and S. Yu, "Adaptive unequal loss protection for scalable video streaming over IP networks," *IEEE Transactions on Consumer Electronics*, vol. 51, pp. 1277–1282, November 2005.

[148] S. Yingbo Shi, W. Chengke Wu, and D. Jianchao Du, "A Novel Unequal Loss Protection Approach for Scalable Video Streaming over Wireless Networks," *IEEE Transactions on Consumer Electronics*, vol. 53, pp. 363–368, May 2007.

[149] X.-J. Zhang, X.-H. Peng, R. Haywood, and T. Porter, "Robust video transmission over lossy network by exploiting H.264/AVC data partitioning," *5th International Conference on Broadband Communications, Networks and Systems, BROADNETS*, pp. 307–314, September 2008.

[150] M. Luby, M. Watson, T. Gasiba, and T. Stockhammer, "Mobile data broadcasting over MBMS tradeoffs in forward error correction," in *Proceedings of the 5th international conference on Mobile and ubiquitous multimedia*, p. 10, ACM, 2006.

[151] T. Gasiba, W. Xu, and T. Stockhammer, "Enhanced system design for download and streaming services using Raptor codes," *European Transactions on Telecommunications*, vol. 20, no. 2, pp. 159–173, 2009.

[152] J. Afzal, T. Stockhammer, T. Gasiba, and W. Xu, "Video streaming over MBMS: A system design approach," *Journal of Multimedia*, vol. 1, no. 5, pp. 25–35, 2006.

[153] A. Alexiou, C. Bouras, and A. Papazois, "A study of forward error correction for mobile multicast," *International Journal of Communication Systems*, vol. 24, no. 5, pp. 607–627, 2011.

[154] D. Munaretto, D. Jurca, and J. Widmer, "Broadcast video streaming in cellular networks: an adaptation framework for channel, video and AL-FEC rates allocation," *Wireless Internet Conference (WICON), 2010 The 5th Annual ICST*, pp. 1–9, 2010.

[155] C. Bouras, N. Kanakis, V. Kokkinos, and A. Papazois, "Application layer forward error correction for multicast streaming over LTE networks," *International Journal of Communication Systems*, 2012.

[156] D. Coley, *An introduction to genetic algorithms for scientists and engineers*. World Scientific Pub Co Inc, 1999.

[157] *Global Optimization Toolbox: User's Guide (r2011b)*. Mathworks, Nov 2011.

[158] JVT, "H.264/AVC reference software JM14.2," *ISO/IEC Std. [Online]. Available: http://iphome.hhi.de/suehring/tml/download/.*

[159] A. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, pp. 476 –489, march 2011.

[160] Z. Kong, S. Aly, and E. Soljanin, "Decentralized coding algorithms for distributed storage in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 261 –267, february 2010.

[161] R. Masiero, G. Quer, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "Data acquisition through joint compressive sensing and principal component analy-

sis," *IEEE Global Telecommunications Conference, GLOBECOM 2009.*, pp. 1 –6, Dec 2009.

[162] S. Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega, "Compressed Sensing and Routing in Multi-Hop Networks," *Technical Report, University of Southern California,*, 2009.

[163] *Available online: http://sensorscope.epfl.ch/index.php.*

[164] L. Xiang, J. Luo, and A. Vasilakos, "Compressed data aggregation for energy efficient wireless sensor networks," *8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011*, pp. 46–54, 2011.

[165] C. Luo, F. Wu, J. Sun, and C. Chen, "Efficient measurement generation and pervasive sparsity for compressive data gathering," *IEEE Transactions on Wireless Communications*, vol. 9, no. 12, pp. 3728–3738, 2010.

[166] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," *MobiCom '09: Proceedings of the 15th annual international conference on Mobile computing and networking*, pp. 145–156, 2009.

[167] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "On the interplay between routing and signal representation for compressive sensing in wireless sensor networks," *Information Theory and Applications Workshop, 2009*, pp. 206 –215, feb. 2009.

[168] M. Rabbat, J. Haupt, A. Singh, and R. Nowak, "Decentralized compression and predistribution via randomized gossiping," *International Symposium on Information Processing in Sensor Networks*, pp. 51–59, 2006.

[169] A. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2809 – 2816, 2006.

[170] M. Heissenbuttel, T. Braun, M. Walchli, and T. Bernoulli, "Optimized stateless broadcasting in wireless multi-hop networks," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1 –12, april 2006.

[171] A. Kini, V. Veeraraghavan, N. Singhal, and S. Weber, "Smartgossip: an improved randomized broadcast protocol for sensor networks," *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 210–217, 2006.

[172] P. Kyasanur, R. Choudhury, and I. Gupta, "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks," *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pp. 91–100, 2006.

[173] B. Bako, F. Kargl, E. Schoch, and M. Weber, "Advanced adaptive gossiping using 2-hop neighborhood information," *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1–6, 2008.

[174] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti, "Localized techniques for broadcasting in wireless sensor networks," *Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pp. 41–51, 2004.

[175] X. Li, K. Moaveninejad, and O. Frieder, "Regional gossip routing for wireless ad hoc networks," *Mobile Networks and Applications*, vol. 10, no. 1, pp. 61–77, 2005.

[176] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware

wireless microsensor networks," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 40–50, 2002.

[177] A. Durresi, V. Paruchuri, S. Iyengar, and R. Kannan, "Optimized broadcast protocol for sensor networks," *Computers, IEEE Transactions on*, vol. 54, no. 8, pp. 1013–1024, 2005.

[178] W. Chen, M. Rodrigues, and I. Wassell, "Distributed compressive sensing reconstruction via common support discovery," *IEEE International Conference on Communications (ICC), 2011*, pp. 1 –5, june 2011.

[179] M. Mahmudimanesh, A. Khelil, and N. Suri, "Reordering for better compressibility: Efficient spatial sampling in wireless sensor networks," *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010*, pp. 50 –57, june 2010.

[180] R. Karp, "Reducibility among combinatorial problems," *50 Years of Integer Programming 1958-2008*, pp. 219–241, 2010.

[181] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[182] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

VITA

Ali Talari

Candidate for the Degree of

Doctor of Philosophy

Dissertation: ENHANCED RATELESS CODING AND COMPRESSIVE SENSING FOR EFFICIENT DATA/MULTIMEDIA TRANSMISSION AND STORAGE IN AD-HOC AND SENSOR NETWORKS

Major Field: Electrical and Computer Engineering

Biographical:

Personal Data: Tehran, Iran, 6, March, 1981

Education:

Received the B.S. degree from University of Kashan, Kashan, Iran, 2003, Electrical Engineering,
Received the M.S. degree from Sharif University of Technology, Tehran, Iran, Jan 2007, Electrical Engineering,
Completed the requirements for the degree of Doctor of Philosophy with a major in Electrical and Computer Engineering at Oklahoma State University in December, 2012.

Name: Ali Talari Date of Degree: December, 2012

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: ENHANCED RATELESS CODING AND COMPRESSIVE SENSING FOR EFFICIENT DATA/MULTIMEDIA TRANSMISSION AND STORAGE IN AD-HOC AND SENSOR NETWORKS

Pages in Study: 198 Candidate for the Degree of Doctor of Philosophy

Major Field: Electrical Engineering

In this dissertation, we investigate the theory and applications of the novel class of FEC codes called *rateless* or *fountain* codes in video transmission and *wireless sensor networks* (WSN). First, we investigate the rateless codes in *intermediate* region where the number of received encoded symbols is less that minimum required for full data-block decoding. We devise techniques to improve the input symbol recovery rate when the erasure rate is unknown, and also for the case where an estimate of the channel erasure rate is available.

Further, we design *unequal error protection* (UEP) rateless codes for *distributed* data collection of data blocks of *unequal* lengths, where two encoders send their rateless coded output symbols to a destination through a *common* relay. We design such distributed rateless codes, and jointly optimize rateless coding parameters at each nodes and relaying parameters. Moreover, we investigate the performance of rateless codes with finite block length in the presence of *feedback* channel. We propose a smart feedback generation technique that greatly improves the performance of rateless codes when data block is finite. Moreover, we investigate the applications of UEP-rateless codes in video transmission systems. Next, we study the optimal cross-layer design of a video transmission system with rateless coding at application layer and fixed-rate coding (RCPC coding) at physical layer.

Finally, we review the emerging *compressive sensing* (CS) techniques that have close connections to FEC coding theory, and designed an efficient *data storage* algorithm for WSNs employing CS referred to by *CStorage*. First, we propose to employ *probabilistic broadcasting* (PB) to form one CS *measurement* at each node and design *CStorage-P*. Later, we can query any *arbitrary* small subset of nodes and recover all sensors reading. Next, we design a novel *parameterless* and more efficient data dissemination algorithm that uses two-hop neighbor information referred to *alternating branches* (AB). We replace PB with AB and design *CStorage-B*, which results in a lower number of transmissions compared to CStorage-P.

ADVISOR'S APPROVAL: _____