Error Estimating Codes for Insertion and Deletion Channels

Jiwei Huang
Department of Computer Science and Technology
Tsinghua University
Beijing, China
huangjw05@gmail.com

Sen Yang
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA
sen.yang@gatech.edu

Ashwin Lall
Department of Math and Computer Science
Denison University
Granville, OH, USA
lalla@denison.edu

Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA
jrom@ece.gatech.edu

Jun (Jim) Xu
School of Computer Science
Georgia Institute of Technology
Atlanta, GA, USA
jx@cc.gatech.edu

Chuang Lin
Department of Computer Science and Technology
Tsinghua University
Beijing, China
chlin@tsinghua.edu.cn

ABSTRACT

Error estimating codes (EEC) have recently been proposed for measuring the bit error rate (BER) in packets transmitted over wireless links. They however can provide such measurements only when there are no insertion and deletion errors, which could occur in various wireless network environments. In this work, we propose "idEEC", the first technique that can do so even in the presence of insertion and deletion errors. We show that idEEC is provable robust under most bit insertion and deletion scenarios, provided insertion/deletion errors occur with much lower probability than bit flipping errors. Our idEEC design can build upon any existing EEC scheme. The basic idea of the idEEC encoding is to divide the packet into a number of segments, each of which is encoded using the underlying EEC scheme. The basic idea of the idEEC decoding is to divide the packet into a few slices in a randomized manner – each of which may contain several segments – and then try to identify a slice that has no insertion and deletion errors in it (called a "clean slice"). Once such a clean slice is found, it is removed from the packet for later processing, and this "randomized divide and search" procedure will be iteratively performed on the rest of the packet until no more clean slices can be found. The BER will then be estimated from all the clean slices discovered through all the iterations. A careful analysis of the accuracy guarantees of the idEEC decoding is provided, and the efficacy of idEEC is further validated by simulation experiments.

SIGMETRICS'14, June 16–20, 2014, Austin, Texas, USA.

http://dx.doi.org/10.1145/2591971.2591976.

Categories and Subject Descriptors

E.4 [**Coding and Information Theory**]: Error control codes; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design - Wireless communication

General Terms

Algorithms, Theory

Keywords

Error Estimating Coding, Insertion Channel, Deletion Channel

1. INTRODUCTION

Recently, there has been a series of works on designing error estimating codes, that is, estimating the bit error rate (BER) in packets transmitted over wireless networks [2, 5, 6]. The basic idea of such codes is to have the transmitter send a small error estimating codeword of $O(\log(n))$ bits along with a packet n bits long, which allows the receiver to decode the information and estimate the number of bits in the packet flipped during the transmission (i.e., the BER). Such codes are in general stronger – and a bit more expensive – than error detection codes, which inform whether or not there are bit errors, but weaker – and much less expensive – than the error correction codes.

The aforementioned techniques, however, can provide such estimations only when there are no insertion and deletion errors. None of them will work even if a single bit is inserted or deleted for the following reason. In all existing error estimating coding (EEC) schemes, a sub-codeword is computed as a function of a subset of bits sampled from the packet as identified by their indices, i.e., their positions in the packet relative to the beginning of the packet. If there are one or more insertion and deletion errors, a very different set of bits will be sampled and hence a very different sub-codeword computed from them. In wireless network applications EEC schemes are designed for, however, insertion and deletion error may occur due to factors such as imperfect synchronization between a transmitter and a receiver, or uneven propagation delays to successive bits in a packet (e.g., due to the movement of a transmit-

ter or receiver). Therefore, for existing EEC schemes to be widely applicable in various wireless network environments, it is important that they are enhanced to work over insertion/deletion channels. However, this appears an extremely difficult task, and readers will appreciate this difficulty when we describe in Sec. 2.

In this work, we make an attempt at filling this gap and propose error estimating codes for the insertion and deletion channels, called "idEEC". More specifically, our idEEC scheme allows for the estimation of the number of bit flipping errors in a packet despite the existence of bit insertions and deletions. Our idEEC scheme can build upon any existing EEC scheme, and its encoding efficiency (i.e., code density) is proportional to that of the underlying EEC scheme. For the purpose of describing how idEEC works – which is the main thrust of this paper – however, we pick the plain vanilla tug-of-war sketch [1] as the underlying EEC, even though its encoding efficiency is worse than the two most recent EEC schemes, namely EToW and gEEC [5, 6], for two reasons. First, the encoding efficiency of Tug-of-War (ToW) is worse than that of EToW and gEEC by only a constant factor, and is still asymptotically optimal. Second, the encoding and decoding algorithms of ToW, which idEEC must "interface" with, are much simpler than that of EToW and gEEC. Building idEEC upon ToW allows us to focus on describing the encoding and decoding algorithms (for withstanding insertion/deletion errors) of idEEC and their mathematical analyses, without being entangled into additional mathematical and algorithmic details involved in "interfacing" with more efficient but mathematically sophisticated EEC schemes such as EToW and gEEC, and risking diluting the main thrust of this paper.

We describe here the basic idea behind idEEC encoding (at the transmitter) and decoding (at the receiver) algorithms, deferring readers to Sec. 3 for details. We assume that, during the transmission, no more than a total of ϕ insertion and deletion errors may occur to the packet. This ϕ is usually a small number (say no more than 10) in the aforementioned wireless applications. The encoding algorithm is very simple. A packet is first divided into a number of segments, and an EEC codeword for each segment is computed. The number of segments needed for accurate BER estimation is in general no more than an order of magnitude larger than ϕ. The idEEC codeword is simply these EEC codewords pieced together, and will be further protected by error detection or correction codes against one or two insertion/deletion errors and a small number of bit flipping errors that may occur to the EEC codeword [4]. Since adding this protection will only increase the size of an idEEC codeword by a small constant factor, it will not affect the asymptotic space complexity of the idEEC.

The decoding algorithm – for inferring the number of flipping errors from the packet and its idEEC codeword – at the receiver's end is more involved. Suppose the actual number of insertion and deletion errors are ϕ. The basic idea of the algorithm is to uniformly randomly insert or delete ϕ or (slightly) more "phantom" bits along the segment boundaries in the packet. These phantom insertions and deletions are intended to partially "cancel out" the actual insertions and deletions that occur to the packet during the transmission, and the uniform randomization is needed for this partial cancellation to be provably robust under all bit insertion and deletion scenarios, with high probability. The "bit-stuffed" packet will then be divided into at least $\phi+1$ slices by the positions (at least ϕ of them as explained earlier) at which the phantom insertions or deletions occur. Each slice may contain one to several segments. It can be proved that no insertion or deletion happens to at least one out of these slices, which we call a "clean slice". In other words, in a clean slice, the index of every bit, relative to beginning of the "bit-stuffed" packet, is equal to a constant (could be 0) plus that relative to the beginning of the original packet.

Our decoding algorithm then searches for and removes such a clean slice from the packet. A clean slice can be found by shifting the "bit-stuffed" packet around, computing the EEC codewords for the shifted slice, and comparing them with the EEC codeword for the slice in the original packet (equal to the sum of EEC codewords for the segments that the slice consists of). Once a clean slice is found, it is removed from the "bit-stuffed" packet, and this "randomized slicing and shift-and-compare" procedure will be performed on the rest iteratively to identify more clean slices, until these ϕ insertion and deletions errors are localized to within roughly ϕ "dirty slices", each of which ideally contains only a single "dirty segment". The number of bit errors in the original packet can then be estimated from the bit errors that occur in all the clean slices discovered through the above iterative procedure, under a very mild assumption on the fluctuations of bit flipping error rates across the original packet. Note the randomizations performed in these iterations are mutually independent, allowing us to establish accuracy guarantees of our estimation procedure as rigorously as possible.

For a packet n bits long, the encoding overhead of the idEEC is $O(\phi \log(n))$, where $O(\log(n))$ is the encoding overhead of the underlying EEC (ToW in our case). Therefore, asymptotically we are paying only a multiplicative factor of ϕ for making idEEC robust against insertion and deletion errors. We are able to make this encoding overhead so small because an idEEC codeword needs to help us localize an insertion and deletion error only to the segment rather than to the very bit position where it occurs, as in the error correction codes for the insertion and deletion channels [4, 9, 11]. The computational complexity of idEEC decoding is $O(\phi^2\Lambda)$, where Λ is computational complexity of decoding the underlying EEC codewords. This multiplicative factor of ϕ^2 is tolerable when ϕ is quite small (< 10) as in the aforementioned communication applications. Furthermore, when there are only insertions or only deletions, this computational complexity decreases to $O(\phi\Lambda)$, for a clean slice can be found without shifting the "bit-stuffed" packet around in such a situation, as we will show in Sec. 3.

Note that even if the transmitted and the received packets (i.e., bit strings) were placed side-by-side, it is not possible to say with certainty how many insertion, deletion, and substitution errors occur respectively that results in the difference between the two strings. For one thing, it could be impossible to distinguish a substitution error from that caused by a pair of insertion and deletion. However, under mild assumptions such as that insertions and deletions happen with very low probabilities, it can be proven that the "simplest" explanation for this difference – as implied by the calculation of the edit distance between the two strings [8] – is also the most likely and with high probability equal to the ground truth. Therefore, by comparing the estimates of substitution/flipping errors with that implied by the edit distance between the two strings in our evaluations of idEEC, we are not distorting the ground truth in any significant way.

The remainder of this paper is organized as follows. In Sec. 2 we discuss the background and related work most pertinent to this paper. In Sec. 3, we propose the EEC scheme for insertion and deletion channels under homogeneous bit flipping model, and provide detailed corresponding analysis and estimator design. In Sec. 4, we generalize the scheme to heterogeneous bit error models. We evaluate the performance of the schemes experimentally in Sec. 5. Finally, we conclude the paper in Sec. 6.

2. BACKGROUND AND RELATED WORK

In this section, we introduce some background on insertion and deletion channels, and briefly survey the existing error correction codes for such channels. We describe prior approaches for solving bit error rate (BER) estimation problems, and demonstrate the difficulty of doing so in insertion and deletion channels.

2.1 Error Correcting Codes for Insertion and Deletion Channels

Channels that allow insertions and deletions as well as substitution (i.e., bit flipping) errors enjoy a long and rich history, and are remarkably challenging. Insertion and deletion errors do occur in various wireless network environments. For example, poor time synchronization between a sender and a receiver may lead to some bits (of a packet) being deleted, or some non-existing random bits being inserted [10]. While much progress on error correction codes for such insertion/deletion channels has been made in the past decade, a coding scheme with provably asymptotically optimal coding efficiency remains elusive thus far [10].

In most of the literatures on insertion and deletion channels, error correcting coding has been playing a key role. Some researchers started with the additional properties of the channels, and proposed 1-deletion / 1-insertion correcting codes. In [9], it has been assumed that there is a minimal gap between insertions and deletions, as many synchronization errors are due to small drifts in clock synchronization and a minimal amount of time is needed before the timing mismatch results in another bit error. They made a segmentation assumption that there is at most one insertion or deletion per segment (a contiguous chunk of a packet), which appears natural for many practical settings. In [4], Reed-Muller RM $(1, m)$ codes were studied and used for correcting substitution errors over channels in which a bit sampling error could cause a synchronization error (similar but not identical to an insertion/deletion error). This scheme can correct multiple substitution errors in the presence of one synchronization error. In [11], codes for correcting synchronization errors were designed, which were capable of correcting a certain number of synchronization errors, in the presence of one substitution error.

2.2 Error Estimating for Channels Without Insertion and Deletion Errors

While there has been much work on error correction codes, error estimating coding began to be studied only recently in the seminal work of Chen et al. [2]. The basic idea of this work [2] is for the transmitter to send along with a packet a set of parity-check bits, each of which is the exclusive-or of a group of bits randomly sampled from the packet. These parity equations are designed in such a way that, by counting how many of them are violated after the packet transmission, the receiver can estimate, with low relative error, this BER.

Hua et al. made further studies on error estimation coding in [5] and [6]. In [5], they used tug-of-war (ToW) sketch, which is a sketch data structure to estimate the L_2 norm of a data stream, for Hamming distance estimation. The ToW sketch of a bit array (packet) $\vec{b}$ consists of a set of sub-sketches, each of which is set to the inner product of $\vec{b}$ with a pre-defined pseudorandom vector $\vec{s}$. Upon the receipt of the transmitted bit array $\vec{b'}$ and the sketch, the receiver computes the inner product of $\vec{b'} \cdot \vec{s}$, takes the difference between it and that contained in the corresponding sub-sketches, and squares the result. Furthermore, they proved that even when approximation and randomization are allowed, the cost of BER estimation to achieve an (ϵ, δ)-approximation is $\Omega(\log(n))$, and EEC in [2] is therefore asymptotically optimal. They also showed however that EEC had not achieve the optimal tradeoff down to the constant factor, and proposed the *Enhanced Tug-of War* (EToW) sketch that attains a better constant factor. In [6], they present an information-theoretic study of EEC, stemming from the question whether EEC achieves the best tradeoff between the space and estimation accuracy, through the lens of Fisher information analysis. This analysis led to a new and better EEC scheme, called "gEEC", which could hold $25 - 35\%$ more information – and hence deliver better estimation accuracy – with the same coding overhead.

Though these approaches can all estimate the BER with good accuracies, they cannot handle insertion and deletion channels. This is because, when an insertion or deletion error occurs, certain parts of the packet will be shifted, and about half of these shifted – but correct – bits will be counted as substitution errors by existing EEC techniques. Our idEEC scheme, to be described next, is designed to fill this gap.

3. DESIGN AND ANALYSIS OF IDEEC UNDER HOMOGENEOUS BIT FLIPPING MODEL

This section describes the encoding and decoding algorithms of idEEC. Our goal is to estimate the number of bit flipping errors as accurately as possible, despite the presence of insertion and deletion errors. Here the estimator used in the decoding algorithm assumes that the flipping error happens to each bit independently and with an identical – but unknown – small probability $p << 1$. This model will be much relaxed later in Sec. 4. We further assume that the probability with which an insertion or deletion error occurs, referred to as p_e, is much smaller than the flipping error probability p. This is clearly a reasonable assumption since it would otherwise be impossible to distinguish between insertion/deletion errors and flipping errors. In the following, we will first give an overview of idEEC in Sec. 3.1, then describe the key routine of identifying a clean slice in Sec. 3.2, and finally present the detailed analysis of the accuracy of our estimations in Sec. 3.3.

3.1 Overview

Like in the Code Division Multiple Access (CDMA) literature, we view a '0' bit as symbol '-1', and a '1' bit as symbol '+1'. Then a packet $\vec{b}$ is modeled as a symbol vector of length n, with each symbol takes value in $\{-1, +1\}$. In idEEC encoding, the packet is first divided into m segments of equal lengths, say $\vec{b} = [\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_m]$. For idEEC to be robust against insertion and deletion errors, m should be at least several times larger than the actual number of insertion and deletion errors, which is typically quite small, as explained before. For each segment $\vec{b}_i$, we generate its EEC codeword. Our idEEC codeword is simply these EEC codewords pieced together. Since this idEEC codeword itself may be subject to insertion/deletion and bit flipping errors during transmission, it needs to be protected by error detection or correction codes against such errors. Since the length of such error detection or correction codes will only be a fraction of that of the idEEC codeword, this extra protection will not increase the asymptotic coding overhead of the idEEC scheme. The details of our encoding scheme are given in Algorithm 2.

The procedures for encoding and decoding of a bit array $\vec{b}$ using plain vanilla tug-of-war sketch are illuminated in Algorithm 1. The ToW sketch consists of a constant number c of counters, each of which is generated by the inner product of $\vec{b}$ with a pre-defined pseudorandom vector $\vec{s}_j \in \{+1, -1\}^{|\vec{b}|}$. At the time of decoding, the receiver computes the inner products of the transmitted bit array

Algorithm 1 The tug-of-war sketch for EEC [5].

SKETCH–CREATION($\vec{b}$)

Input $\vec{b}$: original data bits vector.
Output $\vec{q}$: the sketch encoding $\vec{b}$.
pre-compute random vectors $\vec{s}_{j,1\le j\le c} : [|\vec{b}|] \to \{-1, 1\}$
for $j = 1$ to c **do**
  $\vec{q}_j := (\vec{b} \cdot \vec{s}_j)/2$
return $\vec{q} = \langle \vec{q}_1, \ldots, \vec{q}_c \rangle$

DISTANCE–ESTIMATION($\vec{b}', \vec{q}$)

Input $\vec{b}'$: received data bits vector, $\vec{q}$: received sketch.
Output $\hat{d}$: the estimated number of flipping error.
pre-compute random vectors $\vec{s}_{j,1\le j\le c} : [|\vec{b}|] \to \{-1, 1\}$
for $j = 1$ to c **do**
  $X_j := (\vec{q}_j - \vec{b'} \cdot \vec{s}_j/2)^2$
return $\hat{d}$=average($X_1, \ldots, X_c$)

Algorithm 2 Encoding of idEEC.

Input: $\vec{b} = [\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_m]$: original data bits vector.
Output: $\vec{z}$: the sketches encoding $\vec{b}$.
1: **for** $i = 1$ to m **do**
2: $\vec{q}_i :=$ EEC.SKETCH-CREATION($\vec{b}_i$)
3: **return** $\vec{z} = \langle \vec{q}_1, \ldots, \vec{q}_m \rangle$

$\vec{b}'$ with $\vec{s}_j$, $j = 1, 2, ..., c$, takes the differences between the c inner products and the c counters respectively, and infer the hamming distance $\hat{d}$ between $\vec{b}$ and $\vec{b}'$ from these differences.

Upon the receipt of a packet $\vec{b}'$, the original (before any insertion/deletion) length of the packet l (sent along with the packet and could also be protected by the error correction or detection codes that protect the idEEC codeword), and the sketch $\vec{z}$, the receiver uses a iterative algorithm EST to estimate the BER during the transmission. The EST algorithm returns two values, namely $\hat{d}'$ and l', where $\hat{d}'$ denotes the estimated number of flipping errors in all the clean slices in which the idEEC decoder believes no insertion or deletion errors occur, and l' is the total length of these clean slices. Therefore, the number of flipping errors can be estimated by $\hat{d} = \hat{d}' \cdot \frac{l}{l'}$, and one can further obtain the BER estimation by $\hat{p} = \hat{d}/l$. The accuracy of this estimator will later be analyzed in Sec. 3.3. However, our algorithm may return "Failed" meaning that it is not able to provide an accurate estimate. There are two possible ways for this to happen. First, our algorithm may find that the number of insertions and deletions is larger than the pre-determined upper bound. Second, our algorithm may find that the number of flipping errors exceeds a pre-determined upper bound. In both situations, the idEEC decoder will inform us which situation causes it to "fail". The aforementioned decoding procedure of our idEEC scheme is shown in Algorithm 3. The detailed design and analysis of the iterative algorithm EST will be presented in the following sections.

The receiver views a packet as a "manifold", identifying the first bit of a packet with the bit immediately following the last bit. When cutting up packets into slices for decoding, the receiver will randomly choose a starting position. As will be explained shortly, such randomization is needed for the decoder to be robust against insertions and deletions that happen to the beginning and/or the end of the original packet. Furthermore, the aforementioned shifting-and-compare procedure needed for identifying clean slices will also benefit from this "manifoldization".

Algorithm 3 Decoding of idEEC.

Input: $\vec{b}'$: received data bits vector, $\vec{z}$: received sketch, l: length of original packet $\vec{b}$.
Output: $\hat{d}$: the estimated number of flipping errors.
/ l' is the length of selected part of packet in which there is no insertion or deletion. $\hat{d}'$ is the estimator for the number of flipping errors in this selected part. */*
1: $\langle \hat{d}', l' \rangle =$ EST $(\vec{b}', \vec{z}, l)$
2: **if** $l' \neq 0$ **and** $\hat{d}' \cdot \frac{l}{l'} \le D$ **then**
3: **return** $\hat{d} = \hat{d}' \cdot \frac{l}{l'}$
4: **else**
5: **return** Failed

3.1.1 Case with Only Insertions or Only Deletions

In this section, we study the simpler case where either insertion errors or deletion errors – but not both – occur during the transmission, in addition to flipping errors. We believe this is the more common case, since insertion and deletion errors are often caused by clock speed mismatch between the sender and the receiver, and this mismatch should not change direction often.

We propose a divide and conquer algorithm called EST($\vec{x}, \vec{z}, h$) that given a packet or a packet residue $\vec{x}$ – which is the received packet at the first iteration but later becomes the "residue" of the packet with more and more clean slices removed during the iterations – and its idEEC codeword $\vec{z}$, tries to identify all clean slices within $\vec{x}$, and returns (a) the total number of bit errors in them and (b) the total length of all these clean slices. Here h represents the length of what remains of the packet after zero or more clean slices had been removed from the packet during the recursive execution of the "Est" routine up until this point. The description of the algorithm can be found in Algorithm 4.

In describing algorithm EST, we assume there are only insertions. In cases where there are only deletions, we need only replace the phantom deletions by phantom insertions in Step 6 of the algorithm. One may infer the number of insertions $\hat{\phi}$ from the difference between the lengths of the received packet and the original one. In each iteration of the algorithm, it randomly deletes $\hat{\phi}$ (phantom) bits uniformly along the segment boundaries in the packet, to "cancel out" the actual insertions in the transmitted packet. By the $\hat{\phi}$ positions at which bits are deleted, the packet is divided into $\hat{\phi}+1$ slices. From the pigeonhole principle, there should be at least one slice that has no insertions in it, which we call a "clean slice". Furthermore, with Lemma 1 which will be proven in Sec. 3.2, one should be able to find a clean slice that not only has no insertion errors in it, but also has the same starting bit index, relative to the beginning of the "bit-destuffed" packet, as that relative to the beginning of the original packet. We call such a clean slice a "perfect match". We will show in Theorem 1 that, our scheme benefits from the existence of such a perfect match to reduce the computational overhead by a factor of ϕ.

This algorithm first performs a hypothesis testing to confirm the suspected number of insertions $\hat{\phi}$, by trying to find such a perfect match, after these $\hat{\phi}$ phantom deletions are made. Intuitively, when we try to estimate the number of bit errors in a dirty slice (in which insertion and deletions errors are localized) from its EEC codeword, this number is going to be quite large, with high probability, because a single insertion may lead to many bits being shifted and

Algorithm 4 The iterative algorithm for EEC with insertions only.

EST($\vec{x}, \vec{z}, h$)

Input: $\vec{x}$: received data bits; $\vec{z}$: received sketches corresponding to $\vec{x}$; h: length of the original data bits.

Output: $\langle \hat{d}, \hat{h} \rangle$: $\hat{d}$ is the estimated number of flipping errors in all clean slices, $\hat{d}$ is the total length of clean slices.

1: $\hat{\phi} = |\vec{x}| - h$

2: /* *Test the stopping criterion.* */
3: **if** $\hat{\phi} \geq \frac{hm}{l}$ **or** $\hat{\phi} > \phi_{max}$ **then**
4: **return** $\langle 0, 0 \rangle$

5: /* *Phantom deletions.* */
6: Delete $\hat{\phi}$ phantom bits at random positions, and divide $\vec{x}$ into $\hat{\phi}+1$ slices $\langle \vec{x}_1, \ldots, \vec{x}_{\hat{\phi}+1} \rangle$ with corresponding sketches $\langle \vec{z}_1, \ldots, \vec{z}_{\hat{\phi}+1} \rangle$

7: /* *Hypothesis test if* $\hat{\phi} = \phi$ */
8: **for** $j = 1$ **to** $\hat{\phi}+1$ **do**
9: $\hat{d}_j = \text{EEC.DISTANCE-ESTIMATION}(\vec{x}_j, \vec{z}_j)$
10: **if** $\hat{d}_t = \min_j \{\hat{d}_j\} < threshold$ **then**
11: /* *Find a clean slice and iteratively perform the algorithm on the others.* */
12: $\vec{x'} \Leftarrow$ remove $\vec{x}_t$ from $\vec{x}$;
$\vec{z'} \Leftarrow$ remove $\vec{z}_t$ from $\vec{z}$;
$h_t = |\vec{x}_t|$, $h' = h - h_t$
13: $\langle \hat{d}'_t, h'_t \rangle = \text{EST}\ (\vec{x'}, \vec{z'}, h')$
14: **return** $\langle \hat{d}_t + \hat{d}'_t, h_t + h'_t \rangle$
15: **else**
16: /* *Unable to find a clean slice, then test the hypothesis whether there are both insertions and deletions.* */
17: **return** GEST($\vec{x}, \vec{z}, h, 1$)

hence a totally different EEC codeword computed. Therefore, a clean slice can be identified through the following threshold-based approach. The threshold is given by $\beta \frac{Dh}{(\hat{\phi}+1)l}$, where $\beta > 1$ is a small constant, D is the upper bound of the flipping errors in a packet, and thus $\frac{Dh}{(\hat{\phi}+1)l}$ is the average number of flipping errors in each slice. If the estimated number of bit errors in a slice is above the threshold, it is considered dirty. Otherwise, it is considered clean. If there appears more than one clean slices, our algorithm will only save the cleanest slice (i.e., with fewest estimated flipping errors in it). It then removes this slice from the packet (residue), and repeats the procedure iteratively on the rest, until a stopping criterion is met (described shortly). If our scheme fails to identify such a clean slice in this iteration, it has reason to suspect that both insertions and deletions have occurred. In this case, we call a more sophisticated algorithm GEST, which will be explained in Sec. 3.1.2.

Fig. 1 shows an example indicating how this algorithm works when 3 insertion errors occurred during the transmission. The algorithm randomly divides the packet into 4 slices and deletes 3 phantom bits. The Hamming distance between each slice in the received packet and that in the original packet is estimated using the corresponding tug-of-war counters. Based on these estimated Hamming distances, we are able to identify the third slice as a perfect match, with high probability. The Hamming distance estimation for the third – and perfectly matched – slice is then used to estimate the bit error rate of the transmission. Note that although there is no insertion or deletion error in the second slice, this slice

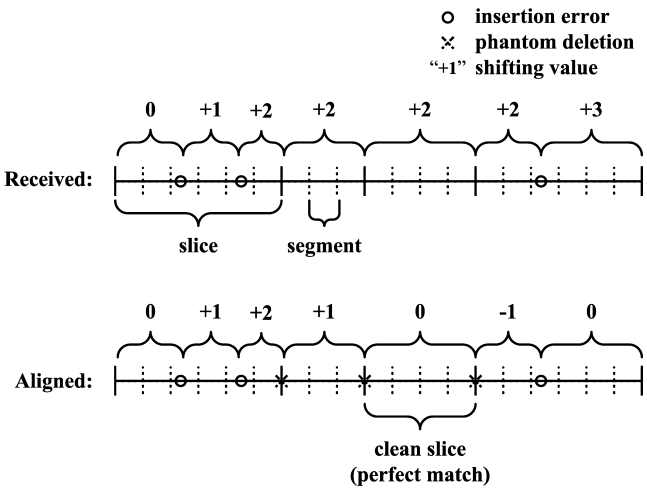

Figure 1: One of the iterations when there are 3 insertions in the received packet.

does not align with the original part, as all the bits are right shifted by one.

The computational complexity of the iterative algorithm when there are only insertions or only deletions is given by the following theorem.

THEOREM 1. *Let Λ denote the total number of operations involved in Hamming distance estimations for all slices in the original packet, using the underlying EEC. If there are only insertions or only deletions in the received packet, the computational complexity of our algorithm is $O(\phi\Lambda)$.*

PROOF. Let s_0 and h_0 denote the total number of operations and number of iterations, respectively. When there are only insertions or deletions, with high probability, the algorithm will keep operating on the packet (residue) iteratively, which shrinks by approximately a multiplicative factor of $\frac{\phi}{\phi+1}$ in length after each iteration instead of calling GEST. Therefore, the number of operations in the j-th iteration is $\left(\frac{\phi}{\phi+1}\right)^{j-1}\Lambda$, and the total complexity is

$$\begin{aligned}\mathbf{E}[s_0] &= \sum_{j=1}^{h_0} \left(\frac{\phi}{\phi+1}\right)^{j-1} \Lambda \\ &= (\phi+1)\left[1 - \left(\frac{\phi}{\phi+1}\right)^{h_0}\right]\Lambda \\ &\leq (\phi+1)\Lambda \\ &= O(\phi\Lambda)\end{aligned}$$

□

Therefore, we are paying only a multiplicative factor of ϕ on the computational overhead of underlying EEC scheme for making idEEC robust against up to ϕ insertion or deletion errors.

3.1.2 Case with Both Insertions and Deletions

Cases where there are both insertions and deletions are more complicated. For one thing, even if we have made a correct hypothesis on the number of insertions and deletions, it is still possible that we cannot find a perfect match using the above algorithm without shifting the packet around. Fig. 2 shows such an example. In case 1, the third slice is a perfect match, because there is one insertion

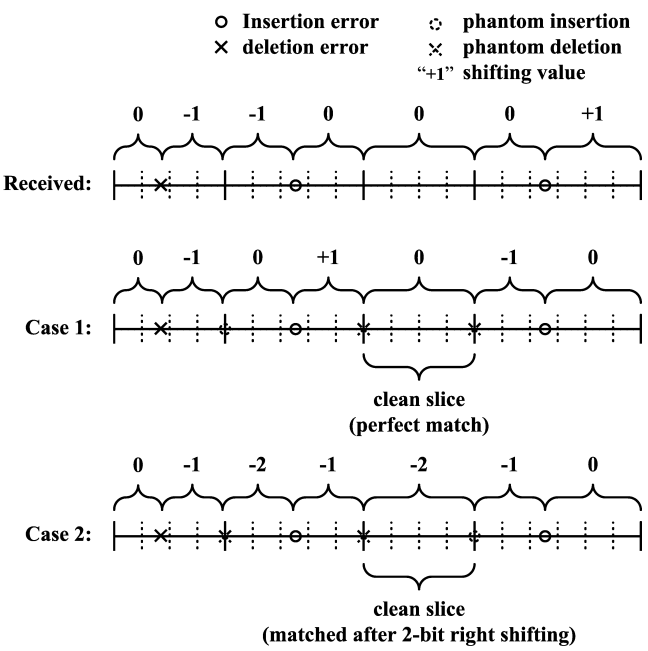

Figure 2: One of the iterations when there are 2 insertions and 1 deletion in the received packet.

and one deletion in the first two slices respectively, and we happen to insert 1 bit and delete 1 bit in these two slices respectively, resulting in no shift to the bit indices of the third (clean) slice. However, in case 2, the third slice is shifted to the left by 2 bits because of the mismatch between the phantom insertions and deletions we make and the actual insertion/deletion errors that occur during transmission. In this case, the algorithm has to shift these slices around in trying to identify a clean slice.

Algorithm 5 shows the details of the generalized estimation of number of flipping errors over both insertion and deletion channels. At level k, we make a hypothesis on the number of insertions n_i and the number of deletions n_d according to the length of the received packet $|\vec{x}|$ and that of the original one h. If we observe that the received packet is longer than the original one ($|\vec{x}| > h$), the hypothesis is that there are $\hat{n}_i = |\vec{x}| - h + k$ insertions and $\hat{n}_d = k$ deletions. Otherwise, $\hat{n}_i = k$, and $\hat{n}_d = h - |\vec{x}| + k$. Then we randomly insert $\hat{n}_d$ and delete $\hat{n}_i$ phantom bits in the packet and divide the packet into $\hat{\phi} + 1$ slices from the $\hat{\phi} = \hat{n}_i + \hat{n}_d$ insertion or deletion positions. The algorithm then attempts to find the clean slices by shifting the packet around to align it with the original one. When a clean slice is identified, the algorithm will remove such a slice from the packet and perform the search (for clean slices) on the rest iterative at the same level k until every single clean slice is located. If not a single clean slice can be identified at this level, which means that the current hypothesis testing has failed, the algorithm will proceed to the next (i.e, the $(k+1)$-th) level, testing the hypothesis that there is one more insertion error and deletion error each in the packet.

The aforementioned threshold-based approach is again taken here to identify a clean slice. However, as insertions and deletions may cancel each other out, it might be difficult to distinguish the flipping errors from the misaligned bits caused by insertions and deletions since the latter could even be smaller than the former, if such insertions or deletions happen very close to the boundary of a slice. Therefore, in order to bound the estimator's error to within a reasonable range, it is assumed that there is a minimal gap between the location of an insertion and that of a deletion. This assumption is in fact quite weak: We need only assume a minimal gap between an insertion and a deletion, and need not make any assumption concerning the distance between insertions or that between deletions like in [9].

Algorithm 5 The iterative algorithm for EEC with both insertions and deletions.

GEST($\vec{x}, \vec{z}, h, k$)

Input: $\vec{x}$: received data bits; $\vec{z}$: received sketches corresponding to $\vec{x}$; h: length of the original data bits; k: level of recursion.

Output: $\langle \hat{d}, \hat{h} \rangle$: $\hat{d}$ is the estimated number of flipping errors in all clean slices, $\hat{d}$ is the total length of clean slices.

1: /* *Estimate the number of insertions and deletions.* */
2: **if** $|\vec{x}| > h$ **then**
3: $\quad \hat{n}_i = |\vec{x}| - h + k$, $\hat{n}_d = k$, $\hat{\phi} = \hat{n}_i + \hat{n}_d$
4: **else**
5: $\quad \hat{n}_d = h - |\vec{x}| + k$, $\hat{n}_i = k$, $\hat{\phi} = \hat{n}_i + \hat{n}_d$
6: /* *Test the stopping criterion.* */
7: **if** $\hat{\phi} \geq \frac{hm}{l}$ **or** $\hat{\phi} > \phi_{max}$ **then**
8: $\quad$ **return** $\langle 0, 0 \rangle$
9: /* *Phantom insertions and deletions.* */
10: Delete $\hat{n}_i$ and insert $\hat{n}_d$ phantom bits at random positions.
11: /* *Hypothesis test if* $\hat{\phi} = \phi$ */
12: **for all** $\sigma \in \{0, \pm 1, \pm 2, \ldots, \pm\hat{\phi}\}$ **do**
13: $\quad$ /* *Test all the shifting to find a clean slice.* */
14: $\quad \vec{x}_\sigma \Leftarrow$ Right shift $\vec{x}$ σ bits (Left shift if $\sigma < 0$).
15: $\quad$ Divide $\vec{x}_\sigma$ into $\hat{\phi} + 1$ slices $\langle \vec{x}_{\sigma,1}, \ldots, \vec{x}_{\sigma,\hat{\phi}+1} \rangle$ by the positions of phantom insertions and deletions, with corresponding sketches $\langle \vec{z}_1, \ldots, \vec{z}_{\hat{\phi}+1} \rangle$
16: $\quad$ **for** $j = 1$ **to** $\hat{\phi} + 1$ **do**
17: $\quad\quad \hat{d}_{\sigma,j} =$ EEC.DISTANCE-ESTIMATION$(\vec{x}_{\sigma,j}, \vec{z}_j)$
18: **if** $d_{s,t} = \min_{\sigma,j}\{\hat{d}_{\sigma,j}\} < threshold$ **then**
19: $\quad \vec{x'} \Leftarrow$ remove $\vec{x}_{s,t}$ from $\vec{x}$;
$\quad\quad \vec{z'} \Leftarrow$ remove $\vec{z}_t$ from $\vec{z}$;
$\quad\quad h_t = |\vec{x}_{s,t}|$, $h' = h - h_t$
20: $\quad \langle \hat{d}'_t, l'_t \rangle =$ GEST $(\vec{x'}, \vec{z'}, h', k)$
21: $\quad$ **return** $\langle \hat{d}_t + \hat{d}'_t, h_t + h'_t \rangle$
22: **else**
23: $\quad$ **return** GEST$(\vec{x}, \vec{z}, h, k+1)$

The computational overhead of our algorithm in this case can be given by the follow theorem.

THEOREM 2. *Again let Λ denote the total number of operations involved in Hamming distance estimations for all slices in the original packet, using the underlying EEC. If there are both insertions and deletions, the computational complexity is $O(\phi^2\Lambda)$.*

PROOF. Similar to the proof of Theorem 1, we let s_k and h_k denote the total number of operations and number of iterations in the k-th level, respectively. When there are both insertions and deletions, the algorithm first has to make $O(\phi)$ hypothesis testings to find out the actual number of insertions and deletions in the packet, each of which has a computational complexity $O(\phi\Lambda)$. Hence this "exploratory" phase aimed at finding the very first perfect match has complexity $O(\phi^2\Lambda)$.

Once the first perfect match is found, the gEST routine then operates on the "residue" of the packet iteratively to find more perfect

matches. The complexity of each iteration here can be as large as $\left(\frac{\phi}{\phi+1}\right)^{j-1}\phi\Lambda$ – which is ϕ times of that in EST – because the packet residue may need to be shifted around up to ϕ times in order for a perfect match to emerge. The total computational complexity of this second phase is

$$\begin{aligned}\mathbf{E}[s_k] &= \sum_{j=1}^{h_k}\left(\frac{\phi}{\phi+1}\right)^{j-1}\phi\Lambda \\ &= \phi(\phi+1)\left[1-\left(\frac{\phi}{\phi+1}\right)^{h_k}\right]\Lambda \\ &\leq \phi(\phi+1)\Lambda \\ &= O(\phi^2\Lambda)\end{aligned}$$

Therefore the total computational complexity of both phases is $O(\phi^2\Lambda)+O(\phi^2\Lambda)=O(\phi^2\Lambda)$. □

The number of insertion and deletion errors in a packet is usually very small. In such cases, the decoding computational complexity of our algorithm is larger than that of the underlying EEC (for channel that are free of insertion/deletion errors) by only a small multiplicative factor.

3.2 Identifying "Clean Slices"

Recall that our iterative algorithm randomly deletes n_i phantom bits and inserts n_d phantom bits, and divides the packet $\vec{x}$ into $\hat{\phi}+1$ slices. The goal is to find a clean slice – which must exist if the "n_i+n_d" hypothesis is correct – by shifting the slices around. Furthermore, in cases where there are only insertions or only deletions, there must exist a perfect match (a slice where the index of every bit relative to the beginning of the received packet is equal to that relative to the beginning of the original packet) in each iteration, provided the corresponding hypothesis is correct. This property is proven in the following lemma.

LEMMA 1. *If there are only insertions (or deletions), there is at least one slice which is a "perfect match".*

PROOF. We prove this lemma constructively. We first define two pointers a and b, which are initialized as $a=b=1$. Denote the number of insertions (or deletions) in the j-th slice by ϕ_j. If $\sum_{j=a}^{b}\phi_j=0$, the b-th slice is a perfect match. Otherwise, update a to b and b to $b+\sum_{j=a}^{b}\phi_j$ and repeat these steps. Since there are $\sum_{j=1}^{\phi+1}\phi_j=\phi$ insertions (or deletions), we always have $b<\phi+1$. Thus we can finally find a perfect match in the $(\phi+1)$ slices. □

As an intuitive explanation of the proof, one can imagine this as a race between the actual insertions (or deletions) and the phantom deletions (or insertions). Since they are equal in number, they should "meet" each other at some point by the finishing line, which indicates that there should be at least one perfect match throughout the packet. Therefore, when there are only insertions (or deletions), it is unnecessary to shift the packet around to find a "clean slice". This, in addition to the fact that there is no need to test all "n_i+n_d" hypothesis, reduces the computational complexity of the general algorithm (for handling the presence of both insertion and deletion errors) by a multiplicative factor of ϕ.

For more complex cases where there are both insertions and deletions, even when the "n_i+n_d" hypothesis is correct, there might still be no perfect match, as we have shown earlier. However, we can prove similarly that a clean slice, albeit shifted, must exist, when the "n_i+n_d" hypothesis is correct. This slice can then be found by the aforementioned shift-and-compare steps in Algorithm 5. In the following section, we will analyze the threshold-based approach to determining whether a (shifted) slice is clean.

3.3 Mean Square Error Analysis

Once the algorithm has identified the segments that are likely to contain insertion and deletion errors, it uses the remaining "clean" segments to estimate the number of flipping errors. The accuracy of the BER estimation thus depends on how reliably we can identify which slices are *clean* (do not have any insertions or deletions within them) at each iteration. As described before, the decoding algorithm tries to identify clean slices by comparing the decoded Hamming distances against a pre-determined threshold.

There are two ways we can make an error in identifying clean slices: a *false alarm* (FA) when we tag slice as dirty even though it is clean, and a *miss* (M) when we do not tag a dirty slice. False alarms and misses have different effects on the BER estimation. When the algorithm terminates, the segments which have been labeled as "clean" will be used to estimate the number of flipping errors in the entire packet. When a miss occurs, this estimation can include segments which have an insertion/deletion error, which will typically have a higher error rate, which in turn biases our estimate upwards. However, as a miss happens only when the estimator is smaller than the threshold, this bias is negligible.

False alarms affect the estimator in a different way. When a false alarm occurs, the corresponding slice is simply dropped in the current iteration, but it will still be involved in the next iteration. If a false alarm occurs in all of the clean slices, the algorithm will directly go to the next level and re-slice the packet. The overall effect of the false alarm is a slight increase in the total amount of dropped segments, up to $O\left(\phi+m\delta_{FA}\right)$ on average, where δ_{FA} is an upper bound on the false alarm rate.

Let $\hat{d}$ denote the our estimator for the number of flipping errors in the received packet. We measure the performance of our estimator using the standard Mean Square Error (MSE) metric, which is defined as

$$MSE(\hat{d})=\mathbf{E}[(\hat{d}-d)^2]=\mathbf{Var}(\hat{d})+\mathbf{E}[Bias(\hat{d})^2]$$

where $Bias(\hat{d})=\mathbf{E}[\hat{d}]-d$.

Theorem 3 below shows how we can incorporate uniform (over all slices visited) bounds on the false alarm and miss probabilities into the accuracy of the flipping error estimator.

THEOREM 3. *Let E_s be the event that a certain slice s contains an insertion or deletion, and let E_s' be the event that our algorithm labels it as such. Let δ_{FA} be an upper bound on the probability of false alarm,*

$$Pr(E_s',\overline{E}_s)=Pr\left(E_s'|\overline{E}_s\right)Pr(\overline{E}_s)\leq\delta_{FA}\quad \textit{for all slices } s,$$

and δ_M be an upper bound on the probability of a miss,

$$Pr(\overline{E}_s',E_s)=Pr\left(\overline{E}_s'|E_s\right)Pr(E_s)\leq\delta_M\quad \textit{for all slices } s.$$

Then the mean square error of our estimator on the number of flipping errors, $\hat{d}$, is bounded by

$$MSE(\hat{d})=O\left(\frac{d^2}{c}+\delta_M D^2+\frac{d(\frac{\phi}{m}+\delta_{FA})}{1-(\frac{\phi}{m}+\delta_{FA})}\right). \quad (1)$$

PROOF. Since $MSE(\hat{d})=\mathbf{Var}(\hat{d})+\mathbf{E}[Bias(\hat{d})^2]$, we will calculate the variance and bias of $\hat{d}$ respectively in the follows.

We first define some notations that will be used in our proof. Let $\vec{x}$ and $\vec{x'}$ denote the received packet and the selected part of it.

Let $\hat{d}$ and $\hat{d}'$ denote the estimator for the number of flipping errors in $\vec{x}$ and $\vec{x'}$ respectively. Let d and d' denote the actual number of flipping errors in $\vec{x}$ and $\vec{x'}$. Let B and B' denote the bias of $\hat{d}$ and $\hat{d}'$. Let $0 < \theta < 1$ denote the proportion of the dropped part of the packet. Define $D' = (1-\theta)D$. Generally we have $\mathbf{E}[B] = O(\delta_M D)$ and $\mathbf{E}[B'] = O(\delta_M D')$. The variance of $\hat{d}'$ is given by

$$\mathbf{Var}(\hat{d}'|d') = \mathbf{E}_{B'}[\mathbf{Var}(\hat{d}'|B', d')] + \mathbf{Var}_{B'}(\mathbf{E}[\hat{d}'|B', d'])$$

Recall the variance of tug-of-war sketch based EEC given in [5], we have

$$\begin{aligned}
\mathbf{E}_{B'}\left[\mathbf{Var}(\hat{d}'|B', d')\right] &= \mathbf{E}\left[\frac{1}{c}\left(2(d'+B')^2 - 2(d'+B')\right)|d'\right] \\
&\leq \frac{1}{c}\mathbf{E}\left[2(d'+B')^2|d'\right] \\
&= \frac{1}{c}\mathbf{E}\left[2(d'^2 + 2d'B' + B'^2)|d'\right] \\
&= O\left(\frac{2}{c}(d'^2 + 2d'\delta_M D' + \delta_M D'^2)\right) \\
&= O\left(\frac{2}{c}(d' + \sqrt{\delta_M}D')^2\right)
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{Var}_{B'}(\mathbf{E}[\hat{d}'|B', d']) &= \mathbf{Var}_{B'}(d' + B'|d') \\
&= \mathbf{Var}(B') \\
&= \mathbf{E}\left[(B' - \mathbf{E}[B'])^2\right] \\
&= \mathbf{E}\left[(B')^2\right] - \mathbf{E}\left[B'\right]^2 \\
&= \mathbf{E}\left[\left(\frac{B'}{D'}\right)^2\right](D')^2 - \mathbf{E}[B']^2 \\
&\leq \mathbf{E}\left[\frac{B'}{D'}\right](D')^2 - \mathbf{E}[B']^2 \\
&= \mathbf{E}[B']D' - \mathbf{E}[B']^2 \\
&= O\left(\delta_M D'^2 - \delta_M^2 D'^2\right) \\
&= O\left(\delta_M D'^2\right)
\end{aligned}$$

In the first inequality, $(\frac{B'}{D'})^2 \leq \frac{B'}{D'}$ because $\frac{B'}{D'} \leq 1$. Since part of the segments are dropped at the end of the algorithm and the overall flipping error of the received packets is estimated just from the selected ones, sampling errors may introduced to our estimator. Since $\hat{d} = \frac{1}{1-\theta}\hat{d}'$, according to the proof of Theorem 2 in [12], we have

$$\begin{aligned}
\mathbf{Var}(\hat{d}) &\approx \frac{1}{(1-\theta)^2}\mathbf{Var}(\hat{d}'|d' = (1-\theta)d) + \frac{d\theta}{1-\theta} \\
&= \frac{1}{(1-\theta)^2} \cdot O\left(\frac{2}{c}\left((1-\theta)d + \sqrt{\delta_M}D'\right)^2 + \delta_M D'^2\right) \\
&\qquad + \frac{d\theta}{1-\theta} \\
&= O\left(\frac{d^2}{c} + \delta_M D^2 + \frac{d\theta}{1-\theta}\right)
\end{aligned}$$

As to the bias of $\hat{d}$, we have

$$\mathbf{E}[B^2] = O\left(\delta_M D^2\right)$$

Generally, we have $\theta = O\left(\frac{\phi}{m} + \delta_{FA}\right)$. Thus

$$\begin{aligned}
MSE(\hat{d}) &= \mathbf{Var}(\hat{d}) + \mathbf{E}[B^2] \\
&= O\left(\frac{d^2}{c} + \delta_M D^2 + \frac{d\theta}{1-\theta}\right) \\
&= O\left(\frac{d^2}{c} + \delta_M D^2 + \frac{d(\frac{\phi}{m} + \delta_{FA})}{1 - (\frac{\phi}{m} + \delta_{FA})}\right)
\end{aligned}$$

□

Here we offer an intuitive explanation of Theorem 3. The first term in the expression for the MSE, $O(\frac{d^2}{c})$, is the same as that of a standard EEC using tug-of-war sketches when there are no insertion/deletion errors. The additional error terms result from the need to deal with insertions and deletions errors. The second term corresponds to the aforementioned bias introduced by a miss. D is a parameter determined by the channel conditions, which is not tunable in our algorithm. It characterizes to what extent potential fluctuations of the BER across the packet can negatively impact the accuracy of the estimator. The third term is the "sampling error" since we estimate the BER only from up to $m - \phi$ clean slices "sampled" out of a total of up to m slices. Even with this sampling error term alone, it is clear that our algorithm requires the number of segments to be proportional to the number of insertions and deletions. This suggests that the coding overhead of our scheme is $O(m \log n) = O(\phi \log n)$. Therefore, we are paying only a multiplicative factor of ϕ for making idEEC robust against insertion and deletion errors. As an extreme case, when $\phi = 0$, we have $MSE(\hat{d}) = O\left(\frac{d^2}{c} + \frac{d\delta_{FA}}{1-\delta_{FA}}\right)$. The second term, $O\left(\frac{d\delta_{FA}}{1-\delta_{FA}}\right)$, comes from the fact that even if there is no insertion/deletion it is still possible for the estimator to exceed the threshold due to the rare event that the underlying ToW sketch grossly overestimates the Hamming distance. Although $\delta_{FA} \neq 0$ when $\phi = 0$, in Lemma 2 below we'll show that this cost is very small since it decreases exponentially when the number of codeword c increases.

It remains to determine a uniform bound on the false alarm and miss probabilities for our algorithm. Lemmas 2, 3, and 4 below bound these probabilities when the slice s is chosen independently of the sketches and bit errors. These conditions hold strictly for the first iteration of the algorithm; in latter iterations, however, a weak dependence develops between the actual bit errors and which slice is the "cleanest" in what remains of the packet (i.e., the "packet residue"). Our analysis ignores this weak dependence and therefore is an approximation from this point on. In the i-th iteration of our algorithm, the threshold for determining whether or not a slice is clean is given by $\alpha_i = \beta\frac{D}{S} \cdot \frac{l_i}{l}$, where $\beta > 1$ is a constant, D is the upper bound on the total number of flipping errors for the entire packet, $S = \hat{\phi} + 1$ is the number of slices at the current iteration, l is the length of the entire packet and l_i is the length of the part of the packet that remains at the i-th iteration.

Let c denote the number of sketches (codewords) for each segment and d denote the actual number of flipping errors in the received packet. The upper bound of the false alarm probability is given by Lemma 2. Its proof is provided in Appendix A.

LEMMA 2. *Consider a slice s chosen independently of the bit errors and sketches. Let E_s be the event that this slice contains an insertion or a deletion, and let E_s' be the event that our algorithm labels it as such. Then if $\phi = 0$, we can bound the probability of*

false alarm with

$$Pr(E_s', \overline{E}_s) < e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{(2\beta-1)c}\right)^2}.$$

For $\phi \neq 0$, we have

$$Pr(E_s', \overline{E}_s) < 1 - \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right) \cdot \left(1 - A_1 e^{-A_2 \cdot \tau_1 \beta \frac{D}{S}}\right)$$

where $0 < \tau_1 < 1$, $A_1 > 0$ and $A_2 > 0$ are constants.

Remark: In the proof of Lemma 2, we have proposed a tighter probability tail bound on the variance of the tug-of-war instance used in our scheme. In particular, our technique improved the bound from vanishing quadratically fast to vanishing exponentially fast. However, this improvement is restricted to the case when the vectors at issue are binary and does not apply to the tug-of-war sketch in general.

For the miss probability, if $\phi = 0$, each slice in the packet is a clean slice, meaning $Pr(E_s) = 0$, and so $Pr(\overline{E}_s', E_s) = 0$ as well.

When $\phi \neq 0$, and if there are only insertions or only deletions in the received packet, the probability of a miss is bounded by Lemma 3 below, whose proof is given in Appendix B.

LEMMA 3. *Consider a slice s chosen as in Lemma 2, and suppose that there are only ϕ insertions or only ϕ deletions in the received packet. If $\phi \neq 0$, for any $0 < \tau_1 < 1$, $\tau_2 \geq 2$ and $\epsilon > 1$, we have the following bound on the miss probability*

$$Pr(\overline{E}_s', E_s) < 1 - \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right)\left(1 - e^{-\left(\frac{\tau_2-2}{2\tau_2}\right)^2 c}\right) \cdot \left(1 - f_1(d,S)e^{-f_2(d,S)\cdot\tau_1\epsilon\frac{d}{S}}\right)\left(1 - 2\tau_2\epsilon \cdot \frac{\phi d}{l}\right)$$

where $f_1(d,S)$ and $f_2(d,S)$ are functions of d and S.

If there are both insertions and deletions in the received packet, it is more difficult to distinguish the flipping errors from the mismatched bits caused by insertions and deletions, as an insertion and an deletion that are close together may cancel each other out. In [9], it is assumed that there is a minimal gap between two insertion or deletion errors, as synchronization errors that cause such insertions or deletions often result from a slow drift in clock synchronization and in this case it takes some time for this drift to build up to cause another synchronization error. Here we make a weaker assumption that there is a minimum gap between an insertion and a deletion (but two insertions or two deletions could occur to bits very close to each other). With this weaker assumption, the miss probability is given by Lemma 4. Its proof can be found in Appendix C.

LEMMA 4. *Suppose there are both insertions and deletions in the received packet, $\phi \neq 0$, and let Δ denote the minimal gap between insertions and deletions. Consider a slice s chosen as in Lemmas 2 and 3. If $\Delta > \max\{\tau_2\beta\frac{D}{S}, \tau_2\epsilon\frac{D}{S}\}$, then we have the following bound on the miss probability*

$$Pr(\overline{E}_s', E_s) < \begin{cases} \max\{\delta_1, \delta_2\} & \text{if } S < \phi + 1 \\ \delta_2 & \text{otherwise} \end{cases}$$

δ_1 and δ_2 are defined as

$$\delta_1 = 1 - \left(1 - e^{-\left(\frac{\tau_2-2}{2\tau_2}\right)^2 c}\right)(1 - 2\tau_2\beta \cdot \frac{\phi D}{l})$$

$$\delta_2 = 1 - \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right)\left(1 - e^{-\left(\frac{\tau_2-2}{2\tau_2}\right)^2 c}\right) \cdot \left(1 - f_1(d,S)e^{-f_2(d,S)\cdot\tau_1\epsilon\frac{d}{S}}\right)\left(1 - 2\tau_2\epsilon \cdot \frac{\phi d}{l}\right)$$

where $f_1(d,S)$ and $f_2(d,S)$ are functions of d and S. $0 < \tau_1 < 1$, $\tau_2 \geq 2$ and $\epsilon > 0$ are constants.

Remark: $f_1(d,S)$ and $f_2(d,S)$ in Lemma 3 and Lemma 4 are factors from Chernoff bound and can be calculated by numerical methods. When $d \to 0$, we have $f_1(d,S) \to 1$ and $f_1(d,S) \to \infty$. τ_1, τ_2 and ϵ in Lemma 2, Lemma 3 and Lemma 4 are constants used to balance the terms in the bounds. One may adjust these factors to make the bound tight enough in different scenarios. As an extreme case, in Lemma 3 when $d \to 0$, we could have the miss probability $Pr(\overline{E}_s', E_s) \to 0$ by making $\tau_1 \to 0$, $\tau_2 \to \infty$ and $\epsilon \to \infty$.

4. EEC UNDER HETEROGENEOUS BIT FLIPPING MODEL

The proposed scheme for accounting for insertion and deletion errors can be extended to the more general case where the bit-flip probability is different in different portions of the packet. The mitigation strategy discussed above depends only on finding gross differences in the mismatch between (re)encoding of the received slices and the code; it is relatively insensitive to subtle variations in the bit error rate.

Throughout the procedure, an estimate of the bit error rate in each of the m segments, $P = [p_i]_{1 \leq i \leq m}$ is maintained. In Step 14 of Algorithm 4 and Step 21 of Algorithm 5, the algorithm not only returns the number of flipping errors $\hat{d}_t$ in the clean slice $\vec{x}_t$, but also stores the estimated BER $p_i = \frac{m\hat{d}_i}{l}$ of each segment $\vec{b}_i$ in the slice, where $\hat{d}_i$ is the estimated Hamming distance between the transmitted segment and the original one. On finishing all the iterations of the algorithm, we can treat this set of estimates as a discrete set of data points tells us about the BER in different regions of the packet. With the segments containing insertion/deletion errors removed, the proportion of bits left in the packet we can use to estimate the BER is at least $\Theta\left((m-\phi)/m\right)$.

The type of information we have about the channel might affect how we combine these estimates in each segment. For example, if we are confident that the bit error rate is changing very little throughout the time it takes to transmit the packet, we might simply average the estimates in each segment. Otherwise, if we have some type of statistical model for how the error rate is changing in time, we might weigh this against our estimates in a Bayesian framework. In particular, if we believe the channel is being affected by "deep fades" which are very localized in time (within one packet), we can imagine detecting and segmenting these fades.

5. EVALUATION

In this section, we evaluate the accuracy of the BER estimation made by idEEC using simulation experiments with various parameter settings. We simulate unreliable communication channels where insertion, deletion and substitution errors may occur during transmission. In order for "enough" insertion and deletion errors to occur despite their low occurence probability, the length of the packet

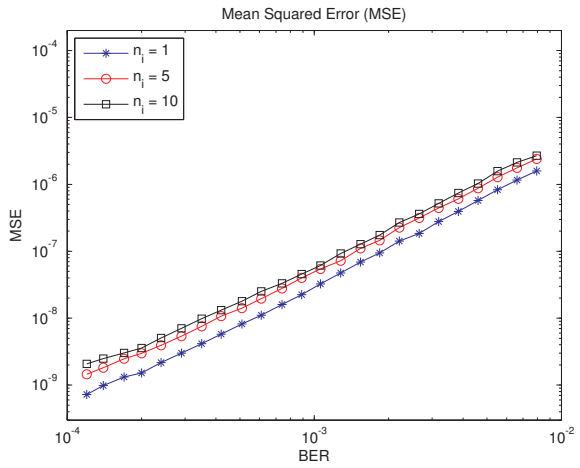

Figure 3: Empirical Results of Estimators for Insertion-Only Channels.

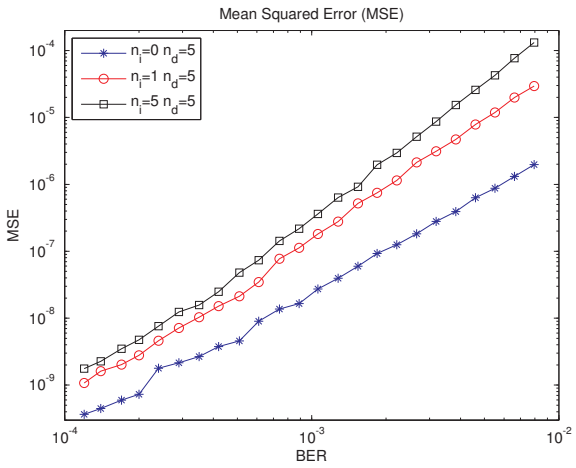

Figure 4: Empirical Results of Estimators for Insertion and Deletion Channels.

is set to $l = 10^5$ bits. For the results to converge properly, each value in each result in this study is obtained by averaging 1,000 simulation runs for it to converge properly. In all experiments, the packet is divided into $m = 50$ segments, each of which is coded by $c = 5$ tug-of-war codewords.

In our experiments, we vary the actual BER and compare it with the estimated number of flipping errors $\hat{p}$, the estimated number of flipping errors $\hat{d}$ divided by the packet length l. In this way, we obtain the empirical MSE's of the estimator under various parameter settings, which be plotted in the following two figures.

We start with the cases where there are only insertions or only deletions. Here we only present the insertions-only cases, since the deletions-only cases have identical performance characteristics. Fig. 3 shows the accuracy of the idEEC scheme under a set of different BER's and three different numbers of insertion errors ($n_i = 1, 5, 10$) in a log-log plot. We observe that the MSE of the estimation grows roughly quadratically with the actual BER in all three cases, as predicted by our analytical BER formula in Theorem 3. In other words, the average relative estimation error remains roughly a constant when the actual BER increases.

Next, we present the cases where there are both insertions and deletions. We fix the number of deletions n_d to 5 and let the number of insertion n_i vary across three different values $0, 1, 5$. The experimental results are shown in Fig. 4. We observe that the lowest line (corresponding to $n_i = 0, n_d = 5$) is almost identical to the middle line (corresponding to $n_i = 5, n_d = 0$) in Fig. 3 as expected. We observe also that we pay a steep price in terms of the MSE increment – as reflected by the wide gap between the lowest line and the middle line – for having just one insertion (i.e., $n_i = 1$). In comparison, this price becomes much smaller thereafter, as reflected by the much narrower gap between the highest line ($n_i = 5$) and the middle line, even though four more insertions are introduced. This behavior is expected since when there are both insertions and deletions, the decoding process becomes more complicated and error-prone. It is also consistent with that predicted by the analytical BER formula. We observe also that when there are one or more insertions, the MSE grows more than quadratically with the actual BER, which implies that the average relative estimation error grows larger with the actual BER.

6. CONCLUSION

Error estimating coding is a novel technique for measuring the bit error rate, with applications towards improving performance of wireless network where recent emerging systems can leverage partially correct packets [2]. However, the existing approaches [2, 5, 6] meet challenges in insertion or deletion channels. In this paper, we designed idEEC scheme that allows for the estimation of the number of flipping errors in a packet despite the existence of bit insertions and deletions during the transmission. Our idEEC scheme can build on any existing EEC scheme, with encoding efficiency proportional to that of the underlying EEC scheme. Taking the plain vanilla tug-of-war sketch as the underlying EEC for example, our design provides provable estimation quality with rather low overhead. The efficacy of our approach has been demonstrated both theoretically and experimentally. To our knowledge, this work represents the first study of building EEC for insertion and deletion channels.

7. ACKNOWLEDGEMENTS

This work has been supported in part by the collaborative NSF project that includes awards CNS 1218092 and CNS 1217758, and by the NSF award CNS 0910592.

8. REFERENCES

[1] ALON, N., GIBBONS, P. B., MATIAS, Y., AND SZEGEDY, M. Tracking join and self-join sizes in limited storage. *Journal of Computer and System Sciences 64*, 3 (2002), 719–747.

[2] CHEN, B., ZHOU, Z., ZHAO, Y., AND YU, H. Efficient error estimating coding: feasibility and applications. In *Proceedings of ACM SIGCOMM* (2010), pp. 3–14.

[3] DAVID, H. A., AND NAGARAJA, H. N. *Order statistics*. John Wiley & Sons, Inc., 1970.

[4] DOLECEK, L., AND ANANTHARAM, V. Using Reed-Muller RM $(1, m)$ codes over channels with synchronization and substitution errors. *IEEE Transactions on Information Theory 53*, 4 (2007), 1430–1443.

[5] HUA, N., LALL, A., LI, B., AND XU, J. A simpler and better design of error estimating coding. In *Proceedings of IEEE INFOCOM* (2012), pp. 235–243.

[6] HUA, N., LALL, A., LI, B., AND XU, J. Towards optimal error-estimating codes through the lens of fisher information analysis. In *Proceedings of ACM SIGMETRICS* (2012), pp. 125–136.

[7] LAURENT, B., AND MASSART, P. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics 28*, 5 (2000), 1302–1338.

[8] LEVENSHTEIN, V. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission 1*, 1 (1965), 8–17.

[9] LIU, Z., AND MITZENMACHER, M. Codes for deletion and insertion channels with segmented errors. *IEEE Transactions on Information Theory 56*, 1 (2010), 224–232.

[10] MITZENMACHER, M. A survey of results for deletion channels and related synchronization channels. *Probability Surveys 6* (2009), 1–33.

[11] WANG, H., AND LIN, B. Designing efficient codes for synchronization error channels. In *Proceedings of IEEE IWQoS* (2011), pp. 1–9.

[12] ZHAO, Q., KUMAR, A., WANG, J., AND XU, J. Data streaming algorithms for accurate and efficient measurement of traffic and flow matrices. In *Proceedings of ACM SIGMETRICS* (2005), pp. 350–361.

APPENDIX

A. PROOF OF LEMMA 2

PROOF. Let ξ_i denote the estimator for the number of bit flipping errors that occur to the i-th slice T_i in the received packet. Recall that this estimator is simply the mean of the c estimations given in Algorithm 1.

$$X_j = Y_j^2 = \left(\sum_{\substack{k \in T_i \\ b_k \neq b'_k}} b_k s_j[k] \right)^2 \quad j = 1, 2, \ldots, c$$

each of which is obtained from taking the difference between the j-th counter value $\sum_{k \in T_i} b_k s_j[k]$ encoding the slice in the tug-of-war sketch that was sent along with the packet, and $\sum_{k \in T_i} b'_k s_j[k]$, the j-th counter value computed from the slice in the received packet, and then squaring the difference up.

$$\xi_i = \frac{1}{c} \sum_{j=1}^{c} X_j = \frac{1}{c} \sum_{j=1}^{c} Y_j^2$$

We will show that, if the slice T_i is indeed clean, this estimator is sharp in the sense, with high probability, the estimated value are very close to the real value. By the Central Limit Theorem, Y_j is approximately a Gaussian distributed random variable, thus ξ_i is approximately a chi-square random variable with c degrees of freedom. Recall the tail bounds for chi-square distributed variables given in [7], for a standard chi-square random variable $X \sim \chi_c^2$, we have

$$P(X - c > 2\sqrt{cx} + 2x) \leq e^{-x} \tag{2}$$

$$P(c - X > 2\sqrt{cx}) \leq e^{-x} \tag{3}$$

As shown in Step 7 to 12 of Algorithm 4 and Step 11 to 19 of Algorithm 5, in each iteration, after attempting all kinds of shift of the packet, the algorithm chooses the slice with the minimal estimated Hamming distance to the original packet, and test if this slice is a "clean one". If the selected slice in this iteration contains no insertion/deletion but fails the test, a false alarm happens and the slice is declared as a "dirty one". In such a case, the estimator for the number of flipping errors in this slice is $\xi = \min_i \xi_i$.

Here, we only derive the false alarm probability for the first iteration. As the threshold changes proportional to the remaining part of the packet, it's easy to generalize this result to other iterations. Let d_i denote the number of flipping errors in the i-th slice. Define random variables $x_{i,j}$ as

$$x_{i,j} = \begin{cases} 1 & \text{if the } j\text{-th bit in the } i\text{-th slice is flipped} \\ 0 & \text{otherwise} \end{cases}$$

Apparently $x_{i,j} \sim Bernoulli(p)$ and $d_i = \sum_{i=1}^{\eta_i} x_{i,j}$, where η_i is the length of the i-th slice.

When $\phi \neq 0$, we have $S \geq 2$. By the theory of order statistics [3], η_i, $i = 1, 2, \ldots, S$, is approximately a Beta distributed random variable[1] with shape parameters 1 and $S-1$, i.e. $\eta_i \sim l \cdot Beta(1, S-1)$. According to Chernoff bound, for any $t > 0$ we have

$$Pr(d_i \geq \tau_1 \alpha_1) \leq e^{-\tau_1 \alpha_1 t} \mathbf{E}[e^{d_i t}] \tag{4}$$

where $0 < \tau_1 < 1$ and

$$\begin{aligned}
\mathbf{E}\left[e^{d_i t}\right] &= \mathbf{E}\left[e^{\sum_{j=1}^{\eta_i} x_{i,j} t}\right] && (5)\\
&= \mathbf{E}_{\eta_i}\left[\mathbf{E}\left[e^{\sum_{j=1}^{\eta_i} x_{i,j} t} \middle| \eta_i\right]\right] && (6)\\
&= \mathbf{E}_{\eta_i}\left[\left(pe^t + (1-p)\right)^{\eta_i}\right] && (7)\\
&\leq \mathbf{E}_{\eta_i}\left[e^{(e^t-1)\eta_i p}\right] && (8)\\
&\approx \int_0^1 (S-1)(1-\omega)^{S-2} e^{(e^t-1)l\omega p} d\omega && (9)\\
&\leq \int_0^1 (S-1)(1-\omega)^{S-2} e^{(e^t-1)\omega D} d\omega && (10)\\
&= \frac{e^{(e^t-1)D}}{((e^t-1)D)^{S-1}} \left(\Gamma(S-1) - \Gamma\left(S-1, (e^t-1)D\right)\right)
\end{aligned}$$

where $\Gamma(S-1)$ is the gamma function and $\Gamma\left(S-1, (e^t-1)D\right)$ is the upper incomplete gamma function.[2] The inequality (8) holds since $1 + x < e^x$ for any $x > 0$.

Minimize the right-hand side of Equation (4) over $t > 0$, we have

$$\min_{t>0} e^{-\tau_1 \alpha_1 t} \mathbf{E}[e^{d_i t}] = A_1 e^{-A_2 \tau_1 \alpha_1}$$

where $A_1 > 0$ and $A_2 > 0$ are deterministic constants.

[1] A random variable $0 \leq X \leq 1$ that is Beta-distributed with shape parameters α and β is denoted as $X \sim Beta(\alpha, \beta)$. The PDF of $Beta(\alpha, \beta)$ is $f(x; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$.

[2] The gamma function is defined as $\Gamma(s) = \int_0^\infty t^{s-1} e^{-t} dt$. The upper incomplete gamma function is defined as $\Gamma(s, x) = \int_x^\infty t^{s-1} e^{-t} dt$

Since $\xi = \min_i \xi_i$, we have

$$\begin{aligned}
&Pr(\overline{E_s'}|\overline{E}_s) \\
&= Pr(\xi < \alpha_1|\overline{E}_s) \\
&\geq Pr(\xi_i < \alpha_1|\overline{E}_s) \\
&\geq Pr\left(\xi_i < d_i + (1-\tau_1)\alpha_1, d_i < \tau_1\alpha_1|\,\overline{E}_s\right) \\
&= Pr\left(\xi_i < d_i + (1-\tau_1)\alpha_1\,|d_i < \tau_1\alpha_1, \overline{E}_s\right) \\
&\qquad \cdot Pr\left(d_i < \tau_1\alpha_1|\,\overline{E}_s\right) \\
&\geq Pr\left(\xi_i < d_i + (1-\tau_1)\alpha_1\,|d_i < \tau_1\alpha_1, \overline{E}_s\right) \qquad (11) \\
&\qquad \cdot Pr\left(d_i < \tau_1\alpha_1\right) \\
&\geq Pr\left(\frac{\xi_i c}{d_i} < c + c(1-\tau_1)\frac{\alpha_1}{d_i}\middle|\, d_i < \tau_1\alpha_1, \overline{E}_s\right) \\
&\qquad \cdot Pr\left(d_i < \tau_1\alpha_1\right) \\
&\geq Pr\left(\chi_c^2 < c + \frac{1-\tau_1}{\tau_1}c\right)(1 - A_1 e^{-A_2\tau_1\alpha_1}) \qquad (12) \\
&\geq \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right)\left(1 - A_1 e^{-A_2\tau_1\cdot\beta\frac{D}{S}}\right)
\end{aligned}$$

In the inequality (11), $Pr\left(d_i < \tau_1\alpha_1|\,\overline{E}_s\right) > Pr\left(d_i < \tau_1\alpha_1\right)$ because the smaller the slice is, the less possible it is corrupted, and vice versa. Also, the inequality (12) holds since $\frac{\xi_i c}{d_i} \sim \chi_c^2$ and $d_i/\alpha_1 > \tau_1$. And finally, the last inequality follows from Inequation (2), in which $x = -\frac{1}{4}\left(-\sqrt{c} + \sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2$.

Thus, when $\phi \neq 0$ the probability of false alarm is

$$\begin{aligned}
&Pr(E_s', \overline{E}_s) \\
&= Pr(E_s'|\overline{E}_s)P(\overline{E}_s) \\
&\leq Pr(E_s'|\overline{E}_s) \qquad (13) \\
&= 1 - Pr(\overline{E_s'}|\overline{E}_s) \\
&< 1 - \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right)\left(1 - A_1 e^{-A_2\cdot\tau_1\beta\frac{D}{S}}\right)
\end{aligned}$$

When $\phi = 0$, we have $S = 1$. Let $\tau_1 = \frac{1}{\beta}$ in Equation (4), we have

$$Pr(d_i \geq \tau_1\alpha_1) = Pr(d_i \geq D) \approx 0$$

Similarly, we can prove

$$\begin{aligned}
Pr(E_s', \overline{E}_s) &\leq Pr(E_s'|\overline{E}_s) \\
&= 1 - Pr(\overline{E_s'}|\overline{E}_s) \\
&< 1 - \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{(2\beta-1)c}\right)^2}\right) \\
&= e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{(2\beta-1)c}\right)^2}
\end{aligned}$$

□

B. PROOF OF LEMMA 3

PROOF. If there are only insertions or only deletions in the received packet, there is at least one clean slice in each iteration. Similar to Lemma 2, we only derive the miss probability for the first iteration as the derivation can be generalized, in a straightforward manner, for other iterations. In the first iteration, let ξ_i denote the estimator for the i-th "clean slice". Let ξ_i' denote the estimator for the i-th "dirty slice". Define $\xi = \min_i \xi_i$ and $\xi' = \min_i \xi_i'$.

Let $t = \arg\min_i \xi_i'$. Slice $\vec{x}_t$ is the dirty slice with minimal estimated Hamming distance. Let λ_t' denote the length of its mismatched part. Let $z_{j,k}$ denote the distance between the j-th insertion (or deletion) and the k-th phantom deletion (or insertion). As shown in Step 6 of Algorithm 4 and Step 10 of Algorithm 5, the phantom deletions (or insertions) are uniformly distributed. Thus,

$$Pr\left(z_{j,k} < \tau_2\epsilon\frac{d}{S}\right) = 2 \cdot \tau_2\epsilon\frac{d}{S} \cdot \frac{1}{l}$$

where $\tau_2 \geq 2$, $\epsilon > 1$ are constants. Since $\lambda_t' \geq \min_{j,k}\{z_{j,k}\}$, we have

$$\begin{aligned}
Pr\left(\lambda_t' < \tau_2\epsilon\frac{d}{S}\right) &\leq Pr\left(\min_{\substack{1\leq j\leq\phi \\ 1\leq k\leq S}}\{z_{j,k}\} < \tau_2\epsilon\frac{d}{S}\right) \\
&\leq \phi S \cdot Pr\left(z_{j,k} < \tau_2\epsilon\frac{d}{S}\right) \\
&= \phi S \cdot 2\tau_2\epsilon\frac{d}{Sl} \\
&= 2\tau_2\epsilon \cdot \frac{\phi d}{l}
\end{aligned}$$

When λ_t' is given, we have $\mathbf{E}[\xi'] \geq \frac{\lambda_t'}{2}$. Thus

$$\begin{aligned}
Pr\left(\xi' > \epsilon\frac{d}{S}\right) &\geq Pr\left(\xi' > \epsilon\frac{d}{S}, \lambda_t' > \tau_2\epsilon\frac{d}{S}\right) \\
&= Pr\left(\xi' > \epsilon\frac{d}{S}\middle|\,\lambda_t' > \tau_2\epsilon\frac{d}{S}\right) Pr\left(\lambda_t' > \tau_2\epsilon\frac{d}{S}\right) \\
&= Pr\left(\frac{c\xi'}{\lambda_t'/2} > \frac{2c\epsilon d}{S\lambda_t'}\middle|\,\lambda_t' > \tau_2\epsilon\frac{d}{S}\right) \\
&\quad \cdot Pr\left(\lambda_t' > \tau_2\epsilon\frac{d}{S}\right) \\
&\geq Pr\left(\chi_c^2 > \frac{2c}{\tau_2}\right)\left(1 - 2\tau_2\epsilon\cdot\frac{\phi d}{l}\right) \qquad (14) \\
&\geq \left(1 - e^{-\left(\frac{\tau_2-2}{2\tau_2}\right)^2 c}\right)\left(1 - 2\tau_2\epsilon\cdot\frac{\phi d}{l}\right)
\end{aligned}$$

The inequality (14) holds since $\frac{c\xi'}{\mathbf{E}[\xi']} \sim \chi_c^2$, $\mathbf{E}[\xi'] \geq \frac{\lambda_t'}{2}$ and $\frac{S\lambda_t'}{\epsilon d} > \tau_2$. The last inequality follows from Inequation (3), in which $x = -(\frac{\tau_2-2}{2\tau_2})^2 c$. Similar to Lemma 2, we can prove

$$\begin{aligned}
&Pr(\xi < \epsilon\frac{d}{S}) \\
&\geq \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right)\left(1 - f_1(d,S)e^{-f_2(d,S)\cdot\tau_1\epsilon\frac{d}{S}}\right)
\end{aligned}$$

where $f_1(d,S)$ and $f_2(d,S)$ are functions of d and S.

Furthermore, we have

$$\begin{aligned}
Pr(\overline{E_s}) &\geq Pr(\xi < \xi') \\
&\geq Pr\left(\xi < \epsilon\frac{d}{S}, \xi' > \epsilon\frac{d}{S}\right) \\
&= Pr\left(\xi < \epsilon\frac{d}{S}\middle|\,\xi' > \epsilon\frac{d}{S}\right) Pr\left(\xi' > \epsilon\frac{d}{S}\right) \\
&\geq Pr\left(\xi < \epsilon\frac{d}{S}\right) Pr\left(\xi' > \epsilon\frac{d}{S}\right)
\end{aligned}$$

Thus

$$
\begin{aligned}
&Pr(\overline{E_s'}, E_s) \\
&= Pr(E_s'|E_s)Pr(E_s) \\
&\leq Pr(E_s) \\
&= 1 - Pr(\overline{E_s}) \\
&\leq 1 - Pr\left(\xi < \epsilon\frac{d}{S}\right) Pr\left(\xi' > \epsilon\frac{d}{S}\right) \\
&< 1 - \left(1 - e^{-\frac{1}{4}\left(-\sqrt{c}+\sqrt{\frac{(2-\tau_1)c}{\tau_1}}\right)^2}\right)\left(1 - e^{-\left(\frac{\tau_2-2}{2\tau_2}\right)^2 c}\right) \\
&\qquad \cdot \left(1 - f_1(d,S)e^{-f_2(d,S)\cdot\tau_1\epsilon\frac{d}{S}}\right)\left(1 - 2\tau_2\epsilon\cdot\frac{\phi d}{l}\right)
\end{aligned}
$$

□

C. PROOF OF LEMMA 4

PROOF. (Notations used in this proof is identical with that used in the proof of Lemma 3.) If there are both insertions and deletions in the packet, when $S < \phi + 1$ it is possible that there is no clean slice in some iterations. If this happens, we have

$$Pr(E_s) = 1$$

and

$$
\begin{aligned}
Pr(\overline{E_s'}, E_s) &= Pr(\overline{E_s'}|E_s)P(E_s) \\
&= Pr(\overline{E_s'}|E_s) \\
&= Pr(\xi' < \alpha_1) \\
&= 1 - Pr(\xi' > \alpha_1) \\
&< \delta_1
\end{aligned} \tag{15}
$$

The proof of the above upper bound for $Pr(\xi' > \alpha_1)$ is quite similar to the proof of $Pr\left(\xi' > \epsilon\frac{d}{S}\right)$ in Lemma 3. The only difference is the lower bound of $\mathbf{E}[\xi']$. Specifically, we want to prove

$$Pr(\mathbf{E}[\xi'] > \frac{1}{2}\tau_2\alpha_1) > 1 - 2\tau_2\beta\cdot\frac{\phi D}{l} \tag{16}$$

Note that $\lambda_t' \geq \min_{j,k}\{z_{j,k}\}$ if there are only insertions or only deletions in the selected slice. Otherwise, there are at least one insertion W_{ins} and one deletion W_{del} in the slice. Without of loss of generality, assume W_{del} is located on the right of W_{ins}, and there's no other deletions between W_{ins} and the left end of the slice. Let $\vec{x}_{t,1}$ denote the part of slice between W_{ins} and W_{del}. Let $\vec{x}_{t,2}$ denote the part between W_{ins} and the left end of the slice. Then, either $\vec{x}_{t,1}$ is mismatched or $\vec{x}_{t,2}$ is mismatched. Since the length of $\vec{x}_{t,1}$ is larger than Δ and the length of $\vec{x}_{t,2}$ is larger than $\min_{j,k}\{z_{j,k}\}$, we have $\lambda_t' \geq \min\{\min_{j,k}\{z_{j,k}\}, \Delta\}$. Hence,

$$
\begin{aligned}
\mathbf{E}[\xi_i] &\geq \frac{1}{2}\lambda_t' \\
&\geq \frac{1}{2}\min\left\{\min_{j,k} z_{j,k}, \Delta\right\}
\end{aligned} \tag{17}
$$

Furthermore, we have

$$
\begin{aligned}
&Pr\left(\min\left\{\min_{j,k} z_{j,k}, \Delta\right\} < \tau_2\alpha_1\right) \\
&= Pr\left(\min_{j,k} z_{j,k} < \tau_2\alpha_1\right) \\
&< 2\tau_2\beta\cdot\frac{\phi D}{l}
\end{aligned} \tag{18}
$$

Thus, Inequation (16) can be obtained by combining Inequation (17) and (18).

When $S \geq \phi+1$, there must exist at least one clean slice. Similar to that in Lemma 3, we can prove

$$Pr(\overline{E_s'}, E_s) \leq Pr(E_s) < \delta_2 \tag{19}$$

Thus the lemma follows by combining Equation (15) and Equation (19). □