A Simpler and Better Design of Error Estimating Coding

Nan Hua
School of CS
Georgia Tech
nanhua@gatech.edu

Ashwin Lall
Department of Math and CS
Denison University
lalla@denison.edu

Baochun Li
Department of ECE
University of Toronto
bli@eecg.toronto.edu

Jun Xu
School of CS
Georgia Tech
jx@cc.gatech.edu

***Abstract*—We study error estimating codes with the goal of establishing better bounds for the theoretical and empirical overhead of such schemes. We explore the idea of using sketch data structures for this problem, and show that the tug-of-war sketch gives an asymptotically optimal solution. The optimality of our algorithms are proved using communication complexity lower bound techniques. We then propose a novel enhancement of the tug-of-war sketch that greatly reduces the communication overhead for realistic error rates. Our theoretical analysis and assertions are supported by extensive experimental evaluation.**

I. Introduction

The seminal paper of Chen et al. [2] has brought to the fore the problem of (approximately) estimating the number of bit errors (correspondingly the bit error rate) that has occurred to a packet during its transmission over a wireless network. It has been shown in [2] that knowing this (approximate) bit error rate (BER) of a packet makes possible a host of advanced packet processing capabilities such as packet re-scheduling, routing, and carrier-selection schemes that can improve the (good) throughput of a wireless network in various ways. A simple yet effective technique, called error estimating codes (EEC), is proposed in [2] to help estimate this BER. Its basic idea 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. This solution has some flavor of low density parity check (LDPC), though its objective and hence parameter settings (e.g., the distribution of "densities") are very different.

In this work, we look at this problem from a very different angle. While the EEC work looks at this problem from by and large a coding theoretic perspective, our work looks at it from a theoretical computer science perspective, modeling it as a so-called two-party computation problem as follows. Two parties Alice and Bob each knows a (local) binary string x and y respectively, but Alice has no knowledge of y and vice versa. Alice and Bob are faced with the problem of computing the value of a function f acting upon the inputs x and y, often approximately. Intuitively, given any "sufficiently nontrivial" function f, for Alice and Bob to compute $f(x, y)$ together even approximately, either Alice has to tell Bob something about x or Bob needs to tell Alice something about y. The theory of two-party computation is concerned with how to evaluate $f(x, y)$ using as little communication (telling the other party about their local strings) between Alice and Bob as possible. Such a minimum amount of communication needed for the two-party computation of $f(x, y)$ is referred to as its *communication complexity*.

In the context of this work, two parties Alice and Bob are the transmitter and the receiver respectively. Alice knows the string (packet) x that is transmitted and Bob knows the string (packet) y that is received. The function f we would like to evaluate on (x, y) is clearly the Hamming distance between x and y, that is, $||x-y||_0$ (L_0 norm of the difference). We would like to find out the minimum amount of extra information (about x) that Alice needs to send to Bob, alongside with x, in order for Bob to approximately estimate $f(x, y)$. Techniques for (most) compactly encoding such extra information (about x) are referred to as *sketching algorithms* and the resulting encodings are called *sketches*.

Casting this BER estimation problem into the rich theoretical framework of two-party computation allows us to look much deeper into its underlying mathematical structures and obtain a set of new and better results. Our first result is to prove that the (randomized) communication complexity for the two-party computation of $||x-y||_0$ is $\Omega(\log n)$, where n is the length of the string x and y. In other words, Alice (the sender) needs to send to Bob (the receiver) a minimum of $\Omega(\log n)$ bits in order for Bob to approximately compute $||x-y||_0$. Since the number of overhead bits used in the EEC algorithm is indeed $O(\log n)$ [2], it matches this lower bound and is therefore asymptotically optimal. However, it is not optimal in terms of the constant factor, as we will explain shortly. Our second result is a new sketching technique that is based on an existing one with significant additional innovations. Our sketch also matches the $\Omega(\log n)$ lower bound, and has a smaller constant factor than the EEC scheme. In fact, our scheme requires a sketch size that is only 30% of the overhead bits used by the EEC scheme to achieve the same or better BER estimation accuracies. Our sketching technique also allows for a much simplified analysis of error guarantees than that in [2].

Here we briefly describe the basic ideas behind our sketch and innovations. Since x and y are binary strings in our context, the Hamming distance $||x-y||_0$ (the number of errors

978-1-4673-0775-8/12/$31.00 ©2012 IEEE

occurring during transmission) is the same as $||x-y||_1$ (L_1 norm) and $||x-y||_2$ (L_2 norm)[1]. Various sketches have been proposed to compactly encode a (long) string x for the two-party computations of $||x-y||_0$, $||x-y||_1$, and $||x-y||_2$. Among them, we discover that the tug-of-war sketch [1] proposed for estimating the L_2 norms is most suitable for our purposes.

However, the tug-of-war sketch *per se* is not yet the right solution to our problem for three reasons:

- EEC is a symmetric scheme in the sense that every bit is equally important or vulnerable: Flipping either an information bit or the parity bit in a parity equation results in this parity equation being violated. When the tug-of-war sketch is used for estimating $||x-y||_2$ (mathematically equivalent to $||x-y||_0$ as explained above) however, these sketch bits are much more important and vulnerable than the information bits in the sense that flipping a sketch bit causes much more "damage" to the BER estimation than flipping an information bit. Therefore, the sketch needs some kind of "extra protection." This is achieved using error detection mechanisms in our scheme.
- The fact that x is a binary vector (rather than a vector of large integers or floating numbers) that can be scrambled to have approximately equal number of 0's and 1's in it provides opportunities for the tug-of-war sketch to be further significantly compressed. In fact, through a statistical truncation technique, described later in Sec. V, we are able to reduce the size of each "sketch word" from $9 \backsim 14$ bits to 5 bits for typical application scenarios. This size reduction carries the additional benefit that only a much smaller sketch needs to be protected through error detection codes.
- The tug-of-war sketch is not as space-efficient as sampling in the high BER region. Our scheme explores the optimal way of combining them to achieve the best size-accuracy tradeoffs possible in that case.

The remainder of this paper is organized as follows. We introduce notations and some background information in Sec. II. Sec. III analyzes the lower bounds of error estimating codes, in terms of the number of overhead bits needed. Sec. IV describes the tug-of-war sketch, and gives a simple analysis to show that it accurately computes BER if the sketch is not corrupted by errors. In Sec. V, we propose our enhanced tug-of-war sketch that removes the assumption of integrity of the sketch and substantially improves its performance. We experimentally validate the efficacy of our scheme in Sec. VI. We discuss the related work in Sec. VII, and conclude in Sec. VIII.

II. Preliminaries

We start by introducing some notations needed to formally define the problem. To make the presentation and analysis of our algorithms much simpler, a 0 bit in a packet is represented by a '-1' signal (like in the Code Division Multiple Access literature), and a 1 bit by a '1' signal. We denote the packet sent out as $b \in \{-1,1\}^n$ (0 represented as '-1') and the packet received with errors as $b' \in \{-1,1\}^n$. Like in the original EEC scheme [], we assume that some of the bits of the packet may have been flipped, but the *length* of the packet remains the same. The goal of this paper is to compute with as little overhead as possible the number of errors in transmission, which is exactly the Hamming distance between b and b', and which we denote as d. We can then use this number to compute the bit error rate (BER) $p = d/n$.

We now briefly introduce the notions of *randomized approximation* and *communication complexity*, which we will later use in this paper.

Randomized approximation: The original EEC scheme, and the tug-of-war sketch presented in Section IV-A of this paper, are (ϵ, δ)-approximation schemes. An (ϵ, δ)-approximation algorithm is one that produces an estimate $\hat{X}$ for some quantity X with the guarantee that the absolute relative error $|\hat{X} - X|/X$ is at most ϵ with probability at least $1-\delta$. It is assumed that $0 < \epsilon, \delta < 1$ are arbitrary constants that can be tuned by the designer of the algorithm. Typically, the cost of the scheme is dependent on these two parameters.

Communication complexity: Many of our lower bounds on this paper make direct use of results from the communication complexity literature [9]. As mentioned before, communication complexity deals with the problem of determining the exact amount of communication needed between two parties to compute some function on their non-overlapping but jointly complete input. The communication complexity of a function at input size n is the largest number of bits that the two parties have to communication with each other using the *optimal* protocol for any input of size n. The basic communication complexity model can be generalized in many ways, two of which—randomization and one-round—appear in this paper. The randomized communication complexity of a function is the communication complexity of an optimal randomized protocol that is correct with some positive constant probability (over random choices of the protocol). The one-round communication complexity of a function is the communication complexity of an optimal protocol in which Alice sends a single message to Bob, and Bob then computes the result of the function with no further communication. The problem of estimating BER clearly corresponds to the one-round model. Randomization is also allowed in our context and is actually used by both EEC and our scheme.

III. Lower Bounds

We next show lower bounds for the BER estimation problem, demonstrating the optimality of our algorithms. In Sec. III.A, we show that deterministically estimating the error rate (i.e., without the use of randomization) requires the coding scheme to use $\Omega(n)$ bits of overhead even when we allow the estimate to err by over 10% from the actual value. Similarly, we show in Sec. III.B that randomization alone cannot produce the exact BER estimation using a well-known result from the

[1] Let x_i and y_i be the i^{th} bit in x and y respectively, $i = 1, 2, ..., n$. Then the L_p norm of the difference vector $x - y$ is defined as $\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$.

area of communication complexity. Based on these two results, one can see why we can only approximate the result with high probability. Finally, we show in Sec. III.C why the $O(\log n)$-bit sketch our algorithms use is necessary.

A. Why Randomization Is Needed

Theorem 1: Any error-estimating scheme that estimates the number of the bits in an n-bit packet that change during transmission to within $n/8$ must use $\Omega(n)$ overhead bits.

Proof: Let n be divisible by 8 (the argument works for all n with some slight modifications). It is known that there exists a family G of $2^{\Omega(n)}$ subsets of $\{1, 2, 3, \ldots, n\}$ such that (i) each set in G has cardinality exactly $n/4$, and (ii) every pair of sets in G have at most $n/8$ elements in common. The existence of such a family can be shown using the probabilistic method, but this is omitted here for brevity.

Let us assume for a contradiction that there exists a deterministic sketch of size less than $\Omega(n)$ bits that allows the computation of the Hamming distance between the original and transmitted codewords within an error of less than $n/8$. Consider what happens when we sketch all the codewords formed by the characteristic vectors of the sets in G. Since the sketch size is less than $\log(|G|) = \Omega(n)$, by the pigeonhole principle we know that two of the sets, say g_1 and g_2, in G must result in the same sketch value, making them indistinguishable. The Hamming distance between these two sets is at least $n/8 + n/8 = n/4$. As a result, since the sketch cannot distinguish between the cases when the original codeword and the transmitted codeword correspond to g_1 and g_1, versus when they correspond to g_1 and g_2, respectively, one of these two cases must have an error of at least $n/8$. ■

B. Why Approximation is Needed

Theorem 2: Any error-estimating scheme that computes the exact number of bits in an n-bit packet that change during transmission with probability at least $3/4$ must use $\Omega(n)$ overhead bits.

Proof: For this result, we use the communication complexity of the Set Disjointness problem. It is known that for two parties to compute whether their subsets of $\{1, 2, 3, \ldots, n\}$ have any elements in common requires $\Omega(n)$ communication, even when randomization (with $1/4$ failure probability) is allowed [7].

Assume for a contradiction that there is a randomized sketch using less than $\Omega(n)$ bits that can be used to compute the Hamming distance between the original and transmitted codewords exactly. We use this to create the following protocol for Set Disjointness. Alice uses the sketch to summarize the characteristic vector of her set and sends the sketch (less than $\Omega(n)$ bits) and the number of elements in her set ($\log n$ bits), call it n_a, to Bob. Bob can now use this information to compute the Hamming distance (call it h), the number of elements in his set (call it n_b), and then compute the size of the intersection of his and Alice's set as $(n_a + n_b - h)/2$. This is a one-round randomized protocol to compute the size of the intersection of Alice and Bob's sets, and hence must use $\Omega(n)$ communication, contradicting our assumption about the size of the sketch. ■

C. Randomized Approximation

We now use a lower bound in [8] to show that the asymptotic complexity of the tug-of-war sketch and the original EEC scheme are optimal in terms of their dependence on n and ϵ, the relative error bound. The lower bound result we use is as follows:

Theorem 3 ([8], [12]): The randomized one-round two-party communication complexity of approximating the Hamming distance of the n-bit vectors of two parties up to a relative error of ϵ with constant probability is at least $\Omega(\log(n)/\epsilon^2)$.

The reduction is the same as the last, and a lower bound of $\Omega(\log(n)/\epsilon^2)$ on the sketch size follows.

IV. Tug-of-War Sketch for Error Estimating Coding

A. The sketch

In this section, we briefly describe and analyze the plain vanilla tug-of-war sketch [1] in the context of error estimating coding, under the assumption that the sketch per se is not subject to bit errors during transmission. The tug-of-war sketch of a bit array (packet) b is comprised of a constant number c of counters (c is determined by the desired error guarantees) that are maintained using the same update algorithm (with possibly different update values) and is sent to the receiver alongside with b. After the execution of these update algorithms, each counter contains the inner product of the bit array[2] $\overrightarrow{b}$ with a pre-defined pseudorandom vector $\overrightarrow{s} \in \{+1, -1\}^n$. Note the actual update algorithm is not shown here because it is not relevant to our context; Only its "net effect" after execution is.

As shown in the following algorithm, upon the receipt of the transmitted bit array $\overrightarrow{b'}$ and the sketch (assumed to have no bit error during transmission), the receiver computes the c inner products using the received packet (possibly with bit errors) $\overrightarrow{b'}$, takes the difference between them and the counters in the sketch sent along the packet, and squares the result. Each of these results is now an unbiased estimate (proved in [1]) of the Hamming distance between the original and the transmitted packets, and can be averaged to give an accurate estimate of the Hamming distance[3], d. The details are given in Algorithm 1.

B. Analysis

We now show that this estimator (the average of component random variables X_1, X_2, $\ldots$, X_c) has low variance. To

[2] Here $\overrightarrow{b}$ is the vector representation of the packet b, where '0's have been converted to '-1's as discussed earlier.

[3] Note that this is a simplified form of the tug-of-war sketch proposed in [1]. The original version reduced the dependence on δ to $\log(1/\delta)$ by computing the average of $O(\frac{1}{\epsilon^2})$ estimators and then finding the median of $O(\log(1/\delta))$ such groups, at the cost of a larger constant multiplicative factor. For simplicity of the analysis in the following section, we omit this asymptotic improvement here.

Algorithm 1 The tug-of-war sketch for EEC.

```
SKETCH–CREATION($\vec{b}$)
  Input $\vec{b}$: original data bits vector.
  Output $z$: the sketch encoding $\vec{b}$.
  pre-compute random vectors $\vec{s}_{j,1\leq j\leq c} : [n] \rightarrow \{-1,1\}$
  for $j = 1$ to $c$ do
    $z_j := (\vec{b} \cdot \vec{s}_j)/2$
  return $z = \langle z_1, \ldots, z_c \rangle$

DISTANCE–ESTIMATION($\vec{b}', z$)
  Input $\vec{b}'$: received data bits vector, $z$: received sketch.
  Output $\hat{p}$: the estimated error rate.
  pre-compute random vectors $\vec{s}_{j,1\leq j\leq c} : [n] \rightarrow \{-1,1\}$
  for $j = 1$ to $c$ do
    $X_j := (z_j - \vec{b}' \cdot \vec{s}_j/2)^2$
  return $\hat{p}$=$\frac{1}{n}$average$(X_1, \ldots, X_c)$
```

compute the variance of each component X_j, we first compute

$$\begin{aligned}
\mathbf{E}[X_j^2] &= \mathbf{E}\left[(\sum_{b_i\neq b'_i} b_i s_j[i])^4\right] \\
&= \mathbf{E}\left[\sum_{b_i\neq b'_i} (b_i s_j[i])^4\right] \\
&\quad +\mathbf{E}\left[6 \sum_{b_i\neq b'_i \wedge b_k\neq b'_k \wedge i\neq k} (b_i s_j[i])^2 (b_k s_j[k])^2\right] \\
&= d + 6d(d-1)/2 \\
&= 3d^2 - 2d, \qquad (1)
\end{aligned}$$

Substituting this into the expression for the variance of X_j, we obtain

$$\begin{aligned}
\mathbf{Var}[X_j] &= \mathbf{E}[X_j^2] - (\mathbf{E}[X_j])^2 \\
&= 3d^2 - 2d - d^2 \\
&\leq 2d^2.
\end{aligned}$$

Although the variance of a single component X_j may look large (giving a standard deviation larger than d itself), averaging c of them reduces it by a factor of c. Using Chebyschev's inequality then allows us to bound the failure probability arbitrarily small as well (depending solely on how large we allow c to get). More concretely, if we pick $c = \frac{2}{\epsilon^2\delta}$, then by Chebyschev's inequality we get that the estimate $\hat{d}$ from the above algorithm has the guarantee

$$\Pr[|\hat{d} - d| \geq \epsilon d] \leq \frac{\mathbf{Var}[\hat{d}]}{\epsilon^2 d^2} \leq \frac{2d^2}{c\epsilon^2 d^2} = \delta.$$

Correspondingly, the relative error of the final estimate of $\hat{p} = \frac{1}{n}\hat{d}$ would also satisfy the $\epsilon - \delta$ bound:

$$\Pr[|\hat{p} - p| \geq \epsilon p] = \Pr[|\hat{d} - d| \geq \epsilon d] \leq \delta.$$

The total overhead of this scheme is that of sending $c = \frac{2}{\epsilon^2\delta}$ (a constant, independent of n) counters, each of which contains a number in the range $[-n, n]$. Hence, the asymptotic cost is $O(\log n)$ bits.

C. Cost and Overhead for EEC applications

In this section, we perform a rough estimate of the total size of the sketch if the plain vanilla tug-of-war sketch is used directly for EEC applications. This is needed for us to compare it with our enhanced sketch, to be described in the next section. To allow for a fair comparison, here we no longer assume the sketch is immune from bit errors during transmissions. The total size of the sketch is determined by three factors: the number of counters c, the size of each counter (denoted as k), and the number of extra bits needed to protect the sketch. Since our enhanced sketch uses the same number of counters, we only need to discuss the second and third factors here for comparison purposes.

To estimate the size of each counter, let us assume that the (maximum) length of the packet is 1500 bytes $= 1.2\times 10^4$ bits. In the worst case $\log_2(n) = \log_2(12000) \approx 14$ is needed per counter, since $\max(\vec{b} \cdot \vec{s}/2) = n/2$ and $\min(\vec{b} \cdot \vec{s}/2) = -n/2$. However, the value of each counter in the sketch, which is a random variable, has its probability densities concentrated around its mean 0, since $\vec{b} \cdot \vec{s}$ is the sum of n i.i.d. Bernoulli random variables $b_i s_i$, each of which takes value $+1$ or -1 with equal probability 0.5. We calculate the tail probability of the resulting Binomial distribution and find $\Pr(|Z| > 255) \approx 3 \times 10^{-6}$. Therefore, if we truncate each counter to 9 bits (including one sign bit, since z could be positive or negative) from 14, we risk overflowing it with probability 3×10^{-6}.

A lower bound of the number of additional bits needed to protect the sketch can be estimated using information theory as follows. Suppose the bit errors are symmetric (equal probability in flipping 1 to 0 and the other way around) and random, the amount of information brought by each bit received is:

$$I(p) = 1 + p \lg p + (1-p) \lg(1-p).$$

Therefore, the final size of the sketch, including all the protection bits, needs to be at least $I(p)^{-1}$ times larger than the original sketch. For example, when the error rate is 0.15, the blowup factor $I(0.15)^{-1}$ is equal to 2.56.

V. ENHANCED TUG-OF-WAR SKETCH: SCHEME AND ANALYSIS

As mentioned before, we enhance the vanilla tug-of-war sketch in the following three ways to achieve better space-accuracy tradeoffs and to be able to handle bit errors that may occur to the sketch. As shown in Algorithm 2, our enhanced sketch contains a number of important improvements.

First, in some BER estimation scenarios, we need only know whether the BER falls into a certain interval like $[2^{-i}, 2^{-i+1}]$, as suggested in [2], rather than the exact BER value. In some others, only a rough BER estimation is called for. The vanilla tug-of-war sketch can be an overkill for both types of scenarios at the cost of an unnecessarily large sketch size. Adding to the problem is the fact that a larger

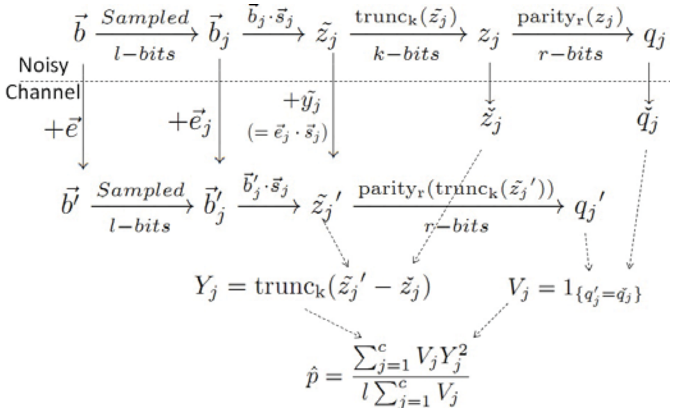
$\vec{b}$
Sampled
$l-bits$
$\vec{b}_j$
$\vec{b}_j \cdot \vec{s}_j$
$\tilde{z}_j$
$\mathrm{trunc_k}(\tilde{z}_j)$
$k-bits$
z_j
$\mathrm{parity_r}(z_j)$
$r-bits$
q_j
Noisy
Channel
$+\vec{e}$
$+\vec{e}_j$
$+\tilde{y}_j$
$(= \vec{e}_j \cdot \vec{s}_j)$
$\check{z}_j$
$\check{q}_j$
$\vec{b}'$
Sampled
$l-bits$
$\vec{b}'_j$
$\vec{b}'_j \cdot \vec{s}_j$
$\tilde{z}_j{}'$
$\mathrm{parity_r}(\mathrm{trunc_k}(\tilde{z}_j{}'))$
$r-bits$
$q_j{}'$
$Y_j = \mathrm{trunc_k}(\tilde{z}_j{}' - \check{z}_j)$
$V_j = 1_{\{q'_j = \check{q}_j\}}$
$\hat{p} = \frac{\sum_{j=1}^{c} V_j Y_j^2}{l \sum_{j=1}^{c} V_j}$

Fig. 1. Overview of Enhanced Tug-of-war Sketch (Algorithm 2)

sketch is more susceptible to bit errors during transmission and requires stronger protections which we can ill afford. Our solution is to combine sampling with sketching, in which we randomly sample l bits of the packet and sketch only these l bits according to Algorithm 2. A smaller l value leads to a smaller counter size and hence a smaller sketch size, at the cost of lower BER estimation accuracy due to higher sampling error. By adjusting this parameter l, we can minimize the sketch size needed to achieve a desired level of accuracy. The analysis needed for tuning this parameter for best size-accuracy tradeoffs is presented later in Sec. V-A.

Second, as we mentioned before, since the counter value (a random variable) stays close to its mean 0 with high probability, we may use fewer (say k) bits to store it without causing an "overflow" most of time. We refer to this enhancement as "statistical truncation", or *truncation* in short, for the lack of a better word. Even when an overflow (at either the sender or the receiver side) does happen (albeit with a small probability), its impact on estimation is small because with high probability truncation happens at both sides, in which case their difference remain the same as when there are no truncations. The impact of statistical truncation on the estimation accuracy will be analyzed in Sec. V-B.

Finally, as previously mentioned, the sketch is not immune to bit errors during transmission and requires some protection. In our scheme, each counter (5 bits long) will be protected by a parity bit (an overhead ratio of 20%). Any counter that fails the parity check will be considered corrupted and will not be included in the estimation. The rationale for this choice will be explained in Sec. V-C.

The parameters of the enhanced tug-of-war sketch in the following analyses are as follows. We let c be the total number of counters, l the number of bit positions sampled, k the length of each counter, and r the length of the parity bits of each counter. For convenience, we denote the sketch by $\mathrm{Sketch}(c, l, k, r)$. The total transmission cost is $c(k + r)$.

A. Analysis of the effect of sampling

Here we provide a simple analysis of the "sampled tug-of-war sketch" and show the benefits of sampling. In this analysis, we assume that no truncation is used and that no bit error happens during the transmission of the sketch. The value of a full (un-truncated) counter (a random variable) is denoted as $\tilde{z}$. Note that we use $\tilde{X}_j := \frac{1}{l}(\tilde{z}_j - z_j)^2$ to estimate the error rate $p = d/n$.

Algorithm 2 Enhanced tug-of-war sketch with parameters (c, l, k, r).

```
SKETCH–CREATION($\vec{b}$)
  Input $\vec{b}$: original data bits vector.
  Output $z$: the sketch of $\vec{b}$.
  pre-compute $l$-bit-long vectors $\vec{s}_{j,1\le j\le c} : [l] \to \{-1, 1\}$
  for $j = 1$ to $c$ do
    $l$-bits-long vector $\vec{b}_j$: sampled with replacement from $\vec{b}$
    Random projection: $\tilde{z}_j := (\vec{b}_j \cdot \vec{s}_j)/2$
    $k$-bits-long truncated projection: $z_j := \mathrm{trunc_k}(\tilde{z}_j)$
    $r$-bits-long parities $q_j := \mathrm{parity_r}(z_j)$
  return $c(k+r)$-bits long sketch $z = \langle z_1, \ldots, z_c\rangle\langle q_1, \ldots, q_c\rangle$

DISTANCE–ESTIMATION($\vec{b}'$, $\check{z}$)
  Input $\vec{b}'$: received data bits vector, $\check{z}'$: received sketch.
  Output $\hat{p}$: estimate of the error rate $p$.
  pre-compute $l$-bit-long vectors $\vec{s}_{j,1\le j\le c} : [l] \to \{-1, 1\}$
  for $j = 1$ to $c$ do
    $l$-bits-long vector $\vec{b}'_j$: sampled with replacement from $\vec{b}'$
    (with the same hash seed pre-configured.)
    Random projection: $\tilde{z}_j{}' := (\vec{b}'_j \cdot \vec{s}_j)/2$
    Estimation $Y_j := \mathrm{trunc_k}(\tilde{z}_j{}' - \check{z}_j)$ ,$X_j = Y_j^2$
    Check parities $V_j := 1_{\{q_j = \mathrm{parity_r}(\mathrm{trunc_k}(\tilde{z}_j{}'))\}}$
  return $\hat{p} = \frac{\sum_{j=1}^{c} V_j X_j}{l \sum_{j=1}^{c} V_j}$ as the estimation of error rate $p$.
```

Note that the Hamming distance between the two sampled segments, denoted by D, is no longer a constant. From Section IV, we have $\mathbf{E}[X_j^2|D_j] = D_j$ and $\mathbf{Var}[X_j^2|D_j] = 3D_j^2 - 2D_j$. Since the l bits are sampled with replacement, D follows a binomial distribution, $\mathbf{E}[D] = lp$ and $\mathbf{Var}[D] = lp(1 - p)$, where p is the error rate.

The new estimator of $\hat{p} = \frac{1}{l}\tilde{X}_j$ is still unbiased:

$$\mathbf{E}[\hat{p}] = \frac{1}{l}\mathbf{E}[\tilde{X}_j] = \frac{1}{l}\mathbf{E}[\mathbf{E}[\tilde{X}_j|D]] = \frac{1}{l}\mathbf{E}[H] = p.$$

As for variance, we have

$$\begin{aligned}
\mathbf{Var}[\hat{p}] &= \frac{1}{l^2}\mathbf{Var}[\tilde{X}_j] \\
&= \frac{1}{l^2}(\mathbf{Var}[\mathbf{E}[\tilde{X}_j|D]] + \mathbf{E}[\mathbf{Var}[\tilde{X}_j|D]]) \\
&= \frac{1}{l^2}(\mathbf{Var}[D] + \mathbf{E}[2D^2 - 2D]) \\
&= \frac{1}{l^2}(lp(1-p) + 2lp(1-p) + 2l^2p^2 - 2lp) \\
&= p^2(2 + \frac{1}{pl} - \frac{3}{l}) < p^2(2 + \frac{1}{pl}). \qquad (2)
\end{aligned}$$

Compared with (1), the relative variance of the sampled tug-of-war sketch is bounded by 2 plus an additional term $\frac{1}{pl}$.

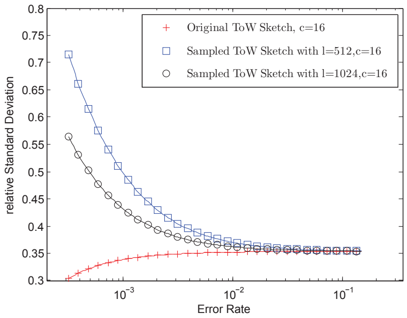

Fig. 2. Comparison of the variance of the sampled tug-of-war sketches and the original one, with $c = 16$.

This means only when p is $\Omega(\frac{1}{l})$ will the estimator achieve good performance. We can clearly observe this difference in Fig. 2. The sampled sketch with $l = 512$ performs very close to the original tug-of-war sketch when $p > 10^{-2}$, and much worse when $p < 10^{-3}$. From a practical perspective, this is exactly what we have intended. In typical application scenarios exemplified by [2], very accurate estimation (constant relative error) for very small bit error rates is not needed. In other words, the sampled tug-of-war sketch enables users to make the best use of bits for the applications we are interested in.

As discussed in Sec. III, the communication complexity bound for measuring the Hamming distance is $\Omega(\log n)$. Both the original tug-of-war sketch and the EEC sketch match this bound. The sampled tug-of-war sketch uses only $O(\log l)$ bits, where l is a constant parameter that can be much less than n. This does not contradict our lower bound, however, since it does not deliver the target estimation accuracy ((ϵ, δ)-approximation) for inputs with certain BER parameter settings.

B. Analysis of truncation and sampling together

In this section, we analyze the impact of truncation on the overall estimation accuracy. The operation of truncating a counter value to a k-bit number (including one bit needed to represent the sign) can be formalized as follows:

$$z = \mathrm{trunc_k}(\tilde{z}) \equiv \tilde{z} (\mathrm{mod}\ 2^k), z \in [-2^{k-1}, 2^k - 1]. \tag{3}$$

As shown in Algorithm 2 and Figure 1, we defined $\tilde{Z_j}' = \vec{b}'_j \cdot \vec{s}_j$ and $Y_j = \mathrm{trunc_k}(\tilde{Z_j}' - Z_j)$. We can find the following relationship between Y_j and the original (un-truncated) sketch values $\tilde{Z}_j$ and $\tilde{Z_j}'$ as follows:

$$\begin{array}{rcll} Y_j = & \mathrm{trunc_k}(\tilde{Z_j}' - Z_j) & , Y \in [-2^{k-1}, 2^k - 1] & \\ \equiv & \tilde{Z_j}' - Z_j & (\mathrm{mod}\ 2^k) & \\ \equiv & \tilde{Z_j}' - \tilde{Z}_j & (\mathrm{mod}\ 2^k), \text{ due to (3)} & \\ \equiv & \mathrm{trunc_k}(\tilde{Z_j}' - \tilde{Z}_j) & (\mathrm{mod}\ 2^k) & \end{array} \tag{4}$$

In the following, we derive the distribution, expectation and variance of each estimator $\hat{p}_j = \frac{1}{l}X_j = \frac{1}{l}Y_j^2 = \frac{1}{l}\mathrm{trunc_k}(\tilde{Z_j}' - \tilde{Z}_j)^2$.

Note Y_i, $i = 1, 2, ..., c$, are i.i.d. discrete random variables. Let Y be an arbitrary Y_i. In the following we will derive the probability mass function (PMF) of Y so that we can analyze the impact of truncation on our estimation accuracy. Define K as 2^{k-1}. Since Y takes values on exactly 2^k integer values $\{-K, -K+1, ..., K-1\}$, its PMF can be determined by a 2^k-dimensional vector $\vec{\gamma}(p, l) \equiv \langle \gamma_{-K}(p, l), \gamma_{-K+1}(p, l), ..., \gamma_K(p, l) \rangle$ where $\gamma_i(p, l) \equiv \Pr(Y = i | p, l)$, $i \in [-K, -K+1, ..., K-1]$. Note that each scalar γ_i is a function of the error rate p and the number of bits sampled l. We show that $\vec{\gamma}(p, l)$ can be computed from the following recurrence relation. We omit its proof in the interest of space.

Lemma 4:

$$\vec{\gamma}(p, l)_{1 \times 2^k} = \vec{\gamma}(p, l-1)_{1 \times 2^k} \mathbf{M}(p)_{2^k \times 2^k}, \tag{5}$$

where

$$\mathbf{M}(p) = \begin{bmatrix} \text{1-p} & \text{p/2} & & \cdots 0 \cdots & & & \text{p/2} \\ \text{p/2} & \text{1-p} & \text{p/2} & & \cdots 0 \cdots & & \\ & \text{p/2} & \text{1-p} & \text{p/2} & \cdots 0 \cdots & & \\ & \cdots & & & \cdots & & \\ & \cdots 0 \cdots & & & \text{p/2} & \text{1-p} & \text{p/2} \\ \text{p/2} & & \cdots 0 \cdots & & & \text{p/2} & \text{1-p} \end{bmatrix}_{2^k \times 2^k}$$

To allow for efficient matrix computations, $\mathbf{M}(p)$ can be diagonalized as follows.

$$\mathbf{M}(p) = \frac{1}{2^k} \mathbf{\Omega}' Diag(d_0, d_1, \ldots, d_{2K}) \mathbf{\Omega},$$

where $\Omega = \{\omega_{ik}\}$, $\omega_{ik} = \exp\left(\frac{2\pi ikj}{K}\right)$, j is the imaginary unit, and $d_i = 1 - 2\sin^2(i\pi/2^k)p$.

Considering that $\vec{\gamma}(p, 0) = [0, \cdots, 0, 1, 0, \cdots]_{1 \times 2^k}$, where 1 appears at the $(2^{k-1} + 1)^{th}$ position, we have

$$\begin{aligned} \vec{\gamma}(p, l) &= [0, \cdots, 1, 0, \cdots] \mathbf{M}(p)^l \\ &= \frac{1}{2^k} [0, \cdots, 1, 0, \cdots] \mathbf{\Omega}' Diag(d_0^l, d_1^l, \ldots, d_K^l) \mathbf{\Omega} \\ &= \frac{1}{2^k} [d_0^l, -d_1^l, \ldots, d_{2^k-2}^l, -d_{2^k-1}^l] \mathbf{\Omega}. \end{aligned} \tag{6}$$

Finally, we can calculate the expectation and the variance of Y^2 from $\vec{\gamma}(p, l)$ as follows:

$$\mathbf{E}[Y^2] = \vec{\gamma}(p, l) \boldsymbol{\beta}_2 \tag{7}$$

$$\mathbf{Var}[Y^2] = \vec{\gamma}(p, l) \boldsymbol{\beta}_4 - (\vec{\gamma}(p, l) \boldsymbol{\beta}_2)^2, \tag{8}$$

where $\boldsymbol{\beta}_i = [(-K)^i, (-K+1)^i, ..., (K-1)^i]$.

After summing up all c counters in the sketch, we finally arrive at the Mean Squared Error of the estimator,

$$\begin{aligned} l^2 \mathbf{MSE}[\hat{p}] &= \mathbf{E}[(\frac{1}{c} \sum Y_i^2 - pl)^2] \\ &= \frac{1}{c} \mathbf{Var}[Y^2] + (\mathbf{E}[Y^2] - pl)^2. \end{aligned} \tag{9}$$

Since $\frac{1}{l}\tilde{Y_j}^2$ is unbiased and $|\mathrm{trunc_k}(x)| \leq |x|$, $\frac{1}{l}Y_j^2$ will be (slightly) negatively biased. Note the aforementioned Chebyshev's inequality still holds for the mean squared error, while it does not for the variance when the estimator is biased.

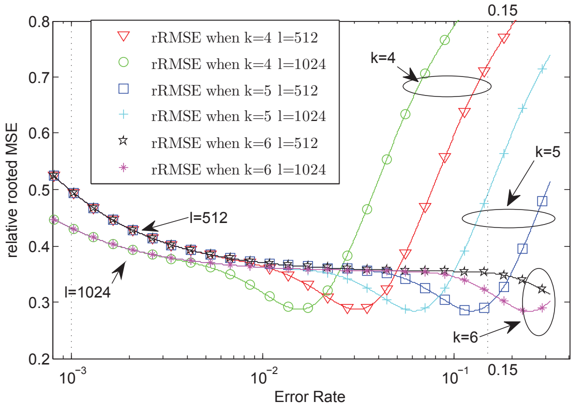

Fig. 3. The relative Rooted Mean Squared Error of the enhanced tug-of-war sketch (fully protected) with different sampling and truncation parameters, when $c = 16, l = \{512, 1024\}, k = \{4, 5, 6\}$.

Because of the bias, the aforementioned (ϵ, δ)-approximation guarantee no longer holds for the truncated version of the tug-of-war sketch.

In Fig. 3, we plot the relative Rooted Mean Squared Error parameterized by several combinations of k and l. It shows that the sampling parameter l shifts the left wing of the relative error curve, while the truncation parameter k shifts the right wing.

C. Impact of bit errors on counters and protection

In this section, we discuss the types of error detection mechanisms that are appropriate for protecting our enhanced sketch and derive the formula for analyzing their error probabilities. An error detection code can be defined by its generating matrix. For example matrix $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$ means there are two parity bits per counter. The first parity bit is XOR of all five bits and the second is the XOR of the three most significant bits. A counter is considered corrupted during transmission if it fails at least one of the parity checks. A corrupted counter thus detected will not be used in the BER estimation. However, all corrupted counters may not be detected. The following analysis will derive the distribution, expectation and variance of the sketch differences Y, denoted by $\boldsymbol{\gamma_q(p)}$, when the effects of both types of corruptions (detected and undetected) are factored. Based on this analysis, our scheme chooses to have one parity bit per counter, which is the XOR of all 5 bits in the counter (corresponding to the first row of the above matrix).

We first model the distributions of the errors that survive the parity checking (undetectable errors). The impact of those errors on the estimates Y_i can be calculated as follows:

$$\boldsymbol{\gamma_q(p)} \approx \gamma(p)\boldsymbol{Q}(p),$$

where $\boldsymbol{Q}(p)$ is determined by the design of the parity bit(s). We can then replace the γ_q in (7-9) with $\boldsymbol{\gamma_q(p)}$ to derive the expectation, variance and MSE of the final estimate Y_i^2.

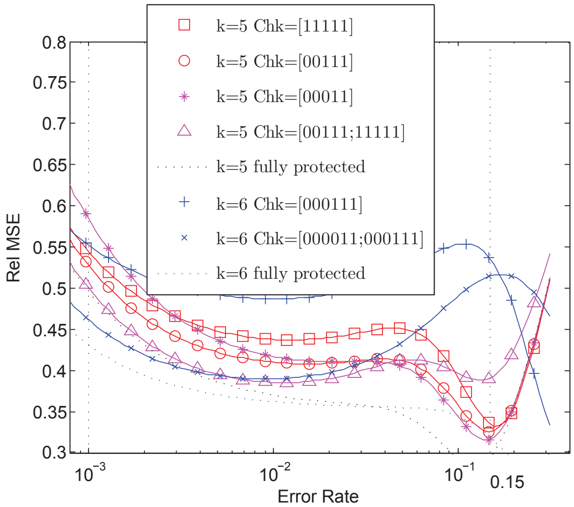

Fig. 4. Comparisons of different constructions of parity-checking bits. When $k = 5$, we use sampling length $l = 512$; when $k = 6$, we use $l = 1024$

The next step is to take into consideration the impact of detectable errors. Such errors will not affect the bias of the final $\hat{p}$, but will increase its MSE because fewer counters are included in the average. The MSE of $\hat{p}$ when considering such errors is as follows:

$$\begin{aligned} l^2\mathbf{MSE}[\hat{p}] &= \sum_{k=1}^{c} \frac{1}{k}\binom{c}{k}\theta^k(1-\theta)^{c-k}\mathbf{Var}[Y^2] \\ &+ (\mathbf{E}[Y^2] - pl)^2, \end{aligned} \qquad (10)$$

where θ is the probability with which each counter survives the parity checking. As discussed earlier, θ is a function of p and the generating matrix. The error distribution of the final estimator is also numerically computable, but we omit the details here in the interest of space.

We compare different constructions of generating matrices in Fig. 4 and the results can be summarized as follows. First, it is not necessary to parity-check all sketch bits. However, if too few sketch bits are parity-checked, the accuracies in estimating low BER can be impaired. This phenomena can be observed by comparing the first three curves in the legend. Second, two parity bits per counter instead of one does not considerably improve the accuracy of the final estimation. For this reason, we choose to have one parity bit per counter in our scheme.

VI. Evaluation

In this section, we evaluate the performance of our sketch experimentally and compare it with the original EEC scheme. For the EEC scheme, we use the parameters recommended in [2] for Wifi applications, i.e., with 9 levels and each level comprised of 32 bits. In total the EEC scheme costs 288 bits per packet and is targeted for estimating error rates in the $[10^{-3}, 0.15]$ region. The authors of [2] have proposed three different estimators for their scheme. A naive estimator for $\hat{p}$ (BER) is $q_i/2^i$ (defined in their paper); Two more sophisticated and accurate estimators are the roots of $\phi(2^i, p) = q_i$ and $\phi(2^i, p) = q_i/2 + q_{i-1}(1 - q_{i-1})$ respectively. We find that the latter two estimators both have better estimation accuracies than the naive one, and neither of them dominates the other. For convenience, we use $\hat{p}_1$ and $\hat{p}_2$ to denote the latter estimators, respectively.

Next, we discuss the parametrization of our sketch. As analyzed in Section V, the sampling parameter l and truncation parameter k can be tuned for different target error rate regions. Since we will show (in Figure 5) that the experimental results are nearly identical to the analytical results, we present only the analytical results with the optimal parameter settings: c=16 or 48, l=768, k=5, and one parity checking bit per counter generated by the matrix $[\ 0\ \ 0\ \ 1\ \ 1\ \ 1\ ]$. The sketch with 16 counters consumes only 96 bits per sketch, 33.3% of that consumed by the original EEC scheme; The sketch with 48 counters consumes 288 bits, the same as the original EEC scheme.

As for the performance metric used for comparisons, the relative deviation from the truth value (rRMSE=$\frac{1}{p}\sqrt{MSE[\hat{p}]}$) is an important indicator of the performance, but is not enough to represent all its characteristics. To complement that, we present also the percentage of trials with relative errors larger than 25%, 50% and 75% respectively. In addition, we directly compare the tail distribution of estimates on some representative BER values, such as 0.005 and 0.05.

The experimental results are shown in Figure 5. Each curve is generated from the results obtained from 8000 experiments. We can make two observations from these results. First, we observe that all experimental results are nearly identical to the analytical results. Second, we observe that the performance (i.e., estimation accuracy) of the enhanced tug-of-war sketch of size 96 bits is close to or even better than the original EEC scheme of size 288 bits in the target error rate region, while the enhanced tug-of-war sketch of size 288 bits performs way better than the original EEC scheme. To summarize, our scheme achieves similar BER estimation accuracies with a sketch size that is only 1/3 of that used by the original EEC scheme.

The computational overhead of our scheme is also very low. The enhanced scheme requires a combination of inner products as well as modulo and other arithmetical operations, all of which have cost $O(n)$. In practice, the inner product is equivalent to the bit counting operation. All of these operation took very little time in our experiments and therefore we do not present any computation time measurements here.

VII. Related Work

The study of error estimating codes was pioneered by Chen et al. in [2]. Although there has been much study of error correcting codes (e.g., LDPC [10]), this was the first paper to formalize the problem of estimating the bit error rate during transmission. Besides an extensive study of the many applications of EEC, ranging from the setting of wi-fi rates to routing, their paper gives the first (ϵ, δ)-approximation algorithm for the problem. We briefly describe the algorithm here for comparison purposes

The seminal paper of Chen et al. [2] proposed an error estimating code scheme that was able to estimate the BER to within small relative error with high probability. The key idea of the scheme proposed in [2] is to send several levels (groups) of additional parity bits with the data together and use those bits to infer the BER in the data. A code bit at different levels $i = 1, 2, ...$ is the parity of different number (l_i, $i = 1, 2, ...$) of sampled data bits. Hence the scheme can maintain a good "estimation resolution" on a wide range of different BERs. They showed that that their scheme with 9 levels and less than 300 additional bits in total per packet would be able to well differentiate BER rate in range $[10^{-3}, 0.15]$, and it would work well in real-world wireless experiments and is a great enhancement. They also show that they provide a (ϵ, δ) bound analysis of the proposed scheme, i.e. they could guarantee at most ϵ relative error that failed with probability less than δ, where ϵ and δ are arbitrarily tunable parameters that determine the overhead cost of their algorithm. In total they need about $O(\log(n))$ overhead for an n bit packet to achieve error estimating rates within the threshold desired by target applications.

One of the major technical contribution that we make in this paper is to adapt sketching algorithms from the field of data streaming to this problem. Data streaming is a well-studied area with a rich literature [11]. In the data streaming model, the input is provided as a long stream of updates in which only a single pass is allowed over the stream and the memory and time of the algorithm is heavily constrained (in particular much smaller than the size of the input). The connection to this problem is that there are many streaming algorithms that can be used to compute the *difference* (or distance) between two streams, and the summaries of these algorithms (called sketches) are what we can use as overhead bits for this problem. We tried several different sketching algorithms, including the count-min sketch [3], the Flajolet-Martin (FM) sketch [5], the stable distribution sketch [6], before settling on our variation on the tug-of-war sketch [1]. The tug-of-war sketch was originally suggested by Alon et al. [1] for estimating the second frequency moment of a data stream. The governing characteristic of this sketch, that we make use of in this paper, is that it is a random projection of the input, thereby allowing for deletions from the received packet. This sketch was modified for measuring the $L1$ distance of streams by Feigenbaum et al. [4].

Finally, our lower bounds make direct use of known results for the communication complexity [9] of the Hamming distance problem. Our main lower bound, showing that $\Omega(\log(n)/\epsilon^2)$ overhead is necessary is a consequence of the communication complexity result from [8].

VIII. Conclusions

In this paper we cast the recently proposed BER estimation problem as a two-party computation problem. From a theoretical standpoint, we proved that even when approximation and

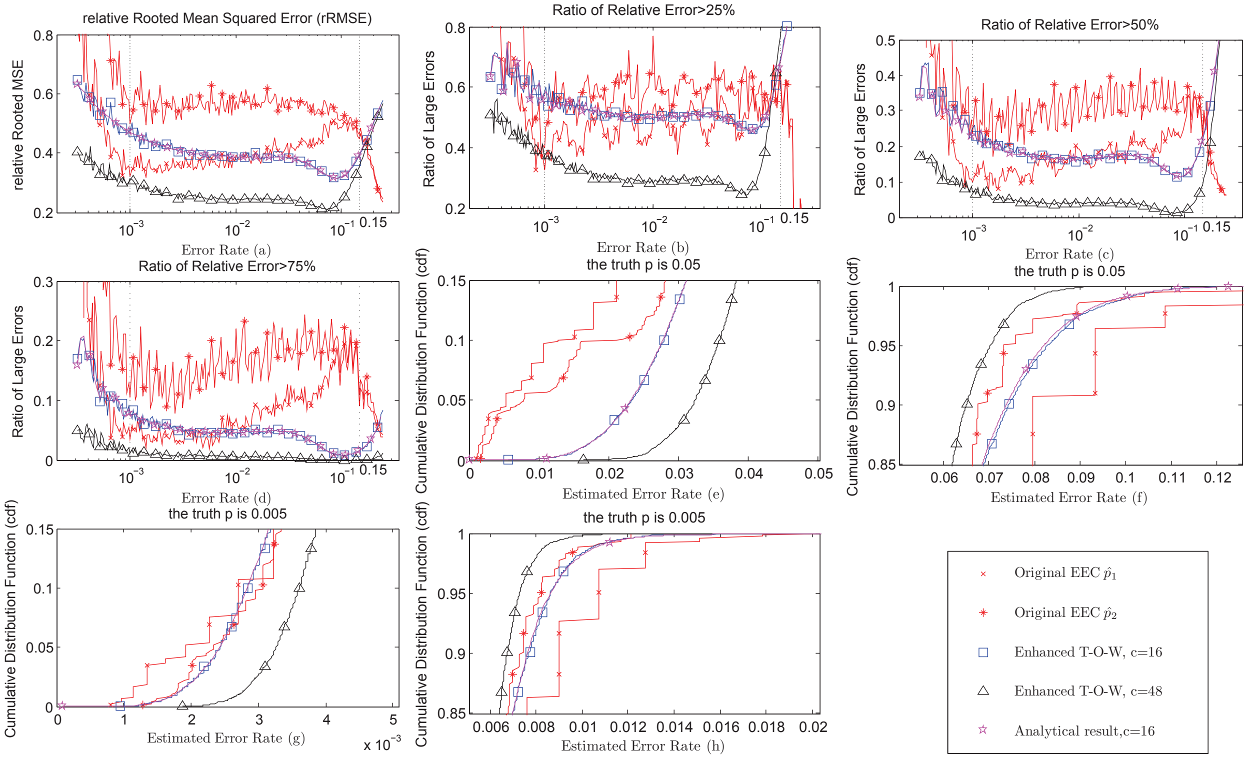

Fig. 5. Comparisons among the original EEC scheme and the enhanced tug-of-war sketches, measured by of relative rooted MSE, tail distributions, and ratio of large errors.

randomization are allowed the cost of this problem is $\Omega(\log n)$, where n is the length of data transmitted, which explains why both the EEC scheme and the tug-of-war sketch both need this much overhead. From a practical standpoint we presented an enhanced tug-of-war sketch with significant additional innovations for better fitting BER estimation applications. We found that our enhanced sketch uses only around 30% percent of the overhead bits used by EEC scheme to deliver comparable performance, and can deliver much better performance when the overhead is the same. The performance of the proposed sketch is fully analyzable and easily tunable. The efficacy of our sketch is demonstrated experimentally.

Acknowledgement: The work of Hua and Xu is supported in part by the US National Science Foundation through grants CNS 0905169 and CNS 0910592.

References

[1] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002.

[2] B. Chen, Z. Zhou, Y. Zhao, and H. Yu. Efficient error estimating coding: feasibility and applications. In *SIGCOMM*, pages 3–14, 2010.

[3] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[4] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate l1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002.

[5] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[6] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[7] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

[8] D. M. Kane, J. Nelson, and D. P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *SODA*, pages 1161–1178, 2010.

[9] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[10] S. Lin and D. J. C. Jr. *Error control coding - fundamentals and applications*. Prentice Hall computer applications in electrical engineering series. Prentice Hall, 1983.

[11] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[12] D. P. Woodruff. personal communication, 2011.