IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 July 2022, accepted 18 September 2022, date of publication 27 September 2022, date of current version 7 October 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3210336*

APPLIED RESEARCH

# Tabular Interpolation Approach Based on Stable Random Projection for Estimating Empirical Entropy of High-Speed Network Traffic

**YU-KUEN LAI[1], (Senior Member, IEEE), CHENG-LIN TSAI[1], CHENG-HAN CHUANG[1], XIU-WEN KU[1], AND JIM HAO CHEN[2]**

[1]Department of Electrical Engineering, Chung Yuan Christian University, Taoyuan 32023, Taiwan
[2]International Center for Advanced Internet Research, Northwestern University, Chicago IL 60611, USA

Corresponding author: Yu-Kuen Lai (ylai@cnsrl.cycu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 108-2221-E-033-015 and Contract MOST 109-2221-E-033-032-MY2.

**ABSTRACT** The empirical entropy of the network flow attributes is an essential measure for identifying anomalous network traffic. However, computing the exact entropy values for high-speed networks in real-time is computationally expensive. Accordingly, the present study replaces the complex computations of existing stable random projection methods for entropy estimation with a simple table lookup procedure. Notably, the size of the lookup table is reduced through a piece-wise linear interpolation heuristic in order to facilitate the implementation of the proposed scheme in resource-constrained pipeline environments. The proposed architecture enables entropy estimation to be performed using both the Log-Mean Estimator (LME) method and the New Estimator of Compressed Counting (NECC) algorithm reported in the literature. The feasibility of the proposed approach is verified empirically using both real-world network traffic traces and synthetic data streams. Moreover, the practical applicability is demonstrated via stream-based implementation in the programmable data planes of the NetFPGA-Plus framework and a Tofino P4 switch, respectively. The results indicate that the proposed tabulation-based entropy estimation scheme allows minimum-sized Ethernet frames to be processed with a wire speed of up to several hundred gigabits per second.

**INDEX TERMS** Empirical entropy, tabulation, stable random projection, programmable data plane, P4, FPGA, network traffic measurement, anomaly detection.

## I. INTRODUCTION

Information-theoretic methods provide effective means of detecting anomalies in network traffic. One such measure is the empirical Shannon entropy, defined as

$$H = -\sum_{i=1}^{n} \frac{m_i}{m} \log \frac{m_i}{m}, \tag{1}$$

where $m$ is the total elements in the data stream, $m_i$ is the frequency of item $i$ appears in the stream, and $n$ is the total number of distinct items in the stream. As shown in Equation (1), the entropy $H$ reaches its minimum value of zero when all the items in the stream are the same, and reaches its maximum value when all of the items are different. Compared to the volume-based traffic anomaly detection methods [1], entropy-based approaches provide a high-sensitive detection capability [2] and finer-grained insights into the network behavior [3] without requiring continuous volume-change monitoring.

Entropy-based methods thus provide a more feasible approach for detecting both low- and high-rate [4] Distributed Denial of Service (DDoS) attacks [5], [6] and for differentiating between Flash Events (FE) and DDoS attacks [7]. However, developing effective techniques [8] for measuring the entropy of high-speed network traffic in real-time still remains a significant challenge [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

### A. MOTIVATIONS

Since 2010, as the IEEE 100 Gbps network standards (IEEE802.3ba) [10] were introduced, wire speeds in data center networks have increased tremendously in order to meet the emerging needs of "Anything as a Service" (XaaS) technologies [11], [13] for low-latency network functions with high-bandwidth and enhanced flexibility. However, as the deployment of high-speed networks has increased, the scale, severity, and number of malicious attacks also increased. These attacks are capable of causing substantial economic losses and disruption [12]. Hence, effective methods for real-time detection of anomalous traffic in large-scale, high-speed networks are urgently required [6], [13]. However, computing the exact entropy values of high-speed and high-volume packet streams for networking applications is extremely challenging, particularly in environments with limited system resources. Moreover, it is generally necessary to compute the entropy of one or multiple combinations of traffic features simultaneously, which further increases the cost and complexity of the detection process.

As shown in Equation (1), a software-based implementation of the entropy computation is straightforward for offline, postmortem analyses of data anomalies in low-speed networks. However, given the high cardinality nature of the network traffic in high-speed networks, sampling techniques [14] are required to overcome the time and space complexity of real-time packet processing. Furthermore, the sampling processes may create data losses, which introduce distortion and measurement bias [15] into the anomaly detection process.

Frequency moment estimation [16] is an essential building block for many stream-based applications and has been widely used to simplify and accelerate the entropy estimation process for high-speed network traffic [17], [18], [19]. In the year 2000, Indyk proposed a unified framework [20] to estimate the $L_p$ norm of a packet stream $\Phi$ where $p \in \{1, 2\}$ based on an $\alpha$ stable distribution. This foundation was later used in [8], [21], [22], and [23] to perform entropy estimation. However, generating the random values required to compute the entropy from the stable distribution involves complex floating-point multiplication, division, logarithmic, and trigonometric operations, which impose a significant processing bottleneck on the estimation process.

Accordingly, motivated by the work of Cormode [24], this study presents a tabulation-based methodology for estimating the empirical Shannon entropy based on the stable random projection method [20] and a piece-wise linear interpolation technique. The overall goal of the proposed method is to obtain high-accuracy estimates of the network entropy with a rapid processing speed. Moreover, achieving low computational and memory cost is also essential. Thereby, supporting real-time anomaly detection in high-speed networks in even low-resource systems is feasible.

### B. CONTRIBUTIONS

This paper presents a tabulation-based method, designated as the *k-parallel* lookup with *m-hash*, for estimating the empirical Shannon entropy using the stable random projection framework proposed by Indyk [20]. The key component of the proposed method is the use of an inverse transform sampling technique to construct an empirical distribution function in a read-only lookup table. To facilitate the implementation of the proposed scheme in resource-constrained environments, the size of the lookup table is reduced through the use of a piece-wise linear interpolation heuristic based on three adaptive parameters (*Span*, *Exponential Head*, and *Exponential Tail*).

The proposed architecture can support both the Log-Mean Estimator (LME) method proposed by Clifford and Cosma [23] and the New Estimator of Compressed Counting (NECC) [25] proposed by Li [26]. The feasibility of the proposed method is verified using both real-world network traffic traces and synthetic data streams. Moreover, the practical applicability is demonstrated via stream-based implementation in the programmable data planes of an Xilinx U200 FPGA and Tofino P4 switch, respectively. The PoC design is capable of processing minimum-sized Ethernet frames at 100 Gbps wire-speed.

The remainder of this paper is organized as follows. Section II presents the background and related work on stream-based entropy estimation. Section III briefly outlines the problem considered in the present study. Section IV introduces the proposed tabulation-based entropy estimation method and interpolation technique, and briefly describes the system implementation. Section V discusses the key parameters of the proposed scheme and explores the corresponding design space. Section VI evaluates the performance of the proposed architecture using both real-world and synthetic traffic traces. Section VII presents the system implementation on the FPGA platform and P4 switch. Section VIII discusses the comparisons of evaluated results, implementation flexibility, and limitation. Finally, Section IX provides some brief concluding remarks and indicates the intended direction of future research.

## II. BACKGROUND AND RELATED WORKS

### A. STREAM COMPUTATION

Data stream computation [27] has been an active research topic ever since the Internet started to undergo exponential growth. Typically, a data stream $\Phi = (a_1, a_2, \ldots, a_m)$ consists of $m$ elements, where some are distinct and others are repeated. The elements arrive at the observation point sequentially at time $t$, where the $t_{th}$ element $a_t = (key_t, d_t)$ consists of a $key_t \in [n]$ and an update $d_t \in \mathbb{R}$. The space of set $[n]$ has a maximum value of $2^{104}$, if the packet stream measurement is based on the 5-tuple traffic flow consisting of the protocol number, source, destination of TCP/UDP ports and IPv4 addresses. Thus, researchers have proposed numerous algorithms for summarizing such massive data flows in a

one-pass fashion [28]. Such stream-based algorithms, commonly known as $(\varepsilon, \delta)$-approximation algorithm, are not only capable of estimating the measurement outcome accurately with an error of less than $\varepsilon$ with a high probability of $1 - \delta$, but are also highly suited to implementation in embedded networking systems with only limited memory and computing resources.

### B. STABLE DISTRIBUTION

Alpha-stable distributions, also known as Levy $\alpha$-stable distributions, are a family of probability distributions with the form $S(x : \alpha, \beta, \gamma, \mu)$, where $\alpha, \beta, \gamma$ and $\mu$ are stability, skewness, scale and location parameters with values of $\alpha \in (0, 2]$, $\beta \in [-1, 1]$, $\gamma > 0$ and $\mu \in \mathbb{R}$ [29]. Such distributions are used to model complex processes in a wide range of fields including income distribution analysis [30], commodity price estimation [31], and financial return prediction [32]. Moreover, statistical models based on $\alpha$-stable distributions have also been used for engineering asset health monitoring [33], network traffic anomaly detection [34], and wireless network performance evaluation [35] for base station deployment.

Chambers *et al.* [36] presented a method for simulating stable random variables based on the arbitrary values of the stability $\alpha \in (0, 2]$ and skewness $\beta \in [-1, 1]$ [37]. Weron [38] provided the proof of the equality constructing a standard stable random variable $X \sim S(x : \alpha, \beta, \gamma, \mu)$.

Indyk [20] proposed a framework for estimate the $L_p$ norm of a packet stream $\Phi$ based on the fact that the magnitude of the product of the key of each data item in stream $\Phi$ and the corresponding random variable R drawn from an $\alpha$-stable distribution is proportional to the $L_p$ norm of a packet stream $\Phi$, where $p \in \{1, 2\}$. This framework was later taken as the foundation for many proposals for stream-based entropy estimation [8], [21], [22], [23], [25]. (Note that for a more in-depth introduction to stable distributions and their related applications are available in the discussion paper of Borak *et al.* [39] and the study of Cormode and Indyk [40] on high-speed data stream processing.)

### C. STREAM-BASED ENTROPY ESTIMATION

As shown in Equation (1), the computation process for estimating the empirical Shannon entropy comprises two parts: (1) determining the frequency statistics of each arriving distinct item $m_i$, and (2) performing logarithmic and summation operations. Many entropy estimation algorithms have been proposed over the years. These algorithms can be broadly categorized into three main groups: (1) Alon-Matias-Szegedy (AMS) sampling [16], (2) Hash tables with sketch data structures, and (3) Random projection based on stable distributions. Table 1 presents a qualitative comparison of these stream-based entropy estimation schemes with that proposed in the present study.

#### 1) AMS SAMPLING

Lall *et al.* [41] presented a data streaming algorithm for estimating the entropy norm $m_i \log m_i$ of high-speed networks based on the second frequency moment estimation obtained via AMS sampling [16]. The authors also presented a sieving methodology for improving the estimation accuracy by separating the larger flows from the smaller flows. The experimental results obtained for a traffic trace consisting of 6 million distinct counts and 67 million packets showed that the proposed algorithm consumed approximately 1.4 Mbytes of memory space and enabled the entropy to be estimated with at most 25% relative error with a probability of 75%.

Chakrabarti *et al.* [18] proposed a similar AMS-based approach for estimating the entropy and entropy norm of data streams. However, while a comprehensive theoretical performance analysis was given, no implementation details were provided.

Bhuvanagiri and Ganguly [46] presented a Hierarchical Sampling over Sketches (HSS) methodology for estimating the entropy over data streams. In the proposed approach, $O(\log m)$ levels of data structures were created from the original data stream, and the Count-Min [47] and Count sketch [48] data structures were then used to estimate the top-k items and frequency of each item at each level. The total memory space requirement was shown to be $O((\frac{top-k}{\varepsilon} \log m + \log\frac{1}{\varepsilon})(\log^3 m)(\log\frac{1}{\delta}))$.

Chakrabarti *et al.* [49], [50] proposed a near-optimal stream-based algorithm for estimating the empirical entropy using the AMS algorithm [16], which requires $O(\varepsilon^{-2} \log\frac{1}{\delta} \log m)$ words of memory space. The implementation described in the paper utilized a reservoir sampling approach and maintained the associate counters in a dictionary structure. Moreover, two heap structures were employed for the the primary and backup samples, respectively. The per-item processing time in the stream was shown to be $O(\log\frac{1}{\varepsilon} + \log\log\frac{1}{\delta} + \log\log m)$.

Harvey *et al.* [51] approximated the Shannon entropy by estimating the Renyi entropy under the boundary condition of $\alpha \to 1$and sufficiently large values of $\varepsilon$. It was shown that, based on the AMS algorithm [16], the proposed scheme consumed just consumed $O(\varepsilon^{-4} \log^4 m)$ words of memory space. In a later study [19], the same group proposed another near-optimal algorithm for estimating the Tsallis entropy and Shannon entropy with $\alpha \to 1$ using just $O(\varepsilon^{-2} \log m)$ words of memory space.

Lapolli *et al.* [42] proposed a pipeline scheme based on AMS sampling algorithm [16] and the Count sketch [48] data structure for performing entropy estimation in the programmable data plane for DDoS attack detection. In the proposed method, the complex logarithmic computations required to estimate the Shannon entropy were replaced with a simple lookup operation on a Longest Prefix Match (LPM) table using Ternary Content-Addressable Memory (TCAM). The experimental results showed that the memory space required to monitor a single 1Gbps link consisting of $2^{18}$ packets in a 250-millisecond observation time was just 58.125 Kbytes. However, a total of 9 Mbytes [42] were required in a high-speed device to monitor 24 links of 10 Gbps.

**TABLE 1. Comparison of stream-based Entropy estimation schemes reported in literature and proposed scheme.**

| | Memory Size (Bytes) | Data Structure | Mean Relative Error | Epoch Time (sec) | Packet Count | Cardinality | Implementation Notes |
|---|---|---|---|---|---|---|---|
| Lall *et al.* [41] | 1.4 M | AMS Sampling | <3% | 60~300 | 2.5 M | 25 K | Intel CPU ($\varepsilon = 0.25, \delta = 0.25$) |
| Zhao *et al.* [8] | 160 M | Random Projection | N/A | 300, 3600 | 400 M | 1.8 M | RDRAM 6400 ($\varepsilon = 0.1, \delta = 0.15$) |
| Lapolli *et al.* [42] | 58.125 K | AMS Sampling | N/A | N/A | N/A | N/A | 1 Gbps |
| Ilha *et al.* [43] | 38.125 K | Count Sketch | N/A | 0.25 | $2^{18}$(256 K) | N/A | Entropy Norm |
| Soto *et al.* [44] | 560 K | CMin CU Sketch | 1.67% | 900 | 2.6~119.9 M | 5.3 M | FPGA Top K=8192 |
| Ding *et al.* [45] | 40 K | Count Sketch | <3% | 5 | ~2 M | N/A | P4 Mininet Simulation |
| Proposed | 480 K~640 K | Random Projection | <3% | 30, 300, 900 | ~98.8 M | ~5.07 M | Xilinx U200 FPGA $P4_{16}$ T2NA |

### 2) HASH TABLE WITH SKETCH DATA STRUCTURE

To compute the empirical Shannon entropy in Equation (1), the frequency count must be updated for every packet. Moreover, the update process should be performed at wire speed. Accordingly, Bartos and Martin [52] proposed a simple hardware accelerator for performing the update process using a logarithmic table with the linear interpolation technique. It was reported that a total memory space of 4.5 Mbits was required to compute the Shannon entropy for ten features of networking traffic. However, no details were given of the traffic traces, measurement accuracy, or attainable throughput.

Soto *et al.* [44] presented a high-throughput hardware accelerator for estimating the entropy of network traffic, in which the estimation process focused mainly on the top-k flows since the least frequent flows had no significant effect on the entropy of the data stream. The core of the accelerator consisted of a priority queue (PQ) array for the top-k flow selection and utilized the Count-Min sketch with Conservative Updates (CM-CU) [47] to evaluate the frequency statistics of the traffic flows. The system consumed approximately 560 K bytes (CM-CU + PQ array) of on-chip Block RAM and achieved a relative estimation error of less than 3% and mean value of 1.67% when evaluated using several real-world networking traffic traces. Moreover, the minimum processing throughput was shown to be 204 Gbps with a 400 Mhz system clock.

Ding *et al.* [45] proposed a framework of *P4DDoS* for detecting DDoS attacks in the data plane of P4 switches by estimating the empirical entropy based on the Count-Min [47], Count sketch [48], and *P4LogLog* with low relative error. The algorithm, designated as *P4LogLog*, was implemented using P4-supported arithmetic, bit shift, and bitwise logical operations. The performance of the proposed algorithm was evaluated using passive traffic traces drawn from the CAIDA dataset (2018) with each 5-second observation period containing around $2^{21}$ packets. The relative estimation error was shown to be close to 3% [45] when utilizing a Count-Sketch size of 40 Kbytes ($5 \times 2,000 \times 32$ bits).

### 3) RANDOM PROJECTION WITH STABLE DISTRIBUTION

Zhao *et al.* [8] proposed a streaming algorithm for estimating the entropy of Origin-Destination (OD) flows using the framework proposed by Indyk [20] for estimating the $L_p$ norm using a symmetric ($\beta = 0$) stable distribution $S(x : \alpha, \beta, \gamma, \mu)$. To avoid the complex computations usually required to generate random variables from a stable distribution, the authors utilized two pre-computed lookup tables implemented using the high-throughput Direct Rambus DRAM DIMM. Up to one million entries were allocated in each table, where each entry consisted of a total of 640 bits arranged in 20 blocks. In the proposed implementation, four tables were allocated, where each table contained 80,000 entries (buckets) of twenty 32-bit blocks. The tables consumed a total memory space of 51.2 Mbits between them. According to the simulation results, the relative error of the estimated entropy was less than 10% with a probability of approximately 0.85 [8].

Li [21] proposed efficient schemes of various estimators for estimating the Shannon entropy using the Compressed Counting (CC) [26] data structure based on a maximally-skewed ($\beta = 1$) stable random projection with $\alpha \to 1$ and a scale parameter of $\gamma = cos(\frac{\alpha\pi}{2})$. In later study, Li and Zhang later presented the New Estimator (NE) [25] for estimating the Shannon entropy, in which up to $k$ number of sketch registers were used to accomplish the random projection process for each packet in the traffic stream. Algorithm 1 presents the corresponding pseudo code, in which the update process based on stable random projection is shown in line 14 and the final entropy estimate is obtained in line 18. As shown in Algorithm 2, the random variables were generated using an observing key (*e.g.,* IPv4 source address) as the seed.

Clifford and Cosma [23] developed a sketching algorithm for entropy estimation over streaming data based on a maximally-skewed $\alpha$-stable random distribution ($\beta = -1$) with a scale parameter $\gamma = \frac{\pi}{2}$. Algorithm 3 shows the pseudo code of the proposed Log-Mean Estimator (LME) for an assumed stability parameter of $\alpha = 1$, in which the Shannon

**TABLE 2. List of notations.**

| Symbol | Description | Comments |
|---|---|---|
| $b$ | Resolution bit | The random number is computed with the increment of $(1/2^b)$. |
| $d_t$ | the *update* of a data item at time $t$ in the stream $\Phi$ | $d_t = 1$ represents packet count being updated. |
| $En$ | The total entry of the read-only lookup table | $En = 2^{th_{head}}/2^{sp}+(th_{tail}-th_{head})+th_{tail}$ |
| $En_{mc}$ | Total entry of the MC table, $Table_{MC}$ | $En_{mc} = 2^{16} = 65,536$ |
| $(\varepsilon, \delta)$ | Error factor $\varepsilon$, with probability at least $(1-\delta)$ | |
| $h_{ij}$ | The 2-Unversal hash function | $i = (0 \sim mp-1)$,$j = (0 \sim kp-1)$ |
| $\hat{H}$ | Estimated Entropy | |
| $H$ | Exact Entropy | |
| $k$ | The size of the data sketch. | |
| $key$ | the *key* of a data item at time $t$ in the stream $\Phi$ | |
| $kp$ | Number of the parallel read-only tables | |
| $mp$ | Number of hash functions used for lookup in one table | |
| $m$ | The total elements of a stream | $m_i$ denotes the frequency of item $i$ in a stream |
| $n$ | The total distinct items of a stream | |
| $\Phi$ | The packet stream | |
| $R$ | Random variable | |
| $s$ | The number of streams | |
| $S$ | $\alpha$ Stable Distribution $S(x:\alpha,\beta,\gamma,\mu)$ | stability $\alpha \in (0,2]$, skewness $\beta \in [-1,1]$, location $\mu \in \mathbb{R}$, scale $\gamma > 0$ |
| $sp$ | Span parameter | the span of $2^{sp}$ for fixed sampling under the cut-off threshold |
| $\triangle t$ | Observation Time | second |
| $th_{head}$ | Cut-off threshold | the threshold of $2^{th_{head}}$ for exponential sampling |
| $th_{tail}$ | Tail threshold | $th_{tail} = \frac{1}{2} log_2\, En_{mc}$ |
| $w$ | The width of the lookup table | bit |
| $X$ | The hash index | $X$=hash($key$) |
| $Y_{ij}$ | Sketch data structure | |
| $Y'_{ij}$ | Estimated sketch data structure | based on the interpolation process |

entropy is estimated by the LME shown in line 19 based on the $k$ sketch registers updated in line 13.

## III. PROBLEM STATEMENT

As shown in Algorithms 1 and 3, the primary process of the stream-based entropy estimation schemes proposed is to project each network traffic element, represented by a key and update pair $(key, d)$, to a random variable, R($key$), drawn from an alpha-stable distribution, $S(x : \alpha, \beta, \gamma, \delta)$. However, given a key, generating the random value from the skewed alpha-stable distribution requires complex computations involving division, logarithmic, and trigonometric operations (as shown in Algorithms 2 and 4, respectively). Consequently, the entropy estimation time is inevitably prolonged, therefore imposing a performance bottleneck on the anomaly detection process. Accordingly, the present study proposes an efficient tabulation scheme for estimating the empirical Shannon entropy $\hat{H}_i$ of high-speed network traffic, in which the complex computations used in Algorithms 2 and 4 to generate the required random values are replaced with a simple lookup table procedure. Moreover, a piece-wise linear interpolation heuristic based on three adaptive parameters, namely *Span*, *Exponential Head*, and *Exponential Tail*, is proposed to minimize the total size of the lookup table compared to that in previous studies [8].

**Algorithm 1** Algorithm Proposed by Li and Zhang [25] for Estimating the Entropy of Data Streams Using the New Estimator of Compressed Counting (NECC)

1: **Input:** $\alpha$, $key_t$
2: **Output:** $\hat{H}(\Phi)$, the estimated Shannon Entropy
3: **Initialization**
4: data sketch $(y_0, \ldots, y_{k-1}) \leftarrow (0, \ldots, 0)$
5: counter $Y \leftarrow 0$
6: $\Delta \leftarrow 1-\alpha$
7: $d_t \leftarrow 1$ //sketch update is based on packet count
8: **Function** $R(key_t)$:
9: generate the random variable $R$ with the maximally skewed alpha-stable distribution of $S(x;\alpha \rightarrow 1,1,1,cos(\frac{\alpha\pi}{2}))$, by using $key_t$ as the seed.
10: // For each incoming packet with *key* within the observation time $\Delta T$
11: Update the counter $Y = Y + 1$
12: **for** $j = 0$ to $k-1$ **do**
13: Generate $R_j(key_t) \sim S(x;\alpha \rightarrow 1,1,1,cos(\frac{\alpha\pi}{2}))$
14: Update $y_j = y_j + R_j(key_t) \times d_t$
15: **end for**
16: // At the end of the observation time $\Delta T$
17: $\hat{J}_\alpha = \frac{\triangle}{k}\sum_{j=0}^{k-1}(y_j)^{-\alpha/\triangle}$
18: $\hat{H}(\Phi) = -log(\hat{J}_\alpha) - \frac{1}{\triangle}log(Y^\alpha)$
19: **Return** $\hat{H}(\Phi)$

**Algorithm 2** Ping Li's Pseudo Codes [25] to Generate the Random Variable $R(key_t)$ With Maximally Skewed Alpha-Stable Distribution of $S(x;\alpha \rightarrow 1,1,1,cos(\frac{\alpha\pi}{2}))$

1: **Input:** $\alpha$, $key_t$ as the seed for generating random numbers
2: **Output:** random variable $R(key_t)$
3: Generate two random numbers $U_1, U_2 \sim Unif(0,1)$
4: $\triangle \leftarrow 1-\alpha$
5: $W_1 \leftarrow \pi U_1$
6: $W_2 \leftarrow -\log U_2$
7: $R(key_t) = \frac{sin(\alpha W_1)}{(sinW_1)^{1/2}}(\frac{sinW_1 \cdot \triangle}{W_2})^{\triangle/\alpha}$
8: **Return** $R(key_t)$

The proposed architecture is compatible with both the Log-Mean Estimator (LME) proposed by Clifford and Cosma [23] (see Algorithm 3) and the New Estimator of Compressed Counting (NECC) proposed by Li and Zhang [25] (see Algorithm 1). Moreover, through the use of a tabulation approach and the piece-wise linear interpolation heuristic, the proposed

method is suitable for implementation in the programmable data plane of limited-memory-space systems with a wire speed of multi-hundred gigabit per second. Table 2 summarizes the notations used in the present study.

**Algorithm 3** Algorithm Proposed by Clifford and Cosma [23] for Estimating the Entropy of Data Streams Using the Log-Mean Estimator (LME) $\hat{\delta}_{lm}(\zeta)$ With $\zeta = 1$

1: **Input:** $key_t$
2: **Output:** $\hat{H}(\Phi)$, the estimated Shannon Entropy
3: **Initialization**
4: data sketch $(y_0, \ldots, y_{k-1}) \leftarrow (0, \ldots, 0)$
5: counter $Y \leftarrow 0$
6: $d_t \leftarrow 1$ //sketch update is based on packet count
7: **Function** $R(key_t)$:
8: Generate the random variable $R$ with the maximally skewed alpha-stable distribution of $S(x; 1, -1, \frac{\pi}{2}, 0)$, by using $key_t$ as the seed.
9: // For each incoming packet with *key* within the observation time $\Delta T$
10: Update the counter $Y = Y + 1$.
11: **for** $j = 0$ to $k - 1$ **do**
12: Generate $R_j(key_t) \sim S(x; 1, -1, \pi/2, 0)$
13: Update $y_j = y_j + R_j(key_t. \times d_t)$
14: **end for**
15: // At the end of the observation time $\Delta T$
16: **for** $j = 0$ to $k - 1$ **do**
17: $y_j = \frac{y_j}{Y}$
18: **end for**
19: **Return** $\hat{H}(\Phi) = -\log(k^{-1} \sum_{j=0}^{k-1} exp(y_j))$

**Algorithm 4** Clifford and Cosma's Pseudo Codes [23] to Generate the Random Variable $R(key_t)$ With Maximally Skewed Alpha-Stable Distribution of $S(x; 1, -1, \frac{\pi}{2}, 0)$

1: **Input:** $key_t$ as the seed for generating random numbers
2: **Output:** random variable $R(key_t)$
3: Generate two random numbers $U_1$,$U_2 \sim Unif(0, 1)$
4: $W_1 \leftarrow \pi(U_1 - \frac{1}{2})$
5: $W_2 \leftarrow -\log U_2$
6: $R(key_t) = \tan(W_1)\left[\frac{\pi}{2} - W_1\right] + \log(W_2 \frac{\cos W_1}{\pi/2 - W_1})$
7: **Return** $R(key_t)$

## IV. SYSTEM DESIGN

### A. OBSERVATION

In general, a random number $r$ can be generated by seeding a pseudo-random number generator with a key $key_t$, where $0 \leq r \leq RAND\_MAX$. The random numbers $U_1, U_2 \sim$ Unif(0, 1), generated in Algorithms 2 and 4, can be calculated as $U_1 = r_1/2^b$ and $U_2 = r_2/2^b$, where parameter $b$ represents the computational resolution of the random values and $RAND\_MAX = 2^b - 1$. Having generated these random numbers, the corresponding values of $R(key_t)$ can be obtained accordingly. Figure 1 presents a three-dimensional (3D) figure of the random value $R(key_t)$ for a maximally skewed alpha-stable distribution with a resolution of $1/(2^{16})$ in Algorithm 4.

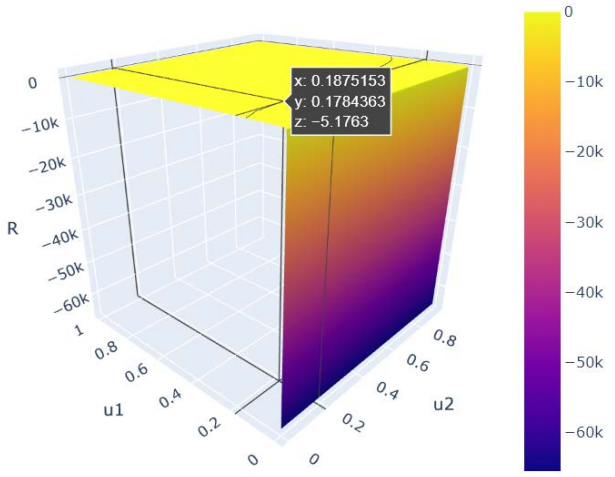



**FIGURE 1.** 3-D figure of pre-computed random value $R$ for a maximally-skewed alpha-stable distribution $S(x; 1, -1, \frac{\pi}{2}, 0)$ with a resolution of $b = 16$. Note that the z-axis represents the value of $R$, and the x- and y-axis represents the random values of $U1$ and $U2$, respectively in Algorithm 4.

Based on general logarithmic properties, and grouping the variables $W_1$ and $W_2$ separately, line 6 in Algorithm 4 can be represented in the form of $R(key_t) = f_1(W_1) + f_2(W_2)$. The values of $f_1(W_1)$ and $f_2(W_2)$ can be computed in advance and stored in two lookup tables $T_1[\,]$ and $T_2[\,]$ each of size $E_n$. The random value $R(key_t)$ can then be obtained simply as $R(key_t) = T_1[r_1] + T_2[r_2]$. A similar technique can be applied to line 7 in Algorithm 2 such that the random value $R(key_t)$ can be obtained as the product of $T_1[r_1]$ and $T[r_2]$. It is noted that Zhao *et al.* [8] adopted a similar approach with a large entry size (*i.e.*, $E_n$ approximately one million) and allocated the tables in an off-chip Rambus DRAM.

### B. PIECE-WISE LINEAR INTERPOLATION HEURISTIC

Algorithm 5 presents the offline table construction process in the proposed present study. Briefly, an inverse transform sampling approach is employed to draw up to $En_{mc}$ pairs of random values $(U1, U2) \sim Unif(0, 1)$ with a resolution of $1/(2^b)$ (lines 21 and 23) from the alpha-stable distribution. These values, shown as blue dots in Figure 2, are then sorted in ascending order and stored in a lookup table $Table_{MC}$ of size $En_{mc}$ (line 26). The purpose here is to re-construct the empirical distribution function of $R(U1, U2)$ in Algorithms 2 and 4 in a tabular form. Note that, the sorting for Algorithm 2 is based on the descending order due to the property of the distribution.

Observing the empirical distribution represented by the blue dots in Figure 2, it is seen that, with more than 98.5% probability, the values are distributed within a small range

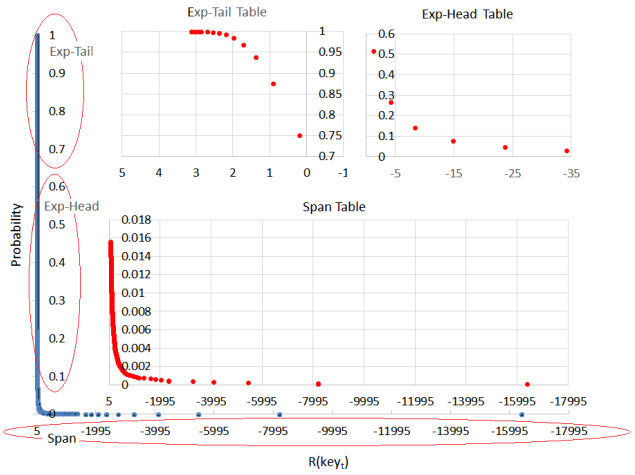



**FIGURE 2.** Empirical distribution of maximally-skewed alpha-stable distribution $S(x; 1, -1, \frac{\pi}{2}, 0)$ based on $x = 65,536$ samples. Note that the interpolation tables are constructed based on the *Span*, *Head* and *Tail* regions of the distribution in Algorithm 4.

of -5 to 5. Furthermore, the empirical distribution contains only a very small number (1.5%) of values in the wider range of -5 to -17,995. Therefore, by using the piece-wise linear interpolation technique illustrated in Figure 3, only a few selected points are sufficient to approximate most of the values in the original empirical distribution [53]. Consequently, the size of the lookup table used to reproduce the random values, $R(key_t)$, in Algorithms 2 and 4 can be substantially reduced. Crucially, the proposed scheme enables the random values not stored in the table (*i.e.*, approximately 98% of the random values stored in the original $Table_{MC}$ of 64 K entries) to be obtained via simple mathematical and logical operations.

To further reduce the size of the lookup table, $Table_{MC}$, the present study decomposes the table into three smaller tables, namely the *Span* table ($Table_{span}$), *Exp-Head* table ($Table_{exp-head}$) and *Exp-Tail* table ($Table_{exp-tail}$), where the values stored in the three tables are selected in accordance with two threshold parameters, $th_{head}$ and $th_{tail}$. Notably, the

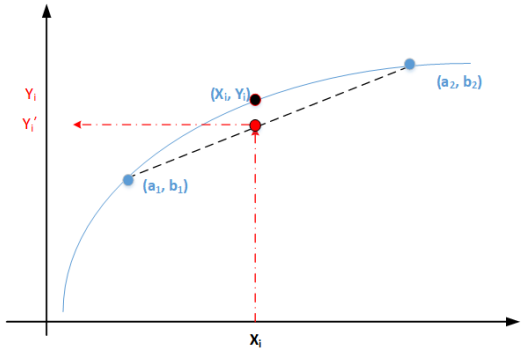



**FIGURE 3.** Schematic illustration of interpolation process $Y_i' = b2 + \frac{(b2-b1)}{(a2-a1)}(X_i - a1)$ based on two known points of $(a1, b1)$ and $(a2, b2)$. The interpolation error $\Delta Y_i$ is defined by $|Y_i - Y_i'|$.

two parameters can be tuned adaptively as required to represent the particular skew characteristics of the alpha-stable distribution. The goal is to achieve an acceptable tradeoff between the total table size (i.e., the memory consumption, smaller is better) and the entropy estimation accuracy (larger is better).

The *Span* table is constructed by selecting values from $Table_{MC}$ with a fixed span of $2^{sp}$, starting from index zero and ending at index $2^{th_{head}}$. In other words, in the entropy estimation process, the *Span* table is accessed for all lookup values with hash indexes less than or equal to $2^{th_{head}}$.

By contrast, the *Exp-Head* table (see upper-right corner of Figure 2) is accessed for all lookup values with hash indexes $2^n$ in the interval between $2^{th_{head}}$ and $2^{th_{tail}}$. Finally, for lookup values with hash indexes greater than $2^{th_{tail}}$, the hash indexes are inverted and the *Exp-Tail* table shown in the upper-left corner of Figure 2 is accessed.

Example: For $x = 65,536$ samples, $sp = 1$, $th_{head} = 11$ and $th_{tail} = 15$, the *Span* table requires $2^{th_{head}}/2^{sp} = 1,024$ entries. Meanwhile, the *Exp-Head* table requires five entries to store the values for lookup indexes $2^{11}, 2^{12}, \ldots$, and $2^{15}$. The *Exp-Tail* table requires 15 entries to store the values for the remaining indexes. In other words, the total size *En* of one K-parallel table is just 1,044 entries (*i.e.*, $En = 1,024 + 5 + 15 = 1,044$). That is, the size of the original table $Table_{MC}$ (65,536 entries) is reduced by 98.4%.

### C. DATA PLANE PACKET PROCESSING

As shown in Algorithms 1 and 3, the number of random values generated from the skewed alpha-stable distribution depends on the size of the data sketch $k$. To avoid the need to perform $k$ sequential lookups over the same table, the present study proposes a *k-parallel* with *m-hash* lookup data structure consisting of $kp$ read-only tables, as shown in Figure 4. For each table, $mp$ hash functions are used to compute the indexes of a given key for table lookup purposes. Through the use of this *k-parallel* structure, the total lookup latency is significantly reduced and the maximum throughput increased correspondingly. Moreover, for values selected from $Table_{MC}$, the hash indexes to the table have the form of two to the power of $n$. Thus, the complex division computation in the proposed linear interpolation scheme (see Figure 3) can be replaced by a simple logic shift operator.

For each incoming packet, up to $mp \times kp$ hash values are computed based on the *key* selected. As shown in Algorithm 6, these hash values are then used to index the relevant lookup tables following simple logical shift and invert operations (see lines 20 and 34).

In particular, as described above, three different address ranges are specified based on the threshold values assigned to $th_{head}$ and $th_{tail}$. In accordance with these ranges, the *Span* ($Table_{span}$), *Exp-Head* ($Table_{exp-head}$) and *Exp-Tail* ($Table_{exp-tail}$) tables are accessed as appropriate to proceed with linear interpolation (see lines 26, 32, and 39) such that the final sketch data structure of $Y_{ij}$ (line 41) can be obtained.

**Algorithm 5** Algorithm for Table Construction in the Offline Stage

**Initialization**
Total entry of the MC table $En_{mc} \leftarrow [2^{14}, 2^{15} \ldots]$
Set number of parallel tables $kp \leftarrow [20, \ldots]$
Set the Resolution bits $b \leftarrow [10, \ldots, 31]$
Set the Span bits $sp \leftarrow [0, 1, 2]$
Set the Head thresholds $th_{head} \leftarrow [10, .., log_2 En_{mc} - 1]$
Set the Tail thresholds $th_{tail} \leftarrow (log_2 En_{mc} - 1)$
**Function** *sort*(*table*):
sort the content in increasing order
**Function** *invert*($x$):
inverting bits in the binary representation of $x$
// Construct the $\alpha$-stable Table
**for** $i = 0$ to $2^b - 1$ **do**
  **for** $j = 0$ to $2^b - 1$ **do**
    $U_1 \leftarrow i/2^b$, $U_2 \leftarrow j/2^b$
    $Table_{alpha}[i][j] \leftarrow S(U_1, U_2 : \alpha, \beta, \gamma, \delta)$
  **end for**
**end for**
// Construct the Empirical Distribution
**for** $j = 0$ to $(kp - 1)$ **do**
  **for** $i = 0$ to $(En_{mc} - 1)$ **do**
    Generate two random numbers $m, n$ where $0 \leq m, n < 2^b$
    ${}_jTable_{tmp}[i] \leftarrow Table_{alpha}[m][n]$
  **end for**
  **for** $i = 0$ to $(En_{mc} - 1)$ **do**
    ${}_jTable_{MC}[i] = \text{sort}\ ({}_jTable_{tmp}[i])$
  **end for**
  // Construct the Span Table
  **for** $i = 0$ to $((2^{th_{head}}/2^{sp}))$ **do**
    ${}_jTable_{span}[i] \leftarrow {}_j\ Table_{MC}[i \ll sp]$
  **end for**
  // Construct the Head Exp Table
  **for** $i = th_{head}$ to $th_{tail}$ **do**
    ${}_jTable_{exp-head}[i - th_{head}] \leftarrow {}_jTable_{MC}[2^i]$
  **end for**
  // Construct the Tail Exp Table
  **for** $i = 0$ to $th_{tail}$ **do**
    ${}_jTable_{exp-tail}[i] \leftarrow {}_jTable_{MC}[invert(2^i - 1)]$
  **end for**
**end for**

At the end of the observation time $\Delta T$, the host CPU collects this data structure from the fast data plane and computes the empirical Shannon entropy using the Log-Mean [23] and New estimator [25] described in Section II-C3. The corresponding pseudo codes for the two proposed estimation processes are shown in Algorithms 7 and 8, respectively.

## V. DESIGN EXPLORATION

The memory space required by the proposed scheme is equal to the total size ($kp \times En$) of the *Span-Head-Tail* tables shown in Figure 4, where $kp$ denotes the number of lookup tables deployed in parallel, and $En$ is the size (*i.e.*, total number of entries) of each table. It is noted that the two parameters ($kp$ and $En$) play a key role in determining the accuracy and variance of the final entropy estimates.

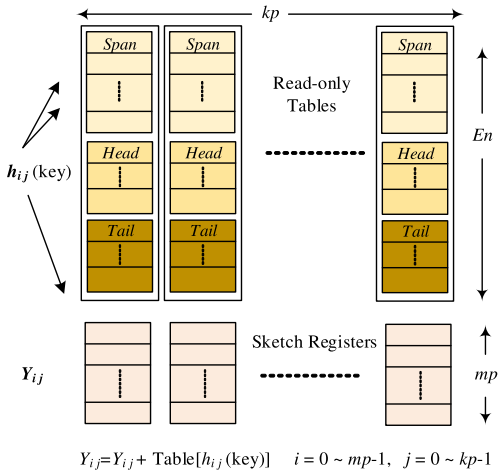



**FIGURE 4.** Block diagrams of the proposed *k*-parallel with *m*-hash data structure consisting of *kp* Span-Head-Tail tables (read only) of size *En* and sketch registers $Y_{ij}$ of size *kp* × *mp*.

**TABLE 3.** Total entry size (*En*) of the *Span-Head-Tail* tables constructed using different *span* and *threshold* parameters. Note that the the MC table is assumed to have a total entry size ($En_{mc}$) of $2^{16}$.

| $sp$ | $th_{head}$ | $th_{tail}$ | Total Entry of Lookup Table ($En$) |
|---|---|---|---|
| 0 | 12 | 15 | $\frac{2^{12}}{2^0} + (15 - 12 + 1) + 15 = 4,115$ |
| 1 | 12 | 15 | $\frac{2^{12}}{2^1} + (15 - 12 + 1) + 15 = 2,067$ |
| 1 | 11 | 15 | $\frac{2^{11}}{2^1} + (15 - 11 + 1) + 15 = 1,044$ |

### A. TABLE SIZE REDUCTION

The total entry ($En$) of the *Span-Head-Tail* table constructed using Algorithm 6 can be summarized as

$$En = \frac{2^{th_{head}}}{2^{sp}} + (th_{tail} - th_{head} + 1) + th_{tail}, \qquad (2)$$

where $th_{tail} = \log_2 En_{mc} - 1$ and $th_{head} \leq th_{tail}$.

The proposed linear interpolation scheme substantially reduces the total lookup table size. As illustrated in Equation (2), $th_{head}$ and $sp$ are the dominating parameters of the table size.

Example: As shown in Table 3, for $th_{head} = 11$, $sp = 1$ and $En_{mc} = 64K$, the *Span* table ($Table_{span}$) consumes only 1,024 entries of memory space. Moreover, for $th_{tail} = 15$, *Exp-Head* table ($Table_{exp-head}$) and *Exp-Tail* table ($Table_{exp-tail}$) require just five and fifteen entries, respectively. Thus, the interpolation-based table construction process (Algorithm 6) reduces the consumed memory space by 98.4%.

The average error distance between the values stored in the original table, $Table_{MC}$, and those derived by the proposed

**Algorithm 6** Algorithm for Table Lookup and Piece-Wise Linear Interpolation for Every Incoming Packet on the Data-plane Within the Observation Time $\Delta T$

**Input:** *key* (e.g., source IP address) of the packet stream $\Phi$
**Output:** Sketch data structure $Y_{ij}$, Total packet count $pkt_{count}$ at the end of each observation time $\Delta T$
**Initialization**
Total entry of the MC table *En*
Set number of parallel tables *kp*
Set number of hash functions used for a table *mp*
Set the Span bits *sp*
Set the Head thresholds $th_{head}$
Set the Tail thresholds $th_{tail}$
$d_t \leftarrow 1$ //sketch update is based on packet count
$Y_{ij} \leftarrow 0, pkt_{count} \leftarrow 0$
**Function** $\phi(x)$:
The position of the most significant 1-bit in the binary representation of $x$
**Function** $h_{ij}(x)$:
compute hash values of $x$ based on the family of 2-Unversal hash functions
// For each incoming packet with *key* within the observation time $\Delta T$
**for** $i = 0$ to $mp - 1$ **do**
  **for** $j = 0$ to $kp - 1$ **do**
    $indx_{ij} \leftarrow h_{ij}(key)$
    $indx'_{ij} \leftarrow indx_{ij} \gg sp$
    **if** $(indx_{ij} < 2^{th_{head}})$ **then**
      $a1_{ij} \leftarrow indx'_{ij} \ll sp$
      $a2_{ij} \leftarrow (indx'_{ij} + 1) \ll sp$
      $b1_{ij} \leftarrow {}_jTable_{span}[indx'_{ij}]$
      $b2_{ij} \leftarrow {}_jTable_{span}[indx'_{ij} + 1]$
      $Y'_{ij} \leftarrow b1_{ij} + \{(b2_{ij} - b1_{ij}) \times (indx_{ij} - a1_{ij})\} \gg sp$
    **else if** $(2^{th_{head}} \leq indx_{ij} \leq 2^{th_{tail}})$ **then**
      $a1_{ij} \leftarrow (1 \ll \phi(indx_{ij}))$
      $a2_{ij} \leftarrow (1 \ll (\phi(indx_{ij}) + 1))$
      $b1_{ij} \leftarrow {}_jTable_{exp-head}[\phi(indx_{ij})]$
      $b2_{ij} \leftarrow {}_jTable_{exp-head}[\phi(indx_{ij}) + 1]$
      $Y'_{ij} \leftarrow b1_{ij} + \{(b2_{ij} - b1_{ij}) \times (indx_{ij} - a1_{ij})\} \gg \phi(indx_{ij})$
    **else**
      $\overline{indx_{ij}} \leftarrow invert(indx)$
      $\underline{a1_{ij}} \leftarrow ((1 \ll (\phi(\overline{indx_{ij}}) + 1)) - 1)$
      $\overline{a2_{ij}} \leftarrow ((1 \ll \phi(\overline{indx_{ij}})) - 1)$
      $b1_{ij} \leftarrow {}_jTable_{exp-tail}[\phi(\overline{indx_{ij}}) + 1]$
      $b2_{ij} \leftarrow {}_jTable_{exp-tail}[\phi(\overline{indx_{ij}})]$
      $Y'_{ij} \leftarrow b1_{ij} + \{(b2_{ij} - b1_{ij}) \times (indx_{ij} - invert(\overline{a1_{ij}}))\} \gg \phi(\overline{indx_{ij}})$
    **end if**
    Update $Y_{ij} \leftarrow Y_{ij} + Y'_{ij} \times d_t$
  **end for**
**end for**
$pkt_{count} + +$
// At the end of the observation time $\Delta T$
**Return** $Y_{ij}, pkt_{count}$

**Algorithm 7** Proposed Algorithm Based on Ping Li's New Estimator [25] for Estimating the Shannon Entropy at the End of Observation Time $\Delta T$ on the Control Plane

**Input:** $\alpha$, Sketch data structure $Y_{ij}$, Total packet count $pkt_{count}$ of packet stream $\Phi$
**Output:** $\hat{H}(\Phi)$, the estimated Shannon Entropy
Set number of parallel tables $kp \leftarrow [10, \ldots]$
Set number of hash functions used for a table $mp \leftarrow [1, 2, 3, \ldots]$
// At the end of the observation time $\Delta T$
$\Delta = 1 - \alpha$
$\hat{J}_\alpha = \frac{\Delta}{mp \cdot kp} \sum_{i=0}^{mp-1} \sum_{j=0}^{kp-1} (Y_{ij})^{-\alpha/\Delta}$
$\hat{H}(\Phi) = -log(\hat{J}_\alpha) - \frac{1}{\Delta} log(pkt_{count}^{\alpha})$
**Return** $\hat{H}(\Phi)$

**Algorithm 8** Proposed Algorithm Based on Clifford and Cosma's Log-Mean Estimator [23] to Estimate the Shannon Entropy at the End of Observation Time $\Delta T$ on the Control Plane

**Input:** Sketch data structure $Y_{ij}$, Total packet count $pkt_{count}$ of packet stream $\Phi$
**Output:** $\hat{H}(\Phi)$, the estimated Shannon Entropy
Set number of parallel tables $kp \leftarrow [10, \ldots]$
Set number of hash functions used for a table $mp \leftarrow [1, 2, 3, \ldots]$
// At the end of the observation time $\Delta T$
**for** $i = 0$ to $mp - 1$ **do**
  **for** $j = 0$ to $kp - 1$ **do**
    $Y_{ij} \leftarrow \frac{Y_{ij}}{pkt_{count}}$
  **end for**
**end for**
$\hat{H}(\Phi) \leftarrow (-log(\frac{1}{mp \cdot kp} \sum_{i=0}^{mp-1} \sum_{j=0}^{kp-1} exp(Y_{ij}))$
**Return** $\hat{H}(\Phi)$

piece-wise linear interpolation process can be evaluated as Equation (3).

$$L1 = \frac{1}{En_{mc}} \sum_{i=0}^{En_{mc}-1} |Y_i - Y'_i| \tag{3}$$

Figure 5 illustrates the selection of cut-off thresholds ($th_{head}$) and the total entry consumed of the proposed interpolation scheme. For ease of comparison, as shown in Figure 6, two baselines of the average error distance are shown, namely (1) the dotted *L1_quarter* line, which represents the L1 distance of the quarter-sized interpolation table in which every $4_{th}$ entries of the original table $Table_{MC}$ is retained; and (2) the dashed *L1_half* line, which represents the average error distance of the half-sized interpolation table in which every $2_{nd}$ entries of the original one is retained. As expected, the average error distance of the *L1_half* scheme is much less

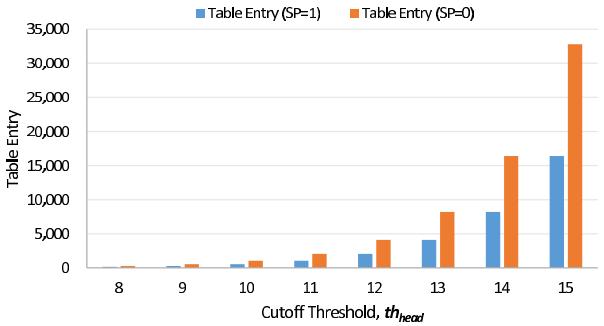



**FIGURE 5.** Table size (entry) of proposed interpolation-based method for entropy estimation using the scheme LME of Clifford & Cosma [23] and various cut-off thresholds. Note that the results are obtained using a resolution of fourteen bits ($b$=14) with spans of one ($sp$=0) and two ($sp$=1), respectively.

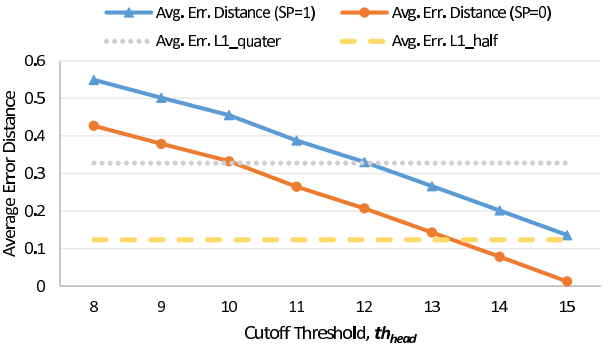



**FIGURE 6.** Average error introduced in proposed interpolation-based method for entropy estimation using the scheme LME of Clifford & Cosma [23] and various cut-off thresholds. Two reference lines (*L1_quarter, L1_half*) of the average error distance are provided. The *L1_quarter* line shows the average error distance of the quarter-sized interpolation table constructed by sampling every $4_{th}$ entry in $Table_{MC}$, while the *L1_half* line shows the average error distance of the half-sized interpolation table constructed by sampling every $2_{nd}$ entry in $Table_{MC}$.

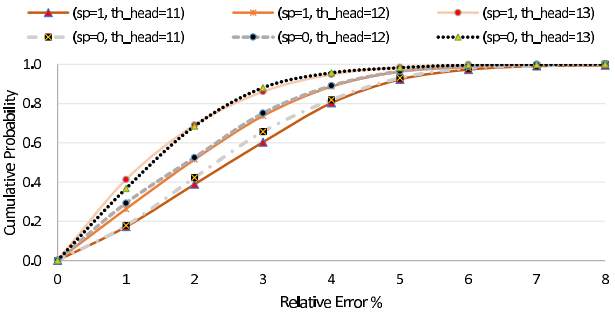



**FIGURE 7.** Cumulative probability of various error percentages for different combinations of cut-off threshold ($th_{head}$) and span ($sp$) parameters. Note that the simulation results are obtained using a 30-second segment of the MAWI traffic trace (200701011400) with thirty parallel tables ($kp = 30$) and twelve hash functions ($mp = 12$) per table based on the interpolation-based LME scheme of Clifford & Cosma [23].

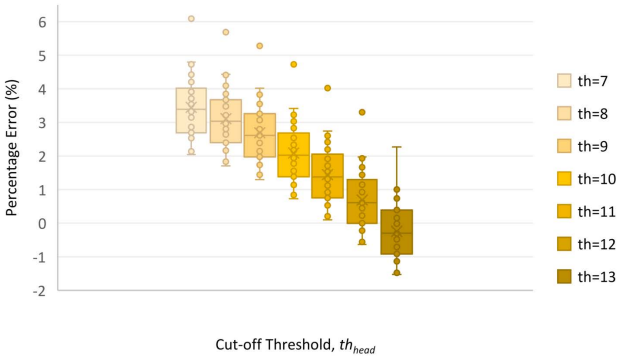



**FIGURE 8.** Box-and-whisker plot of the percentage errors for different cut-off thresholds ($th_{head} = 7, 8, \ldots, 13$) with span of one ($sp = 1$). Note that the simulation results are obtained using a 30-second segment of the MAWI traffic trace (200701011400) with thirty parallel tables ($kp = 30$) and twelve hash functions ($mp = 12$) per table based on the interpolation-based LME scheme of Clifford & Cosma [23].

than that of the *L1_quarter* scheme since twice the number of original values are preserved in the reconstructed table.

A simple approach for constructing an interpolation table with half the size of the original MC table is to skip every second entry in the original table. However, the resulting tradeoff between the interpolation error and the memory space saving may be sub-optimal. As shown in Figure 5, by using a cut-off threshold of thirteen ($th_{head} = 13$) and storing all the values ($sp = 0$) in the original table, the L1 distance reaches almost the same level as that provided by the *L1_half*. Moreover, the consumed memory space of the interpolation table $Table_{span}$ is equal to just 12.5% of that of the original table $Table_{MC}$. Furthermore, if every $2_{nd}$ entries ($sp = 1$) is selected, the memory space can be halved (4,114 entries) while the average error distance is still less than the level of the *L1_quarter* line.

Accordingly, simulations were performed using a real-world Internet traffic trace extracted from the MAWI dataset [54] to evaluate the effects of the cut-off threshold ($th_{head}$) and span ($sp$) on the cumulative probability of various error percentages. The corresponding results are shown in Figure 7. It is seen that for parameter settings of $th_{head} = 13$ and $sp = 0$, the cumulative probability reaches almost 0.9 within an error percentage of 3%. Furthermore, the table size can be halved using a span size of two ($sp = 1$) with no more than a minor reduction in the cumulative probability. Figure 8 demonstrates the box-and-whisker plot of the percentage errors for different cut-off thresholds. The distribution of relative errors declines as the cut-off threshold increases. For $th_{head} = 10$, the upper quartile is less than 3%.

### B. K-PARALLEL TABLE WITH M-HASH

In practice, the variance of the entropy estimates obtained using the proposed scheme can be further reduced through an averaging approach by using multiple tables in parallel and independent hash functions for each table lookup. Figure 9 shows the cumulative probability of the relative error of the entropy estimates given the use of different numbers of lookup tables. The relative error, defined as $(\frac{|\hat{H}-H|}{H}) \times 100\%$, is the absolute error divided by the magnitude of the exact entropy. It can be seen that for a cumulative probability of 0.9, for example, the relative error reduces from 7% to just

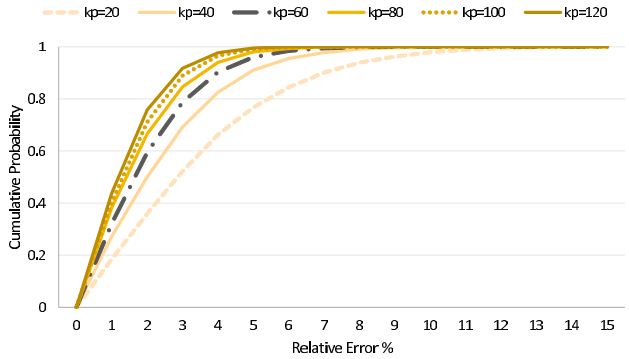



**FIGURE 9.** Cumulative probability of relative error for estimated Entropy using different numbers of tables ($kp = 20 \sim 120$) in parallel. Note that the number of hash function is eight ($mp = 8$) in every case.

2.9% as the number of tables increases from twenty ($kp = 20$) to forty ($kp = 40$) given the use of eight hash functions ($mp = 8$) in every case. By contrast, as shown in Figure 10, the cumulative probability increases from 0.71 to 0.98 as the number of hash functions increases from two ($mp = 2$) to sixteen ($mp = 16$) for a relative error of 5%.

Note that the results presented in Figures 9 and 10 are obtained using the interpolation-based scheme of Clifford and Cosma [23] and a synthetic traffic stream consisting of one million elements. In addition, the skew parameter [55] is set as Zipf=1.4 (*i.e.*, the same as that of the real-world MAWI trace used in the previous simulations) and the span and cut-off threshold parameters have values of $sp = 1$, $th_{head} = 11$, and $th_{tail} = 15$, respectively.

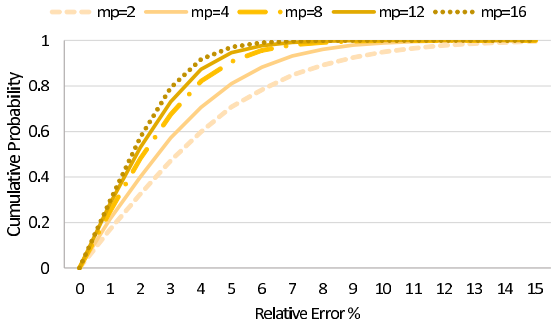



**FIGURE 10.** Cumulative probability of relative error for estimated entropy (LME) using different numbers of hash functions for each table ($mp = 2 \sim 16$). Note that the number of tables used in parallel is twenty ($kp = 20$) in every case.

### C. RESOLUTION

Parameter $b$ in the proposed table construction algorithm (Algorithm 5) defines the resolution, $1/2^b$, of the values stored in the lookup table. For a fixed value of $En_{mc}$ (*e.g.*, 65,536, see above), a higher resolution results in a more accurate estimation performance for a larger number of distinct keys in the packet stream. Figure 11 shows the mean absolute percentage error (MAPE), defined as $\frac{1}{s}\sum_{i=1}^{s}(\frac{|\hat{H}_i - H_i|}{H_i}) \times 100\%$, of the estimated entropy for one thousand streams ($s = 1,000$) of different distributions and resolutions.

It is seen that for a typical real-world network trace [54] with a moderate skew (Zipf=1.4) and a length of one million elements (approximately 18 K distinct elements), a resolution bit size of 12 is sufficient to yield an accurate entropy estimation performance. However, for synthetic data streams containing 100 K, 1 M, and 10 M different items [55], resolution bit sizes of at least 16, 18 and 22, respectively, are required to achieve an adequate estimation accuracy.

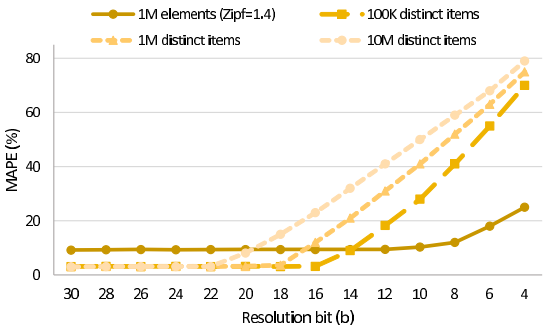



**FIGURE 11.** Mean absolute percentage error (MAPE) of estimated entropy for data streams of different distributions and different resolution bits. Note that the evaluation results are obtained using the interpolation-based LME of Clifford and Cosma with a sketch size of $k = 20$.

### D. SIZE AND ERROR TRADEOFF

In general, the entry size ($En_{mc}$) of $Table_{MC}$ used to store the values selected from $Table_{alpha}$ through the inverse transform sampling process should not be too small. The reason is that if those random values are sampled coarse-grained, it is hard to represent the original stable distribution; hence, the interpolation error increases. Based on two distributions (Zipf=0.1 and 1.9), Figure 12 illustrates the mean absolute percentage error with different sizes of the $Table_{MC}$ for the interpolation-based entropy estimations of LME and NECC ($kp = 20$, $mp = 1$). The mean value is obtained based on simulations of 1, 000 times.

It is apparent that for a highly-skewed stream (Zipf=1.9), the mean absolute percentage error remains the same as the size increases. In contrast, for a uniform-distributed stream (Zipf=0.1), the larger the entry size, the lower the error. The LME and NECC interpolation schemes achieve less than 5% of the mean error with the 512 K entry of the MC table.

Please be noted that $Table_{MC}$ is a temporary data structure storing the random values generated by the inverse transform sampling. Based on this table, the proposed interpolation process further creates the *Span-Head-Tail* table with a much smaller size suitable for system implementation.

As indicated in Equation (2), the size of the *Span-Head-Tail* table is mainly affected by the parameters of cut-off ($th_{head}$) and span parameters ($sp$). Figures 13 and 14 demonstrate the mean absolute percentage error with different cut-off thresholds ($th_{head}$) and span parameters ($sp$) for the interpolation-based entropy estimations.

Obviously, using a lower span value, the larger the cut-off threshold, the lower the mean absolute percentage error

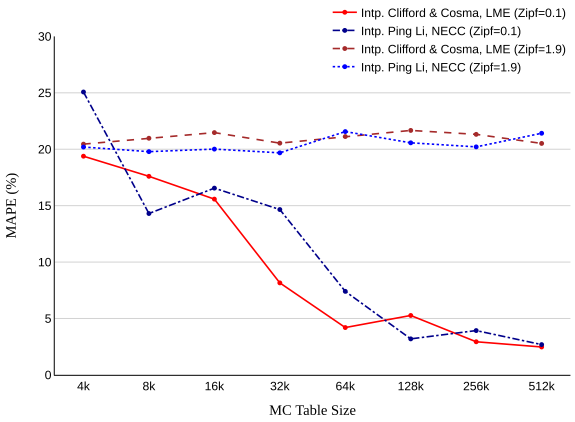



**FIGURE 12.** Mean absolute percentage error of the proposed interpolation-based methods with different sizes ($En_{mc}$) of the MC Table. The simulation is conducted based on two synthetic data streams of Zipf=0.1 and 1.9.

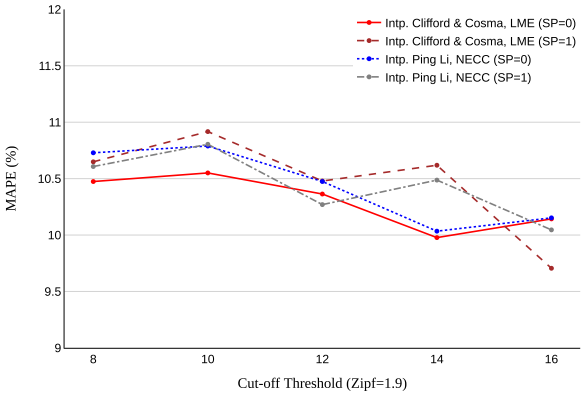



**FIGURE 14.** Mean absolute percentage error of the proposed interpolation-based LME and NECC methods with different cut-off thresholds ($th_{head}$) and span parameters ($sp = 0, 1$). The simulation is conducted based on a skewed synthetic data stream (Zipf=1.9) with the MC table size ($En_{mc}$) of 512$K$.

for the estimation. However, for a fixed span parameter, the table size ($En$) increases exponentially as the cut-off threshold increases. Therefore, for approximately 4 K-entry of the k-parallel table implementation, suitable selections of the span and cut-off threshold parameters can be ($sp = 0$, $th_{head} = 12$) or ($sp = 1$, $th_{head} = 13$).

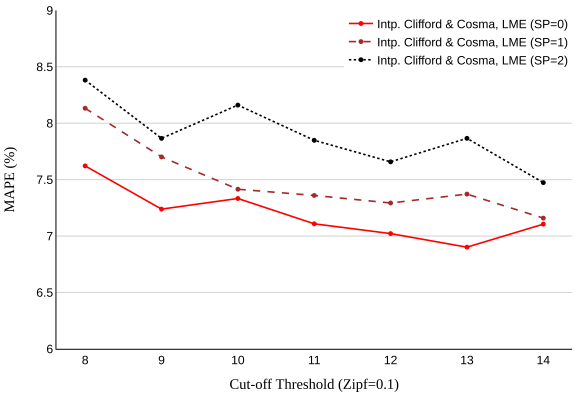



**FIGURE 13.** Mean absolute percentage error of the proposed interpolation-based LME methods with different cut-off thresholds ($th_{head}$) and three span parameters ($sp = 0, 1, 2$). The simulation is conducted based on a uniform synthetic data stream (Zipf=0.1) with the MC table size ($En_{mc}$) of 512$K$.

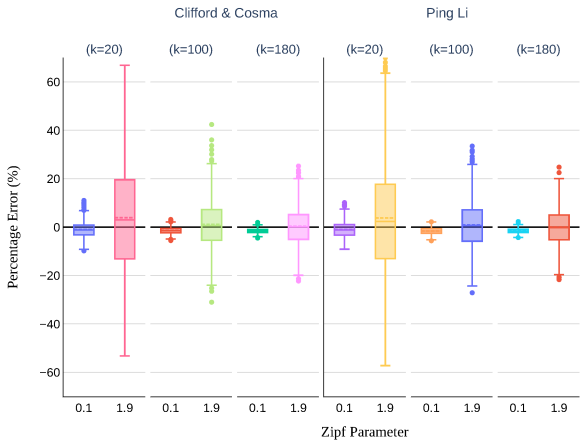



**FIGURE 15.** Box-and-whisker plot of the relative percentage errors of entropy estimates obtained using the schemes of original LME of Clifford & Cosma (left section) and NECC of Ping Li (right section) for Zipf parameters of 0.1 and 1.9 and different numbers of sketch data structure ($k = 20 \sim 180$).

## VI. SYSTEM EVALUATION

The feasibility of the proposed interpolation-based platform for estimating the Shannon entropy using either the algorithm of NECC Ping Li and Zhang [25] or LME schemes of Clifford and Cosma [23] was evaluated using both synthetic data stream and real-world traffic traces adopted from the MAWI dataset [54] and the CAIDA DDoS attack dataset [56].

### A. BASELINE

In general, up to $k$ numbers of $R_k$ values are used in the LME [23] and NECC [25] schemes to minimize the variance of the empirical Shannon entropy estimates. For example, Zhao *et al.* [8] used twenty blocks of sketch data ($k = 20$) in the lookup table implementation. Figure 15 presents box-and-whisker plots of the relative errors obtained using the schemes of original LME of Clifford & Cosma (left section) and NECC of Ping Li (right section) for distributions with two extreme cases ($Zipf = 0.1, 1.9$) and three different table configurations ($k = 20, 100, 180$). The variance of the estimated entropy can be minimized effectively with a higher number of sketch data structures ($k$) in both schemes.

Figure 16 compares the mean estimated entropy values obtained from the original LME [23] and NECC [25] schemes with those obtained from the correspondence interpolation-based schemes ($kp = 20, mp = 1$). For both schemes, the simulations were repeated 1,000 times with different hash parameters each time and the results were then derived as mean values with the standard deviation shown as error bars.

On average, as shown in Figure 17, the original LME and NECC schemes yield a relative error of less than 5% compared to the exact entropy solutions with a Zipf value less

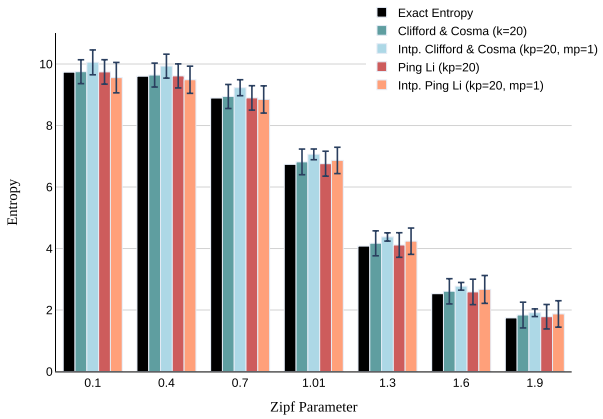



**FIGURE 16.** Estimated entropy obtained using original and proposed interpolation-based LME [23] and NECC [25] schemes for stable random distributions with different Zipf parameters. Note that the original LME and NECC schemes are implemented using a data sketch size of twenty ($k = 20$), while the proposed interpolation-based schemes are implemented using twenty tables ($kp = 20$) in parallel and one hash function per table ($mp = 1$).

than 1.01. By contrast, the proposed schemes overestimate the entropy value due to the interpolation error introduced. In particular, the mean relative error of both schemes increases to approximately 18% as the stream elements become highly-skewed (Zipf = 1.9) distributed. Fortunately, as shown in Figure 17, the mean relative error for a traffic distribution with a Zipf value of 1.9 can be reduced to around 6% using the proposed interpolation-based schemes given the use of more hash functions per table ($kp = 20$, $mp = 12$).

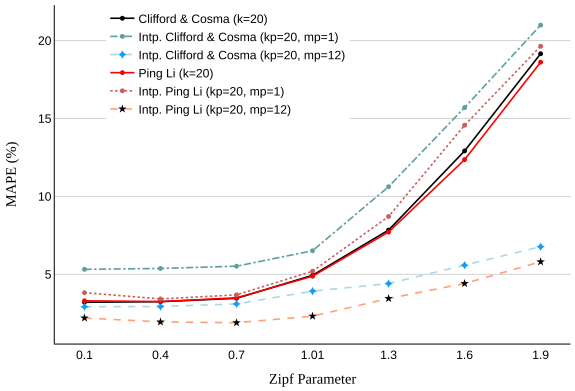



**FIGURE 17.** Mean absolute percentage error of estimated entropy obtained using original and proposed interpolation-based LME [23] and and NECC [25] schemes for stable random distributions with different Zipf parameters. Note that the original LME and NECC schemes are implemented using a data sketch size of twenty ($k = 20$), while the proposed interpolation-based schemes are using twenty tables ($kp = 20$) in parallel and one hash function per table ($mp = 1$) and twelve ($mp = 12$).

The simulations considered synthetic data streams, each consisting of 30 K items, generated using different Zipf parameters in the range of 0.1 to 1.9 [55]. For the interpolation-based scheme, the head threshold parameter was set as ten ($th_{head} = 10$), the tail threshold parameter was set as fifteen ($th_{tail} = 15$), and the span parameter was set as one ($sp = 1$). Finally, the resolution bit size was set as eighteen ($b = 18$).

### B. MAWI TRACE

Six real-world network traffic traces from the MAWI Working Group Traffic Archive [54] are used to evaluate the performance of the proposed system. Those 15-minute long packet traces are originated from part of the 24h-long trace at MAWI's samplepoint B, F, and G. As shown in Table 4, the average number of distinct source IPv4 addresses ranged from 51.7 K to 5.07 M, while the total number of packets ranged from 3.5 M to 588.7 M approximately.

**TABLE 4.** Real-world network traffic traces from MAWI.

| MAWI Traffic Trace | Total # Distinct IPv4 Source Address (900 Sec.) | Total # Packets (900 Sec.) |
|---|---|---|
| Samplepoint-B 200302270000 | 51.7 K | 3.51 M |
| Samplepoint-F 200701011400 | 79.8 K | 6.94 M |
| Samplepoint-F 201501011400 | 5.074 M | 58.1 M |
| Ditl2019 201904091800 | 4.792 M | 98.8 M |
| Ditl2020 202004080800 | 315.3 K | 251.7 M |
| Samplepoint-G 202004081400 | 318.6 K | 588.7 M |

Simulations were performed to compute the cumulative probabilities of the relative error of the estimated entropy of the six traces given the use of both estimation schemes [23], [25]. The estimation process was confined to a small portion (30 seconds) of the original trace (15 minutes) and the estimation procedure was repeated 500 times using different hash parameters and table contents. Table 5 shows the cumulative probability of 3% and 5% relative errors of the estimated entropy when using the interpolation-based approach for LME and NECC algorithms, respectively. It is seen that when the estimation process is implemented using forty lookup tables ($kp = 40$) in parallel, the cumulative probability reaches 0.89 for a 3% relative error when using the LME scheme and 0.9 when using the NECC method. Moreover, the total memory space consumption ($sp = 1$, $th_{head} = 13$) is 640 K bytes, assuming that each table entry utilizes a 32-bit counter.

For each 15-minute-long packet trace, an estimated entropy value $\hat{H}_i$ and exact value $H_i$ were obtained using an observation time of 30 seconds. The corresponding MAPEs were then computed. The simulation process was repeated 20 times with different hashing parameters and table contents each time. Box-and-whisker plots were plotted for the MAPE values of the six traces given the use of the interpolation-based LME and NECC schemes, respectively. The corresponding results are presented in Figure 18 and Figure 19, respectively. It is seen that for all six traffic traces, the means of the box-and-whisker plots are less than 3% for both estimation schemes. Following the LME simulation results using only thirty lookup tables ($kp = 30$), the MAPE for the trace of *201501011400* and *201904091800* are 1.6% and 1.83%, respectively. The total memory space consumption ($sp = 1$,

**TABLE 5.** Cumulative probabilities of relative error for entropy estimation of 3% and 5% given use of the LME and NECC schemes ($\triangle t = 30$).

| MAWI Traffic Trace | Interpolation of Clifford and Cosma | | | | Interpolation of Ping Li | | | | Parameters mp=12 |
|---|---|---|---|---|---|---|---|---|---|
| | kp=30 | | kp=40 | | kp=30 | | kp=40 | | |
| | 3% Error | 5% Error | 3% Error | 5% Error | 3% Error | 5% Error | 3% Error | 5% Error | |
| samplepoint-B 200302270000 | 0.842 | 0.988 | 0.85 | 0.990 | 0.718 | 0.958 | 0.750 | 0.982 | b=14,sp=1, th_head=11 |
| samplepoint-F 200701011400 | 0.904 | 0.996 | 0.922 | 0.998 | 0.880 | 0.986 | 0.916 | 0.994 | b=14,sp=1, th_head=11 |
| samplepoint-F 201501011400 | 0.751 | 0.952 | 0.806 | 0.980 | 0.742 | 0.958 | 0.846 | 0.994 | b=18,sp=1, th_head=13 |
| ditl2019 201904091800 | 0.804 | 0.978 | 0.890 | 0.996 | 0.840 | 0.978 | 0.906 | 0.996 | b=18,sp=1, th_head=13 |
| ditl2020 202004080800 | 0.65 | 0.908 | 0.736 | 0.950 | 0.684 | 0.898 | 0.704 | 0.920 | b=14,sp=1, th_head=13 |
| samplepoint-G 202004081400 | 0.726 | 0.936 | 0.780 | 0.960 | 0.744 | 0.916 | 0.7556 | 0.9467 | b=14,sp=0, th_head=13 |

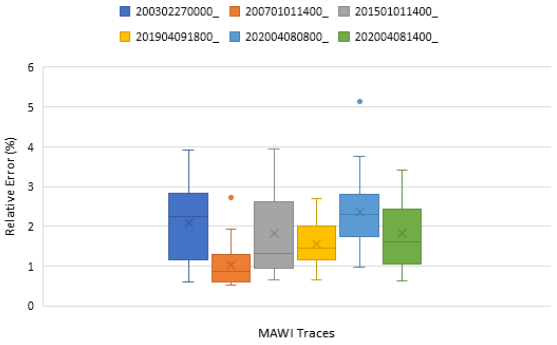



**FIGURE 18.** Box-and-whisker plot of relative error values for six MAWI traffic traces. The simulation is based on the LME of Clifford and Cosma with interpolation parameters of $mp = 12$ and $kp = 40$.

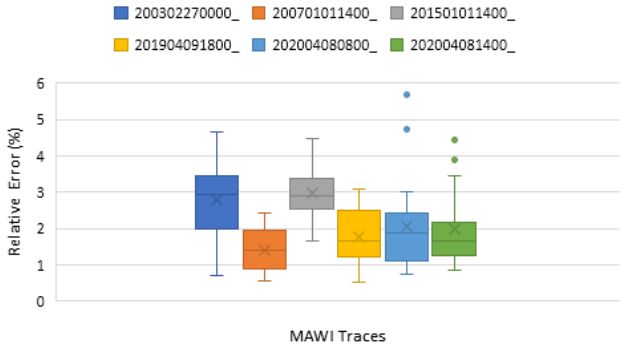



**FIGURE 19.** Box-and-whisker plot of relative error for the six MAWI traffic traces. The simulation is based on the NECC of Ping Li and Zhang ($\alpha = 0.999$) with interpolation parameters of $mp = 12$, and $kp = 40$.

$th_{head} = 13$) is 480 K bytes. Noted that, as illustrated in Table 4, the *201904091800* trace contains the highest number of distinct source IP addresses (197.9 K) in an average of 30-second observation time. Table 6 illustrates the mean absolute percentage errors of the LME estimation on MAWI traces in an observation time of 300 seconds. The errors are all less than 3% except for the trace of *202004081400*. Besides, the standard deviation for *202004081400* trace is higher than

**TABLE 6.** Mean absolute percentage error of the interpolation-based LME estimation on MAWI network traffic traces with observation time of 300 seconds ($\triangle t = 300$). The simulation is based on the configuration of $kp = 40$, $sp = 1$, $th_{head} = 13$, $b = 18$. The total memory consumed is 640 K bytes.

| MAWI Traffic Trace | Avg. # Distinct Source IP (300 Sec.) | Avg. # Packets (300Sec) | MAPE | S.D. | Hash Functions (mp) |
|---|---|---|---|---|---|
| 200302270000 | 25.7 K | 1.15 M | 1.06% | 0.85% | 12 |
| 200701011400 | 40.4 K | 2.32 M | 0.67% | 0.46% | 12 |
| 201501011400 | 1.64 M | 18.4 M | 2.85% | 0.88% | 12 |
| 201904091800 | 1.69 M | 28.2 M | 1.20% | 0.90% | 12 |
| 202004080800 | 105.1 K | 83.6 M | 2.81% | 1.29% | 24 |
| 202004081400 | 106.2 K | 196.2 M | 3.58% | 1.71% | 24 |

those of the other traces. This is mainly due to the excess number of packets (196.2 M) processed.

Noted that, as shown in Table 4, the whole traces (900 seconds) of *201501011400* and *201904091800* contain approximately five million distinct source IP addresses with total packet counts of 58.1 M and 98.8 M, respectively. Therefore, according to the simulation results shown in Figure 11, a higher resolution bit ($b$) needs to be applied in the table construction phase. Thus, as shown in Figure 20, with resolution bits of 22, the interpolation-based LME and NECC can estimate the empirical entropy of these two 900-second traces with less than 3% of mean absolute percentage error.

### C. CAIDA 2007 DDoS TRACE

The entropy estimation performance of the proposed architecture was further evaluated using the CAIDA 2007 DDoS dataset [56]. In particular, a one-hour-long packet trace was adopted from the MAWI Working Group Traffic Archive (MAWI 2019 DITL Trace) [54] and was merged as a background traffic with the CAIDA DDoS attack trace.

As shown in Figure 21, the synthetic trace contained two DDoS attacks, where these attacks were simulated simply by inserting the same DDoS attack records into the

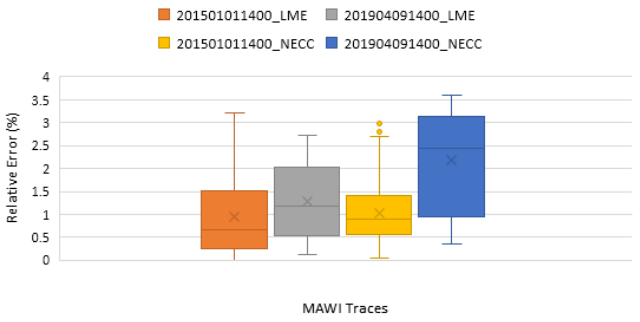



**FIGURE 20.** Box-and-whisker plot of relative error for the MAWI traces of 201501011400 and 201904091800. The simulation is based on the interpolation-based LME and NECC algorithms with the configuration of $kp = 40$, $sp = 1$, $th_{head} = 13$, $b = 22$. The observation time is 900 seconds ($\triangle t = 900$).

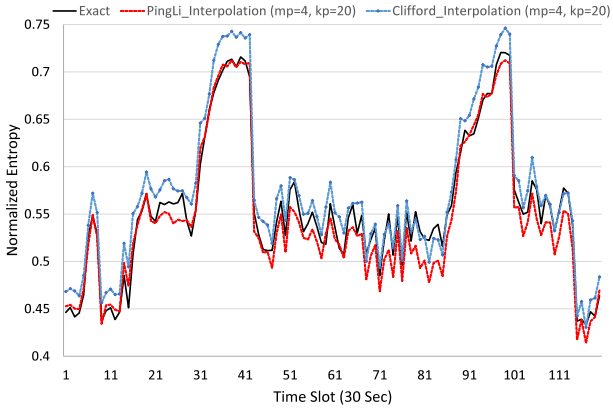



**FIGURE 21.** Estimated entropy values of source IP address obtained using interpolation-based LME and NECC algorithms. Note that the CAIDA 2007 DDoS dataset [56] is used with each time slot corresponds to an observation time of 30 seconds ($\triangle t = 30$).

background traffic twice. The entropy values were estimated using the interpolation-based LME [23] and NECC [25] schemes based on the source IP address and an observation time of 30 seconds in both cases. For both estimators, the interpolation scheme was implemented using parameter settings of $b = 18$, $sp = 0$, $th_{head} = 11$, and $th_{tail} = 15$. Moreover, the interpolation process was performed using twenty tables in parallel ($kp = 20$) with four hash functions for each table ($mp = 4$). A detailed inspection of Figure 21 shows that the MAPE values for the interpolation scheme of LME and NECC ($\alpha$=0.999) is 3.3% and 2.46%, respectively.

In order to compare the entropy values in different observation time slots, the entropy values are often normalized based on the distinct count or the total count [41] of the stream elements. Figure 21 presents the entropy normalized based on the exact distinct count for illustration purposes. As the cardinality estimation is outside the scope of this study, we refer the readers to the literature of Flajolet and Martin [57], [58] for the original algorithm. Furthermore, Kulkarni *et al.* [59] and Soto *et al.* [44] presented the estimation accelerator in FPGAs, and Ding *et al.* [45] introduced the practical implementations in the P4 programming language.

## VII. PRACTICAL IMPLEMENTATION

### A. FPGA

The practical feasibility of the proposed architecture was demonstrated by implementing it in the data plane of the NetFPGA-Plus [60] project in an UltraScale+ XCU200 FPGA consisting of 2, 160 blocks of 36 K-bit BRAMs. The total processing latency of the hardware design comprised ten clock cycles for frame parsing and key hashing (2 cycles), table lookup (1 cycle), and interpolation (7 cycles).

Figure 23 presents the FPGA system operation estimating the entropy values of five-tuple attributes by replaying the CAIDA DDoS 2007 network traffic trace [56] through a 100 Gbps network interface card. Each incoming packet was processed in a pipeline fashion using a 250 MHz AXI-Stream bus with 512-bit. A typical minimum-sized Ethernet frame consisted of a 12-byte inter-frame gap, a preamble of 8 bytes, a 14-byte frame header, a 46-byte payload, and a 4-byte CRC checksum of the wire. Thus, two cycles were required to process the 84-byte frame on the 512-bit AXI-Stream bus. The interpolation architecture was implemented using two sets of fifty tables in parallel ($kp = 100$ in total), where each table utilized two dual-port BRAMs with a size of 2 K $\times$ 36 bits. Accordingly, four hash lookups ($mp = 4$) were performed for each table within two clock cycles enabling 148, 809, 524 frames to be processed per second at a 100 Gbps wire speed. The Verilog HDL implementation is synthesized and the resource utilization is presented in Table 7.

**TABLE 7.** FPGA resource utilization.

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 283746 | 1182240 | 24.00 |
| LUTRAM | 18696 | 591840 | 3.16 |
| FF | 331493 | 2364480 | 14.02 |
| BRAM | 631.50 | 2160 | 29.24 |
| URAM | 11 | 960 | 1.15 |

Given the UltraScale+ BlockRAM's maximum clock frequency of 825 MHz [61], the proposed design was capable of processing up to twelve hash-lookups ($mp = 12$) within two clock cycles at a 100 Gbps wire speed. The theoretical processing throughput of the proposed pipelined design was thus 422.4 Gbps for minimum-sized Ethernet frames.

### B. P4

The proposed Entropy estimation scheme was also implemented in the data plane of a P4 switch using P4-16 programming language [62]. The interpolation parameters were set as $sp = 1$, $th_{head} = 10$, and $th_{tail} = 15$, and the estimation process was performed using twenty tables ($kp = 20$) in parallel and one hash function (CRC32) per table ($mp = 1$). Each entry stored a 32-bit fixed-point value consisting of a 12-bit fraction and a 20-bit integer. Figure 24 shows an excerpt of the P4 code for the interpolation operation in the behavioral model.

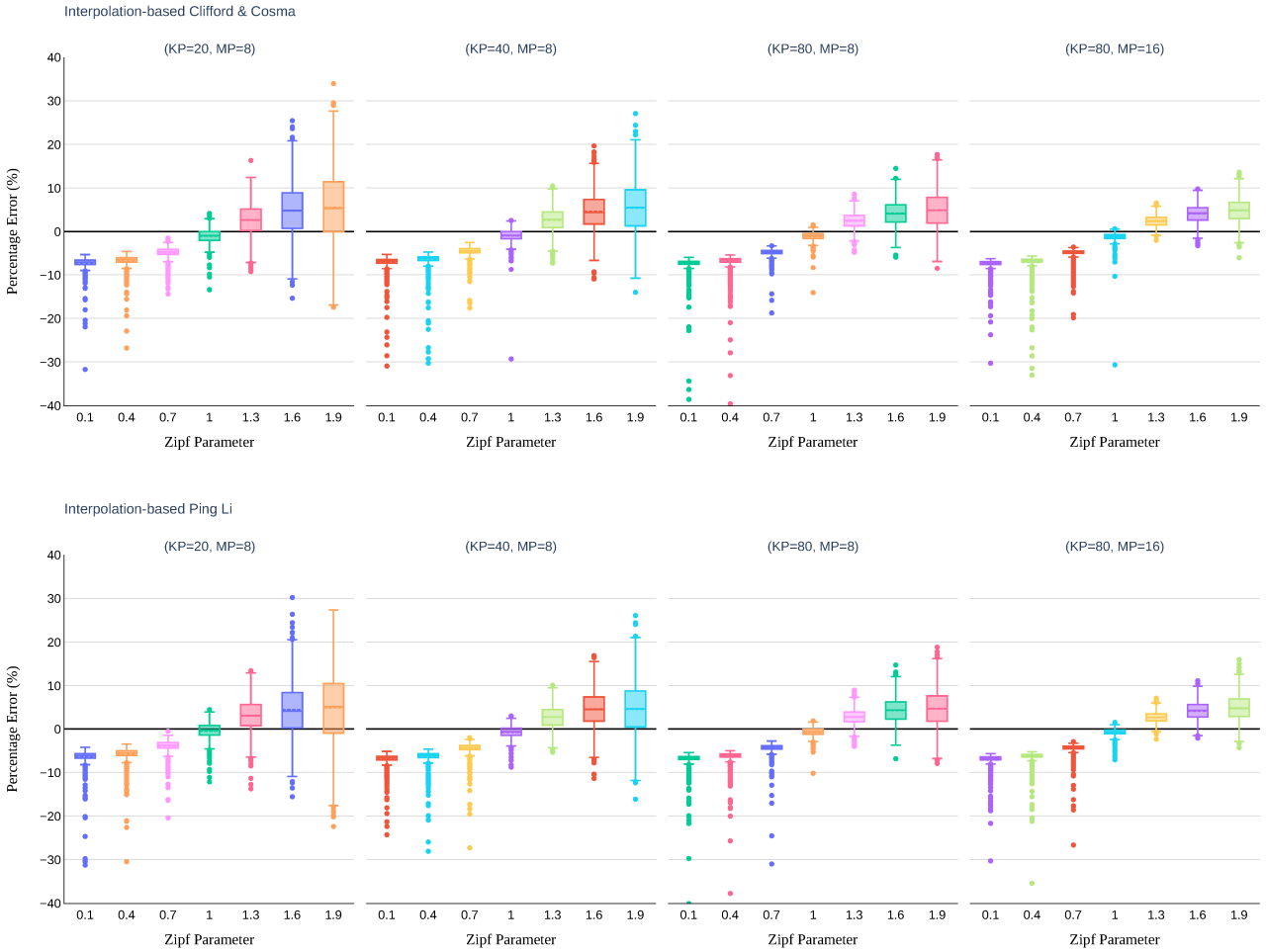



**FIGURE 22.** Box-and-whisker plot of the relative errors of entropy estimates obtained using interpolation-based LME of Clifford & Cosma (see upper panel) and NECC of Ping Li (see lower panel) schemes for Zipf parameters ranging from 0.1 to 2.0 and different numbers of tables (*kp*) and hash functions (*mp*). The synthetic data stream consists of 300 K items.

**FIGURE 23.** FPGA testbed used to evaluate entropy estimation performance with CAIDA 2007 DDoS trace [56].

Most of the programmable data plane architecture [63] can not support complex mathematical operations such as multiplication and division. The solution is to adopt the approximation techniques [64] of bit-shifting with adders and table lookups. In addition, more complex operations such as exponential and logarithmic can also be realized [45] based on the approximation techniques with binomial series expansion.

Assuming that a total of $m$ packets of *key* are observed within the range of $a_1$ and $a_2$ during the observation window $\bigtriangleup t$, and $X_j = hash(key_j)$, the estimation sketch data structure $Y'_j$ obtained using the interpolation process shown in Fig. 3 can be expressed as

$$\sum_{j=0}^{m-1} Y'_j = \sum_{j=0}^{m-1} (b_2 + \frac{(b_2 - b_1)}{(a_2 - a_1)}(X_j - a_1)) \tag{4}$$

$$= m \times b_2 + \frac{(b_2 - b_1)}{(a_2 - a_1)}(\sum X_j - m \times a_1). \tag{5}$$

Since the values of $a_1, b_1, a_2$, and $b_2$ are all known constant values, the multiplication step in of the interpolation process can be conducted in batch mode by using the host CPU in the control plane. In order to implement the proposed scheme in

```
460     #define FIXED_OFFSET 12
461     action interpolation_span_cal_k_0(bit<32> y1,bit<32> y2,bit<16>x1) {
462             table_lookup_result_k_0  =
463             (bit<64>)y1 +((bit<64>)(y2-y1)>>1)*((bit<64>)hash_result_k_0-(bit<64>)x1);
464     }
465     action interpolation_exp_cal_k_0(bit<32> y1,bit<32> y2,bit<16>x1,bit<8> shifter) {
466             table_lookup_result_k_0   =
467             (bit<64>)y1 +(((bit<64>)(y2-y1)*((((bit<64>)hash_result_k_0-(bit<64>)x1)<<FIXED_OFFSET)>>shifter))>>FIXED_OFFSET);
468     }
```

**FIGURE 24.** Excerpt of P4 codes for interpolation operation in behavioral model (BMv2).

**TABLE 8.** Average hardware resource consumption of Tofino switch.

| Exact Match Xbar | Hash Dist Unit | Hash Bit | Logical Table ID | SRAM | Stats ALU | TCAM | Pipeline Stages |
|---|---|---|---|---|---|---|---|
| 10.5% | 53.3% | 12.1% | 26.6% | 8.1% | 25.0% | 17.5% | 11 |

the Tofino Native Architecture (TNA) pipeline without the need for a multiplication operation, the implemented design utilizes additional register tables to accumulate the summation of the hash index $X_i$ and corresponding frequency count $m_i$ within the range of $a_i$ and $a_{i+1}$ for each incoming packet in the data plane. The average resource consumption of the final implementation ($kp = 20, mp = 1$) using eleven pipeline stages is shown in Table 8. Figure 25 presents a photograph of the P4 testbed. A total of 100 K minimum-sized Ethernet frames were generated at a rate of 100 Gbps by the Thor-400G-7S-1P test module in the ValkyrieCompact chassis (XENA Networks). Three different distributions of the IPv4 source addresses were configured, namely random, linear-increasing, and fixed. The relative error of the estimated entropy was found to be less than 11% in all three cases.

## VIII. DISCUSSION

### A. PACKET COUNT AND DISTINCT ITEM

Typically, the network traffic analysis adopts observation epochs of 30∼900 seconds [8], [41], [44]. Thus, based on the memory size ranging from 480 K to 640 K bytes, the proposed scheme can process traces of up to 98.8 million packets and handle approximately up to five million distinct items with less than 3% of mean absolute percentage error.

The proposed schemes can handle a more different number of distinct items for a traffic stream as the resolution bit value ($b$) increases. Hence the accuracy of the entropy estimates improves. A high-resolution setting extends the depth range of the *span* region. Thus, the parameters of $th_{head}$ and $sp$ need to be adjusted accordingly to meet the required estimation accuracy.

### B. IMPLEMENTATION FLEXIBILITY

The number of lookup tables ($kp$) and hash functions ($mp$) in the proposed interpolation-based scheme provides valuable flexibility in the system implementation to minimize the variance of the estimated entropy. Figure 22 presents box-and-whisker plots of the relative errors obtained using the interpolation-based LME of Clifford & Cosma (upper panel)

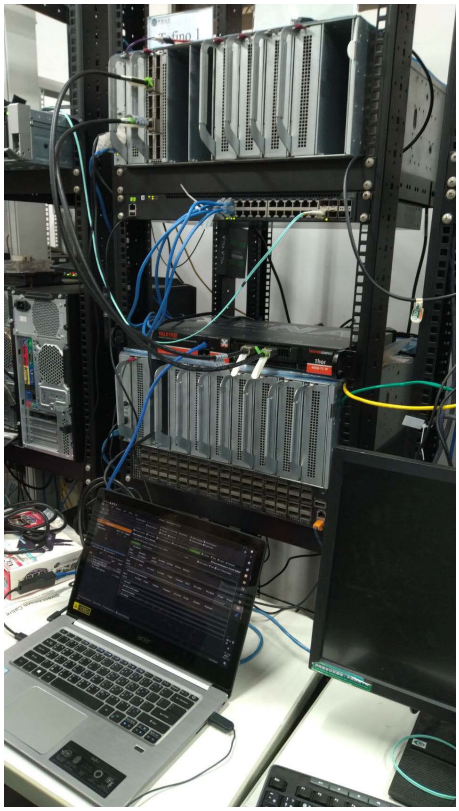


**FIGURE 25.** P4 testbed used to evaluate entropy estimation performance.

and NECC of Ping Li (lower panel) schemes for distributions with different Zipf parameters. Three different table configurations are adopted ($kp = 20, 40, 80$), and the number of hash functions used for each table is equal to eight ($mp = 8$) or sixteen ($mp = 16$).

The designer can deploy the proposed schemes with a fair number of tables and hash functions based on the available memory space and processing throughput requirement. Thus, compared to the existing hardware solutions [8], [41], [44], [45] we can implement the proposed scheme in the programmable data plane with P4 on the Tofino Native Architecture (TNA) and FPGA easily. Moreover, instead of conducting read-modify-write operations (2∼3 clock cycles) on the entire sketch memory [44], the read-only lookup procedure (1 clock cycle) of the proposed scheme provides a faster processing speed for the packet updates. Since the table lookup procedure can be performed in parallel, the latency is reduced. Hence the estimation process is favorable for

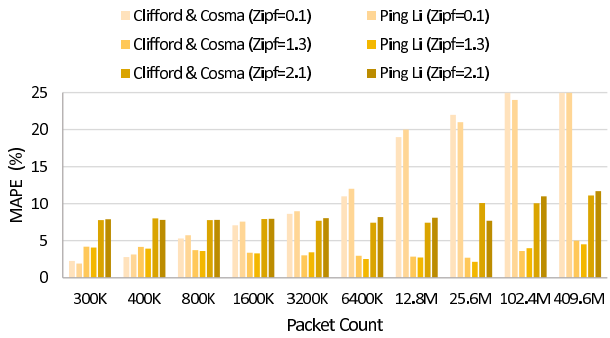



**FIGURE 26. Mean absolute percentage error for the interpolation-based LME of Clifford & Cosma and NECC of Ping Li with different numbers of packet counts processed and Zipf parameters (*Zipf* = 0.1, 1.3, 2.1). The simulation is based on the configuration of *kp* = 20, *mp* = 12, *sp* = 1, $th_{head}$ = 13, *b* = 18.**

high-speed network traffic (providing that sufficient memory space is available).

### C. LIMITATION

Increasing the number of tables can minimize the variance to a certain extent (*e.g.*, $kp > 80$) since the variance is inherently dependent on the magnitude of the interpolation error. For a given number of tables ($kp = 80$) shown in Figure 22, the variance can also be reduced by increasing the number of hash functions from eight ($mp = 8$) to sixteen ($mp = 16$). However, most on-chip embedded memories have only a limited number of read-ports. As a result, the lookup process must be conducted sequentially, causing a long latency delay. Consequently, the number of hash functions must be balanced in such a way as to meet the wire-speed processing requirement.

Figure 26 presents the mean absolute percentage error for the interpolation-based LME and NECC with different numbers of packet counts processed. The simulation is conducted based on three synthetic data streams of Zipf=0.1, 1.3, and 2.1 with parameters of $kp = 20$ and $mp = 12$. For a traffic distribution of Zipf=1.3, the interpolation-based LME and NECC schemes can process up to approximately 400 M packets with less than 5% of error. However, for an extreme case where the traffic is very uniformly distributed (Zipf=0.1), the error of the entropy estimation increases rapidly as the packet count grows. The reason is primarily due to the accumulation of interpolation errors originating from the table lookup process.

## IX. CONCLUSION AND FUTURE WORK

This paper has proposed a tabular interpolation scheme for estimating the empirical Shannon entropy of network traffic based on the stable random projection method. The existing entropy estimation methods, such as the Log-Mean Estimator (LME) [23] and the New Estimator of Compressed Counting (NECC) [25], [26], required complex computations. In contrast, the present study derives the required data structures using a simple table lookup process and a piece-wise linear interpolation technique. The total size of the lookup table is reduced by separating the table into three smaller tables in accordance with parameters of *sp*, $th_{head}$ and $th_{tail}$.

Notably, the parameters can be adjusted by the particular characteristics of the skewed alpha-stable distribution. The purpose is to correctly reproduce the distribution and achieve an acceptable tradeoff between the proposed scheme's memory consumption and the entropy estimates' accuracy.

The feasibility of the proposed architecture has been demonstrated using both real-world traffic traces and synthetic data streams. The scheme has additionally been evaluated, delivering the capability of processing network traffic at a 100 Gbps wire speed on a Xilinx U200 FPGA platform and a Tofino programmable P4 switch. In general, the results have shown that the proposed architecture is compatible with both the LME scheme [23] and the NECC method [25].

In addition, the simulation results have indicated that the proposed scheme can process traces of up to 98.8 million packets and handle up to five million distinct items with a mean relative error of less than 3%. The total memory space consumed is 480 K bytes ($kp = 30$) and 640 K bytes ($kp = 40$), respectively based on the configuration of ($mp = 12$, $sp = 1$, $th_{head} = 13$).

Since the primary entropy estimation involves only the lookup of read-only tables and the update of some sketch registers, the process has very low latency. Thus, a theoretical processing throughput in excess of 400 Gbps can be achieved given the latest advances in FPGA technology with a Block RAM frequency of 825 MHz. In future studies, we plan to optimize the proposed design further and deploy it in real-world network environments for traffic monitoring and anomaly detection applications.

## REFERENCES

[1] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 4th Quart., 2013.

[2] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2005, pp. 217–228, doi: 10.1145/1080091.1080118.

[3] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection," in *Proc. 8th ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, 2008, pp. 151–156, doi: 10.1145/1452520.1452539.

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection," *Pattern Recognit. Lett.*, vol. 51, pp. 1–7, Jan. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786551400244X

[5] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proc. DARPA Inf. Survivability Conf. Expo.*, vol. 1, Apr. 2003, pp. 303–314.

[6] M. H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita, "Detecting distributed denial of service attacks: Methods, tools and future directions," *Comput. J.*, vol. 57, no. 4, pp. 537–556, 2014. [Online]. Available: https://academic.oup.com/comjnl/article/57/4/537/407932

[7] S. Behal, K. Kumar, and M. Sachdeva, "Characterizing DDoS attacks and flash events: Review, research gaps and future directions," *Comput. Sci. Rev.*, vol. 25, pp. 101–114, Aug. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574013717300941

[8] H. Zhao, A. Lall, M. Ogihara, O. Spatscheck, J. Wang, and J. Xu, "A data streaming algorithm for estimating entropies of od flows," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas. (IMC)*, 2007, pp. 279–290, doi: 10.1145/1298306.1298345.

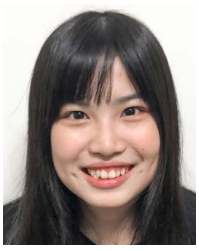
[9] A. K. Marnerides, A. Schaeffer-Filho, and A. Mauthe, "Traffic anomaly diagnosis in internet backbone networks: A survey," *Comput. Netw.*, vol. 73, pp. 224–243, Nov. 2014, doi: 10.1016/j.comnet.2014.08.007.
[10] *IEEE 802.3 50 Gb/s, 100 Gb/s, and 200 Gb/s Ethernet Task Force*. Accessed: Oct. 25, 2021. [Online]. Available: http://www.ieee802.org/3/cd/
[11] T. Taleb, A. Ksentini, and R. Jantti, "'Anything as a service' for 5G mobile systems," *IEEE Netw.*, vol. 30, no. 6, pp. 84–91, Nov. 2016.
[12] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51691–51713, 2019.
[13] S. S. Kim and A. L. N. Reddy, "Statistical techniques for detecting traffic anomalies through packet header data," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 562–575, Jun. 2008, doi: 10.1109/TNET.2007.902685.
[14] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proc. 6th ACM SIGCOMM Internet Meas. (IMC)*, 2006, pp. 159–164. [Online]. Available: http://portal.acm.org/citation.cfm?id=1177101
[15] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proc. 6th ACM SIGCOMM Internet Meas. (IMC)*, 2006, pp. 165–176, doi: 10.1145/1177080.1177102.
[16] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 20–29, doi: 10.1145/237814.237823.
[17] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," in *Proc. Joint Int. Conf. Meas. Modeling Comput. Syst. SIGMETRICS/Perform.*, 2006, pp. 145–156. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.1684
[18] A. Chakrabarti, K. Do Ba, and S. Muthukrishnan, "Estimating entropy and entropy norm on data streams," in *Proc. 23rd Annu. Conf. Theor. Aspects Comput. Sci. (STACS)*. Berlin, Germany: Springer, 2006, pp. 196–205, doi: 10.1007/11672142_15.
[19] N. J. Harvey, J. Nelson, and K. Onak, "Sketching and streaming entropy via approximation theory," in *Proc. 49th Annu. IEEE Symp. Found. Comput. Sci.* Philadelphia, PA, USA, Oct. 2008, pp. 489–498. [Online]. Available: http://ieeexplore.ieee.org/document/4690982/
[20] P. Indyk, "Stable distributions, pseudorandom generators, embeddings and data stream computation," in *Proc. 41st Annu. Symp. Found. Comput. Sci.*, 2000, pp. 189–197.
[21] P. Li, "A very efficient scheme for estimating entropy of data streams using compressed counting," 2008, *arXiv:0808.1771*.
[22] P. Li, "Estimating entropy of data streams using compressed counting," 2009, *arXiv:0910.1495*.
[23] P. Clifford and I. Cosma, "A simple sketching algorithm for entropy estimation over streaming data," in *Proc. Artif. Intell. Statist.*, 2013, pp. 196–206.
[24] G. Cormode, "Stable distributions for stream computations: It's as easy as 0,1,2," in *Proc. Workshop Manag. Process. Data Streams*, 2003, pp. 1–2.
[25] P. Li and C.-H. Zhang, "A new algorithm for compressed counting with applications in Shannon entropy estimation in dynamic data," in *Proc. 24th Annu. Conf. Learn. Theory*, Dec. 2011, pp. 477–496. [Online]. Available: http://proceedings.mlr.press/v19/li11a.html
[26] P. Li, "Compressed counting," in *Proc. 20th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2009, pp. 412–421. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496770.1496816
[27] *Computing on Data Streams, SRC Technical Note. 1998-011*. Accessed: Jul. 3, 2021. [Online]. Available: https://www.hpl.hp.com/techreports/Compaq-DEC/SRC-TN-1998-011.pdf
[28] S. Muthukrishnan, "Data streams: Algorithms and applications," *Found. Trends Theor. Comput. Sci.*, vol. 1, no. 2, pp. 117–236, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1166410
[29] V. M. Zolotarev, *One-Dimensional Stable Distributions* (Translations of Mathematical Monographs), vol. 65. Providence, RI, USA: American Mathematical Society, 1986.
[30] B. Mandelbrot, "The Pareto–Lévy law and the distribution of income," *Int. Econ. Rev.*, vol. 1, no. 2, pp. 79–106, 1960. [Online]. Available: https://www.jstor.org/stable/2525289
[31] B. Mandelbrot, "New methods in statistical economics," *J. Political Economy*, vol. 71, no. 5, pp. 421–440, Oct. 1963. [Online]. Available: https://www.jstor.org/stable/1829014
[32] J. P. Nolan, "Financial modeling with heavy-tailed stable distributions," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 6, no. 1, pp. 45–55, Jan. 2014, doi: 10.1002/wics.1286.
[33] K. Mcdonald and S. Sundaram, "Asset health monitoring using stable distributions for heavy-tailed data," WO Patent 2011160943 A1, Dec. 29, 2011. [Online]. Available: http://www.google.ch/patents/WO2011160943A1
[34] F. Simmross-Wattenberg, J. I. Asensio-Perez, P. Casaseca-de-la-Higuera, M. Martin-Fernandez, I. A. Dimitriadis, and C. Alberola-Lopez, "Anomaly detection in network traffic based on statistical inference and alpha-stable modeling," *IEEE Trans. Depend. Sec. Comput.*, vol. 8, no. 4, pp. 494–509, Jul. 2011.
[35] Y. Zhou, R. Li, Z. Zhao, X. Zhou, and H. Zhang, "On the alpha-stable distribution of base stations in cellular networks," *IEEE Commun. Lett.*, vol. 19, no. 10, pp. 1750–1753, Oct. 2015.
[36] J. M. Chambers, C. L. Mallows, and B. Stuck, "A method for simulating stable random variables," *J. Amer. Stat. Assoc.*, vol. 71, pp. 340–344, 1976.
[37] V. M. Zolotarev, *On the Representation of Stable Laws by Integrals*, vol. 71. Moscow, Russia: Nauka, 1964, pp. 46–50.
[38] R. Weron, "On the chambers-mallows-stuck method for simulating skewed stable random variables," *Statist. Probab. Lett.*, vol. 28, no. 2, pp. 165–171, Jun. 1996. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0167715295001131
[39] S. Borak, W. Härdle, and R. Weron, "Stable distributions," in *Statistical Tools for Finance and Insurance*. Cham, Switzerland: Springer, 2005, pp. 21–44.
[40] G. Cormode and P. Indyk, "Stable distributions in streaming computations," in *Data Stream Management: Processing High-Speed Data Streams* (Data-Centric Systems and Applications), M. Garofalakis, J. Gehrke, and R. Rastogi, Eds. Berlin, Germany: Springer, 2016, pp. 283–300, doi: 10.1007/978-3-540-28608-0_14.
[41] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 1, pp. 145–156, Jun. 2006, doi: 10.1145/1140103.1140295.
[42] A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Offloading real-time DDoS attack detection to programmable data planes," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, 2019, p. 9.
[43] A. D. S. Ilha, A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A fully in-network, P4-based approach for real-time DDoS attack detection and mitigation," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 3121–3139, Sep. 2021.
[44] J. E. Soto, P. Ubisse, Y. Fernandez, C. Hernandez, and M. Figueroa, "A high-throughput hardware accelerator for network entropy estimation using sketches," *IEEE Access*, vol. 9, pp. 85823–85838, 2021.
[45] D. Ding, M. Savi, and D. Siracusa, "Tracking normalized network traffic entropy to detect DDoS attacks in p4," 2021, *arXiv:2104.05117*.
[46] L. Bhuvanagiri and S. Ganguly, "Estimating entropy over data streams," in *Proc. 14th Conf. Annu. Eur. Symp.*, vol. 14. London, U.K.: Springer, 2006, pp. 148–159. [Online]. Available: http://dl.acm.org/citation.cfm?id=1276191.1276207
[47] G. Cormode and M. Muthukrishnan, "Approximating data with the count-min sketch," *IEEE Softw.*, vol. 29, no. 1, pp. 64–69, Jan. 2012.
[48] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Theor. Comput. Sci.*, vol. 312, no. 1, pp. 3–15, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/B6V1G-496P0C0-1/2/364d9cc8e25d5f99e316799e8d918e40
[49] A. Chakrabarti, G. Cormode, and A. McGregor, "A near-optimal algorithm for computing the entropy of a stream," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 328–335. http://dl.acm.org/citation.cfm?id=1283383.1283418
[50] A. Chakrabarti, G. Cormode, and A. Mcgregor, "A near-optimal algorithm for estimating the entropy of a stream," *ACM Trans. Algorithms*, vol. 6, no. 3, p. 51:1–51:21, 2010, doi: 10.1145/1798596.1798604.
[51] N. J. A. Harvey, J. Nelson, and K. Onak, "Streaming algorithms for estimating entropy," in *Proc. IEEE Inf. Theory Workshop*, May 2008, pp. 227–231.
[52] V. Bartos and M. Zádnik, "Hardware precomputation of entropy for anomaly detection," in *Proc. ACM/IEEE 7th Symp. Archit. Netw. Commun. Syst.*, Oct. 2011, pp. 219–220, doi: 10.1109/ANCS.2011.41.
[53] P. Arbenz and W. Guevara-Alarcón, "Piecewise linear approximation of empirical distributions under a Wasserstein distance constraint," *J. Stat. Comput. Simul.*, vol. 88, no. 16, pp. 3193–3216, Nov. 2018, doi: 10.1080/00949655.2018.1506454.

[54] K. Cho. (2020). *MAWI Working Group Traffic Archive*. [Online]. Available: https://mawi.wide.ad.jp/mawi/
[55] *MassDAL Public Code Bank:Sketches, Frequent Items, Changes (Deltoids)*. Accessed: Jun. 4, 2020. [Online]. Available: https://www.cs.rutgers.edu/~emuthu/massdalsketches.zip
[56] CAIDA. *The CAIDA UCSD DDoS Attack 2007 Dataset*. Library Catalog. Accessed: Nov. 18, 2015. [Online]. Available: https://www.caida.org/data/passive/ddos-20070804-dataset.xml
[57] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comput. Syst. Sci.*, vol. 31, no. 2, pp. 182–209, Oct. 1985. [Online]. Available: http://portal.acm.org/citation.cfm?id=5215
[58] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "HyperLogLog: The analysis of a near-optimal cardinality estimation algorithm," in *Proc. Conf. Anal. Algorithms (AofA)*. DMTCS, Jan. 2007. [Online]. Available: https://dmtcs.episciences.org/3545, doi: 10.46298/dmtcs.3545.
[59] A. Kulkarni, M. Chiosa, T. B. Preußer, K. Kara, D. Sidler, and G. Alonso, "HyperLogLog sketch acceleration on FPGA," 2020, *arXiv:2005.13332*.
[60] T. Yuta. (Dec. 2021). *NetFPGA/NetFPGA-PLUS*. [Online]. Available: https://github.com/NetFPGA/NetFPGA-PLUS
[61] *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics*. Accessed: Jul. 3, 2021. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds923-virtex-ultrascale-plus.pdf
[62] ONF. (Oct. 2019). *Next-Gen SDN Tutorial (Advanced), Open Networking Foundation*. [Online]. Available: https://github.com/opennetworkinglab/ngsdn-tutorial
[63] N. K. Sharma, A. Kaufmann, T. Anderson, C. Kim, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the power of flexible packet processing for network resource allocation," in *Proc. 14th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 67–82.
[64] P. Cui, H. Pan, Z. Li, J. Wu, S. Zhang, X. Yang, H. Guan, and G. Xie, "NetFC: Enabling accurate floating-point arithmetic on programmable switches," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–11.

**YU-KUEN LAI** (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from North Carolina State University, Raleigh, NC, USA, in 1997 and 2006, respectively. From 1997 to 2002, he was a Senior ASIC Design Engineer with Delta Networks, Inc., and Applied Micro Circuit Corporation (AMCC), Research Triangle Park, NC, USA. He is currently an Associate Professor with the Electrical Engineering Department, Chung Yuan Christian University (CYCU), Chung-li, Taiwan. He is also the Director of the Information Technology Division, C.Y. Chang Memorial Library, CYCU. His research interests include software-defined networking, streaming data processing, network traffic analysis, FPGA systems design, and computer network security. He was a recipient of the CYCU Distinguished Teaching Award, in 2011, and the CYCU Outstanding Teaching Award, in 2017. He served as the Electronic Communication Officer for the IEEE Taipei Section, in 2015.

**CHENG-LIN TSAI** received the M.S. degree in electrical engineering from Chung Yuan Christian University, Chung-li, Taiwan, in January 2022. He is currently a Network System Design Engineer at Senao Networks, Inc. His research interests include software-defined networking and computer systems design.

**CHENG-HAN CHUANG** received the M.S. degree in electrical engineering from Chung Yuan Christian University (CYCU), Chung-li, Taiwan, in August 2022. He is currently a Research Assistant at the Computer and Network Systems Research Laboratory, CYCU. His research interests include low-latency FPGA systems design and computer network security.



**XIU-WEN KU** received the M.S. degree in electrical engineering from Chung Yuan Christian University, Chung-li, Taiwan, in August 2022. Her research interests include FPGA systems design and computer networks.

**JIM HAO CHEN** is currently the Associate Director of the International Center for Advanced Internet Research (iCAIR), Northwestern University, where he is also responsible for the center's research infrastructure design and engineering. Before joining iCAIR, he was a Coordinator for technology testbeds at Northwestern. He leads multiple projects focus on the design and development of high performance platforms for advanced network systems and advanced network applications. His research interests include include 100G network exchanges and data movement, high performance digital media networks, high resolution 2D/3D digital media streaming over networks, international collaboration virtual environments for science, programmable network testbeds, science cloud networks, and virtual science environments.