Towards a Universal Sketch for Origin-Destination Network Measurements

Haiquan (Chuck) Zhao[1], Nan Hua[1], Ashwin Lall[2],
Ping Li[3], Jia Wang[4], and Jun (Jim) Xu[5]

[1] Georgia Tech {chz,nanhua,jx}@cc.gatech.edu,
[2] Denison University lalla@denison.edu
[3] Cornell University pingli@cornell.edu
[4] AT&T Research jiawang@research.att.com

Abstract. Despite its importance in today's Internet, network measurement was not an integral part of the original Internet architecture, i.e., there was (and still is) little native support for many essential measurement tasks. Targeting the inadequacy of counting/accounting capabilities of existing routers, many data streaming and sketching techniques have been proposed to estimate the important statistics of traffic going through a network link. Most of these techniques are, however, developed to track one specific statistic and/or answer a specific type of query. Since there are a large number of such statistics and queries of interest, it is very difficult, if not impossible, for network vendors and operators to implement and deploy data streaming/sketching solutions for all of them, due to router resource (memory, CPU, bus bandwidth, etc.) constraints.
In this paper, we propose a general-purpose solution that can not only answer a wide range of queries, but also be able to answer types of queries that were not known *a priori*. In particular, we introduce the use of the Conditional Random Sampling (CRS) sketch data structure for succinctly capturing network traffic data between a set of nodes in the network. This sketch is the first step towards a "universal" sketch data structure in the sense that it is not tied to measurement of a single quantity. We show that the CRS sketch can compute unbiased estimates for any linear summary statistic in the intersection of a pair of traffic streams, e.g., traffic and flow matrix information, flow counts, and entropy. We present detailed experiments, using data collected at a tier-1 ISP, that show that our sketch is capable of estimating this wide range of statistics with fairly high accuracy.

1 Introduction

As the Internet continues to grow and evolve, network traffic measurement is essential to Internet service providers (ISPs) and application service providers (e.g., Google, Yahoo, Akamai) for a variety of network operation tasks and business decision-making. Despite its importance in today's Internet, network measurement was not an integral part of the original Internet architecture, i.e., there was (and still is) little native support for many essential measurement tasks. Besides some basic SNMP counters (e.g., total packet and byte counts across a link) and simple flow collection facilities (e.g., Cisco Netflow), today's routers still lack sophisticated measurement functionality. Such limitations have

constrained many network management tasks, and forced significant research and engineering efforts to be expended on finding creative solutions. In particular, targeting the aforementioned inadequacy of counting/accounting capabilities of existing routers, many data streaming and sketching techniques (e.g., [14, 15, 33]) have been proposed to estimate the important statistics of, and to approximately answer various queries concerning, traffic going through a network link.

Most existing data streaming and sketching techniques are developed to track one specific statistic (e.g., second moment [1]) and/or answer a specific type of query (e.g., existence and identities of elephants [9, 16]). Since there are a large number of such statistics and queries of interest, it is very difficult, if not impossible, for network vendors and operators to implement and deploy data streaming/sketching solutions for all of them, due to router resource (memory, CPU, bus bandwidth, etc.) constraints. We believe this partly explains the lack of commercial success of these solutions. An emerging solution to this problem is so-called universal data sketches, each of which is a single versatile data structure that can track several different types of statistics and/or answer a wide range of queries, including those that were not known *a priori*, simultaneously. Overall a universal sketch consumes much less resources than individual sketches for tracking/answering all these statistics/queries combined. A conceivable tradeoff here is that a universal sketch might not provide as accurate answers to a type of query as data sketches designed specifically for it, given the same resource consumption.

In this paper we introduce the Conditional Random Sampling (CRS) sketch [19–21], a universal sketch for succinctly summarizing network traffic data. The CRS sketch, if deployed across multiple routers, can perform a type of "coordinated" sampling wherein the flows sampled at different locations can be highly correlated. This permits the accurate computation of statistics of the *intersection* of the data at both locations (i.e., origin-destination measurements) using the sketches at each. Moreover, CRS is not tied to measurement of a single quantity. We show that it is capable of computing any linear summary statistic of the intersection of streams. Examples of these statistics include (but are not limited to) computation of the traffic matrix, counting numbers of distinct flows, entropy estimation, and computation of higher moments for detection of large changes in the traffic distribution. A key contribution of this paper is to show that the estimator for the sketch is unbiased, settling an open problem raised implicitly in [21].

We perform a thorough analysis of CRS on packet trace data collected from a tier-1 ISP at the core of the Internet. Whereas CRS has been studied before [19, 21] for natural language applications, we had to tune parameters and make non-trivial adaptations to make it work on networking data. We were able to show that for most applications, such as traffic matrix, entropy estimation, etc., the sketch performed remarkably well while using very little memory. For most of the common measurement statistics that we estimated, it gave low measurement error while using memory comparable with a specialized algorithm for any single statistic.

The rest of this paper is organized as follows. Section 2 defines the problem that we solve in this paper. Section 3 presents the CRS sketch and how we apply it on network traffic measurement. We evaluate the CRS sketch using data collected from a tier-1

ISP in Section 4. Finally, we discuss related work in Section 5 before concluding in Section 6.

2 Problem Statement

Throughout this paper, we assume that there are ℓ spatially-separated nodes (say, ingress and egress nodes in a network). Each node encounters a stream of tuples of the form $\langle i,c\rangle$, where i is an element from a universe of size n (e.g., in the case of IP flows, $n = 2^{96}$, where $96 = 32+32+16+16$ is the length of the flow label consisting of source and destination IP addresses and port numbers) and c is the increment to the count for this element (e.g., the packet size in bytes). The classic streaming problem is to compute functions of the frequency distribution of the streams at individual nodes. In this paper, we expand this definition to answering queries for arbitrary *pairs* of nodes based on the sketches at the ℓ nodes. Note that this is a much harder problem since we must be able to answer $\binom{\ell}{2} = \Theta(\ell^2)$ queries from just ℓ sketches.

While previous work in data streaming has focused on sketches for a specific query/statistic, our goal is to have a single sketch that can be used to compute unbiased approximations for multiple types of queries performed on the intersection of traffic seen at any pair of nodes. In particular, we aim to compute any linear summary statistic, i.e., any function of the form $\sum_x g(x)$ where the sum is over the flow sizes in the OD-flows between pairs of streams. Examples of the types of queries that we expect to answer include:

1. The F_1 count between pairs of distributed streams, or simply put, the amount of traffic between two nodes in the network. Using the sketch proposed in this paper, we can estimate the traffic between all pairs of nodes and hence construct the traffic matrix [12].
2. The F_0 count is simply the number of distinct flows that are in the intersection of the two streams. This quantity is useful for network provisioners to determine the *number* of flows between a source-destination pair, rather than the aggregate traffic.
3. Prior work [33] has looked at the computation of the entropy norm of traffic between origin-destination pairs. This is defined as $\sum x \ln x$, where the sum is over the all the flows in OD-flow between a given pair of nodes. It is also often useful to compute the entropy, which is defined to be $\log L - (1/L)\sum x \ln x$, where L is the F_1 count of the OD-flow.
4. The higher-order moments $F_2, F_3, \ldots$ (where F_p is defined as $\sum_x x^p$) are also useful for detecting skewness and major shifts in the traffic distribution.

3 Conditional Random Sampling

We propose the use of Conditional Random Sampling [19–21] to sketch network data at ingress and egress nodes in the network. The CRS sketch has been shown to be capable of computing arbitrary linear summary statistics of the intersection of pairs of streams, and we evaluate its performance on networking data. Additionally, we show in Section 3.3 that it is possible to extract an *unbiased* estimate of any linear statistic of the intersection, a fact that was unknown to the authors of [21].

Algorithm 1 CRS Sketch

```
initialize shared permutation function π
initialize max-heap heap
for each packet p in the stream do
    let flow := π(flow_id(p))
    if flow ≤ heap.maxElement then
        if flow is in heap then
            update the record of flow in heap
        else
            if size(heap) ≥ k then
                evict the max element
            end if
            insert flow into the heap
        end if
    end if
end for
```

3.1 The CRS Algorithm

The CRS algorithm is executed independently at any set of ingress and egress points in the network (more generally at any router or interface of interest), and produces a sketch of the local stream. There is no requirement of universal deployment (e.g., allowing it to be run on only a subset of routers) and is capable of answering queries for any pair of participating routers.

At the heart of CRS lies a random permutation of the flow records in the streams. By ensuring that all the measurement interfaces all use the same permutation, we are able to perform "coordinated" sampling to accurately estimate statistics of the intersection of multiple streams. More precisely, each copy of the CRS sketch is maintained by retaining the k flows with smallest permuted flow IDs in a balanced max-heap data structure that is pruned to have at most k items in it at any time.[5] Here, k is the maximum number of flow records that can be retained in the sketch. Flows in the sketch are updated, and whenever a new flow enters the sketch (by replacing a flow with a larger permitted flow ID) a new record is created for it. See Algorithm 1 for details. By the monotonicity of the *min* operation, we are guaranteed to have complete counts for all the records in the sketch at the termination of the stream.

The CRS sketch strictly controls its memory consumption by guaranteeing that it never maintains more than k records. In contrast, most other sampling mechanisms such as uniform sampling, flow sampling [7, 8], sample-and-hold [9], etc., can only provide probabilistic guarantees on how much space they will consume.

After the sketches are computed, they can be shipped (due to their small size) to a central location for storage. To compute any linear summary statistic of an arbitrary pair of sketches, we compute the sum of the statistic over the set of flows that are in the intersection of the two sketches, excluding the max element of the two sketches. This sum is a sample of the desired statistic, and we can compute an estimate by reverting by

[5] Each sketch can use a different k.

Algorithm 2 CRS Intersection

1: **INPUT:** A pair of CRS sketches c_1 and c_2, function g
2: $Z_1 :=$ largest ID in c_1
3: $Z_2 :=$ largest ID in c_2
4: $Z := \min(Z_1, Z_2)$
5: $n :=$ total number of possible flows
6: $sum := 0$
7: **for** each record f in both c_1 and c_2, and $f.ID < Z$ **do**
8: $\quad sum := sum + g(f.size)$
9: **end for**
10: **OUTPUT** $sum \times n/(Z-1)$

factor $n/(Z-1)$, where n is the cardinality of the space of flow IDs and Z is the minimum of the largest permuted ID of the two sketches. This is illustrated is Algorithm 2. In Section 3.3 we show that this factor is precisely the one needed to get an unbiased estimate for any linear summary statistic. We next discuss some implementation issues for the CRS sketch.

3.2 Implementation Details

Since a random permutation is expensive to implement, in an actual implementation we would use a hash function with a large range and maintain the k flows with the smallest hash signature. Such approximation has also been used in [2]. The range can be smaller than the domain of flow IDs but should be much larger than F_0^2 so that the probability of hash collisions is negligible. In Algorithm 2 we replace n with the size of the range.

Since this sketch has to work on streaming packet data, it is critical that the update time per packet should be extremely fast: inter-packet arrival times may be as small as 10–20 nanoseconds. The CRS sketch is extremely well-suited for updating at high speeds. For almost all the packets in the stream, the only processing necessary is the computation of two hashes: one to compute the hash of the flow ID, and another to perform a hash table lookup of the flow in the table (in a real implementation, the heap will contain pointers to the flows in the hash table). Since the hashes do not need to be of cryptographic strength, low-complexity hash functions such as Carter-Wegman's H_3 hashes [3] can be used. Alternatively, hardware support for hash functions are also available [26]. For a very small subset of updates, specifically when the first k distinct flows are seen in the stream and when a flow is replaced in the heap (lines 9-12 in Algorithm 1), we have to perform heap operations that cost $O(\log k)$ operations for a balanced heap (e.g., Red-Black tree). The former case happens precisely k times in the stream and the latter can be shown experimentally to be infrequent, and hence these slower operations do not significantly affect the run-time performance of CRS.

3.3 Unbiasedness of CRS

The CRS algorithm described in [21] can handle multiple streams and any linear summary statistic over all the streams. The flow size distribution of each stream is viewed

as a vector over the space of all flow IDs. Consider a matrix of m row vectors of length n, denoted $\{x_{ij}\}$. If we construct a CRS sketch for each row, then we can get an estimate for any linear statistic of the form $\sum_{j=1}^{n} g(x_{1,j},\cdots,x_{m,j})$. The CRS algorithm we described in Section 3.1 is simply a special case, with $m=2$, $g(0,*)=0, g(*,0)=0$, and the assumption that $x_{1,j}=x_{2,j}$ whenever $x_{1,j}\neq 0, x_{2,j}\neq 0$.

While [21] only claimed CRS to be unbiased for F_0 estimation for one stream, we discovered that CRS is strictly unbiased in the general case. This follows as a corollary of the following theorem.

Theorem 1. *Consider m rows of n balls each. The balls are either red or white, and we denote this as an $m\times n$ $\{0,1\}$-valued matrix $\{a_{i,j}\}$ where 1 stands for red. Let $k_i\geq 2, \forall 1\leq i\leq m$ be integer constants.*

Let $\pi:[n]\rightarrow[n]$ be a random permutation that is applied to the column indices. We will collect the columns from the left until for at least one row i we have collected k_i red balls. Let Z_i be the index of the k_i^{th} red ball in the i^{th} row of the permuted matrix, i.e., $a_{i,\pi^{-1}(Z_i)}=1$ and $\sum_{j=1}^{Z_i} a_{i,\pi^{-1}(j)}=k_i$. If there are less than k_i red balls in the i^{th} row, let $Z_i=n+1$. Let $Z=\min_i Z_i$. Then the first $Z-1$ columns are an unbiased sample of all the columns, i.e.,

$$\mathrm{E}_\pi\left[\frac{1_{\{1\leq\pi(j_0)\leq Z-1\}}}{Z-1}\right]=\frac{1}{n}, \forall 1\leq j_0\leq n.$$

Remark: $\mathrm{E}_\pi\left[1_{\{1\leq\pi(j_1)\leq Z-1\}}\right]=\mathrm{E}_\pi\left[1_{\{1\leq\pi(j_2)\leq Z-1\}}\right]$ is not true in general.

Proof. Let $h(m,n,a,k,j_0)\equiv \mathrm{E}_\pi\left[\frac{1_{\{1\leq\pi(j_0)\leq Z-1\}}}{Z-1}\right]$. We want to prove by induction on n that $h(m,n,a,k,j_0)=\frac{1}{n}$ for any m,a,k,j_0.

When $n=1$, since we require $k_i\geq 2$, we have $Z_i=2,\forall i$, so $Z=2$, and the statement is true.

Now assume it is true for $n-1$, and we prove it for n.

We observe that if k_i is larger than the number of red balls on the i^{th} row, i.e., $k_i>\sum_{j=1}^{n} a_{i,j}$, then $Z_i=n+1$ always, and Z_i has no effect on Z, so we can remove the i^{th} row from a. If there is only one row left, and $k_1>\sum_j a_{1,j}$, then $Z=Z_1=n+1$, and the equation holds true. So we only need to prove for the cases where $k_i\leq\sum_{j=1}^{n} a_{i,j}, \forall i$.

We write the expectation conditioned on which column is mapped to n:

$$\begin{aligned} h(m,n,a,k,j_0) &= \mathrm{E}\left[\mathrm{E}\left[\frac{1_{\{1\leq\pi(j_0)\leq Z-1\}}}{Z-1}|\pi^{-1}(n)\right]\right] \\ &= \sum_{j'=1}^{n}\mathrm{E}\left[\frac{1_{\{1\leq\pi(j_0)\leq Z-1\}}}{Z-1}|\pi^{-1}(n)=j'\right]\Pr[\pi^{-1}(n)=j']. \end{aligned}$$

For $j'=j_0$, i.e., $\pi(j_0)=n$, we get a zero term. This is because we assumed $k_i\leq\sum_{j=1}^{n} a_{i,j}$, so $Z_i\leq n$, therefore $Z-1\leq n-1$, so $1_{\{1\leq\pi(j_0)\leq Z-1\}}=0$.

For $j'\neq j_0$, i.e., $\pi(j_0)\neq n$: Let us consider the same algorithm with the last column removed, i.e., with parameter m,n',a',k,j_0' where $n'=n-1$, a' is matrix a with the j'^{th} column removed, and j_0' is the position of j_0 among $\{1,...,j'-1,j'+1,...,n-1\}$. Each permutation π with $\pi(j')=n$ has a one-to-one correspondence with a permutation π'. We will argue that $Z_i=Z_i'$, given our assumption that $k_i\leq\sum_{j=1}^{n} a_{i,j}$.

– If $a_{i,j'} = 0$, i.e., it's a white ball; Or if $a_{i,j'} = 1$ and $k_i < \sum_{j=1}^{n} a_{i,j}$, i.e., it's a red ball but there are at least k_i red balls left in the i^{th} row of a', then the position of the k_i^{th} red ball is the same with or without the last ball, so $Z_i = Z_i'$.
– If $a_{i,j'} = 1$, and $k_i = \sum_{j=1}^{n} a_{i,j}$, then $Z_i = n$. There are less than k_i red balls left in the i^{th} row of a', so $Z_i' = (n-1)+1 = n$.

Therefore $Z = Z'$, so $\mathrm{E}\left[\frac{1_{\{1\le\pi(j_0)\le Z-1\}}}{Z-1}|\pi^{-1}(n) = j'\right] = h(m,n-1,a',k,j_0') = \frac{1}{n-1}$ by induction assumption. So

$$h(m,n,a,k,j_0) = \sum_{\substack{1\le j'\le n \\ j'\neq j_0}} \frac{1}{n-1}\frac{1}{n} + 0\frac{1}{n} = \frac{1}{n-1}\frac{n-1}{n} + 0\frac{1}{n} = \frac{1}{n}.$$

Corollary 1. *Let ball (i,j) be assigned value $x_{i,j}$. Let $g : \mathbb{R}^m \to \mathbb{R}$ be any function. We have*

$$\mathrm{E}_\pi\left[\frac{1}{Z-1}\sum_{j=1}^{Z-1} g(x_{1,\pi^{-1}(j)},\ldots,x_{m,\pi^{-1}(j)})\right] = \frac{1}{n}\sum_{j=1}^{n} g(x_{1,j},...,x_{m,j}).$$

Proof.

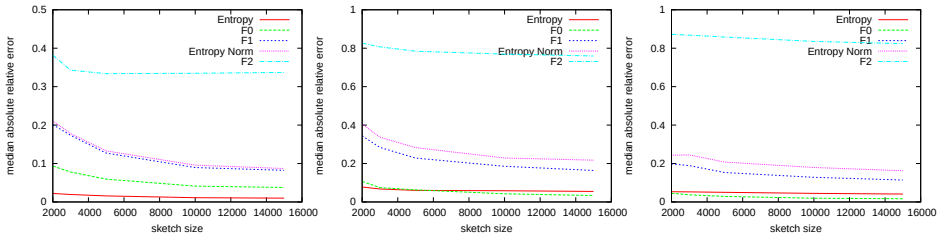
$$\begin{aligned}
&\mathrm{E}_\pi\left[\frac{1}{Z-1}\sum_{j=1}^{Z-1} g(x_{1,\pi^{-1}(j)},\ldots,x_{m,\pi^{-1}(j)})\right] \\
&= \mathrm{E}_\pi\left[\frac{1}{Z-1}\sum_{j=1}^{n} 1_{\{1\le\pi(j)\le Z-1\}} g(x_{1,j},\ldots,x_{m,j})\right] \\
&= \sum_{j=1}^{n} g(x_{1,j},...,x_{m,j})\mathrm{E}_\pi\left[\frac{1_{\{1\le\pi(j)\le Z-1\}}}{Z-1}\right] \\
&= \frac{1}{n}\sum_{j=1}^{n} g(x_{1,j},...,x_{m,j}).
\end{aligned}$$

Thus we see that the CRS algorithm is unbiased for multiple streams for any linear summary statistic.

4 Experiments

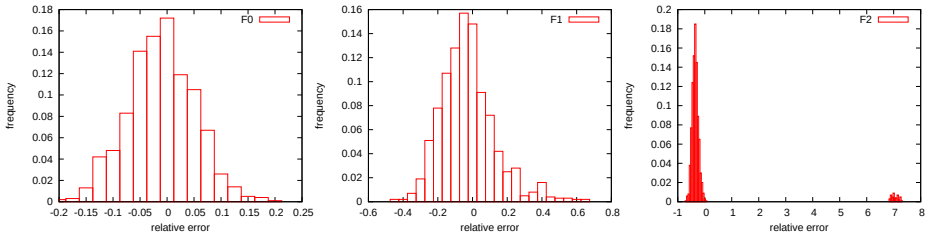
In this section we evaluate the performance of our algorithm using real Internet traffic traces obtained from a tier-1 ISP. We will investigate the impact of different sketch sizes on the estimation of the F_0, F_1,F_2, entropy and entropy norm and also investigate the distribution of relative error of the estimations. The results are compared with the entropy norm estimated by [33] and the F_2 estimated by stable random projection (SRP) [13]. We would also investigate the impact of the number of flows on the estimation error.

The traffic traces in use were collected from an ingress router of the tier-1 ISP. We also obtained the routing table at the ingress router to determine the egress router for



(a) Egress 1 (b) Egress 2 (c) Egress 3

Fig. 1. Median Absolute Relative Error with Varying Sketch Sizes. Summary statistics are listed in decreasing order of MARE.



(a) F_0 (b) F_1 (c) F_2

Fig. 2. Distribution of relative error of F_0, F_1, F_2 of the intersection, with sketch size fixed at $10,000$

each packet, thus determining the OD flows to each possible egress router. Because we do not have traffic traces collected at egress routers, we generate synthetic traffic traces at egress routers in addition to the OD flows observed at the ingress router. The synthetic traffic is generated with an identical distribution to that of the ingress. For the following experiments we pick a five-minute trace which contains $227,722$ flows, and we pick three different egress router whose OD traffic contains $5,442$ flows (Egress 1), $5,362$ flows (Egress 2), and $25,276$ flows (Egress 3).

We use H_3 hash function [3] in the simulations, since we notice that hash functions based on Linear Feedback Shift Register (LFSR) do not work well for our application. This is because some flows differ from each other by only a few bits in the flow label. For example, different HTTP sessions from the same client to the same web server only differ in the source port number field. An LFSR-based hash function often does not distribute the hash values uniformly in such cases while H_3 does. To compute an H_3 hash function, the input bit vector is multiplied by a fixed (randomly-chosen) $\{0,1\}$-valued matrix modulo 2 to get the hashed bit vector. This hash function has been also shown to be very amenable for hardware implementation [26] since it involves only simple logic such as bitwise XOR and parity computation. Therefore, we recommend that H_3 functions be used in implementing CRS on future Internet routers.

We run the experiments 1000 times for each configuration of varying sketch sizes. Figure 1 depicts the median absolute relative error (MARE) for different statistics using

different sketch sizes on all three egresses. The results for the three egresses are similar. We observe that for F_0, F_1, and entropy norm, the relative error decreases as sketch size increases, just as expected. For entropy, the error is very small even at small sketch sizes. On the other hand, the relative error of F_2 remains large even at large sketch sizes. The F_0 errors of Egress 1 and Egress 2 are similar, however the F_1 error of Egress 2 is worse than Egress 1 and 3 since the traffic to Egress 2 is actually more heavy-tailed than the other two. The errors of Egress 3 are smaller than Egress 2 due to the larger number of flows to Egress 3.

The entropy estimation error is much smaller than the error for F_1 and entropy norm, the two components for estimating entropy. This is because the CRS sketch will either overestimate both F_1 and entropy norm, or underestimate both, so the errors end up largely canceling each other out. Compared with the method in [33], our method achieves MARE of around 10% with a sketch of $10,000$ entries for entropy norm, which is comparable with [33] using $200,000$ entries. (The size of each entry in our sketch , however, is a little over twice that of [33].) Our explanations for this is as follows: The estimation of entropy norm in [33] is based on the estimation of $F_{1-\alpha}$. The formulae (3) and (4) in [33] show that any error for estimating $F_{1-\alpha}$ will go opposite ways for F_1 and entropy norm, thus causing larger errors for the estimation of entropy.

We now fix the sketch size to 10000 entries and investigate the distribution of the relative error. Figure 2 shows the distribution of relative errors of F_0, F_1, and F_2 estimates of traffic from the ingress to Egress 1. The distributions for Egress 2 and 3 are similar and hence omitted. We see that for F_0 and F_1, the distribution function are both roughly bell curves centered slightly off 0. However, there are two modes of the distribution of F_2 error, one centered around -0.3 and the other around 7. This means that F_2 estimation most likely underestimates by 30%, and occasionally overestimates by 700%. This is caused by the presence of a large elephant in the OD flow.

An alternative algorithm for estimating F_2 is to use random stable projections (RSP) [13]. In Figure 3 we compare the theoretical standard deviation (SD) of the CRS sketch and the random stable projection sketch for this dataset. We see that the stable projection sketch would perform better, due to the heavy tailed nature of this OD flow. However we need to emphasize that a single CRS sketch can be used to measure many quantities, while a stable sketch for F_2 can only be used for F_2 measurement alone.

We also plotted the sample SD of the CRS sketch from 1000 runs in Figure 3. We could see the sample SD is very close to the theoretical SD for large sketch sizes, which validates our experiments. However the variance of the SD is so high that for small sketch sizes the sample SD cannot accurately capture the true SD with 1000 runs, which explains that deviations for sketch sizes below 6000.

Finally, we change the number of flows for Egress 1 while maintaining the same set of OD flows. We plot the errors in Figure 4. We can see that the left side of the error curves are flat when the number of egress flows is no more than the number of ingress flows, but the right side goes up. This demonstrates that the performance of CRS sketch depends on the max of the number of flows of ingress and egress. In other words, the greater the traffic at either node relative to the OD traffic, the less accuracy we expect from the CRS sketch.

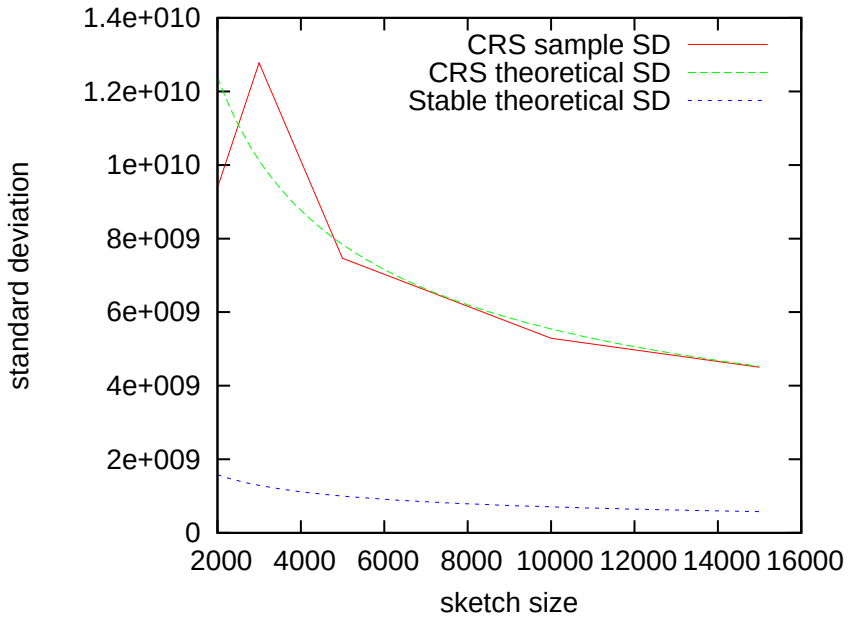


Fig. 3. Comparison with stable distribution sketch for F_2 estimation (varying sketch size)

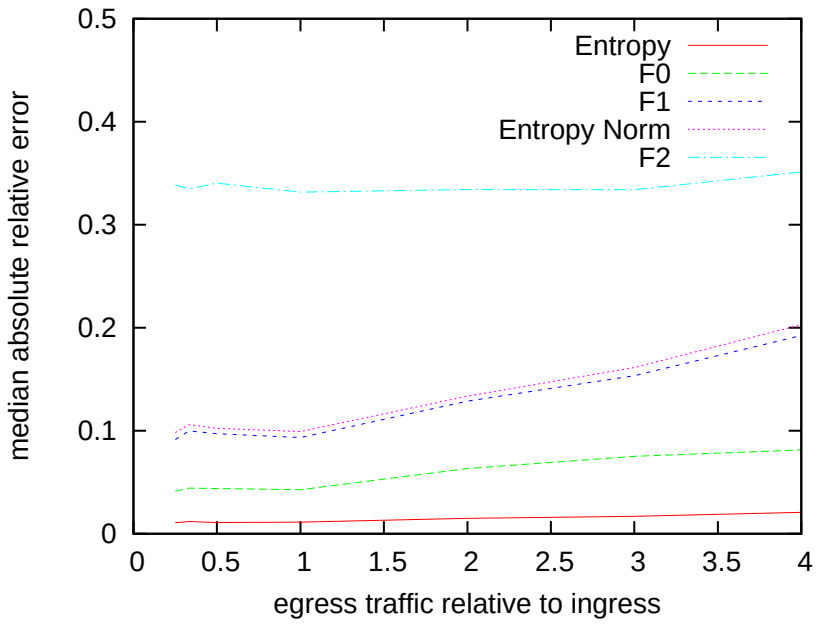


Fig. 4. Varying Egress Flows (Egress 1)

5 Related Work

This work is most similar to, and in fact was inspired by, the Conditional Random Sampling sketch [21]. The CRS sketch was initially proposed for sparse language data, and we show in this paper how it can be adapted to the networking context. Since the target goals and data characteristics are extremely different in these two domains, the issues in adapting this sketch were far from trivial. In addition, we present a novel proof in this paper that the CRS sketch can give *unbiased* estimates of arbitrary functions of the intersection of a pair of streams.

The key idea of bottom-k sketch [4, 5] is close to that of the CRS sketch. The bottom-k sketch also maintains the k items with the lowest-k hashed values, and its hashing can be also coordinated hence supporting queries on intersections. However, although the bottom-k sketch can be applied to streaming applications, it is only for the situations in which the weight of each item is known and fixed. The estimators derived are for the aggregate of functions of these weights. In comparison, the CRS sketch performs coordinated sampling of a subset and then estimates any function of the whole set based on the sampled set.

The idea of coordinated sampling can be traced back to Trajectory sampling [6]. Trajectory sampling samples and stores the flows with hashed flow IDs in the same range. Compared with our method, the size of sketch sampled by Trajectory sampling on each router would be in proportion to the amount of traffic passing through it. Hence it lacks flexibility with regard to traffic fluctuations and traffic imbalance among routers. A recent work c-Samp [27] also leverages coordinated flow sampling. However, its design goal is to assign different sampling ranges for different routers in order to achieve larger monitoring coverage.

There has been a long line of work done on computing the traffic matrix within a network [28, 11, 23, 31, 32, 25, 24, 34]. While much of the previous work in this area was done using SNMP link loads [28, 31, 32, 24], there has been some recent interest in using data streaming techniques to measure the matrices [11, 25, 34]. Our algorithm is closer to these latter works in that it directly uses traffic data, rather than SNMP.

Entropy has recently been proposed, in several different contexts [10, 17, 29, 30], as a means for classifying and detecting anomalies in network traffic. Computing the entropy of OD-pairs was first proposed in [33], in which the authors propose the use of a sketch at each ingress and egress node in a network that can be used to compute the entropy of any arbitrary pair of nodes. The sketch in this paper is able to solve the same problem, but it is not restricted to solving just this single problem.

Recently, there is a line of work on *Compressed Counting* [18, 22], which applied *maximally-skewed stable random projections* to estimate the F_p frequency moment near $p = 1$ and then use the estimated F_p to approximate the Shannon entropy with a very high accuracy. For example, [22] conducted some comparisons with CRS to illustrate that, in order to achieve the same accuracy for estimating the Shannon entropy, Compressed Counting can reduce the required sample size of CRS by 2-fold to 113-fold (data-dependent).

6 Conclusions

In this paper we present and evaluate a sketch for computing any norm on the intersection of pairs of spatially separated streams. As data on the Internet gets more voluminous and more distributed, it will become increasingly important to be able to sketch and compare datasets at remote locations. The work in this paper is a first step towards addressing this challenging new problem.

The distinctive property of our sketch, as compared with prior work, is that it is not limited to a single type of query and can be used to execute arbitrary queries, even ones realized after the data has been collected. It is also designed to be lightweight, both in terms of memory requirements as well as update costs. We show that it is unbiased and that it performs well empirically by evaluating it on real Internet data collected at a tier-1 ISP.

Acknowledgement

The work of Hua, Lall, Xu, and Zhao was supported in part by the NSF grant CNS 0905169. The work of Ping Li was supported in part by the NSF grant DMS 0808864, the ONR grant YIP N000140910911, and a grant from Microsoft.

References

1. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–143, 1999.
2. A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, 1997.
3. L. Carter and M. N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
4. E. Cohen and H. Kaplan. Bottom-k sketches: better and more efficient estimation of aggregates. In *SIGMETRICS*, 2007.
5. E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *PODC*, 2007.
6. N. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE transaction of Networking*, pages 280–292, June 2001.
7. N. Duffield, C. Lund, and M. Thorup. Estimating flow distribution from sampled flow statistics. In *Proc. ACM SIGCOMM*, Aug. 2003.
8. N. G. Duffield, C. Lund, and M. Thorup. Flow sampling under hard resource constraints. In *Sigmetrics*, 2004.
9. C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proc. ACM SIGCOMM*, 2002.
10. L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical approaches to DDoS attack detection and response. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2003.
11. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE transaction on Networking*, June 2001.
12. A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large ip backbone–a comparison on real data. In *USENIX/ACM SIGCOMM IMC*, 2004.

13. P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.
14. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *IMC*, 2003.
15. A. Kumar, M. Sung, J. Xu, and E. Zegura. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In *Proc. ACM SIGMETRICS*, June 2005.
16. A. Kumar and J. Xu. Sketch guided sampling–using on-line estimates of flow size for adaptive data collection. In *Proc. IEEE INFOCOM*, Mar. 2006.
17. A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *SIGCOMM*, 2005.
18. P. Li. Improving compressed counting. In *UAI*, 2009.
19. P. Li and K. W. Church. Using sketches to estimate associations. In *Human Language Technology and Empirical Methods in Natural Language Processing (HLT)*, 2005.
20. P. Li, K. W. Church, and T. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In *NIPS*, 2006.
21. P. Li, K. W. Church, and T. Hastie. One sketch for all: Theory and application of conditional random sampling. In *NIPS*, 2008.
22. P. Li and C.-H. Zhang. A new algorithm for compressed counting with applications in shannon entropy estimation in dynamic data. In *COLT*, 2011.
23. A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation:existing techniques and new directions. In *SIGCOMM*, 2002.
24. A. Nucci, R. Cruz, N. Taft, and C. Diot. Design of igp link weight changes for estimation of traffic matrices. In *Proc. IEEE INFOCOM*, Mar. 2004.
25. K. Papagiannaki, N. Taft, and A. Lakhina. A distributed approach to measure traffic matrices. In *Proc. ACM/SIGCOMM IMC*, Oct. 2004.
26. M. V. Ramakrishna, E. Fu, and E. Bahcekapili. Efficient hardware hashing functions for high performance computers. *IEEE Trans. Computers*, 46(12):1378–1381, 1997.
27. V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. R. Kompella, and D. G. Andersen. csamp: A system for network-wide flow monitoring. In *NSDI*, 2008.
28. Y. Vardi. Internet tomography: estimating source-destination traffic intensities from link data. *Journal of American Statistics Association*, pages 365–377, 1996.
29. A. Wagner and B. Plattner. Entropy Based Worm and Anomaly Detection in Fast IP Networks. In *Proceedings of IEEE International Workshop on Enabling Technologies, Infrastructures for Collaborative Enterprises*, 2005.
30. K. Xu, Z.-L. Zhang, and S. Bhattacharya. Profiling internet backbone traffic: Behavior models and applications. In *SIGCOMM*, 2005.
31. Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proc. ACM SIGMETRICS*, June 2003.
32. Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. ACM SIGCOMM*, Aug. 2003.
33. H. Zhao, A. Lall, M. Ogihara, O. Spatscheck, J. Wang, and J. Xu. A data streaming algorithm for estimating entropies of OD flows. In *IMC*, 2007.
34. Q. Zhao, A. Kumar, J. Wang, and J. Xu. Data streaming algorithms for accurate and efficient measurement of traffic and flow matrices. In *SIGMETRICS*, June 2005.