Sketching Unaggregated Data Streams for Subpopulation-Size Queries

Edith Cohen
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
edith@research.att.com

Nick Duffield
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
duffield@research.att.com

Haim Kaplan
School of Computer Science
Tel Aviv University
Tel Aviv, Israel
haimk@cs.tau.ac.il

Carsten Lund
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
lund@research.att.com

Mikkel Thorup
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
mthorup@research.att.com

ABSTRACT

IP packet streams consist of multiple interleaving IP flows. Statistical summaries of these streams, collected for different measurement periods, are used for characterization of traffic, billing, anomaly detection, inferring traffic demands, configuring packet filters and routing protocols, and more. While queries are posed over the set of flows, the summarization algorithm is applied to the stream of packets. Aggregation of traffic into flows before summarization requires storage of per-flow counters, which is often infeasible. Therefore, the summary has to be produced over the unaggregated stream.

An important aggregate performed over a summary is to approximate the size of a subpopulation of flows that is specified a posteriori. For example, flows belonging to an application such as Web or DNS or flows that originate from a certain Autonomous System. We design efficient streaming algorithms that summarize unaggregated streams and provide corresponding unbiased estimators for subpopulation sizes. Our summaries outperform, in terms of estimates accuracy, those produced by packet sampling deployed by Cisco's sampled NetFlow, the most widely deployed such system. Performance of our best method, step sample-and-hold is close to that of summaries that can be obtained from pre-aggregated traffic.

Categories and Subject Descriptors: G.3: probabilistic algorithms; C.2.3: network monitoring

General Terms: Algorithms, Measurement, Performance

Keywords: sketches, data streams, subpopulation size, IP flows.

1. INTRODUCTION

Measurement, collection, and longer-term storage of historic data on network traffic is necessary for many applications. Applications view traffic as a collection of flows but the underlying data stream consists of interleaved packets of multiple flows. The flow that a packet belongs to is identified by data present in the header fields of the packet: the source and destination IP address, source and destination ports, and protocol. We refer to such streams, where items are broken into interleaving chunks, as *unaggregated data streams*. Due to the sheer volume of the data, the practice is to collect and record a sketch of each time period that allows for approximate query processing. The sketch is produced by an algorithm that is applied to the raw packet stream.

IP packet streams are processed in real-time at the routers by systems such as Cisco's sampled NetFlow (NF) or processed by software tools such as Gigascope [6]. Algorithms maintain a set of statistics counters. Each counter corresponds to a single flow and counts some of the packets of this flow. These counters reside in memory and memory speed is critical. High volume routers use fast (but expensive) SRAM for this counting. Typical Zipf-like flow-size distributions have a large number of small flows and therefore a large number of distinct flows. The limited memory and large number of flows prevent us from counting and aggregating all packets of all flows. Therefore, the summarization algorithms have to be applied to the unaggregated stream and not to the aggregated set of flows.

NF samples the packet stream at a fixed rate and then aggregates it into flows. That is, there is a single active counter for each flow in the sampled stream. The sampling at the packet level reduces the number of distinct flows by filtering out the bulk of the small flows and this allows for per-flow counting of the sampled stream.

An algorithm that is able to collect more information than NF using the same number of statistics counters is *sample-and-hold* (SH) [11, 10]. With SH, as with NF, packets are sampled at a fixed rate and once a packet from a particular flow is sampled, the flow is *actively counted*. The difference is that with SH, once a flow is actively counted, all subsequent packets that belong to the same flow are counted (with NF, only sampled packets are counted). A disadvantage of SH over NF is that it requires processing of every packet.

NF and SH use a fixed sampling rate, as a result, the number of distinct flows that are sampled and therefore the number of statistics counters required is variable. When conditions are stable, the number of distinct flows sampled using a given sampling rate has small

PODS'07, June 11–14, 2007, Beijing, China.

variance and one can manually adjust the sampling rate so that the number of counters does not exceed the memory limit and most counters are utilized [10]. Anomalies such as DDoS (Distributed Denial of Service) attacks and port scanners, however, introduce many small flows and can greatly increase the number of distinct flows. A fixed-sampling-rate scheme can not react to such variability: required memory would exceed the available memory, and result in disruption of measurement or degraded router performance. These issues promoted the proposal of adaptive schemes that adjust the sampling rate on the go so that the number of active counters does not exceed some fixed value k. The adaptive variant of NF, which we refer to as *adaptive sampled NetFlow* (ANF) along with implementation were suggested in [11, 9, 14]. The adaptive variant of SH, that we refer to as *adaptive SH* (ASH) was proposed in [11, 10].

An important query type is the amount of traffic that belongs to a specified subpopulation of flows (for example, a particular protocol, from a specified source, or both). The specification of the subpopulation is often provided after the sketch is produced. Therefore, it is important to retain sufficient meta data information and provide estimators that facilitate such queries.

Overview

We design and evaluate stream algorithms that obtain sketches from unaggregated streams. We derive estimators that allow us to obtain unbiased estimates of arbitrary subpopulations from these sketches and analyze the variance of these estimators.

The sketches have the form of a subset of the flows along with the flow attributes and an *adjusted weight* associated with each flow. The adjusted weight of flows not in the sample is defined to be zero. The adjusted weight has the property that for each flow, the expectation of the adjusted weight is equal to its actual size (thus, the adjusted weight of a flow that appears in the sample is generally larger than its actual weight). Therefore, an unbiased estimate for the size of a subpopulation of flows can be obtained by summing the adjusted weights of flows in the sketch that belong to this subpopulation. The quality of the adjusted weight assignment depends on the distribution over subsets of flows that are included in the sketch, the information collected by the streaming algorithm, and the procedure used to calculate these weights. All methods considered obtain unbiased estimates, and therefore, performance depends on the variance of these assignments.

The adaptive ASH implements each decrease of the sampling rate by reducing the counts in order to emulate those obtained with a fixed sampling rate that is equal to the current rate. By doing so, however, it discards the more informative counts obtained with the lower rate.

We design *step-counting sample-and-hold* (SSH). Similarly to the adaptive ASH, the sampling rate is decreased in steps to ensure that the number of flow counters does not exceed the limit. Unlike ASH, SSH records the current counts for all surviving flows when the sampling rate is decreased. These per-step counts are transferred to secondary storage and are used for calculating the adjusted weights.

This design distinguishes between *primary memory* used to actively process the stream and slower *secondary memory* used for intermediate storage. In the case of IP routers, the primary memory corresponds to fast and expensive SRAM and the secondary memory to cheaper and slower DRAM. This distinction was used in [17, 16] for more efficient processing of IP packet streams. The available SRAM is used for fast counting with small counter sizes and intermediate counts are transferred periodically to the slower DRAM.

More informative counts can provide an advantage only if lower-variance adjusted weight can be derived from these counts. An important contribution we make is the understanding of what information to retain and how to use it to obtain correct adjusted weights. Using NF, the adjusted weight of each flow in the sketch is simply a scaling of its sampled weight (dividing the counted size by the sampling rate). We derive correct adjusted weights for ANF, SH, ASH, and SSH. The per-step counts recorded by SSH are necessary for computing correct step-based adjusted weights. After the adjusted weights are computed, these counts can be discarded. Therefore, SSH uses the same mechanism for active counting (and same size primary memory) as ASH, produces the same-size final sketch, but does require a larger intermediate storage.[1]

We show that the distribution of the subsets of flows included in the final sketch, for all methods, correspond to a weighted sample without replacement drawn from aggregated flows (WS). Therefore, the difference in accuracy stems only from the adjusted weights. We establish that for *all distributions and all subpopulations*, the variance of the adjusted weight assignment of SSH is at most that of ASH which is at most that of ANF. We use a derivation of adjusted weights for WS [2], and show that the variance of SSH is at least that of WS. The adjusted weights in [2] assume that the exact weight of each sampled flow is provided, an information that can not be collected using a single pass on an unaggregated data stream (two passes are sufficient).

We argue that there should be a notable performance gain for SSH over ASH and for ASH over ANF on populations that consist of flows with frequency that is well above the sampling rate. We evaluate the performance of these methods on flow sizes drawn using Pareto distributions with different parameters. To understand how performance depends on the consistency of the subpopulation, we use subpopulations of large or small flows. We observe that SSH is considerably more effective than other methods on subpopulations that include mid-size to larger flows and performs closely to estimates obtained using WS.

2. RELATED WORK

A related problem is producing a summary of *aggregated* data [12], that is, data where each item appears as a single quantity. Two methods that obtain a sketch with adjusted weights are weighted sampling without replacement (WS) (adjusted weights derived in [2]) and priority sampling [8] (PRI). These methods can be combined with an algorithm for unaggregated streams when the bottleneck resource is not the number of active counters but rather, the size of the final sketch: If the sketch obtained from the unaggregated stream provides unbiased adjusted weights, these methods can be applied to the sketch with the adjusted weights in order to obtain a smaller sketch with unbiased adjusted weights. In such a case, the variance of the estimates also depend on the variance of the adjusted weights assigned by the unaggregated stream algorithm. In our evaluation we compare our unaggregated streams estimators with the (unattainable) aggregated streams estimator.

ASH counts were used in [11, 10] to find heavy hitters (elephant flows). An extension of ASH that does not discard counts when the sampling rate is adjusted was proposed in [10]. This extension, however, is not adequate for estimating subpopulation sizes: while the unadjusted count itself is a better estimator than the reduced count, it is biased, and the bias heavily depends on where in the measurement epoch the packets occur. In fact, there is not sufficient information maintained to obtain an unbiased estimate or to

[1] The size of intermediate storage is larger by at most a logarithmic factor but in practice, by much less.

compensate for the bias.

Kumar et al [15] proposed a streaming algorithm for IP traffic that produces sketches that allow us to estimate the flow size distribution (FSD) of subpopulations. Their design executes two modules concurrently. The first is a sampled NetFlow module that collects flow statistics, along with full flow labels, over sampled packets. The second is a streaming module that is applied to the full packet stream and uses an array of counters, accessed by hashing. Estimating the flow size distribution is a more general problem than estimating the size of a subpopulation, and therefore this approach can be used to estimate the subpopulations sizes. To be accurate, however, the number of counters in the streaming module should be roughly the same as the number of flows and therefore the size of primary memory is proportional to the number of distinct flows. As we see here, however, accurate estimates for subpopulation sizes can be obtained more efficiently using other approaches.

3. SAMPLED NETFLOW

NF uses fixed-rate packet sampling, where packets are sample at a fixed rate p and the sampled packets are aggregated into flows. The size of the sketch is the number of distinct flows that are sampled. The adjusted weight assigned to each flow is the number of sampled packets that belong to the flow, divided by p. The number of active flow counters that is used is equal to the size of the sketch.

With ANF, decrease of the sampling rate from p_1 to $p_2 < p_1$ can be implemented by resampling every sampled packet with rate p_2/p_1. Flows with 0 resampled packets are discarded. This resampling process can be performed more efficiently using the binomial distribution. A flow with i counted packets with sampling rate p_1 has $B(i, p_2/p_1)$ sampled packets with sampling rate p_2. Implementation design for ANF in IP routers was provided in [9].

We can view the decrease of the sampling rate as an incremental process: when we obtain the $(k+1)$st active flow using the current value of the sampling rate (this flow has a single counted packet), we decrease the sampling rate "just enough" so we remain with exactly k active flows. This decrease itself is a random variable and so is the distribution over the k flows that remain and their counts. It can be simulated by drawing for each packet of each active flow, a value from $U[0, p_1]$. We then consider the lowest value over the packets of each flow. The new sampling rate p is the largest of these values among the $k+1$ flows. We then reset the counts to only consider packets with value less than p, as a result, one flow is discarded and we remain with k active flows.

3.1 Rank-based view of the sample space

For the analysis of NF and other algorithms, we use the following *rank-based view of the sample space*: Each point in the sample space is a *rank assignment*, where each packet is assigned a rank value that is independently drawn from $U[0, 1]$. For each flow $f \in F$ and point in the packet stream, we define the *current rank value* $r(f)$ to be the smallest rank assigned to a packet of the flow that occured before the current point in the packet stream.

A NF sketch with sampling rate p is equivalent to obtaining a rank assignment and counting all packets that have rank value $< p$. The actively counted flows at a given time are those with rank that is at most p. The number of flows in the sketch is $|f \in F | r(f) < p|$ at the end of the measurement epoch.

For adaptive algorithms, including ANF, we define the *current sampling rate* to be the $(k+1)$st smallest rank among $r(f)$ $(f \in F)$. The set of actively counted flows at a given time are those with rank that is below the current sampling rate. We refer to the value of the current sampling rate at the end of the measurement period as the *effective sampling rate*.

ANF with sketch of size k includes in the sketch the flows with final rank value that is below the effective sampling rate. The number of packets counted for each flow are those with rank value that is below the effective sampling rate. It is not hard to see that this rank-based view results in the same distribution of ANF sketches and counts as the process that decreases the sampling rate "just enough" as to remain with k flows.

3.2 Adjusted weights for ANF

A subtle issue is the assignment of correct (unbiased) adjusted weights for ANF. Since we change the conditioning it is not clear that we can simply scale the counts by the final sampling rate. We derive adjusted weights for a flow by partitioning the sample space as in [2, 3, 8]. It turns out that to obtain unbiased adjusted weights, we can not simply use *a* sampling rate that yields k distinct flows but rather the *highest* sampling rate (as selected in this incremental adjustment process) that yields k (and not $k+1$) distinct flows. In the rank-based view, all sampling rates between the kth smallest and the $(k+1)$st smallest rank values yield k distinct flow on that sample. This "highest" rate, however, is the $(k+1)$st smallest rank, which we call the effective sampling rate.

LEMMA 3.1. *Correct adjusted weights for* ANF *are obtained by scaling the counts by* $1/p'$*, where* p' *is the effective sampling rate.*

PROOF. Consider a flow f', and the probability subspace where the kth smallest rank among $r(f)$ $(f \in F \setminus \{f'\})$ is p'. Consider the conditional distribution of the number of packets of flow f' that are counted. The number of packets is just like what would have been counted with NF with rate p'. Therefore, scaling this count with $1/p'$ yields unbiased adjusted weight for f' within this probability subspace. $\square$

4. SAMPLE AND HOLD

With SH, like with NF, packets are sampled uniformly at a fixed-rate p. If a sampled packet belongs to a flow that is not actively counted (not yet sampled), a new counter is created. With SH, however, all subsequent packets of active flows are counted. Therefore SH processes every packet (not only sampled packets) to determine if the flow is actively counted.

The rank-based view of SH (see Subsection 3.1) includes in the sketch a count of all packets in the rank assignment such that the current rank of the flow (including the current packet) is smaller than p. Therefore, for a given rank assignment and rate p, the set of flows that are being actively counted, and the set of flows in the final sketch, is the same for SH and NF.

4.1 Adjusted weights assignment

LEMMA 4.1. *A correct adjusted weight assignment is for each sampled flow to have weight equal to its count plus* $(1-p)/p$.

PROOF. The only information retained for the sampled flows is the counts and the adjusted weight depends only on the sampling rate and the count. We derive $A_p^{\mathrm{SH}}(i)$, the adjusted weight assigned to a flow with count of i packets (we omit the superscript and subscript when clear from context) and show that this is the unique deterministic assignment.

We have $A_p(0) = 0$ (items that are not sampled at all obtain zero adjusted weight). In order for the assignment to be unbiased for 1-packet flows, we have to have $pA_p(1) + (1-p)A_p(0) = 1$. Substituting $A_p(0)$, we obtain that $A_p(1) = 1/p$. To be correct for n-packet items, we must have $\sum_{i=0}^{n}(1-p)^i pA_p(n-i) = n$.

We can solve these for $n = 2, 3, 4, \ldots$ and obtain that $A_p(n) = (1 + (n-1)p)/p = (1-p)/p + n$ for $n \geq 1$. $\square$

This assignment can also be derived by applying the Horvitz-Thompson estimator to each packet. For each packet, we look at the partition of the sample space to two parts, one where a previous packet is sampled, and the other where a previous packet is not sampled. The adjusted weight assigned to a packet is unbiased on each part: if a previous packet is sampled, the probability that a packet is counted is 1 and its adjusted weight is 1. If no previous packet is sampled, then the probability that the packet is sampled is p and the Horvitz-Thompson adjusted weight is $1/p$. The adjusted weight of the flow is the sum of the adjusted weights of sampled packets. This assignment can be interpreted as the first sampled packet of the flow representing $1/p$ unseen packets whereas subsequent counted packets of the flow represent only themselves.

4.2 Adaptive sample and hold

Like ANF, ASH adaptively decreases the sampling rate so that we obtain a sketch of fixed size k and use at most k active flow counters. Decrease of the sampling rate from p_1 to $p_1 > p_2$ can be emulated as follows [11]: We perform the following for a flow with a count of i: The count remains i with probability p_2/p_1 (this corresponds to resampling the first packet as to emulate sampling it with probability p_2, if it is resampled, all subsequent packets are counted.). Otherwise, (if the first packet is not re-sampled) subsequent packets are re-sampled with probability p_2 until one is sampled and then all the packets following it are counted. If no packet is re-sampled, the flow is no longer active and the counter is discarded. More formally, with probability (p_2/p_1) the count remains i. Otherwise, let r be random variable from a geometric distribution with p_2. The count decreases from i to $\max\{0, i-1-r\}$. If the count of the flow becomes 0, the flow becomes inactive.

As with ANF, the decrease of the sampling rate occurs when we have $(k+1)$ active flow and the rate is reduced "just enough" so we remain with k active flows. The decrease of the sampling rate, and the updated counts, are a random variable that depends on the current counts and sampling rate. The notion of "just enough" is formalized as follows. Independent values from $U[0, p_1]$ are drawn for the first counted packet and from $U[0, 1]$ for each subsequent packet. We then consider the lowest value for each flow, and set the new sampling rate p to be the highest of these $(k+1)$ values. The set of counted packets includes all packets with value above p.

With the rank-based view of ASH, a flow is actively counted when its rank is below the current sampling rate and the set of counted packets is all packets that occured when the rank of the flow was below the current sampling rate. It is not hard to see that the distribution of sketches and counts and sketches is equivalent to that of the process that decreases the sampling rate "just enough" as to remain with k active flows.

LEMMA 4.2. *Let p' be the effective sampling rate. Adjusted weights assigned using flow count plus $(1-p')/p'$ are unbiased.*

PROOF. The proof is similar to that of Lemma 3.1: For each flow, we consider the probability subspace where the kth smallest rank among other flows is fixed and assign unbiased weights within this subspace. $\square$

5. STEP SAMPLE-AND-HOLD

The active counting of packets performed by SSH is just like ASH: exactly the same set of flows is actively counted at any given time and the same implementation can be used to determine this set of flows. The difference is in the information SSH records. ASH maintains one count for each flow and the counter is adjusted when the current sampling rate decreases; if the flow counter is adjusted to 0 the flow is no longer actively counted. SSH, instead of adjusting the counter, records the current count for the flows that are actively counted after the rate decrease, and initializes the counters of these flows to zero in the current step. Counts of previous steps for flows that are not actively counted are discarded. Therefore, the information maintained by SSH for each actively counted flow, is a per-step count for all previous steps and an active counter for the current step. For the analysis, we use the rank-based view, where packets have associated ranks drawn independently from $U[0, 1]$.

We will show how to compute correct adjusted weights for the sampled flows. The information collect by SSH is the step function $1 \geq p_1 > p_2 > \cdots > p_r$ denoted by the vector $\mathbf{p} = (p_1, \ldots, p_r)$ of the current sampling rate and for each sampled flow f, the counts $\mathbf{i}(f) = (i_1(f), i_2(f), \ldots, i_r(f))$ of the number of packets recorded at each step. The adjusted weight we assign to f is a function of the steps and the counts of f. We use the notation

$$A^{\mathrm{SSH}}_{\mathbf{p}}(\mathbf{i}(f)) \equiv A^{\mathrm{SSH}}_{p_1, p_2, \ldots, p_r}(i_1(f), i_2(f), \ldots, i_r(f))$$

for the adjusted weight. The adjusted weights are computed after the algorithm terminates. The per-step counts can be discarded after the adjusted weights are computed, and therefore, the size and form of the sketch is the same as ASH and ANF.

As in the proofs of Lemma 3.1 and Lemma 4.2 we obtain adjusted weights for a flow f that are unbiased in the probability subspace defined by the steps of the rank value of the current kth-smallest rank of a flow among $F \setminus f$. These steps are the same as the current sampling rate when the flow is actively counted. Technically, we need to consider the kth-smallest rank of an actively counted flow on steps that precede the active counting of f. As we shall see, however, the adjusted weight function has the property

$$A_{p_1, p_2, \ldots, p_r}(0, 0, \ldots, i_j, i_{j+1}, i_r) = A_{p_j, p_{j+1}, \ldots, p_r}(i_j, i_{j+1}, i_r) \quad (1)$$

and therefore does not depend on the current sampling rate in the duration before the final contiguous period where the flow is actively counted. This means that it is sufficient to record the steps of the current sampling rate.

We motivate SSH through the following example. A 200 packets is counted using ASH where the final (effective) sampling rate is $p = 1/10$. Suppose that the sampling rate has 2-step, with the first step having sampling rate $1/2$, and 100 packets arriving in each step. The final ASH count mimics a fixed sampling rate of $1/10$. Therefore, on average, about 10 packets of the flow are not counted. With SSH, the expected number of packets that are not counted is only slightly more than 2. Therefore, SSH can potentially obtain estimates with lower variance.

5.1 Adjusted weights

For each sampled flow we have the number of packets counted in each step $\mathbf{i} = (i_1, \ldots, i_r)$. In the computation of the adjusted weight $A(\mathbf{i})$, we assume without loss of generality that $i_1 > 0$ (see Eq. (1)). We express the adjusted weight as a solution of a triangular system of equations.

We use the notation $q[\mathbf{i}|\mathbf{n}] = q[i_1, i_2, \ldots, i_r | n_1, n_2, \ldots, n_r]$ for the probability that a flow with n_j packets in step j has an observed count of i_j in step j (for $j = 1, \ldots, r$). Evidently, any valid observed count $(i_1, \ldots, i_r)$ has the form $(0, 0, i_j, n_{j+1}, \ldots, n_r)$, where $1 \leq i_j \leq n_j$. We refer to vectors of this form as *suffixes* of the vector $\mathbf{n} = (n_1, \ldots, n_r)$. We use the notation $\mathrm{SUFF}_{\mathbf{n}}(j, i_j)$ for the suffix $(0, \ldots, 0, i_j, n_{j+1}, \ldots, n_r)$ of $\mathbf{n}$ (subscript is omitted when clear from context). We utilize the total order over the suffixes $\mathrm{SUFF}_{\mathbf{n}}(i, i_j)$ defined by the lexicographic order on $(-j, i_j)$.

That is,

$$\begin{aligned}
& \text{SUFF}(r,1) = (0,\ldots,0,1) \prec \text{SUFF}(r,2) = (0,\ldots,0,2) \prec \ldots \\
\prec\ & \text{SUFF}(r,n_r) = (0,\ldots,0,n_r) \prec \text{SUFF}(r-1,1) = (0,\ldots 0,,1,n_r) \\
\prec\ & \cdots \prec \text{SUFF}(1,n_1) = (n_1,\ldots,n_r)
\end{aligned}$$

A correct adjusted weight assignment must have, for any vector $\mathbf{n} = (n_1,\ldots,n_r)$,

$$\sum_{\mathbf{s}\preceq\mathbf{n}} q[\mathbf{s}|\mathbf{n}]A(\mathbf{s}) = \sum_{j=1}^{r} n_j \ . \tag{2}$$

(For a flow with counts $(n_1,\ldots,n_r)$, the expectation of the adjusted weight should be equal to the size $\sum_{j=1}^{r} n_j$ of the flow.)

To compute $A(\mathbf{i})$, we obtain a system of equations using an instance of Equation 2 for each suffix $\mathbf{x} \preceq \mathbf{i}$ of the observed count. The system of equations includes the variables $A(\mathbf{x})$ for all $\mathbf{x} \preceq \mathbf{i}$ and the coefficients $q[\mathbf{s}|\mathbf{x}]$ for all $\mathbf{s} \preceq \mathbf{x}$.

5.2 Expressing the probabilities $q[\mathbf{s}|\mathbf{n}]$.

We define the values $f(j,t,\mathbf{n})$ $(j \leq t \leq r)$ where $f(j,t,\mathbf{n})$ is the probability that all packets in the suffix that starts at step j are counted, given that the rank of the flow at the beginning of step j is in the interval $(p_{t+1}, p_t]$. For convenience, we define $p_{r+1} \equiv 0$, $f(r,r,\mathbf{n}) \equiv 1$, and $f(j,t,\mathbf{n}) = 0$ for $t < j$. We use the following recursion: for $j < r$ and $t \geq j$ we have,

$$\begin{aligned}
f(j,t,\mathbf{n}) &= (1-p_{t+1})^{n_j} f(j+1,t,\mathbf{n}) \\
&+ \sum_{i=t+1}^{r} ((1-p_{i+1})^{n_j} - (1-p_i)^{n_j}) f(j+1,i,\mathbf{n}) \ .
\end{aligned}$$

To see this, observe that $((1-p_{i+1})^{n_j} - (1-p_i)^{n_j})$ is the probability that the min rank obtained in step j is between p_i and p_{i+1}, and that $(1-p_{t+1})^{n_j}$ is the probability that the min rank is larger than p_{t+1}. We rewrite as follows to simplify the computation:

$$\begin{aligned}
f(j,t,\mathbf{n}) &= f(j,t+1,\mathbf{n}) - \\
&\quad (1-p_{t+1})^{n_j} (f(j+1,t+1,\mathbf{n}) - f(j+1,t,\mathbf{n})) \ .
\end{aligned}$$

$q[\text{SUFF}(j,i_j)|\mathbf{n}]$ is the probability that the first $n_1+\cdots+n_{j-1}+(n_j-i_j)$ packets have rank value above p_j, times the (independent) probability that the (n_j-i_j+1)th packet has rank below p_j, times the sum, over $i > j$, of the probability that the min-rank value in step j is in $(p_{i+1}, p_i]$ multiplied by $f(j+1,i,\mathbf{n})$. Observe that the probability that the min-rank value in step k is at most p_{i+1} conditioned on the first $n_j - i_j - 1$ packets in the step having ranks strictly above p_j and the $(n_j - i_j)$th packet having rank at most p_j is $1-(1-p_i/p_j)(1-p_i)^{i_j-1}$. Therefore, under the same conditioning, the probability that the min rank value is in the interval $(p_{i+1}, p_i]$ is

$$((1-p_{i+1}/p_j)(1-p_{i+1})^{i_j-1} - (1-p_i/p_j)(1-p_i)^{i_j-1}) \ .$$

Therefore,

$$q[\text{SUFF}_{\mathbf{n}}(j,i_j)|\mathbf{n}] = (1-p_j)^{n_1+\cdots+n_{j-1}+(n_j-i_j)} p_j \tag{3}$$

$$\sum_{i=j+1}^{r} ((1-\frac{p_{i+1}}{p_j})(1-p_{i+1})^{i_j-1} - (1-\frac{p_i}{p_j})(1-p_i)^{i_j-1}) f(j+1,i,\mathbf{n}) \ .$$

The $f(j,t,\mathbf{n})$ values $(j \leq t \leq r)$ can be computed in time quadratic in the number of steps. Given the $f()$ values for $\mathbf{n}$, $q[\mathbf{s} = \text{SUFF}_{\mathbf{n}}(j,i_j)|\mathbf{n}]$ for each suffix $\mathbf{s} \preceq \mathbf{n}$ can be computed in time proportional to the number of steps.

5.3 Calculating the adjusted weights

The system of equations is triangular and can be solved by substitutions, starting from $A(\text{SUFF}_{\mathbf{n}}(r,1))$, yielding a unique solution. Computation of adjusted weights that directly utilizes the set of equalities above requires computing the adjusted weights for all suffixes of the observed count along with all the respective $q[]$ values. There is an equation for each suffix with number of variables that is equal to the number of suffixes of the respective suffix, and therefore, the number of operations is proportional to the product of the number of nonzero coordinates of $\mathbf{n}$ and the square of the number of observed packets $\sum_{h=1}^{r} n_i$. This quadratic dependence in the number of packets would make the computation very intensive for large flows. In subsequent work we provide a more efficient way to compute the adjusted weights [1]:

THEOREM 5.1. *[1] The adjusted weight $A^{\text{SSH}}_{\mathbf{p}}(\mathbf{n})$ can be computed using number of operations that is quadratic in the number of steps with a non-zero count.*

The number of operation is quadratic in the number of steps (which is logarithmic in the number of packets.) In fact, the number of operations is quadratic in the number of steps where the flow has a non-zero count. This distinction is important since many flows, in particular bursty or small flows, can have non-zero count on a single step or very few steps.

The expression of the adjusted weights uses the values $c_{i,j}(\mathbf{p},\mathbf{n})$ $(r \geq j \geq i \geq 1)$ defined as follows (the parameters $(\mathbf{p},\mathbf{n})$ are omitted when clear from context, and we assume $n_1 > 0$ wlog):

$$\begin{aligned}
r \geq j \geq 1: &\quad c_{1,j} = (1-p_j) \\
r \geq j \geq 2: &\quad c_{2,j} = (1-p_j)^{n_1-1}(c_{1,j} - c_{1,1}) \\
r \geq j \geq i \geq 3: &\quad c_{i,j} = (1-p_j)^{n_{i-1}}(c_{i-1,j} - c_{i-1,i-1})
\end{aligned}$$

- For $r \geq j \geq i \geq 2$, $c_{i,j}(\mathbf{p},\mathbf{n})$ is the probability that the flow $\mathbf{n}$ is fully counted by SSH until the transition into step i, and at the beginning of step i, the rank of the flow is at least p_j.
- For $r \geq j \geq 1$, $c_{1,j}$ is the probability that the first packet of the flow obtains rank value at least p_j.

Therefore, $c_{i,i}$ $(i \in \{1,\ldots,r\})$ is the probability that the SSH counting of the flow progressed continuously from the start until the transition into step i, and halted in this transition (as the current rank of the flow was above p_i.). Therefore,

$$q[\mathbf{n}|\mathbf{n}] = 1 - \sum_{h=1}^{r} c_{h,h} \ . \tag{4}$$

The following theorem expresses the adjusted weight $A(\mathbf{n})$ as a function of the diagonal sums $\sum_{h=1}^{i} c_{h,h}$ $(h = 1,\ldots,r)$.

THEOREM 5.2. *[1]*

$$A(\mathbf{n}) = \frac{(1-p_1) + \sum_{i=1}^{r} n_i (1 - \sum_{h=1}^{i} c_{h,h})}{1 - \sum_{h=1}^{r} c_{h,h}} \ .$$

6. ESTIMATING OTHER AGGREGATES

In applications, including IP traffic, packets can have different weights. Our presentation so far considered the unweighted version of the problem where all packets have uniform weights. The sketching algorithms can be easily adapted to estimate bytes instead of packets: The count values are applied to bytes and are captured as follows. For ASH or SSH, is a packet belongs to a cached flow, the number of bytes is added to the active counter. Otherwise, we use the geometric distribution to determine what part of a packet (if at all) should be counted. For a continuous variant of this process we can use the exponential distribution.

6.1 Unbiased FSD and flow count estimators

A flow attribute that is lost when there is no pre-aggregation is the size of the flow (exact number of packets or bytes). This could be an important attribute for some aggregations, for example, to trace the origin of port scanning or worm activity we may want to aggregate over all flows that originate from a certain AS and have at most 10 packets. We also may want to estimate the number of flows in a subpopulation that are within a certain range of sizes.

FSD estimation can be viewed as assigning *adjusted FSD estimators* $\alpha_i^L()$ $(i \geq 1)$ such that for any flow f, $E(\alpha_{|f|}^L(f)) = 1$ and for $i \neq |f|$, $E(\alpha_i^L(f)) = 0$.

As with adjusted weights, an unbiased estimator of the number of flows of size i in a subpopulation J is $\sum_{f \in J} \alpha_i^L(f)$. By combining, we can obtain unbiased estimates for the number of flows in any range or set of sizes in the subpopulation.

A special case of interest is the total number of flows in a subpopulation, which can be estimated using *adjusted counts* $\#^L()$ (for any flow f, $E(\#^L(f)) = 1$). We can obtain adjusted counts from adjusted FSD estimators using $\#^L(\mathbf{p}, \mathbf{n}) = \sum_i \alpha_i^L(\mathbf{p}, \mathbf{n})$.

$L \in \{\text{NF}, \text{ANF}\}$. Let O_i be the random variable representing the number of flows of count i with NF (or ANF). Let p be the sampling rate (or effective sampling rate), and let C_i be the number of flows of weight i.

For $i \geq 1$, the expectation of O_i is $p^i C_i + i(1-p)p^{i+1}C_{i+1} + \cdots = \sum_{j \geq i} \binom{j}{i} p^i (1-p)^{j-i} C_j$. Therefore, the inverse of the matrix with entries $\binom{j}{i} p^i (1-p)^{j-i}$, expresses each C_j as a linear combination of $E(O_i)$'s, and provides unbiased estimators [13, 7]. The resulting estimators, however, are not well behaved [7]. Better estimators that use the TCP syn flag were proposed in [7]. The coefficients of this inverted matrix are the FSD estimators $\alpha_j^L(p, i)$.

$L \in \{\text{SH}, \text{ASH}\}$. We use a similar derivation for adjusted FSD estimators for SH and ASH. These estimators can assume negative values, but are well behaved. Let O_i be the random variable representing the number of flows of count i, p is the sampling rate, and C_i be the number of flows of weight i.

LEMMA 6.1. *$\hat{C}_i = O_i/p - O_{i+1}(1-p)/p$ is an unbiased estimate of the number of flows of size i.*

PROOF. The expectation of O_i is $C_i p + C_{i+1}(1-p)p + C_{i+2}(1-p)^2 p + \cdots = \sum_{j \geq i} C_j (1-p)^{j-i} p$. Therefore, the expectation of $O_i/p - O_{i+1}(1-p)/p$ is C_i. $\square$

The respective nonzero coefficients are $\alpha_j^L(p, j) = 1/p$ and $\alpha_j^L(p, j+1) = -(1-p)/p$ for $j \geq 1$.

The estimator for the total number of flows is $O_1/p + \sum_{i>1} O_i$, which corresponds to the adjusted counts $\#^{\text{ASH}}(p, 1) = 1/p$ and $\#^{\text{ASH}}(p, 1) = 1$.

SSH. The SSH adjusted FSD estimators are more accurate (have smaller variance) than those obtained using SH and ASH sketches. Let $O_{\mathbf{s}}$ be the random variable representing the number of flows with observed count vector $\mathbf{s}$.

LEMMA 6.2. *Let $\mathbf{n}$ be the observed count and $\mathbf{p} = (p_1, \ldots, p_r)$ the corresponding sampling rate steps. Assume WLOG that $n_\ell > 0$ for $1 \leq \ell < r$. The following are correct adjusted FSD estimators $\alpha_i^{\text{SSH}}(\mathbf{p}, \mathbf{n})$ $(i \geq 1)$. We state only the nonzero values.*

- *If $|\mathbf{n}| = 1$, $\alpha_1^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = 1/p_r$. Otherwise,*
- *If $n_r = |\mathbf{n}| > 1$ (there is only one step and it is the last one) then $\alpha_{|\mathbf{n}|}^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = 1/p_r$ and $\alpha_{|\mathbf{n}|-1}^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = -(1-p_r)/p_r$.*
- *For $\ell = 3, \ldots, r-1$, for $\ell = 2$ if $n_1 > 1$, and for $\ell = r$ if $n_r > 0$:*
$$\alpha_{\sum_{h=\ell}^r n_h}^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = \frac{-c_{\ell,\ell}}{q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]} .$$
- *If $n_1 = 1$, $\alpha_{\sum_{\ell=2}^r n_h}^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = \frac{-(1-p_2)}{q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]}$.*
- *If $n_1 > 1$ then $\alpha_{\mathbf{n}, -1+\sum_{h=1}^r n_h}(\mathbf{p}, \mathbf{n}) = \frac{-(1-p_1}{q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]}$*
- $\alpha_{\sum_{h=1}^r n_h}^{\text{SSH}} = \frac{1}{q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]}$

PROOF. The proof is along the same lines of Lemma 7.7. We compute the expectation of the ASH FSD estimator conditioned on the SSH observed counts and sampling rate steps. Therefore, we also establish that the SSH-based estimator has at most the same variance as the ASH estimator.

Denote by $\pi(\mathbf{p}, \mathbf{n}, j)$ the conditional probability of an ASH count j given SSH count $\mathbf{n}$ and steps $\mathbf{p}$. For all $i \geq 1$,

$$\begin{aligned} \alpha_i^{\text{SSH}}(\mathbf{p}, \mathbf{n}) &= \sum_{j=1}^{|\mathbf{n}|} \pi(\mathbf{p}, \mathbf{n}, j) \alpha_i^{\text{ASH}}(p_r, j) \\ &= \frac{\pi(\mathbf{p}, \mathbf{n}, i)}{p_r} - \frac{\pi(\mathbf{p}, \mathbf{n}, i+1)(1-p_r)}{p_r} \end{aligned}$$

If $|\mathbf{n}| = 1$ then $\pi(\mathbf{p}, \mathbf{n}, 1) = 1$ and $\pi(\mathbf{p}, \mathbf{n}, j) = 0$ for $j \neq 1$ (the ASH count is 1 with probability 1). Therefore $\alpha_1^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = 1/p_r$ (and $\alpha_j^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = 0$ for all $j \neq 1$).

If $|\mathbf{n}| = n_r$ then $\pi(\mathbf{p}, \mathbf{n}, n_r) = 1$ and $\pi(\mathbf{p}, \mathbf{n}, j) = 0$ for $j \neq n_r$ (the ASH count is n_r with probability 1). Therefore, the only nonzero $\alpha_j^{\text{SSH}}(\mathbf{p}, \mathbf{n})$ are $\alpha_{n_r}^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = 1/p_r$ and $\alpha_{n_r+1}^{\text{SSH}}(\mathbf{p}, \mathbf{n}) = -(1-p_r)/p_r$.

Consider $\mathbf{n}$ such that $|\mathbf{n}| > 1$ and $|\mathbf{n}| > n_r$. Assume WLOG that $n_\ell > 0$ for $1 \leq \ell < r$. The probabilities $\pi(\mathbf{p}, \mathbf{n}, x) > 0$ are positive if only if $|\mathbf{n}| \geq x \geq n_r + 1$ and are as follows:

$$\ell = 2, \ldots, r-1 \text{ and } \sum_{h=\ell+1}^r < x \leq \sum_{h=\ell}^r n_h :$$
$$\pi(\mathbf{p}, \mathbf{n}, x) = \frac{(c_{\ell,r} - c_{\ell,\ell})(1-p_r)^{\sum_{h=\ell}^r n_h - x} p_r}{q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]} .$$
$$\sum_{h=2}^r n_h < x < \sum_{h=1}^r n_h :$$
$$\pi(\mathbf{p}, \mathbf{n}, x) = \frac{(c_{1,r} - c_{1,1})(1-p_r)^{\sum_{h=1}^r n_h - 1 - x} p_r}{q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]} .$$
$$x = |\mathbf{n}| :$$
$$\pi(\mathbf{p}, \mathbf{n}, |\mathbf{n}|) = p_r / q^{\text{SSH}}[\mathbf{n}|\mathbf{n}] .$$

Therefore, $\alpha_i^{\text{SSH}}(\mathbf{p}, \mathbf{n})$ can be nonzero only for $|\mathbf{n}| \geq i \geq n_r$.

We compute these values case by case:

i **such that** $\sum_{j=\ell+1}^r n_j < i < \sum_{j=\ell}^r n_j$ **for some** $2 \leq \ell \leq r-1$**:** (ASH count of i begins in the second or later step and does not start with the first SSH-counted packet of the step.)

$$\begin{aligned}
&\alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n})\\
&= \frac{\pi(\mathbf{p},\mathbf{n},i)}{p_r} - \frac{\pi(\mathbf{p},\mathbf{n},i+1)(1-p_r)}{p_r}\\
&= \frac{(c_{\ell,r}-c_{\ell,\ell})(1-p_r)^{\sum_{h=\ell}^{r} n_h - i}p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&\quad - \frac{(1-p_r)(c_{\ell,r}-c_{\ell,\ell})(1-p_r)^{\sum_{h=\ell}^{r} n_h - i-1}p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&= 0
\end{aligned}$$

if $n_1 \geq 3$, i such that $\sum_{j=2}^{r} n_j < i < \sum_{j=1}^{r} n_j - 1$: (ASH count of i begins in the third or later packet of the first step.)

$$\begin{aligned}
&\alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n})\\
&= \frac{\pi(\mathbf{p},\mathbf{n},i)}{p_r} - \frac{\pi(\mathbf{p},\mathbf{n},i+1)(1-p_r)}{p_r}\\
&= \frac{(c_{1,r}-c_{1,1})(1-p_r)^{\sum_{h=1}^{r} n_h - i-1}p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&\quad - \frac{(1-p_r)(c_{1,r}-c_{1,1})(1-p_r)^{\sum_{h=1}^{r} n_h - i-2}p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&= 0
\end{aligned}$$

$i = \sum_{j=\ell}^{r} n_j$ for some $2 \leq \ell \leq r$; If $\ell = 2$ we assume $n_1 > 1$, If $\ell = r$ we assume $n_r > 0$ because otherwise $\alpha_{n_r}()$ is not defined.:

$$\begin{aligned}
&\alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n})\\
&= \frac{\pi(\mathbf{p},\mathbf{n},i)}{p_r} - \frac{\pi(\mathbf{p},\mathbf{n},i+1)(1-p_r)}{p_r}\\
&= \frac{(c_{\ell,r}-c_{\ell,\ell})p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&\quad - \frac{(1-p_r)(c_{\ell-1,r}-c_{\ell-1,\ell-1})(1-p_r)^{n_\ell-1}p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&= \frac{-c_{\ell,\ell}}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]}
\end{aligned}$$

$i = \sum_{j=2}^{r} n_j$, $n_1 = 1$:

$$\begin{aligned}
&\alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n})\\
&= \frac{\pi(\mathbf{p},\mathbf{n},i)}{p_r} - \frac{\pi(\mathbf{p},\mathbf{n},i+1)(1-p_r)}{p_r}\\
&= \frac{(c_{2,r}-c_{2,2})p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r} - \frac{(1-p_r)p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&= \frac{-(1-p_2)}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]} \equiv \frac{-c_{1,1}-c_{2,2}}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]}
\end{aligned}$$

$i = -1 + \sum_{j=1}^{r} n_j$, $n_1 > 1$:

$$\begin{aligned}
&\alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n})\\
&= \frac{\pi(\mathbf{p},\mathbf{n},i)}{p_r} - \frac{\pi(\mathbf{p},\mathbf{n},i+1)(1-p_r)}{p_r}\\
&= \frac{(c_{1,r}-c_{1,1})p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r} - \frac{(1-p_r)p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r}\\
&= \frac{-(1-p_1)}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]} = \frac{-c_{1,1}}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]}
\end{aligned}$$

$i = \sum_{j=1}^{r} n_j$:

$$\begin{aligned}
\alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n}) &= \frac{\pi(\mathbf{p},\mathbf{n},i)}{p_r} - \frac{\pi(\mathbf{p},\mathbf{n},i+1)(1-p_r)}{p_r}\\
&= \frac{p_r}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]p_r} - 0 = \frac{1}{q^{\mathrm{SSH}}[\mathbf{n}|\mathbf{n}]}
\end{aligned}$$

□

Using the relation $\#^{\mathrm{SSH}}(\mathbf{p},\mathbf{n}) = \sum_{i\geq 1} \alpha_i^{\mathrm{SSH}}(\mathbf{p},\mathbf{n})$ we obtain

- If $|\mathbf{n}| = 1$, $\#^{\mathrm{SSH}}(\mathbf{p},\mathbf{n}) = 1/p_r$.
- If $n_r \geq 1$ and $|\mathbf{n}| > 1$, $\#^{\mathrm{SSH}}(\mathbf{p},\mathbf{n}) = 1$.
- If $n_r = 0$, $|\mathbf{n}| > 1$, $\#^{\mathrm{SSH}}(\mathbf{p},\mathbf{n}) = 1 + c_{r,r}/q[\mathbf{n}|\mathbf{n}]$.

Adjusted FSD estimators are computed based on the step-counts and therefore to facilitate FSD estimation, they need to be computed before the step-counts are discarded.

7. QUALITATIVE COMPARISON

We compare the estimates obtained using ANF, ASH, and SSH. We evaluate performance as a function of the size k of the sketch (number of flows). For these methods, the number of active counters is equal to the size of the sketch.

We also consider methods that are applied to pre-aggregated traffic: priority sampling [8] (PRI) and weighted sampling without replacement (WS) with sample size k. [2] The adjusted weights for WS are obtained using the rank conditioning method [2, 3]. The adjusted weight assigned to each flow is equal to its weight (number of packets) divided by the probability that the flow is included in the sample[3] in some probability subspace that includes the current sample. The probability subspace is defined as all runs that have the same effective sampling rate p' and therefore the probability is equal to $1-(1-p')^{|f|}$, where $|f|$ is the number of packets in the flow and p' is as defined in Lemma 3.1. Therefore, the adjusted weight is equal to $|f|/(1-(1-p')^{|f|})$.

7.1 Distribution of flows included in sketch

We consider the distribution over subsets of flows that are included in the sketch.

LEMMA 7.1. *The subset of flows included in the sketch produced by* ANF, ASH, *and* SSH *is a weighted sample without replacement of size k from the set of flows.*

PROOF. We use the rank-based view of these algorithms. The set of flows selected in each of these methods are those with rank that is below the effective sampling rate. (The set of flows with k smallest rank values). To see that this is a weighted sample without replacement, consider a monotone conversion to exponentially distributed ranks. The rank of a flow is then an exponential random variable with parameter that is the number of packets. The k smallest ranked flows constitute a weighted sample of size k without replacement (see, for example, [4]). □

PRI results in a different distribution that can not be obtained without pre-aggregation.

Since these algorithms share the same distribution, the difference in estimate accuracy stems from the adjusted weight assignment. The quality of the assignment depends on the information the algorithm gathers and the method we apply to derive the adjusted weights. When the adjusted weights have smaller variance, the estimates we obtain are more accurate.

[2] WS and PRI are not applicable to unaggregated streams, but WS can be implemented in two passes over the data and therefore can be applied on some unaggregated data.

[3] This is the Horvitz-Thompson unbiased estimator obtained by dividing the weight of the item by the probability that it is sampled.

7.2 Variance of adjusted weights

We use the notation $a^{\mathrm{L}}(f)$, where L $\in$ {ASH,ANF,SSH,NF,SH}, for the random variable that is the adjusted weight assigned to the flow f by the algorithm L. A convenient property of the adjusted weight assignment by these algorithms is that the covariances are zero. The zero covariance property is trivial for fixed-rate sampling (NF or SH), since each flow is selected independently. With fixed sample size, however, the adjusted weights are not independent since inclusion of one flow makes it less likely for the other flow to be included. For WS, this property is established in [2]. For PRI, it is established in [8].

LEMMA 7.2. *Consider $L \in$ {ASH,ANF,SSH} and two flows $f_1 \neq f_2$. Then* $\text{COV}(a^{L}(f_1), a^{L}(f_2)) = 0$.

PROOF. It suffices to show that $E(a(f_1)a(f_2)) = w(f_1)w(f_2)$ (we omit the superscript L). We adapt the proof method in [2] used to establish zero covariance for rank conditioning estimators. We partition the sample space according to the $(k-1)$th smallest rank value among the flows in $F \setminus \{f_1, f_2\}$. Consider one part and let r_{k-1} be that rank value. The product $a(f_1)a(f_2)$ is positive only when $r(f_1) < r_{k-1}$ and $r(f_2) < r_{k-1}$ (it is zero otherwise, since at least one of f_1 or f_2 is not included in the sketch). In this case, the effective sampling rate is equal to r_{k-1}. The count obtained for f_1 using ASH or ANF only depends on r_{k-1} and not on the count of f_2. Therefore $a(f_1)$ and $a(f_2)$ are independent conditioned on them both having ranks below r_{k-1}. The expectation of $a(f_i)$ under this conditioning is $w(f_i)/\text{PR}\{r(f_i) < r_{k-1}\}$, and the conditional expectation

$$\begin{aligned} & E(a(f_1)a(f_2)|(r(f_1) < r_{k-1}) \wedge (r(f_2) < r_{k-1})) \quad (5) \\ = & \frac{w(f_1)w(f_2)}{\text{PR}\{r(f_1) < r_{k-1}\}\text{PR}\{r(f_2) < r_{k-1}\}} . \end{aligned}$$

Therefore, on this part

$$\begin{aligned} & E(a(f_1)a(f_2)) \quad (6) \\ = & \frac{\text{PR}\{r(f_1) < r_{k-1}\}\text{PR}\{r(f_2) < r_{k-1}\}w(f_1)w(f_2)}{\text{PR}\{r(f_1) < r_{k-1}\}\text{PR}\{r(f_2) < r_{k-1}\}} \\ = & w(f_1)w(f_2) . \end{aligned}$$

We next consider SSH. We use the following property: fix the step function $\mathbf{p}$ of the current sampling rate and let p_r be the effective sampling rate. The expectation of $a^{\mathrm{SSH}}(f_1)$ conditioned on $r(f_1) < p_r$ is equal to $w(f_1)/\text{PR}\{r(f_1) < p_r\}$. We partition the sample space according to the step functions of r_{k-1} and r_k (the kth and $(k-1)$th smallest current rank value among the flows in $F \setminus \{f_1, f_2\}$). Consider a part in this partition. The product $a(f_1)a(f_2)$ is positive only if $r(f_1) < r_{k-1}$ and $r(f_2) < r_{k-1}$. In this case, the current sampling rate is r_{k-1}. Fix the ranks of f_2 packets. The current sampling rate is determined and it is a step function $\mathbf{p}$ with effective sampling rate $p_r = r_{k-1}$. The conditioned expectation of $a(f_1)$ in this part after fixing f_2 ranks, and given that it has $r(f_1) < r_{k-1}$ is $w(f_1)/\text{PR}\{r(f_1) < r_{k-1}\}$. It is independent of the f_2 ranks and therefore, the adjusted weight $a(f_1)$ and $a(f_2)$ in this part, conditioned on them both having ranks below r_{k-1}, are independent. Hence, the conditioned expectation of the product is as in Eq. 5 and therefore Eq. 6 also holds. □

The zero covariance property of the random variables $a^{\mathrm{L}}(f)$ $(f \in F)$ implies that

COROLLARY 7.3. *For any $J \subset F$ and $L \in$ {ASH,ANF,SSH},* $\text{VAR}(a^{L}(J)) = \sum_{f \in J} \text{VAR}(a^{L}(f))$.

Therefore, to show that an adjusted weight assignment has lower variance than another on *all* subpopulations, it suffices to show lower variance on all individual flows.

An algorithm dominates another if we can use its output to emulate an output of the other algorithm. It is not hard to see that SSH dominates ASH, that ASH dominates ANF, and that they are all dominated by WS. The following theorem shows that the variance of the adjusted weight assignments reflects this dominance relation, with the more dominant methods having lower variance. The proof is contained in the remainder of this section.

THEOREM 7.4. *For any packet stream and any flow f we have the following relation between the variance of the adjusted weight assignment for f.*

$$\text{VAR}(a^{\mathrm{WS}}(f)) \leq \text{VAR}(a^{\mathrm{SSH}}(f)) \quad \leq \quad \text{VAR}(a^{\mathrm{ASH}}(f)) \leq \text{VAR}(a^{\mathrm{ANF}}(f))$$

Consider a flow f with $|f|$ packets and the probability subspace where ranks of packets belonging to all other flows $(F \setminus \{f\})$ are fixed. It is sufficient to establish the relation between the methods in this subspace. Consider such a subspace. Let $\mathbf{p}$ be the steps of the effective sampling rate and p_r be the final effective sampling rate. The adjusted weight assignment for all methods has expectation $|f|$ within each such subspace. We consider the variance of the different methods within such subspace. and use the notation $\text{VAR}(a^{\mathrm{SSH}}(f)|\mathbf{p})$, and $\text{VAR}(a^{\mathrm{L}}(f)|p_r)$ for $L \in$ {WS,ANF,ASH}.

The variance for ANF is that of a binomial random variable and therefore is

$$\text{VAR}(a^{\mathrm{ANF}}(f)|p_r) = |f|(1-p_r)/p_r . \qquad (7)$$

For WS, the adjusted weight is $|f|/(1-(1-p_r)^{|f|})$ with probability $1-(1-p_r)^{|f|}$ and zero otherwise, and therefore, the variance is

$$\text{VAR}(a^{\mathrm{WS}}(f)|p_r) = |f|^2(1/(1-(1-p_r)^{|f|})-1) . \qquad (8)$$

For ASH, it is

$$\begin{aligned} \text{VAR}(a^{\mathrm{ASH}}(f)|p_r) &= \sum_{i=0}^{|f|-1}(1-p_r)^i p_r(|f|-i+(1-p_r)/p_r)^2-|f|^2 \\ &= ((1-p_r)-(1-p_r)^{|f|+1})/p_r^2 . \qquad (9) \end{aligned}$$

For SSH the variance $\text{VAR}(a^{\mathrm{SSH}}(f)|\mathbf{p})$ depends not only on $|f|$ and p_r but on the steps $\mathbf{p}$ and the way the packets of the flow f are distributed across these steps. The variance is lowest when all packets occur when the sampling probability is highest, and the variance is highest, and equal to that of ASH, when all packets occur on the step with the lowest sampling probability.

Using the explicit expressions (Eq. 7,8,9) and the inequality $(1-p)^n \geq 1-np$ for all natural n and $0 \leq p \leq 1$ it follows that $\text{VAR}(a^{\mathrm{WS}}(f)|p_r) \leq \text{VAR}(a^{\mathrm{ASH}}(f)|p_r) \leq \text{VAR}(a^{\mathrm{ANF}}(f)|p_r)$.

We establish the relation between the variance of the different methods via direct arguments that provide more insights and intuition and allow us to consider $\text{VAR}(a^{\mathrm{SSH}}(f)|\mathbf{p})$.

Each algorithm, data stream, and flow, can be viewed as a mapping from the space of all possible rank assignments to packets into nonnegative adjusted weight values.

LEMMA 7.5. *Consider two mappings A_1 and A_2 and suppose there exists a partition of the sample space S into subspaces such that within each subspace $S' \subset S$,*

- *$\mu_{S'}(A_1) = \mu_{S'}(A_2)$ (that is, A_1 and A_2 have the same expectation on the subspace S'), and*
- *$\mu^2_{S'}(A_1) \geq \mu^2_{S'}(A_2)$ (the variance, or alternatively, the second raw moment, of A_1 is at least as large as A_2).*

Then $\mu_S(A_1) = \mu_S(A_2)$ and $\mu^2_S(A_1) \geq \mu^2_S(A_2)$ (A_1 and A_2 have the same expectation and the variance of A_1 is at least as large as A_2.)

COROLLARY 7.6. *Let A_1 be an estimator and consider a partition of the sample space. Consider the estimator A'_1 that has a value that is equal to the expectation of A_1 on the partition. Then $\mu(A_1) = \mu(A'_1)$ and $\mu^2(A_1) \geq \mu^2(A'_1)$.*

If f is not included in the sample ($r(f) > p_r$), it obtains an adjusted weight of zero with all four methods. Therefore, it suffices to compare the methods based on the variance in the adjusted weight assignment within the probability subspace when the flow is sampled ($r(f) \leq p_r$). Since all methods are unbiased, they all have the same expectation on this subspace. We apply Lemma 7.5. With WS, f obtains an adjusted weight of $|f|/(1-(1-p_r)^{|f|})$, which is fixed, therefore the variance is zero. Therefore, this assignment is optimal among all methods that yield the same probability distributions over subsets of flows. In particular, $\text{VAR}(a^{\text{WS}}(f)|p_r) \leq \text{VAR}(a^{\text{SSH}}(f)|\mathbf{p})$.

We next compare ANF and ASH. We further partition the probability subspace by fixing the position $i \in [1,\ldots,|f|]$ of the first packet in f that obtains rank that is at most p_r. The adjusted weight assigned by ASH is (a fixed value) of $(1/p_r)+(|f|-i)$, and therefore the variance is zero. ANF assignment is $(1/p_r)$ plus $(1/p_r)$ times a binomial random variable with parameters $(|f|-i)$ and p_r. This assignment has the same conditional expectation as the ASH assignment within this subspace, but also has a nonnegative, and therefore at least as large, variance (the variance is strictly positive when $i < |f|$). Therefore, using Lemma 7.5, ASH has a smaller variance overall than ANF.

We compare the variance of SSH and ASH by again applying Lemma 7.5. (Observe that we can not use the same partition that we used for comparing ASH and ANF, since SSH does not have the same expectation as ASH on each such subspace.) We partition the sample space according to the *suffix of the packets of f that are counted using* SSH. That is, each subspace contains all rank assignments to packets of f that result in this suffix being counted. The adjusted weight assigned by SSH is fixed within each partition and therefore has zero variance. The ASH adjusted weight depends on the first packet that obtains rank value below p_r, and therefore varies. What remains to show is that ASH and SSH adjusted weights have the same expectation within each such subspace.

We apply the following simple observation. Consider a flow with counts $\mathbf{n}$ and a vector $\mathbf{s} \preceq \mathbf{n}$. The conditional probability that i packets are counted using ASH given that $\mathbf{s}$ is counted using SSH is independent of the choice of $\mathbf{n}$ (depends only on $\mathbf{s}$). Therefore, the expectation of the adjusted weight assigned by ASH conditioned on SSH counting $\mathbf{s}$ out of $\mathbf{n}$ is equal to the expectation when SSH counts $\mathbf{s}$ out of $\mathbf{s}$. This simplifies the proof, as it suffices to establish equality for flows that are fully counted by SSH:

LEMMA 7.7. *Consider a flow with counts $\mathbf{n}$. The conditional expectation of the adjusted weight assigned to the flow by* ASH*, given that the flow is fully counted by* SSH*, is equal to $A(\mathbf{n})$ (the adjusted weight assigned by* SSH *for observed count $\mathbf{n}$).*

PROOF. We compute the expectation of the adjusted weight of ASH when considered nonzero only on rank assignments such that SSH fully counts the flow. We refer to this expectation as the *combined expectation*. The conditional expectation we seek is then the ratio of the combined expectation and $q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]$. Therefore, we need to show that the combined expectation is equal to

$$q^{\text{SSH}}[\mathbf{n}|\mathbf{n}]A^{\text{SSH}}(\mathbf{n}) \ .$$

To facilitate that, we compute, for each step $r-1 \geq \ell > 1$ the "contribution" to the combined expectation if ASH started counting the flow during step ℓ.[4] The probability that at the beginning of the step the rank value was at least p_r is $(c_{\ell,r} - c_{\ell,\ell})$. Conditioned on that, the contribution to the expectation is

$$\begin{aligned} &\sum_{t=0}^{n_\ell - 1}(1-p_r)^t p_r \left(\sum_{h=\ell}^{r} n_h + (1-p_r)/p_r - t\right) \\ =\ & \sum_{h=\ell}^{r} n_h - (1-p_r)^{n_\ell} \sum_{h=\ell+1}^{r} n_h \ . \end{aligned} \quad (10)$$

Therefore, the contribution is the product

$$\begin{aligned} &(c_{\ell,r} - c_{\ell,\ell})(\sum_{h=\ell}^{r} n_h - (1-p_r)^{n_\ell} \sum_{h=\ell+1}^{r} n_h) \\ =\ & (c_{\ell,r} - c_{\ell,\ell}) \sum_{h=\ell}^{r} n_h - c_{\ell+1,r} \sum_{h=\ell+1}^{r} n_h \ . \end{aligned} \quad (11)$$

We need to be slightly careful with the first step by considering the rank of the first packet and then the other $n_1 - 1$ packets. With probability p_r, the first packet obtains rank value at most p_r and ASH uses the adjusted weight $\sum_{h=1}^{r} n_h + (1-p_r)/p_r$ and therefore the contribution is the product

$$(1-p_r) + p_r \sum_{h=1}^{r} n_h \ . \quad (12)$$

With probability $p_1 - p_r \equiv c_{1,r} - c_{1,1}$, the first packet obtains rank value in $(p_r, p_1]$, and applying a similar derivation as Eq. 10 for the remaining $n_1 - 1$ packets, we obtain $-1 + \sum_{h=1}^{r} n_h - (1-p_r)^{n_1 - 1} \sum_{h=2}^{r} n_h$. The contribution is therefore,

$$p_r - p_1 + (c_{1,r} - c_{1,1}) \sum_{h=1}^{r} n_h - c_{2,r} \sum_{h=2}^{r} n_h \ . \quad (13)$$

Summing the contributions of all the steps in Eq. 11, Eq. 12, and Eq. 13 we obtain that this expectation is

$$\begin{aligned} &(1-p_r) + p_r \sum_{h=1}^{r} n_h + p_r - p_1 + (c_{1,r} - c_{1,1}) \sum_{h=1}^{r} n_h \\ -\ & c_{2,r} \sum_{h=2}^{r} n_h + \sum_{\ell=2}^{r-1} \left((c_{\ell,r} - c_{\ell,\ell}) \sum_{h=\ell}^{r} n_h - c_{\ell+1,r} \sum_{h=\ell+1}^{r} n_h \right) \\ =\ & p_1 n_1 + (1-p_1) + \sum_{\ell=2}^{r}(1 - \sum_{h=1}^{\ell} c_{h,h}) n_\ell = q^{\text{SSH}}[\mathbf{n}|\mathbf{n}] A^{\text{SSH}}(\mathbf{n}) \ . \end{aligned}$$

(Using Theorem 5.2 and Eq. 4.) □

Lemma 7.7 constitutes an alternative derivation of the adjusted weights $A^{\text{SSH}}(\mathbf{n})$ (Theorem 5.2). Unbiasedness follows from Corollary 7.6 and the unbiasedness of the adjusted weights of ASH.

7.3 Bounds on the number of steps

The number of steps (adaptations of the sampling rate) affects performance of the adaptive algorithms. For SSH, the size of secondary memory used to temporarily store the count vectors and the final computation of the adjusted weights depend on the number of steps in which an actively-counted flow had a nonzero count.

[4] If $r > 1$, step r has contribution zero (for SSH to fully count the flow the rank must be at most p_r before the beginning of the step.)

LEMMA 7.8. *For a packet stream of size m and flow cache of size k, the expected number of rate adaptations is* $\leq (k+1)\ln m$.

PROOF. The expected number of updates to the set of cached flows for aggregated data streams is analyzed in [2, 5]. This corresponds to a situation where each flow appears as a consecutive burst and is therefore a special case of our model. The argument used in [5] for uniform weights (a stream of 1-packet flows) trivially extends to a model where there is a stream of 1-packet flows where rank values of cached flows (flows included in the current sketch) can be arbitrarily decreased at arbitrary times.

We now express an unaggregated stream of multi-packet flows in this model: Packets are processed as if they belong to a stream of 1-packet flows. Once a flow is cached, we examine the rank value of subsequent packets that belong to the flow and then delete these packets. If the rank of the deleted packet is smaller than the current rank of its flow, we simulate an arbitrary rank decrease and decrease the rank of the flow to that of the packet. An important point for the analysis is that packet deletion is independent of the rank of the deleted packet. The probability that the ith undeleted packet modifies the sketch (has rank value that is smaller than the $(k+1)$st smallest rank) is at most $\min\{1,(k+1)/i\}$. The total number of undeleted packets is at most m. Therefore, the expected number of updates is at most $\sum_{i=1}^{m}\min\{1,(k+1)/i\} \leq (k+1)\ln m$. □

This bound is tight for streams of 1-packet flows [5].

8. SIMULATIONS

We compare the accuracy of subpopulation-size estimates obtained using SNF, ASH, and SSH. We evaluate performance as a function of the size k of the sketch. For these three methods, the number of active counters is equal to the size of the sketch.

In the evaluation, we include two methods that are not applicable to unaggregated streams, PRI and WS, and a naive method that samples k packets (*k-packets*). The adjusted weight we assigned to each flow represented in the k-packets sample was $(i/k)w(F)$, where i is the number of packets of the flow that are sampled.

Data. We generate data sets using two Pareto distributions with $\alpha = 1.1$ and $\alpha = 1.5$. From each distribution, we generated a distribution of flow sizes by drawing 1000 items (flow sizes). The cumulative distributions of the weight of the top i flow sizes for each distribution is provided in Figure 1.

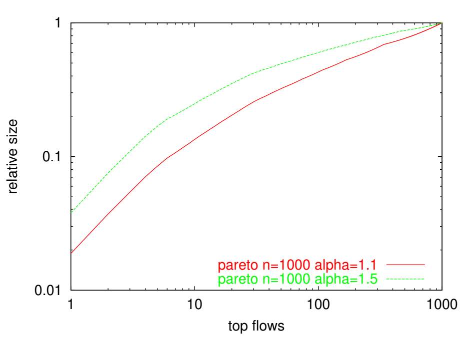


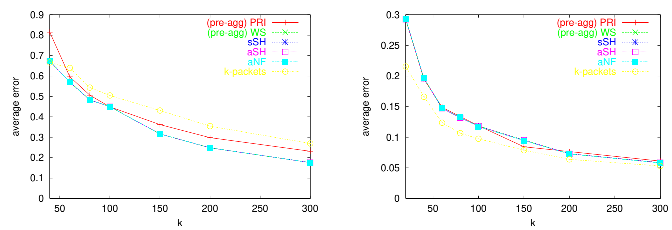
Figure 1: Cumulative distribution of flow sizes in our datasets.

The subsets (subpopulations) we consider are the i largest flows (for $i = 5, 10, 50$) and the i smallest flows (for $i = 100, 750$). This selection enables us to understand how performance depends on the consistency of the subset (many smaller flows or fewer larger flows) and the skew of the data.

Results The simulation results are provided in Figures 2 and 3. We evaluate the estimate accuracy by considering the averaged absolute error over 200 runs. The simulations show the relations we expected to see: WS outperforms SSH, which outperforms ASH, which in turn outperforms ANF.

For subpopulations consisting of large flows, the performance gaps are more pronounced. This is because on flows with frequency higher than the sampling rate, the more involved methods are able to collect additional information and utilize it to obtain adjusted weights with smaller variance. PRI performs best on these subpopulations since it is more likely to select the largest flows. SSH performance is closer to that of WS and is significantly better than ASH, therefore, SSH is the best performer on unaggregated streams.

For subpopulations consisting of smaller flows, the performance curves for the methods where the sketch is a weighted sample are very close. This is expected, since on small flows, the information collected is similar. For subpopulations with only tiny flows (bottom-100 flows), the performance is similar to k-sample, since the information collected is a single packet from each flow. PRI performs slightly worse than these methods, again, because of its bias towards sampling larger flows. Interestingly, the k-sample method slightly outperforms others in some cases. We discuss this in Section 9.



bottom-100 flows $\alpha = 1.1$ bottom-750 flows $\alpha = 1.5$

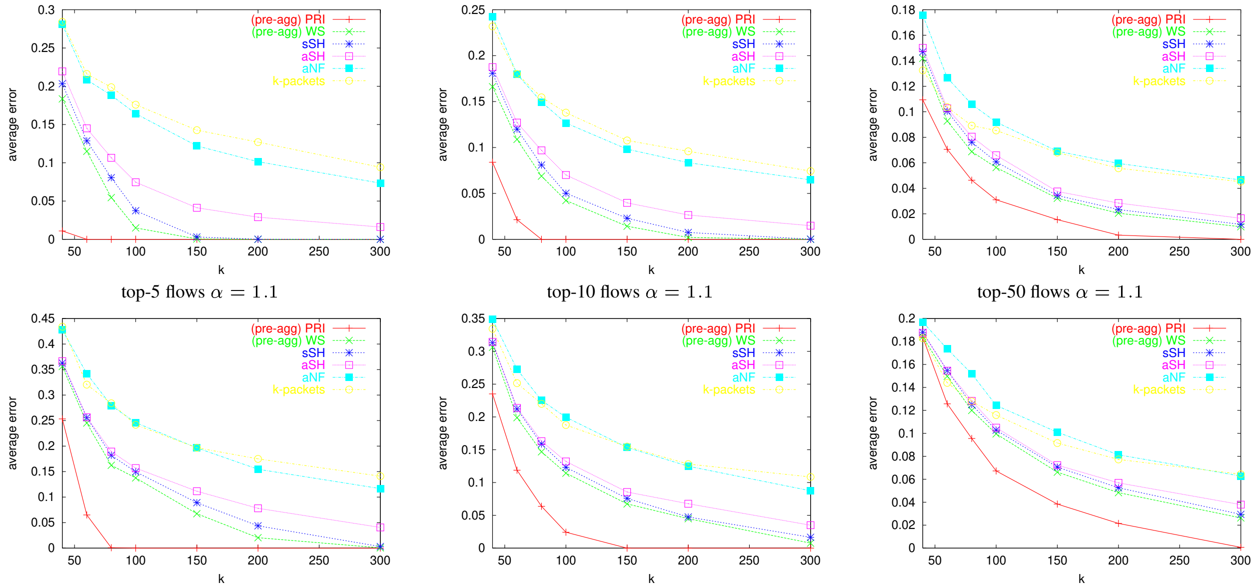
Figure 3: Pareto $\alpha = 1.1$ and $\alpha = 1.5$, estimating subpopulations consisting of the 100 and the 750 smallest flows.

9. CONCLUSION AND FURTHER WORK

We designed schemes to sketch unaggregated packet stream so that we can answer approximate queries of subpopulation sizes. The queries are posed over the aggregated view of the stream (that is, over the set of flows). The sketches consists of a set of flows and an adjusted weight assignment for each flow in the sketch. The adjusted weights have the property that for each flow, the expectation of the adjusted weight is equal to the size of the flow. Performance depends on the adjusted weight assignment and on the information collected by the algorithm to facilitate this assignment.

The schemes we consider are the pre-existing NF and SH, their adaptive variants ANF and ASH, and the newly proposed SSH. We carefully derive correct adjusted weights assignments for ANF, ASH, and SSH. The derivation requires some delicate considerations and particularly for SSH, is far from trivial. We compare the schemes by analyzing the per-flow variance. We show that for any stream and subset of flows, the variance of the adjusted weight of SSH is at most that of ASH, which is at most that of ANF. We use simulations that show that SSH is considerably more accurate than ASH which is considerably more accurate than ANF on realistic distributions. We conclude with a discussion of ongoing further work.

Using $w(F)$. We observed that the k-packets estimator is tighter than other methods on some subpopulations (small flows and good



top-5 flows $\alpha = 1.1$ · top-10 flows $\alpha = 1.1$ · top-50 flows $\alpha = 1.1$

top-5 flows $\alpha = 1.5$ · top-10 flows $\alpha = 1.5$ · top-50 flows $\alpha = 1.5$

Figure 2: Pareto $\alpha = 1.1$ and $\alpha = 1.5$, estimating subset consisting of top-5, top-10, and top-50 flows.

fraction of the total weight). This estimator uses a piece of information that the other estimators we consider do not use: the total weight $w(F)$. This estimator has higher per-flow variances than all other estimators we considered, but unlike the other methods, that have zero covariances, it has *negative covariances*, which means that the variance on estimating the weight of a subpopulation is smaller than the sum of variances. Therefore, it can obtain smaller variance than other methods on some subpopulations, in particular, $a(F) = w(F)$, and therefore $\text{VAR}(a(F)) = 0$. This understanding raises a question: when $w(F)$ is readily available, can we obtain estimators with comparable or better per-flow variances to our current estimators, negative covariances, and $a(F) \equiv w(F)$? We first observe that this can not be achieved for the fixed-rate algorithms (NF and SH): There is a positive probability of an empty sketch with $a(F) = 0$ and therefore we can not have $a(F) = w(F)$ on nonempty sketches and be unbiased. For the adaptive algorithms, a "corrected" version of ANF, where flow counts are scaled by the (inverse) fraction of packets that are sampled (instead of the inverse sampling rate) has the desired properties. We derive corrected versions for ASH and SSH. Interestingly, there are "corrected" estimator with these desirable properties for WS [2, 3] as well.

Deploying summarization schemes at routers [1]. SH and its variants use the same memory as the respective NF variants, but they require processing of every packet to determine if it belongs to a cached flow. Processing power is constrained as well as memory. We therefore (i) design step-counting NF (SNF), that processes the same number of packets and uses the same number of counters as ANF but is considerably more accurate, and (ii) we design *hybrid algorithms* that use two sampling rates, one for processing packets and one for creating new active flows. The hybrid algorithms provide a smooth tradeoff between NF and SH. Another implementation issue is the number of rate adaptations. We design schemes that instead of $O(k \log m)$ adaptations perform at most $\log m$ adaptations. These variants slightly under utilize memory but retain unbiasedness of the adjusted weights.

10. REFERENCES

[1] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Algorithms and estimators for accurate summarization of Internet traffic. Manuscript, 2007.

[2] E. Cohen and H. Kaplan. Bottom-k sketches: Better and more efficient estimation of aggregates. In *Proceedings of the ACM SIGMETRICS'07 Conference*, 2007. poster.

[3] E. Cohen and H. Kaplan. Sketches and estimators for subpopulation weight queries. Manuscript, 2007.

[4] E. Cohen and H. Kaplan. Spatially-decaying aggregation over a network: model and algorithms. *J. Comput. System Sci.*, 73:265–288, 2007.

[5] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *Proceedings of the ACM PODC'07 Conference*, 2007.

[6] C. Cranor, T. Johnson, V. Shkapenyuk, and O. Spatcheck. Gigascope: A stream database for network applications. In *Proceedings of the ACM SIGMOD*, 2003.

[7] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. In *Proceedings of the ACM SIGCOMM'03 Conference*, pages 325–336, 2003.

[8] N. Duffield, M. Thorup, and C. Lund. Flow sampling under hard resource constraints. In *Proceedings the ACM IFIP Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, pages 85–96, 2004.

[9] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better netflow. In *Proceedings of the ACM SIGCOMM'04 Conference*. ACM, 2004.

[10] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM'02 Conference*. ACM, 2002.

[11] M. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD*. ACM, 1998.

[12] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proceedings of the ACM SIGMOD*, 1997.

[13] N. Hohn and D. Veitch. Inverting sampled traffic. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 222–233, 2003.

[14] K. Keys, D. Moore, and C. Estan. A robust system for accurate real-time summaries of Internet traffic. In *Proceedings of the ACM SIGMETRICS'05*. ACM, 2005.

[15] A. Kumar, M. Sung, J. Xu, and E. W. Zegura. A data streaming algorithm for estimating subpopulation flow size distribution. *ACM SIGMETRICS Performance Evaluation Review*, 33, 2005.

[16] S. Ramabhadran and G. Varghese. Efficient implementation of a statistics counter architecture. In *Proc. of ACM Sigmetrics 2003*, 2003.

[17] D. Shah, S. Iyer, B. Prabhakar, and N. McKeown. Maintaining statistics counters in router line cards. *IEEE Micro*, 22(1):76–81, 2002.