Algorithms and Estimators for Accurate Summarization of Internet Traffic

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

Statistical summaries of traffic in IP networks are at the heart of network operation and are used to recover information on arbitrary subpopulations of flows. It is therefore of great importance to collect the most accurate and informative summaries given the router's resource constraints. Cisco's sampled NetFlow, based on aggregating a sampled packet stream into flows, is the most widely deployed such system.

We observe two sources of inefficiency in current methods. Firstly, a single parameter (the sampling rate) is used to control utilization of both memory and processing/access speed, which means that it has to be set according to the bottleneck resource. Secondly, the unbiased estimators are applicable to summaries that in effect are collected through uneven use of resources during the measurement period (information from the earlier part of the measurement period is either not collected at all and fewer counter are utilized or discarded when performing a sampling rate adaptation).

We develop algorithms that collect more informative summaries through an even and more efficient use of available resources. The heart of our approach is a novel derivation of *unbiased* estimators that use these more informative counts. We show how to efficiently compute these estimators and prove analytically that they are superior (have smaller variance on *all* packet streams and subpopulations) to previous approaches. Simulations on Pareto distributions and IP flow data show that the new summaries provide significantly more accurate estimates. We provide an implementation design that can be efficiently deployed at routers.

Categories and Subject Descriptors: G.3: probabilistic algorithms; C.2.3: network monitoring **General Terms:** Algorithms, Measurement, Performance **Keywords:** NetFlow, Network management, sketches, data streams, subpopulation queries, IP flows.

IMC'07, October 24-26, 2007, San Diego, California, USA.

1. INTRODUCTION

Collection and summarization of network traffic data is necessary for many applications including billing, provisioning, anomaly detection, inferring traffic demands, and configuring packet filters and routing protocols. Traffic consists of interleaving packets of multiple flows but the summaries should support queries on statistics of subpopulations of IP flows, such as the amount of traffic that belongs to a particular protocol, originate from a particular AS, or both. These queries are posed after the sketch is produced and therefore it is critical to retain sufficient meta data information and provide estimators that facilitate such queries.

Two critical resources in the collection of data are the high-speed memory (usually expensive fast SRAM) and CPU power that are used to process the incoming packets. The available memory limits the number of cached flows that can be actively counted. The processing power limits the level of per-packet processing and the fraction of packets that can undergo higher-level processing.

The practice is to obtain periodic summaries (sketches) of traffic by applying a data stream algorithm to the raw packet stream. The most widely deployed such system is Cisco's sampled NetFlow (NF): packets are sampled randomly at a fixed rate. Once a flow is sampled, it is cached, and a counter is created that counts subsequent sampled packets of the same flow. The number of counters is the number of distinct sampled flows. The packet-level sampling that NF performs serves two purposes: Firstly, it addresses the memory constraint by reducing the number of distinct flows that are cached (the bulk of small flows is not sampled). Without sampling, we need a counter for each distinct flow in the original stream. Secondly, the sampling reduces the processing power needed for the aggregation, since only sampled packets require the higher-level processing required in order to determine if they belong to a cached flow.

An algorithm that is able to count more packets than NF using the same number of statistics counters (memory) is *sample-and-hold (*SH*)* [10, 9]. 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 cached. 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). SH sketches are considerably more accurate than NF sketches [10, 9]. A disadvantage of SH over NF, however, is that the summarization module must process *every* packet in order to determine if it belongs to a cached flow. This additional processing makes it less practical for high volume routers.

NF and SH use a *fixed* packet 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 variance. Therefore one can manually adjust the sampling rate so that the number of counters does not exceed the memory limit and most counters are utilized [9]. Anomalies such as DDoS attacks, however, can greatly affect the number of distinct flows. A fixed-sampling-rate scheme can not react to such anomalies as its memory requirement would exceed the available memory. Therefore, anomalies would cause disruption of measurement or affect router performance. These issue is addressed by *adaptive* variants that include *adaptive sampled NetFlow* (ANF) [10, 8, 13] and *adaptive SH* (ASH) [10, 9]. These variants adaptively decrease the sampling rate and adjust the values of the statistics counters as to emulate sampling with a lower rate.

Overview

We design sketching algorithms for packet streams that obtain considerably more accurate statistics than existing approaches. We focus on subpopulation-size queries (packets or bytes of a subpopulation) but the sketches can be used for other queries such as flow size distribution. We use available resources in a balanced and load-sensitive way to collect more information from the packet sample. Then we design unbiased estimators that use the additional information. Our algorithms are robust to anomalies and changes in traffic patterns, and gracefully degrade performance when there is a decrease in available resources. They are supported by rigorous analysis.

Step counts for NF *and* SH. NF, SH, and their adaptive variants do not equally utilize available resources through the measurement period: The number of cached flows increases through the measurement period and reaches its maximum only at the end. The adaptive ANF and ASH fully utilize all counters, but this utilization is in a sense "wasted" and does not translate into more accurate estimates, as each *rate adaptation* (decrease of the sampling rate) is implemented by discarding the more informative counts obtained with the lower sampling rate [10, 8].

Step-counting NetFlow (SNF) and *step-counting sample-and-hold* (SSH) process the same packets as their adaptive counterparts, but when performing rate adaptation, they *retain* the current counts. SSH and SNF build on a simple but powerful design [17, 15] of transferring partial counts from (fast and expensive) SRAM to (slower and cheaper) DRAM, which allows us to use smaller size counters in SRAM and add the counts to larger DRAM counters when the SRAM counters are about to overflow. This design allows us to distinguish between the resources required for active counting and those required for intermediate storage. While applicable to all methods, SNF and SSH are able to make a further use of this design by transferring, after each rate adaptation, the counts into DRAM. Counting more of the processed packets in the final summary is the key for obtaining better estimates.

Hybrids of NF *and* SH. There are multiple resource constraints for gathering statistics. At the router, the memory size that determines the number of statistics counters and the CPU processing (or size of specialized hardware) that determines the fraction of packets that can be examined against the flow cache. Other constraints are the available bandwidth and storage to transmit and store the final sketch. Previous schemes, however, use a *single parameter* (sampling rate) with these multiple constraints: NF (and ANF and SNF), must set (or adjust) the sampling rate to be low enough so that the number of counters does not overflow or over utilize the router memory. As a result, resources available for processing packets may not be fully utilized. SH (and ASH and SSH) , on the other hand, do not address CPU processing constraints at all, and all packets are processed. We propose *hybrid* schemes of SH variants that use *two* packet sampling rates. The first one controls the fraction of packets that are processed in order to determine if they belong to an already-cached flow. The second, and lower, rate, determines the fraction of packets that can create new entries of cached flows.

Estimators. The sketches we produce (for all algorithms) have the form of a subset of the flows along with the flow attributes and an *adjusted weight* associated with each flow. Adjusted weights have the property that for each flow, the expectation is equal to its actual size (Adjusted weights of flows not included in the sketch is defined to be zero). 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 per-flow unbiasedness property is highly desirable as accuracy increases when aggregating over larger subpopulations and when combining estimates obtained from sketches of different time periods. The heart of our work is the calculation and analysis of unbiased adjusted weights.

The derivation of adjusted weights for NF, which applies fixed-rate sampling, is standard: a simple scaling of the counts by the inverse sampling rate. Adjusted weights assignments for ANF and ASH are provided in [2] along with non explicit derivation for SSH. We provide an explicit and efficiently computable expressions for SSH and SNF adjusted weights. An important contribution we make is the understanding of what information to gather and how to use it to obtain correct adjusted weights. The derivation and efficient computation of correct unbiased adjusted weights for SNF and SSH is novel and highly nontrivial. The adjusted weights we derive have minimum variance among all estimators that use the same information (the counts gathered by the algorithm for the flow), and in this sense are *optimal*.

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 algorithm for these flows, and the procedure used to calculate these weights. The distribution of the subsets of flows included in the sketch produced by each of the algorithms NF, SH, hybrids, and variants, is that of drawing a weighted sample without replacement from the full set of aggregated flows (for ASH, ANF, and SSH this is shown in [2].) Therefore, the difference in the quality of the sketches stems only from the variance of the adjusted weights we assign. More informative counts are beneficial only if they correspond to adjusted weights with lower variance. We analyze the variance of the adjusted weight assignment and establish a strong relation between the different methods that holds for any packet stream and any flow or subpopulation of flows.

There are typically multiple numeric properties of interest over subpopulations, such as total bytes, total packets, and number of distinct flows. We derive unbiased estimators for other aggregates that are applicable to the same sketches.

Implementation. Our implementation design piggybacks on several existing ingredients. The basis is the flow counting mechanism that Cisco's NF deploys. (Proposed improved implementation such as [15, 8] can also be integrated.) A router implementation of adaptive sampling rate for ANF was proposed in [8, 13] (rate adaptation was termed *renormalization*. This design can also be used for ASH and the step-counting and hybrid variants.

Discretized sampling rates. The pure adaptive models perform a rate adaptation each time a flow is "evicted" from the cache. Rate adaptations, however, are intensive operations [8, 13]. We design a "router friendly" variant of the pure model with discretized sampling rates. This design drastically reduces the number of rate adaptations and also simplifies their implementation. As in [8], discretization allows us to efficiently perform rate adaptations. The discretized model, however, differs mathematically from the pure sampling schemes. We show how to apply the estimators derived for the pure schemes to the discretized schemes. More importantly, we show that these estimators are also unbiased and retain other key properties of the estimators for the pure model. Furthermore, the particular discretization we used was critical for the unbiasedness arguments to hold.

Performance study. We evaluate the performance of these methods on IP flows data collected by *unsampled* NF running on a gateway router and on synthetic data obtained using Pareto distribution with different parameter values. On the IP data, we consider subpopulations of flows that belong to specified applications, and on the synthetic data, we consider prefixes and suffixes of the flow size distribution. We observe that the step-counting SNF and SSH provide significantly more accurate estimates than their adaptive counterparts. The SH variants significantly dominate their NF counterparts and the hybrid version provides a smooth performance curve between these two extremes. Even with low sampling rates, the hybrids are able to provide much more accurate estimates than plain NF, ANF, and SNF. We also show that our implementation design can be tuned to provide a very low number of rate adaptations.

2. RELATED WORK

An orthogonal summarization problem is summarizing *aggregated* data [11]. For example, using k-mins or bottom-k sketches [1, 3, 5, 4, 7]. Estimators developed for these summaries utilize the weight of each item, which is not readily available in our unaggregated setup. Direct application requires *pre-aggregation*, that is, obtaining an exact packet count for each flow as when running unsampled NF. This is infeasible in high volume routers as it requires processing of every packet and storing an active counter for every flow. These methods can be used,however, to trim the size of a sketch obtained using any method that obtains unbiased adjusted weights (including NF, SH, and their variants), when trimming is needed in order to address transmission bandwidth or storage constraints.

An extension of ASH that does not discard counts when a rate adaptation is performed was considered in [9] for finding "elephant flows." While this extension attempts to provide similar benefit to step-counting, it is not adequate for estimating subpopulation sizes. The unadjusted count itself is indeed a better estimator than the reduced count for each individual flow, but this estimator is inherently biased. The bias depends on where in the measurement period the packets occurred, and an unbiased estimator can not be constructed from the counts collected. The relative bias is very large on smaller flows (of the order of the inverse sampling-rate) and if used to estimate subpopulation sizes for such flows, we can obtain a large relative error on such subpopulations (even is the subpopulation size is large).

Kumar et al [14] 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 fast memory (SRAM) should be 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.

In some cases, protocol-level information such as testing for the TCP syn flag [6] on sampled packets and using TCP sequence numbers [16] can be used to obtain better estimates of the size of the flow from sampled packets. These methods can significantly increase the accuracy of estimating the flow size distribution of TCP flows from packet samples, but are not as critical for subpopulation size estimates for subpopulations with multiple flows. We focus on general-purpose all-protocols estimators, but these methods can be integrated with our sketches.

This paper builds on and expands our earlier work on summarization of unaggregated data streams [2]. The main new contributions are introducing SNF, closed-form expressions and fast algorithms for computing unbiased adjusted weights for SNF and SSH sketches, hybrid sketching algorithms, router-friendly design and discretized variants, and performance study.

3. SKETCHING ALGORITHMS

We provide the underlying mathematical models of the different flow sampling schemes. These models are used in the analysis and are mimicked by the "router friendly" implementations. Our sampling schemes are data stream algorithms that are applied to a stream of packets.

Sampled NF performs fixed-rate packet sampling. Packets are sampled independently at a rate p and sampled packets are aggregated into flows. All flows with at least one sampled packet are *cached* and there is an *active counter* for each flow. The sketch includes all flows that are cached in the end of the measurement period.

SH, like NF, samples packets at a fixed rate p and maintains a cache of all flows that have at least one sampled packet. SH, however, processes all packets and not only sampled packets. If a processed packet belongs to a cached flow, it is counted.

We facilitate the analysis through a *rank-based view* [2] 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 $U[0, 1]$. The actions of each sampling scheme are defined by the rank assignment. Implementations do not track per-packet random rank values or even draw rank values. They maintain just enough "partial" information on the rank assignment to maintain a flow cache and counts that are consistent with the rank-based view.

For each flow $f \in F$ and position 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 occurred before the current position in the packet stream.

An NF sketch with sampling rate p is equivalent to obtaining a rank assignment and counting all packets that have rank value $< p$.

The set of actively counted flows at a given time is $\{f \in F | r(f) < p\}$.

An SH sketch with sampling rate p is equivalent to obtaining a rank assignment and counting all packets such that the current rank of the flow (including the current packet) at the time the packet is processed is smaller than p.

The adaptive algorithms ANF and ASH work with a fixed limit k on the number of cached flows and produce a sketch of k flows. When the number of cached flows exceeds k, the sampling rate is decreased, and counts are adjusted, so that only k flows remain cached. The current *sampling rate* is defined to be the $(k+1)$st smallest rank among $r(f)$ $(f \in F)$ (it is defined to be 1 if there are fewer than $(k+1)$ distinct flows.)

It is sometimes necessary to set a limit $p_{\text{start}} < 1$ on the initial sampling rate. In this case, the current sampling rate is defined to be p_{start} if there are fewer than $(k+1)$ distinct flows and otherwise is the minimum of p_{start} and the $(k+1)$st smallest rank among $r(f)$ $(f \in F)$.

The sampling rate is determined by the rank assignment, the prefix of processed packets, and p_{start} (it does not depend on the sampling scheme). The *effective sampling rate* is defined as the value of the sampling rate at the end of the measurement period.

The set of cached flows at a given time (by either ANF or ASH) are the flows with current rank that is below the current sampling rate. The (current) ANF count of a cached flow is the number of packets seen so far with rank that is below the current sampling rate. The (current) ASH count of a cached flow f is the number of packets seen so far such that the rank of the flow just after the processing of the packet is below the current sampling rate.

The sketch includes the set of cached flows and their counts at the end of the measurement period.

Step-counting

We refer to a decrease of the current sampling rate as *rate adaptation*. The adaptive algorithms [10, 8] implement rate adaptation by decreasing the more informative flow counts that corresponded to the higher sampling rate. The step-counting algorithms, SNF and SSH, record these counts instead of adjusting them down. The active counting of packets performed by the step-counting algorithms is just like their adaptive counterparts, but the step-counting algorithms produce a vector of counts, rather than a single count, for each flow in the sketch.

Using the rank-based view, SNF counts include all packets such that (i) the rank of the packet is below the value of the sampling rate when (just after) the packet is processed (ii) the rank of the flow remains continuously below the sampling rate since the packet was processed until the current time. SSH counts include all packets such that when (just after) the packet is processed and continuously until the current time, the rank of the flow is below the sampling rate. With both SNF and SSH, a flow is cached if and only if its rank is below the current sampling rate. Figure 1 motivates, through an example, the design of the step-counting algorithms. The figure shows two flows and the expected number of packets included in the final count by each method. The step-counting SNF and SSH count many more packets than their adaptive counterparts.

Hybrids

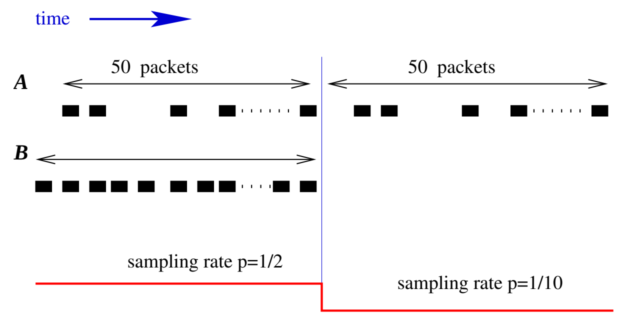
Figure 1 shows that ASH and SSH count many more packets than ANF and SNF. This advantage, however, comes at a cost: ANF and SNF simply ignore all packets that are not sampled while ASH and SSH have to process all packets in order to determine if they belong to a cached flow. Our *hybrid* algorithms provide a smooth tradeoff between the fraction of packets that are processed and the fraction that are included in the final count. Hybrid sketching algorithms use a base sampling rate parameter p_{base}, which controls the fraction of packets that are processed by the algorithm. The initial sampling rate is $p_{\text{start}} \leq p_{\text{base}}$. A hybrid algorithm samples all packets independently at a fixed rate p_{base} and then applies a respective basic algorithm (SH, ASH, or SSH) to the p_{base}-sampled stream with initial sampling rate $p'_{\text{start}} = p_{\text{start}}/p_{\text{base}}$. (Hybrid NF variants with $p_{\text{start}} \leq p_{\text{base}}$ are equivalent to applying the underlying NF variant and therefore we only consider hybrid SH variants).

The rank-based view of the hybrid algorithms is as follows. Hybrid-ASH counts include a packet if and only if (i) the rank of the packet is below p_{base} and (ii) the rank of the flow at the time (just after) the packet is processed is below the current sampling rate. Hybrid-SSH counts include a packet if and only if (i) the rank of the packet is below p_{base} and (ii) continuously, from (just after) the time the packet is processed until the current time, the rank of the flow was below the sampling rate. A flow is cached with hybrid-ASH and hybrid-SSH if and only if its rank value is below the current sampling rate.

An equivalent rank-based view of the hybrid algorithms discards all packets with rank value above p_{base}, scales the rank values of the remaining packets and p_{start} by p_{base}^{-1}, and applies the respective basic algorithm.

The following table shows the expected number of packets that are counted and processed for the example 100-packet flows in Figure 1. The hybrids use $p_{\text{start}} = p_{\text{base}} = 0.5$ and have respective sampling rates of 1 and 0.2 on the p_{base}-sampled stream. The step-counting algorithm count more of the processed packets than their adaptive counterparts. The hybrids provides a tradeoff that preserves the higher ratios between counted and processed packets.

	A	B
ANF	10/30	10/50
ASH	≈91/100	≈ 91/100
hybrid-ASH	≈46/50	≈46/50
SNF	≈30/30	≈50/50
SSH	≈99/100	≈99/100
hybrid-SSH	≈50/50	≈50/50



	A	B
ANF	10	10
ASH	91	91
SNF	≈30	≈50
SSH	≈99	≈99

Figure 1: Expected counts of two 100-packet flows. Flow *A* has 50 packets with current sampling rate 0.5 and 50 packets with current rate 0.1. Flow *B* has 100 packets with current sampling rate 0.5. The final counts with ANF and ASH only depends on the final sampling rate. SNF and SSH benefit from the higher sampling rate at the beginning of the measurement period and the final counts account for more packets.

4. IMPLEMENTATION DESIGN USING DISCRETIZED SAMPLING RATES

We design an alternative to the pure models that addresses important practical implementation issues. The first is the number of rate adaptations performed. The second is the implementation of each rate adaptation, namely, the tracking of flow ranks that determines which flows are evicted and how counts are adjusted. We then establish that the discretized version preserves important properties of the pure model that allow for unbiased estimation and for other properties of the variance of the adjusted weights to carry over.

Our discretized algorithm uses three tunable parameters that can be set by the router manufacturer. The first is $p_{\text{base}} \leq 1$ that determines the fraction of packets that are processed. The second is $p_{\text{start}} \leq p_{\text{base}}$ that determines the initial sampling rate. The third parameter is $0 < \mu < 1$ which controls the discretization of the sampling rates.

The number of rate-adaptations is a performance factor for all adaptive algorithms. Executing each adaptation is an intensive operation and therefore it is desirable to both limit the number of rate-adaptations and to carefully implement them [8]. For the step-counting algorithms, the number of rate adaptations also affects the size of intermediate storage and the computation of the adjusted weights (which depend on the number of rate-steps in which an actively-counted flow had a nonzero count).

The pure models perform a rate adaptation to evict a single cached flow at a time. It follows from the rank-based view that all adaptive algorithms (ANF, ASH, SNF, and SSH) perform the same rate adaptations. Rate adaptation occurs when the current sampling rate (the $(k+1)$st-smallest rank value of a flow) decreases. This happens when the cache is full (has k flows) and a sampled packet (equivalently, packet with rank value which is smaller than the current sampling rate) does not belong to a cached flow. The number of rate adaptations depends logarithmically on the size of the stream, but linearly on the size of the flow cache:

LEMMA 4.1. *[2] Let m be the size of the packet stream. The expected number of rate adaptations is $\leq (k+1)\ln(p_{start} m)$.*

This bound is nearly tight for streams that consist of 1-packet flows [5], and it is $\Omega(k \ln(p_{\text{start}} m))$ (asymptotically tight) when at least a constant fraction of packets belong to small flows. Large number of small flows is common in Zipf-like data and small flows are often introduced in DDoS attacks, port or IP address scanning, and other anomalies.

The actions of a run of a discretized variant of ANF, ASH, SNF, SSH, and hybrids, are equivalent to those of the original variant except that when the flow cache overflows (there are more than k flows with current rank value that is below the current sampling rate), the sampling rate is decreased by a factor of μ until at least one flow (but in expectation at most $(1-\mu)$ fraction of the flows) are evicted. Observe that the discretized implementation does not always produce sketches with k flows: The number of flows is at most k but can be smaller even if there are k or more distinct flows in the p_{base}-sampled stream.

We provide a discretized rank-based view of the discretized sketching algorithms. This view is equivalent to replacing sampling rates and rank values of packets and flows $x \in [0,1]$ with $\lceil \log_\mu (x/p_{\text{base}}) \rceil$[1]. Packets of the p_{base}-sampled stream are assigned discretized ranks using a geometric distribution with parameter $(1-\mu)$. The *discretized rank of a flow* is the largest rank of a packet of the flow. The *discretized current sampling rate* is initially set to $\lceil \log_\mu (p_{\text{start}}/p_{\text{base}}) \rceil$. After k distinct flows are cached, it is the $(k+1)$st largest discretized rank of a flow, which is equal to the largest discretized rank of a cached flow plus 1. The *discretized effective sampling rate* is the discretized sampling rate at the end of the measurement period. The flow counts collected over the p_{base}-sampled stream correspond to those for the pure model: A packet is included in the current discretized ANF count if and only if its discretized rank value is above the discretized current sampling rate; With discretized ASH, the packet is included if and only if the discretized rank of the flow after the packet is processed is above the discretized current sampling rate, and similarly for SNF and SSH.

The following property, that holds for the pure model, extends to the discretized model:

LEMMA 4.2. *A flow f is cached in the discretized model if and only if its discretized rank is larger than the kth largest discretized rank of the flows in $F \setminus \{f\}$. (If there are fewer than k flows in $F \setminus \{f\}$ with positive discretized rank value, the flow is cached if and only if its rank is positive.)*

This property is critical for extending the analysis of unbiased estimators and variance relations to the discretized model. It allows us to simply "plug in" sampling rates and the respective flow counts into the unbiased estimators developed for the pure model. The subtle arguments do not carry over to other conceivable implementations of rate adaptations such as removing a constant fraction of (highest-ranked) cached flows [13, 8] without having to maintain additional state.

A side benefit of discretization is that fewer bits are needed to encode rank values of active flows, as the expected maximum discretized rank of a flow is $\log_\mu(m p_{\text{base}})$. Other advantages of such discretization, such as layered transmission of summaries, are provided in [8].

The number of rate adaptations is tuned using the parameters μ and p_{start}. Larger values of μ correspond to a higher number of rate adaptations but also to better memory utilization and more flows in the final sketch (the expected number of active counters and number of flows in the final sketch is about $k(1+\mu)/2$). Lower values of p_{start} eliminate up to $\lceil \log_\mu (p_{\text{start}}/p_{\text{base}}) \rceil$ rate adaptations. The counts of the step-counting algorithms are reduced by lower p_{start}, but if p_{start} is larger than the effective sampling rate, then the sketch produced by ANF and ASH are not affected by the lower p_{start}. The discretized implementation has a considerably better bound on the number of rate adaptations than the pure model. In particular, the linear dependence on k exhibited by the bound obtained for the pure version, is eliminated.

LEMMA 4.3. *The expected number of rate adaptations performed is at most $\log_\mu(p_{start} * m)$.*

We outline an implementation of the discretized algorithms. The execution is divided into *counting phases* and *rate adaptations* (a design of [8] allows them to run concurrently.)

Counting phase. Each counting phase starts with a set of statistics counters indexed by the flow attributes of cached flows and applied to the p_{base}-sampled packet stream. Each packet is labeled as "sampled" (again) with probability μ^t, where t is the current discretized sampling rate. We perform the following[2]: (i) If the flow is cached then: If the algorithm is one of ASH, SSH, and hybrids **or** the packet is labeled sampled and the algorithm is one of ANF and SNF, we increment the respective flow counter. (ii) If the packet is labeled sampled **and** the flow is not cached, we create a new entry

[1] Smaller values have larger discretized values.

[2] With SH variants and hybrids we need to sample the packet only if the flow is not cached.

in the flow cache. If there are k cached flows, we perform a *rate adaptation*.

Rate adaptation. The adaptive algorithms, ANF, ASH, and hybrid-ASH maintain a single packet count for each cached flow. These counts are updated during the rate adaptation until at least one flow has a zero count. For ANF, the updated count is a binomial random variable with parameters μ and the current count; for ASH, with probability μ the count remains unchanged and otherwise it is a geometric random variable with parameters that are the current count minus 1 and rate μ^{t+1}, where t is the current discretized sampling rate. After the counts of all cached flows are updated, the current discretized sampling rate is incremented. The update process is repeated until at least one flow has a count of zero.[3] All flows with a count of zero are then evicted from the cache.

The step-counting algorithms SNF, SSH, and hybrid-SSH store a discretized rank for each cached flow. The flow ranks are updated using the counts collected in the most recent counting phase. The set of cached flows with smallest discretized rank value are then evicted. The discretized sampling rate is updated to be the discretized rank of evicted flows.

The update process of the rank of cached flows emulates the following process that assigns ranks individually to packets counted in the recent counting step. The first packet counted (SH and hybrid variants with flow that was not cached at the beginning of the phase) and all packets counted (NF variants) obtain a random rank from a geometric distribution with parameter $(1-\mu)$, conditioned on it being larger than the current discretized sampling rate. The rank of each flow is updated to be the maximum of its current rank and the ranks assigned to the packets of the flow counted at the recent counting phase.

These updates can be performed efficiently (computation steps proportional to the number of cached flows with non-empty counts at the recent step) using the exponential distribution to find the maximum discretized rank over a set of packets.

5. ADJUSTED WEIGHTS

We convert the counts collected by the sketching algorithms to adjusted weights that are associated with each flow that is included in the sketch. The adjusted weights are always a function of the observed counts. It turns out that there is a unique deterministic assignment of adjusted weights for each of the algorithms considered. Since deterministic assignment has smaller variance than any randomized one, it is preferable.

Clearly, a correct adjusted weight for NF counts is the number of counted packets divided by the sampling rate p. The derivation of the adjusted weight assignments for SH, ANF, and ASH counts is provided in [2]. For SH counts, the adjusted weight is equal to the count of the flow plus $(1-p)/p$. 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.

Adjusted weights for ANF are obtained by scaling the counts by $1/p'$, and correct adjusted weights for ASH are obtained by adding $(1-p')/p'$ to the count, where p' is the effective sampling rate [2].

The information we collect using SNF and 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, 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 assignment for a flow f is a function of $\mathbf{p}$ and $\mathbf{i}(f)$. We use the notation

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

for the adjusted weight assigned by SNF, and similarly, $A^{\mathrm{SSH}}_{\mathbf{p}}(\mathbf{i}(f))$ for the adjusted weight assigned by SSH.

Adjusted weights are computed after the counting period is terminated. After they are computed, the count vectors can be discarded. Therefore, SNF and SSH produce a sketch of size k.

We will show the following:

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

We review some definitions and properties from [2] and derive formulations that allow for efficient computation of adjusted weights for SSH and SNF sketches.

There is subtlety in the assignment of correct (unbiased) adjusted weights for ANF and ASH. Since the sampling rate varies it is not clear that we can "plug in" the effective (final) sampling rate in the adjusted weights expressions of the non-adaptive variant. The argument for unbiasedness is based on the fact that the adjusted weights of each flow are unbiased on each part of some partition of the sample space [3, 4]. For a flow f', we look at the probability subspace where the kth smallest rank among $r(f)$ $(f \in F \setminus \{f'\})$ is p' and 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 a fixed-rate respective variant with rate p'. Therefore, using the respective fixed-rate assignment with rate p' yields unbiased adjusted weight for f' within this probability subspace, and therefore, unbiased weights overall.

For the step-counting algorithms, the adjusted weights assigned to a flow f 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. The adjusted weight function, however, has the property

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

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. The following generalization of Eq. (1) states that the adjusted weight assignment does not depend on the values of the current sampling rate in durations when there are no counted packets. This allows us to state the adjusted weight of a flow in terms of an equivalent flow where the number of steps is equal to the number of steps where the original flow had a nonzero count.

LEMMA 5.2. *Consider a correct assignment of adjusted weight $A_{\mathbf{p}}(\mathbf{n})$. For an observed count $\mathbf{i}$ and $\mathbf{p}$, let $1 \leq j_1 < j_2 < \cdots < j_{r'} = r$ be the coordinates such that $i_{j_k} > 0$ or $i_{j_k} = r$ (that is, r is included also if $i_r = 0$).*

$$A_{p_1,p_2,\ldots,p_r}(0,\ldots,0,i_{j_1},0,\ldots,0,i_{j_2},\ldots) = A_{p_{j_1},p_{j_2},\ldots,p_{j_{r'}}=p_r}(i_{j_1},i_{j_2},\ldots,i_{j_{r'}}) \quad (2)$$

5.1 Adjusted weights for SSH

Let r be the number of steps and $p_1 > \cdots > p_r$ the corresponding sampling rates. For a flow f, let $\mathbf{n} = (n_1, \ldots, n_r)$ be the

[3]We can avoid the repeated process by storing discretized rank value for each flow as proposed next for the step-counting algorithms.

number of packets of f in each step and let $\mathbf{i} = (i_1, \ldots, i_r)$ be the number of counted packets in each step. We denote by $q[\mathbf{i}|\mathbf{n}]$ the probability that a flow with $\mathbf{n}$ packets has a count of $\mathbf{i}$.

In our previous work [2] we derived estimators for SSH sketches via a system of linear equations of size that is quadratic in the number of counted packets $(\sum_{h=1}^{r} n_i)^2$. We provide an exponentially-faster way to compute the adjusted weights which is quadratic in the number of steps. The total number of steps is logarithmic in the number of packets and therefore reducing the dependence from the number of packets to the number of steps constitutes a significant improvement. We then argue that the number of operations can be further reduced to be 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.

We define the values $c_{i,j}(\mathbf{p}, \mathbf{n})$ $(1 \leq i \leq j \leq r)$ as follows (the parameters $(\mathbf{p}, \mathbf{n})$ are omitted when clear from context, and we assume $n_1 > 0$ w.l.o.g.):

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

The following two lemmas are immediate from the definitions.

LEMMA 5.3.

- *For $1 \leq j \leq r$, $c_{1,j}$ is the probability that the rank of the first packet of the flow is at least p_j.*
- *For $2 \leq i \leq j \leq r$, $c_{i,j}(\mathbf{p}, \mathbf{n})$ 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.*

LEMMA 5.4. *The computation of the partial sums $\sum_{h=1}^{i} c_{h,h}$ for $i = 1, \ldots, r$ can be performed in $O(r^2)$ operations.*

By lemma 5.3, $c_{i,i}$ $(i \in \{1, \ldots, r\})$ 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.). So

$$q[\mathbf{n}|\mathbf{n}] = 1 - \sum_{h=1}^{r} c_{h,h} \ . \tag{3}$$

The following theorem expresses the adjusted weight $A^{\text{SSH}}(\mathbf{n})$ as a function of the diagonal sums $\sum_{h=1}^{i} c_{h,h}$ $(h = 1, \ldots, r)$. The proof is provided in Section 8.1.

THEOREM 5.5.

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

LEMMA 5.6. *The adjusted weight $A^{\text{SSH}}(\mathbf{n})$ can be computed using $O(r^2)$ operations.*

PROOF. The proof follows from Lemma 5.4 and Theorem 5.5. □

The proof of Theorem 5.1 is an immediate corollary of Lemma 5.6 and Lemma 5.2 (According to Lemma 5.2 when we compute $q[\mathbf{n}|\mathbf{n}]$ and the $c_{i,j}$'s we can remove from $\mathbf{n}$ and $\mathbf{p}$ all entries which are 0 in $\mathbf{n}$ except for the last entry in $\mathbf{n}$ which remains even if it is 0.)

5.2 Adjusted weights for hybrids

Unbiased adjusted weights for hybrid-ASH and hybrid-SSH are obtained by scaling by p_{base}^{-1} the adjusted weights computed for the non-hybrid variant that is applied to the p_{base}-sampled stream.

5.3 Adjusted weights for SNF

We define $d_{i,j}(\mathbf{p}, \mathbf{n})$ $(2 \leq i \leq j \leq r)$ as follows.

$$\begin{aligned} 2 \leq j \leq r: &\quad d_{2,j} = \left(\frac{p_1 - p_j}{p_1}\right)^{n_1} \prod_{h=1}^{r} p_h^{n_h} \\ 3 \leq i \leq j \leq r: &\quad d_{i,j} = \left(\frac{p_{i-1} - p_j}{p_{i-1}}\right)^{n_{i-1}} (d_{i-1,j} - d_{i-1,i-1}) \end{aligned}$$

For $2 \leq i \leq j \leq r$, $d_{i,j}(\mathbf{p}, \mathbf{n})$ is the probability that all packets of the flow $\mathbf{n}$ have rank values below the sampling rate at packet arrival time, that the flow is fully counted by SNF until the transition into step i, and that at the beginning of step i, the rank of the flow is at least p_j.

The probability that all packets are counted by SNF is equal to $\prod_{h=1}^{r} p_h^{n_h}$ minus the probability that the counting halts at the transition into steps $2, \ldots, r$:

$$q^{\text{SNF}}[\mathbf{n}|\mathbf{n}] = \prod_{h=1}^{r} p_h^{n_h} - \sum_{j=2}^{r} d_{j,j} \ . \tag{4}$$

THEOREM 5.7.

$$A^{\text{SNF}}[\mathbf{n}] = \frac{\sum_{j=1}^{r} \frac{n_j}{p_j} (\prod_{h=1}^{r} p_h^{n_h} - \sum_{\ell=2}^{j} d_{\ell,\ell})}{\prod_{h=1}^{r} p_h^{n_h} - \sum_{j=2}^{r} d_{j,j}}$$

The proof of the Theorem is provided in Section 8.3.

5.4 Adjusted weights for discretized algorithms

Unbiased adjusted weights for the discretized algorithms are obtained by recording the discretized sampling rates for each step. We then convert each discretized rate t to a corresponding sampling rate $(p_{\text{start}}/p_{\text{base}})\mu^t$ and plug them in the corresponding expressions for ANF, ASH, SNF, or SSH adjusted weights. For the hybrid versions ($p_{\text{base}} < 1$), we scale the adjusted weights by p_{base}^{-1}.

The arguments for correctness, that are based on obtaining an unbiased estimator on each part in a partition of the same space, extend to the discretized version using Lemma 4.2. If the ranks of all packets in $F \setminus \{f\}$ are fixed, the discretized sampling rate when f is counted depends only on these fixed ranks (and not on ranks assigned to previous packets of f) and is equal to the sampling rate at measurement time. Therefore, we can compute unbiased adjusted weights while treating the effective sampling rate (for ASH and ANF) or the steps of the current sampling rate (for SNF and SSH) as being fixed.

5.5 Relation between the sketching algorithms

The rank-based view shows that the distribution over subsets of flows included in the sketch is the same for ANF, ASH, SNF, SSH. The different algorithms applied with the same rank assignment result in the same set of k cached flows (or all flows if there are fewer than k distinct flows in the packet stream.). The hybrid algorithms result in "almost" the same distribution: if the p_{base}-sampled packet stream contains fewer than k distinct flows then the sketch will only include those flows, but the included flows are a subset of the flows included in a sketch generated by the non-hybrid algorithms using the same underlying assignment.

This distribution is equivalent to weighted sampling without replacement of k flows (WS). WS is performed as follows over the set of aggregated flows: repeatedly, k times, a flow is selected from the set of unsampled flows with probability proportional to its weight. Adjusted weights for WS can be obtained using the rank conditioning method [4]. These weights are computed using the *exact packet*

alg	sketch size	active counters	counts collected
ANF	k	k	for k flows
ASH	k	k	for k flows
SNF	k	k	per-step for k flows
SSH	k	k	per-step for k flows
WS	k	$\|F\|$	for all flows in F

Table 1: Methods that obtain a sketch of size k that is a weighted sample without replacement from the set of flows F.

count of each flow and therefore can not be obtained by a stream algorithm with size-k flow cache. We include these WS sketches in our evaluation in order to understand to what extent performance deviates in comparison. The adjusted weight assigned to each flow is equal to the number of packets divided by the probability that the flow is included in the sample[4] 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 the effective sampling rate. Therefore, the adjusted weight is equal to $|f|/(1-(1-p')^{|f|})$.

Since these algorithms (see Table 1) 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. We explore the relation of estimate quality between the different sketches. The proofs are provided for the pure models where these algorithms are all applied with a flow cache of size k, but they extend to the discretized model as well.

5.6 Variance of adjusted weights

An algorithm dominates another, in terms of the information it collects on each sketched flow, if we can use its output to emulate an output of the second algorithm. It is not hard to see that SNF dominates NF, that SSH dominates both ASH and SNF (and therefore also dominates NF), that SNF and ASH are incomparable, and that they are all dominated by WS. Therefore, SSH sketches are the most powerful and ANF sketches are the least powerful. The variance of the adjusted weight assignments reflects this dominance relation, with lower variance for the methods that gather more information. In [2] it is shown that for any packet stream and any flow f, $\text{VAR}(A^{\text{WS}}(f)) \leq \text{VAR}(A^{\text{SSH}}(f)) \leq \text{VAR}(A^{\text{ASH}}(f)) \leq \text{VAR}(A^{\text{SNF}}(f))$. We extend this relation to SNF: (The proof is provided in Sections 8.2 and 8.3)

THEOREM 5.8. *For any packet stream and any flow f we have the following relation between the variance of the adjusted weight assignment for f.*

$$\text{VAR}(A^{\text{SSH}}(f)) \leq \text{VAR}(A^{\text{SNF}}(f)) \leq \text{VAR}(A^{\text{ANF}}(f)) \quad (5)$$

The relation is established by showing that adjusted weights for "more informative" sketches have smaller variance. This approach extends to show that the relation between the variance of the different methods also applies to the discretized variants of the sampling algorithms and to fixed-rate and fixed-steps variants of WS NF and SH. A variance relation also holds for the hybrids: the variance is non-increasing with the packet-processing rate p_{base}.

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

6. ESTIMATING OTHER AGGREGATES

The sketches support estimators for aggregates of other numeric flow properties over a queried subpopulation. We distinguish between *flow-level* and *packet-level* properties.

6.1 Flow-level properties

We classify a numeric property $h(f)$ of the flow f as flow level if it can be extracted from any packet of the flow and some external data (therefore, we know $h(f)$ for all the flows that are included in the sketch). Examples are the number of hops to the destination AS, unity (flow count), and flow identifiers (source or destination IP address and port, protocol). Flow-level properties can be aggregated *per-packet* or *per-flow*.

Per-packet aggregation. For a subpopulation $J \subset F$, the *per-packet sum* of $h()$ over J is $\sum_{f\in J} w(f)h(f)$. The *per-packet average* is $\frac{\sum_{f\in J} w(f)h(f)}{\sum_{f\in J} w(f)}$. If $h(f)$ is the number of AS hops traveled by the flow f then the per-packet sum is the total number of AS hops traveled by packets in the subpopulation J and the per-packet average is the average number of hops traveled by a packet in J. If $h(f)$ is unity, the per-packet sum is the weight of the subpopulation. It is not hard to see that for a sketch with unbiased adjusted weights, $\sum_{f\in J} A(f)h(f)$ is an unbiased estimator of the per-packet sum of $h()$ over J. (A (possibly biased) estimator for the per-packet average is $\frac{\sum_{f\in J} A(f)h(f)}{\sum_{f\in J} A(f)}$.)

Per-flow aggregation. The *per-flow sum* of $h()$ over J is $\sum_{f\in J} h(f)$. The *per-flow average* of $h()$ over J is $\sum_{f\in J} h(f)/|J|$. If $h(f) \equiv 1$, the per-flow sum is the number of distinct flows in a subpopulation. If $h(f)$ is the number of AS hops then the per-flow average is the average "length" of a flow in J.

Our generic estimator for per-flow sums is based on assignments of *adjusted counts* to flows. An adjusted count $\#(f)$ for $f \in F$ is a random variable that is zero for flows that are not included in the sketch **s** and for all $f \in F$, $E(\#(f)) = 1$. Hence, for any subpopulation J, $\sum_{f\in J} \#(f)$ is an unbiased estimator on $|J|$, the number of distinct flows in J.

For each $f \in F$, $E(\#(f)h(f)) = h(f)$, therefore

$$\sum_{f\in J} \#(f)h(f) = \sum_{f\in J\cap \mathbf{s}} \#(f)h(f)$$

is an unbiased estimator of the per-flow sum of $h()$ over J. We use the notation $\#^L_{\mathbf{p}}(f) \equiv \#^L_{\mathbf{p}}(\mathbf{n})$ for the adjusted count assigned to a flow f by algorithm L (for sampling rate $\mathbf{p}$ and when f has observed count $\mathbf{n}$). For $L \in \{\text{SNF}, \text{SSH}\}$, $\mathbf{p}$ and $\mathbf{n}$ are vectors and for $L \in \{\text{ANF}, \text{ASH NF}, \text{SH}\}$ they are scalars. For the adaptive algorithms, we use the effective sampling rate.

We derive expressions for adjusted counts by applying the methods we developed for adjusted weight derivation (see Section 8). As is the case for adjusted weights, there is a unique adjusted counts function $\#^L_{\mathbf{p}}(\mathbf{n})$. Adjusted counts can also be obtained from the unbiased flow size distribution (FSD) estimators, that have a form of a linear combination over observed counts, with coefficients that correspond to adjusted counts. (NF variants have ill-behaved FSD estimators [12, 6] and SH variants [2] have well-behaved estimators.)

- $L \in \{\text{SH}, \text{ASH}\}$: $\#^L_p(1) = 1/p$; for $n > 1$, $\#^L_p(n) = 1$.
- SSH: If $|\mathbf{n}| = 1$ (there is one packet counted) then $\#^{\text{SSH}}_{\mathbf{p}}(\mathbf{n}) = 1/p_r$. Otherwise ($|\mathbf{n}| > 1$), if $\mathbf{n}$ has at least one packet in the last step, then $\#^{\text{SSH}}_{\mathbf{p}}(\mathbf{n}) = 1$. Otherwise, let $\ell < r$ be the last step with a positive packet count. $\#^{\text{SSH}}_{\mathbf{p}}(\mathbf{n}) =$

$1 + \frac{c_{\ell+1,r}}{q[\mathbf{n}|\mathbf{n}]}$. (If we eliminate steps with zero count except the last step, $\ell + 1 \equiv r$.) The adjusted counts need to be computed when the adjusted weights are computed, before discarding the per-step counts.

- NF variants: Adjusted counts are defined by applying methods developed for adjusted weights derivation (see Section 8).

  $L \in \{\text{NF}, \text{ANF}\}$: the solution of the triangular system of linear equations obtained from the the unbiasedness constraints: For a flow with count $|f|$,

  $$\sum_{i=0}^{|f|} \binom{|f|}{i} p^i (1-p)^{|f|-i} \#_p^L(i) = 1 \ .$$

  SNF: Solution of a system of linear equations with variables $\#_{\mathbf{p}}^{\text{SNF}}(\mathbf{s})$ for all $\mathbf{s} \le \mathbf{n}$. (The vector notation $\mathbf{s} \le \mathbf{n}$ means that the relation holds per coordinate) using the unbiasedness constraints. Alternatively, they can be defined using dominance of SNF over ANF. As is the case for FSD estimators, these systems are ill-behaved for low sampling rates.

6.2 Packet-level properties

Packet-level properties have numeric $h()$-values that are associated with each packet. For a flow f we define the $h()$-value of f as $h(f) = \sum_{c \in f} h(c)$. If $h(c)$ is the number of bytes in the packet c then $h(f)$ is the number of bytes in the flow. If $h(c)$ is unity, then $h(f) = w(f)$ is the number of packets of the flow. $h(f)$ is available only if all packets of f are processed and therefore is not provided for flows included in NF and SH variants sketches.

The algorithms are adapted to collect information needed to facilitate unbiased estimators. For any desired packet-level property $h()$, we produce *adjusted* $h()$*-values* $H_{\mathbf{p}}^L(f)$. For any f, $H(f)$ is an unbiased estimator of $h(f)$ and $H(f) = 0$ for flows that are not included in the sketch. For any subpopulation J, $\sum_{f \in J} H(f) = \sum_{f \in J \cap \mathbf{s}} H(f)$ is an unbiased estimator for $\sum_{f \in J} h(f)$.

- $L \in \{\text{NF}, \text{ANF}\}$: Let $N(f)$ be the set of counted packets and let $n(f)$ be the packet count maintained by L for a cached flow f. To facilitate $h()$-values estimation, the algorithm maintains the $h()$ counts $n^{(h)}(f) = h(N)$. The adjusted $h()$-value is $H_p^L(f) = n^{(h)}(f)/p$. A subtle point is correctly updating $n^{(h)}$ for ANF when performing rate adaptation. The updated value should include the sum of the $h()$ values of resampled packets which if strictly done, requires us to store the $h()$ value of all packets in $N(f)$, which is prohibitive. We argue, however, that it is sufficient to store the total $n^{(h)}(f)$ and update it proportionally to the reduction in the packet count $n(f)$. That is, if resampling reduces the packet count to $n'(f)$, we update the $h()$-count $n^{(h)}(f) \leftarrow n^{(h)}(f) n'(f)/n(f)$.

  The updated $n^{(h)}(f)$ is the expectation of the updated $h()$ count that would have been obtained if we explicitly maintained N and per-packet $h()$ values and sampled from N a subset of size $n'(f)$. (all subsets of $N(f)$ that are of the same size have the same probability of being in the resample, regardless of the packet position or its $h()$ value.)

  This consideration extends to a sequence of rate adaptations: The final $n^{(h)}(f)$ has the expectation of the $h()$ count over all resamples that resulted in the same sequence of packet count reductions. Interestingly, done this way, we obtain a lower variance estimator than if we had maintained and used per-packet $h()$ values for $N(f)$, as the $h()$ count we use is the expectation of the latter in each part of a partition of the sample space.

- $L \in \{\text{SH}, \text{ASH}\}$: we obtain the unbiased estimator $H^{\text{ASH}}(f) = h(c_0(f))\frac{1-p}{p} + h(N(f))$, where $N(f)$ is the set of counted packets of f and $c_0(f) \in N(f)$ is the first counted packet of f. To facilitate this estimator, the algorithm needs to record the $h()$ value of the first packet and the sum of $h()$ values of all subsequent packets. For ASH, the resampling makes a direct implementation infeasible, as we need to record per-packet $h()$ values for all packets in N. We can not use averaging as we did for ANF, since later packets are more likely to be counted than earlier packets. Fortunately, there is efficient implementation for SSH.

- $L \in \{\text{SNF}, \text{SSH}\}$: To facilitate estimators, the algorithms need to maintain per-step $h()$ counts,

  $$h(N_1), h(N_2), \ldots, h(N_r) \ ,$$

  where N_i is the set of packets counted in step i. For SSH, we also need to maintain the $h()$ value of the first packet c_0 when the flow enters the cache. The expressions for adjusted $h()$ values are provided in Eq. (7) (for SSH) and Eq. (15) (for SNF).

- Hybrid SH variants: We obtain adjusted $h()$ values for the sampled stream and scale them by p_{base}^{-1}.

6.3 Flow size distribution

We derive unbiased estimators for subpopulations flow size distribution queries using hybrid sketches. We use existing flow size distribution estimators as components, applicable to ASH and SSH sketches [2] and NF sketches [12, 6].

For hybrid-ASH (or hybrid-SSH) we apply the respective (ASH or SSH) estimators in [2] to the counts obtained on the p_{base}-sampled stream. We obtain unbiased estimates M_j ($j \ge 1$) for the number of flows of size j in the p_{base}-sampled stream. We then "treat" these M_j values as observed counts with NF with sampling rate p_{base}, and "plug them" in a corresponding estimator. Since the NF estimators are ill behaved for low sampling rates, these estimators are ill behaved for low values of p_{base}.

LEMMA 6.1. *The resulting estimates $\hat{C}'_i$ ($i \ge 1$) are unbiased estimates of the number of flows of size i in the original stream.*

PROOF. We review the derivation of flow size distribution estimators. These estimators are derived by expressing O_j ($j \ge 1$), the expected number of flows with a certain observed counts (for step-counting algorithms we use observed count vectors) as a linear combination of C_i ($i \ge 1$), the number of flows of size i. The matrix is then inverted, and we express each C_i as a linear combination of the expectations of the observed counts. The estimates $\hat{C}_i$ are linear combinations of the observed counts. Since this is a linear combination, we can replace the observed counts with any other random variables with the same expectation and still obtain unbiased estimators $\hat{C}'_i$. □

7. SIMULATIONS

We use simulations in order to understand several performance parameters: The accuracy of the estimates derived and its dependence on the algorithms, parameter settings, and the consistency of the subpopulation, the tradeoffs of the hybrid approach, and the effectiveness of the parameters controlling the number of rate adaptations. The simulations were performed using the discretized variants of the algorithms, with parameters k (maximum number of counters), $0 < p_{\text{base}} \leq 1$ that determines the fraction of processed packets (hybrid approach), and μ and p_{start}, that control a tradeoff between accuracy/utilization and the number of rate adaptations.

7.1 Data

We used both synthetic and IP flows datasets. The IP flows data was collected using *unsampled* NetFlow (flow-level summary of each 10 minute time period that includes a complete packet count for each flow) deployed at a gateway router. A typical period has about 5000 distinct flows and 100K packets. The synthetic datasets were produced using Pareto distributions with parameters $\alpha = 1.1$ and $\alpha = 1.5$. We generated distributions of flow sizes by drawing 5000 flow sizes. We simulated a packet stream from each distribution of flow sizes by randomly permuting the packets of all flows.

The cumulative distributions of the weight of the top i flow sizes for each distribution is provided in Figure 2.

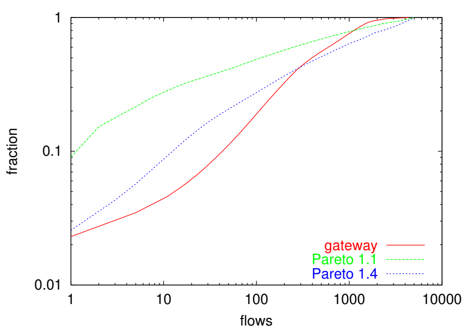


Figure 2: Fraction of packets belonging to the i heaviest flows.

The subsets (subpopulations) we considered for the synthetic datasets were the 2^i largest flows and the 50%, 30%, and 10% smallest flows. This selection enables us to understand how performance depends on the consistency of the subpopulation (many smaller flows or fewer larger flows) and the skew of the data. The subpopulations we used for the IP flows (gateway) data were a partition of the flows according to destination port.

7.2 Quality of sketches

We compare the accuracy of subpopulation-size estimates obtained using ANF, SNF, ASH, and SSH. We evaluate performance as a function of the size k of the sketch (and the size of the flow cache). We also include weighted sampling without replacement.

Results that show the average absolute value of the relative error as a function of the cache size k are provided in Figures 3 and 4. The averaging for each data point was performed over 200 runs. The figures reflect the relation established theoretically: SSH dominating ASH which in turn dominates ANF and that SNF dominates ANF. They also show that ASH dominates SNF on our data. For subpopulations consisting of large flows, such as top-i flows and applications with medium to large flows, the performance gaps are more pronounced. This is because on these flows, the more dominant methods count more packets and obtain adjusted weights with smaller variance. Our adjusted weight assignments have minimum variance with respect to the information they use (the counts collected for the flow). This "optimality" enables us to translate the larger counts to smaller errors.

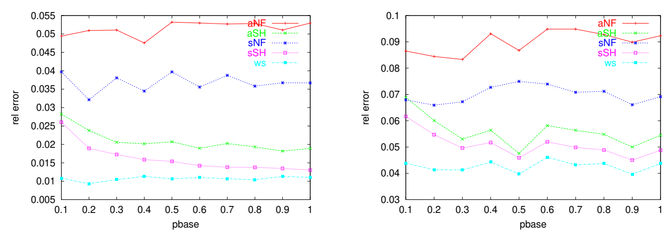
On subpopulations of very small flows, such as bottom-50% of flows or DNS (port 53) traffic (only the latter is shown), all methods have similar performance. In particular, there is no advantage for WS (the strongest method) over ANF (the weakest) on subpopulations consisting of 1-packet flows.

The results strongly support the use of step-counting as an alternative to the adaptive variants: On subpopulations consisting of many medium to large size flows, the relative error obtained using SNF and SSH is significantly smaller than what is obtained using ANF or ASH, respectively.

7.3 Evaluation of the hybrid algorithms

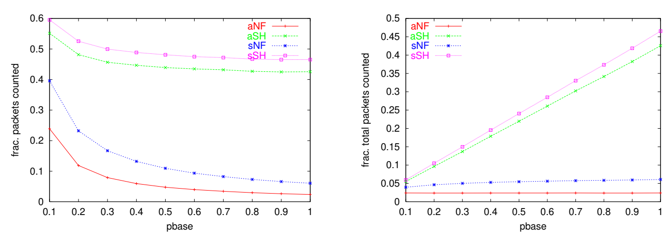
We decrease the parameter p_{base} while maintaining the same flow cache size $k = 400$. Figure 5 shows that the estimate quality gracefully degrades and a smooth performance curve is obtained. Even for $p_{\text{base}} = 0.1$, the hybrid-ASH and hybrid-SSH outperform ANF and SNF, respectively.

Figure 6 shows the number of packets counted as a function of p_{base} as a fraction of the number of packets in the p_{base}-sampled stream and as a fraction of the total number of packets. The hybrids provide a desirable smooth tradeoff that provides high counting rate of processed packets, while enabling us to fully control the fraction of total packets that are processed.



pareto 1.1, top-64   pareto 1.4, top-64

Figure 5: Sweeping p_{base} to evaluate hybrid algorithm. Estimating subpopulations of top-64 flows in Pareto 1.1 and Pareto 1.4, $k = 400$, $p_{\text{start}} = p_{\text{base}}$, $\mu = 0.9$.

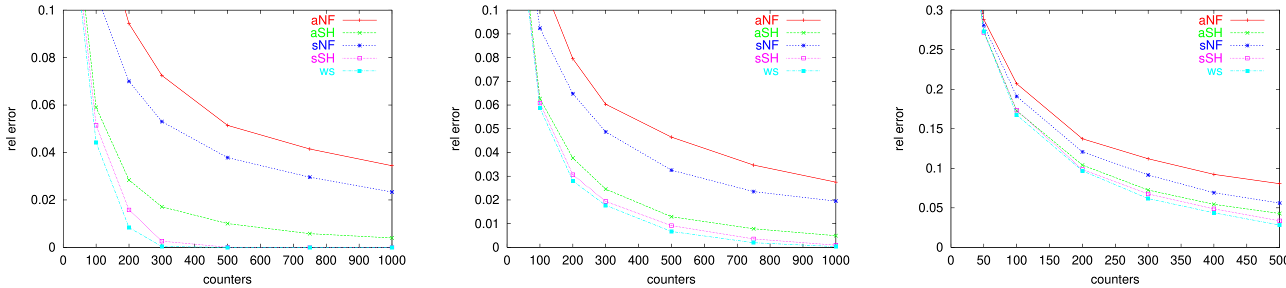


pareto 1.1, top-64   pareto 1.4, top-64

Figure 6: Sweeping p_{base} to evaluate the tradeoffs in the hybrid algorithms. Fraction of packets in the $pbase$-sampled stream that are counted (left). Fraction of total packets that are counted (right). ($k = 400$, $p_{\text{start}} = p_{\text{base}}$, $\mu = 0.9$).

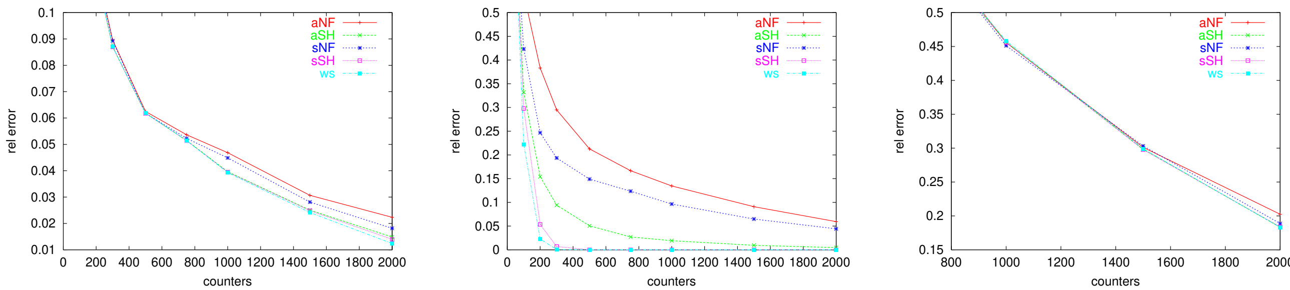
7.4 Controlling the number of rate adaptations

We sweep the parameter μ, which controls the rate of decrease of the sampling rate, and through it, the total number of rate adaptations performed. We expect the (absolute value of the) relative error of the estimates to increase when μ is decreased, as fewer packets are counted and reflected in the final sketch. On the other hand, the number of rate adaptations performed and the size of intermediate temporary storage needed to store the count vectors for SNF and SSH should decrease with μ. The effectiveness of the



Pareto 1.1, top-16 Pareto 1.1, top-64 Pareto 1.4, top-64

Figure 3: Estimating subpopulations consisting of top flows $p_{\textbf{base}} = 1$, $\mu = 0.9$, $p_{\textbf{start}} = 1$.



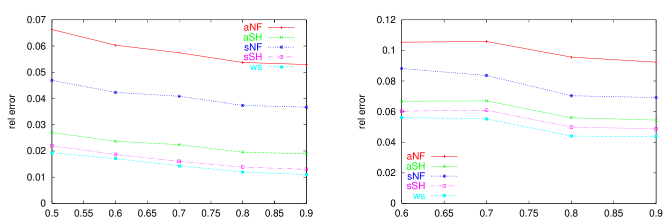
gateway, port 80 gateway, port 3818 gateway, port 53

Figure 4: Estimating subpopulations of IP flows for various port numbers, $p_{\textbf{base}} = 1$, $\mu = 0.9$, $p_{\textbf{start}} = 1$.

parameter μ and the feasibility of a router implementation depends on this tradeoff.

Figure 7 shows the dependence of the average absolute value of the relative error on the parameter μ. We can see that there is minimal performance loss in terms of estimate quality when μ is reduced from 0.9 to 0.5.

Figure 8 shows that selecting a smaller $\mu = 0.5$ is very effective. First, the number of rate adaptations is much smaller, and secondly, the size of intermediate temporary storage needed for collecting the count vectors, is much smaller with $\mu = 0.5$ than with $\mu = 0.9$.

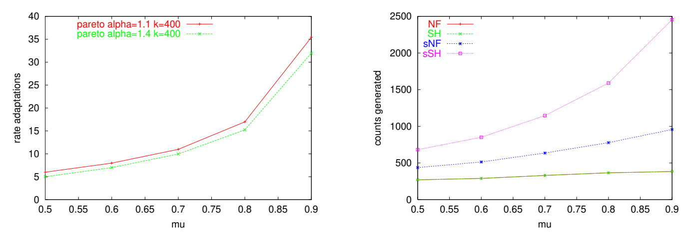


pareto 1.1, top-64 pareto 1.4, top-64

Figure 7: Sweeping μ to evaluate how it affects the (average absolute value of the) relative error. The figure shows subpopulations of top-64 flows for pareto 1.1 and pareto 1.4 $p_{\textbf{start}} = p_{\textbf{base}} = 1$, $k = 400$.

8. DEFERRED PROOFS

The adjusted weights we assign are a function of the observed count of the flow and the sampling rate. The sampling rate (effective sampling rate or sampling rate steps) in the adaptive algorithms is treated as fixed because for any flow f, it is determined by the



Rate adaptations number of counts pareto 1.1, top-64

Figure 8: Sweeping μ to evaluate effectiveness decrease. Number of rate adaptations and number of counts $p_{\textbf{start}} = p_{\textbf{base}} = 1$, $k = 400$.

sampling performed on all "other" flows ($F \setminus \{f\}$). Therefore, within the probability subspace where the sampling on all other flows is fixed, the sampling rate is fixed. The adjusted weight of each flow is unbiased within each such subspace.

We deploy three different techniques to derive adjusted weights. These techniques are general tools applicable to other quantities such as adjusted counts, unbiased FSD estimators, and adjusted selectivities.

- *System of equations*: The unbiasedness constraints correspond to linear equations over the variables $A^L_{\mathbf{p}}(\mathbf{n})$. For a flow f, the expected adjusted weight over all possible observable counts $\mathbf{n} \preceq f$ of f must be equal to $w(f)$: $w(f) = \sum_{\mathbf{n} \preceq f} q[\mathbf{n}|f] A^L_{\mathbf{p}}(\mathbf{n})$, where $q[\mathbf{n}|f]$ is the conditional probability that L obtains a count of $\mathbf{n}$ for a flow f. The system of equations can be used to derive expressions for the adjusted weights, be solved numerically to compute adjusted weights for each instance, or establish properties of the solution such as uniqueness (A unique solution to this system implies that there is a unique deterministic assignment of adjusted weights that is a function of the observed counts of the flow and the sampling rate).

- *Dominance*: Algorithm A_1 dominates A_2 [2] if A_1 counts are "more informative", that is, each possible set of A_1 counts corresponds to a probability distribution over sets of A_2 counts such that applying A_1 and drawing an A_2 count from the corresponding distribution is equivalent to applying A_2. For example, sSH dominates aSH and aSH dominates aNF. We can derive adjusted weights for A_1 from those of A_2 by taking the expectation of A_2 adjusted weights over the distribution that corresponds to the A_1 counts. If there is no closed form for A_1 adjusted weights, we can draw multiple times from the corresponding distribution and take an average of the A_2 adjusted weights. The resulting per-item adjusted weights of A_1 are unbiased and have equal or lower variance than A_2 adjusted weights (same arguments as for mimicked sketches [5]). Dominance implies that if an algorithm has unique deterministic adjusted weights, the per-item variance of these adjusted weights is at most that of any algorithm that it dominates.

- *Per-packet Horvitz-Thompson (*HT*) analysis.* The HT estimator is applicable when both the weight and the sampling probability of each item are provided. The weight is available for packets, but in our unaggregated setting, not for flows, and therefore we compute an adjusted weight for each packet. The adjusted weight of the flow is then the sum of the adjusted weights of packets of the flow. The second ingredient needed for HT estimators is the sampling probability, which can not be determined from the sketch. We apply HT with sample space partitioning [4] to "bypass" knowledge of the sampling probability. Per-packet analysis allows us to derive unbiased estimators for other packet-level properties.

8.1 Adjusted weights for sSH

We prove Theorem 5.5 using a per-packet application of the HT estimator.

Let h be a per-packet weight function and let $h(f) = \sum_{c \in f} h(c)$ be the $h()$-value of f. ($h(c) \equiv w(c) = 1$ for packet counts but we can also use other packet-level properties such as the number of bytes in c.) We compute unbiased adjusted $h()$ values $H^{\text{SSH}}(c)$ for each packet from which we can obtain unbiased adjusted $h()$ values for each flow using $H^{\text{SSH}}(f) = \sum_{c \in f} H^{\text{SSH}}(c)$. By definition, packets that are not counted have adjusted $h()$-values zero.

The HT estimator of $h(c)$ is the ratio of $h(c)$ and the probability that the packet c is counted in the sketch. It is clearly unbiased. This probability, however, can not be computed from the sketch. We use a partition of the sample space such that within each subspace in the partition there is a positive probability that the packet is sampled and this conditional probability can be determined from the sketch. The adjusted $h()$ value for each packet is an application of the HT estimator within this subspace.

We consider the adjusted $h()$ value $H^{\text{SSH}}(c)$ of a counted packet c of a flow f. We partition the sample space such that all rank assignment in the same subspace of the partition share the following.

1. The rank values of packets in $F \setminus \{f\}$.

2. The number of packets of f that are counted continuously up to and not including c (Note that this could be 0).

Note that the subspace that our rank assignment is mapped to also includes rank assignments where f does not appear in the sketch at all or that f appears but c is not counted. This happens if the current sampling rate drops below the current rank of the flow right before or after c is processed.

We compute the conditional probability that c is counted assuming that the rank assignment belongs to the particular subspace that it maps to. Since the ranks of packets of flows in $F \setminus \{f\}$ are fixed in this subspace then so are the steps, $\mathbf{p}$, of the kth smallest rank of a flow in $F \setminus \{f\}$. Furthermore, in any rank assignment in the given partition where packet c is counted, we count the same number of packets in each step. Let $\mathbf{n}$ be the vectors of counts we obtain for f in any rank assignment where c is counted. (In rank assignments in the same subspace where c is not counted this vector could be different.)

$H^{\text{SSH}}(c)$ is computed according to one of the following cases.

1. Packet c is one of the n_i packets counted in step i for some $i > 1$. In this case the conditional probability that c is counted is
$$\frac{q[\mathbf{n}|\mathbf{n}]}{1 - \sum_{h=1}^{i} c_{h,h}} . \tag{6}$$
To see this, fix the ranks of packets of $F \setminus \{f\}$. Then $(1 - \sum_{h=1}^{i} c_{h,h})$ is the probability that we count all $n_1 + \cdots + n_i$ up to and including the packets of step i. $q[\mathbf{n}|\mathbf{n}]$ is the probability that we count all $\mathbf{n}$ packets. Therefore, Eq. (6) is the conditional probability that we count $\mathbf{n}$ given that we counted all packets up to c. It follows that $H^{\text{SSH}}(c) = h(c)\frac{1-\sum_{h=1}^{i} c_{h,h}}{q[\mathbf{n}|\mathbf{n}]}$.

2. Suppose c is the first packet among the n_1 packets of step 1. In this case, the conditional probability that c is counted is $q[\mathbf{n}|\mathbf{n}]$, and $H^{\text{SSH}}(c) = h(c)/q[\mathbf{n}|\mathbf{n}]$.

3. Suppose that c is a packet of step 1 other than the first. Fixing the ranks of packets of flows in $F \setminus \{f\}$ we count the packets of step 1 with probability p_1: That is the probability that the first packet in step 1 is counted. So the conditional probability that c is counted is $q[\mathbf{n}|\mathbf{n}]/p_1$ and $H^{\text{SSH}}(c) = h(c)p_1/q[\mathbf{n}|\mathbf{n}]$.

Let N_i be the set of packets counted in step i, and let c_0 be the first counted packet.

$$H^{\text{SSH}}(f) = \sum_{c \in f} H^{\text{SSH}}(f) = \sum_{j \geq 1} \sum_{c \in N_i} H^{\text{SSH}}(c) =$$

$$\frac{h(c_0) + (h(N_1) - h(c_0))(1 - c_{1,1}) + \sum_{i=2}^{r} h(N_i)(1 - \sum_{h=1}^{i} c_{h,h})}{q[\mathbf{n}|\mathbf{n}]} =$$

$$\frac{h(c_0)c_{1,1} + \sum_{i=1}^{r} h(N_i)(1 - \sum_{h=1}^{i} c_{h,h})}{q[\mathbf{n}|\mathbf{n}]} . \tag{7}$$

To facilitate this estimator, the algorithm needs to collect per-step sums $h(N_i)$ over counted packets in the step and to separately record $h(c_0)$.

8.2 Variance relation

We denote by $A^{\text{L}}(f)$, where L $\in$ {aSH,aNF,sSH,NF,SH,sNF}, the random variable that is the adjusted weight assigned to the flow f by the algorithm L.

An important property of $L \in$ {aSH,aNF,sSH,NF,SH} (established in [2]) is *zero covariances*: For two flows $f_1 \neq f_2$, we have $\text{cov}(A^{\text{L}}(f_1), A^{\text{L}}(f_2)) = 0$. The proof is based on conditioning on the rank values of packets belonging to flows in $F \setminus \{f_1, f_2\}$, and the methodology carries over to establish this property for SNF and for the discretized versions. This property implies that for any $J \subset F$ $\text{VAR}(A^{\mathbf{L}}(J)) = \sum_{f\in J} \text{VAR}(A^{\mathbf{L}}(f))$.

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

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

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 $\text{VAR}(A^{\mathbf{L}}(f)|\mathbf{p})$ for $L \in \{$SSH,SNF$\}$, and $\text{VAR}(A^{\mathbf{L}}(f)|p_r)$ for $L \in \{$WS,ANF,ASH$\}$. This conditioning is equivalent to establishing the variance relation when the sampling rate p_r is fixed or when the steps $\mathbf{p}$ are fixed (and the last step is p_r). It is the key for extending the proofs to the discretized version, since we simply condition on a different step function $\mathbf{p}$ determined by the discretized $(k+1)$th largest rank. It also shows that the variance relation holds for the fixed-rate and fixed-steps variants of WS NF and SH.

For SNF, similarly to SSH, the variance $\text{VAR}(A^{\text{SNF}}(f)|\mathbf{p})$ depends on 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 ANF, when all packets occur on the step with the lowest sampling probability. The variance relation is established using the following Lemma.

LEMMA 8.1. *[2] 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 respective part of the partition. Then $E(A_1) = E(A_1')$ and $\text{VAR}(A_1) \geq \text{VAR}(A_1')$.*

LEMMA 8.2. $\text{VAR}(A^{\text{SSH}}(f)|\mathbf{p}) \leq \text{VAR}(A^{\text{SNF}}(f)|\mathbf{p})$

PROOF. Consider a flow f and a probability space $\Omega^{(\mathbf{p})}$ containing all rank assignments such that the steps (as defined by the kth smallest rank in $F \setminus f$) are $\mathbf{p}$. Consider a partition of Ω to subspaces $\Omega^{(\mathbf{p})}_{\mathbf{n}}$ according to the SSH count vector $\mathbf{n}$ obtained for f.

Consider one such subspace $\Omega^{(\mathbf{p})}_{\mathbf{n}}$. By definition, the adjusted weight assigned to the flow f in this subspace is fixed and is equal to $A^{\text{SSH}}_{\mathbf{p}}(\mathbf{n})$.

We define another SSH adjusted weight assignment, $A'^{\text{SSH}}_{\mathbf{p}}(\mathbf{n})$ as the expectation of the estimator $A^{\text{SNF}}_{\mathbf{p}}$ over rank assignments in $\Omega^{(\mathbf{p})}_{\mathbf{n}}$.

For any rank assignment, a packet is counted by SNF only if it is counted by SSH. The first counted packet by SSH, must also be counted by SNF. Therefore, $\mathbf{s}$ is a possible SNF count of f in $\Omega^{(\mathbf{p})}_{\mathbf{n}}$ if and only if it has the form $\mathbf{s} \leq \mathbf{n}$ (component wise) and $s_1 > 0$ (we assume WLOG that $n_1 > 0$). For notation convenience, we define the vectors $\mathbf{n}' = (1, n_1 - 1, n_2, \ldots, n_r)$, $\mathbf{s}' = (1, s_1 - 1, s_2, \ldots, s_r)$ and $\mathbf{p}' = (p_1, p_1, p_2, \ldots, p_r)$. (That is, we create a "dummy" step with probability p_1, that precedes the first step and contains the first packet of $\mathbf{n}$.) This notation allows us to specify that the first "packet" of $\mathbf{n}$ is counted.

The probability over $\Omega^{(\mathbf{p})}_{\mathbf{n}}$ of a rank assignment with corresponding SNF count $\mathbf{s}$ is equal to

$$q^{\text{SNF}}_{\mathbf{p}'}[\mathbf{s}'|\mathbf{n}']/q^{\text{SSH}}_{\mathbf{p}}[\mathbf{n}|\mathbf{n}] \; .$$

Therefore,

$$A'^{\text{SSH}}_{\mathbf{p}}(\mathbf{n}) = \sum_{\mathbf{s}\leq\mathbf{n}|s_1>0} \frac{q^{\text{SNF}}_{\mathbf{p}'}[\mathbf{s}'|\mathbf{n}']}{q^{\text{SSH}}{\mathbf{p}}[\mathbf{n}|\mathbf{n}]} A^{\text{SNF}}_{\mathbf{p}}(\mathbf{s}) \; . \qquad (8)$$

It follows from Equation (8) that $A'^{\text{SSH}}_{\mathbf{p}}(\mathbf{n})$ is a deterministic function of $\mathbf{p}$ and $\mathbf{n}$.[5] Using corollary 8.1, the estimator $A'^{\text{SSH}}_{\mathbf{p}}$ is unbiased and has variance that is at most that of $A^{\text{SNF}}_{\mathbf{p}}$ over $\Omega_{\mathbf{p}}$ (and therefore, over any probability space that consists of subspaces of the form of $\Omega_{\mathbf{p}}$.)

In Sections 5.1 and 8.1 we show that $A^{\text{SSH}}_{\mathbf{p}}(\mathbf{n})$ is the unique solution of a system of equations. Therefore, it is the only possible assignment of adjusted weights that are a deterministic function of $\mathbf{p}$ and $\mathbf{n}$ and are unbiased (has expectation $|f|$) for any possible f and a corresponding probability space $\Omega^{(\mathbf{p})}$. Since the estimator $A'^{\text{SSH}}_{\mathbf{p}}(\mathbf{n})$ is also a deterministic function of $\mathbf{n}$ and $\mathbf{p}$ and is unbiased on $\Omega^{(\mathbf{p})}$ it follows that $A^{\text{SSH}}_{\mathbf{p}} \equiv A'^{\text{SSH}}_{\mathbf{p}}$.

□

8.3 Adjusted weights for SNF

We provide the proof of Theorem 5.7 (derivation of adjusted weights for SNF). This proof also establishes the variance relation $\text{VAR}(A^{\text{SNF}}(f|\mathbf{p})) \leq \text{VAR}(A^{\text{NF}}(f|p_r))$.

PROOF. We fix the ranks of the packets of flows in $F \setminus f$. The the steps $\mathbf{p}$ of the kth smallest rank of a flow in $F \setminus f$ are then fixed. Let $\mathbf{n}$ be the number of packets of f in each of these steps. (We assume without loss of generality that $n_1 > 0$.) We consider the subspace V of rank assignments where f is fully counted. Let $A^{\text{SNF}}(\mathbf{n})$ be the adjusted weight that f obtains at any point of this subspace. We show that $A^{\text{SNF}}(\mathbf{n})$ is the average of the adjusted weight of NF in V. Note that the fraction of V is $q^{\text{SNF}}[\mathbf{n}|\mathbf{n}]$.

Consider first points in V where the first packet that has rank at most p_r is a packet t of step 1.

The probability that SNF counted all packets and packet $t+1$ was the first packet to obtain rank value at most p_r is $(p_1 - p_r)^t p_r p_1^{n_1-t-1} \prod_{h=2}^r p_h^{n_h}$. Conditioned on this, the adjusted weight assigned by NF is the number of counted packets divided by p_r. The expected number of counted packets from step 1 is $1 + (n_1 - t - 1)(p_r/p_1)$ and the expected number from step $2 \leq j \leq r$ is $n_j(p_r/p_j)$. Therefore, the expected adjusted weight assigned by NF is $(1/p_r + \sum_{j=1}^r n_j/p_j - (t+1)/p_1)$. We sum over $t = 0, \ldots, n_1 - 1$ and divide by $q^{\text{SNF}}[\mathbf{n}|\mathbf{n}]$ to obtain the contribution to the average adjusted weight of NF in V of the points where the first packet that has rank at most p_r is of step 1.

$$\frac{\prod_{h=2}^r p_h^{n_h} \sum_{t=0}^{n_1-1} (p_1 - p_r)^t p_r p_1^{n_1-t-1} \left(\frac{1}{p_r} + \sum_{j=1}^r \frac{n_j}{p_j} - \frac{(t+1)}{p_1}\right)}{q^{\text{SNF}}[\mathbf{n}|\mathbf{n}]} \; . \qquad (9)$$

[5]This also follows the fact that SSH *dominates* SNF (this notion of dominance was defined in [2]) in the sense that SNF sketches can be emulated from SSH sketches. That is, given $\mathbf{p}$ and $\mathbf{n}$, we can draw an SNF sketch from $\Omega^{(\mathbf{p})}_{\mathbf{n}}$.

The derivation of the contribution to the average of points where the first packet having rank at most p_r is in steps $\ell = 2, \ldots, r-1$[6] is similar to that of Eq.(9), observing that $\frac{(d_{\ell,r}-d_{\ell,\ell})}{p_\ell^{n_\ell}}(p_\ell - p_r)^t p_r p_\ell^{n_\ell - t-1}$ is the probability that sNF fully counted the flow and the first packet to obtain rank value at most p_r was packet $t+1$ during step ℓ. We omit the denominator $q^{\text{SNF}}[\mathbf{n}|\mathbf{n}]$ from the following two Equations.

$$\frac{(d_{\ell,r} - d_{\ell,\ell})}{p_\ell^{n_\ell}} \sum_{t=0}^{n_\ell - 1} (p_\ell - p_r)^t p_r p_\ell^{n_\ell - t - 1} \left(\frac{1}{p_r} + \sum_{j=\ell}^{r} \frac{n_j}{p_j} - \frac{t+1}{p_\ell} \right) . \tag{10}$$

We obtain that Eq. (10) (contribution of step $\ell > 1$) is equal to

$$\begin{aligned} & (d_{\ell,r} - d_{\ell,\ell}) \frac{p_r}{p_\ell} \sum_{t=0}^{n_\ell - 1} \left(\frac{p_\ell - p_r}{p_\ell} \right)^t \left(\frac{1}{p_r} + \sum_{j=\ell}^{r} \frac{n_j}{p_j} - \frac{t+1}{p_\ell} \right) \\ = \quad & (d_{\ell,r} - d_{\ell,\ell}) \frac{p_r}{p_\ell} \{ \left(\frac{p_\ell - p_r}{p_\ell p_r} + \sum_{j=\ell}^{r} \frac{n_j}{p_j} \right) \sum_{t=0}^{n_\ell - 1} \left(\frac{p_\ell - p_r}{p_\ell} \right)^t \\ & \frac{1}{p_\ell} \sum_{t=0}^{n_\ell - 1} \left(\frac{p_\ell - p_r}{p_\ell} \right)^t t \} \end{aligned} \tag{11}$$

The first sum in the expression above is geometric, and the second is of the form $\sum_{k=0}^{m} kq^k$ for $q = \frac{p_\ell - p_r}{p_\ell}$. Since $\sum_{k=0}^{m} kq^k = \sum_{k=0}^{m} (f(k+1) - f(k)) = f(m+1) - f(0)$ where $f(x) = \frac{1}{q-1}\left(xq^x - \frac{q^{x+1}}{q-1} \right)$, we obtain that

$$\sum_{k=0}^{m} kq^k = \frac{1}{q-1} \left((m+1)q^{m+1} - \frac{q^{m+2}}{q-1} + \frac{q}{q-1} \right) .$$

Using these observation we obtain that Equation (11) is equal to

$$\begin{aligned} = \quad & (d_{\ell,r} - d_{\ell,\ell}) \{ \left(\frac{p_\ell - p_r}{p_\ell p_r} + \sum_{j=\ell}^{r} \frac{n_j}{p_j} \right) \left(1 - \left(\frac{p_\ell - p_r}{p_\ell} \right)^{n_\ell} \right) \\ & + \frac{n_\ell}{p_\ell} \left(\frac{p_\ell - p_r}{p_\ell} \right)^{n_\ell} + \frac{p_\ell - p_r}{p_\ell p_r} \left(\frac{p_\ell - p_r}{p_\ell} \right)^{n_\ell} - \frac{p_\ell - p_r}{p_\ell p_r} \} \\ = \quad & \left(\frac{p_\ell - p_r}{p_\ell p_r} + \sum_{j=\ell}^{r} \frac{n_j}{p_j} \right) (d_{\ell,r} - d_{\ell,\ell} - d_{\ell+1,r}) \\ & + \frac{n_\ell}{p_\ell} d_{\ell+1,r} + \frac{p_\ell - p_r}{p_\ell p_r} d_{\ell+1,r} - \frac{p_\ell - p_r}{p_\ell p_r} (d_{\ell,r} - d_{\ell,\ell}) \\ = \quad & (d_{\ell,r} - d_{\ell,\ell}) \sum_{j=\ell}^{r} \frac{n_j}{p_j} - d_{\ell+1,r} \sum_{j=\ell+1}^{r} \frac{n_j}{p_j} \end{aligned} \tag{12}$$

By applying similar manipulations to Eq. (9), we obtain that the numerator of that equation is equal to

$$\prod_{h=1}^{r} p_h^{n_h} \sum_{j=1}^{r} \frac{n_j}{p_j} - d_{2,r} \sum_{j=2}^{r} \frac{n_j}{p_j} \tag{13}$$

We sum the contributions of steps $\ell = 1, \ldots, r-1$ (Eq. (13) and Eq. (12) for $\ell = 2, \ldots, r-1$) and obtain that the total contribution to the expectation is

$$\prod_{h=1}^{r} p_h^{n_h} \sum_{j=1}^{r} \frac{n_j}{p_j} - \sum_{\ell=2}^{r} d_{\ell,\ell} \sum_{j=\ell}^{r} \frac{n_j}{p_j} = \sum_{j=1}^{r} \frac{n_j}{p_j} (\prod_{h=1}^{r} p_h^{n_h} - \sum_{\ell=2}^{j} d_{\ell,\ell}) .$$

[6]Observe that the contribution of the last step must be zero unless it is the only step, since if the flow is fully counted its rank must be at most p_r before the beginning of the last step.

Therefore,

$$A^{\text{SNF}}(\mathbf{n}) = \frac{\sum_{j=1}^{r} \frac{n_j}{p_j} (\prod_{h=1}^{r} p_h^{n_h} - \sum_{\ell=2}^{j} d_{\ell,\ell})}{q^{\text{SNF}}[\mathbf{n}|\mathbf{n}]} . \tag{14}$$

The proof follows using Eq.(4). $\square$

By applying the HT estimator to obtain an adjusted $h()$-value for each observed packet, as we did for sSH, we obtain the following expression for adjusted $h()$-values.

$$H^{\text{SNF}}(f) = \frac{\sum_{i=1}^{r} \frac{h(N_i)}{p_i} \prod_{h=1}^{r} p_h^{n_h} - \sum_{j=2}^{i} d_{j,j}}{\prod_{h=1}^{r} p_h^{n_h} - \sum_{j=2}^{r} d_{j,j}} . \tag{15}$$

(where $N_i \subset f$ $(i = 1, \ldots, r)$ is the set of packets of f that are counted at step i.)

9. CONCLUSION

Accurate summarization of IP traffic is essential for many network operations. We design summarization algorithms that generate a sketch of the packet streams that allows us to process approximate subpopulation-size queries and other aggregates. Our algorithms build on existing designs, but are yet able to obtain significantly better estimates through better utilization of available resources and careful derivation of unbiased statistical estimators that have minimum variance with respect to the information they use.

10. REFERENCES

[1] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.*, 55:441–453, 1997.

[2] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Sketching unaggregated data streams for subpopulation-size queries. In *Proc. of the 2007 ACM Symp. on Principles of Database Systems (PODS 2007)*. ACM, 2007.

[3] 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.

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

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

[6] 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.

[7] 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.

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

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

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

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

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

[13] 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.

[14] 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.

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

[16] B. Ribeiro, D. Towsley, T. Ye, and J. Bolot. Fisher information of sampled packets: an application to flow size estimation. In *Proceedings of the 2006 Internet Measurement Conference*. ACM, 2006.

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