Learn More, Sample Less: Control of Volume and Variance in Network Measurement

Nick Duffield, Carsten Lund, Mikkel Thorup
AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932, USA.
{duffield,lund,mthorup}@research.att.com

***Abstract*— This paper deals with sampling objects from a large stream. Each object possesses a size, and the aim is to be able to estimate the total size of an arbitrary subset of objects whose composition is not known at the time of sampling. This problem is motivated from network measurements in which the objects are flow records exported by routers and the sizes are the number of packet or bytes reported in the record. Subsets of interest could be flows from a certain customer or flows from a worm attack. This paper introduces *threshold sampling* as a sampling scheme that optimally controls the expected volume of samples and the variance of estimators over any classification of flows.**

This paper provides algorithms for dynamic control of sample volumes and evaluate them on flow data gathered from a commercial IP network. The algorithms are simple to implement and robust to variation in network conditions. The work reported here has been applied in the measurement infrastructure of the commercial IP network. To not have employed sampling would have entailed an order of magnitude greater capital expenditure to accommodate the measurement traffic and its processing.

***Index Terms*— Estimation, Flows, Internet Measurement, Sampling, Variance Reduction**

I. Introduction

We wish to sample objects from a large stream. Each object possess a size, and our aim is to be able to estimate the total size of an arbitrary subset of objects whose composition is not known at the time of sampling. More formally, consider the following estimation problem. A set of objects $i = 1, 2, \dots, n$, each endowed with a size $x_i \in \mathbb{N}$ and a key c_i taking values in some set K. We wish to estimate subset sums of the form $X(C) = \sum_{i:c_i \in C} x_i$, i.e., the total size of all objects with key in some $C \subset K$ which is *not known at the time of sampling*.

How should the sampling distribution be chosen in order to jointly control both the variance of the estimators $\widehat{X}(C)$ of $X(C)$ and the number of samples taken? This is an abstract version of a practical problem that arises in estimating usage of network resources due to different users and applications. The usage is determined from network measurements, and sampling is employed to control the resources consumed by the measurements themselves. We start this paper by explaining the motivation behind the stated sampling problem, and showing how the constraints imposed by the intended use of the measurements lead us to employ flow sampling.

A. Motivation

1) The need for detailed network usage data: The collection of network usage data is essential for the engineering and management of communications networks. Until recently, the usage data provided by network elements (e.g. routers) has been coarse-grained, typically comprising aggregate byte and packet counts in each direction at a given interface, aggregated over time windows of a few minutes. However, these data are no longer sufficient to engineer and manage networks that are moving beyond the undifferentiated service model of the best-effort Internet. Network operators need more finely differentiated information on the use of their network. Examples of such information include (i) the relative volumes of traffic that use different protocols or applications; (ii) traffic matrices, i.e., the volumes of traffic originating from and/or destined to given ranges of Internet Protocol (IP) addresses or Autonomous Systems (AS's); (iii) the packet and byte volumes and durations of user sessions, and of the individual flows of which they comprise. Such information can be used to support network management, in particular: traffic engineering, network planning, peering policy, customer acquisition, usage-based pricing, and network security; some applications are presented in details in [3], [16], [17]. An important application of traffic matrix estimation is to efficiently redirect traffic from overloaded links. Using this to tune OSPF/IS-IS routing one can typically accommodate 50% more demand; see [20].

From our point of view the central observation is that many network management applications, the traffic is to be regarded as divided into a large number of classes, where the divisions may not be known at the time of measurement, and the input data required by an application is the aggregate traffic volume in each class, over some set of time periods. Satisfying the data needs of the applications requires gathering usage data differentiated by IP header fields (e.g. source and destination IP address and Type of Service), transport protocol header fields (e.g. source and destination TCP/UDP ports), router information specific to a packet (e.g. input/output interfaces used by a packet), information derived from these and routing state (e.g. source and destination AS numbers), or combinations of these. Collecting the packet headers themselves as raw data is infeasible due to volume: we expect that a single direction of an OC48 link could produce as much as 100GB of packet headers per hour, this estimate based on statistics collected for the experiments reported later in this paper.

2) Flow level statistics: In this paper we focus on a measurement paradigm that is widely deployed in the current Internet and that offers some reduction in the volume of gathered data. This is the collection—by routers or dedicated traffic monitors–of IP flow statistics, which are then exported to a remote collection and analysis system. (Some alternative paradigms are reviewed in Section VIII).

Most generally, an IP flow is a set of packets, that are observed in the network within some time period, and that share some common property which we call a "key". Particularly interesting for us are "raw" flows: a set of packets observed at a given network element, whose common property is the set of values of those IP header fields that are invariant along a packet's path. (For example, IP addresses are included, Time To Live is not). The common property may include joins with state information at the observation point, e.g., next hop IP address as determined by destination IP address, and routing policy. Thus the keys characterizing flows are complex multidimensional objects, and the number of potential keys is so enormous (2^{100} or more, depending on what fields are included) as to preclude maintaining counters for each possible key; see also the discussion in Section VIII.

The granularity at which a router differentiates keys is configurable. Applications will typically want to aggregate flows (i.e. form the total size) over subsets of keys specific for their purpose. We can refer to these key subsets also as "keys"; those which cannot be subdivided (i.e. the keys of raw flows) will be termed primary. Since the required aggregations vary across applications, and may not be known at the time of sampling, differentiation of keys in the measurements should be as fine as possible, distinguishing usage according to primary key by collecting statistics on raw flows.

A router keeps statistics—including total bytes and packets—for active flows passing through it. When a packet arrives at the router, the router determines if a flow is active for the packet's key. If so, it updates statistics for that key, incrementing packet and byte counters. If not, is instantiates a new set of counters for the packet's key. A router will designate a flow as terminated if any of a number of criteria are met. When the flow is terminated, its statistics are flushed for export, and the associated memory released for use by new flows. Termination criteria can include (i) timeout: the interpacket time within the flow will not exceed some threshold; (ii) protocol: e.g., observation a FIN packet of the Transmission Control Protocol (TCP) [31] that is used to terminate a TCP connection; (iii) memory management: the flow is terminated in order to release memory for new flows; (iv) aging: to prevent data staleness, flows are terminated after a given elapsed time since the arrival of the first packet of the flow.

Flow definition schemes have been developed in research environments, see e.g. [1], [5], and are the subject of standardization efforts [23], [33]. Reported flow statistics typically include the properties that make up flows defining key, its start and end times, and the number of packets and bytes in the flow. Examples of flow definitions employed as part of network management and accounting systems can be found in Cisco's NetFlow [4], Inmon's sFlow [22], Qosient's Argus [32], Riverstone's LFAP [35] and XACCT's Crane [36].

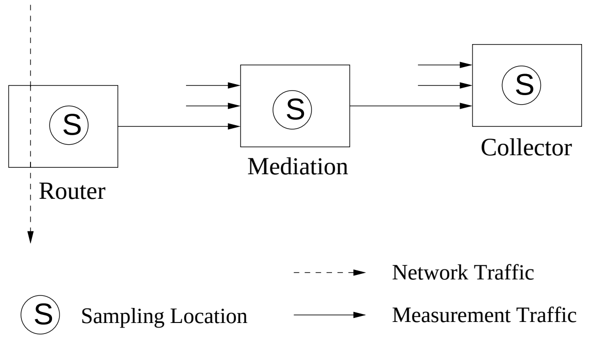

Fig. 1. FLOW OF MEASUREMENT TRAFFIC, AND POTENTIAL SAMPLING POINTS: (left to right) packet capture at router; flow formation and export; staging at mediation station; measurement collector.

Flow statistics offer considerable compression of information over packet headers, since the flow key is specified once for a given flow. For example the previous OC48 example, the volume of flow statistics is roughly 3GB per hour, i.e., up to a roughly 30-fold volume decrease compared with packet headers, depending on the balance of long and short flows in the traffic.

3) Measurement collection architecture: Figure 1 illustrates an architecture for measurement collection that has been realized in a commercial network. Flow level summaries are constructed at a router from the packets that traverse it. Records containing the summaries are transmitted to the measurement collector, possibly through one or more mediation stations. These may be employed for reasons of reliability: measurement export commonly uses the User Datagram Protocol (UDP), which has no facility to detect and resend measurements lost in transit. However, export to a nearby mediation station over a Local Area Network (LAN) may in practice be very reliable, as compared with export over the Internet to a distant collection point. The mediation station may then reliably transmit measurements to the ultimate collection point, the requisite memory and processing resources being cheaper in workstations than in the routers. The mediation station can also perform aggregation to support distributed network management applications.

4) Data volumes and the need for sampling: The volumes of flow records generated by a large network with many interfaces is potentially massive, placing large demands on memory resources at routers, storage and computation resources at the collector, and transmission bandwidth between them. This motivates *sampling* the flow record to reduce data volume, while at the same time maintaining a representative view of raw flow data. Flow records can be sampled by any system on the path from generation to collection. This may be necessary for scaling reasons. The architecture forms a tree, with multiple routers sending measurements to each of several mediation stations. Progressive resampling can be applied at each stage to avoid "implosion" as data progresses up the tree. At the collector or mediation station, a typical analysis application executes a query that performs a custom aggregation (i.e. one not previously performed) over all flows collected in a given time period. Here, the role of sampling is to reduce the execution time of the query. For final storage at the collector, sampling can be used to permanently reduce the volume of historical flow data (non-sampled flows would be discarded) while maintaining the ability to execute custom queries at the finest differentiation of keys.

5) Effectiveness of sampling methods: An elementary sampling method is to uniformly select 1 in N of the flows, either independently (i.e. each object is selected independently with probability $1/N$) or deterministically (objects $N, 2N, \dots$ are selected and all others are discarded). The statistical properties of any proposed sampling scheme must be evaluated. Do inferences drawn from the samples reflect the properties of the raw data stream? What is the impact of sampling on the variance of usage estimates?

A striking feature of flow statistics is that the distributions of the number of packet and bytes in flows are heavy-tailed [19]. This property contributes to the roughly 30-fold average compression of data volumes when passing from packet headers to flows that was remarked upon above. Uniform sampling from heavy tailed distributions is particularly problematic, since the inclusion or exclusion of a small number of data points can lead to large changes in usage estimates; these are subject to high variance due to the sampling procedure itself. Note that a sampling strategy that samples all big flows and a fair fraction of the smaller flows could reduce the estimator variance. This raises the question: which is the best such sampling strategy? This is the problem we study in this paper.

6) Knowledge of sampling parameters: Sampling parameters used for flow selection must be known when the data is analyzed, in order that it can be interpreted correctly. For example, to estimate the byte rate in a raw packet stream from samples gathered through 1 in N sampling, we need to multiply the byte rate represented in the sampled stream by N. Since raw traffic volumes vary by location and time, we expect that sampling rates will have to vary in order to limit resource usage due to measurement to acceptable levels. For example, an unexpected surge in traffic may require dynamic lowering of sampling rates.

Generally, then, sampling rates will not be global variables independent of the data. We believe sampling parameters must be bound to the data (e.g. to individual measurements, or inserted into the stream of measurements). Each entity that samples or resamples the flows must bind its sampling parameters to the data appropriately. We eschew keeping sampling parameters in a separate database to be joined with the data for analysis. This is not robust against bad synchronization between the two data sets, or undocumented manual changes of sampling parameters.

B. Contribution

1) Sampling requirements and the basic problem: The contribution of this paper is to study what makes a good flow sampling strategy. Here the flows are represented by the flow records exported by the routers. We do not interfere with the mechanism for creating these records. The goals are fourfold: (i) to constrain samples to within a given volume; (ii) to minimize the variance of usage estimates arising from the sampling itself; (iii) to bind the sampling parameters to the data in order that usage can be estimated transparently; and (iv) with progressive resampling, the composite sampling procedure should enjoy properties (i)–(iii).

To formalize the problem, recall the set of flows labeled $i = 1, 2, \dots, n$ equipped with a size x_i and key c_i. We wish to sample them in such a way that we can estimate the total size of flows with keys in a set C, i.e., the sums $X(C) = \sum_{i:c_i \in C} x_i$, knowing only the sampled sizes and their sampling probabilities, without the need to retain other information on the original set of flow sizes.

We propose to continuously stratify the sampling scheme so the probability that an flow record is selected depends on its size x. This attaches more weight to larger flows whose omission could skew the total size estimate, and so reduce the impact of heavy tails on variance. We must renormalize the sampled sizes x_i in order that their total over any key set C becomes an unbiased estimator of $X(C)$; this will be achieved by coordinating the renormalization with the sampling probabilities. We will show that our sampling scheme can be optimized in the sense of minimizing a cost function that expresses the undesirability of having either a large number of samples, or a large variance in estimates formed from them. Sampling with this optimal choice of sampling probabilities will be characterized by a certain threshold, and we name it *threshold sampling* and we shall use this term in the paper.[1] Finally, we require a mechanism to tune the sampling probabilities in order that the volume of sampled records can be controlled to a desired level, even in the presence of temporal or spatial inhomogeneity in the offered load of flows.

2) Outline: Section II develops the basic theory of the sampling method. Section II-A establishes a general relationship between the sampling probabilities (as a function of objects size) and the renormalization (applied to the sizes) in order that the estimates of $X(C)$ be unbiased. We then turn to examine the relationship between sample volume and estimator variance due to sampling. In Section II-B we point out that uniform sampling offers no further ability to control the variance once the average sampling volume is fixed. Instead, in Section II-C, we show how the sampling probabilities should be chosen in order to minimize a cost function that takes the form of a linear combination of sample volume and estimator variance. In Section II-D we show that optimizing the cost function for the total traffic stream, also optimizes for each key individually.

We follow up with a number of subsidiary results. In Section II-E we extend the formalism to multidimensional size attributes, e.g. to use both packet and byte sizes of flows. In Section II-F we describe unbiased estimators of the variance of the sample volume and attribute total. In Section II-G we show that the set of sampling operations considered is closed under certain compositions; consequently, the total effect of a set of sampling operations applied successively (e.g. at a router, then at mediation station) is statistically equivalent to

[1]The term *smart sampling* has also been used to encompass this and other related sampling methods.

a single sampling operation with appropriate parameters.

In Section III we demonstrate how the method achieves our aims by applying it to datasets of NetFlow statistics gathered on the backbone of a major service provider. In Section III-A we show that the variance of usage estimation from flow records is greatly reduced by using size dependent sampling as opposed to uniform sampling. In Section III-B we find that even when packet sampling is mandated at a router (e.g. by software speed constraints), any further desired reduction in measurement volume is best achieved (i.e. with noticeably smaller usage estimator variance) by forming flows and applying size dependent sampling, as opposed to dispensing with flow formation altogether and just performing further packet sampling. In Section III-C we describe an efficient algorithm for size dependent sampling used in the experiments that avoids the need for explicit pseudorandom number generators.

In Section IV we extend the method to the dynamic control of sample volumes. The cost function described above contains a positive parameter z that encodes the relative importance we attach to constraining sample volumes as opposed to reducing sample variance. We present an iterative scheme by which z can be adjusted in order that the expected sample volume meets a given constraint. This is useful because even a static flow length distribution may not be well characterized in advance. In this case it would be difficult to determine in advance the value of z needed in order to meet a given sampling volume constraint. We analyze the rate of convergence of the iteration.

Substituting mean sample volume with the actual number of samples taken in the iterative scheme yields an algorithm to dynamically control sample volumes. This enables control of sample volumes under variation in the offered load of flows, e.g., during a sharp rise in the number of short traffic flows commonly that can occur during denial of service attacks [26]. Several variants of the control are discussed in Section V. In Section VI we illustrate the effectiveness of such controls with the NetFlow traces that exhibit large transient phenomena. An efficient randomized algorithm for a root finding problem that arises in our work in described in Section VII, along with a Large Deviation-based analysis of its performance under data resampling. The feasibility of a number of alternative measurement schemes and related work are discussed in Section VIII. We conclude in Section IX by outlining a current application in a commercial network of the work described here, and listing some further developments.

II. Sampling Theory

A. Sampling and renormalization

The key elements of our algorithm are size-dependent sampling, and renormalization of the samples. These are described by the following two functions. A *sampling function* is a function $p : \mathbb{R}_+ \to [0,1]$. The interpretation of the sampling function is that attribute x is to be sampled with probability $p(x)$. $\mathcal{P}$ will denote the set of sampling functions. A *renormalization function* r is a function $\mathbb{R}_+ \to \mathbb{R}_+$. A sampled size x is then renormalized by replacing it with $r(x)$.

Consider a set of n sizes $\{x_i\}_{i=1,\ldots,n}$ prior to sampling. Initially we can think of this set either as the sizes of all objects, or as the set sizes of objects of a particular key. The total size of the n objects is

$$X = \sum_{i=1}^{n} x_i. \tag{1}$$

Suppose first that we wish to obtain an estimate of X from a subset of sampled values, and, generally, without needing to know the original number n of sizes. Consider an estimate of X comprising a sum of sampled sizes that are then renormalized:

$$\widehat{X} = \sum_{i=1}^{n} w_i r(x_i) \tag{2}$$

where the $\{w_i\}_{i=1,\ldots,n}$ are independent random indicator variables, w_i taking the value 1 with probability $p(x_i)$ and 0 with probability $1 - p(x_i)$.

Denote by $\mathsf{E}\widehat{X}$ the expected value of $\widehat{X}$ over the distribution of the random variables $\{w_i\}$. In what follows we shall treat the sizes $\{x_i\}$ as a given deterministic set: randomness resides only in the $\{w_i\}$. $\widehat{X}$ is said to be an unbiased estimator of X if $\mathsf{E}\widehat{X} = X$. Since $\mathsf{E}w_i = p_i$, $\widehat{X}$ is unbiased if

$$X = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} p(x_i) r(x_i) = \mathsf{E}\widehat{X} \tag{3}$$

This happens for all collections $\{x_i\}$ if and only if

$$r(x) = x/p(x)\,, \quad \text{for all } x. \tag{4}$$

In the rest of this paper we will assume that (4) holds.

B. Bounding sample volumes

The sample volume, i.e, the number of samples, is $\widehat{N} = \sum_{i=1}^{n} w_i$. Suppose now that we wish to impose a limit on the expected sample volume $\mathsf{E}\widehat{N}$. Consider first the case of drawing samples from a set known size n. The expected number of samples is thus less than some target M if $\mathsf{E}\widehat{N} = \sum_{i=1}^{n} p(x_i) \le M$. For this to be true for all collections $\{x_i\}$ requires that $p(x_i) = M/n$ for all n: the sampling probability $p(x)$ is independent of x.

The choice of a constant sampling function has an interesting and potentially troubling consequence. This is that is there is no further latitude in the choice of the sampling function that could be used to attain some other desirable statistical property of $\widehat{X}$, such as keeping its variance small. Indeed, our motivating example comes from cases where the the $\{x_i\}$ have a heavy tailed distribution, and hence the inclusion or exclusion of a small number of sample points can have a great influence on $\widehat{X}$.

One approach to this problem would be to explicitly take into account the distribution of x when choosing the sampling function. This would entail choosing a non-constant p such that the report volume constraint would be satisfied for a class of size distributions, although not for all. This approach has the potential disadvantage that if the size distribution does not fall into the given class, the constraint will no longer be satisfied. Furthermore, it is not clear in examples that the byte or packet distributions can be characterized in a universal fashion, independently in changes in network technologies and applications.

C. Static control of sample volume and variance

Instead, we take an approach which allows us to jointly control the volume of samples $\widehat{N}$ and the variance of the estimator $\widehat{X}$ without assumptions on the distribution of the sizes $\{x_i\}$. We form a cost function C that embodies our aim of controlling the variance of $\widehat{X}$ and the expected number of samples $\mathsf{E}\widehat{N}$. For $z \in \mathbb{R}$ we define for each $p \in \mathcal{P}$

$$C_z(p) = \mathsf{Var}\,\widehat{X} + z^2\mathsf{E}\widehat{N}. \tag{5}$$

z is a parameter that expresses the relative importance attached to minimizing $\mathsf{E}\widehat{N}$ versus minimizing $\mathsf{Var}\,\widehat{X}$. The variance of $\widehat{X}$ is

$$\begin{aligned}\mathsf{Var}\,\widehat{X} &= \mathsf{Var}\sum_{i=1}^{n} w_i r(x_i) = \sum_{i=1}^{n} r^2(x_i)\,\mathsf{Var}\,w_i \\ &= \sum_{i=1}^{n} r^2(x_i)(1-p(x_i))p(x_i) \\ &= \sum_{i=1}^{n} x_i^2(1-p(x_i))/p(x_i). \end{aligned}\tag{6}$$

Thus

$$C_z(p) = \sum_{i=1}^{n}\left(x_i^2(1-p(x_i))/p(x_i) + z^2 p(x_i)\right) \tag{7}$$

Define $p_z \in \mathcal{P}$ by $p_z(x) = \min\{1, x/z\}$.

Theorem 1: $C_z(p_z) \le C_z(p)$ for all $p \in \mathcal{P}$ and $\{x_i\}$, with equality only if $p = p_z$.

Proof: $q \mapsto x^2(1-q)/q + z^2 q$ is strictly convex on $(0,\infty)$ and minimized at $q = x/z$. Thus it is minimized on $(0,1]$ by $p_z(q)$, and the result follows. ∎

We can interpret the form of p_z as follows. High values of the size x, those greater than the threshold z, are sampled with probability 1, whereas lower values are sampled with progressively smaller probability. Thus the contributions of higher values of x to the estimator $\widehat{X}$ have greater reliability than smaller values. This is desirable since uncertainty about the higher values can adversely impact the variance of $\widehat{X}$, especially for heavy tailed distributions of x. We note that for the sampling function p_z, the renormalization function takes the simple form $r_z(x) = \max\{x, z\}$. We write $\widehat{X}_z$ to denote the specific random variable arising from the choice of z.

It is worth noting that unbiased estimators formed using *any* renormalization function of the form (4) will have non-zero variance in general. In the present case, the renormalization $x \mapsto \max\{x, z\}$ can give rise to a large estimated usage z of a flow of small size x. This might be though of as a disadvantage for some applications, e.g., usage-based charging, where it is important not to overestimate usage. However, it is possible to couple charging scheme to the present threshold sampling method in such a way that the estimated usage is relatively insensitive to the inherent estimation variance from threshold sampling; see [12] for further details.

D. Sampling and key partitions

Recall from the introduction, that in our typical application, we think of the packets as being keyed, and that our aim is to estimate the total sizes of the packets with key c of interest. If c_i is the key of packet i, $X(c) = \sum_{c_i=c} x_i$ is the total size of packets with key c, and our unbiased estimator is then $\widehat{X}(c) = \sum_{c_i=c} w_i r(x_i)$, that is, $\widehat{X}(c)$ is obtained simply by summing the sampled normalized sizes of key c. Let $\widehat{N}(c) = \sum_{c_i=c} w_i$ be the number of sampled packets with key c. By linearity of expectation, $\mathsf{E}N = \sum_c \mathsf{E}\widehat{N}(c)$. Also, since each x_i is picked independently, the $\widehat{X}(c)$ are independent for each c, and hence $\mathsf{Var}\,X = \sum_c \mathsf{Var}\,\widehat{X}(c)$. Thus,

$$C_z(p) = \mathsf{Var}\,\widehat{X} + z^2\mathsf{E}\widehat{N} = \sum_c \{\mathsf{Var}\,\widehat{X}(c) + z^2\mathsf{E}\widehat{N}(c)\} \tag{8}$$

That is, our objective function $C_z(p)$ minimizes itself locally over each key class. One could easily imagine scenarios where one wanted different objectives for different keys. However, in our application, the sampling device is not assumed to distinguish keys, and in addition, we imagine that our samples may latter be analyzed with respect to many different aggregate key definitions. Theorem 1 show that the strength of our sampling strategy is that it is the unique optimal strategy with respect to (8), no matter how the keys are defined.

Indeed, finer control of sampling by key, within a given volume constraint, can only increase estimator variance. Suppose that we wish to control individually the sample volume M_c arising from each key c while achieving the same total sample volume $M = \sum_c M_c$ over all keys. This would be achieved by applying a differing threshold z_c to the sampling packets from each key c. The price of imposing finer grained volume control is to increase the aggregate variance of the $\widehat{X}(c)$. The following is a direct corollary of (8) Theorem 1 on noting that the $p_{z_c} \ne p_z$ is suboptimal for $C_z(p)$.

Corollary 1: Let $\{\mathcal{S}_c : c = 1, \dots, j\}$ be a partition of $\{1, \dots, n\}$, and for each member c of the partition let $M_c < \#\mathcal{S}_c$ be a target sample volume. Suppose z_c^* solves $z = \sum_{i\in\mathcal{S}_c}\min\{x_i, z\}/M_c$ and z^* solves $z = \sum_{i=1}^{n}\min\{x_i, z\}/\sum_c M_c$. (Such z^* and z_c^* exist by Theorem 4 following). Let $\widehat{X}_z(c) = \sum_{i\in\mathcal{S}_c} w_i r_z(x_i)$ with the w_i distributed according to p_z. Then

$$\sum_c \mathsf{Var}\,\widehat{X}_{z^*}(c) \le \sum_c \mathsf{Var}\,\widehat{X}_{z_c^*}(c). \tag{9}$$

Proof: Let $N_z^j = \sum_{i\in S_j} w_i r_z(x_i)$ with the w_i distributed according to p_z. By Theorem 1,

$$\sum_j(\mathsf{Var}\,X_{z^*}^j + (z^*)^2\mathsf{E}\widehat{N}_{z^*}^j) \le \sum_j(\mathsf{Var}\,X_{z_j^*}^j + (z^*)^2\mathsf{E}\widehat{N}_{z_j^*}^j). \tag{10}$$

But $\sum_j \mathsf{E}\widehat{N}_{z^*}^j = \mathsf{E}\widehat{N}_{z^*} = M = \sum_j M_j = \sum_j \mathsf{E}N_{z_j^*}^j$ and hence the result follows. ∎

E. Multidimensional sizes

In practical cases, the sizes x can be multidimensional. For example, a flow record can contain both byte and packet counts for the flow. Estimation of multiple components of the size may be required. One approach would be to apply the foregoing analysis to each component independently. However, this would require a separate sampling decision for each size, leading to the creation of multiple sampled streams. This is undesirable, both for the extra computation required, and the increased volume of samples resulting. Another approach is to base sampling on one component x_i (e.g. flow byte size) and then use the factor $1/p_z(x_i)$ to renormalize other components y_i. Although this results in an unbiased estimator for $\sum_i y_i$, it does not minimize the corresponding cost function for the components y_i.

Instead, we outline a simple extension of the previous section that creates only single samples per size. Consider a multidimensional size $\boldsymbol{x} = (x(1), \ldots, x(m)) \in \mathbb{R}^m$ presented by each potential sample. Analogous to the one-dimensional case we have now the sampling function $p : \mathbb{R}^m \to [0,1]$ and renormalization function $\boldsymbol{r} : \mathbb{R}^m \to \mathbb{R}^m$. Given a set of sizes $\{\boldsymbol{x}_i\}$, the binary random variable w_i takes the value 1 with probability $p(\boldsymbol{x}_i)$.

Similar to before, one can show that $\widehat{\boldsymbol{X}} = \sum_i w_i \boldsymbol{r}(\boldsymbol{x}_i)$ is an unbiased estimator of $\boldsymbol{X} = \sum_i \boldsymbol{x}_i$ for all size sets $\{\boldsymbol{x}_i\}$ if and only if $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{x}/p(\boldsymbol{x})$ for all $\boldsymbol{x}$. Let $\boldsymbol{z}^j \in \mathbb{R}^m$ denote the vector with components $(z(1)^j, \ldots, z(m)^j)$. Suppose now we wish to minimize a cost function of the form.

$$C^1_{\boldsymbol{z}}(p) = \mathsf{Var}(\boldsymbol{X} \cdot \boldsymbol{z}^{-1}) + \sum_{i=1}^n p(\boldsymbol{x}_i) \tag{11}$$

Similarly to Theorem 1, one finds that $C_{\boldsymbol{z}}(p_{\boldsymbol{z}}) \leq C_{\boldsymbol{z}}(p)$ for all sampling function p for the function $p^1_{\boldsymbol{z}}(\boldsymbol{x}) = \min\{1, \boldsymbol{x} \cdot \boldsymbol{z}^{-1}\}$.

Variants of this approach are possible. Consider replacing the variance of the single variable $\mathsf{Var}\, \boldsymbol{X} \cdot \boldsymbol{z}^{-1}$ with the sums of variances of the form $\sum_j \mathsf{Var}(X(j)/z(j))$. The corresponding cost function is

$$C^2_{\boldsymbol{z}}(p) = \sum_{j=1}^n \frac{\mathsf{Var}\, X(j)}{z^2(j)} + \sum_{i=1}^n p(\boldsymbol{x}_i). \tag{12}$$

The sampling function p that minimizes $C^2_{\boldsymbol{z}}(p)$ is $p^2_{\boldsymbol{z}} = \min\{1, \sqrt{\boldsymbol{x}^2 \cdot \boldsymbol{z}^{-2}}\}$.

F. Estimating variance from samples

Although Theorem 1 allows us to minimize a linear combination of sample volume and variance, it does not provide a means of estimating the variance of $\widehat{X}$ direct from the sample since the variance is expressed as a sum over all sizes x_i, not just those that have been sampled. However, we can apply the same ideas as were used above to find an unbiased estimate of X, but now finding an unbiased estimate $\widehat{V}_X$ of $\mathsf{Var}\, X$.

The restriction that $\widehat{V}_X$ be computed only from the samples requires it to be of the form $\widehat{V}_X = \sum_{i=1}^n w_i v(x_i)$ for some function v. For unbiasedness we require that $\mathsf{E}\widehat{V}_X = \mathsf{Var}\,\widehat{X}$ for all $\{x_i\}$. This requirement is equivalent to $\sum_{i=1}^n p(x_i)v(x_i) = \sum_{i=1}^n x_i^2(1-p(x_i))/p(x_i)$ for all $\{x_i\}$. Hence

$$v(x) \quad = \quad \left(\frac{x}{p(x)}\right)^2 (1-p(x)), \tag{13}$$

and

$$\widehat{V}_X \quad = \quad \sum w_i \left(\frac{x_i}{p(x_i)}\right)^2 (1-p(x_i)). \tag{14}$$

The specific forms of $\widehat{V}_{X_z}$ and $\mathsf{E}\widehat{V}_{X_z}$ arising from the choice $p = p_z$ reduce to

$$\widehat{V}_{X_z} = \sum_i w_i z(z-x_i)^+, \quad \mathsf{E}\widehat{V}_{X_z} = \sum_i x_i(z-x_i)^+, \tag{15}$$

where $y^+ = \max\{0, y\}$. It is interesting to note that there is no contribution to $\widehat{V}_X$ for $x_i > z$. This is because such x_i are selected with probability 1, and hence their contribution to $\widehat{X}$ has no variance. Indeed, the summand in the expression for $\mathsf{E}\widehat{V}_X$ is zero for $x \geq z$, and maximized when $x = z/2$.

By similar reasoning we can find and unbiased estimator $\widehat{V}_N$ of the variance of N. Writing $\widehat{V}_N = \sum_i w_i u(x_i)$ and requiring $\mathsf{E}\widehat{V}_N = \sum_i p(x_i)u(x_i) = \sum_i p(x_i)(1-p(x_i)) = \mathsf{Var}(\widehat{N})$ for all $\{x_i\}$, we obtain $u(x) = (1-p(x))$ and hence

$$\widehat{V}_{N_z} \quad = \quad \sum_i w_i(1-x_i/z)^+ = z^{-1}\widehat{V}_{X_z}, \tag{16}$$

and

$$\mathsf{E}\widehat{V}_{N_z} \quad = \quad \sum_i (x_i/z)(1-x_i/z)^+ = z^{-1}\mathsf{E}\widehat{V}_{X_z}. \tag{17}$$

As with (15), terms with $x_i \geq z$ vanish identically, and the largest contributions to $\mathsf{E}\widehat{V}_{N_z}$ arise when $x = z/2$.

G. Composing sampling operations

The sampling and renormalization operations defined above can be composed. We envisage such composition when data is fed forward through a number of processing stages operating in different locations, or under different volume or processing constraints. Consider then a composition of m sampling procedure controlled by thresholds $z_1 \leq \cdots \leq z_m$. The threshold z increases at each stage of the composition, corresponding to progressively finer sampling. Let $\widehat{N}_{z_1,\ldots z_j}$ denote the number of samples present after passing through the j samplers in the composition, and $\widehat{X}_{z_1,\ldots z_j}$ the corresponding unbiased estimator of X. The following result says that the expected sample volume and variance of X at stage j are equal to those that would result from a single sampler controlled by threshold z_j. Given a set Ω of sizes, let $S_z(\Omega)$ denote the set of sampled and renormalized sizes obtained using the threshold z, i.e.,

$$S_z(\Omega) = \{\max\{z, x\} : x \in \Omega, w_x = 1\} \tag{18}$$

where w_x are independent random variables taking the value 1 with probability $p_z(x)$ and 0 otherwise.

Theorem 2: Let $0 < z_1 \leq \cdots \leq z_j$. Then for each set Ω of size, $S_{z_j}(S_{z_{j-1}} \ldots (S_{z_1}(\Omega)\ldots)$ has the same distribution as $S_{z_j}(\Omega)$. In particular $\mathsf{E}\widehat{N}_{z_1,\ldots z_j} = \mathsf{E}\widehat{N}_{z_j}$ and $\mathsf{Var}\,\widehat{X}_{z_1,\ldots z_j} = \mathsf{Var}\,\widehat{X}_{z_j}$.

Proof: Any attribute that survives as far as stage j of sampling is renormalized according to $r_{z_1,\ldots,z_j}(x) := r_{z_j} \circ \cdots \circ r_{z_1}(x) = \max\{z_j, \ldots, z_1, x\} = \max\{z_j, x\} = r_{z_j}(x)$. It survives with probability $p_{z_1,\ldots,z_j}(x)$ that obeys the recursion $p_{z_1,\ldots,z_j}(x) = p_{z_1,\ldots,z_{j-1}}(x) p_{z_j}(r_{z_1,\ldots,z_{j-1}}(x))$. But $p_{z_j}(r_{z_1,\ldots,z_{j-1}}(x)) = \min\{1, \max\{z_{j-1}, x\}/z_j\}$. Since $\min\{1, x/z_{j-1}\}\min\{1, \max\{x, z_{j-1}\}/z_j\} = \min\{1, x/z_j\}$ when $z_{j-1} \leq z_j$, a simple induction show that $p_{z_1,\ldots,z_j}(x) = p_{z_j}(x)$. Thus the renormalization and sampling probability for the chain is determined by the last component z_j, and the stated properties follow immediately. ∎

H. Threshold sampling of packet sampled flows

We emphasize again that in our network application it is the completed flow records that are sampled, not the packets that contribute to them. In some routers packets may be sampled prior to the formation of flow statistics, e.g., in Sampled NetFlow; see [4]. In this case, our technique is applied to the flow summaries so formed, and hence to the estimation of the volumes of sampled packets.

To estimate the volumes of traffic prior to packet sampling, it is necessary to apply an initial normalization to the flow byte or packet sizes before threshold sampling. With packet sampling at a rate 1 in N, the sizes are multiplied by N in order to obtain an unbiased estimate of traffic volumes prior to packet sampling. The random selection of packets also contributes to estimation variance. In [11] it is shown that with independent packet sampling at rate $1/N$, the effect is to increase estimator variance by a quantity no larger than $(N-1)Xx_{\max}$ where $x_{\max}$ is the largest packet size in flow prior to sampling.

III. Comparison of Static Sampling Strategies

In this section we perform an experimental comparison of the proposed threshold flow sampling strategies with other sampling strategies. We show that, out of all the compared methods, threshold flow sampling provides the least estimator variability for a given volume of flow measurements.

A. Comparison of uniform and threshold flow sampling

In this section we compare the experimental performance of threshold sampling and uniform sampling on a set of flow records. This comparison is relevant for selecting a sampling method for the selective export of flows, or at a mediation station or measurement collector, or within a database used to store flows.

Our data for the experiments comprised a trace of NetFlow statistics collected during a two hour period from a number of routers in a commercial IP backbone. The total sample comprised 108,064,914 individual flow records. The mean number of bytes per flow was 10,047. We partitioned the total sample according to 3,500 keys, determined by the router interface that generated the record, and by a partition of the address space of the external IP address of the flow record (source address for incoming flows, destination address for outgoing flows). Thus the exact partition was somewhat arbitrary, but it satisfied the following criteria: (i) the had large variation in total bytes (from less than 100 bytes to more than 10^{11} bytes), (ii) the partition was not based on the byte values, and (iii) there were a reasonably large number of keys. Our implementation of the threshold sampling strategy is described in Section III-C below. Our aim here is to exhibit the relative accuracy of the sampling methods in estimating the byte size of the individual keys, and particularly for the "important" keys, i.e., those with large bytes sizes.

We compared two sampling strategies: uniform33 which is 1 in N sampling with $N = 33$, and sample100K which is threshold sampling with threshold $z = 100,000$. We chose these (relative) values in order that the total number of records sampled in each case would be similar: 3.03% and 3.04% of the records respectively for the two strategies. We compared three sets of numbers: $X(c)$, the actual number of bytes for key c; $\widehat{X}(c)$, the number of bytes for key c as determined by threshold sampling and renormalization with sample100K; and $\widehat{X}_{\text{un}}(c)$, the number of bytes for key c as determined with uniform33 sampling and renormalization through multiplication by $N = 33$,

In Figure 2 we plot nonsampled byte counts $X(c)$ and sampled byte counts ($\widehat{X}(c)$ in the left plot, $\widehat{X}_{\text{un}}(c)$ in the right plot) against key index c, with the groups sorted in decreasing order of $X(c)$. On the vertical axis the byte counts are displayed on a log-scale. When no record from a group were sampled, we set the log byte count to 1: these are the points that lie on the horizontal axis. Note that otherwise $\widehat{X}_{\text{un}}(c) \geq 100,000$ since sample100K never yields a value less than the threshold z. Figure 3 is similar to Figure 2 except we have zoomed into a small number of groups in the middle. We have added error bars for the sampling error, corresponding to 2 standard deviations computed using (15). In Figure 4 we similarly compare the strategies through the relative error, i.e., $(X(c) - \widehat{X}(c))/X(c)$ in the left plot, and $(X(c) - \widehat{X}_{\text{un}}(c))/X(c)$ in the right plot. The vertical axes display the relative error as a percentage, chopped to lie between $\pm 100\%$.

It is clear from the Figures that sample100K achieves a considerable reduction in estimator variance over uniform33 for groups with a fair amount of traffic, say at least 1,000,000 bytes. This is exactly the desired behavior: we get good estimates for important groups. It also shows how bad uniform sampling is in the case with heavy-tail sizes: even for group with significant number of bytes the uniform33 strategy can be very poor. The root mean square error is about 500 times greater for uniform than for threshold sampling.

B. Comparison of packet and flow sampling methods

We investigate the relative tradeoffs of sample volume against estimation accuracy for sampling methods at routers, and in particular the relative effectiveness of packet and flow sampling methods. Packet sampling may be required at the router even if flow statistics are constructed, since flow cache lookup may be infeasible at the router interface line rate. Uniform periodic or random sampling is typically employed in practice. A recent proposal, sample and hold [15], is to sample potential new flow cache entries in order to focus on longer flows.

As in Section III-A we used a trace of NetFlow records to perform the comparison. There were 17,898,241 flows divided into 13,110 keys according to destination IP address. Each flow record details the duration t of the flow, and the total packets k and bytes b that it comprises. We want to determine the average amount F of measurement traffic that would arise if the packet stream represented in the flow records with parameter (k, b, t) were subjected to a given sampling strategy. For a pure packet sampling method, i.e. with no flow statistics formed, then F is the number of sampled packets. With flow statistics (whether formed from a full or sampled packet stream) F is the

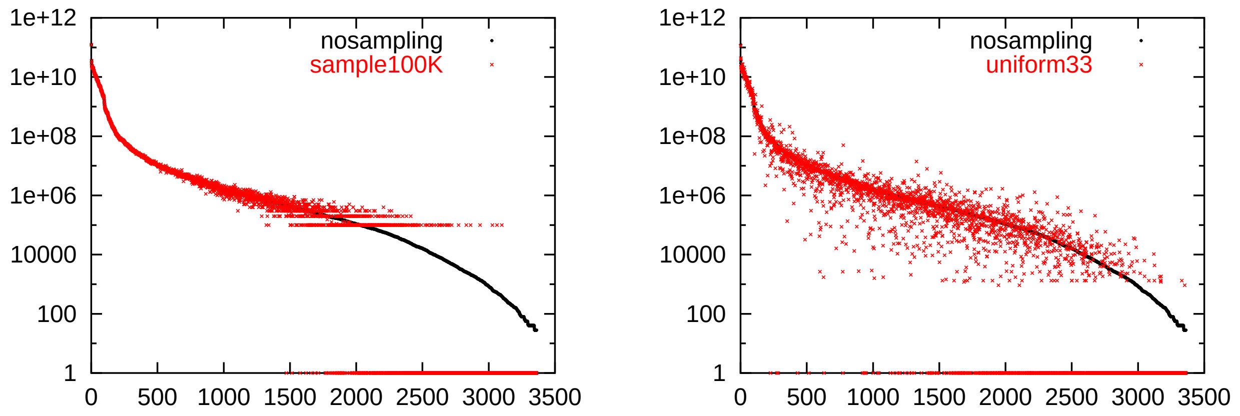

Fig. 2. COMPARING THRESHOLD AND UNIFORM SAMPLING. Left: threshold, $\widehat{X}(c)$ and $X(c)$ vs. key index. Right: uniform, $\widehat{X}_{\mathrm{un}}(c)$ and $X(c)$ vs. key index. Note increased variability of $\widehat{X}_{\mathrm{un}}(c)$ ad compared with $\widehat{X}(c)$.

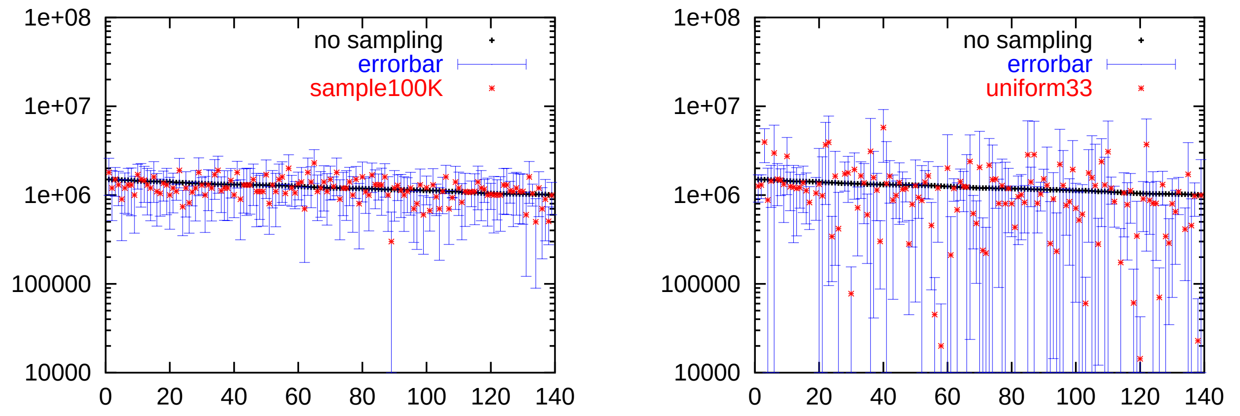

Fig. 3. COMPARING THRESHOLD AND UNIFORM SAMPLING: ERROR BARS: portions of corresponding plots from Figure 2 with error bars for sampling variability added.

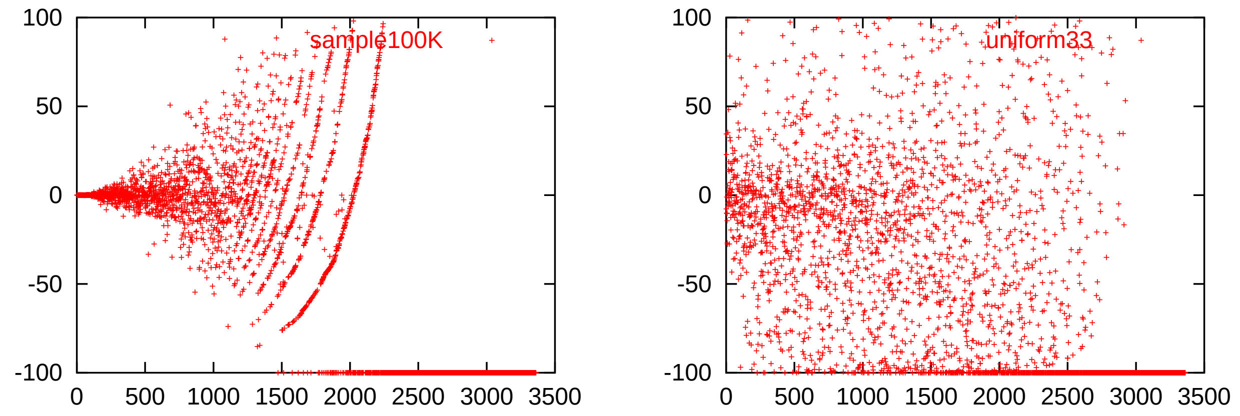

Fig. 4. COMPARING THRESHOLD AND UNIFORM SAMPLING: RELATIVE ERROR. Left: threshold, $(X(c) - \widehat{X}(c))/X(c)$. Right: uniform, $(X(c) - \widehat{X}_{\mathrm{un}}(c))/X(c)$.

final number of flow statistics formed. V will denote a mean squared error of byte usage estimated from the measurement traffic generated from the packet stream of the original flow (k, b, t). For the unbiased estimators, V is the variance.

For a given sampling method, let $F(c)$ and $V(c)$ denote the sums of F and V respectively over all flow records of key c.

method	F	V
threshold flow	$p_z(b)$	$b(z-b)^+$
uniform flow	$1/N$	$b^2(N-1)$
uniform packet	k/N	$b^2(N-1)/k$
unif. pkt→ flow (upper)	$f(k,t,N,T)$	$b^2(N-1)/k$
unif. pkt→ flow (lower)	$1-(1-1/N)^k$	$b^2(N-1)/k$
sample/hold	$1-(1-p)^b$	$V_{\rm sh}$
sample/hold (modified)	$1-(1-p)^b$	$V_{\rm sh:mod}$

TABLE I
EXPECTED NUMBER OF FLOWS F AND BYTES ESTIMATOR MEAN SQUARE ERROR V FROM SAMPLING A SINGLE FLOW OF DURATION t, k PACKETS AND b BYTES, WITH FLOW TIMEOUT T. THE FUNCTION f IS DEFINED IN (19). $V_{\rm lb}$ AND $V_{\rm nu}$ ARE DEFINED IN SECTION III-B.5

Thus $F(c)$ is the total expected number of measurements in that key, while $V(c)$ is the total squared error due to sampling of the estimated total key c bytes. We describe F and V for the different sampling methods; they are summarized in Table I.

1) Threshold flow sampling: Threshold sampling of flows records formed from the full packet stream. With sampling threshold z, $F = p_z(b)$ and $V = b(z-b)^+$, the expected variance from (15).

2) Uniform flow sampling: 1 in N sampling of flow records formed from the full packet stream. The expected number of flows is $F = 1/N$; the estimated bytes take the values Nb with probability $1/N$ and 0 otherwise, hence $V = b^2(N-1)$.

3) Uniform packet sampling: The atomic measurements are packets drawn by 1 in N sampling form the full packet stream; no flows records are formed. A flow of k packets gives rise to k/N atomic measurements on average. Let $b_1,\ldots,b_k$ be the sizes of the packets in the flow. Then estimator variance is $(N-1)\sum_{i=1}^k b_i^2 \geq (N-1)b^2/k$. We use this lower bound on the estimator variance, equivalent to simplifying assumption that all packets have the same size b/k.

4) Uniform packet sampling → flows: Packets are sampled uniformly, and flow records formed from the sampled packets. The flow records are not further sampled: estimator variance is the same as for uniform packet sampling.

Assume that packets are independently sampled; the probability that at least one packet of a flow of k packet is sampled $1-(1-1/N)^k$. This provides a lower bound on the number of resulting flows. A larger number of flows may result if the separation of sampled packets exceeds the flow timeout T. The mean number of sampled packets is k/N. If this exceeds 1, the worst case is that sampled packets are equally spaced at the mean spacing $tN/(k-1) > T$, the flow timeout. In this case, there will be one flow per packet. Thus $F = f(k,t;N,T)$, where

$$f(k,t;N,T) = \begin{cases} 1 & \text{if } Nt \leq (k-1)T \text{ and } N < k \\ k/N & \text{otherwise} \end{cases} \tag{19}$$

For comparisons we assume that T will be no less than the typical value $30s$ used for an unsampled packet stream; since $f(k,t;N,T)$ is nonincreasing in T, taking $T = 30$ gives an upper bound on the sample volumes.

5) Sample and hold: Packets are sampled at the router as follows [15]: a cache lookup is performed for each incoming packet, and if a match is found, the flow's statistics are updated as usual. (See also Section VIII-.4). If a match is not found, a new cache entry is created with probability $1-(1-p)^a$ when the packet has a bytes, for some fixed p. Thus the probability that a flow comprising b bytes is sampled is $F = 1-(1-p)^b$, independent of the manner in which the bytes of the flow are divided amongst its packets. The estimate byte size of the flow reported is then the number of bytes in the packets that were actually sampled, which is always a lower bound on the actual number b of bytes in the flow.

In can be shown that the MSE in estimating a flow of b unit size packets is

$$V_{\rm sh} = \frac{1-p}{p^2}\,(2-p-(1-p)^b(2-p-2pb)) \tag{20}$$

and that this is an upper bound on the MSE for general packets.

This estimator is negatively biased. In the case of unit packets, an unbiased estimator is obtained by adding the quantity $s_0 = (1-p)/p$ to the reported flow sizes. The resulting MSE for this modified estimator is

$$V_{\rm sh:mod} = V_{\rm sh} + s_0^2 g(b,p) \tag{21}$$

where

$$g(x,b) = (1-p)^b + 2bp(1-p)^{b-1} - 1 \tag{22}$$

It turns out that $g(b,p)$ can be positive or negative; reducing the bias may actually increase the MSE. It can be proved that, for large b, this happens once p is less than about $1.26/b$. The qualitative reason is that the correction to the bias introduces a larger error into the estimates of the size of small flows.

6) Summary statistics and comparison: To form a compact measure of estimator sampling variability we average the per-key relative standard deviations $\sqrt{V(c)}/X(c)$ with the key byte totals $X(c)$ as weights, yielding the weighted mean relative standard deviation:

$$S = \frac{\sum_c \sqrt{V(c)}}{\sum_c X(c)} \tag{23}$$

This attaches greater importance to a given relative standard error in estimating a larger byte total. Let K denote the total number of packets represented in the flows, i.e., the sum of k over all flow records. The effective sampling period $K/\sum_c F(c)$ is the ratio of the total number K of packets represented in the original flow records, to the number of measurements $\sum_c F(c)$. As an example, for uniform 1 in N packet sampling, the effective sampling period is N.

Figure 5 displays S as a function of the effective sampling period for the various sampling methods; the points on a given curve are generated by varying the underlying sampling rates. The sampling methods that depend on flow size (threshold sampling and sample and hold) provide the best accuracy by at least one order of magnitude over at least five orders of magnitude of the sampling period, the accuracy being more pronounced at lower sampling periods. Amongst these methods there are some differences in accuracy. For effective packet sampling rates below around 500, threshold sampling is most accurate, while for larger rates, it has the almost the same

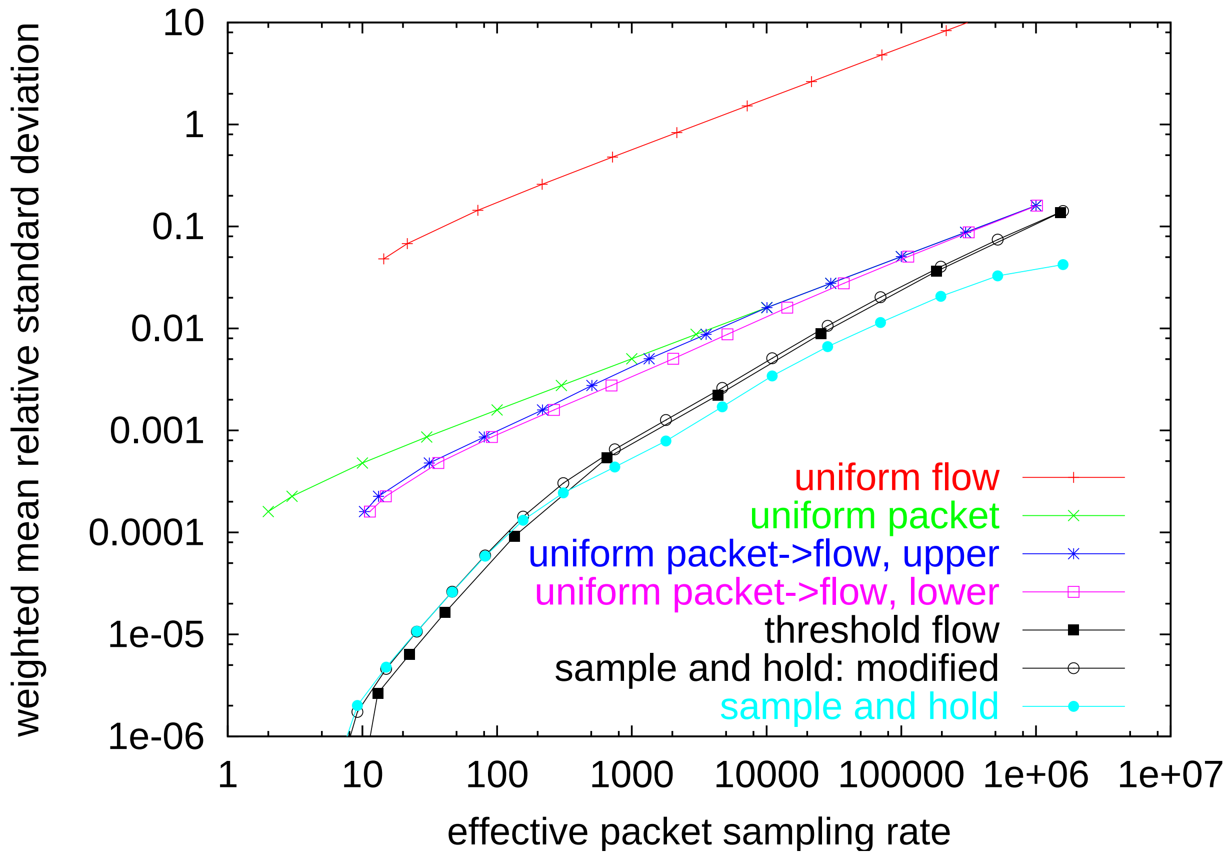

Fig. 5. COMPARISON OF SAMPLING METHODS: weighted mean relative standard deviation vs. effective packet sampling period

accuracy as the modified version of sample and hold, while the modified version of sample and hold does best. Reasons for this were discussed in Section III-B.5. The accuracy of all methods (except uniform flow sampling) tightens for very large sampling periods. We believe this happens because only the same very long flows are sampled by each method.

Threshold sampling combines the compression advantages of forming flow records, with low estimator variance due to size dependent sampling. We conclude that even when a certain degree of packet sampling is mandated by constraints on resources for packet processing in the router, the formation and size sampled flow statistics allows more accurate usage estimation than further packet sampling.

C. *Quasi-random implementation of threshold sampling*

Independent pseudorandom selection of flows according to a given sampling function $p(x)$ can be performed using a well-known random number generator; see e.g. [24]. However, the computational costs of effective generators may prohibit their use in some applications. In this section we describe a simple implementation of the sampling strategy described in Section II, that was used for all experiments reported in this paper. The implementation nearly as efficient as the 1 in N sampling strategy. The pseudo code in Figure 6 describes our implementation of a function that determines whether to sample the data. We assume that the input record has an integer field x that contains the size to be sampled over, e.g., bytes in a flow record. Furthermore the record has a real field `samplingFactor` that indicates the multiplicative renormalizing factor $1/p_z(x)$ that will be applied to the data if sampled. The function returns 1 if the data is to be sampled, 0 if not, i.e., the function returns the indicator variable w_i for a given flow i.

We chose to return the normalizing factor rather than perform the normalization directly, since it may be desired to apply the factor to other sizes. For example, in the context of flow sampling, we could apply the factor obtained for sampling byte count to packet counts as well. This results in an unbiased estimator of the total packet count, although it does not in general have minimal variance. A more sophisticated approach to sampling joint variable can be based on our work on multidimensional sampling described in Section II-E.

The function keeps track on the sum of sizes of small data record (i.e., `data.x` $< z$) modulo z using a static integer `count`. If `count` is uniformly distributed on the integers between 0 and $z-1$, then if the size of the record is greater than z then it will always be sampled, while if the size is less than z then the record is sampled with probability x/z. Thus under the uniform distribution assumption, the heuristic implements the optimal strategy.

Theorem 3: The algorithm described in Figure 6 implements the threshold sampling strategy, under the assumption

```
int sampleData (DataType data, int z) {
  static int count = 0;
  if (data.x > z)
    data.samplingFactor = 1.0;
  else {
    count += data.x;
    if (count < z)
      return 0; // do not sample this data
    else {
      data.samplingFactor = ((double)z)/size;
      count = count - z;
    }
  }
  return 1; // sample this data
}
```

Fig. 6. Quasi-random implementation of the data sampling algorithm.

that the variable `count` is a uniformly distributed integer between 0 and $z-1$.

A simple condition for the assumption on `count` to hold is that (a) the subsequence of flow sizes $\{x_i : x_i < z\}$ are i.i.d. random variables, and (b) the support of the distribution of the x_i generates $\{0, 1, \ldots, z-1\}$ under addition modulo z. It then follows from the theory of random walks that the distribution of `count` converges to the uniform distribution. In fact, the same result holds under weaker dependence conditions of the x_i, e.g., that they form an ergodic Markov chain. See [21] for related analysis.

Although the marginal sampling probabilities conform to $p_z(x)$, sampling of different flows will not in general be independent using the algorithm of Figure 6. This is for two reasons. First, sampling, rather than being independent, more closely resembles periodic sampling at the appropriate size dependent rate. To see this, note that a stream of flows of uniform size x will be sampled roughly periodically, with average period z/x. Second, the driving sequence of underlying flow sizes may not be independent. However, we do not believe that this will be a significant issue for estimation. We found that correlation between the sampling indicators for successive flows fell rapidly (in lag) to near zero. Furthermore, flow records of a given key are interspersed with flows of other keys, further reducing correlations of the sampling indicators of flows of a given key.

Finally, we note that dependence between flow sizes is irrelevant when sampling with a good pseudorandom number generator; sampling decisions are independent, or at least as independent as the sequence of numbers produced by pseudorandom generator.

IV. Dynamic Control of Mean Sample Volume

In Section II we showed that the cost function $C_z(p) = \mathsf{Var}\,\widehat{X} + z^2\mathsf{E}\widehat{N}$ was minimized by taking $p(x) = p_z(x) = \min\{1, x/z\}$ as the sampling function. In the measurement applications that motivate us, we want to be able to directly control the number of samples taken, in order that their volume does not exceed the capacity available for processing, transmission and storage. Clearly the sample volume depends on z, and so the question is: how should z be chosen?

In a dynamic context, the volume of objects presented for sampling will generally vary with time. Thus, in order to be useful, a mechanism to control the number of samples must be able to adapt to temporal variations in the rate at which objects are offered up for sampling. This is already an issue for uniform sampling: it may be necessary to adjust the sampling period N, both between devices and at different times in a single device, in order to control the sampled volumes. In threshold sampling, the threshold z controls the sampling volume. At first sight the task appears more complex than for uniform sampling, since the threshold sampling volume depends on both the volume of offered flows, and their sizes. However, we have devised an algorithm to adapt z to meet a given volume constraint which requires knowledge only of the target and current sample volumes.

Consider the case the target mean sample volume M is less than n, the total number of objects from which to sample. $\widehat{N}_z = \sum_i w_i$ is the total number of samples obtained using the sampling function p_z. Now the expected number of samples $N_z = \mathsf{E}\widehat{N}_z = \sum_i p_z(x_i)$ is clearly a non-increasing function of z, and indeed we show below that there exists a unique z^* for which $N_{z^*} = M$. A direct approach to finding z^* is to construct an algorithm to find the root. In Section VII we shall provide a algorithm that does just this by recursion on the set of sizes $\{x_i\}$. However, this approach is not suited to all sampling applications. For example, storage or processing cycles may not be available to perform the recursion.

Instead, we first present a simple iterative scheme to determine z^* that works by repeated application to the set of sizes $\{x_i\}$. The advantage of this approach over a recursive one will become most evident when we come on to consider the application to dynamically changing sets $\{x_i\}$ in Section IV. Whereas recursion must complete on a given set $\{x_i\}$, the iterative method allows us to replace the set of sizes $\{x_i\}$ with new data after each stage of the iteration, without requiring to store the sizes.

Theorem 4: Assume that $x_i > 0$ for all $i = 1, \ldots n$, that $n > M$, and that $z_1 > 0$. Define $g(z) = zN_z/M$ and set $z_{k+1} = g(z_k)$, $k \in \mathbb{N}$.

(i) $g(z)$ is concave and the equation $g(z) = z$ has a unique positive solution z^*.
(ii) $k \mapsto z_k$ is monotonic, and $\lim_{k\to\infty} z_k = z^*$.
(iii) $k \mapsto N_{z_k}$ is monotonic, and $\lim_{k\to\infty} N_{z_k} = M$.

Proof: $g(z)$ has the form $\sum_i \min\{x_i, z\}/M$, from which the following observations can be made. First, $g(z) = zn/M > 1$ for $z \leq x_{\min} = \min_{i=1}^n x_i$. Second, as a sum of concave functions, g is concave. Third, $g(z) = \sum_i x_i/M$ for $z \geq x_{\max} = \max_{i=1}^n x_i$.

From these properties, $g(z) = z$ has a unique solution $z^* > 0$. Furthermore $g(z) > z$ (resp. $g(z) < z$) for $z < z^*$ (resp. $z > z^*$), and hence $\{z_k\}$ is a strictly monotonic sequence bounded above by $\max\{z^*, z_1\}$. Therefore $\lim_{k\to\infty} z_k = z^*$ since g is bounded and continuous on $(0, \max\{z^*, z_1\})$.

N_z is clearly continuous and non-increasing in z and hence N_{z_k} is monotonic in k and converges to N_{z^*} as $k \to \infty$, converging from above if $z_1 < z^*$ (i.e. if $N_{z_1} > M$), and converging from below if $z_1 > z^*$ (i.e. if $N_{z_1} < M$). ∎

We illustrate the form of g and convergence of the sequence $\{z_n\}$ in Figure 7. In some cases, the fixed point z^* can be achieved in finitely many iterations. Suppose $\sum_i x_i/M > x_{\max} = \max_i x_i$. Then $z^* = \sum_i x_i/M$ and $g(z) = z^*$ for $z \geq x_{\max}$. Thus once z_n falls in the interval $[x_{\max}, \infty)$, the next iteration yields $g(z_n) = z^*$; see Figure 8. We note that by Theorem 4 $\{z_k\}$ and $\{N_{z_k}\}$ are monotonic and so z_k will never overshoot the fixed point z^*.

A. Rates of convergence

Having established convergence of z_k to the unique fixed point z^*, we now investigate how quickly convergence occurs. Our first result show that the number of iterations required to bring N_z to within a given factor of the target M is controlled uniformly in terms of the initial and target thresholds z_1 and z^*.

Theorem 5: Starting with $z = z_1$, it requires no more than $1+|\log_{1+\varepsilon}(z^*/z_1)|$ iterations to get N_z within a factor $(1+\varepsilon)$ of M, i.e, . $M/(1+\varepsilon) < N_z < M(1+\varepsilon)$.

Proof: We prove for $N_{z_1} > M$; the case $N_{z_1} < M$ is similar. Let $n = \max\{k \mid N_{z_k} > M(1+\varepsilon)\}$. Then

$$\frac{z^*}{z_1} > \frac{z_{n+1}}{z_1} = \prod_{j=1}^{n} \frac{z_{j+1}}{z_j} = \prod_{j=1}^{n} \frac{N_{z_j}}{M} > (1+\varepsilon)^n \tag{24}$$

Thus $n < \log_{1+\varepsilon}(z^*/z_1)$ and the required number of iterations is no more than $n+1$. ■

Note that if ε is small, $|\log_{1+\varepsilon}(z^*/z_1)| \approx |\log(z^*/z_1)|/\varepsilon$. In a neighborhood of z^*, the rate of convergence of z_k is governed by the derivative of g. Let

$$X_z = \sum_{i:x_i \leq z} x_i \quad \text{and} \quad R_z = \#\{i : x_i > z\}. \tag{25}$$

Observe that $N_z = X_z/z + R_z$ and hence that $g(z) = X_z/M + zR_z/M$. g is piecewise linear with right derivative R_z/M and left derivative R_z^-/M where $R_z^- = R_z + \#\{i : x_i = z\}$. We now express convergence of the absolute difference of N_z and M in terms of these derivatives.

Theorem 6: (i) Adopt the assumptions of Theorem 4. $|z_k - z^*| < \rho^k |z_1 - z^*|$ where ρ depends on z_1 as follows. If $z_1 > z^*$, take $\rho = R_{z^*}/M < 1$. Otherwise, for sufficiently small $\varepsilon > 0$, and sufficiently large $z_1 < z^*$, we can take $\rho = R_{z^*}^- + \varepsilon < 1$. Thus, subject to these conditions, the number of iterations required to bring z_k within a distance δ of z^* is no greater than $|\log(\delta/|z_1 - z^*|)/\log\rho\,|$.

(ii) $|N_z - N_{z'}| \leq (z - z') \cdot n/M$ for all $z, z' \geq 0$. Hence $|N_{z_k} - M| < \rho^k (n/M)|z_1 - z^*|$.

Proof: (i) Suppose $z_1 > z^*$. From from the concavity of g, $z_{k+1} = g(z_k) \leq g(z^*) + (z_k - z^*)R_{z^*}/M$. Now $R_{z^*}/M = 1 - X_{z^*}/(z^* M) < 1$ and so the bound on the required number of iterations is trivial. Otherwise, for $z_1 < z^*$, using concavity of g and since z_n is increasing, $z^* - z_1 \leq z^* + (z^* - z_1)R_{z_1}^-/M$. For sufficiently small $\varepsilon > 0$, $R_{z_1}^- \leq R_{z^*}^- + \varepsilon < M$.

(ii) Denote $z_- = \min\{z, z'\}$ and $z_+ = \max\{z, z'\}$. $N_{z'}$ and N_z have identical contributions from those x_i not in the interval $[z_-, z_+]$. Hence

$$|N_z - N_{z'}| = \sum_{i:z_- < x_i < z_+} (x_i - z_-)/M \leq |z - z'| \cdot n/M. \quad ■ \tag{26}$$

Note the rate ρ is uniform for $N_{z_1} < M$. We believe this is the more interesting case: when the population size n is unknown, one may conservatively choose a large initial z_1 in order that $N_{z_1} < M$.

Worst-case convergence of the absolute difference $N_z - M$ occurs when R_{z^*}/M is as close as possible to 1. Since $R_z \leq N_z < M$, the worst case possible case would entail $R_{z^*} = M - 1$. (One can construct such an example by letting each x_i take one of two sufficiently separated values, with $M - 1$ of the x_i taking the larger value). According to Theorem 6, the number of iterations k required for a given absolute difference $|N_{z_k} - M|$ is then $O(M)$. In the next section we show that a modification of the iteration allows exact convergence of N_{z_k} in *finitely* many steps, whose number grows no worse than $O(M)$.

B. Location of z^* in finitely many steps

We now show how a simple modification of the iterator g enables exact determination of z^* within finitely many iterations for any collection of sizes $\{x_i\}$. Let

$$\tilde{g}(z) = \begin{cases} g(z) & \text{if } R_z \geq M \\ z(N_z - R_z)/(M - R_z) & \text{if } R_z < M \end{cases} \tag{27}$$

The modification $\tilde{g}$ of g makes use of the subsidiary quantity R_z, the number of sizes that exceed a given threshold z. Since such sizes are sampled with probability 1 and $r_z(x) = \max\{x, z\}$, R_z can be determined from the stream of sampled renormalized sizes alone: it is the number of such objects that exceed z.

Since $\tilde{g}(z) = z$ iff $N_z = M$, $\tilde{g}$ shares with g the unique fixed point z^*. We say that iteration with $\tilde{g}$ terminates when successive values are identical. Since z^* is the unique fixed point of $\tilde{g}$, termination can happen only at z^*. Since $\tilde{g}(z) < g(z)$ for $z > z^*$, we expect convergence to be faster with $\tilde{g}$ than g for initial $z > z^*$. We also find that $\tilde{g}(z) \geq z^*$ for all z such that $R_z < M$, so that after at most one iteration we enter the regime of convergence to z^* from above.

Theorem 7: Starting with z such that $R_z < M$, iteration with $\tilde{g}$ terminates on z^* after at most $O(\min\{M, \log MX\})$ iterations.

Proof: First consider the case that $R_z \leq N_z < M$, and hence $z > z^*$. We show this implies that $\tilde{g}^n(z)$ is a decreasing sequence bounded below by z^*. Now $N_z < M = N_{z^*}$ implies $z' := \tilde{g}(z) < z$. Since $N_z = X_z/z + R_z$, z' satisfies

$$M = X_z/z' + R_z. \tag{28}$$

Hence

$$\begin{aligned} N_{z'} &= X_{z'}/z' + R_{z'} \\ &= X_z/z' + R_z + \sum_{i:z' < x_i \leq z} (1 - x_i/z') \\ &\leq X_z/z' + R_z = M, \end{aligned} \tag{29}$$

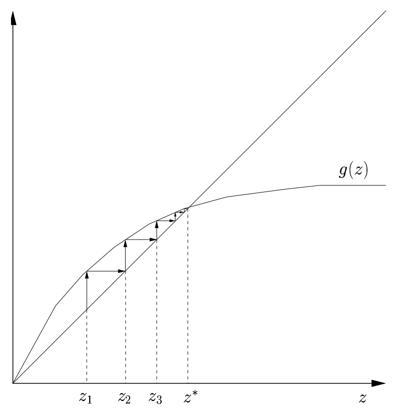

Fig. 7. ITERATION WITH g: with sequence $\{z_n\}$ converging to z^* from below.

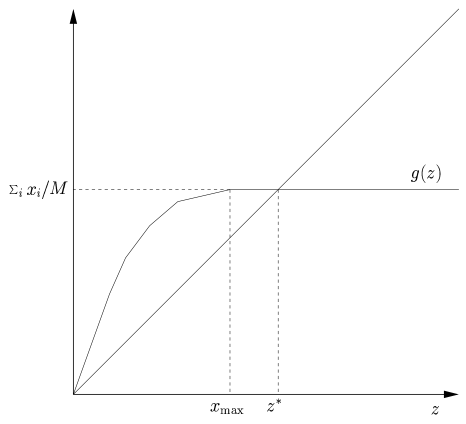

Fig. 8. ONE-STEP CONVERGENCE: $g(z) = z^*$ for $z \geq x_{\max}$ if $\sum_i x_i/M \geq x_{\max}$.

and so $z' \geq z^*$. If there are no attributes x_i in $(z', z]$, then $X_z = X_{z'}$, $R_z = R_{z'}$ and so $M = N_{z'}$ by (28) and the iteration terminates with $z' = z^*$. An immediate consequence is that each non-terminating iteration increases R_z by at least 1. Since $R_z \leq N_z < M$, there can be at most M such iterations.

Define $\alpha_z > 0$ such that $M = (1+\alpha_z)N_z$. We are going to show that in each iteration, either X_z or α_z is at least halved. Assume that $\beta := \min\{x_i, \min_{x_i \neq x_j}\{|x_i - x_j|\}\} > 0$. $\beta \geq 1$ if x_i takes positive integer values. Since $X \geq X_z \geq \beta$, there can be at most $O(\log X_z)$ iterations in which X_z is halved.

Suppose X_z is not halved, i.e., that $X_{z'} > X_z/2$. Now, $\alpha_z N_z = M - N_z = M - X_z/z - R_z = X_z(1/z' - 1/z)$. However, using (29) with z and z' interchanged, we find $\alpha_z N_z - \alpha_{z'} N_{z'} = N_{z'} - N_z \geq X_{z'}(1/z' - 1/z) \geq (X_z/2)(1/z' - 1/z) = \alpha_z N_z/2$. Hence $\alpha_{z'} N_z \leq \alpha_{z'} N_{z'} \leq \alpha_z N_z/2$, as desired.

Recall that an iteration that does not yield z^* must increase R_z. Hence after one such iteration we have $N_z \geq R_z > 1$, and so $\alpha_z = M/N_z - 1 \leq M - 1$. To finish the bound we need to show that α_z cannot get too small without the algorithm terminating. Here we exploit that if $\alpha_z < \beta/(2Mz)$, then $z - z' = z\alpha_z\beta N_z/(M - R_z) < \beta/(2(M - R_z)) < \beta/2$ due to the integer inequalities $R_z \geq R_{z_*} > M$. Hence, in one of the next two iterations, z does not cross an x_i, in which case R_z does not change on that iteration, which implies that the algorithm terminates. Thus $\alpha_z > \beta/(2Mz) > \beta/(2Mz_*)$, and there can be at most $O(\log M)$ iterations in which α is halved.

Next consider the case that initially $N_z > M > R_z$. Similarly to (29) one has for $z' = \tilde{g}(z)$ that

$$\begin{aligned} N_{z'} &= X_{z'}/z' + R_{z'} \\ &= X_z/z' + R_z + \sum_{i: z < x_i \leq z'} (x_i/z' - 1) \\ &\leq X_z/z' + R_z = M \end{aligned} \tag{30}$$

and hence $z' > z^*$. Thus after one iteration the problem reduces to the case $R_z \leq N_z < M$. ∎

V. PATHWISE DYNAMIC CONTROL OF SAMPLE VOLUMES

The strength of an iterative method is that it controls the sample volume $\widehat{N}$ in both the static case that the distribution of sizes of the original population $\{x_i\}$ is unknown, and—as we shall in this section—the dynamic case that the the population is changing. The latter ability is important for network measurement applications, where both the volume of offered flows and their sizes can change rapidly. As an example, volumes of short flows have been observed to increase rapidly during denial of service attacks.

The setup for our analysis is as follows. Consider a sequence of time windows labeled by $k \in \mathbb{N}$. In each window a set of sizes $\{x^{(k)}\} = \{x_i^{(k)} : i = 1, 2, \ldots n_k\}$ is available for sampling. Given a threshold z, then the total sample volume in window k is $\widehat{N}_z^{(k)} = \sum_{i=1}^{n_k} w_i^{(k)}$ where $w_i^{(k)}$ are independent random variables taking the value 1 with probability $p_z(x_i^{(k)})$ and 0 otherwise. The set of sampled renormalized sizes from window k is $S_z^{(k)} = \{\max\{x_i^{(k)}, z\} : w_i^{(k)} = 1\}$, and the estimate of the total size of $\{x^{(k)}\}$ is $\widehat{X}^{(k)} = \sum_{y \in S^{(k)}} y$.

Generally we want to use control schemes that require relatively sparse knowledge about the past. For example, we do *not* want to require knowledge of the full sets of original sizes $\{x^{(k)}\}$ from past windows, due to the storage requirements this would entail. The control schemes that we will use here are based on the iterative algorithms of Section IV. The value of z_{k+1} to use for sampling during window $k+1$ is a function of only the target sample volume M, the following quantities from the the immediately previous window: the threshold z_k, the sample volume $\widehat{N}_{z_k}^{(k)}$ and, optionally, the set of sampled renormalized sizes $S^{(k)}$. Let $R_z^{(k)} = \#\{y \in S_z^{(k)} : y > z\}$.

We first specify three candidate control algorithms based on the results of Section IV. The *Conservative Algorithm* is based on the basic iteration defined in Theorem 4. The *Aggressive Algorithm* uses the modification described in Section IV-B with aim of speeding up convergence when $\widehat{N} < M$. The *Root Finding Algorithm* uses a limited recursion to speed up convergence when $\widehat{N} > M$. This assumes the requisite storage and processing cycles are available. These algorithms can be supplemented by two additional control mechanisms.

Emergency Volume Control rapidly adjusts the threshold z if systematic variations in the offered load cause $\widehat{N}$ to significantly exceed the target M before the end of the window. *Variance Compensation* estimates the inherent variance of the sample volume, and adjusts the target volume M downwards accordingly.

A. Conservative algorithm

$$z_{k+1} = z_k \frac{\max\{\widehat{N}^{(k)}_{z_k}, 1\}}{M} \tag{31}$$

The maximum with 1 comes into effect when z is so large that no sizes end up being sampled, i.e., when $\widehat{N}_{z_k} = 0$. The effect is to drive z down and so make some sampling more likely at the next iteration.

B. Aggressive algorithm

$$z_{k+1} = \begin{cases} z_k \frac{\widehat{N}^{(k)}_{z_k}}{M} & \text{if } \widehat{N}^{(k)}_{z_k} \geq M \\ z_k \frac{\max\{\widehat{N}^{(k)}_{z_k} - \widehat{R}^{(k)}_{z_k}, 1\}}{M - \widehat{R}^{(k)}_{z_k}} & \text{if } M > \widehat{N}^{(k)}_{z_k} \geq 0 \end{cases} \tag{32}$$

Here we apply the form of the iteration (27) only when $M > \widehat{N}^{(k)}_{z_k}$, rather than under the weaker condition $M > R^{(k)}_{z_k}$. The reason for this that the jump from $z < z^*$ to $\tilde{g}(z) > z^*$ can lead to oscillatory behavior in the presence of size variability.

C. Root finding algorithm

The third candidate arises from the fact, established in Section II-G, that sampling and renormalizing a set of sizes $\Omega = \{x_i : i = 1, \ldots, n\}$ using threshold z, and then resampling the resulting set with threshold $z' > z$ is statistically equivalent to performing a single sampling operation with threshold z'.

Suppose for a given value z we obtain the sample volume $\widehat{N}_z > M$. Conditioning on the fixed set of sampled sampled values $S_z(\Omega)$, we can write the conditional expectation of the sample volume $\widehat{N}'_{z',z} = \sum_{x \in S_z(\Omega)} p_{z'}(x)$ under resampling from $S_z(\Omega)$ but using the threshold $z' > z$. We rewrite $\widehat{N}'_{z',z} = \sum_{x \in \Omega} w_x p_{z'}(r_z(x))$ where w_x are independent random variables taking the value 1 with probability $p_z(x)$ and 0 otherwise. From Theorem 2 it follows that $\mathsf{E}\widehat{N}'_{z',z} = N_{z'}$ when $z' \geq z$, i.e., $\widehat{N}'_{z',z}$ is an unbiased estimator of $N_{z'}$. Furthermore, for a given realization of the $\{w_i\}$, $z' \mapsto \widehat{N}'_{z',z}$ is non-increasing. Consequently it is relatively simple to determine the root $\widehat{z}^* > z$ of the equation $\widehat{N}'_{\widehat{z}^*,z} = M$. We denote this root by $\mathcal{Z}(S_z, M)$. An algorithm to determine the root is presented in Section VII below. The root finding algorithm is then:

$$z_{k+1} = \begin{cases} \mathcal{Z}(S^{(k)}_{z_k}, M) & \text{if } \widehat{N}^{(k)}_{z_k} > M \\ z_k \frac{\max\{\widehat{N}^{(k)}_{z_k}, 1\}}{M} & \text{if } M \geq \widehat{N}^{(k)}_{z_k} \geq 0 \end{cases} \tag{33}$$

We will also speak of combining the aggressive with the root finding approach, by which we mean using (32) when $M > \widehat{N}^{(k)}_{z_k} \geq 0$ and (33) when $\widehat{N}^{(k)}_{z_k} > M$.

D. Emergency volume control

If the arrival rate of objects to be sampled grows noticeably over a time scale shorter than the window width, $\widehat{N}$ may significantly exceed the target M. We propose additional control mechanisms that can be used to control the sample volume in emergency. The idea is that if a target sample volume is already exceeded before the end of a window, we should immediately change the threshold z. In this context, we can regard the windowing mechanism as a timeout that takes effect if $\widehat{N}$ has not exceeded M by the end of the window. We present three variants on this theme, tighter control being possible when more information on the original data stream is available.

1) Emergency control using timing information: If $\widehat{N}$ already exceeds M at time t from a start of a window of length τ, we immediately replace z by $z\tau/t$. This approach derives from (31) since the target $\widehat{N}$ over the interval $[0, t)$ is Mt/τ. If we have control over the window boundaries, then we may just start a new window at that time. Otherwise, from time t we reaccumulate the sample count $\widehat{N}$ from zero, and the test and remedy are repeated as needed for the remainder of the window.

2) Emergency control using population size: If timing information is not directly available, but the number n in the original population in the window is available, we can repeat the above procedure using i/n as a proxy for t/τ, where i is the sequence number of size x_i in the window. (Note this control violates our aim to sample without knowledge of the original population size, and so may not be applicable in all circumstances).

3) Emergency control with tolerance parameter: If neither timing nor size information are available, we select an additional tolerance parameter $\gamma > 1$. If $\widehat{N}$ exceeds γM during a window, we immediately replace z by γz, again following (31).

E. Compensating for sampling variability

Each of the algorithms above may be modified to be more conservative by adjusting the target M downwards to take account of the sampling variability. From Section II-F we know that $\widehat{V}_{N_z} = \sum_{i=1}^n w_i(1 - x_i/z)^+$ is an unbiased estimator of $\mathsf{Var}\,\widehat{N}_z$. Therefore, for a given multiplier $\sigma > 0$, we can elect to replace M by $M' = M - \sigma\sqrt{V_{N_{z_k}}}$ in (31), (32) and (33) above. The effect is to guard against statistical fluctuations of $\widehat{N}$–due to sampling alone–of up to σ standard deviations above the mean.

A simple upper bound on the sampling variance is obtained through

$$\mathsf{Var}\,\widehat{N}_z = \sum_{i=1}^n p_z(x_i)(1 - p_z(x_i)) \leq \sum_{i=1}^n p_z(x_i) = \mathsf{E}\widehat{N}_z \tag{34}$$

Thus in aiming for a target M we expect a relative error on $\widehat{N}$ of about $1/\sqrt{M}$. This leads to a simpler estimate for variance compensation: in order to guard against statistical fluctuations of up to s standard deviations from a target M, one should use the compensated target $M_s = M - s\sqrt{M}$. Although we do not

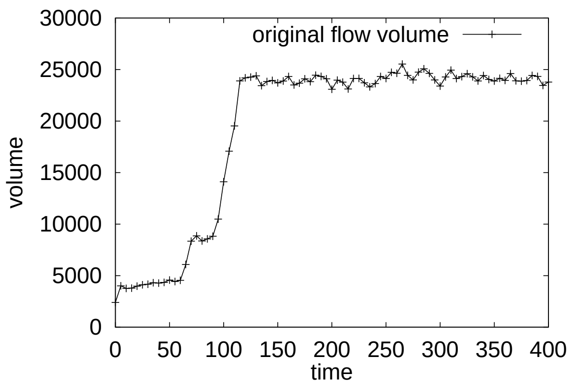

Fig. 9. ORIGINAL FLOW VOLUMES: in trace used for studies, over 5 second windows. Observe volume increase during period to time 115s.

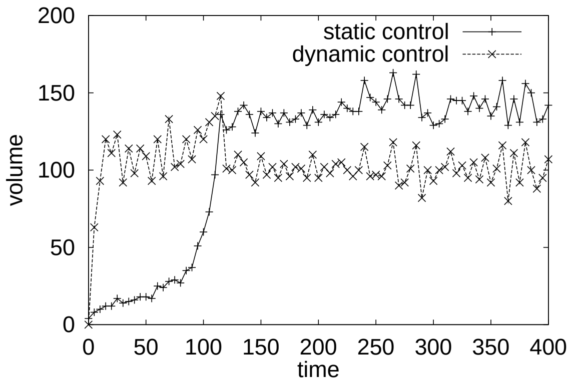

Fig. 10. STATIC AND DYNAMICALLY CONTROLLED SAMPLED FLOW VOLUMES: to obtain same average sampled flow volumes over 400 second period.

detail it here, we remark that a similar approach can be used to mitigate the effects of high uncertainty surrounding small sample volumes $\widehat{N}$ near zero. This leads to a modification of the "take the maximum with 1" approach used in (31), (32) and (33), instead replacing $\widehat{N}_z$ with $(s/2 + ((s/2)^2 + \widehat{N}_z)^{1/2})^2$ in order to guard against small unreliable values of $\widehat{N}_z$.

F. On sampling variability and timescales of variation

Suppose we wish to limit report volumes to a rate ρ per unit time. Over a window of width τ, our target volume is $M = \rho\tau$. A target typical relative error of ε in the report rate requires $\varepsilon = \sqrt{M}$. Eliminating M between these two relations, we find that for a report rate ρ with target relative error ε a window of width $\tau = 1/(\rho\varepsilon^2)$ is required. Emergency volume control guards against systematic variations in the original flow stream on time scales shorter than τ. If only the weakest form of emergency control is available, i.e. using a tolerance parameter γ, we select $\gamma = 1 + \varepsilon$.

G. Computational issues

We have seen in Section III-C that quasi-random sampling of each size may be performed with hardly more complexity than the simple deterministic 1 in N sampling strategy.

We now address the complexity of the subsequent calculations with sample values for the three algorithms described above. Both the conservative and aggressive algorithms keep only accumulated quantities $\widehat{N}$ (and $\widehat{R}$ for the aggressive algorithm), and require only $O(1)$ operations to iterate z from these quantities.

Although we may expect faster convergence with root finding when $\widehat{N} > M$, the trade-off is in the increased storage and computation requirements with $\mathcal{Z}$. Root finding from m quantities requires $O(m)$ storage and $O(m)$ operations to iterate z. In the application described in Section V-C we thus have $m \approx M$ when control to (near) volume M is effective. On the other hand, root finding from the original flow stream may be prohibitive in both storage and computational requirements, since m is then the number n of original flows present in a given time window.

VI. EXPERIMENTAL STUDIES OF DYNAMIC CONTROL OF SAMPLE VOLUMES

The previous section define a set of adaptive mechanisms to control the rate at which samples are produced. This section gives a trace-based investigation into their performance. In these experiments we focused on sampling streams of flow reports from single router interfaces. This gives insight into the behavior of the threshold sampling algorithm were it to be applied to reduce the volume of transmission of flows from the interface to the collection point. Most of these studies reported here focus on 400 seconds worth of data from a single network trace whose original flow volumes over 5 second windows is shown in Figure 9. We selected this trace because it displays a systematic increase in the original volume over the period between 80s and 100s. This provides a good opportunity to evaluate the performance of the algorithms under dynamic conditions. The empirical distributions of the byte and packet sizes of all flows in the trace are shown in Figure 11. Note the approximate linearity of the tail probabilities on a log-log scale; the slope is a little larger than 1, indicative of a heavy tailed distribution. We evaluate the dynamic algorithms along the following dimensions: (i) the benefit of dynamic choice of z versus the best average choice; (ii) the speed of convergence after a perturbation in the offered load; (iii) the ability of variance compensation and emergency control to limit the effects on the sample volume of rapid growth in the offered load; (iv) accuracy in estimating the total byte volume represented in the original set of flow records.

A. Dynamic vs. static control

To show the benefit of dynamic over static sampling we compare the sample volumes using the conservative algorithm at target volume $M = 100$ per 5 second window, with static control at a fixed threshold z chosen so as to yield the same average sample rate over 400 seconds. The two trajectories of the sample volume are shown in Figure 10. Unsurprisingly, the volume under static control follows closely the original volume displayed in Figure 9, leveling off at around 150 samples per window after the original volume increase at about 115s. On the other hand, the volume of samples obtained with dynamic

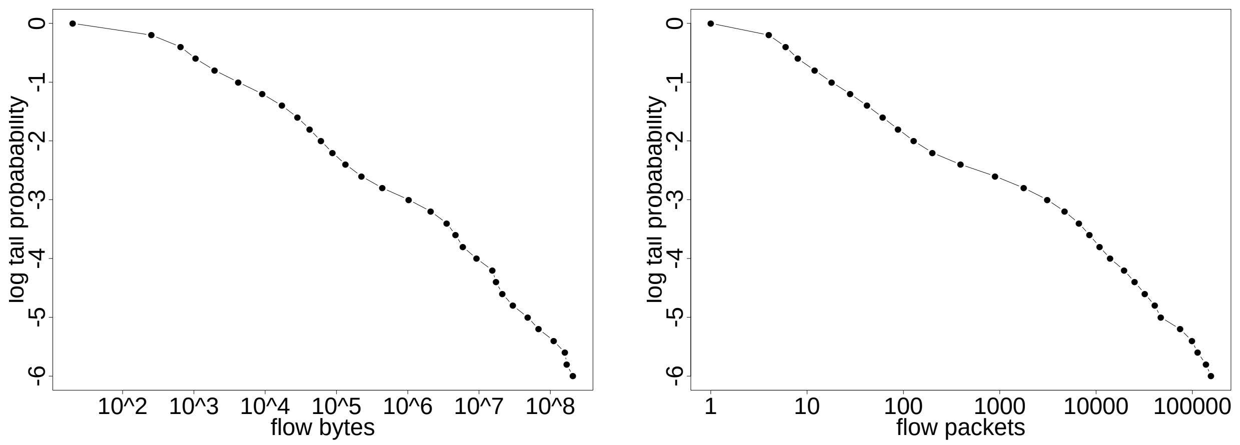

Fig. 11. EMPIRICAL DISTRIBUTION OF FLOW BYTE AND PACKET SIZES: approximate linearity of tail probabilities on log-log scale with slope around 1 indicative of heavy tailed distribution.

control increases only slowly during the period up to time 115s, falling back slightly to around 100 after the plateau in the original volume is reached. Clearly, the ability of the dynamic algorithm to adapt is better when the ratio of the time scale of increase to the sampling window duration is smaller. In the example, original volume increases by about a factor 5 over the 10 windows between 65s and 115s. This leads to some noticeable increase of report volume over the $M = 100$ towards the end of this period. The initial transient in the sample volume is due to a small initial value of the threshold z.

B. Relative performance of the dynamic algorithms

The two variants on the conservative algorithm defined in Section V, namely the aggressive and the root finding approach, were motivated by the desire to speed up convergence of $\widehat{N}$ to the target M. In order to assess the extent to which these variants met these aims, we modified the experiments to reproduce the effects of transients in the arrivals of flow records. We did this by artificially perturbing the calculated value of the threshold z, every tenth window, alternately upwards or downwards by an independent random factor that uniformly distributed between 1 and 200. The effect of these perturbations is to perturb the sampling rate downwards or upwards, resulting in a dip or spike in $\widehat{N}$. The perturbed value of z is a legacy for subsequent estimation: we wish to determine how quickly its effects are forgotten in the subsequent evolution. For these experiments the window width was 5 seconds, with a sample volume target of $M = 100$ out of an average unsampled stream of 19,980 flows per window.

We compare the responses of the various algorithms to the perturbations in Figure 12. The left plot shows the typical responses of the aggressive and conservative algorithms. Recall the aggressive algorithm aims to speed convergence when $\widehat{N} < M$. Observe that after the downward spike, aggressive algorithm indeed recovers more quickly.

The right-hand plot compares the root finding and conservative algorithm. Root finding aims to improve the convergence rate when $\widehat{N} > M$. Observe that after the upward spike, root finding returns $\widehat{N}$ to near the target value M in only one step; by comparison the conservative algorithm decays more gradually, needing typically 3 steps to return close to the target. Summarizing, the quickest recovery is obtained by combining the aggressive algorithm (to speed up recovery of $\widehat{N}$ from below) with root finding (to speed up recovery of $\widehat{N}$ from above).

C. The benefits of variance compensation and emergency control

Variance compensation was proposed in Section V-E as a means to mitigate the effects of the inherent variability due to random sampling. In Figure 13 we give an example on the effects of compensation and emergency control. The target volume is $M = 100$ flows per 5 second window. The upper trace (no compensation or control) exhibits a rise to $\widehat{N} = 147$ at time 115s in response to the rapid increase in the original volume evident in Figure 9. Thereafter, the original volume reaches a plateau, and the sampled volume exhibits fluctuations of order $\sqrt{M} = 10$ around the target M.

The effect of compensation at $s = 1$ is to lower the sample volume by roughly $\sqrt{M} = 10$. By adding emergency volume control (based on flow arrival times) to a level of $M = 100$, the peak sample volume at time 115s was reduced by a further amount 27 to 110. The lower trace in Figure 13 shows the sample volume obtained by augmenting the conservative algorithm with both variance compensation (at $s = 1$) and emergency control. In further experiments, we found the proportion of windows in the plateau region for which the target $M = 100$ is exceeded is 44% when $s = 0$, 10% when $s = 1$ and 2.6% when $s = 2$.

D. Summary and comparison

Emergency control and variance compensation are simple and effective means of guarding against systematic variation in the offered load and variability inherent due to sampling.

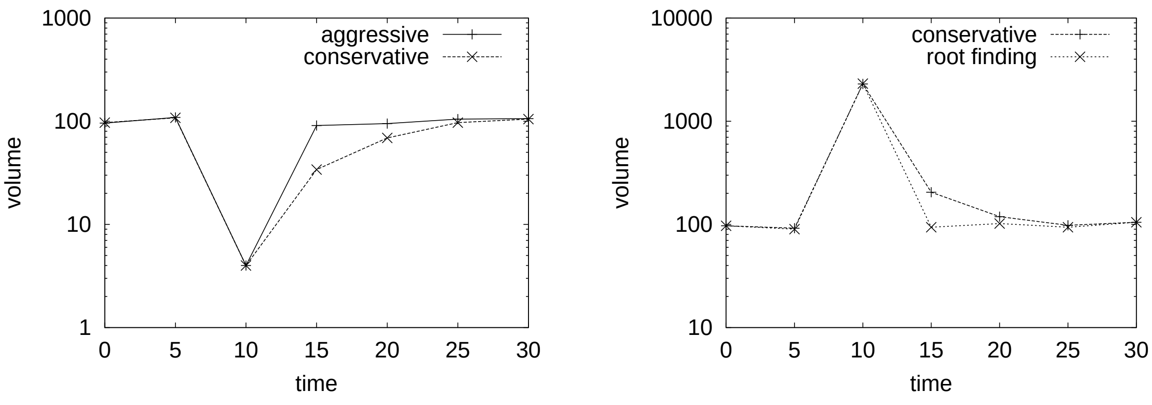

Fig. 12. RELATIVE RECOVERY RATES OF DYNAMIC SAMPLING ALGORITHMS UNDER ARTIFICIAL PERTURBATION: Left: conservative vs. aggressive. Right: conservative vs. root finding.

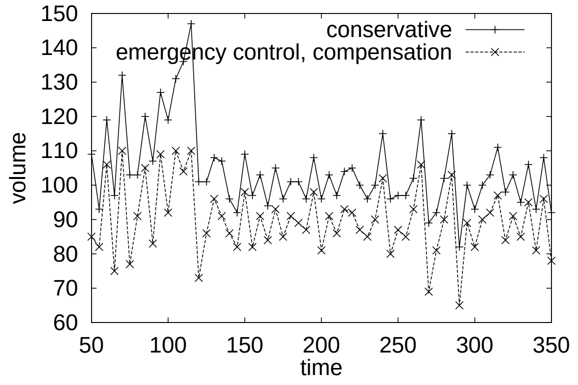

Fig. 13. VARIANCE COMPENSATION AND EMERGENCY CONTROL: Decrease in sample volume obtained by applying compensation for 1 standard deviation and emergency control. Relative to conservative algorithm, emergency control reduces impact of rapid increase in offered load up to time 115s that is evident from Figure 9.

The root finding algorithm gives some speed up in downward convergence after $\widehat{N}$ has exceeded its target M, but this requires additional storage and computational resources that may not be available in all cases. On the other hand, the aggressive algorithm speeds up convergence of $\widehat{N}$ from below M, with hardly any additional resource usage. For general application, we therefore favor the conservative or aggressive algorithms, combined with variance compensation and some form of emergency volume control.

VII. ROOT FINDING: ALGORITHM AND SAMPLING STATISTICS

In this section we detail an efficient root finding algorithm, and investigate the effect on root finding accuracy of omitting some sizes x_i, e.g., in order to reduce resource usage.

A. Implementation of root finding algorithm

In this section we consider a class of root finding problems arising from the requirements for a root finding algorithm in Section V-C. First, we consider how to solve for z the equation $N_z = M$. For $\circ =<,>,\leq,\geq,=$, define

$$X_{\circ z} = \{x \in X | x \circ z\}$$

Suppose we are given a fixed set of sizes $X = \{x_1, ... x_n\}$ and a target M. Let

$$N_z(X) = (\sum X_{\leq z})/z + |X_{>z}|$$

That is, N_z is our expected number of samples from X with threshold z. Our goal is to find $z^* = \mathcal{Z}(X, M)$ such that $N_{z^*}(X) = M$. We note that $N_z(X) = |X|$ for $z \leq \min X$, and that $N_z(X)$ is strictly decreasing for $z \geq \min X$. Thus, z^* is uniquely defined assuming $M < |X|$.

Our basic idea is to pick some value z and compare N_z with M. If $N_z = M$ we are done. If $N_z < M$, we know $z > z^*$ and if $N_z > M$, we know $z < z^*$. If $z > z^*$, we wish to recurse on $X_{<x}$ and if $z < z^*$, we wish to recurse on $X_{>x}$. To define the recursion, however, we define

$$N_z(X, B) = B/z + N_z(X)$$

Also, define $z^* = \mathcal{Z}(X, B, M)$ such that $N_{z^*}(X, B) = M$. Then $N_z(X, 0) = N_z(X)$ and $\mathcal{Z}(X, 0, M) = \mathcal{Z}(X, M)$. Note that if $B > 0$, $N_z(X, B)$ is strictly decreasing for all $z > 0$, and hence z^* is always unique. This leads to the following *uniqueness assumption* for $\mathcal{Z}(X, B, M)$: either $B > 0$ or $M < |X|$.

Lemma 1: If $N_z(X, B) < M$,

$$\mathcal{Z}(X, B, M) = \mathcal{Z}(X_{<z}, B, M - |X_{\geq z}|)$$

Proof: If $N_z(X, B) < M$, $z^* = \mathcal{Z}(X, B, M) < z$. For any z', $N_{z'}(X, B) = N_{z'}(X_{<z}, B) + N_{z'}(X_{\geq z})$, and for $z' < z$, $N_{z'}(X_{\geq z}) = |X_{\geq z}|$. Hence

$$N_{z^*}(X_{<z}, B) = N_{z^*}(X, B) - |X_{\geq z}| = M - |X_{\geq z}|. \quad (35)$$

If All we need to argue now is that z^* is the unique solution to (35). If $B > 0$ we are done. Otherwise, we have $M < |X|$, implying $M - |X_{\geq z}| < |X_{<z}|$. ∎

Lemma 2: If $N_z(X, B) > M$,

$$\mathcal{Z}(X, B, M) = \mathcal{Z}(X_{>z}, B + \sum X_{x \leq z}, M)$$

Proof: If $N_z(X,B) > M$, $z^* = \mathcal{Z}(X,B,M) > z$. For any z', $N_{z'}(X,B) = N_{z'}(X_{\leq z}, B) + N_{z'}(X_{>z})$, and for $z' > z$, $N_{z'}(X_{\leq z}, B) = (B + \sum X_{\leq z})/z'$. Hence

$$N_{z^*}(X_{>z}, B + \sum X_{\leq z}) = N_{z^*}(X,B) = M \tag{36}$$

All we need to argue now is that z^* is the unique solution to (36). If $X_{\leq z} = \emptyset$, there is nothing to prove. Otherwise, $\sum X_{\leq z} > 0$, in which case (36) always has a unique solution. ■

The above lemmas immediately imply that the following recursive algorithm correctly determines $\mathcal{Z}(X,B,M)$ when $B > 0$ or $M < |X|$.

Algorithm $\mathcal{Z}(X,B,M)$: Assume that $B > 0$ or $M < |X|$.

```
1) If $X = \emptyset$, return $B/M$.
2) Pick a random $z \in X$,
   a) If $N_z(X,B) = M$, return $z$.
   b) If $N_z(X,B) < M$, return $\mathcal{Z}(X_{<z}, B, M - |X_{\geq z}|)$.
   c) If $N_z(X,B) > M$, return $\mathcal{Z}(X_{>z}, B + \sum X_{\leq z}, M)$.
```

Each recursive step involves $O(|X|)$ operations. The point in picking z randomly from X is that, in each recursive step, we expect to get rid of a constant fraction of the elements in X, and so the total expected running time is $O(|X|)$. For a more precise analysis, we note that the algorithm behaves as standard randomized selection after reaching a neighbor of z^* in X. A formal analysis of the expected linear time for such algorithms is found in [8, pp. 187–189]. Concerning step 2, if the items in X are not ordered in relation to their sizes, we can just pick z as the first element in X. With this simplification, an iterative C-code version of the algorithm is presented in Figure 14.

```
double Z(double *X, int n, double B, double M) {
   int i,j,n1,n2;
   double B,B1,z;

   while (n>0) {
      z=X[0];
      n1=0;
      B1=0.0;
      for (i=0; i<n; i++) {
         if (X[i]>z) n1++;
         else B1+=X[i];
      }
      j=0;
      n2=n;
      if (((B+B1)/z) < (M-n1)) {
        for (i=0; i<n2; i++) {
           if (X[i]>=z) {
              M--;
              n--;
           } else {
              X[j]=X[i];
              j++;
           }
        }
      } else {
        for (i=0; i<n2; i++) {
           if (X[i]<=z) {
              B+=X[i];
              n--;
           } else {
              X[j]=X[i];
              j++;
           }
        }
      }
   }
   return B/M;
}
```

Fig. 14. C-code implementation of the recursive pseudo-code for $\mathcal{Z}$ from Section VII

B. Estimating z^* from subsets of sizes

One way to reduce the computational resources required for root finding is to reduce the number of sizes used as input. In this section, we consider the problem of estimating z^* from a random subset of the original sizes $\{x_1, \ldots, x_n\}$. Here sizes are sampled uniformly with some probability $q \in (0,1)$, i.e.,independently of the size values. The random set Q of selected sizes is used to estimate z^*. P_q will denote the corresponding distribution.

Let $N_z^Q = \sum_{x \in Q} p_z(x)$ and for each Q estimate z^* through z^Q, the solution z to the equation $N_z^Q = qM$. Here we have scaled down the target volume in the same proportion as the mean number of sizes used to estimate z^*. Our aim here is to analyze the sampling error arising from this procedure.

The following Theorem uses a form of Chernoff bounds (see [27, Chapter 4])) that says if X is a sum of independent random variables each with range in the interval $[0,1]$ and $\mathsf{E}X = \mu$, then

$$\mathsf{P}[X < (1-\delta)\mu] \quad \leq \quad \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu} \tag{37}$$

and

$$\mathsf{P}[X > (1+\delta)\mu] \quad \leq \quad \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{\mu} \tag{38}$$

for $\delta \in (0,1)$ and $\delta > 0$ respectively.

Theorem 8: N_{z^Q} converges exponentially quickly to M. In particular

$$\mathsf{P}_q[N_{z^Q} > (1+\eta)M] \quad \leq \quad \left[(1+\eta)e^{-\eta}\right]^{Mq} \tag{39}$$

and

$$\mathsf{P}_q[N_{z^Q} < (1-\eta)M] \quad \leq \quad \left[(1-\eta)e^{-\eta}\right]^{Mq} \tag{40}$$

for $\eta > 0$ and $\eta \in (0,1)$ respectively.

Proof: Given $\eta > 0$, define z_- to be the solution to $N_{z_-} = (1+\eta)M$. Observe that $N_{z^Q} > (1+\eta)M$ iff $z^Q < z_-$ iff $N_{z_-}^Q < qM$ iff $N_{z_-}^Q < \mathsf{E}_q[N_{z_-}^Q]/(1+\eta)$. The last equivalence is because $\mathsf{E}_q[N_{z_-}^Q] = qN_{z_-} = qM(1+\eta)$. Now N_z^Q is a sum of independent random variables each taking values in $[0,1]$: each $x \in \{1,\ldots,n\}$ contributes to the sum N_z^Q with probability q, and if present contributes $p_z(x)$. The first bound then follows from the first inequality in (37) on identifying $1+\eta$ with $(1-\delta)^{-1}$. The second bound is obtained similarly by considering the solution z_+ to $N_{z_+} = M(1-\eta)$ and identifying $1-\eta$ with $(1+\delta)^{-1}$. ■

VIII. Alternative Measurement Methodologies and Related Work

Our work is immediately applicable to current network measurement infrastructure, specifically in mediation stations and flow collectors fed by the large installed base on NetFlow enabled routers. In this section we discuss the feasibility, or otherwise, of alternative measurement methods for meeting our goals, and discuss some related work.

1) Counters: Sometimes it is suggested that routers maintain byte and packet counters and export flow records periodically to the collection system. There are a number of reasons why this can be a bad idea. If the export period is too short, a larger number of flow records will be produced since longer flows will be divided up into many records, increasing measurement infrastructure costs. If the export period is too long, a large number of counters will have maintained at the router, many of them for inactive flows. For example, a heavily loaded backbone router with many interfaces might see 10's or even 100's of millions of distinct flow keys per hour. Moreover, the export latency between periodic exports limits the response time of analysis, and leaves the flow data vulnerable to loss in the event of router reset or failure.

The methods of generating flow statistics currently deployed in network routers—upon which our work is based—are able to circumvent these problems; These can be viewed as an economical and adaptive way of keeping counters, since counters are kept only for active flows, with counter memory released on termination. The need to control memory consumption is one reason that the interpacket timeout is kept small in practice, typically about a minute or less. A recent phenomenon reemphasizes this need. Some denial of service attack tools generate packet streams in which the source IP address is randomly forged [26]. Assuming for simplicity that no source addresses are repeated, each packet gives rise to a separate flow cache entry in the router. Thus memory consumption at the router arising from such flows grows linearly with the flow timeout.

2) Packet header collection: Collection, sampling and export of packet headers suffers from several drawback relative to flow collection. The main drawback is volume, as noted in the introduction. Secondly, there is currently a large installed base of routers that have no ability to export packet-level measurements. Although this may change in future, volume constraints would limit the taking of full (i.e. unsampled) header traces to filtered subsets of packets.

Since flow statistics do not include the locations of packets within the flow, they are of limited use in studies of packet level traffic processes, such as self-similar [25] or multifractal properties [18], [34], or TCP dynamics [29]. However, whereas packet level detail is crucial to understand these matters, many important network management applications and supporting research work — including all those mentioned in Section I-A.1 — do not require it. For these, it suffices to know network usage differentiated by primary keys (i.e. as fine as those commonly used in flow definitions: IP address, ports, AS's, etc.), at timescales no smaller than the duration of the flows themselves. We have argued that differentiation is also necessary to satisfy the needs of a variety of network management applications.

3) Sampling Methods Currently Implemented: Packet sampling may in any case be required for the construction of flow records. This is because flow cache lookup at full line rate, if possible, requires fast expensive memory. Sampling spaces out the packets to allow for execution in slower cheaper memory. In Section III we showed that the formation of packet sampled flow statistics provides greater estimation accuracy that packet sampling alone for the same amount of measurement traffic. For this reason, we recommend using only as much packet sampling as is necessary to control load in the router: any further data reduction that is required should be performed using threshold flow sampling.

4) Proposed Sampling Methods: Sample and Hold [15], uses modified packet sampling in the router in order to focus on longer flows. A flow cache lookup is performed for each incoming packet, and if a match is found, the flow's statistics are updated as usual. If a match is not found, a new cache entry is created with probability depending on the number of bytes in the packet. Section III showed the performance of sample and hold and threshold sampling in the measurement infrastructure to be very close. However, Sample and Hold is not available in current routers; whereas threshold sampling is currently operational, and immediately applicable to the the date produced by the large installed base of NetFlow enabled routers.

5) Filtering and Aggregation: Filtering and aggregation by key are means of reducing data volume. Filtering restricts attention to a particular subset of data, e.g., all traffic to or from a range of IP addresses. However, not all questions can be answered from measurements that has passed through preconfigured filters. For example, we do not know in advance which addresses to look for to determine the most popular destination web site for traffic on a given link.

Similarly, aggregation over keys (e.g. over ranges of IP addresses, rather than individual addresses) would require that the finest required level of aggregation be known in advance. This would generally limit the ability to perform exploratory traffic studies. When queries are known in advance, then more specialized sketching algorithms can be applied to the stream of flow records; see [2], [28] for a survey. In fact, our sampling may be used as a common front-end to several more sophisticated streaming algorithms. The point here is that threshold sampling is simple enough to keep up with a very fast data stream, producing a reduced data set that can be fed into a more sophisticated but slower streaming algorithms. However, threshold sampled flow data can be arbitrarily aggregated to perform analyses not anticipated at the time the data was collected.

6) Related work: Stratified sampling is a well known technique by which to reduce the variance associated with statistical estimation from a population, by varying the selection probability of a datum according to its value; see e.g. [7]. Optimization results for the choice of strata exist when the population distribution lies in certain parameterized families. Independent and deterministic 1 in N sampling, as well as stratified sampling out of finitely many bins, have been compared for packet sampling in [6]. The aim of this work was to efficiently estimate packet size distributions.

The focus of our work is different. We estimate total usage in a given key by ensuring that we sample all important contributions to the total. Moreover, we make no assumption on the underlying distribution of flow sizes. In a further development, we coupled the threshold sampling scheme described here to the problem of usage-sensitive pricing; see [12]. The main new idea is that since larger usage can be more reliably estimated, charging should be sensitive only to usage that exceeds a certain level; for lesser usage a flat rate applies.

Other methodological work on sampling for network measurement includes Trajectory Sampling [10], a method to sample a packet at either all links or no links of network. Another paper deals with reconstruction of the statistics of original flow records from those of packet sampled flows; see [13]. Standards for network event sampling based on randomizing the inter-sample time have been set out in [30]. Sample and Hold [15] has already been discussed. We saw in section III-B that, although it is not in general possible to construct an unbiased usage estimator from the measurements produced by this method, its accuracy is similar to that of the present method. However, unlike the present work, this method would require modification of routers. A review of these and other sampling methods used or proposed for passive Internet measurement can be found in [9].

IX. Conclusions and Further Developments

This paper was motivated by the requirement to estimate of fined-grained volume of network traffic with different attributes (e.g. IP address and port numbers) when it is infeasible either to accumulate counters on attribute of interest, or to simply collect a portion of each object for further analysis. We have studied how best to sample a stream of measurements in order to minimize a weighed sum of the total number of samples and the variance of the estimated sum of the measurements. The main complication is that if the measurements have heavy tailed distributions, such as occur for the byte and packet sizes of IP flows, then the elementary method of sampling at a fixed rate performs very poorly.

We presented a simple sampling model, and we determined a parameterized family of optimal solutions to this model. We showed that the optimal solution performs very well when we simulated it on a trace of real flow measurements gathered at routers. We also developed simple heuristics that (i) implement this optimal solution and (ii) given a collection of data and a desired amount of sampled data determined the sampling parameter to use. For applications that need true dynamic control, we specified several dynamic control algorithms, with additional features to enhance convergence and ameliorate the effects of systematic variations in the offered load of objects to be sampled. We showed that they performed very well on both real data, and real data with artificially created variations.

These heuristics and dynamic control algorithms could be incorporated into router vendor software and thus give network managers control on the amount network measurements traffic that needs to be processed. With capabilities available today one can sample at the mediation stations and the collector. The sampling strategy described in this paper is currently used to collect NetFlow records collected extensively from a large IP backbone. The collection infrastructure deals with millions of NetFlow records per second. Measurements from some tens of routers are transmitted to a smaller number of distributed mediation servers. The servers aggregate the records in multiple dynamic ways, serving the needs of a variety of network management applications. The aggregates are sent to a central collection point, for report generation, archival, etc. Without the sampling strategy, there would need to be an order of magnitude greater capital investments to provide the facilities to process the measurement stream.

The ideas in this paper have been further developed since it was first submitted for publication, resulting in a number of other publications. These we now briefly list. Usage-based charging can be coupled to the present threshold sampling method in such a way that the estimated usage is relatively insensitive to sampling errors from threshold sampling; see [12] for further details. The interaction between packet sampling (such as occurs in sampled NetFlow) and threshold sampling, and the dimensioning of measurement infrastructures that use both these two methods is examined in [11]. Threshold sampling with hard constraints on the volume of samples taken is examined in [14].

Acknowledgment.

We thank Jennifer Rexford and Cristi Estan for comments on previous versions of this paper.

References

[1] J. Apisdorf, K. Claffy, K. Thompson, and R. Wilder, “OC3MON: Flexible, Affordable, High Performance Statistics Collection,” For further information see http://www.nlanr.net/NA/Oc3mon

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom”, “Models and Issues in Data Stream Systems”, In: Proceedings of the Twenty-First ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 1–16, 2002.

[3] R. Cáceres, N.G. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B.Krishnamurthy, D. Lavelle, P.P. Mishra, K.K. Ramakrishnan, J. Rexford, F. True, and J.E. van der Merwe, “Measurement and Analysis of IP Network Usage and Behavior”, IEEE Communications Magazine, vol. 38, no. 5, pp. 144–151, May 2000.

[4] Cisco NetFlow; for further information see http://www.cisco.com/warp/public/732/netflow/index.html and http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s11/12s_sanf.htm

[5] K.C. Claffy, H.-W. Braun, and G.C. Polyzos. Parameterizable methodology for internet traffic flow profiling. *IEEE Journal on Selected Areas in Communications*, 13(8):1481–1494, October 1995.

[6] Kimberly C. Claffy, George C. Polyzos, and Hans-Werner Braun. Application of Sampling Methodologies to Network Traffic Characterization. *Computer Communication Review*, 23(4):194–203, October 1993, appeared in Proceedings ACM SIGCOMM’93, San Francisco, CA, September 13–17, 1993.

[7] W. Cochran, “Sampling Techniques”, Wiley, 1987.

[8] T. H. Cormen and C. E. Leiserson and R. L. Rivest, ”Introduction to Algorithms”, MIT Press/McGraw-Hill, Cambridge, Massachusetts 1990.

[9] N.G. Duffield, “Sampling for Passive Internet Measurement: A Review”, Statistical Science, 2004, to appear.

[10] N. G. Duffield and M. Grossglauser, “Trajectory Sampling for Direct Traffic Observation”, *IEEE/ACM Transactions on Networking*, April 2001, to appear. Abridged version appeared in Proc. ACM Sigcomm 2000, Computer Communications Review, Vol 30, No 4, October 2000, pp. 271–282.

[11] N.G. Duffield and C. Lund, “Predicting Resource Usage and Estimation Accuracy in an IP Flow Measurement Collection Infrastructure”, ACM SIGCOMM Internet Measurement Conference 2003, Miami Beach, Fl, October 27-29, 2003

[12] N.G. Duffield, C. Lund, M. Thorup, “Charging from sampled network usage,” ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco, CA, November 1-2, 2001.

[13] N.G. Duffield, C. Lund, M. Thorup, “Estimating flow distributions from sampled flow statistics”, ACM Sigcomm 2003, Karlsruhe, Germany, August 25-29, 2003.

[14] N.G. Duffield, C. Lund, M. Thorup, “Flow Sampling Under Hard Resource Constraints”, In Proc ACM SIGMETRICS 2004, New York, NY, June 12-16, 2004

[15] C. Estan and G. Varghese, “New Directions in Traffic Measurement and Accounting”, Proc SIGCOMM 2002, Pittsburgh, PA, August 19–23, 2002.

[16] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, F. True, ”Deriving traffic demands for operational IP networks: methodology and experience”, In Proc. ACM Sigcomm 2000, Computer Communications Review, Vol 30, No 4, October 2000, pp. 257–270.

[17] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, “NetScope: traffic engineering for IP networks”, IEEE Network, vol. 14, no. 2, pp. 11–19, March-April 2000.

[18] A. Feldmann, A.C. Gilbert and W. Willinger, “Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic”, Proceedings ACM SIGCOMM ’98, Vancouver, BC, 1998.

[19] A. Feldmann, J. Rexford, and R. Cáceres, “Efficient Policies for Carrying Web Traffic over Flow-Switched Networks,” *IEEE/ACM Transactions on Networking*, vol. 6, no.6, pp. 673–685, December 1998.

[20] B. Fortz and M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights”, In: Proceeding IEEE INFOCOM 2000, Tel Aviv, Israel, 2000, pp. 519–528.

[21] B. Hajek, “Extremal splittings of point processes”, Math. Oper. Res. vol. 10, pp. 543–556, 1985.

[22] Inmon Corporation, “sFlow accuracy and billing”, see: http://www.inmon.com/PDF/sFlowBilling.pdf

[23] ”Internet Protocol Flow Information eXport” (IPFIX). IETF Working Group. See: http://net.doit.wisc.edu/ipfix/

[24] P. L’Ecuyer, ”Efficient and portable combined random number generators”, Communications of the ACM 31:742–749 and 774, 1988.

[25] W.E. Leland, M.S. Taqq, W. Willinger, D.V. Wilson, “On the self-similar nature of Ethernet traffic”, Proceddings ACM SIGCOMM ’93, San Francisco, CA, 1993

[26] D. Moore, G. Voelker, S. Savage, “Inferring Internet Denial of Service Activity”, Proceedings of the 2001 USENIX Security Symposium, Washington D.C., August 2001.

[27] R. Motwani and P. Raghavan, “Randomized algorithms”. Cambridge University Press, Cambridge, 1995.

[28] S. Muthukrishnan, “Data Streams: Algorithms and Applications”, Manuscript based on invited talk from *14th SODA*, 2003.

[29] V. Paxson, “Automated Packet Trace Analysis of TCP Implementations”, Proceedings ACM SIGCOMM ’97, Cannes, France, 1997.

[30] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, “Framework for IP Performance Metrics”, RFC 2330, available from: ftp://ftp.isi.edu/in-notes/rfc2330.txt, May 1998.

[31] J. Postel, “Transmission Control Protocol,” RFC 793, September 1981.

[32] Qosient, LLC, “Argus”, see: http://www.qosient.com/argus/index.htm

[33] Real Time Flow Measurement, see: http://www.auckland.ac.nz/net/Internet/rtfm/.

[34] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, “A Multifractal Wavelet Model with Application to Network Traffic”, IEEE Transactions on Information Theory, vol. 45, pp. 992-1018, 1999.

[35] Riverstone Networks, Inc., see: http://www.riverstonenet.com/technology/

[36] XACCT Technologies, Inc., see: http://www.xacct.com