6-1-2011

Effects of network trace sampling methods on privacy and utility metrics

Phillip A. Fazio
Dartmouth College

Recommended Citation

Fazio, Phillip A., "Effects of network trace sampling methods on privacy and utility metrics" (2011). *Dartmouth College Undergraduate Theses*. 77.
https://digitalcommons.dartmouth.edu/senior_theses/77

Effects of network trace sampling methods on privacy and utility metrics

Phil Fazio
Dartmouth College, Hanover NH, USA
phillip.a.fazio.10@alum.dartmouth.org

Senior Undergraduate Thesis
Dartmouth Computer Science Technical Report TR2011-697

ABSTRACT

Researchers studying computer networks rely on the availability of traffic trace data collected from live production networks. Those choosing to share trace data with colleagues must first remove or otherwise anonymize sensitive information. This process, called sanitization, represents a tradeoff between the removal of information in the interest of identity protection and the preservation of data within the trace that is most relevant to researchers. While several metrics exist to quantify this privacy-utility tradeoff, they are often computationally expensive. Computing these metrics using a sample of the trace, rather than the entire input trace, could potentially save precious time and space resources, provided the accuracy of these values does not suffer.

In this paper, we examine several simple sampling methods to discover their effects on measurement of the privacy-utility tradeoff when anonymizing network traces prior to their sharing or publication. After sanitizing a small sample trace collected from the Dartmouth College wireless network, we tested the relative accuracy of a variety of previously-implemented packet and flow-sampling methods on a few existing privacy and utility metrics. This analysis led us to conclude that, for our test trace, no single sampling method we examined allowed us to accurately measure the tradeoff, and that some sampling methods can produce grossly inaccurate estimates of those values. We were unable to draw conclusions on the use of packet versus flow sampling in these instances.

1. INTRODUCTION

Computer-network researchers depend on the availability of traffic trace data collected from live production networks. It is often difficult to obtain this type of data, however, which aid in the solving of problems related to the structure and usage of real-world networks. These difficulties are caused in no small part by the imposing infrastructure requirements and the need to obtain appropriate permissions to collect trace data. Due to this scarcity of data, it becomes extremely important for the community of network researchers to share available traces amongst several research projects; this has resulted in the creation of data archive resources such as CRAWDAD [10].

Those that choose to share trace data with their colleagues encounter the additional burden to remove or otherwise anonymize sensitive information (e.g., identities of network users or network topology). Identifying and properly removing this information is often a daunting and complex task, one that may deter researchers from releasing their collected trace data. It is particularly difficult to identify the types of data that are sensitive in network traces, leading to the increased possibility of sensitive information remaining within a trace thought to be properly sanitized.

Sanitization, by nature, represents a tradeoff between the removal of information to fulfill privacy requirements and the preservation of information that may be useful in later analysis. To streamline the process of trace sanitization and to better analyze this tradeoff, we proposed the NetSANI (Network Trace Sanitization and ANonymization Infrastructure) framework and API [14]. Using either existing or user-defined metrics, the framework is designed to allow the researcher to analyze an anonymized trace to determine whether it meets prespecified privacy and utility goals. In computing the metrics, NetSANI works with a *sample* of the collected trace with the goal of saving precious time and resources when developing an anonymization scheme.

Trace sampling is hardly a new concept; the benefits and accuracy of various sampling techniques have been analyzed with respect to anomaly detection [1, 2, 23, 24], computation of traffic flow statistics [5, 7, 12], network management [15], and sample space efficiency [32]. However, little or no attention has been paid to the effects of sampling when analyzing anonymized network traces.

In this thesis, we apply several well-known sampling methods to network traces and analyze their effect on some existing privacy and utility metrics. By comparing these results to the same analysis on the unsampled traces, we seek to discover which sampling methods produce the most accurate estimates of the tradeoff between privacy and utility and examine any trends in the experiments. These results can then be used to benefit NetSANI, by either the identification of globally-effective sampling methods or the discovery of specific sampling methods that benefit a specific trace analysis; the end goal is to allow researchers to quickly identify an appropriate anonymization strategy for their network traces.

In Sections 2 and 3, we summarize some established network-trace sampling methods and metrics to compute privacy and utility, respectively. Then, in Section 4, we test a few of these sampling methods on an anonymized TCP/IP network trace to determine their effect on analyses of the privacy/utility tradeoff on that trace. We discuss the results of these experiments in Section 5 and how they may aid a researcher to develop an adequate anonymization strategy. Finally, we discuss related and future work in Section 6 and conclude in Section 7.

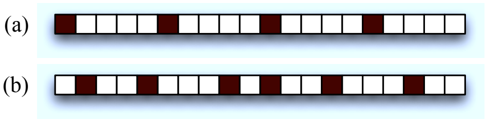

Figure 1: Examples of (a) periodic (1-of-n) sampling and (b) random sampling.

2. SAMPLING METHODS

In this section, we introduce several methods used to sample or otherwise filter data. Because this thesis is concerned about the effects of these sampling methods on network traces in particular, we choose to group these methods into two somewhat broad categories: those that operate by examining individual packets or records (*record-based*), and those that operate on network flows (*flow-based*). The latter apply only to packet traces, whereas the former may also apply to other trace forms (such as network access-point association logs).

Each of the sampling methods here behaves according to fixed predefined rules for the entire sampling process; therefore, they are considered examples of *conventional sampling.* Conversely, methods that are more complex and change behavior based on already seen data are considered *adaptive sampling* [15].

2.1 Basic record-based sampling

The simplest of sampling methods, *periodic sampling* (also known as deterministic or 1-of-n sampling), functions by choosing every nth record in the original trace [7, 15]. Applying a probability distribution to the records results in a variant method, *random sampling.* Each record processed then has a probability $1/n$ of being included in the sample; depending on the distribution function (such as uniform or Poisson distributions), the number of records present in a sample may vary run-to-run. Figure 1 shows basic examples of periodic and random sampling.

Brauckhoff et al. [2] demonstrated that periodic sampling of packets preserved the ability to accurately estimate packet statistics, such as total packet count and average packet size. Additionally, changes in the distribution of packet fields over time – also called *feature entropy* – were also reasonably well-preserved in the sampled trace even at high values of n. Preservation of feature entropy is especially useful, as variances in the entropies of specific fields have been used to detect network anomalies such as DoS and portscans [20]. Unfortunately, periodic sampling was also shown to significantly degrade the ability to accurately estimate similar statistics for flows contained within the trace, producing "grossly inaccurate" results even at low values of n [2, 32].

2.2 Stratified record-based sampling

Stratified sampling attempts to improve on the methods discussed in Section 2.1 by first using predetermined rules to sort and separate records into mutually-exclusive groups, or strata. Individual strata are then sampled separately, often by periodic or random sampling as described above. For example, Hernandez et al. [15] separated records into fixed time-duration strata prior to sampling a single record at random in each interval; this basic example of stratified sampling is illustrated in Figure 2.

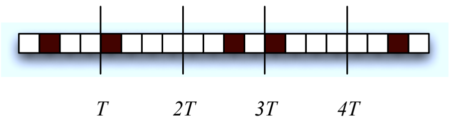

Figure 2: Example of stratified sampling, as shown by Hernandez et al. [15]. Upon dividing the trace into fixed-time strata, each strata was then independently sampled to choose one of its records at random.

2.3 Flow-based sampling

A *network flow* (also known as a *traffic flow*) is an artificial construct analogous to a physical phone connection or call, officially defined in RFC 2722 as "a portion of traffic, delimited by a start and stop time, that belongs to [a network entity]" [3]. The NetFlow protocol developed by Cisco Systems defines by default a flow as a connection that either terminates according to TCP protocol, or has seen no new traffic in one minute [23].

The sampling of complete flows, or *flow sampling*, is an alternative to record-based sampling, albeit one with a large penalty in performance and space requirements, that allows for more accurate estimates of unsampled flow statistics [1, 16]. Methods used to sample flows are similar to their record-based counterparts once the flows present within the packet trace are identified.

Periodic and uniform random sampling of flows has been shown to have a bias towards larger, so-called "heavy-hitter" or "elephant" flows [16]. Specific flow-sampling algorithms have been developed that attempt to correct (or take advantage of) this bias and to address the high resource requirements. In this section, we summarize a few of those methods.

2.3.1 *Opportunistic flow sampling*

Androulidakis et al. [1] describe the concept of *opportunistic sampling*, in which the sampling algorithm specifically targets either larger or smaller flows to "[magnify] the appearance of anomalies within the sampled data set", especially when an anomaly may be detected by outlying traffic. The targeting of large flows is called *smart sampling* and the targeting of small flows is called *selective sampling.*

Smart sampling. Smart sampling [1, 23] dictates that for each flow F of size F_x bytes, the probability $p(F)$ that flow F is selected in the sample is dictated by the expression in Equation 1, where z is a threshold flow size to reduce space requirements:

$$p(F) = \begin{cases} F_x/z & x < z \\ 1 & x \geq z \end{cases} \tag{1}$$

Flows larger than the threshold are always chosen for the sample, whereas flows smaller than the threshold are sampled with probability proportional to their size. Mai et al. [23] analyzed smart sampling with respect to portscan detection, which relies on the presence of small flows to detect possible anomalies; predictably, smart sampling performed poorly in this instance.

Selective sampling. Unlike smart sampling, selective sampling takes the opposite approach, targeting smaller flows that may be responsible for attacks such as DDoS, portscans, and worm propogations. Selective sampling [1] is defined in Equation 2, where c is a constant probability $0 < c < 1$ that a flow may be selected, z is a packets-per-flow threshold, and $n \geq 1$ is a parameter designed to allow further control of the number of "heavy-hitting" flows in the sample:

$$p(F) = \begin{cases} c & x \leq z \\ \frac{z}{n \cdot F_x} & x > z \end{cases} \tag{2}$$

Under certain circumstances, selective sampling was shown to improve detection of anomalies, especially those that depend on the presence of small flows; Androulidakis et al. found "anomalies that would otherwise be untraceable" [1].

2.3.2 Hybrid methods

Many have attempted, with mixed results, to combine the estimation accuracy benefits of flow-based sampling methods with the performance advantages of record-based sampling methods.

An example of a hybrid approach is seen in the *sample-and-hold* (S&H) method [13], in which a flow table $\mathcal{F}$ is maintained during the parsing of records from a network trace $\mathcal{T}$. Each packet $\mathcal{T}_i$, part of a flow $F_{\mathcal{T}_i}$, is treated as follows:

- if the corresponding flow is in the table, that is, $F_{\mathcal{T}_i} \in F$, add $\mathcal{T}_i$ to the sample set;
- otherwise, sample the record with a probability p proportional to its size. If packet $\mathcal{T}_i$ is selected for addition to the sample set, also add $F_{\mathcal{T}_i}$ to F at this time.

Despite the potential benefits to this middle-of-the-road approach, Mai et al. showed that the performance of S&H was inferior to *all* other methods described above when small-sized flows are desired in the sample [23]. In addition, by definition, S&H may miss the beginnings of several flows, and fail to recognize the termination of a flow. Therefore, we did not implement this method in our experiments but mention it here for completeness.

2.4 Adaptive and "fair" methods

Adaptive sampling methods, those that dynamically change the sampling rate based on prior traffic, have also been proposed primarily for their benefits during live collection of data and passive traffic measurement, rather than post-capture analysis; therefore, we only briefly mention them here. Those performing passive traffic analysis are confronted with several challenges, including the limited memory resources at collection points (routers) and the increasing amount of network traffic given increased link capacity and speeds.

When collecting trace data, it is also of interest to use these resources to produce the "fairest" trace possible. Because the conventional sampling methods described above tend to produce samples biased towards "heavy-hitting" flows, "fair" in this context means determining methods to correct this bias [17, 26], often at the expense of simplicity. To make "fair" methods more feasible to implement, further efforts have focused on reducing the space complexity required to perform a fair sampling [32].

3. PRIVACY AND UTILITY METRICS

There are two types of metrics used in the analysis of network traces: *privacy metrics*, which measure the degree to which a sanitization method fulfills its predetermined requirements, and *utility metrics*, which measure the usefulness of data preserved after the sanitization of a trace.

3.1 Privacy metrics

A *privacy metric* (also known as an *anonymity metric*), is defined by Kelly et al. [18] as a quantification of how well an anonymization strategy hides the identity of sensitive information or users against a particular attack.

Individual privacy metrics may be sensitive to the underlying types or structure of the dataset to be analyzed or may be generic enough to apply to a wider range of data formats. In our work, we classify anonymity metrics into two broad models: those that are *microdata-based* and those that are *network-based.* Microdata-based metrics assume the dataset is organized like a relational database in which certain fields may be designated as "sensitive" *a priori*; network-based metrics use statistical and probabilistic information about the dataset to simulate an adversary's knowledge. The NetSANI framework [14] is designed to accommodate both models, and our experiments here use examples of both types of metrics.

3.1.1 k-anonymity

We begin with a well-known and simple microdata-based metric, *k-anonymity* [6]. When preparing to release a dataset that contains fields known to contain "sensitive" information, we must recognize that the adversary may use some non-sensitive fields in conjunction with externally available information to identify sensitive data; this set of attributes are called *quasi-identifiers* [6]. An adversary may have external information that maps quasi-identifiers to actual identities, and thus may be able (with the released dataset) to map identities to sensitive values. The microdata-based metric of *k-anonymity* states that for each combination of values of quasi-identifiers, that combination can be matched to at least k identities.

The dataset presented in Table 1 is an example of an anonymized dataset that achieves 2-anonymity with the set of sensitive attributes $S = query$. This means that for each individual record in the dataset, there exist at least 2 instances of a single quasi-identifier $Q = ip, date, time$ that could be associated with that record's sensitive value. In our sample dataset, this means that the sensitive query of "skin rash" could have originated from at least 2 records with the quasi-identifier (96.234.68.2*, 2008-10-**, 23**).

k-anonymity is quite limited in its scope, and does not attempt to measure the variety of sensitive values associated with each quasi-identifier; for instance, all records within the quasi-identifier (96.234.69.**, 2008-10-2*, 234*) in the sample dataset can be associated with the sensitive query "AIDS medicine". Additional microdata-type metrics such as l-diversity and t-closeness attempt to address this and other shortcomings [18].

We can define k-anonymity as a simple ternary privacy metric, as shown in Table 2; it measures privacy at three levels: preserved, degraded, or eliminated. By specifying a threshold value z, we measure whether the network trace is k-anonymous such that $k \geq z$. If so, then privacy is considered to be preserved; else privacy is considered degraded, or

IP address	Date	Time	Query
96.234.69.*	2008-10-2*	234*	AIDS medicine
96.234.69.*	2008-10-2*	234*	AIDS medicine
222.154.155.***	2008-10-**	23**	m-invariant
222.154.155.***	2008-10-**	23**	l-diversity
96.234.68.2*	2008-10-**	23**	cook book
96.234.68.2*	2008-10-**	23**	skin rash
96.234.68.2*	2008-10-**	23**	filling station
96.234.68.2*	2008-10-**	23**	winter coats
129.170.111.1**	2008-10-2*	235*	tan salon
129.170.111.1**	2008-10-2*	235*	mcdonalds jobs

Table 1: Sample dataset for a search engine network log anonymized such that k-anonymity is achieved for $k = 2$ [18].

in the case of $k = 1$, eliminated [18].

Privacy level	Metric level = z
Preserved	$k \geq z$
Degraded	$k < z$
Eliminated	$k = 1$

Table 2: k-anonymity privacy metric [18].

3.1.2 L1-similarity

Prior to introducing the next metric, we introduce the concept of a *network object* [8, 14]. A network object is an entity whose identity a trace publisher seeks to protect and/or retain utility, such as a host, subnet, or web page. It is important to note that an object may be defined by more than one record in a trace (multiple packets may be from the same host); the converse holds, as a record may belong to one or more network objects (for example, a TCP packet refers to both the *src* host and the *dst* host).

L1-similarity [18] estimates the anonymity of an object by computing a distance between the distribution of values of an anonymized object X and the distribution of values of an unanonymized object Y, defined as

$$sim(X, Y) = 2 - \sum_{z \in X \cup Y} |P(X = z) - P(y = z)|. \qquad (3)$$

The maximum value of $sim(X, Y)$ is 2, which represents an identical distribution of features between the two objects. This notion is somewhat counterintuitive, representing the maximum preservation of anonymity because an attacker fails to gain additional knowledge from the anonymized dataset. Conversely, if the distributions of the two objects are totally disjoint, the similarity is 0, and the attacker gains "complete or substantial knowledge of identities and relationships" [18]. This metric is summarized in Table 3.

Privacy level	Metric level = $sim(X, Y)$
Preserved	$sim(X, Y) = 2$
Degraded	$0 < sim(X, Y) < 2$
Eliminated	$sim(X, Y) \approx 0$

Table 3: L1-similarity anonymity metric [18].

Privacy level	Metric level = D_r
Preserved	$D_r = 1$
Degraded	$0 \leq D_r < 1$
Eliminated	$D_r \approx 0$

Table 4: Entropy anonymity degree (EAD) privacy metric [18].

3.1.3 Entropy anonymity degree (EAD)

Consider a situation of an adversary attempting to discern the author of each of several messages sent across a network. The adversary knows the set of all the possible authors, but at the onset, it appears equally likely that any author may have sent a given message. However, upon learning additional information, such as how prolific each author is, the adversary is able to guess with more certainty which author may have written a given message. The metric of *entropy anonymity degree* uses entropy to measure how much information the adversary has gained, and thus, the degree of anonymity that the author of a message retains after the attack [11].

Mathematically, let I be the set of distinct values that are represented in a probability distribution X; in this case, each $i \in I$ represents an author a_i in the set of all possible authors A. Let p_i represent the probability that a_i is responsible for a message. Therefore, $p_i = Pr(X = i)$, where Pr is a probability mass function.

The entropy of this probability distribution is defined as follows, where N is the size of the sample space [8, 11]:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i) \qquad (4)$$

Entropy anonymity degree (noted D_r) normalizes the result of $H(X)$ above, dividing it by the maximum entropy $H_{max} = \log_2 N$ of the system:

$$D_r = \frac{H(X)}{H_{max}}, \qquad (5)$$

where $0 \leq D_r \leq 1$ [29].

Logically, the maximum degree of anonymity is achieved when the attacker finds it equally likely that any author is responsible for sending a given message; likewise, anonymity has been eliminated when the attacker is certain or near-certain of the author of that message. This may be represented in an anonymity metric as follows (summarized in Table 4): when $D_r = 1$, all values across the attribute are equally likely, and privacy is fully preserved. As D_r decreases, privacy becomes increasingly degraded until, when $D_r = 0$, privacy is considered fully eliminated.

Note that the value of EAD, D_r, is specific to the probability mass function Pr of the sample space X, which is dependent on the type of information that the attacker possesses.

3.2 Utility metrics

Quantified measurement of utility is difficult, because those looking to use trace data (researchers) often have specific use cases. Therefore, unlike the privacy metrics, developing an overarching utility metric is a much more challenging endeavor [4]. Almost all metrics to date center on the concept of anomaly detection as a measure of utility, because often the search for anomalous traffic (e.g., DoS attack, portscan) requires a wide range of useful data from the trace. For example, several important anomalies may be mined from examining the entropy of traffic features [1, 20]:

- **DDoS attack**: a *distributed denial of service* attack attempts to target a single service, in order to make a resource unavailable to others; the attack may come from many sources.
  Fields affected: large decrease in H for *dst* and *dstprt*.

- **Portscan**: in a *portscan* attack, a single sender sends packets to a host over a wide range of ports, with the intent to identify services available at the host.
  Fields affected: large decrease in H for *src*, *dst*, and *srcprt*, slight increase in H for *dstprt*, and slight decrease in H for F_x, the flow size.

- **Worm propogation**: a program that replicates itself in an attempt to exploit and infect other machines.
  Fields affected: large decrease in H for *src* and *dstprt*, slight increase for *dst* and *srcprt*, and slight decrease in H for F_x.

3.2.1 Snort

While it is possible to implement separate metrics measuring detection of the attacks above, the open-source intrusion-detection tool Snort [28] contains the tools necessary to determine the type and extent of a wide range of anomalies during either live packet capture or post-capture analysis [21, for example]. Because Snort contains rules designed to discover instances of all three attacks described above, we develop a simple metric (Table 5) based on the number and severity of alerts when running a TCP/UDP network trace through the intrusion-detection tool; alerts with higher severity and/or lower frequency are assigned higher utility value.

Table 5: Snort utility metric: *value* assigned to alert by type

Class	*Value*
No Alert	0.0
Truncated header	0.1
TCP reset	0.2
SNMP request	0.3
TCP window close	0.3
SNMP trap	0.3
ICMP ping	0.4
ICMP ping NMAP	0.4
ICMP redirect host	0.4
ICMP unreachable host/port	0.4
SQL ping	0.7
TCP portscan	0.9
TCP portsweep	1.0

Because a trace theoretically has no measurable utility limit, we compare the values of the utility metric of the raw trace against the same analysis run on the sanitized trace.

4. EVALUATION

In this section, we present the results of our analysis of the effects of the aforementioned sampling methods on privacy metrics and Snort utility measurements on a modest-sized trace.

The trace used for this analysis was collected over a nine-hour period on the Dartmouth College campus wireless network in December 2003 and contains 1,651,553 IP packets with either TCP or UDP headers. It was collected using 18 wireless sniffers located in 14 buildings across campus.

4.1 Trace preparation

Prior to sampling the trace, we used the open-source `anontool` [19] program to produce a sanitized version of our trace. Its configuration was as follows:

- *IP addresses*: prefix-preserving mapping
- *Port numbers*: one-to-one mapping
- *Payload*: hashed

The payload of TCP/UDP packets was truncated during collection of the trace. Thus, `anontool` calculates the hash of the remaining payload data, if any exists.

By using this sanitization configuration, the count of packets and flows is consistent between the raw and sanitized trace, and the distributions of features remain constant, which allows us to measure the changes incurred upon these distributions by the sampling methods.

4.2 Sampling configuration

In this subsection, we summarize the implementations of our sampling methods and the configurations employed, to produce sampled versions of our raw (pre-sanitization) and sanitized traces.

4.2.1 Experiment framework and configuration

Packet-based sampling
To perform packet-based sampling on our traces in `pcap` format, we used a custom Python [27] wrapper for the C library `libpcap` [22] to count the number of TCP and UDP packets contained within the trace and collect field information as necessary (e.g., *src* values when performing a stratified sample on that field), performed the selected sampling method on the input trace, and constructed a new `pcap` trace containing only the packets selected for the sample.

Flow-based sampling
Using the tool `tcptrace` [31] on default settings, we first constructed a list of the distinct connections (flows) present within our packet trace, and determined the total number of TCP and UDP connections in the trace. With the detailed output from `tcptrace`, we were then able to collect the appropriate data needed about the trace (e.g., each flow's size in bytes, for smart sampling) – using this data, we wrote a Python program to perform the selected sampling method on the trace. Given the set of flows to be included in the sample, we again used `tcptrace` to filter the trace and determine the packets belonging to those flows; as before, we then constructed a new `pcap` trace containing the packets selected for the sample.

4.2.2 Method parameters

For this analysis, we implemented deterministic and uniform random packet sampling, stratified packet sampling, deterministic and uniform random flow sampling, smart sampling, and selective sampling. Their configuration parameters in our experiments are described in Table 6.

Table 6: Parameter configurations for sampling methods used in analyses.

Packet	**Deterministic**	$n = 1, 51, 111, ..., 961$
	Uniform random	$p = 1/1, 51, 111, ..., 961$
	Stratified (Deterministic)	field= *src* $n = 1, 51, 111, ..., 961$
		field= *srcprt* $n = 1, 51, 111, ..., 961$
Flow	**Deterministic**	$n = 1, 51, 111, ..., 961$
	Uniform random	$p = 1/1, 51, 111, ..., 961$
	Smart	$z = 3, 503, 1003, ..., 9503$
	Selective	$c = 0.03$ $n = 1, 100, 1000, 10000$ $z = 6, 8, 10, ..., 98$

We chose parameters so that trends in privacy or utility measurements could be best distinguished, with a secondary goal that the sample trace sizes are roughly comparable between sampling methods. While deterministic and random sampling can decrease the sample size (e.g., by increasing n or decreasing p) until the size of the sample is 0, the opportunistic sampling methods' sample size converges towards a non-zero value, making direct comparison difficult between individual traces produced by differing sampling methods.

4.3 Metric configuration

In this subsection, we briefly describe the specific implementations of the metrics described in Section 3.

4.3.1 k-anonymity

When calculating k-anonymity of the trace, we consider one feature to be "sensitive" and the others to be quasi-identifiers, leading to four cases:

Sensitive Field	Quasi-identifiers
src	$\{dst, srcprt, dstprt\}$
dst	$\{src, srcprt, dstprt\}$
srcprt	$\{src, dst, dstprt\}$
dstprt	$\{src, dst, srcprt\}$

4.3.2 L1-similarity

We defined as a network object an entity called *host*. Hosts are unique IP addresses that may appear in the *src* or *dst* fields of a record in a packet trace; in other words, a host is either a sender or a receiver of data during the time period in which the trace was collected.

Formally, each host object A consists of records from $\mathcal{T}$ with values on features $\mathcal{F} = \{src, dst, srcprt, dstprt\}$, such that $A = t \in \mathcal{T}$: $(t_{src} = h \vee t_{dst} = h)$, where h is the IP address for A. To measure the similarity between two objects, we first calculate the distributions of distinct values on each feature for the records contained within the host object.

We measured the L1-similarity between an unanonymized host $A_r \in A_R$ and an anonymized host $A_s \in A_S$: for each feature $f \in F$, we compared the distribution of distinct values in A_{r_f} and A_{s_f}, as in Equation 3. Due to trace sanitization, these distinct values cannot be compared directly (e.g., IP address "67.23.134.45" anonymizes to "123.145.167.189") and we were thus forced to indirectly compare the distributions. We did this in our analysis by considering the most frequently occurring value in A_{s_f} to represent the most frequently occurring value in A_{r_f}, the second most frequently values in A_{s_f} and to A_{r_f} to be the same, and so on. As described in Section 3.1.2, the maximum value for $sim(A_{r_f}, A_{s_f})$ is 2, which indicates that the two objects have identical distributions on the feature f, and there is a strong possibility that $A_{r_f} = A_{s_f}$.

4.3.3 Entropy anonymity degree

As mentioned in Section 3.1.3, the probability mass function Pr required to calculate EAD is not specified. Therefore, in our analysis, we were required to define our own mass function.

The mass function we implemented for our experiments was previously used by Coull et al. [8] as a portion of their analysis to measure privacy over several iterations of adversary deanonymization. The mass function is defined as follows:

$$Pr(A_{r_f} = A_{s_f}) = \frac{sim(A_{r_f}, A_{s_f})}{\sum_{a \in A_R \cup A_S} sim(a_{r_f}, a_{s,f})}. \qquad (6)$$

This definition of the probability mass function assigns the highest masses where similarity between two objects is relatively high compared to the total similarity measured between all pairs of objects. For example, let $sim(A_{r_x}, A_{s_y}) = 1.8$; if the average similarity between a raw object and sanitized object is 0.5, $Pr(A_{r_x}, A_{s_y})$ would be much higher than if the average similarity were 1.7.

We then calculated Shannon's entropy (Equation 4), on the set of values in Pr on each feature to calculate the EAD at the field level. At the host level, the EAD is the sum of the field-level entropies; to calculate the normalized EAD, whose value is between 0 and 1, we divided the field-level EAD and host-level EAD by $\log_2 |A_S|$ and $|F| \log_2 |A_S|$, respectively, as in Equation 5.

4.3.4 Snort

To perform our experiments, we obtained the core Snort Engine and installed it on our test machine. Because the Snort intrusion-detection system depends on an updated set of rules to test traces against, we updated our rules to the set current as of 2011-04-27.

To collect alert logs, we then ran Snort with these rules on each input sampled trace, filtering for alerts on IP packets only but otherwise run on default settings. (Alerts generated by packets with protocols other than TCP or UDP were present in our log files; we discuss this effect when examining the results of our analyses in Section 4.4.3). After collection, we used a script to further parse and process these logs to obtain counts of each unique alert type for our sampled traces.

4.4 Experiments

In this subsection, we present the results of our experiments on the sample traces generated as described in Section 4.2.

4.4.1 L1-similarity and EAD

The potential resource benefits of sampling are clear upon examination of our chosen algorithm to calculate L1-similarity between raw and anonymized objects. Because we compare each raw object to each anonymized object, use of this metric requires $O(n^2)$ time and space; reducing the input size, even by a small amount, would therefore result in tangible performance gains.

Due to our implementation of the L1-similarity metric as the core for our EAD metric, we were able to directly examine whether similarity values are altered by the sampling process, and if so, whether entropy values based on that metric were also affected.

Calculating these metrics on our original unsampled trace served as a baseline by which we could compare the accuracy of the same calculations when run on sampled raw and sanitized traces; these baseline values are located in Table 7. Note that in this and all other L1-similarity calculations in our experiments, the value listed represents the average of the L1-similarity values across all sanitized objects.

Unique hosts	1679			
L1-similarity	*src*	*dst*	*srcprt*	*dstprt*
	1.5314	1.4925	0.9112	1.4503
	host: 1.3463			
EAD	*src*	*dst*	*srcprt*	*dstprt*
	0.99530	0.99513	0.97028	0.99486
	host: 0.98889			

Table 7: L1-similarity and EAD for unsampled sanitized trace.

These results indicate that, over the 1,679 unique hosts located in the trace file, the average similarity of hosts across the *src* and *dst* objects are roughly the same and that the distributions of these similarities are also comparable; the entropy values for both *src* and *dst* are approximately 0.995 in the unsampled trace. The *srcprt* feature, however, features notably less similarity between objects, raising the possibility that a host may be more sensitive to identification by using external information about the *srcprt* field – the lower EAD for the field seems to confirm this. We may, therefore, consider the *srcprt* to be the "least private" field when measuring the anonymity of hosts in our unsampled trace.

Figure 3 presents the results of our calculations of the L1-similarity of each feature $f \in \{src, dst, srcprt, dstprt\}$ and the *host* object for all samples described in Table 6. Larger versions of these plots for each feature may be found in Appendix A.

We chose the number of distinct hosts as our x-axis in these plots because they reveal a clear trend in similarity values for all features as the sampled trace size decreases; L1-similarity measurements become artificially high even at low sampling rates and then begin to decrease again as the sample trace size approaches 0. None of the sampling methods tested here alters or mitigates this trend towards an overestimation of L1-similarity values. Were there an end-user who attempted trace sampling in conjunction with an L1-similarity metric, that user could be led to believe, in error, that the anonymized trace is more secure than it actually is.

Relationships in similarity between features are also distorted (or lost entirely) in every sampling method tested. For example, the similarity of field *src* is slightly greater than the similarity of field *dst* in our non-sampled trace, but as the sample size decreases, this situation quickly reverses itself. Distortions such as this make developing an effective sanitization strategy more difficult, as they can alter the perception of which fields are most insecure.

The results in Figure 3 are likely a result of the bias of the sampling methods to select packets from larger "elephant" flows and more prominent hosts in the trace – even selective sampling fails to correct this bias; the resulting decreased diversity in feature distributions would cause the similarity values to increase. As sample size approaches zero, similarity values are much more varied and scattered – especially when using flow-sampling methods. This variation is due to the fact that as the sampled trace shrinks, the removal of additional packets or flows could just as easily increase the similarity (as the sample becomes relatively more homogeneous) or decrease it (if the sample becomes less saturated with dominant "elephant"-sized flows).

Figure 4 shows the calculation of entropy anonymity degree using the same input traces and our similarity metric from Figure 3 at its core. For all fields except *srcprt* (*host* is based on an average of values including *srcprt*) the behavior of anonymity calculations is similar: as the sample size (measured in distinct network hosts) shrinks, entropy calculations remain at or slightly above the accurate value. When the sample size decreases beneath approximately 800 distinct hosts, or about half the unsampled trace size, entropy begins to decrease across the board, with this decrease accelerating as the sample size approaches zero.

Entropy decreases when fewer network hosts are present in the sample because the distributions between unanonymized and anonymized objects are more easily placed in a one-to-one mapping than may have occurred in a larger sample set; each sanitized object is more likely to have a unique distribution of features that can be matched with an equivalent unsanitized object, whereas in a larger trace, there may be several unanonymized objects with a similar distribution of features.

These trends in entropy do not seem to apply to the *srcprt* field, however, and the most likely reason is the large distortion in similarity values for this field. As seen in Figure 3, the rate of increase in L1-similarity values on the *srcprt* feature as sample size shrinks is greater than that of any other feature. This result is likely due to an increased diversity in source port values relative to the other fields, a theory confirmed in this case by examining some typical deterministic packet samples, with increasing values of n (Table 8).

While the number of distinct source IPs and destination IPs shrinks by about a factor of 4 between these three samples, the distinct count of source ports decreases by a factor of 14 – this indicates that the number of records associated with each source port is small. As sample size decreases, the feature is dominated by a few more common ports and many lesser-used ports are quickly filtered out even at low sampling rates. This in turn causes the L1-similarity mea-

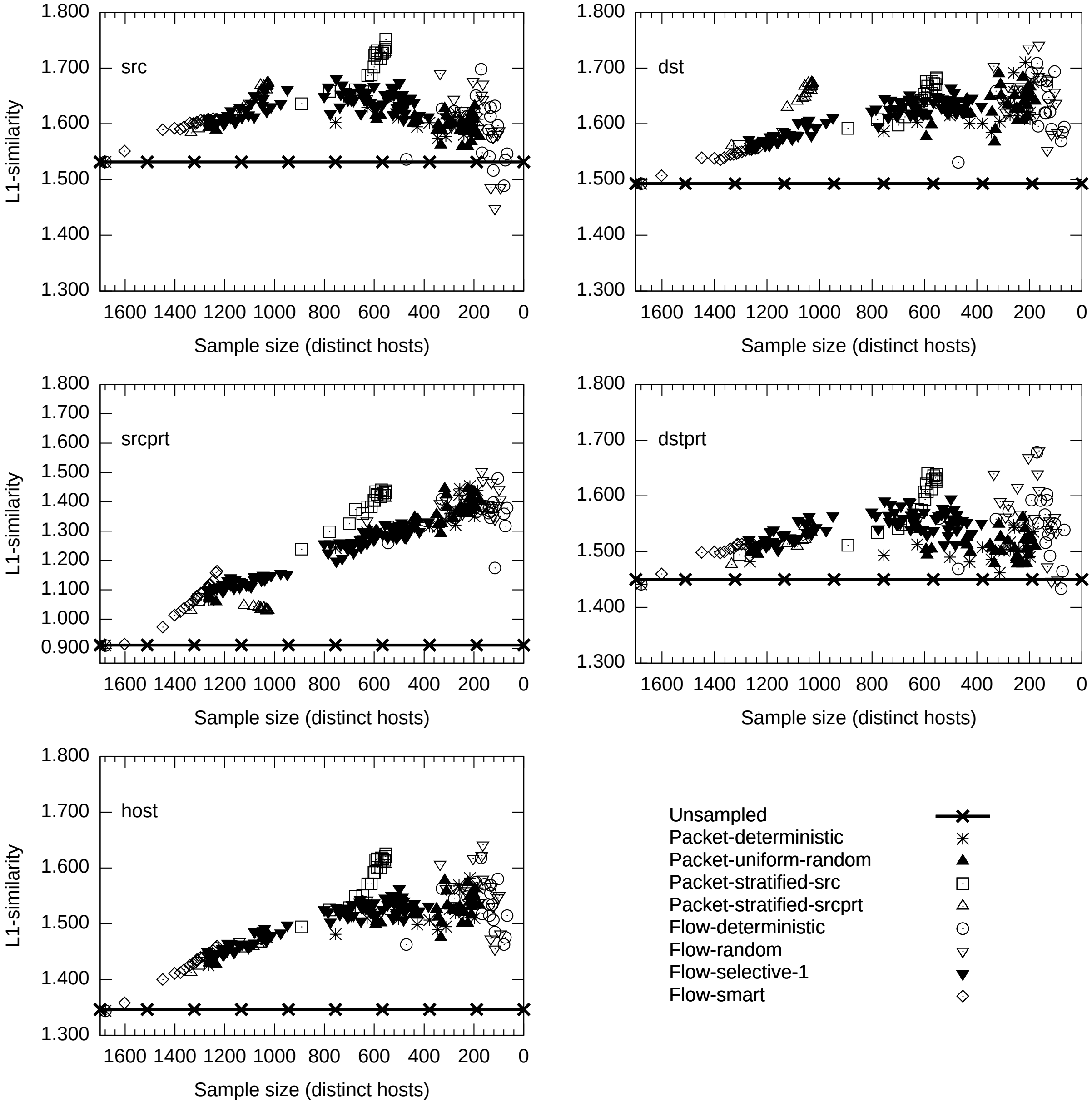

Figure 3: Relationship between number of distinct hosts in a sampled trace and the corresponding L1-similarity of each feature (including the *host* object). Note the different y-axis scale for the *srcprt* plot.

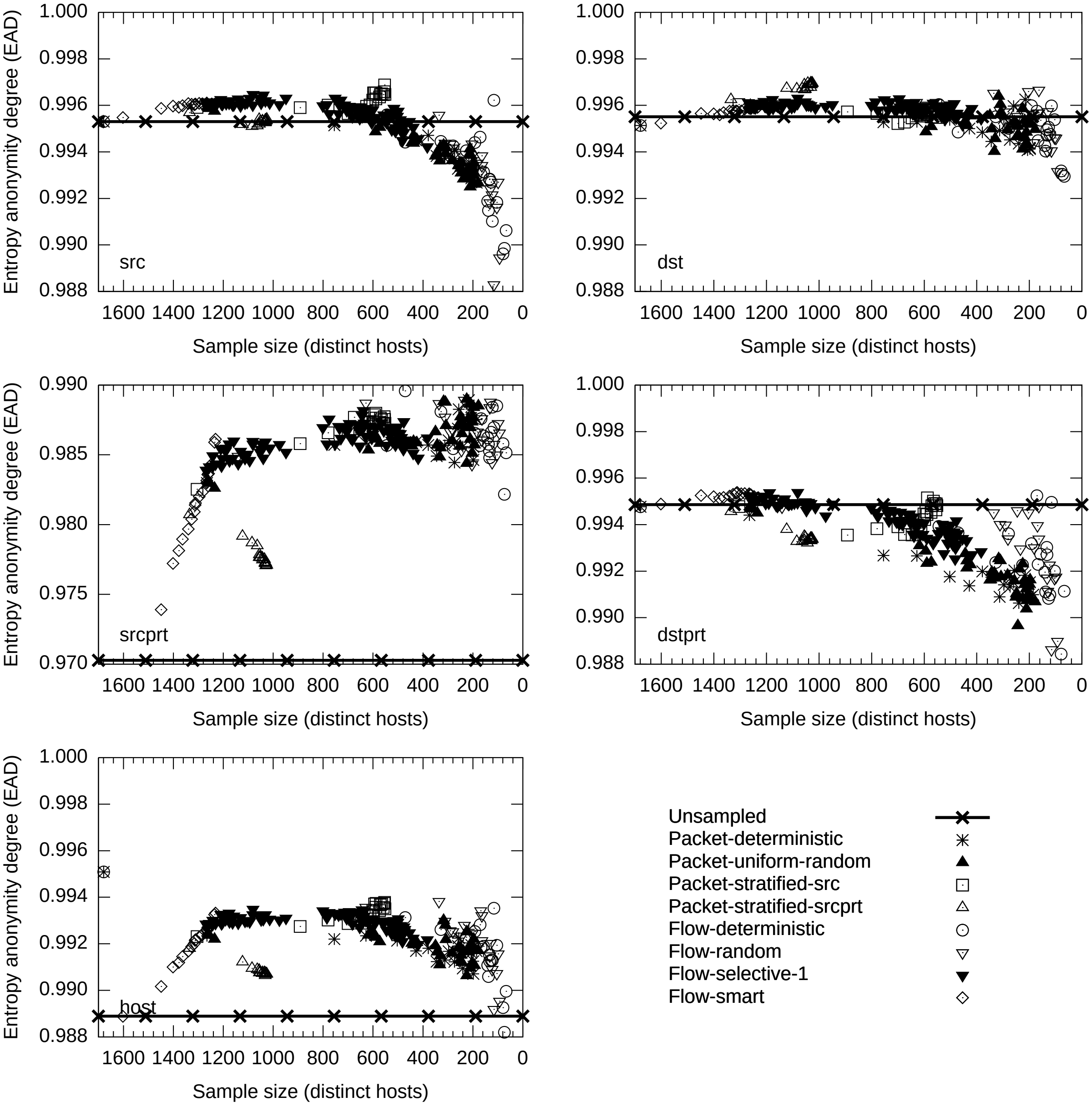

Figure 4: Relationship between number of distinct hosts in a sampled trace and the corresponding EAD of each feature (including the *host* object). Note the different y-axis scale for the *srcprt* plot.

n	*src*	*dst*	*srcprt*	*dstprt*
11	1163	474	7629	2708
111	566	221	1718	1009
511	219	117	538	389

Table 8: Distinct values for each feature in deterministic packet samples.

surements for *srcprt* on the sample to increase and flatten relative to each other, inaccurately increasing entropy at the same time.

4.4.2 k-anonymity

The effects of sampling on k-anonymity are less clear, however, due to the fact that k for the sanitized trace is equal to 1 on all fields. This is to be expected, as our sanitization configuration did not involve the truncation of any data. By not truncating data, we do not make any alterations to the equivalence classes present in the unsanitized trace, and the presence of just one quasi-identifier with one instance in the dataset is enough to make $k = 1$ for the entire field.

The sole use of k-anonymity as a privacy metric is difficult, however, due to the inability to identify sensitive attributes with certainty. In our experiments, we attempted to judge each field as sensitive, using the other three tested fields as quasi-identifiers, as described in Section 4.3.1, with mixed results. While an increase in k-anonymity in one of our samples, such that $k > 1$, would indicate an inaccurate value that could give a researcher a false sense of security, we did not see this result in any of the sampled traces on any field.

Because k remains constant at 1 across all fields of all samples, we instead examined trends in the average size of equivalence class for each feature present in our sampled traces (Figure 5). Class size drops rapidly and non-linearly as the host count decreases, indicating that the largest equivalence classes quickly disappear from samples, as new smaller equivalence classes are created, however, rather than destroyed by the sampling process; this causes k to remain constant at 1. The type of feature appears to affect equivalence class size more than the distance feature count; equivalence set size for both fields *src* and *dst* decreases to approximately 10 at sample size 1250 and approximately 2.5 at the smallest sample sizes, despite there being over twice as many unique sources as destinations in the trace (Table 8).

4.4.3 Snort alerts

Investigation of the effects of sampling on intrusion and anomaly detection was similarly difficult because of the tendency of sampling to reduce substantially the number of alerts that Snort detects, which adversely affects the ability to accurately predict the utility of the whole trace. A summary of the 2068 alerts triggered by the unsampled trace is contained in Table 9.

Fully half of our sample traces (129 of 257) failed to trigger any Snort alerts, while those that did trigger alerts only triggered a small fraction of the utility total on its own; the results are plotted in Figure 6. While the overall utility of the sanitized trace was measured by Snort to be 568.4, this figure includes ICMP packets that were filtered out in all of the sampling processes – the sans-ICMP utility measure of the unsampled anonymized trace was thus 371.6.

Only when using smart sampling, whose sample sizes all contained greater than 1200 unique hosts, were utility measurements somewhat reliable and followed a basic trend, decreasing sharply in a somewhat-linear fashion until reaching the smallest smart sample ($z = 9503$, 1231 unique hosts, utility $= 0.46$). When sample size shrinks further, alert generation becomes hit-or-miss regardless of sampling method – because different random samples can behave differently at the same parameter settings – with no sample registering a utility of greater than 11.4 (which occurred with deterministic flow sampling, $n = 211$). Because of the unreliability (or total lack) of alert counts, it is difficult to conclude that any particular sampling method outperforms another. With the possible exception of smart sampling, none of the sampling methods tested would have allowed an accurate measurement of utility; all would have severely underestimated the utility of the trace, as defined by our metric.

Table 9: Snort alerts triggered by the unsampled trace.

Alert type	Frequency
ICMP unreachable host	1
ICMP unreachable port	206
ICMP ping	285
TCP reset	1012
TCP window close	564
total	2068
utility (Section 3.2.1)	568.4
utility (without ICMP included)	371.6

4.5 Experiment limitations

The tests and measurements that we performed were conducted on a single, relatively small (200MB) trace, which precludes us from making broad-reaching conclusions about the applicability of trace sampling to measure the privacy-utility tradeoff in most circumstances or use cases. Because only TCP and UDP headers were examined in our experiments, there may be other types of records or protocols that are more amenable to sampling. Finally, our experiments themselves may not provide a comprehensive summary of sampling methods to measure privacy and utility, as we have focused on a small number of distinct metrics and did not test any adaptive sampling methods, which may or may not outperform the conventional methods tested here.

5. DISCUSSION

For the trace used in performing the above analyses, it is apparent that none of the sampling methods tested would have yielded accurate information about the privacy and utility of the sanitized trace that could aid a researcher in the task of releasing a network trace to the community. This result does not rule out the ability of trace sampling to allow an accurate estimation of the privacy-utility tradeoff, as the trace used for these experiments was relatively small and the sampling methods used here may differ in behavior on larger or more diverse traces.

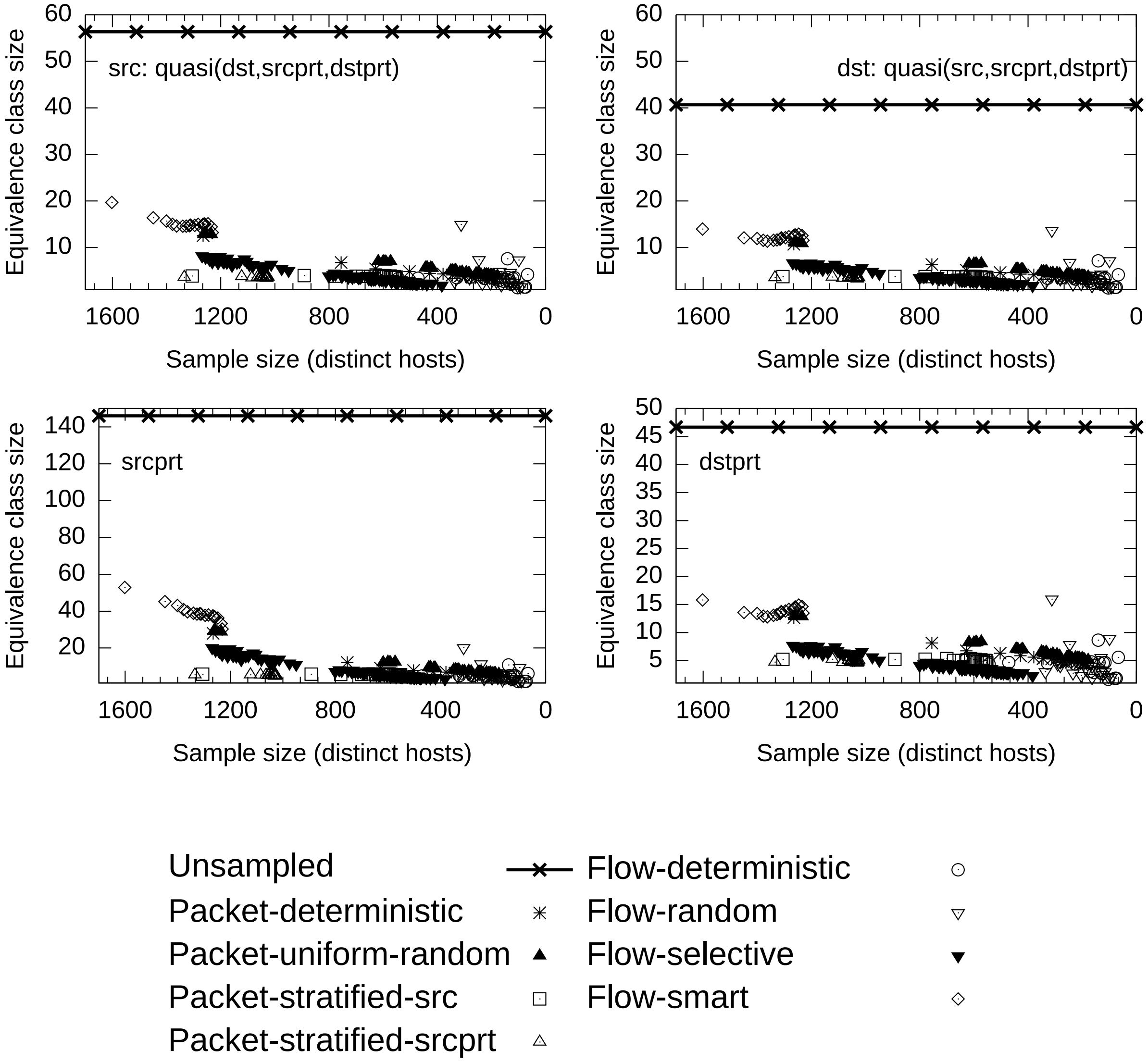

Figure 5: Relationship between number of distinct hosts in the sample size and average quasi-identifier equivalence class size per field. Note the different y-axis scales.

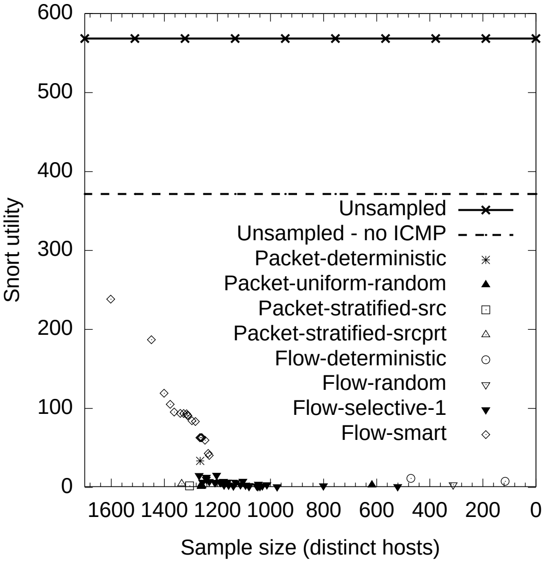

Figure 6: Relationship between sample size (in distinct hosts) and the utility measured by Snort alerts.

But these results suggest that either additional research into new sampling algorithms is needed, or that it may be inappropriate to pursuit a one-size-fits-all approach to sampling traces when measuring the privacy-utility tradeoff without a more intimate knowledge of the metrics being used to measure that tradeoff.

Metrics relying on entropy, such as entropy anonymity degree, seemed more able to absorb the changes in successively-smaller sampled traces and produce somewhat accurate measurements despite the noted changes in the L1-similarity metric values that EAD was itself based on. Attempts to distill EAD values for individual features down to a single per-host privacy value could be misleading, however, as the behavior of one field (*srcprt*) largely defines the EAD for the entire *host* object in our experiments.

Because flow-based sampling did not definitively outperform packet-based sampling in any of our experiments, and given its increased resource requirements, it would be difficult to recommend its use over packet-based sampling based on these results. Were we to recommend a sampling strategy for the trace tested in our experiments, a potential sampling strategy to yield minor performance gains without significant loss of accuracy could include:

1. limited deterministic (sample rate ≈ 2) packet sampling to perform L1-similarity and EAD tests at a feature level, as they are the most performance-intense metrics and least susceptible to disturbance by sampling,

2. measuring k-anonymity without sampling, as the results (while somewhat unhelpful in this case) are relatively easy to calculate and equivalence class sizes degrade quickly with even limited sampling, and

3. measuring utility using Snort without sampling, as any sampling can seriously affect the number of alerts triggered (and thus the inferred utility).

6. RELATED WORK AND FUTURE DIRECTIONS

Related work on packet sampling and its effects on network traffic characterization was conducted by Claffy et al. [7] and Hernandez et al. [15], the latter introducing adaptive methods to equal or outperform non-adaptive packet sampling while reducing hardware or storage requirements to collect the relevant traces.

Brauckhoff et al. [2] specifically examined the impact of packet sampling on flow statistics and the ability to detect the Blaster worm by examining changes in entropy. Mai et al. [23, 24] demonstrated the ability to detect a range of anomalies using a number of packet and flow-sampling methods, and Androulidakis et al. [1] used flow-sampling methods to specifically target anomalies dependent on changes in entropy across fields.

Kelly et al. [18] gathered a list of existing privacy metrics, including metrics based on information entropy described by Diaz et al. [11]. Coull et al. [8, 9] examined the sensitive information that can be inferred from traces, methods to measure privacy using a combination of entropy and L1-similarity of feature distributions, and described an iterative strategy to simulate an adversary's attempt to deanonymize data using externally available information.

Lakkaraju and Slagell [21] examined the use of Snort to measure utility of a network trace. Research from Pang et al. [25] and Slagell and Yurcik [30] discuss the inherent tradeoff between privacy and utility when sanitizing network traces. Finally, Fazio et al. [14] outlined a framework to streamline the process of sanitizing traces for researchers looking to best address this tradeoff.

To the best of our knowledge, however, there is no existing work that measures the effect of packet sampling (or flow sampling) on the simultaneous measurement of privacy and utility of a network trace, the so-called privacy-utility tradeoff. Additionally, this work treats trace sampling as one of a series of steps to best utilize the resources of a researcher seeking to anonymize and release a network trace with the specific goal of allowing colleagues to conduct useful research on the anonymized traces.

Future directions in work related to sampling and its effect on measuring the privacy-utility tradeoff in network traces include research towards more generally effective sampling methods, and their specific effects on privacy and utility metrics, and the ability to accurately "correct" measurements using sampled traces to their unsampled equivalents. Finally, future research could also examine multi-stage sampling methods, or processes that combine one or more distinct sampling methods based on context, and their effects on privacy and utility compared to using a single globally-applicable sampling method.

7. SUMMARY

In this paper, we examine several simple non-adaptive sampling methods to discover their effects on measurement of the privacy-utility tradeoff when sanitizing network traces prior to their sharing or publication. The results will be applied to the recently-introduced NetSANI framework for trace sanitization that seeks to ease the burden on researchers to adequately sanitize their network traces (while preserving useful data) and share them with their colleagues. While packet and flow sampling have been used and analyzed in the past for such applications as anomaly detection and traffic measurement, little or no research has been done to sampling's direct effect on measuring both privacy and utility at the same time.

After sanitizing a small sample trace collected from the Dartmouth College wireless network, we tested the relative accuracy of a variety of previously-implemented packet and flow-sampling methods when measuring privacy with microdata, similarity, and entropy-based metrics, and on utility by use of the Snort intrusion-detection system. The results of this analysis led us to conclude that, for the test trace, no single sampling method we examined was able to accurately measure privacy and utility, and that some sampling methods can produce grossly inaccurate estimates of those values. We also found it unlikely that a single "universal" sampling method could be used to perform this analysis accurately on a trace of any size. We were unable to draw conclusions on the use of packet versus flow sampling in these instances, nor were we able to gauge the accuracy of these experiments on larger traces.

Acknowledgements

I would like to acknowledge and thank Professor David Kotz for his assistance, guidance, and patience throughout this project, from its conception and brainstorming until final proofreading. Additionally, I would like to thank Keren Tan for input and technical assistance aimed at improving my analysis and finished product; Chris McDonald for his insight and suggestions; Anna Shubina for locating the traces used in the experiments; and the developers of the tools that aided in the execution of the experiments and presentation of this paper.

8. REFERENCES

[1] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou. Network anomaly detection and classification via opportunistic sampling. *The Magazine of Global Internetworking*, 23:6–12, January 2009. DOI 10.1109/MNET.2009.4804318.

[2] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 159–164. ACM, 2006. DOI 10.1145/1177080.1177101.

[3] N. Brownlee, C. Mills, and G. Ruth, Internet Engineering Task Force. Traffic Flow Measurement: Architecture. RFC 2722 (Informational), Oct. 1999. Online at http://www.ietf.org/rfc/rfc2722.txt.

[4] M. Burkhart, D. Brauckhoff, M. May, and E. Boschi. The risk-utility tradeoff for IP address truncation. In *Proceedings of the ACM Workshop on Network Data Anonymization (NDA)*, pages 23–30. ACM, October 2008. DOI 10.1145/1456441.1456452.

[5] Y. Chabchoub, C. Fricker, F. Guillemin, and P. Robert. Deterministic versus probabilistic packet sampling in the internet. In *Proceedings of the 20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks (ITC)*, pages 678–689, Berlin, Heidelberg, 2007. Springer-Verlag. DOI 10.1007/978-3-540-72990-7_60.

[6] V. Ciriani, S. Capitani di Vimercati, S. Foresti, and P. Samarati. *k*-anonymity. In T. Yu and S. Jajodia, editors, *Secure Data Management in Decentralized Systems*, volume 33 of *Advances in Information Security*, pages 323–353. Springer US, 2007. DOI 10.1007/978-0-387-27696-0_10.

[7] K. C. Claffy, G. C. Polyzos, and H.-W. Braun. Application of sampling methodologies to network traffic characterization. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 194–203. ACM, 1993. DOI 10.1145/166237.166256.

[8] S. Coull, C. Wright, F. Monrose, A. Keromytis, and M. Reiter. Taming the Devil: Techniques for evaluating anonymized network data. In *Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS)*, Feb. 2008. Online at http://www.cs.unc.edu/~reiter/papers/2008/NDSS.pdf.

[9] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter. Playing Devil's advocate: Inferring sensitive information from anonymized network traces. In *Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS)*. IEEE Press, Feb. 2007. Online at http://www.isoc.org/isoc/conferences/ndss/07/papers/playing_devils_advocate.pdf.

[10] Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD). Online at http://www.crawdad.org/, visited 2010.

[11] C. Diaz, J. Claessens, S. Seys, and B. Preneel. Information theory and anonymity. In B. Macq and J.-J. Quisquater, editors, *Werkgemeenschap voor Informatie en Communicatietheorie*, pages 179–186, 2002.

[12] N. Duffield, C. Lund, and M. Thorup. Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 159–171. ACM, 2002. DOI 10.1145/637201.637225.

[13] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 323–336. ACM, 2002. DOI 10.1145/633025.633056.

[14] P. Fazio, K. Tan, J. Yeo, and D. Kotz. Short paper: The NetSANI framework for analysis and fine-tuning of network trace sanitization. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*. ACM Press, June 2011. To appear.

[15] E. A. Hernandez, M. C. Chidester, and A. D. George. Adaptive sampling for network management. *Journal of Network and Systems Management*, 9:409–434, December 2001. DOI 10.1023/A:1012980307500.

[16] N. Hohn and D. Veitch. Inverting sampled traffic. *ACM/IEEE Transactions on Networking (TON)*, 14:68–80, February 2006. DOI 10.1109/TNET.2005.863456.

[17] W. Jun-feng, Y. Jian-hua, Z. Hong-xia, X. Gao-gang,

Z. Ming-tian, and V. No. Adaptive sampling methodology in network measurements. *Journal of Software*, 15:1227–1236, 2008. Online at http://www.jos.org.cn/1000-9825/15/1227.pdf.

[18] D. J. Kelly, R. A. Raines, M. R. Grimaila, R. O. Baldwin, and B. E. Mullins. A survey of state-of-the-art in anonymity metrics. In *Proceedings of the ACM Workshop on Network Data Anonymization (NDA)*, pages 31–40. ACM Press, 2008. DOI 10.1145/1456441.1456453.

[19] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, P. Trimintzios, and M. Fukarakis. CRAWDAD tool tools/sanitize/generic/anontool (v. 2006-09-26). Downloaded from http://crawdad.cs.dartmouth.edu/tools/sanitize/generic/AnonTool, Sept. 2006.

[20] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 217–228. ACM, 2005. DOI 10.1145/1080091.1080118.

[21] K. Lakkaraju and A. Slagell. Evaluating the utility of anonymized network traces for intrusion detection. In *Proceedings of the International Conference on Security and Privacy in Communication Networks (SecureComm)*, pages 1–8. ACM, 2008. DOI 10.1145/1460877.1460899.

[22] tcpdump/libpcap public repository. Online at http://www.tcpdump.org/, visited 2011.

[23] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of the Internet Measurement Conference (IMC)*, pages 165–176, 2006. DOI 10.1145/1177080.1177102.

[24] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye. Impact of packet sampling on portscan detection. *IEEE Journal on Selected Areas in Communications*, 24(12):2285–2298, December 2006. DOI 10.1109/JSAC.2006.884027.

[25] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29–38, 2006. DOI 10.1145/1111322.1111330.

[26] A. Patcha and J.-M. Park. An adaptive sampling algorithm with applications to denial-of-service attack detection. In *Proceedings of the 15th International Conference on Computer Communications and Networks (ICCCN)*, pages 11–16, October 2006. DOI 10.1109/ICCCN.2006.286238.

[27] Python/C API Reference Manual. Online at http://docs.python.org/c-api/, visited 2010.

[28] M. Roesch. Snort: A free, open source network intrusion detection and prevention system, 1998. Version 2.9.0.5, Online at http://www.snort.org/.

[29] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the International Symposium on Privacy Enhancing Technologies (PET)*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, 2002. DOI 10.1007/3-540-36467-6_4.

[30] A. Slagell and W. Yurcik. Sharing computer network logs for security and privacy: a motivation for new methodologies of anonymization. In *Proceedings of the International Conference on Security and Privacy for Emerging Areas in Communication Networks*, pages 80–89, 2005. DOI 10.1109/SECCMW.2005.1588299.

[31] tcptrace - Official Homepage. Online at http://www.tcptrace.org/, visited 2011.

[32] J. Zhang, X. Niu, and J. Wu. A space-efficient fair packet sampling algorithm. In *Proceedings of the 11th Asia-Pacific Symposium on Network Operations and Management: Challenges for Next Generation Network Operations and Service Management (APNOMS)*, pages 246–255. Springer-Verlag, 2008. DOI 10.1007/978-3-540-88623-5_25.

APPENDIX

A. ADDITIONAL PLOTS

This section contains larger versions of the plots in Section 4.4.1, to allow more detailed analysis of the individual sampling methods.

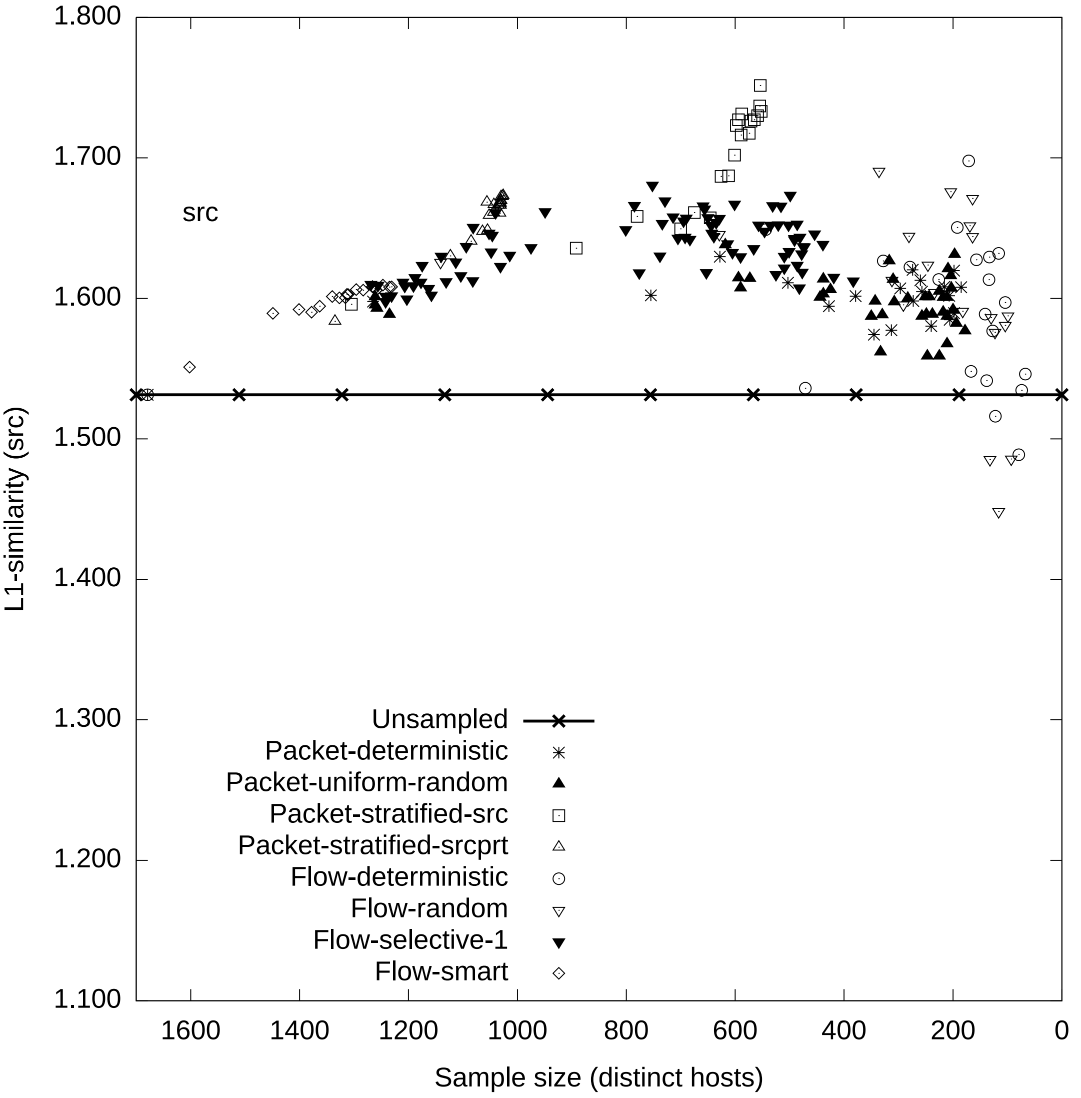

Figure 7: Relationship between sample size, in number of distinct hosts, and the measured L1-similarity on the *src* field.

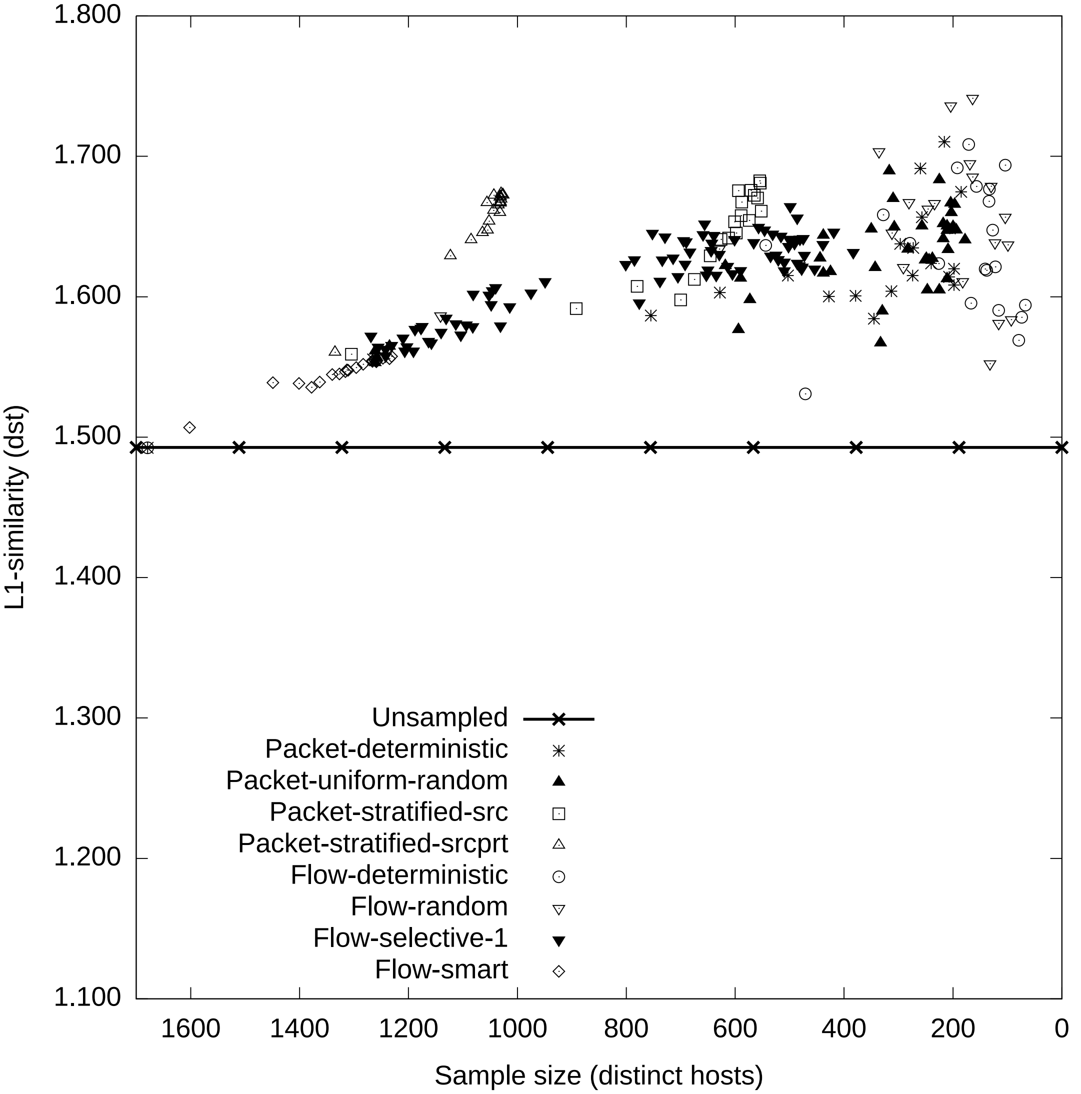

Figure 8: Relationship between sample size, in number of distinct hosts, and the measured L1-similarity on the dst field.

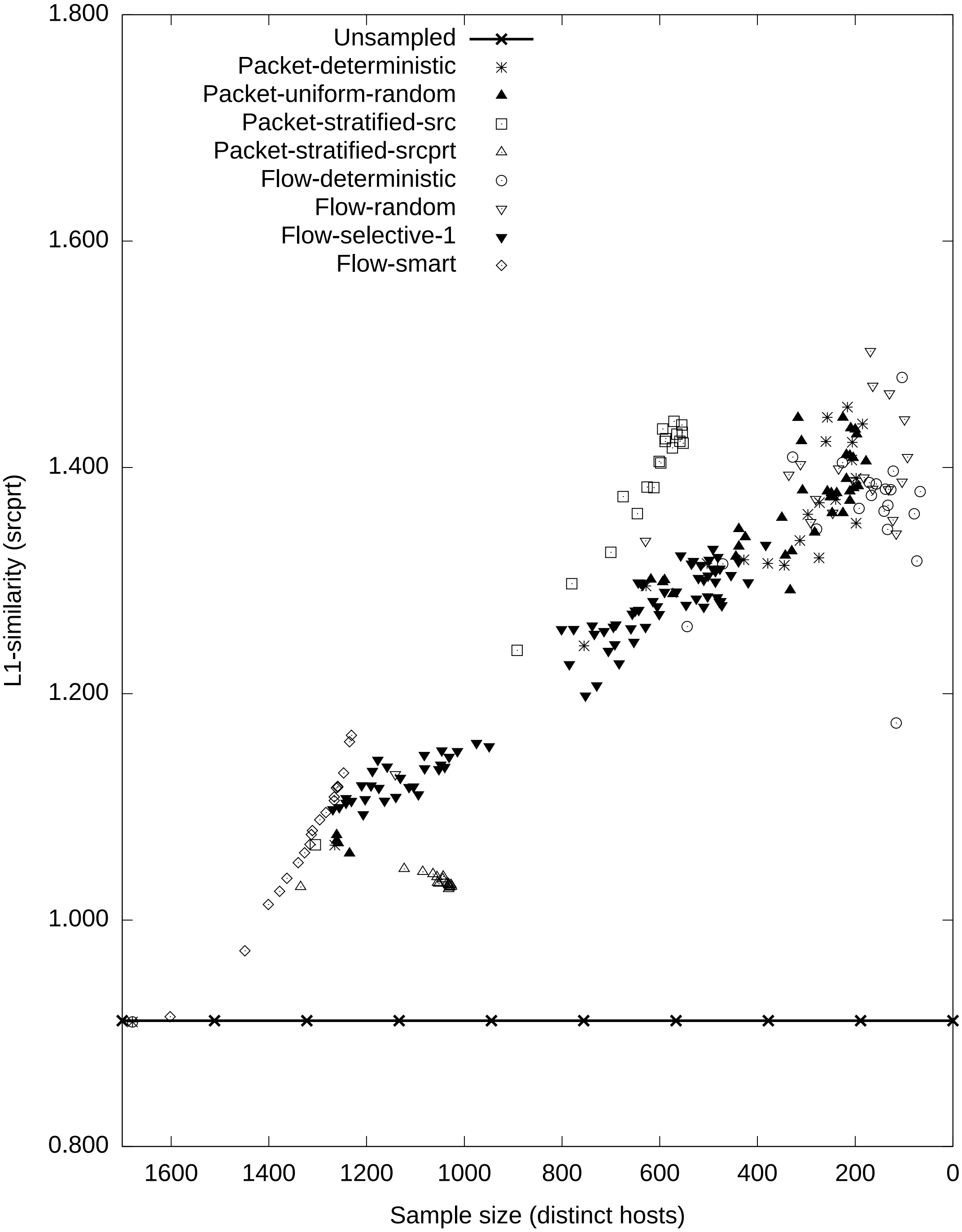

Figure 9: Relationship between sample size, in number of distinct hosts, and the measured L1-similarity on the *srcprt* field.

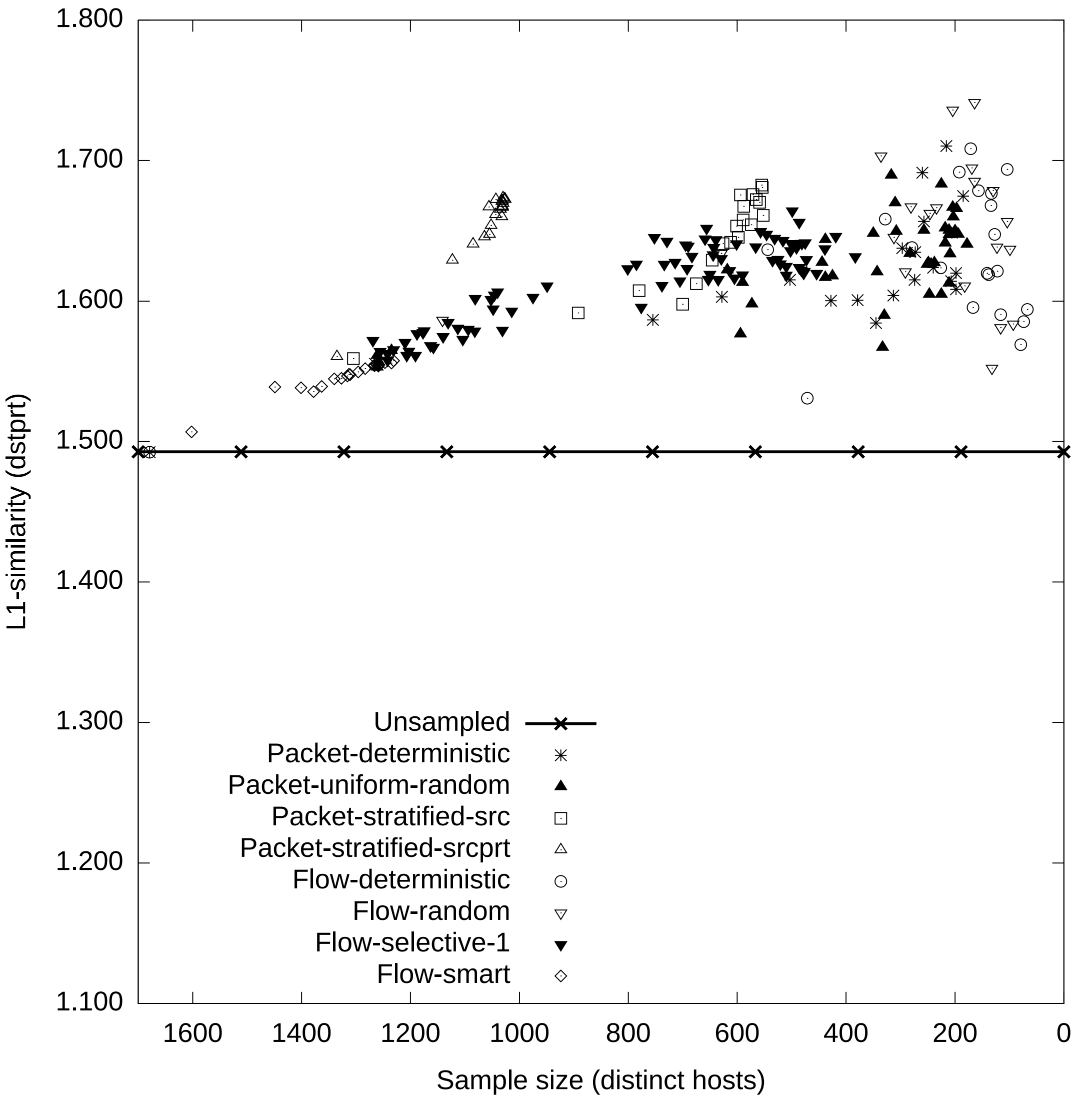

Figure 10: Relationship between sample size, in number of distinct hosts, and the measured L1-similarity on the *dstprt* field.

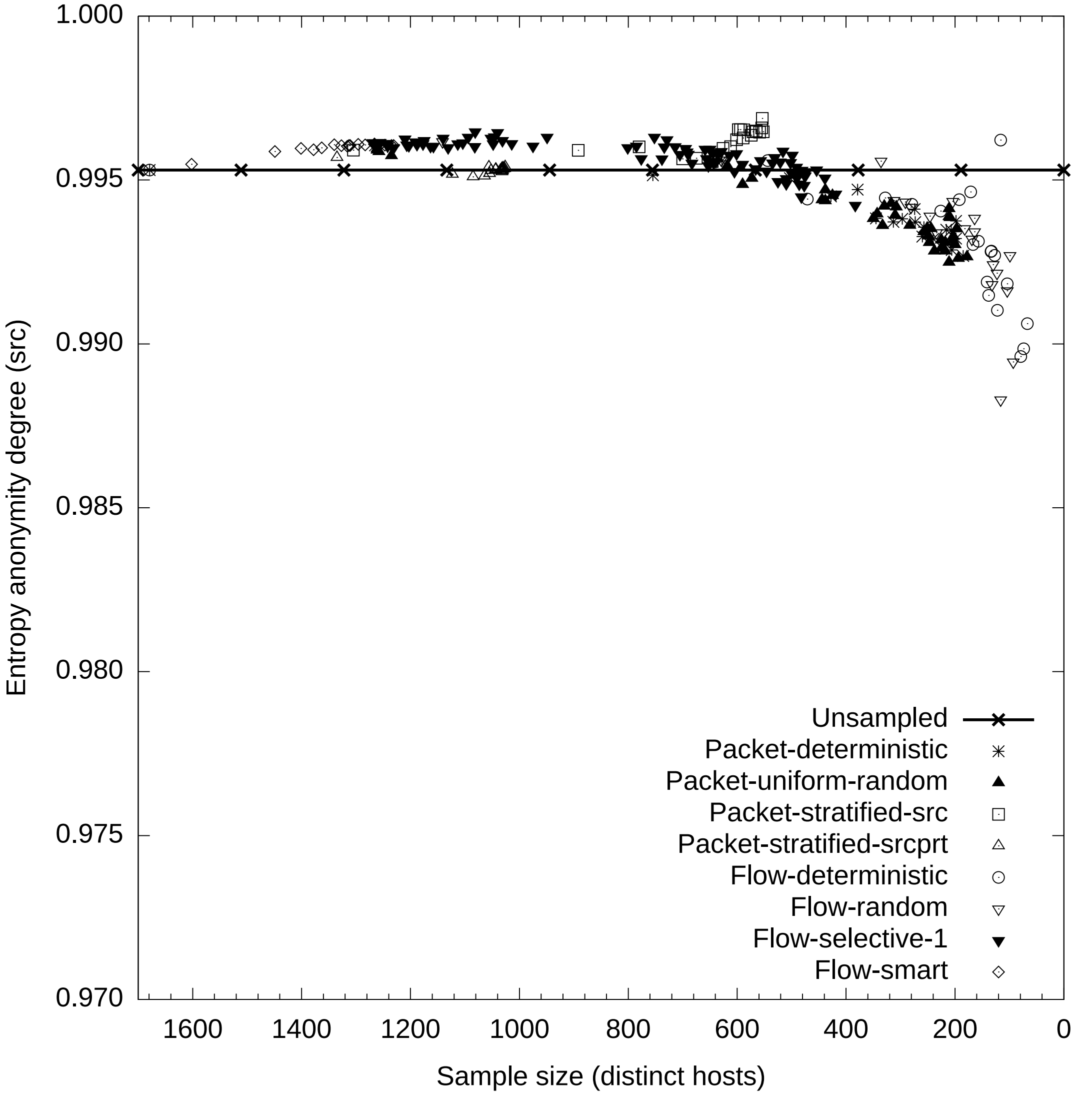

Figure 11: Relationship between sample size, in number of distinct hosts, and the measured entropy anonymity degree on the *src* field.

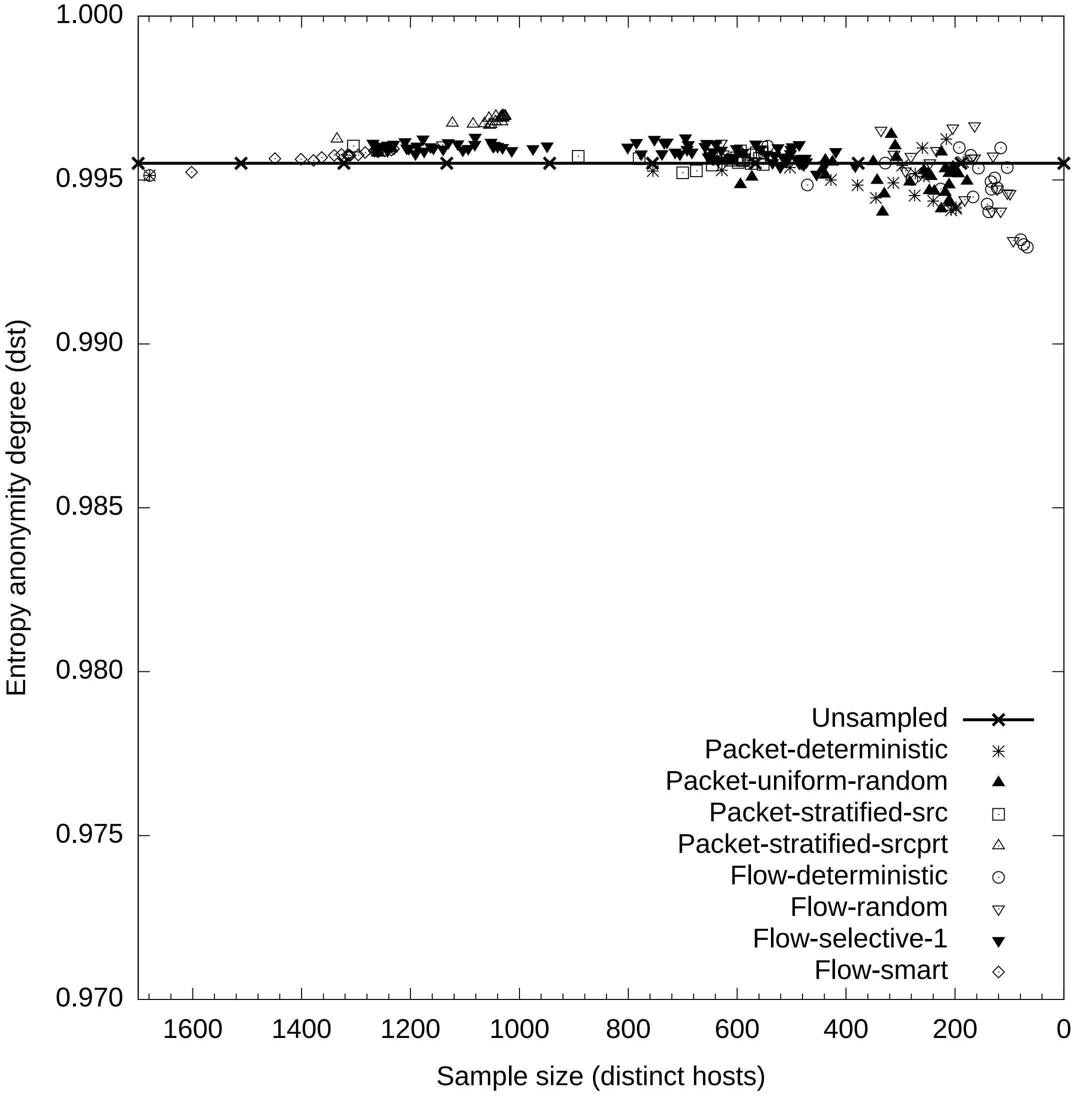

Figure 12: Relationship between sample size, in number of distinct hosts, and the measured entropy anonymity degree on the *dst* field.

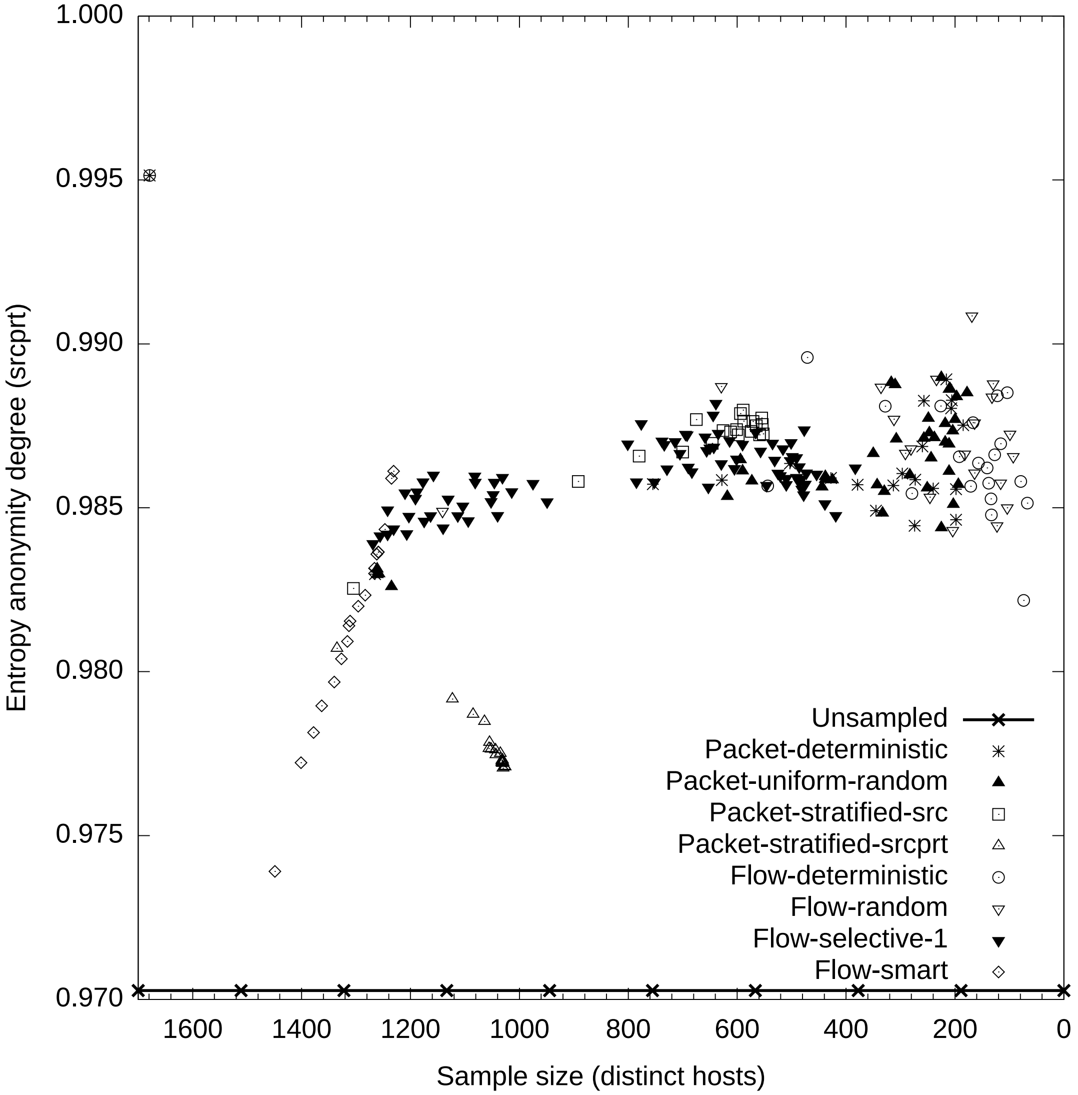

Figure 13: Relationship between sample size, in number of distinct hosts, and the measured entropy anonymity degree on the *srcprt* field.

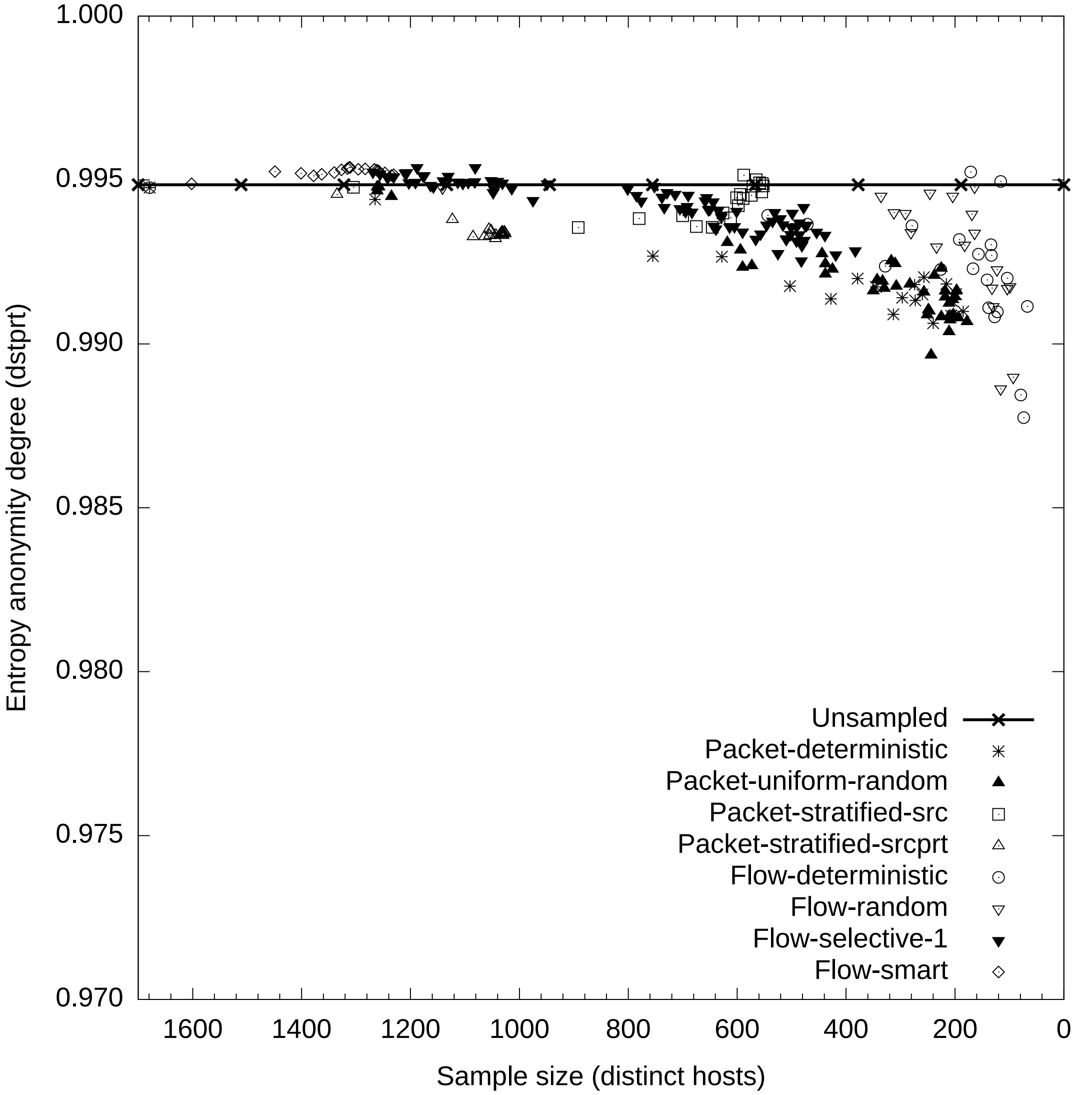

Figure 14: Relationship between sample size, in number of distinct hosts, and the measured entropy anonymity degree on the *dstprt* field.