Internet Draft
Document: <draft-ietf-psamp-sample-tech-11.txt>
Intended status: Proposed Standard
Expires: December 2008

T. Zseby
Fraunhofer FOKUS
M. Molina
DANTE
N. Duffield
AT&T Labs-Research
S. Niccolini
NEC Europe Ltd.
F. Raspall
EPSC-UPC
July 9, 2008

Sampling and Filtering Techniques for IP Packet Selection

Status of this Memo

Copyright Notice

Abstract

This document describes Sampling and Filtering techniques for IP packet selection. It provides a categorization of schemes and defines what parameters are needed to describe the most common selection schemes. Furthermore it shows how techniques can be combined to build more elaborate packet Selectors. The document provides the basis for the definition of information models for configuring selection techniques in Metering Processes and for reporting the technique in use to a Collector.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction

There are two main drivers for the growth in measurement infrastructures and their underlying technology. First, network data rates are increasing, with a concomitant growth in measurement data. Secondly, the growth is compounded by the demand of measurement-based applications for increasingly fine grained traffic measurements. Devices such as routers, which perform the measurements, require increasingly sophisticated and resource intensive measurement capabilities, including the capture of packet headers or even parts of the payload, and classification for flow analysis. All these factors can lead to an overwhelming amount of measurement data, resulting in high demands on resources for measurement, storage, transfer and post processing.

The sustained capture of network traffic at line rate can be performed by specialized measurement hardware. However, the cost of the hardware and the measurement infrastructure required to accommodate the measurements preclude this as a ubiquitous approach. Instead some form of data reduction at the point of measurement is necessary.
This can be achieved by an intelligent packet selection through Sampling or Filtering. Another way to reduce the amount of data is to use aggregation techniques (not addressed in this document). The motivation for Sampling is to select a representative subset of packets that allow accurate estimates of properties of the unsampled traffic to be formed. The motivation for Filtering is to remove all packets that are not of interest. Aggregation combines data and allows compact pre-defined views of the traffic. Examples of applications that

benefit from packet selection are given in [PSAMP-FW]. Aggregation techniques are out of scope of this document.

2. PSAMP Documents Overview

This document is one out of a series of documents from the PSAMP group.

[PSAMP-FW]:	"A Framework for Packet Selection and Reporting" describes the PSAMP framework for network elements to select subsets of packets by statistical and other methods, and to export a stream of reports on the selected packets to a Collector.
[PSAMP-TECH]:	"Sampling and Filtering Techniques for IP Packet Selection" (this document) describes the set of packet selection techniques supported by PSAMP.
[PSAMP-PROTO]:	"Packet Sampling (PSAMP) Protocol Specifications" specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Collecting Process.
[PSAMP-INFO]:	"Information Model for Packet Sampling Exports" defines an information and data model for PSAMP.

3. Terminology

The PSAMP terminology defined here is fully consistent with all terms listed in [PSAMP-FW] but includes additional terms required for the description of packet selection methods. An architecture overview and possible configurations of PSAMP elements can be found in [PSAMP-FW]. PSAMP terminology also aims at consistency with terms used in [RFC3917]. The relationship between PSAMP and IPFIX terms is described in [PSAMP-FW].

In the PSAMP documents all defined PSAMP terms are written capitalized. This document uses the same convention.

3.1 Observation Points, Packet Streams and Packet Content

* Observation Point

  An Observation Point is a location in the network where packets can be observed. Examples include:

  (i) A line to which a probe is attached;

(ii) a shared medium, such as an Ethernet-based LAN;

(iii) a single port of a router, or set of interfaces (physical or logical) of a router;

(iv) an embedded measurement subsystem within an interface.

Note that one Observation Point may be a superset of several other Observation Points. For example one Observation Point can be an entire line card. This would be the superset of the individual Observation Points at the line card's interfaces.

* Observed Packet Stream

  The Observed Packet Stream is the set of all packets observed at the Observation Point.

* Packet Stream

  A packet stream denotes a set of packets that flows past some specified point within the metering process. An example of a Packet Stream is the output of the selection process.
  Note that packets selected from a stream, e.g. by Sampling, do not necessarily possess a property by which they can be distinguished from packets that have not been selected. For this reason the term "stream" is favored over "flow", which is defined as set of packets with common properties [RFC3917].

* Packet Content

  The packet content denotes the union of the packet header (which includes link layer, network layer and other encapsulation headers) and the packet payload. At some Observation Points the link header information may not be available.

3.2 Selection Process

* Selection Process

  A Selection Process takes the Observed Packet Stream as its input and selects a subset of that stream as its output.

* Selection State

  A Selection Process may maintain state information for use by the Selection Process. At a given time, the Selection State

may depend on packets observed at and before that time, and other variables. Examples include:

(i) sequence numbers of packets at the input of Selectors;

(ii) a timestamp of observation of the packet at the Observation Point;

(iii) iterators for pseudo-random number generators;

(iv) hash values calculated during selection;

(v) indicators of whether the packet was selected by a given Selector;

Selection Processes may change portions of the Selection State as a result of processing a packet. Selection state for a packet is to reflect the state after processing the packet.

* Selector

A Selector defines the action of a Selection Process on a single packet of its input. If selected, the packet becomes an element of the output Packet Stream.

The Selector can make use of the following information in determining whether a packet is selected:

(i) the packet's content;

(ii) information derived from the packet's treatment at the Observation Point;

(iii) any selection state that may be maintained by the Selection Process.

* Composite Selector

A Composite Selector is an ordered composition of Selectors, in which the output Packet Stream issuing from one Selector forms the input Packet Stream to the succeeding Selector.

* Primitive Selector

A Selector is primitive if it is not a Composite Selector.

* Selection Sequence

From all the packets observed at an Observation Point, only a few packets are selected by one or more Selectors. The Selection Sequence is a unique value per Observation Domain describing the Observation Point and the Selector IDs through which the packets are selected.

3.3 Reporting

* Packet Reports

  Packet Reports comprise a configurable subset of a packet's input to the Selection Process, including the packet's content, information relating to its treatment (for example, the output interface), and its associated selection state (for example, a hash of the packet's content)

* Report Interpretation:

  Report Interpretation comprises subsidiary information, relating to one or more packets, that is used for interpretation of their packet reports. Examples include configuration parameters of the Selection Process.

* Report Stream:

  The Report Stream is the output of a Metering Process, comprising two distinguished types of information: Packet Reports, and Report Interpretation.

3.4 Metering Process

A Metering Process selects packets from the Observed Packet Stream using a Selection Process, and produces as output a Report Stream concerning the selected packets. The PSAMP Metering Process can be viewed as analogous to the IPFIX metering process [RFC5101], which produces flow records as its output. While the Metering Process definition in this document specifies the PSAMP definition, the PSAMP protocol specifications [PSAMP-PROTO] will use the IPFIX Metering Process definition, which also suits the PSAMP requirements. The relationship between PSAMP and IPFIX is described more in [PSAMP-INFO] and [PSAMP-PROTO].

3.5 Exporting Process

* Exporting Process:

An Exporting Process sends, in the form of Export Packet, the output of one or more Metering Processes to one or more Collectors.

* Export Packet:

  An Export Packet is a combination of Report Interpretation and/or one or more Packet Reports are bundled by the Exporting Process into an Export Packet for exporting to a Collector.

3.6 PSAMP Device

* PSAMP Device

  A PSAMP Device is a device hosting at least an Observation Point, a Metering Process (which includes a Selection Process) and an Exporting Process. Typically, corresponding Observation Point(s), Metering Process(es) and Exporting Process(es) are co-located at this device, for example at a router.

3.7 Collector

* Collector

  A Collector receives a Report Stream exported by one or more Exporting Processes. In some cases, the host of the Metering and/or Exporting Processes may also serve as the Collector.

3.8 Selection Methods

* Filtering
  A filter is a Selector that selects a packet deterministically based on the Packet Content, or its treatment, or functions of these occurring in the Selection State. Two examples are:

  (i) Property match filtering: a packet is selected if a specific field in the packet equals a predefined value.

  (ii) Hash-based selection: a hash function is applied to the Packet Content, and the packet is selected if the result falls in a specified range.

* Sampling

  A selector that is not a filter is called a sampling operation. This reflects the intuitive notion that if the

selection of a packet cannot be determined from its content alone, there must be some type of sampling taking place. Sampling operations can be divided into two subtypes:

(i) Content-independent sampling, which does not use Packet Content in reaching sampling decisions. Examples include systematic sampling, and uniform pseudo-random sampling driven by a pseudo-random number whose generation is independent of Packet Content. Note that in Content-independent Sampling it is not necessary to access the Packet Content in order to make the selection decision.

(ii) Content-dependent sampling, in which the Packet Content is used in reaching selection decisions. An application is pseudo-random selection according to a probability that depends on the contents of a packet field, e.g., sampling packets with a probability dependent on their TCP/UDP port numbers. Note that this is not a Filter.

* Hash Domain

A subset of the Packet Content and the packet treatment, viewed as an N-bit string for some positive integer N.

* Hash Range

A set of M-bit strings for some positive integer M that define the range of values the result of the hash operation can take.

* Hash Function

A Hash Function defines a deterministic mapping from the Hash Domain into the Hash Range.

* Hash Selection Range

The Hash Selection Range is a subset of the Hash Range. The packet is selected if the action of the Hash Function on the Hash Domain for the packet yields a result in the Hash Selection Range.

* Hash-based Selection

Hash-based Selection is a Filtering specified by a Hash Domain, a Hash Function, and Hash Range and a Hash Selection Range.

* Approximative Selection

  Selectors in any of the above categories may be approximated by operations in the same or another category for the purposes of implementation. For example, uniform pseudo-random Sampling may be approximated by Hash-based Selection, using a suitable Hash Function and Hash Domain. In this case, the closeness of the approximation depends on the choice of Hash Function and Hash Domain.

* Population

  A Population is a Packet Stream, or a subset of a Packet Stream. A Population can be considered as a base set from which packets are selected. An example is all packets in the Observed Packet Stream that are observed within some specified time interval.

* Population Size

  The Population Size is the number of all packets in the Population.

* Sample Size

  The number of packets selected from the Population by a Selector.

* Configured Selection Fraction

  The Configured Selection Fraction is the ratio of the number of packets selected by a Selector from an input Population, to the Population Size, as based on the configured selection parameters.

* Attained Selection Fraction

  The Attained Selection Fraction is the actual ratio of the number of packets selected by a Selector from an input Population, to the Population Size.

For some sampling methods the Attained Selection Fraction can differ from the Configured Selection Fraction due to, for example, the inherent statistical variability in sampling decisions of probabilistic Sampling and Hash-based Selection. Nevertheless, for large Population Sizes and properly configured

Selectors, the Attained Selection Fraction usually approaches the Configured Selection Fraction.

4. Categorization of Packet Selection Techniques

Packet selection techniques generate a subset of packets from an Observed Packet Stream at an Observation Point. We distinguish between Sampling and Filtering.

Sampling is targeted at the selection of a representative subset of packets. The subset is used to infer knowledge about the whole set of observed packets without processing them all. The selection can depend on packet position, and/or on packet content, and/or on (pseudo) random decisions.

Filtering selects a subset with common properties. This is used if only a subset of packets is of interest. The properties can be directly derived from the packet content, or depend on the treatment given by the router to the packet. Filtering is a deterministic operation. It depends on packet content or router treatment. It never depends on packet position or on (pseudo) random decisions.

Note that a common technique to select packets is to compute a Hash Function on some bits of the packet header and/or content and to select it if the Hash Value falls in the Hash Selection Range. Since hashing is a deterministic operation on the packet content, it is a Filtering technique according to our categorization. Nevertheless, Hash Functions are sometimes used to emulate random Sampling. Depending on the chosen input bits, the Hash Function and the Hash Selection Range, this technique can be used to emulate the random selection of packets with a given probability p. It is also a powerful technique to consistently select the same packet subset at multiple Observation Points [DuGr00]

The following table gives an overview of the schemes described in this document and their categorization. An X in brackets (X) denotes schemes for which also content-independent variants exist. It easily can be seen that only schemes with both properties, content dependence and deterministic selection, are considered as filters.

Selection Scheme	Deterministic Selection	Content-dependent	Category

Systematic Count-based	X	_	Sampling
Systematic Time-based	X	-	Sampling
Random n-out-of-N	-	-	Sampling
Random Uniform probabilistic	-	-	Sampling
Random Non-uniform probabil.	-	(X)	Sampling
Random Non-uniform flow-state	-	(X)	Sampling
Property Match Filtering	X	(X)	Filtering
Hash Function	X	X	Filtering

In the table x means that the characteristic applies to the selection scheme and (x) means that the characteristic only partly applies to the selection scheme. For instance property match filtering is typically based on packet content and therefore content dependent. But as explained in section 6.1 it may also depend on router state and then would be independent of the content.

The categorization just introduced is mainly useful for the definition of an information model describing Primitive Selectors. More complex selection techniques can be described through the composition of cascaded Sampling and Filtering operations. For example, a packet selection that weights the selection probability on the basis of the packet length can be described as a cascade of a Filtering and a Sampling scheme. However, this descriptive approach is not intended to be rigid: if a common and consolidated selection practice turns out to be too complex to be described as a composition of the mentioned building blocks, an ad hoc description can be specified instead and added as a new scheme to the information model.

5. Sampling

The deployment of Sampling techniques aims at the provisioning of information about a specific characteristic of the parent population at a lower cost than a full census would demand. In order to plan a suitable Sampling strategy it is therefore crucial to determine the needed type of information and the desired degree of accuracy in advance.

First of all, it is important to know the type of metric that should be estimated. The metric of interest can range from simple packet counts [JePP92] up to the estimation of whole distributions of flow characteristics (e.g. packet sizes)[ClPB93].

Secondly, the required accuracy of the information and with this, the confidence that is aimed at, should be known in advance. For instance for usage-based accounting the required confidence for the estimation of packet counters can depend on the monetary value that corresponds to the transfer of one packet. That means that a higher confidence could be required for expensive packet flows (e.g. premium IP service) than for cheaper flows (e.g. best effort). The accuracy requirements for validating a previously agreed quality can also vary extremely with the customer demands. These requirements are usually determined by the service level agreement (SLA).

The Sampling method and the parameters in use must be clearly communicated to all applications that use the measurement data. Only with this knowledge a correct interpretation of the measurement results can be ensured.

Sampling methods can be characterized by the Sampling algorithm, the trigger type used for starting a Sampling interval and the length of the Sampling interval. These parameters are described here in detail. The Sampling algorithm describes the basic process for selection of samples. In accordance to [AmCa89] and [ClPB93] we define the following basic Sampling processes:

5.1 Systematic Sampling

Systematic Sampling describes the process of selecting the start points and the duration of the selection intervals according to a deterministic function. This can be for instance the periodic selection of every k-th element of a trace but also the selection of all packets that arrive at pre-defined points in time. Even if the selection process does not follow a periodic

function (e.g. if the time between the Sampling intervals varies over time) we consider this as systematic Sampling as long as the selection is deterministic.

The use of systematic Sampling always involves the risk of biasing the results. If the systematics in the Sampling process resemble systematics in the observed stochastic process (occurrence of the characteristic of interest in the network), there is a high probability that the estimation will be biased. Systematics in the observed process might not be known in advance.

Here only equally spaced schemes are considered, where triggers for Sampling are periodic, either in time or in packet count. All packets occurring in a selection interval (either in time or packet count) beyond the trigger are selected.

Systematic count-based
In systematic count-based Sampling the start and stop triggers for the Sampling interval are defined in accordance to the spatial packet position (packet count).

Systematic time-based
In systematic time-based Sampling time-based start and stop triggers are used to define the Sampling intervals. All packets are selected that arrive at the Observation Point within the time-intervals defined by the start and stop triggers (i.e. arrival time of the packet is larger than the start time and smaller than the stop time).

Both schemes are content-independent selection schemes. Content dependent deterministic Selectors are categorized as filter.

5.2 Random Sampling

Random Sampling selects the starting points of the Sampling intervals in accordance to a random process. The selection of elements are independent experiments. With this, unbiased estimations can be achieved. In contrast to systematic Sampling, random Sampling requires the generation of random numbers. One can differentiate two methods of random Sampling:

5.2.1   n-out-of-N Sampling

In n-out-of-N Sampling n elements are selected out of the parent population that consists of N elements. One example would be to generate n different random numbers in the range [1,N] and select all packets which have a packet position equal to one of

the random numbers. For this kind of Sampling the Sample Size n is fixed.

5.2.2 Probabilistic Sampling

In probabilistic Sampling the decision whether an element is selected or not is made in accordance to a pre-defined selection probability. An example would be to flip a coin for each packet and select all packets for which the coin showed the head. For this kind of Sampling the Sample Size can vary for different trials. The selection probability does not necessarily has to be the same for each packet. Therefore we distinguish between uniform probabilistic Sampling (with the same selection probability for all packets) and non-uniform probabilistic Sampling (where the selection probability can vary for different packets).

5.2.2.1 Uniform Probabilistic Sampling

For Uniform Probabilistic Sampling packets are selected independently with a uniform probability p. This Sampling can be count-driven, and is sometimes referred to as geometric random Sampling, since the difference in count between successive selected packets are independent random variables with a geometric distribution of mean 1/p. A time-driven analog, exponential random Sampling, has the time between triggers exponentially distributed.
Both geometric and exponential random Sampling are examples of what is known as additive random Sampling, defined as Sampling where the intervals or counts between successive samples are independent identically distributed random variable.

5.2.2.2 Non-Uniform Probabilistic Sampling

This is a variant of Probabilistic Sampling in which the Sampling probabilities can depend on the selection process input. This can be used to weight Sampling probabilities in order e.g. to boost the chance of Sampling packets that are rare but are deemed important. Unbiased estimators for quantitative statistics are recovered by re-normalization of sample values; see [HT52].

5.2.2.3 Non-Uniform Flow State Dependent Sampling

Another type of Sampling that can be classified as probabilistic Non-Uniform is closely related to the flow concept as defined in [RFC3917], and it is only used jointly with a flow monitoring function (IPFIX metering process). Packets are selected,

dependent on a selection state. The point, here, is that the selection state is determined also by the state of the flow the packet belongs to and/or by the state of the other flows currently being monitored by the associated flow monitoring function. An example for such an algorithm is the "sample and hold" method described in [EsVa01]:

- If a packet accounts for a flow record that already exists in the IPFIX flow recording process, it is selected (i.e. the flow record is updated)
- If a packet doesn't account to any existing flow record, it is selected with probability p. If it has been selected a new flow record has to be created.

A further algorithm that fits into the category of non-uniform flow state dependent Sampling is described in [Moli03].

This type of Sampling is content dependent because the identification of the flow the packet belongs to requires analyzing part of the packet content. If the packet is selected, then it is passed as an input to the IPFIX monitoring function (this is called "Local Export" in [PSAMP-FW]. Selecting the packet depending on the state of a flow cache is useful when memory resources of the flow monitoring function are scarce (i.e. there is no room to keep all the flows that have been scheduled for monitoring).

5.2.2.4 Configuration of non-uniform probabilistic and flow-state Sampling

Many different specific methods can be grouped under the terms non-uniform probabilistic and flow state Sampling. Dependent on the Sampling goal and the implemented scheme, a different number and type of input parameters is required to configure such scheme.

Some concrete proposals for such methods exist from the research community (e.g. [EsVa01],[DuLT01],[Moli03]). Some of these proposals are still in an early stage and need further investigations to prove their usefulness and applicability. It is not our aim to indicate preference amongst these methods. Instead, we only describe here the basic methods and leave the specification of explicit schemes and their parameters up to vendors (e.g. as extension of the information model).

6. Filtering

Filtering is the deterministic selection of packets based on the packet content, the treatment of the packet at the Observation Point, or deterministic functions of these occurring in the selection state. The packet is selected if these quantities fall into a specified range. The role of Filtering, as the word itself suggest, is to separate all the packets having a certain property from those not having it. A distinguishing characteristic from Sampling is that the selection decision does not depend on the packet position in time or in the space, or on a random process.
We identify and describe in the following two Filtering techniques.

6.1 Property Match Filtering

With this Filtering method a packet is selected if specific fields within the packet and/or properties of the router state equal a predefined value. Possible filter fields are all IPFIX flow attributes specified in [RFC5102]. Further fields can be defined by proposing new information elements or defining vendor specific extensions.

A packet is selected if Field=Value. Masks and ranges are only supported to the extent to which [RFC5102] allows them e.g. by providing explicit fields like the netmasks for source and destination addresses.

AND operations are possible by concatenating filters, thus producing a composite selection operation. In this case, the ordering in which the filtering happens is implicitly defined (outer filters come after inner filters). However, as long as the concatenation is on filters only, the result of the cascaded filter is independent from the order, but the order may be important for implementation purposes, as the first filter will have to work at a higher rate. In any case, an implementation is not constrained to respect the filter ordering, as long as the result is the same, and it may even implement the composite filtering in filtering in one single step.

OR operations are not supported with this basic model. More sophisticated filters (e.g. supporting bitmasks, ranges or OR operations etc.) can be realized as vendor specific schemes.

All IPFIX flow attributes defined in [RFC5102] can be used for property match filtering. Further information elements can be easily defined. Typical header fields that should be supported for property match operations are the following:

(i) the IP header (excluding options in IPv4, stacked headers in IPv6)

(ii) transport protocol header (e.g. TCP, UDP)

(iii) encapsulation headers (e.g. the MPLS label stack, if present)

When the PSAMP Device offers property match filtering, and, in its usual capacity other than in performing PSAMP functions, identifies or processes information from IP, transport protocol or encapsulation protocols, then the information should be made available for filtering. For example, when a PSAMP Device routes based on destination IP address, that field should be made available for filtering. Conversely, a PSAMP Device that does not route is not expected to be able to locate an IP address within a packet, or make it available for Filtering, although it may do so.

Since packet encryption conceals the real values of encrypted fields, property match filtering must be configurable to ignore encrypted packets, when detected.

The Selection Process may support filtering based on the properties of the router state:

(i) Ingress interface at which packet arrives equals a specified value

(ii) Egress interface to which packet is routed to equals a specified value

(iii) Packet violated Access Control List (ACL) on the router

(iv) Failed Reverse Path Forwarding (RPF)

(v) Failed Resource Reservation (RSVP)

(vi) No route found for the packet

(vii) Origin Border Gateway Protocol (BGP) Autonomous System (AS) [RFC4271] equals a specified value or lies within a given range

(viii)Destination BGP AS equals a specified value or lies within a given range

Packets that match the Failed Reverse Path Forwarding (RPF) condition are packets for which ingress filtering failed as defined in [RFC3704].
Packets that match the Failed Resource Reservation condition are packets that do not fulfill the RSVP specification as defined in [RCF2205].

Router architectural considerations may preclude some information concerning the packet treatment being available at line rate for selection of packets. For example, the Selection Process may not be implemented in the fast path that is able to access routing state at line rate. However, when filtering follows sampling (or some other selection operation) in a Composite Selector, the rate of the Packet Stream output from the sampler and input to the filter may be sufficiently slow that the filter could select based on routing state.

6.2 Hash-based Filtering

A Hash Function h maps the Packet Content c, or some portion of it, onto a Hash Range R. The packet is selected if h(c) is an element of S, which is a subset of R called the Hash Selection Range. Thus Hash-based Selection is a particular case of Filtering. The object is selected if c is in inv(h(S)). But for desirable Hash Functions the inverse image inv(h(S)) will be extremely complex, and hence h would not be expressible as, say, a Property Match Filter or a simple combination of these.

Hash-based selection is mainly used to realize a coordinated packet selection. That means that the same packets are selected at different Observation Points. This is useful for instance to observe the path (trajectory) that a packet took through the network or to apply packet selection to passive one-way measurements.

A pre-requisite for the method to work and to ensure interoperability is that the same Hash Function with the same parameters (e.g. input vector) is used at the observation points.

A consistent packet selection is also possible with property match filtering. Nevertheless, hash-based selection can be used to approximate a random selection. The desired statistical properties are discussed in section 6.2.2.

In the following subsections we give some application examples for coordinated packet selection.

6.2.1 Application Examples for Coordinated Packet Selection

6.2.1.1 Trajectory Sampling

Trajectory Sampling is the consistent selection of a subset of packets at either all of a set of Observation Points or none of them. Trajectory Sampling is realized by Hash-based Selection if all Observation Points in the set use a common Hash Function, Hash Domain and selection range. The Hash Domain comprises all or part of the packet content that is invariant along the packet path. Fields such as Time-to-Live, which is decremented per hop, and header CRC, which is recalculated per hop, are thus excluded from the Hash Domain. The Hash Domain needs to be wider than just a flow key, if packets are to be selected quasi-randomly within flows.

The trajectory (or path) followed by a packet is reconstructed from PSAMP reports on it that reach a Collector. Reports on a given packet originating from different observations points are associated by matching a label from the reports. The label may comprise that portion invariant packet content that is reported, or possibly some digest of the invariant packet content that is inserted into the packet report at the Observation Point. Such a digest may be constructed by applying a second Hash Function (distinct from that used for selection) to the invariant packet content. The reconstruction of trajectories, and methods for dealing with possible ambiguities due to label collisions (identical labels reported for different packets) and potential loss of reports in transmission, are dealt with in [DuGr00], [DuGG02] and [DuGr04].

Applications of trajectory Sampling include (i) estimation of the network path matrix, i.e., the traffic intensities according to network path, broken down by flow key; (ii) detection of routing loops, as indicated by self-intersecting trajectories; (iii) passive performance measurement: prematurely terminating trajectories indicate packet loss, packet one way delay can be determined if reports include (synchronized) timestamps of packet arrival at the Observation Point; (iv) network attack tracing, of the actual paths taken by attack packets with spoofed source addresses.

6.2.1.2 Passive One-way Measurements

Coordinated packet selection can be applied for instance to one-way delay measurements in order to reduce the required resources. In one-way delay measurements packets are collected

at different Observation Points in the network. A packet digest is generated for each packet that helps to identify the packet. The packet digest and the arrival time of the packet at the observation point are reported to a process that calculates the delay. The delay is calculated by subtracting the arrival time of the same packet at the observation points (e.g. [ZsZC01]). With high data rates, capturing all packets can require a lot of resources for storage, transfer and processing. To reduce resource consumption packet selection methods can be applied. But for such selection techniques it has to be ensured that the same packets are collected at different observation points.

6.2.1.3 Generation of Pseudo-random Numbers

Although pseudo-random number generators with well understood properties have been developed, they may not be the method of choice in settings where computational resources are scarce. A convenient alternative is to use Hash Functions of packet content as a source of randomness. The hash (suitably re-normalized) is a pseudo-random variate in the interval [0,1]. Other schemes may use packet fields in iterators for pseudo-random numbers. However, the statistical properties of an ideal packet selection law (such as independent Sampling for different packets, or independence on packet content) may not be exactly rendered by an implementation, but only approximately so.

Use of packet content to generate pseudo-random variates shares with Non-uniform Probabilistic Sampling (see Section 3.1.2.2.2 above) the property that selection decisions depend on Packet Content. However, there is a fundamental difference between the two. In the former case the content determines pseudo-random variates. In the latter case the content only determines the selection probabilities: selection could then proceed e.g., by use of random variates obtained by an independent pseudo-random number generator.

6.2.2 Desired Properties of Hash Functions

Here we formulate desired properties for hash functions. For this we have to distinguish whether a hash function is used for packet selection or just as a packet digest. The main purpose of this document is on packet selection. Nevertheless, we also provide some requirements for the use of hash functions as packet digest.

First of all we need to define suitable input fields from the packet. In accordance to [DuGr00] input field should be

- invariant on the path
- variable among packets

Only if the input fields are the same at different observation points it is possible to recognize the packet. The input fields should be variable among packets in order to distribute the hash results over the Selection Range.

6.2.2.1 Requirements for Packet Selection

In accordance to considerations in [MoND05] and [Henk08] we define the following desired properties of hash functions used for packet selection:

(i) Speed: The hash function has to be applied to each packet that traverses the observation point. Therefore it has to be fast in order to cope with the high packet rates. In the ideal case the hash operation should not influence the performance on the PSAMP device.

(ii) Uniformity: The Hash Function h should have good mixing properties, in the sense that small changes in the input (e.g. the flipping of a single bit) cause large changes in the output (many bits change). Then any local clump of values of c is spread widely over R by h, and so the distribution of h(c) is fairly uniform even if the distribution of c is not. Then the Sampling Fraction is #S/#R, which can be tuned by choice of S.

(iii) Unbiasedness: The selection decision should be as independent of packet attributes as possible. The set of selected packets should not be biased towards a specific type of packets.

(iv) Representativeness of sample: The sample should be as representative as possible for the observed traffic.

(v) Non-linearity: The function should not be linear. This increases the mixing properties (uniformity criterion). In addition to this it decreases the predictability of the output and therefore the vulnerabilities against attacks.

(vi) Robustness against vulnerabilities: The hash function should be robust against attacks. Potential vulnerabilities are described in section 6.2.3.

6.2.2.2 Requirements for Packet Digesting

For digesting Packet Content for inclusion in a reported label, the most important property is a low collision frequency. A secondary requirement is the ability to accept variable length input, in order to allow inclusion of maximal amount of packet as input. Execution speed is of secondary importance, since the digest need only be formed from selected packets.

6.2.3 Security Considerations for Hash Functions

A concern for Hash-based Selection is whether some large set of related packets could be disproportionately sampled, i.e., that the Attained Sampling Fraction is significantly different from the Configured Sampling Fraction. This can happen either

(i) through unanticipated behavior in the Hash Function, or

(ii) because the packets had been deliberately crafted to have this property.

The first point underlines the importance of using a Hash Function with good mixing properties. For this the statistical properties of candidate Hash Functions need to be evaluated. Since the hash output depends on the traffic mix, the evaluation should be done preferably on up-to-date packet traces from the network in which the hash-based selection will be deployed.

However, hash functions which perform well on typical traffic may not be sufficiently strong to withstand attacks specifically targeted against them. Such potential attacks have been described in [GoRe07].

The following we point out different potential attack scenarios. We encourage the use of standardized hash functions. Therefore we assume that the hash function itself is public and hence known to an attacker.
Nevertheless, we also assume the possibility of using a private input parameter for the hash function that is kept secret. Such an input parameter can for instance be attached to the hash input before the hash operation is applied. With this at least parts of the hash operation remains secret.

For the attack scenarios we assume that an attacker uses its knowledge of the hash function to craft packets which are then dispatched, either as the attack itself, or to elicit further information which can be used to refine the attack.

Two scenarios are considered. In the first scenario, the attacker has no knowledge about whether the crafted packets are

selected or not. In the second scenario the attacker uses some knowledge of sampling outcomes. The means by which this might be acquired is discussed below. Some additional attacks that involve tampering with export packets in transit, as opposed to attacking the PSAMP device, are discussed in [GoRe07].

6.2.3.1 Vulnerabilities of Hash-based selection without knowledge of selection outcomes

(i) The hash function does not use a private parameter.

If no private input parameter is used, potential attackers can easily calculate which packets result in which hash values. If the selection range is public, an attacker can craft packets whose selection properties are known in advance. If the selection range is private, an attacker cannot determine whether a crafted packet is selected. However by computing the hash on different trial crafted packets, and selecting those yielding a given hash value, the attacker can construct an arbitrarily large set of distinct packets with a common selection properties, i.e., packets that will be either all selected or all not selected. This can be done whatever the strength of the hash function.

(ii) The hash function is not cryptographically strong.

If the hash function is not cryptographically strong, it may be possible to construct sequences of distinct packets with the common selection property even if a private parameter is used.

An example is the standard CRC-32 hash function used with a private modulus (but without a private string post-pended to the input). It has weak mixing properties for low order bits. Consequently, simply by incrementing the hash input, one obtains distinct packets whose hashes mostly fall in a narrow range, and hence are likely commonly selected; see [GoRe07]

Suitable parameterization of the hash function can make such attacks more difficult. For example, post-pending a private string to the input before hashing with CRC-32 will give stronger mixing properties over all bits of the input. However, with a hash function, such as CRC-32, that is not cryptographically strong, the possibility of discovering a method to construct packet sets with the common selected property cannot be ruled out, even when a private modulus or post-pended string is used.

6.2.3.2 Vulnerabilities of Hash-based selection using knowledge of selection outcomes

Knowledge of the selection outcomes of crafted packets can be used by an attacker to more easily construct sets of packets which are disproportionately sampled and/or are commonly selected. For this the attacker does not need any a priori knowledge about the hash function or selection range.

There are several ways an attacker might acquire this knowledge about the selection outcome:

(i) Billing Reports: if samples are used for billing purposes, then the selection outcomes of packets may be able to be inferred by correlating a crafted packet stream with the billing reports that it generates. However, the rate at knowledge of selection outcomes can be acquired depends on the temporal and spatial granularity of the billing reports, being slower the more aggregated the reports are.

(ii) Feedback from an Intrusion Detection System: e.g., a botmaster adversary learns if his packets were detected by the intrusion detection system by seeing if one of his bots is blocked by the network.

(iii) Observation of the Report Stream: export packets sent across a public network may be eavesdropped on by an adversary. Encryption of the export packets provides only a partial defense, since it may be possible to infer the selection outcomes of packets by correlating a crafted packet stream with the occurrence (not the content) of packets in the export stream that it generates. The rate at which such knowledge could be acquired is limited by the temporal resolution at which reports can be associated with packets, e.g. due to processing and propagation variability, and difficulty in distinguishing report on attack packets from those of background traffic, if present. The association between packets and their reports on which this depends could be removed by padding export packets to a constant length and sending them at a constant rate.

We now turn to attacks that can exploit knowledge of selection outcomes. Firstly, with a non-cryptographic hash function, knowledge of selection outcomes for a trial stream may be used to further craft a packet set with the common selection property. This has been demonstrated for the modular hash f(x) = a x + b mod k, for private parameters a, b, and k. With sampling rate p, knowledge of the sampling outcomes of roughly 2/p is sufficient for the attack to succeed, independent of the values of a, b and k. With knowledge of the selection outcomes of a larger number of packets, the parameters a b and k can be determined; see [GoRe07].

A cryptographic hash function employing a private parameter and operating in one of the pseudo-random function modes specified above is not vulnerable to these attacks, even if the selection range is known.

6.2.3.3 Vulnerabilities to Replay Attacks

Since hash-based selection is deterministic, any packet or set of packets with known selection properties can be replayed into a network and experience the same selection outcomes provide the hash function and its parameters are not changed. Repetition of a single packet may be noticeable to other measurement methods if employed (e.g. collection of flow statistics), whereas a set of distinct packets that appears statistically similar to regular traffic may be less noticeable.

Replay attacks may be mitigated by repeated changing of hash function parameters. This also prevents attacks that exploit knowledge of sampling outcomes, at least if the parameters are changed at least as fast as the knowledge can be acquired by an attacker. In order to preserve the ability to perform Trajectory Sampling, parameter changed would have to be simultaneous (or approximately so) across all observation point.

6.2.4   Choice of Hash-Function

The specific choice of hash function represents a trade-off between complexity and ease of implementation. Ideally, a cryptographically strong hash function employing a private parameter and operating in pseudo-random function mode as specified above would be used, yielding a good emulation a random packet selection at a target sampling rate, and giving maximal robustness against the attacks described in the previous section. Unfortunately there is currently no single hash function that fulfills all the requirements.

As detailed in section 6.2.3, only cryptographic hash functions employing a private parameter operating in pseudo-random function mode are sufficiently strong to withstand the range of conceivable attacks. For example, fixed or variable length inputs could be hashed using a block cipher (like AES) in cipher-block-chaining mode. Fixed length inputs could also be hashed using an iterated cryptographic hash function (like MD5

or SHA1), with a private initial vector. For variable length inputs, iterated cryptographic hash function (like MD5 or SHA1) should employ private string post-pended to the data in addition to a private initial vector. For more details, see the "append-cascade" construction of [BeCK96]. We encourage the use of such cryptographically strong hash function wherever possible.

However, a problem with using such function is the low performance. As shown for instance in [Henk08], the computation time for MD5 and SHA are about 7-10 times higher compared to non-cryptographic functions. The difference increases for small hash input lengths.

Therefore it is not assumed that all PSAMP devices will be capable of applying a cryptographically strong hash function to every packet at line rate. For this reason, the hash functions listed in this section will be of a weaker variety. Future protocol extensions that employ stronger hash functions are highly welcome.

Comparisons of hash-functions for packet selection and packet digesting with regard to various criteria can be found in [MoND05] and [Henk08].

6.2.4.1 Hash Functions for Packet Selection

If hash-based packet selection is applied, the BOB function MUST be used for packet selection operations in order to be compliant with PSAMP. The specification of BOB is given in the appendix. Both the parameter (the init value) and the selection range should be kept private. The initial vector of the hash function MUST be configurable out of band to prevent security breaches like exposure of the initial vector content.

Other functions, such as CRC-32 and IPSX MAY be used. The IPSX function is described in the appendix, the CRC-32 function is described in [RFC1141]. If CRC-32 is used, the input should first be post-pended with a private string that acts as a parameter, and the modulus of the CRC should also be kept private.

IPSX is simple to implement and was correspondingly about an order of magnitude faster to execute per packet than BOB or CRC-32 [MoND05].

All three hash functions evaluated showed relatively poor uniformity with 16 byte input that was drawn from only invariant fields in the IP and TCP/UDP headers (i.e. header fields that do not change from hop to hop). IPSX is inherently limited to 16 bytes.
BOB and CRC-32 exhibits noticeably better uniformity when 4 or more bytes from the payload are also included in the input [MoND05]. Also with other criteria BOB performed quite well [Henk08]

Although the characteristics have been checked for different traffic traces, results cannot be generalized to arbitrary traffic. Since hash-based selection is a deterministic function on the packet content, it can always be biased towards packets with specific attributes. Furthermore, it should be noted that all Hash Functions were evaluated only for IPv4.

None of these hash functions is recommended for cryptographic purposes. Please also note that the use of a private parameter only slightly reduces the vulnerabilities against attacks. As shown in section 6.2.3. functions that are not cryptographically strong (e.g., BOB and CRC) cannot prevent attackers from crafting packets that are disproportionally selected even if a private parameter is used and the selection range is kept secret.

As described in section 6.2.2 the input bytes for the Hash Function need to be invariant along the path the packet is traveling. Only with this it is ensured that the same packets are selected at different observation points. Furthermore they should have a high variability between different packets to generate a high variation in the Hash Range. An evaluation of the variability of different packet header fields can be found in [DuGr00], [HeSZ08] and [Henk08].

If a hash-based selection with the BOB function is used with IPv4 traffic, the following input bytes MUST be used.
- IP identification field
- Flags field
- Fragment offset
- Source IP address
- Destination IP address
- A configurable number of bytes from the IP payload, starting
  at a configurable offset.

Due to the lack of suitable IPv6 packet traces, all candidate

Hash Functions in [DuGr00], [MoND05] and [Henk08] were evaluated only for IPv4. Due to the IPv6 header fields and address structure it is expected that there is less randomness in IPv6 packet headers than in IPv4 headers. Nevertheless, the randomness of IPv6 traffic has not yet been evaluated sufficiently to get any evidence. In addition to this, IPv6 traffic profiles may change significantly in future when IPv6 is used by a broader community.

If a hash-based selection with the BOB function is used with IPv6 traffic, the following input bytes MUST be used.
- Payload length (2 bytes)
- Byte number 10,11,14,15,16 of the IPv6 source address
- Byte number 10,11,14,15,16 of the IPv6 destination address
- A configurable number of bytes from the IP payload, starting at a configurable offset. It is recommended to use at least 4 bytes from the IP payload.

The payload itself is not changing during the path. Even if some routers process some extension headers they are not going to strip them from the packet. Therefore the payload length is invariant along the path. Furthermore it usually differs for different packets. The IPv6 address has 16 bytes. The first part is the network part and it contains low variation. The second part is the host part and contains higher variation. Therefore the second part of the address is used. Nevertheless, the uniformity has not been checked for IPv6 traffic.

6.2.4.2 Hash Functions Suitable for Packet Digesting

For this purpose also the BOB function SHOULD be used. Other functions (such as CRC-32) MAY be used. Among the functions capable of operating with variable length input BOB and CRC-32 have the fastest execution, BOB being slightly faster. IPSX is not recommended for digesting because it has a significantly higher collision rate and takes only a fixed length input.

7. Parameters for the Description of Selection Techniques

This section gives an overview of different alternative selection schemes and their required parameters. In order to be compliant with PSAMP at least one of proposed schemes MUST be implemented.

The decision whether to select a packet or not is based on a function which is performed when the packet arrives at the selection process. Packet selection schemes differ in the input parameters for the selection process and the functions they

require to do the packet selection. The following table gives an overview.

Scheme	input parameters	functions
systematic count-based	packet position Sampling pattern	packet counter
systematic time-based	arrival time Sampling pattern	clock or timer
random n-out-of-N	packet position Sampling pattern (random number list)	packet counter, random numbers
uniform probabilistic	Sampling probability	random function
non-uniform probabilistic	e.g. packet position, packet content(parts)	selection function, probability calc.
non-uniform flow-state	e.g. flow state, packet content(parts)	selection function, probability calc.
property match	packet content(parts) or router state	filter function or state discovery
hash-based	packet content(parts)	Hash Function

7.1 Description of Sampling Techniques

In this section we define what elements are needed to describe the most common Sampling techniques. Here the selection function is pre-defined and given by the Selector ID.

Sampler Description:
SELECTOR_ID
SELECTOR_TYPE
SELECTOR_PARAMETERS

Where:

SELECTOR_ID:
Unique ID for the packet sampler.

SELECTOR_TYPE

For Sampling processes the SELECTOR TYPE defines what Sampling algorithm is used.
Values: Systematic Count-based | Systematic Time-based | Random n-out-of-N | Uniform Probabilistic | Non-uniform Probabilistic | Non-uniform Flow-state

SELECTOR_PARAMETERS
For Sampling processes the SELECTOR PARAMETERS define the input parameters for the process. Interval length in systematic Sampling means, that all packets that arrive in this interval are selected. The spacing parameter defines the spacing in time or number of packets between the end of one Sampling interval and the start of the next succeeding interval.

Case n out of N:
- Population size N, Sample size n

Case Systematic Time Based:
- Interval length (in usec), Spacing (in usec)

Case Systematic Count Based:
- Interval length(in packets), Spacing (in packets)

Case Uniform Probabilistic (with equal probability per packet):
- Sampling probability p

Case Non-uniform Probabilistic:
- Calculation function for Sampling probability p (see also section 5.2.2.4)

Case flow state:
- Information reported for flow state sampling are not defined in this document (see also section 5.2.2.4)

7.2 Description of Filtering Techniques

In this section we define what elements are needed to describe the most common Filtering techniques. The structure closely parallels the one presented for the Sampling techniques.

Filter Description:
SELECTOR_ID
SELECTOR_TYPE
SELECTOR_PARAMETERS

Where:

SELECTOR_ID:

Unique ID for the packet filter. The ID can be calculated under consideration of the SELECTION SEQUENCE and a local ID.

SELECTOR_TYPE
For Filtering processes the SELECTOR TYPE defines what Filtering type is used.
Values: Matching | Hashing | Router_state

SELECTOR_PARAMETERS
For Filtering processes the SELECTOR PARAMETERS define formally the common property of the packet being filtered. For the filters of type Matching and Hashing the definitions have a lot of points in common.

Values:

Case Matching
- Information Element (from [RFC5102])
- Value (type in accordance to [RFC5102])

In case of multiple match criteria, multiple "case matching" have to be bound by a logical AND.

Case Hashing:
- Hash Domain (Input bits from packet)
    - <Header type = IPv4>
    - <Input bit specification, header part>
    - <Header type = IPv6>
    - <Input bit specification, header part>
    - <payload byte number N>
    - <Input bit specification, payload part>
- Hash Function
    - Hash function name
    - Length of input key (eliminate 0x bytes)
    - Output value (length M and bitmask)
    - Hash Selection Range, as a list of non overlapping intervals [start value, end value] where value is in [0,2^M-1]
    - Additional parameters dependent on specific Hash Function (e.g. hash input bits (seed))

Notes to input bits for Case Hashing:
- Input bits can be from header part only, from the payload part only or from both.
- The bit specification, for the header part, can be specified for IPv4 or IPv6 only, or both

- In case of IPv4, the bit specification is a sequence of 20 Hexadecimal numbers [00,FF] specifying a 20 bytes bitmask to be applied to the header.
- In case of IPv6, it is a sequence of 40 Hexadecimal numbers [00,FF] specifying a 40 bytes bitmask to be applied to the header
- The bit specification, for the payload part, is a sequence of Hexadecimal numbers [00,FF] specifying the bitmask to be applied to the first N bytes of the payload, as specified by the previous field. In case the Hexadecimal number sequence is longer than N, only the first N numbers are considered.
- In case the payload is shorter than N, the Hash Function cannot be applied. Other options, like padding with zeros, may be considered in the future.
- A Hash Function cannot be defined on the options field of the IPv4 header, neither on stacked headers of IPv6.
- The Hash Selection Range defines a range of hash-values (out of all possible results of the Hash-Operation). If the hash result for a specific packet falls in this range, the packet is selected. If the value is outside the range, the packet is not selected. E.g. if the selection interval specification is [1:3], [6:9] all packets are selected for which the hash result is 1,2,3,6,7,8, or 9. In all other cases the packet is not selected.

Case Router State:

- Ingress interface at which the packet arrives equals a specified value
- Egress interface to which the packet is routed equals a specified value
- Packet violated Access Control List (ACL) on the router
- Reverse Path Forwarding (RPF) failed for the packet
- Resource Reservation is insufficient for the packet
- No route found for the packet
- Origin AS equals a specified value or lies within a given range
- Destination AS equals a specified value or lies within a given range

Note to Case Router State:
- All Router state entries can be linked by AND operators

8. Composite Techniques

Composite schemes are realized by combining the selector IDs into a Selection Sequence. The Selection Sequence contains all selector IDs that are applied to the packet stream subsequently. Some examples of composite schemes are reported below.

8.1 Cascaded Filtering->Sampling or Sampling->Filtering

If a filter precedes a Sampling process the role of Filtering is to create a set of "parent populations" from a single stream that can then be fed independently to different Sampling functions, with different parameters tuned for the population itself (e.g. if streams of different intensity result from Filtering, it may be good to have different Sampling rates). If Filtering follows a Sampling process, the same Sampling Fraction and type is applied to the whole stream, independently of the relative size of the streams resulting from the Filtering function. Moreover, also packets not destined to be selected in the Filtering operation will "load" the Sampling function. So, in principle, Filtering before Sampling allows a more accurate tuning of the Sampling procedure, but if filters are too complex to work at full line rate (e.g. because they have to access router state information), Sampling before Filtering may be a need.

8.2 Stratified Sampling

Stratified Sampling is one example for using a composite technique. The basic idea behind stratified Sampling is to increase the estimation accuracy by using a-priori information about correlations of the investigated characteristic with some other characteristic that is easier to obtain. The a-priori information is used to perform an intelligent grouping of the elements of the parent population. In this manner, a higher estimation accuracy can be achieved with the same Sample Size or the Sample Size can be reduced without reducing the estimation accuracy.

Stratified Sampling divides the Sampling process into multiple steps. First, the elements of the parent population are grouped into subsets in accordance to a given characteristic. This grouping can be done in multiple steps. Then samples are taken from each subset.

The stronger the correlation between the characteristic used to divide the parent population (stratification variable) and the characteristic of interest (for which an estimate is sought after), the easier is the consecutive Sampling process and the higher is the stratification gain. For instance, if the dividing characteristic were equal to the investigated characteristic, each element of the sub-group would be a perfect representative of that characteristic. In this case it would be sufficient to take one arbitrary element out of each subgroup to get the actual distribution of the characteristic in the parent population. Therefore stratified Sampling can reduce the costs for the Sampling process (i.e. the number of samples needed to achieve a given level of confidence).

For stratified Sampling one has to specify classification rules for grouping the elements into subgroups and the Sampling scheme that is used within the subgroups. The classification rules can be expressed by multiple filters. For the Sampling scheme within the subgroups the parameters have to be specified as described above. The use of stratified Sampling methods for measurement purposes is described for instance in [ClPB93] and [Zseb03].

9. Security Considerations

Security considerations concerning the choice of sampling hash function have been discussed in Section 6.2.2. That section discussed a number of potential attacks to craft packet streams which are disproportionately detected and/or discover the hash function parameters, the vulnerabilities of different hash functions to these attacks, and practices to minimize these vulnerabilities.

In addition to this a user can gains knowledge about the start and stop triggers in time-based systematic sampling e.g. by sending test packets. This knowledge might allow users to modify their send schedule in a way that their packets are disproportionately selected or not selected [GoRe07].

For random sampling cryptographically-strong random number generator should be used in order to prevent that an advisory can predict the selection decision [GoRe07].

Further security threats can occur when sampling parameters are configured or communicated to other entities. The configuration and reporting of sampling parameters are out of scope of this document. Therefore the security threats that originate from this kind of communication cannot be assessed with the information given in this document.

Some of these threats can probably be addressed by keeping configuration information confidential and by authenticating entities that configure sampling. Nevertheless a full analysis and assessment of threats for configuration and reporting has to be done if configuration or reporting methods are proposed.

10. Acknowledgements

We would like to thank the PSAMP group, especially Benoit Claise and Stewart Bryant, for fruitful discussions and for proofreading the document. We thank Sharon Goldberg for her input on security issues concerning hash-based selection.

11. IANA Considerations

This document has no actions for IANA.

12. Normative References

[RFC2119]   Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, BCP 14, RFC 2119, March 1997

13. Informative References

[AmCa89]   Paul D. Amer, Lillian N. Cassel, "Management of Sampled Real-Time Network Measurements", 14th Conference on Local Computer Networks, October 1989, Minneapolis, pages 62-68, IEEE, 1989.

[BeCK96]   M. Bellare, R. Canetti and H. Krawczyk, "Pseudorandom Functions Revisited: The Cascade Construction and its Concrete Security", Symposium on Foundations of Computer Science, 1996.

[ClPB93]   K.C. Claffy, George C. Polyzos, Hans-Werner Braun, "Application of Sampling Methodologies to Network Traffic Characterization", Proceedings of ACM SIGCOMM'93, San Francisco, CA, USA, September 13 - 17, 1993.

[DuGG02]   N.G. Duffield, A. Gerber, M. Grossglauser, "Trajectory Engine: A Backend for Trajectory Sampling", IEEE Network Operations and Management Symposium 2002, Florence, Italy, April 15-19, 2002.

[DuGr00]   N.G. Duffield, M. Grossglauser, "Trajectory Sampling for Direct Traffic Observation", Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August 28 - September 1, 2000.

[DuGr04]   N. G. Duffield and M. Grossglauser "Trajectory Sampling with Unreliable Reporting", Proc IEEE Infocom 2004, Hong Kong, March 2004.

[DuLT01]   N.G. Duffield, C. Lund, and M. Thorup, "Charging from Sampled Network Usage", ACM Internet Measurement Workshop IMW 2001, San Francisco, USA, November 1-2, 2001.

[EsVa01]   C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting", ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco (CA) Nov. 2001.

[GoRe07]   S. Goldberg, J. Rexford, "Security Vulnerabilities and Solutions for Packet Sampling", IEEE Sarnoff Symposium, Princeton, NJ, May 2007.

[HT52]   D.G. Horvitz and D.J. Thompson, "A Generalization of Sampling without replacement from a Finite Universe" J. Amer. Statist. Assoc. Vol. 47, pp. 663-685, 1952.

[Henk08]   Christian Henke, Evaluation of Hash Functions for Multipoint Sampling in IP Networks, Diploma Thesis, TU Berlin, April 2008.

[HeSZ08]   Christian Henke, Carsten Schmoll, Tanja Zseby, Evaluation of Header Field Entropy for Hash-Based Packet Selection, Proceedings of Passive and Active Measurement Conference PAM 2008, Cleveland, Ohio, USA, April 2008.

[RFC5102]   J. Quittek, S. Bryant, B. Claise, P. Aitken, J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

[RFC5101]   B. Claise (Editor) "Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.

[Jenk97]   B. Jenkins, "Algorithm Alley", Dr. Dobb's Journal, September 1997. http://burtleburtle.net/bob/hash/doobs.html

[JePP92]   Jonathan Jedwab, Peter Phaal, Bob Pinna, "Traffic Estimation for the Largest Sources on a Network, Using Packet Sampling with Limited Storage", HP technical report, Managemenr, Mathematics and Security Department, HP Laboratories, Bristol,

```
                March 1992,
                http://www.hpl.hp.com/techreports/92/HPL-92-35.html

   [Moli03]    M.Molina, "A scalable and efficient methodology for
                flow monitoring in the Internet", International
                Teletraffic Congress (ITC-18), Berlin, Sep. 2003

   [MoND05]    M. Molina, S.Niccolini, N.G.Duffield "A Comparative
                Experimental Study of Hash Functions Applied to
                Packet Sampling" International Teletraffic Congress
                (ITC-19), Beijing, August 2005.

   [PSAMP-FW]  Nick Duffield (Ed.), "A Framework for Packet
                Selection and Reporting", RFC XXXX [currently
                Internet Draft draft-ietf-psamp-framework-11, work
                in progress, May 2007].

   [PSAMP-INFO] T. Dietz, F. Dressler, G. Carle, B. Claise,
                "Information Model for Packet Sampling Exports",
                RFC XXXX. [Currently Internet Draft, draft-ietf-
                psamp-info-06, June 2007]

   [PSAMP-PROTO] B. Claise (Ed.), "Packet Sampling (PSAMP) Protocol
                Specifications", RFC XXXX. [Currently Internet
                Draft draft-ietf-psamp-protocol-07.txt, work in
                progress, October 2006].

   [RFC1141]   T. Mallory, A. Kullberg, "Incremental Updating of
                the Internet Checksum", RFC 1141, January 1990
                (updated by RFC1624).

   [RFC1624]   A. Rijsinghani, Computation of the Internet
                Checksum via Incremental Update, RFC1624, May 1994

   [RFC2205]   R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, S.
                Jamin, Resource ReSerVation Protocol (RSVP) -
                Version 1 Functional Specification, RFC2205,
                September 1997

   [RFC3704]   F. Baker, P. Savola, Ingress Filtering for
                Multihomed Networks, RFC3704, March 2004


   [RFC3917]   J. Quittek, T. Zseby, B. Claise, S. Zander,
                "Requirements for IP Flow Information Export", RFC
                3917, October 2004.
```

[RFC4271]   Y. Rekhter, T. Li, S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[Zseb03]   T. Zseby, "Stratification Strategies for Sampling-based Non-intrusive Measurement of One-way Delay", Proceedings of Passive and Active Measurement Workshop (PAM 2003), La Jolla, CA, USA, pp. 171-179, April 2003.

[ZsZC01]   Tanja Zseby, Sebastian Zander, Georg Carle. Evaluation of Building Blocks for Passive One-way-delay Measurements. Proceedings of Passive and Active Measurement Workshop (PAM 2001), Amsterdam, The Netherlands, April 23-24, 2001.

14. Authors' Addresses

Tanja Zseby
Fraunhofer Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31
10589 Berlin
Germany
Phone: +49-30-34 63 7153
Email: tanja.zseby@fokus.fraunhofer.de

Maurizio Molina
DANTE
City House
126-130 Hills Road
Cambridge CB21PQ
United Kingdom
Phone: +44 1223 371 300
Email: maurizio.molina@dante.org.uk

Nick Duffield
AT&T Labs - Research
Room B-139
180 Park Ave
Florham Park NJ 07932, USA
Phone: +1 973-360-8726
Email: duffield@research.att.com

Saverio Niccolini
Network Laboratories, NEC Europe Ltd.
Kurfuerstenanlage 36
69115 Heidelberg
Germany

Phone: +49-6221-9051118
Email: saverio.niccolini@netlab.nec.de

Fredric Raspall
EPSC-UPC
Dept. of Telematics
Av. del Canal Olimpic, s/n
Edifici C4
E-08860 Castelldefels, Barcelona
Spain
Email: fredi@entel.upc.es

15. Contributors

Sharon Goldberg contributed to the security considerations for hash-based selection.

Sharon Goldberg
Department of Electrical Engineering
Princeton University
F210-K EQuad
Princeton, NJ 08544, USA
Email: goldbe@princeton.edu

16. Intellectual Property Statement

17. Copyright Statement

18. Disclaimer

Appendix A: Hash Functions

A.1 IP Shift-XOR (IPSX) Hash Function

The IPSX Hash Function is tailored for acting on IP version 4 packets. It exploits the structure of IP packet and in particular the variability expected to be exhibited within different fields of the IP packet in order to furnish a hash value with little apparent correlation with individual packet fields. Fields from the IPv4 and TCP/UDP headers are used as input. The IPSX Hash Function uses a small number of simple instructions.

Input parameters: None

Built-in parameters: None

Output: The output of the IPSX is a 16 bit number

Functioning:

The functioning can be divided into two parts: input selection, which forms are composite input from various portions of the IP packet, followed by computation of the hash on the composite.

Input Selection:
The raw input is drawn from the first 20 bytes of the IP packet header and the first 8 bytes of the IP payload. If IP options are not used, the IP header has 20 bytes, and hence the two portions adjoin and comprise the first 28 bytes of the IP packet. We now use the raw input as 4 32-bit subportions of these 28 bytes. We specify the input by bit offsets from the start of IP header or payload.

f1 = bits 32 to 63 of the IP header, comprising the IP identification field, flags, and fragment offset.

f2 = bits 96 to 127 of the IP header, the source IP address.

f3 = bits 128 to 159 of the IP header, the destination IP address.

f4 = bits 32 to 63 of the IP payload. For a TCP packet, f4 comprises the TCP sequence number followed by the message length. For a UDP packet f4 comprises the UDP checksum.

Hash Computation:
The hash is computed from f1, f2, f3 and f4 by a combination of XOR (^), right shift (>>) and left shift (<<) operations. The intermediate quantities h1, v1, v2 are 32-bit numbers.

```
1.    v1 = f1 ^ f2;
2.    v2 = f3 ^ f4;
3.    h1 = v1 << 8;
4.    h1 ^= v1 >> 4;
5.    h1 ^= v1 >> 12;
6.    h1 ^= v1 >> 16;
7.    h1 ^= v2 << 6;
8.    h1 ^= v2 << 10;
9.    h1 ^= v2 << 14;
10.   h1 ^= v2 >> 7;
```

The output of the hash is the least significant 16 bits of h1.

A.2 BOB Hash Function

The BOB Hash Function is a Hash Function designed for having each bit of the input affecting every bit of the return value and using both 1-bit and 2-bit deltas to achieve the so called

avalanche effect [Jenk97]. This function was originally built
for hash table lookup with fast software implementation.

Input Parameters:
The input parameters of such a function are:
- the length of the input string (key) to be hashed, in bytes.
The elementary input blocks of Bob hash are the single bytes,
therefore no padding is needed.
- an init value (an arbitrary 32-bit number).

Built in parameters:
The Bob Hash uses the following built-in parameter:
- the golden ratio (an arbitrary 32-bit number used in the hash
function computation: its purpose is to avoid mapping all zeros
to all zeros);

Note: the mix sub-function (see mix (a,b,c) macro in the
reference code in 3.2.4) has a number of parameters governing
the shifts in the registers. The one presented is not the only
possible choice.

It is an open point whether these may be considered additional
built-in parameters to specify at function configuration.

Output.
The output of the BOB function is a 32-bit number. It should be
specified:
- A 32 bit mask to apply to the output
- The selection range as a list of non overlapping intervals
[start value, end value] where value is in [0,2^32]

Functioning:
The hash value is obtained computing first an initialization of
an internal state (composed of 3 32-bit numbers, called a, b, c
in the reference code below), then, for each input byte of the
key the internal state is combined by addition and mixed using
the mix sub-function. Finally, the internal state mixed one last
time and the third number of the state (c) is chosen as the
return value.

```
typedef unsigned long int  ub4;   /* unsigned 4-byte quantities
*/
typedef unsigned      char ub1;   /* unsigned 1-byte quantities
*/

#define hashsize(n) ((ub4)1<<(n))
#define hashmask(n) (hashsize(n)-1)
```

```
   /* -------------------------------------------------------
     mix -- mix 3 32-bit values reversibly.
     For every delta with one or two bits set, and the deltas of
   all three high bits or all three low bits, whether the original
   value of a,b,c is almost all zero or is uniformly distributed,
     * If mix() is run forward or backward, at least 32 bits in
   a,b,c have at least 1/4 probability of changing.
     * If mix() is run forward, every bit of c will change between
   1/3 and 2/3 of the time.  (Well, 22/100 and 78/100 for some 2-
   bit deltas.) mix() was built out of 36 single-cycle latency
   instructions in a structure that could supported 2x parallelism,
   like so:
           a -= b;
           a -= c; x = (c>>13);
           b -= c; a ^= x;
           b -= a; x = (a<<8);
           c -= a; b ^= x;
           c -= b; x = (b>>13);
           ...
   Unfortunately, superscalar Pentiums and Sparcs can't take
   advantage of that parallelism.  They've also turned some of
   those single-cycle latency instructions into multi-cycle latency
   instructions

   ------------------------------------------------------------*/

     #define mix(a,b,c)  \
     { \
       a -= b; a -= c; a ^= (c>>13); \
       b -= c; b -= a; b ^= (a<<8); \
       c -= a; c -= b; c ^= (b>>13); \
       a -= b; a -= c; a ^= (c>>12);  \
       b -= c; b -= a; b ^= (a<<16); \
       c -= a; c -= b; c ^= (b>>5); \
       a -= b; a -= c; a ^= (c>>3);  \
       b -= c; b -= a; b ^= (a<<10); \
       c -= a; c -= b; c ^= (b>>15); \
     }

     /* ------------------------------------------------------------
   hash() -- hash a variable-length key into a 32-bit value
   k       : the key (the unaligned variable-length array of bytes)
   len     : the length of the key, counting by bytes
   initval : can be any 4-byte value
   Returns a 32-bit value.  Every bit of the key affects every bit
   of the return value.  Every 1-bit and 2-bit delta achieves
   avalanche. About 6*len+35 instructions.
```

```
   The best hash table sizes are powers of 2.  There is no need to
   do mod a prime (mod is sooo slow!).  If you need less than 32
   bits, use a bitmask.  For example, if you need only 10 bits, do
   h = (h & hashmask(10));
   In which case, the hash table should have hashsize(10) elements.

   If you are hashing n strings (ub1 **)k, do it like this:
   for (i=0, h=0; i<n; ++i) h = hash( k[i], len[i], h);

   By Bob Jenkins, 1996.  bob_jenkins@burtleburtle.net.  You may
   use this code any way you wish, private, educational, or
   commercial.  It's free. See
   http://burtleburtle.net/bob/hash/evahash.html
   Use for hash table lookup, or anything where one collision in
   2^^32 is acceptable.  Do NOT use for cryptographic purposes.
    ----------------------------------------------------------- */

     ub4 bob_hash(k, length, initval)
     register ub1 *k;        /* the key */
     register ub4  length;   /* the length of the key */
     register ub4  initval;  /* an arbitrary value */
     {
        register ub4 a,b,c,len;

        /* Set up the internal state */
        len = length;
        a = b = 0x9e3779b9; /*the golden ratio; an arbitrary value
   */
        c = initval;         /* another arbitrary value */

   /*------------------------------------ handle most of the key */

        while (len >= 12)
        {
           a += (k[0] +((ub4)k[1]<<8) +((ub4)k[2]<<16)
   +((ub4)k[3]<<24));
           b += (k[4] +((ub4)k[5]<<8) +((ub4)k[6]<<16)
   +((ub4)k[7]<<24));
           c += (k[8] +((ub4)k[9]<<8)
   +((ub4)k[10]<<16)+((ub4)k[11]<<24));
           mix(a,b,c);
           k += 12; len -= 12;
        }

        /*---------------------------- handle the last 11 bytes */
        c += length;
        switch(len)       /* all the case statements fall through*/
        {
```

```
        case 11: c+=((ub4)k[10]<<24);
        case 10: c+=((ub4)k[9]<<16);
        case 9 : c+=((ub4)k[8]<<8);
           /* the first byte of c is reserved for the length */
        case 8 : b+=((ub4)k[7]<<24);
        case 7 : b+=((ub4)k[6]<<16);
        case 6 : b+=((ub4)k[5]<<8);
        case 5 : b+=k[4];
        case 4 : a+=((ub4)k[3]<<24);
        case 3 : a+=((ub4)k[2]<<16);
        case 2 : a+=((ub4)k[1]<<8);
        case 1 : a+=k[0];
          /* case 0: nothing left to add */
        }
        mix(a,b,c);
        /*-------------------------------- report the result */
        return c;
     }
```