Internet Draft
Document: <draft-ietf-psamp-sample-tech-07.txt>
Expires: January 2006

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

July 2005

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 Measurement Processes and for reporting the technique in use to a Collector.

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 by 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, transport 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, Filtering, or aggregation. 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

[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-MIB]: "Definitions of Managed Objects for Packet Sampling" describes the PSAMP Management Information Base.

[PSAMP-PROTO]: "Packet Sampling (PSAMP) Protocol Specifications" specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Colleting 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 to be consistent with terms used in [IPFIX-REQ]. The relationship between some PSAMP and IPFIX terms is described in [PSAMP-FW].

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 subset of the Observed Packet Stream that flows past some specified point within the measurement process. An example of a Packet Stream is the output of the selection process.

* Packet Stream

A packet stream denotes a set of packets that flows past some specified point within the measurement 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 [IPFIX-REQ].

* 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.

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 and/or the reporting 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 pseudorandom 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.

3.3 Reporting Process

* Reporting Process:

A Reporting Process creates a Report Stream on packets selected by a Selection Process, in preparation for export. The input to the Reporting Process comprises that information available to the Selection Process per selected packet, specifically:

(i) the selected packet's content;

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

(iii) any Selection State maintained by the inputting Selection Process, reflecting any modifications to the Selection State made during selection of the packet.

* Packet Reports

Packet Reports comprise a configurable subset of a packet's input to the reporting 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 and of the Reporting Process.

* Report Stream:

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

3.4 Measurement Process

* Measurement Process

A Measurement Process is the composition of a Selection Process that takes the Observed Packet Stream as its input, followed by a Reporting Process.

3.5 Exporting Process

* Exporting Process:

An Exporting Process sends, in the form of Export Packet, the output of one or more Measurement 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 a 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 Measurement Process and an Exporting Process. Typically, corresponding Observation Point(s), Measurement 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 Measurement 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. Examples include field match Filtering, and Hash-based Selection.

* 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.

* Content-independent Sampling

A Sampling operation that does not use Packet Content (or quantities derived from it) as the basis for selection is called a Content-independent Sampling operation. Examples include systematic Sampling, and uniform pseudorandom Sampling driven by a pseudorandom 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.

* Content-dependent Sampling

  A Sampling operation where selection is dependent on Packet Content is called a Content-dependent Sampling operation. Examples include pseudorandom selection according to a probability that depends on the contents of a packet field. Note that this is not a filter, because the selection is not deterministic.

* 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 deterministic map from the Hash Domain into the Hash Range.

* Hash Selection Range

  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

  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 pseudorandom 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

Field match filtering	X	X	Filtering
Hash Function	X	X	Filtering
Router state filtering	X	(X)	Filtering

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 renormalization 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 [IPFIX-REQ], 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). See [MolFl03] for a more detailed description of the motivations for this type of Sampling and the impact on the IPFIX metering.

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 three Filtering techniques. The first two (Match Filtering and Hashing Filtering) are stateless, and therefore can make their decision based on the analysis of portion of the packet only. The other (router state Filtering) requires to access state information after the analysis of part of the packet and is therefore more complex: its usage makes sense only in particular circumstances, as described below.

6.1 Field Match Filtering

We here define a basic Filtering schemes based on the IPIFIX flow definition. With this method a packet is selected if a specific field in the packet equals a predefined value. Possible filter fields are all IPFIX flow attributes specified in [IPFIX-INFO]. Further fields can be defined by vendor specific extensions.
A packet is selected if Field=Value. Masks and ranges are only

supported to the extend to which [IPFIX-INFO] allows them e.g.

by providing explicit fields like the netmasks for source and destination addresses. AND operations are possible by concatenating filters. 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.

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 indeed 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 field match filter or a simple combination of these.
Hash-based selection has mainly two types of usage: it offers a way to approximate random Sampling by using packet content to generate pseudorandom variates, and a way to consistently select subsets of packets that share a common property (e.g. at different Observation Points).

In the following subsections we give more details about them. However, both usages require that the Hash Functions has two statistical properties.

First, the Hash Function h must 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.

The second desirable property depends more closely on the statistics of the content c. In applications, the content c comprises a number of distinct fields, c1 ... cm, e.g. source and destination IP Address, IP identification, and TCP/UDP port numbers (if present) for a packet. With a Hash Function satisfying the first properties above, selection decisions will appear uncorrelated with the contents of any individual field, if the complementary fields are (i) sufficiently variable themselves, and (ii) sufficiently uncorrelated with cj.

6.2.1 Application Examples for Hash-based Selection

6.2.1.1 Approximation of Random Sampling

Although pseudorandom 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 renormalized) is a pseudorandom variate in the interval [0,1]. Other schemes may use packet fields in iterators for pseudorandom 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 pseudorandom 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 pseudorandom 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 pseudorandom number generator.

6.2.1.2 Trajectory Sampling and Consistent packet selection.

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 quasirandomly 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 [DuGr01], [DuGeGr02] 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.

6.2.2 Guarding Against Pitfalls and Vulnerabilities

A concern for Hash-based Selection is whether some large set of related packets could have an Attained Sampling Fraction significantly different from the Configured Sampling Fraction, 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. Examples of such are CRC32 and Hash Functions based on modular arithmetic, see 6.4 in [Knuth98]. The statistical properties of candidate Hash Functions need to be evaluated, preferably on packet traces before adoption for hash-based Sampling

Hash-based selection could be overloaded or evaded by an attacker if the Hash Function and the selection range are both known. A service provider could build a first defense keeping the Hash Selection Range S private. Then, an attacker could not determine whether a crafted packet is selected, but would still know that a crafted set of packets all with the same hash is either all selected or all not selected. A stronger defense is to employ a parametrizable Hash Function and keep the parameter private. Without knowledge of the parameter, a set of packets with common hash value cannot be constructed. Examples of parameters are the initial vector in CRC32, and moduli in hashes based on modular arithmetic.

6.2.3 Recommended Hash Functions

We here indicate some Hash Functions that can be used for packet selection. The description of the IPSX and BOB Hash Functions can be found in the appendix. The CRC-32 function is described

in [crc32]. The comparison of hash-functions with regard to

collision probability, the randomness of the packet selection (i.e. the uniformity of the distribution of values in the Hash Range) and the speed of the functions is described in [MoND05]. Although the uniformity has been checked for different traffic traces, results cannot be generalized to arbitrary traffic. Since the hash-based selection is a deterministic function on the packet content, it can always be biased towards packets with specific attributes. Furthermore, please note that all Hash Functions were evaluated only for IPv4.

6.2.3.1 Hash Functions Suitable for Packet Selection

For hash-based packet selection, the most important requirements for the Hash Function are high execution speed, because the selection must operate at line rate, and the uniformity of the distribution of values in the Hash Range in order to provide a good emulation of a random packet selection.

The IPSX function is simple and easy to implement. The evaluation results showed that the IPSX function is faster than the BOB function. Nevertheless, IPSX is limited to 16 bytes input. All investigated Hash Functions showed a poor uniformity if only 16 bytes are used as input. Therefore, if a hash-based packet selection is implemented, the BOB function SHOULD be used for packet selection operations. Other functions, like IPSX, MAY be used.

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.

6.2.3.2 Input Bytes for the BOB Hash Function for IPv4

All investigated Hash Functions showed a poor uniformity if only 16 bytes are used as input. If the input consists of 20 bytes which include parts of the IP packet payload (i.e. everything following the IP header) the uniformity was significantly improved. 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.

Please note that the uniformity increases with the number of additional input bytes from the IP payload. The default setting is to use 4 bytes from the IP payload. Using less bytes can lead to a significant bias in the selection.

6.2.3.3 Input Bytes for the BOB Hash Function for IPv6

All investigated Hash Functions 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 was not evaluated in the tests mentioned above. 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. It is possible that the selection is biased towards packets with specific attributes.

6.2.3.4 Hash Functions Suitable 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.

For this purpose also the BOB function is recommended. 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.

6.3 Router State Filtering

This class of filters selects a packet on the basis of router state conditions. The following list gives examples for such conditions. Conditions can be combined with AND operators.

- 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 BGP AS [RFC1771] equals a specified value or lies within a given range
- Destination BGP AS equals a specified value or lies within a given range

Router architectural considerations may preclude some information concerning the packet treatment, e.g. routing state, being available at line rate for selection of packets. However, if selection not based on routing state has reduced down from line rate, subselection based on routing state may be feasible.

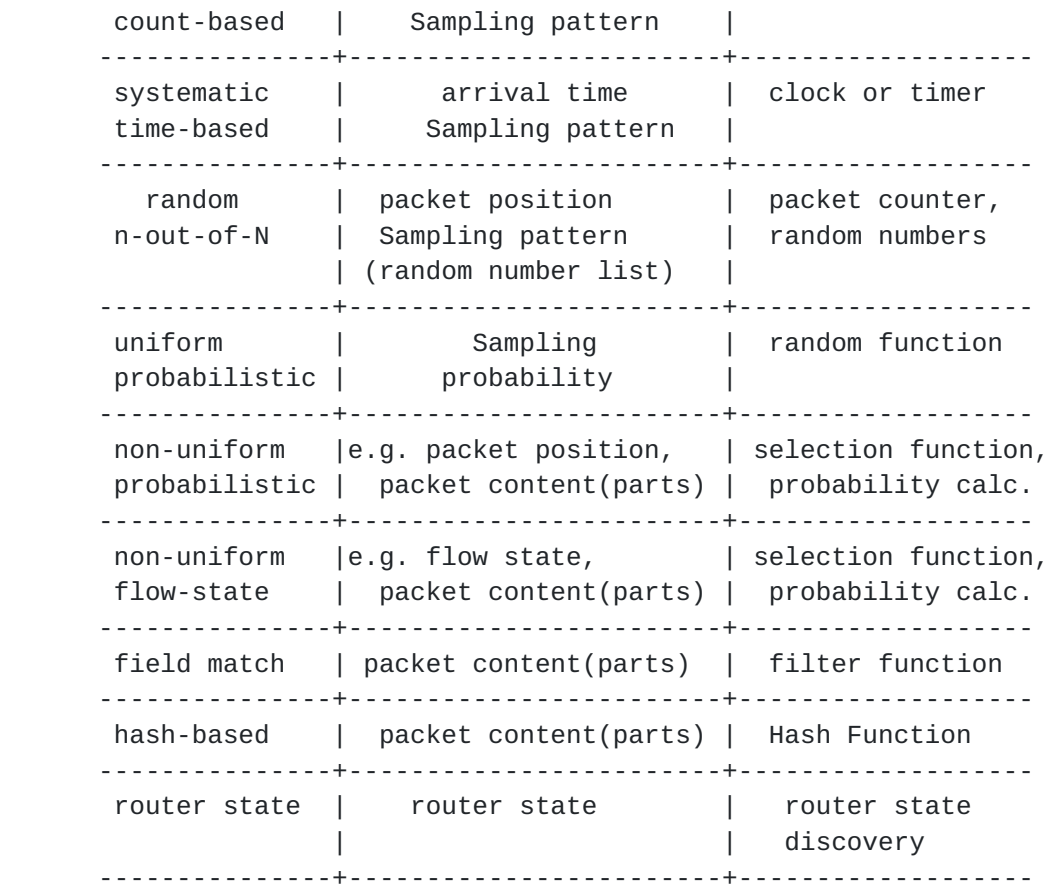
7. Parameters for the Description of Selection Techniques

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 | packet position | packet counter



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
- ASSOCIATIONS

Where:

SELECTOR_ID:
Unique ID for the packet sampler, calculated as combination of the ASSOCIATIONS and a local ID.

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 can be found in [MolFl03](see also section 5.2.2.4)

ASSOCIATIONS
The ASSOCIATIONS field describes the Observation Point and optionally the IPFIX processes to which the packet Selector is associated. If no IPFIX process is specified, the selector is applied to all IPFIX processes on the observation point. The STREAM ID denotes the origin of the data stream that is input to the selection function. It can be the Observation Point directly or the ID of another Selector. With this it is possible to define combined schemes. If the STREAM ID contains IDs from other Selectors, one can derive the original Observation Point from the Selector definitions of these specified Selectors.

Values: <STREAM ID, IPFIX Metering process ID, IPFIX Exporting process ID, IDs of other associated processes>
With STREAM ID: Observation point ID AND List of SELECTOR_IDs

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
- ASSOCIATIONS

Where:

SELECTOR_ID:
Unique ID for the packet filter. The ID can be calculated under consideration of the ASSOCIATIONS 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 [IPFIX-INFO])
- Value (type in accordance to [IPFIX-INFO])

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 then 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

ASSOCIATIONS
The ASSOCIATIONS field describes the Observation Point and optionally the IPFIX processes to which the packet Selector is associated. If no IPFIX process is specified, the Selector is applied to all IPFIX processes on the observation point. The STREAM ID denotes the origin of the data stream that is input to the selection function. It can be the Observation Point directly or the ID of another Selector. With this it is possible to define combined schemes. If the STREAM ID contains IDs from other Selectors, one can derive the original Observation Point from the Selector definitions of these specified Selectors.

Values: <STREAM ID, IPFIX Metering process ID, IPFIX Exporting process ID, IDs of other associated processes>
With STREAM ID: Observation point ID AND List of SELECTOR_IDs

8. Composite Techniques

Composite schemes are realized by using the STREAM ID in the information models. The STREAM ID denotes from which Selectors the input stream originates. If multiple stream IDs are given, this means that the Selector operates on the packet stream simply resulting from the time superposition of the output of all the listed filters and samplers. 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. With this 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

Malicious users or attackers may wish to hide packets from service providers or network operators. For instance if packet

Selectors are used for accounting or intrusion detection

applications, users may want to prevent certain packets from being selected. If a deterministic Sampling scheme is used or a selection scheme that takes packet content into account, the user can shape or send packets in a way that they are less likely to be selected (see also section 6.2.2). Even if the selection function is unknown to the user, some insight into the function can be obtained by performing experiments with different packet sequences. This has to be taken into account when choosing an appropriate packet selection technique.

Further security threats can occur if the configuration of Sampling parameters or the communication of Sampling parameters to the application is corrupted. This document only describes Sampling schemes that can be used for packet selection. It neither describes a mechanism how those parameters are configured nor how these parameters are communicated to the application. Therefore the security threats that originate from this kind of communication cannot be assessed with the information given in this document.

10. Acknowledgements

We would like to thank the PSAMP group, especially Benoit Claise and Stewart Bryant, for fruitful discussions and for proofreading the document.

11. Normative References

[PSAMP-FW] Nick Duffield (Ed.): A Framework for Packet Selection and Reporting, RFC XXXX [currently Internet Draft draft-ietf-psamp-framework-08, work in progress, October 2004]

[PSAMP-MIB] T. Dietz, B. Claise: Definitions of Managed Objects for Packet Sampling, RFC XXXX. [Currently Internet Draft, draft-ietf-psamp-mib-03.txt, work in progress, July 2004.]

[PSAMP-PROTO] B. Claise (Ed.): Packet Sampling (PSAMP) Protocol Specifications, RFC XXXX. [Currently Internet Draft draft-ietf-psamp-protocol-01.txt, work in progress, February 2004.]

[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-02, July 2004]

[IPFIX-INFO] J. Meyer, J. Quittek, S. Bryant: Information Model for IP Flow Information Export, RFC XXXX [Currently Internet Draft, draft-ietf-ipfix-info-04, July 2004]

[IPFIX-REQ] J. Quittek, T. Zseby, B. Claise, S. Zander: Requirements for IP Flow Information Export, RFC 3917, October 2004

12. 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

[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

[CoGi98] I. Cozzani, S. Giordano: Traffic Sampling Methods for end-to-end QoS Evaluation in Large Heterogeneous Networks. Computer Networks and ISDN Systems, 30 (16-18), September 1998.

[crc32] R. Braden, D. Borman, C. Partridge: Computing the Internet Checksum, RFC 1071, Sep. 1988 (updated by RFCs 1141 and 1624)

[DuGeGr02] 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,

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

[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.

[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

[Knuth98] Donald E. Knuth: The Art of Computer Programming, Volume 3: Searching and Sorting, Addison Wesley, 1998

[MolFl03] M.Molina: Flow selection support in IPFIX, Internet Draft <draft-molina-flow-selection-00.txt>, work in progress, October 2003.

[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

[RFC1771] Rekhter, Y. and T. Li: A Border Gateway Protocol 4 (BGP-4), RFC 1771, March 1995.

[Zseb03] T. Zseby: Stratification Strategies for Sampling-based Non-intrusive Measurement of One-way Delay. Proceedings of Passive and Active Measurement Workshop (PAM 20003), La Jolla, CA, USA, pp. 171-

13. Authors' Addresses

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

14. Intellectual Property Statement

property rights claimed in regard to some or all of the

specification contained in this document. For more information, see http://www.ietf.org/ietf/IPR/att-ipr-draft-ietf-psamp-framework.txt

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

15. Copyright Statement

16. Disclaimer

17. Appendix: Hash Functions

17.1 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.

17.2 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;
     }
```