Internet Draft
draft-zseby-ipfix-applicability-00.txt
Expires: November 2002

Tanja Zseby
FhI FOKUS
Reinaldo Penno
Nortel Networks
Nevil Brownlee
CAIDA

June 2002

IPFIX Applicability

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Abstract

This document describes how various applications can use the IP Flow Information Export (IPFIX) protocol. It furthermore shows how the IPFIX framework relates to other architectures and frameworks.

1. Introduction

The IPFIX protocol defines how IP Flow information can be exported from routers, measurement probes or other devices. It is intended to provide input for various applications. This document describes how applications can use the IPFIX protocol. Furthermore the relationship of IPFIX to other frameworks and architectures are described.

2. Applications of IPFIX

2.1.Accounting with IPFIX

Usage based accounting is one of the major application for which the IPFIX protocol has been developed. IPFIX flow records contain the number of transferred bytes per flow. This information is an essential input for usage-based accounting.

In order to realize usage-based accounting with IPFIX the flow definition has to be chosen in accordance to the tariff model. A tariff can for instance be based on individual end-to-end stream. Accounting in such a scenario can be realized for instance with a flow definition determined by the quintuple that consists of source address, destination address, protocol and portnumbers. Another example is a class-dependent tariff (e.g. in a DiffServ networks). For this flows could be distinguished just by DiffServ codepoint (DSCP) and source address.

2.2.Intrusion detection with IPFIX

Intrusion detection systems (IDS) monitor and control security incidents. A typical IDS system includes components like sensor, event collector, and management stations. Sensors monitor network and system traffic for attacks and other security-related events. Sensors respond to and notify the administrator about these events as they occur. Event collectors are a middle-tier component responsible for

transmitting events from sensors to the console and database. The management component serves the following purposes:

- visually monitors events (with a console)
- collects data from sensors (with one or more event collectors)
- stores data from sensors (in a database)

With IPFIX, events of interest can be reported to the sensor either by the collecting process or directly by the exporting process. It depends on the scenario and the events of interest which solution is better. Getting information directly from the exporting process has the advantage that the sensor gets the information faster. It does not need to wait for collector processing time or until the collector has all relevant data.
Getting the information from a collector allows to correlate data from different exporting processes (e.g. from different routers) to get a better picture about what is going on in the network.

2.3.QoS Monitoring with IPFIX.

The performance of QoS monitoring is one target application for using the IPFIX protocol. QoS monitoring is the passive observation of transmission quality for single flows or traffic aggregates in the network. One example of its usefulness is the validation of QoS guarantees in service level agreements. Some QoS metrics require the correlation of data from multiple measurement points. For this the clock of the involved exporting devices need to be synchronized. Furthermore such measurements would benefit from post-processing functions (e.g. packet ID generation) at the exporter and/or collector. This section describes how the monitoring of different metrics can be performed with IPFIX. The following metrics are considered: round trip time, one-way-delay, loss and delay variation.

2.3.1.Measurement of Round-trip-time (RTT) with IPFIX

The passive measurement of round-trip-times (RTT) can be performed by using packet pair matching techniques as described in [Brow00]. For the measurements, request/response packet pairs from protocols like DNS, ICMP, SNMP or TCP (syn/syn-ack, data/ack) are utilized to passively observe the RTT [Brow00]. As always for passive measurements this only works if the required traffic of interest is actually present in the network. In order to use this measurement technique, the IPFIX metering process needs to measure both directions. A classification of the protocols mentioned above has to be done. That means parts of the transport header are used for the classification. Since a differentiation of flows in accordance to the transport header is one of the requirements for IPFIX, such classification can be performed without extensions. Nevertheless, the meter needs to recognize request and response packets for the given protocols and therefore needs to look further into the packets. The capability to do this analysis is not part of the IPFIX requirements

but can be achieved by optional extensions to the classification process. The exporting device needs to assign a timestamp for the arrival of the packets. The calculation of the RTT can be done directly at the exporter or at the collector. In the first case IPFIX would transfer the calculated RTT to the collector. In the second case IPFIX needs to send the observed packet types and the timestamps to the collector.

2.3.2.Measurement of One-way-delay (OWD) with IPFIX

Passive one-way-delay measurements require the collection of data at two measurement points. It is necessary to recognize packets at the second measurement point to correlate packet arrival events from both points. This can be done by capturing packet header and parts of the packet that can be used to recognize the same packet at the subsequent measurement point.
To reduce the amount of measurement data a unique packet ID can be calculated from the header and part of the content e.g. by using a CRC or hash function [GrDM98, DuGr00, ZsZC01].Since IPFIX is not targeted at packet capturing these functionalities do not need to be supported by a standard IPFIX meter. Nevertheless, in some scenarios it might be sufficient to calculate a packet ID under consideration of header fields including datagram ID and maybe sequence numbers (from transport protocols) without looking at parts of the packet content. If packet IDs need to be unique only for a certain time interval or a certain amount of packet ID collisions is tolerable this can be a sufficient solution.

2.3.3.Measurement of Loss with IPFIX

Passive loss measurements for single flows can be performed at one measurement point by using sequence numbers that are present in higher layer protocols. This requires the capturing of the sequence numbers of subsequent packets of the observed flow by the IPFIX metering process. An alternative to this is to perform a two-point measurement as described above and just consider packets as lost that do not arrive at the second measurement point in a given maximum time frame.

2.3.4.Measurement of delay variation with IPFIX

Delay variation is defined as the difference of one-way-delay values for selected packets [DeCi01]. Therefore this metric can be calculated by performing passive measurement of one-way-delay for subsequent packets (e.g. of a flow) and then calculating the differences.

2.3.5.Sampling for QoS Monitoring

Since QoS monitoring can produce an overwhelming amount of measurement data, methods such as aggregation of results, and

sampling would greatly increase the efficiency of the collection and analysis process. Sampling methods can be grouped according to the sampling strategy (systematic, random or stratified) and the trigger that starts a sampling interval (count-based, time-based or packet-content-based) [ClPB93]. Sampling can also be used as a method to dynamically reduce resource consumption if the meter is overloaded. Then the IPFIX meter can switch to a sampling method if too many packets have to be observed. Since the expected estimation error heavily depends on the deployed sampling strategy, the application that receives the data needs to be aware of the sampling scheme and the parameters in use. Therefore it is important that the IPFIX exporter informs the collector precisely about the used sampling strategy. This is especially important if the metering process dynamically invokes sampling.

3. Relation of IPFIX to other frameworks and protocols

3.1.IPFIX and AAA

AAA defines a protocol and architecture for authentication, authorization and accounting for service usage. The DIAMETER protocol is used for AAA communication for network access services (Mobile IP, NASREQ, and ROAMOPS). The AAA architecture [RFC2903] provides a framework for extending the AAA support also for other services. DIAMETER defines the exchange of messages between AAA entities, e.g. between AAA clients at access devices and AAA servers and among AAA servers. It is used also for the transfer of accounting records. Usage-based accounting requires measurement data from the network. IPFIX defines a protocol to export such data from routers, measurement probes and other devices.

The provisioning of accounting with IPFIX can be realized without an AAA infrastructure. The collector can directly forward the measurement information to an accounting application Nevertheless, if an AAA infrastructure is in place, IPFIX can provide the input for the generation of accounting records and several features of the AAA architecture can be used. Features include the mapping of a user ID to the flow information (by using authentication information), the generation of DIAMETER accounting records and the secure exchange of accounting records between domains with DIAMETER. Three possibilities to connect IPFIX and AAA can be distinguished:

3.1.1.Connecting via an AAA Client

One possibility to connect IPFIX and AAA is to run an AAA client on the IPFIX collector. This client can generate DIAMETER accounting messages and send them to an AAA server. The mapping of the flow information to a user ID can be done in the AAA server by using data from the authentication process. DIAMETER accounting messages can be sent to the accounting application or to other AAA servers (e.g. in roaming scenarios).

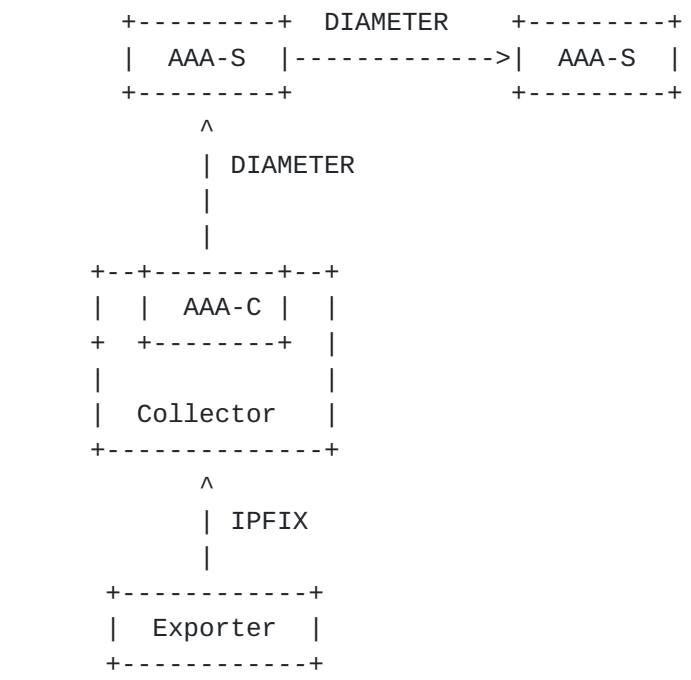
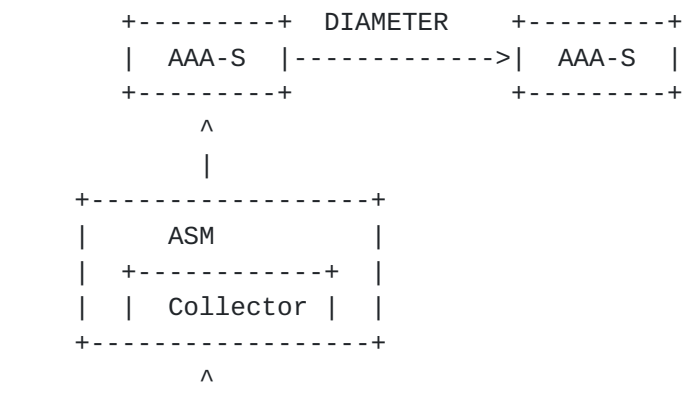


Figure 2: IPFIX collector connects to AAA server via AAA client

3.1.2.Connecting via an Application Specific Module (ASM)

Another possibility is to directly connect the IPFIX collector with the AAA server via an application specific module (ASM). Application specific modules have been proposed by the IRTF AAA architecture research group (AAARCH) in [RFC2903]. They act as an interface between AAA server and service equipment. In this case the IPFIX collector is part of the ASM. The ASM acts as an interface between the IPFIX protocol and the input interface of the AAA server. The ASM translates the received IPFIX data into an appropriate format for the AAA server. The AAA server then can add information about the user ID and generate a DIAMETER accounting record. This accounting record can be sent to an accounting application or to other AAA servers.



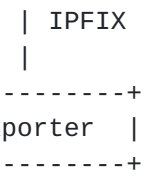


Figure 3: IPFIX connects to AAA server via ASM

3.2.IPFIX and RTFM

This section compares the Real-time Traffic Flow Measurement (RTFM) framework with the IPFIX framework.

3.2.1.Definition of 'flow'

RTFM and IPFIX both use the same definition of flow; a flow is a set of packets which share a common set of end-point address attribute values.A flow is therefore completely specified by that set of values, together with an inactivity timeout. A flow is considered to have ended when no packets are seen for at least the inactivity time.

RTFM flows are bidirectional, which has given rise to some confusion. At the simplest level, a flow information exporter may achieve this by maintaining two unidirectional flows, one for each direcion. To export bidirectional flow information, e.g. to- and from- packet counts, for a flow from A to B, the exporter has only to search its flow table to find the matching flow from B to A.

RTFM, however, takes bi-directionality a stage further, by including in the RTFM architecture [RFC 2722] a fully-detailed algorithm for realtime matching of the two directions of a flow. This was done for two reasons, to reduce the memory required to store each flow (common address attributes for each direction), and to allow for attributes which required fine detail for the two directions, e.g. short-term bit rate distributions [RFC 2724].
** So far there has been no suggestion that IPFIX should do this.

3.2.2.Configuration and Management

The RTFM architecture specifies a complete system for gathering flow information. It defines three entities,
- Meters are very similar to IPFIX exporters.
- Meter Readers are very similar to IPFIX collectors.
- Managers co-ordinate the activities of meters and meter readers, and download configuration to them.

Note that the whole RTFM system is asynchronous, many readers may collector flow data from a meter, and any reader may collect flow data from many meters.

Rulesets allow the user to specify which flows are of interest,

which are the source and destination ends of each flow, and what level of address granularity is required in the metered flows. For example, one may select all packets from 192.168/16, but build flow information for 192.168/24. RTFM selection is done by testing under masks, and the masks do not have to use consective ones from the left. Non-contiguous masks were considered important for handling some OSI protocols, but the need for that has diminished considerably.

The RTFM approach is based on RMON, in that if a user wants to collect flow data for some particular set of flows, this can be achieved by writing a ruleset, i.e. an SRL program [RFC 2723], to specify what flows are of interest, requesting a manager to download that ruleset to a meter, and requesting the manager to have a meter reader collect the flow data at specified intervals.

The details of how the manager communicates this information to meters and meter readers is not specified in the architecture. RTFM has a Meter MIB [RFC 2720], which is a standard which can be used to configure a meter, but nothing is said about how to configure a meter reader.

The extent to which IPFIX should specify how meters or exporters should be configured is, at this stage, an open question. Clearly a collector needs some way to be sure of what it's collecting, e.g. by receiving 'templates' from the meter.

RTFM and IPFIX both leave parts of the system unspecified. For RTFM flow data to be useful one must know the ruleset used to configure the meter, but a user can specify the ruleset. For IPFIX one knows what the data is from the templates, but we have yet to determine whether in-band configuration will be supported.

3.2.3.Data Model details

3.2.3.1.Count in one bucket

Within a ruleset, a packet may only be counted on one bucket, i.e. it may only be included in one flow. This means that the meter does not have to keep track of overlapping flows - if such aggregation is required, it must be done after the raw flow data has been read by a meter reader.

>From time to time one may wish to collect flow data for different levels of aggreation at the same time. RTFM allows a meter to run several rulesets at the same time, and meter readers must specify which rulesets they are collecting data from.

The 'count in one bucket' rule, together with the ability to run multiple rulesets, has proved very simple and effective in practice.

3.2.3.2.Counter wrapping

For its packet- and byte-count attributes RTFM uses continuously-incrementing 64-bit counters, which are never reset. This makes asynchronous meter reading easy, any reader simply has to remember its previous reading and compute the difference. The only caveat is that the meter should be read often enough to avoid situations when the counter has cycled more than once between readings.

3.2.3.3.Sampling issues

RTFM provides 1 out of N sampling as a configuration option, so that some metering interfaces may only process every Nth packet. The RTFM Arcitecture [RFC 2722] does not discuss the statistical implications of this, merely saying that users will need to satisfy themselves that sampling makes sense in their environment.

RTFM makes no provision for flow sampling. Recently there has been a lot of interest in flow sampling schemes which favour the 'most important' flows, perhaps we need to consider this for IPFIX.

3.2.4.Application/transport protocol

RTFM has a standards-track Meter MIB [RFC 2720], which can be used both to configure a meter and to read flow data from it. The MIB provides a way to read lists of attributes with a single Object Identifier (called a 'package'), which dramatically reduces the SNMP overhead for flow data collection. NeTraMet, a widely-used open-source RTFM implementation, uses SNMPv2C for configuration and data collection.

SNMP, of course, normally uses UDP as its transport protocol. Since RTFM requires a reliable flow data transport system, an RTFM meter reader must time out and resend unanswered SNMP requests.
Apart from being clumsy, this can limit the maximum data transfer rate from meter to meter reader. SNMP over TCP would be a better approach, but that is currently an IRTF project.

On the other hand, RTFM does not specify an application protocol in its architecture, leaving this as an implementation issue. For example, a team at IBM Research implemented a RTFM meter and meter reader in a single host, with the reader storing the flow data directly into a large database system. Simlarly, many NeTraMet user run the meter and meter reader on the same host system.
A need for high flow data rates highlights the need for careful systems design when building a flow data collection system. When data rates are high, and it is not possible to use a high level of aggregation, then it makes sense to have the collectors very close to their exporters. Once the data is safely on a dedicated host machine, large volumes of it can be moved using 'background' techniques such as FTP.

The RTFM architecture only specifies a pull model for getting data out of a meter. To implement push mode data transfer would require specification of triggers to indicate when data should be sent for each flow.

3.3.IPFIX Considerations for Middleboxes

A Middlebox is a network intermediate device that implements one or more of the middlebox services. Policy based packet filtering (a.k.a. firewall), Network address translation (NAT), Intrusion detection, Load balancing, Policy based tunneling and IPsec security are all examples of a middlebox function (or service). [MCFW]. For instance, a NAT middlebox is a middlebox implementing NAT service and a firewall middlebox is a middlebox implementing firewall service.

It is expected that the exporter in the IPFIX architecture will probably implement some form of middlebox service given its ubiquity today. Since some of these middlebox services might affect flow exportation and how the collector interprets them, there is a need to provide an analysis of these implications in relation to the IPFIX architecture and a set of recommendations.

The following sections provide a non-exhaustive analysis of middlebox services, its implications on the IPFIX architecture and a corresponding set of recommendations.

3.3.1.Firewall

Firewall is a policy based packet filtering middlebox function, typically used for restricting access to/from specific devices and applications. The policies are often termed Access Control Lists (ACLs)[MCFW].

The firewall middlebox service allows the exporter to explicitly drop packets based on some administrative policy. In this case, the exporter can take one of the following actions that will have a direct impact on the information provided by the collector to the user.

- Silently Discard

In this case the packet is discarded and no flow information record is sent to the collector.

- Discard and export flow

In this case the packet is discarded, and the flow information record is sent to the collector.

- Discard and export flow with discard notification

In this case the packet is discarded, and the flow information record is sent to the collector with an indication that the packet was discarded.

3.3.2.Network Address Translation

Network Address Translation is a method by which IP addresses are mapped from one address realm to another, providing transparent routing to end hosts. Transparent routing here refers to modifying end-node addresses en-route and maintaining state for these updates so that when a datagram leaves one realm and enters another, datagrams pertaining to a session are forwarded to the right end-host in either realm [NAT-TERM].

>From an exporter (middlebox) perspective, a NAT is composed of two flows, one from the client to the NAT middlebox and another from the NAT middlebox to the destination. Based on this fact, the exporter has several modes of operation, i.e., it can export the private realm flow, the public realm, or both. This is further constrained by the flavor of NAT implemented, meaning that in order for the exported information to be useful for the collector, sometimes the associated flows on the two realms need to be exported in the same flow record.

Although there are many flavors of address translation that lend themselves to different applications, this section will only address the IPFIX architecture implications of traditional NAT, bi-directional NAT and twice NAT.

3.3.2.1.Traditional NAT

Traditional NAT would allow hosts within a private network to transparently access hosts in the external network, in most cases. In a traditional NAT, sessions are unidirectional, outbound from the private network. This is in contrast with bi-directional NAT, which permits sessions in both inbound and outbound directions. A detailed description of traditional NAT may be found in section [NAT-TERM].

If the exporter is providing traditional NAT service and only the private realm flow is exported, only destination based information can be inferred from the collector. The reason for this is twofold. First, the collector will not be able to find the reverse flow (when applicable) associated with a private realm flow record, and second is that in the more general scenario the collector can be connected to several exporters providing NAT service and there might be overlapping private realm addresses between the networks connected to these exporters.

In a traditional NAT scenario the exporter SHOULD export private and public realm information in the same flow record or provide the collector with a unique key to associate the two if exported on

different flow records.

3.3.2.2.Bi-Directional NAT

With a bi-directional NAT, sessions can be initiated from hosts in the public network as well as the private network. Private network addresses are bound to globally unique addresses, statically or dynamically as connections are established in either direction. Detailed description of Bi-Directional may be found in section [NAT-TERM].

3.3.2.3.Twice NAT

Twice NAT is a variation of NAT in that both the source and destination addresses are modified by NAT as a datagram crosses address realms. This is in contrast to Traditional-NAT and Bi-Directional NAT, where only one of the addresses (either source or destination) is translated. Note, there is no such term as 'Once-NAT'. Detailed description of Bi-Directional may be found in section [NAT-TERM].

In the case of twice NAT the exporter MUST export private and public realm information in the same flow record or provide the collector with a unique key to associate the two if exported on different flow records.

3.3.3.Traffic Conditioners

A traffic conditioner may contain the following elements: meter, marker, shaper, and dropper. A traffic stream is selected by a classifier, which steers the packets to a logical instance of a traffic conditioner[DIFF-ARCH].

>From an IPFIX architecture perspective we are going to address marking, shaping and dropping services.

3.3.3.1.Marking

Diffserv packet markers set the DS field of a packet to a particular codepoint, adding the marked packet to a particular DS behavior aggregate. The marker may be configured to mark all packets which are steered to it to a single codepoint, or may be configured to mark a packet to one of a set of codepoints used to select a PHB in a PHB group, according to the state of a meter. When the marker changes the codepoint in a packet it is said to have "re-marked" the packet [DIFF-ARCH].

>From and IPFIX architecture perspective, the exporter can take one of the following actions when it needs to remark a packet.

- Unmarked flow

The exporter can export the flow before it was remarked. This mode of operation is strongly discouraged.

- Remarked flow

The exporter can export the flow after it was remarked.

- Unmarked and remarked flow.

The exporter can export the flow before and after it was remarked or export the flow before it was remarked and an indication of what was the DSCP after it was remarked.

3.3.3.2.Shapers

Shapers delay some or all of the packets in a traffic stream in Order to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold the delayed packets.

For an IPFIX perspective, since the discard of a packet by a shaper is not voluntary, no indication should be sent to the collector.

3.3.3.3.Droppers

Droppers discard some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is known as "policing" the stream. Note that a dropper can be implemented as a special case of a shaper by setting the shaper buffer size to zero (or a few) packets.

In a manner analogous to the middlebox firewall service, middlebox policing services also allow the exporter to explicitly drop packets based on some administrative policy.

The three possible export behaviors described for the firewall service when a packet needs to be dropped are also applicable to here, i.e., silent discard, discard and export flow, discard and export flow with discard notification.

3.3.4.Tunneling

The exporter can export the flows before and/or after they get tunneled. In the later case the encapsulated flow information might not be available due to confidentiality precautions such as those used by IPsec, or due to the fact the exporter lacks the necessary intelligence to inspect the encapsulated packet. Moreover, depending on where in the network the exporter is located, it might only be able to export flows associated with the tunnel, without visibility into the encapsulated packet.

Apart from what was said above, it should also be factored in the fact that flows exported before they get tunneled, will provide information about the private network sites, but not necessarily about the backbone, since the route taken by the packet it's now know at that point in time (and vice-versa).

3.3.5.VPNs

The term "Virtual Private Network" (VPN) refers to the communication between a set of sites, making use of a shared network infrastructure. Multiple sites of a private network may therefore communicate via the public infrastructure, in order to facilitate the operation of the private network. The logical structure of the VPN, such as addressing, topology, connectivity, reachability, and access control, is equivalent to part of or all of a conventional private network using private facilities [RFC2764] [VPN-2547BIS].

There are multiple flavors of VPNs, the one with more relevance to the IPFIX architecture is the PE-based-VPN. A PE-based VPN (or Provider Edge-based Virtual Private Network) is one in which PE devices in the SP network provide the VPN. This allows the existence of the VPN to be hidden from the CE devices, which can operate as if part of a normal customer network. A detailed discussion of VPNs can be found in [PPVPN-FR].

3.3.5.1.Layer 3 PE-based VPN

A layer 3 PE-based VPN is one in which the SP takes part in IP level forwarding based on the customer network's IP address space. In general, the customer network is likely to make use of private and/or non-unique IP addresses. This implies that at least some devices in the provider network needs to understand the IP address space as used in the customer network. Typically this knowledge is limited to the PE device [PPVPN-FR] which is directly attached to the customer.

In a layer 3 PE-based VPN the provider will need to participate in some aspects of management and provisioning of the VPNs, such as ensuring that the PE devices are configured to support the correct VPNs. This implies that layer 3 PE-based VPNs are by definition provider provisioned VPNs [PPVPN-FR].

In order to connect the different VPN sites belonging to the same VPN the SP uses a tunneling technique such as MPLS, L2TP or IPsec. These tunnels originate and terminate on PE devices.

One of the characteristics of a layer 3 PE-based VPNs is that they offload some aspects of VPN management from the customer network. >From an IPFIX architecture perspective, this means that the SP is the one that potentially will be providing the IPFIX service for the VPNs that it provides connectivity.

The exporter in Layer 3 PE-based VPN can be located on the customer's network, on the SP's backbone (P Router) or on the edge (PE router). The premise of this discussion is that the exporter is the one providing middlebox services, so in the case of VPNs we assume that the exporter is located in a PE device.

3.3.5.1.1.Overlapping Address Realms

In the case the exporter plays the role of a PE router [VPN-2547BIS] in a provider provisioned VPN scenario and has VPNs with overlapping private address realms, it can only provide useful non-conflicting information to the provider for intra-VPN traffic if it uses a technique that allows the collector to uniquely identify to which VPN the flow belongs.

Several techniques could be used to accomplish this goal. One of these techniques is to include the VPN Global unique identifier [VPN-ID] as one of the keys in the flow record.

In the case the exporter supports VPNs with overlapping private address realms, it MUST include the VPN-ID [VPN-ID] in the exported flow record.

3.3.5.2.Layer 2 PE-based VPN [TBD]

A layer 2 PE-based VPN is one in which the network is aware of the VPN, but does only layer 2 forwarding and signaling. This implies that the SP provisions and maintains layer 2 connectivity between CE devices [VPN-L2].

Forwarding options include MAC addresses (such as LAN emulation), use of point-to-point link layer connections (FR or ATM), multipoint-to-point (using MPLS multipoint to point LSPs), and point-to-multipoint (e.g. ATM VCCs).

For a layer 2 PE-based VPN, the PE device may be a router, LSR, or IP switch. From the CE's perspective, the PE will be operating as a switch.

4. Security Consideration

This document describes the usage of IPFIX in various scenarios. Currently only the IPFIX target applications (accounting, QoS monitoring, traffic profiling, traffic engineering and intrusion detection) are addressed. The security requirements for those applications are already addressed in the IPFIX requirements draft. These requirements must be considered for the selection and specification of the IPFIX protocol. The IPFIX extensions proposed in this document do not induce further security hazards.

The second section of this document describes how IPFIX can be used in combination with other frameworks. New security hazards can arise

when two individually secure frameworks are combined. For the combination of AAA with IPFIX an ASM or an IPFIX collector can function as transit point for the messages. It has to be ensured that at this point the applied security mechanisms (e.g. encryption of messages) are maintained.

6. References

[QuZC02] J. Quittek ,et. Al "Requirements for IP Flow Information Export ", (work in progress) ,Internet Draft, Internet Engineering Task Force, <draft-ietf-ipfix-reqs-01.txt>, February 2002

[Wood02]M. Wood ,et al.," Intrusion Detection Message Exchange Requirements",(work in progress), Internet Draft, Internet Engineering Task Force, draft-ietf-idwg-requirements-06,February 2002.

[Awdu02] Daniel O. Awduche, et. al.," Overview and Principles of Internet Traffic Engineering", (work in progress), Internet Draft, Internet Engineering Task Force, draft-ietf-tewg-principles-02.txt, May 2002

[Brow00] Nevil Brownlee: Packet Matching for NeTraMet Distributions, http://www2.auckland.ac.nz/net//Internet/rtfm/meetings/47-adelaide/pp-dist/

[DeCi01] C. Demichelis, P. Cimento: IP Packet Delay Variation Metric for IPPM, <draft-ietf-ippm-ipdv-08.txt>, November 2001

[RFC2680] G. Almes, S. Kalidindi, M. Zekauskas: A One-way Packet Loss Metric for IPPM, September 1999

[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

[GrDM98] Ian D. GRAHAM, Stephen F. DONNELLY, Stele MARTIN, Jed MARTENS, John G. CLEARY: Nonintrusive and Accurate Measurement of Unidirectional Delay and Delay Variation on the Internet, INET'98, Geneva, Switzerland, 21-24 July, 1998

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

[RFC2679] G. Almes, S. Kalidindi, M. Zekauskas: A One-way Delay Metric for IPPM, Request for Comments: 2679, September 1999

[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

[MCFW] Srisuresh, S. et al. "Middlebox Communication Architecture and framework," work in progress. October 2001.

[NAT-TERM] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[NAT-TRAD] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[PPVPN-FR] Callon, R., Suzuki, M., et al. "A Framework for Provider Provisioned Virtual Private Networks ", work in progress, <draft-ietf-ppvpn-framework-03.txt>, January 2002.

[VPN-L2] Rosen, E., "An Architecture for L2VPNs," Internet-draft <draft-ietf-ppvpn-l2vpn-00.txt>, July 2001.

[RFC2475] Black, D., Blake, S., Carlson, M., Davies, E., Wang, Z. and **W. Weiss, "An Architecture for Differentiated Services", RFC 2475,** December 1998.

[RFC2903] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence, "Generic AAA Architecture", RFC 2903, August 2000

7. Acknowledgements

We would like to thank the following persons for their contribution, discussion on the mailing list and valuable comments:

Robert Loewe

8. Author's Addresses

Tanja Zseby
Fraunhofer Institute for Open Communication Systems(FOKUS)
Kaiserin-Augusta-Allee 31
10589 Berlin
Germany
Phone: +49 30 3463 7153
Email: zseby@fokus.fhg.de

Reinaldo Penno
Nortel Networks, Inc.
2305 Mission College Boulevard

Building SC9-B1240
Santa Clara, CA 95134
Email: rpenno@nortelnetworks.com

Nevil Brownlee
CAIDA (UCSD/SDSC)
9500 **Gilman Drive**
La Jolla, CA 92093-0505
Phone : +1 858 534 8338
Email : nevil@caida.org

9. Full Copyright Statement