IPFIX Working Group
Internet Draft
Expiration Date: December 2002

EDITORS: K.C. Norseth
Consultant
Ganesh Sadasivan
Cisco Systems, Inc.
June 2002

Architecture Model for IP Flow Information Export

draft-ietf-ipfix-architecture-02.txt

Status of this Memo

Abstract

This memo defines the architecture, for the export of measured IP flow information out of an IPFIX device to a collector, per the requirements defined in [2].

Conventions used in this document

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

Table of Contents

1. Introduction

There are several applications e.g., Usage-based Accounting, Traffic Profiling, Traffic engineering, Attack/Intrusion Detection, QoS Monitoring, that require require flow-based IP traffic measurements. It is hence important to have a standard way of exporting information related to IP flows. This document defines architecture for IP traffic flow monitoring, measuring and exporting. It provides a high-level description of the key components and their functions.

2. Scope

This document defines architecture for IPFIX. The main objective of this document is to:

* Describe the key architectural components of IPFIX.
* Define the architectural requirements, e.g., Recovery, Security, etc for the IPFIX framework.
* Define the criteria to select the IPFIX Protocol.
* Specify the control/data message formats and handshaking details to pass the IP flow information.

3. Terminology

* IP Traffic Flow or Flow:
  A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties derived from the data contained in the packet and from the packet treatment at the observation point.

  In this draft we define the flow more specifically. A flow is defined as a set of packets passing an observation point in the network during a certain time interval. All packets belonging to

a particular flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. One or more of packet header fields (eg. destination IP address)
2. One or more properties of the packet itself (eg. packet length)
3. One or more of fields derived from packet treatment (eg. AS number)

A packet is defined to belong to a flow if it matches all the defined properties of the flow. Each of the fields from 1., 2.

and 3. mentioned above are referred to as flow keys. Though a flow could match a general application-level end-to-end stream, its definition is not restricted to this alone. The above definition covers a broad range from a flow containing all packets observed on a set of observation points to a flow consisting of just a single packet between two applications with a specific sequence number observed at a single observation point. Some examples of flows are listed below:

Example 1: To create flows, we define the different fields to distinguish flows. The different combination of the field values creates unique flows. If the keys are defined as {source IP address, destination IP address, TOS}, then all of these are different flows.

1. {192.1.40.1, 171.6.23.5, 4}
2. {192.1.40.23, 171.6.23.67, 4}
3. {192.1.40.23, 171.6.23.67, 2}
4. {198.20.9.200, 171.6.23.67, 4}

Example 2: To create flows, we can apply a match function to all the packets that pass through an observation point, in order to aggregate some values. This could be done by defining the keys as {source IP address, destination IP address, TOS} like in the example 1, and applying the function which masks the least significant 8 bits of the source IP address and destination IP address (i.e. the resultant is a /24 address). The 4 flows from example 1 would now be aggregated into 3 flows, by merging the flows 1. and 2. into a single flow.

1. {192.1.40.0/24, 171.6.23.0/24, 4}
2. {192.1.40.0/24, 171.6.23.0/24, 2}
3. {198.20.9.0/24, 171.6.23.0/24, 4}

Example 3: To create flows, we can filter some field values on all packets that pass the observation point, in order to select only certain flows. The filter is defined by choosing fixed values for specific fields from the packet.

All the packets that go from a customer network 192.1.40.0/24 to another customer network 171.6.23.0/24 with TOS value of 4 define a flow. All other combinations don't define a flow and are not taken into account. The 3 flows from example 2 would now be reduced to 1 flow, by filtering away the second and the third flow. {192.1.40.0/24, 171.6.23.0/24, 4}.

The above example can be thought of as a function F takes as input {source IP address, destination IP address, TOS}. The

function selects only the packets which satisfy all the 3 conditions which are:

* mask the least significant 8 bits of source IP address, compare against 192.1.40.0.
* mask the least significant 8 bits of destination IP address, compare against 171.6.23.0.
* tos value equal to 4.

Depending on the values of {source IP address, destination IP address, TOS} of the different observed packets, the metering process function F would choose/filter/aggregate different sets of packets, which would create different flows. In other words, based on various combination of values of {source IP address, destination IP address, TOS}, F(source IP address, destination IP address, TOS) would result in the definition of one or more flows. The function F is referred to as Flow Type.

* Flow Key:
  Each of the fields which belong to
  1. Packet header (eg. destination IP address)
  2. Property of the packet itself (eg. packet length)

3. Derived from packet treatment (eg. AS number)

which is used to define a flow is termed as flow key.

* Flow Type:
  A function F which would take input as a set of flow keys and the output would be one or more flows depending on the combination of values for the set of flow keys.

* Flow Record:
  A flow record contains information about a specific flow that was metered at an observation point. A flow record contains measured properties of the flow (e.g. the total number of bytes of all packets of the flow) and usually characteristic properties of the flow (e.g. source IP address).

* Export Process:
  The process of sending flow records to one or more collectors.

* IPFIX Device:
  A device hosting at least an observation point, a metering process and a export process. Typically, corresponding observation point(s), metering process(es), and exporter process(es) are co-located at this device, for example, at a router.

* Collector:
  The collector receives flow records from one or more exporters. The collector might process or store received flow record, but these actions are out of the scope of this document.

* Observation Point:
  The observation point is a location in the network where IP packets can be observed. Examples are, a line to which a probe is attached, a shared medium, such as an Ethernet-based LAN, a single port of a router, or a set of interfaces (physical or logical) of a router.

* Metering Process:
  The metering process generates flow records. Input to the process are IP packets observed in an observation point. The metering

process consists of a set of functions that includes packet header capturing, timestamping, sampling, classifying, and maintaining flow records.

* Observation Domain:
  The set of observation points which is the largest aggregatable set of flow information at the IPFIX Device is termed as an observation domain. The observation domain presents itself a unique ID to the collector for identifying the export packets generated by it. One or more Observation Domains can interface with the same export process. Example: The observation domain could be a router line-card, composed of several interfaces with each interface being an observation point.

* Template:
  Template is an ordered n-tuple (eg. <type,length>, TLV), used to completely identify the structure and semantics of a particular information that needs to be communicated from the IPFIX Device to the collector. Each template is uniquely identifiable by some means (eg. by using a Template ID).

* Control Information, Data Stream:
  The information that needs to be exported from the IPFIX device can be classified into the following categories:

  - Control Information :
    This includes the flow type definition, selection criteria for packets within the flow send by the export process and any IPFIX protocol messages (eg. Keepalives). This stream carries all the information for the end-points to understand the IPFIX protocol and specifically for the receiver to understand and interpret the data send by the sender.

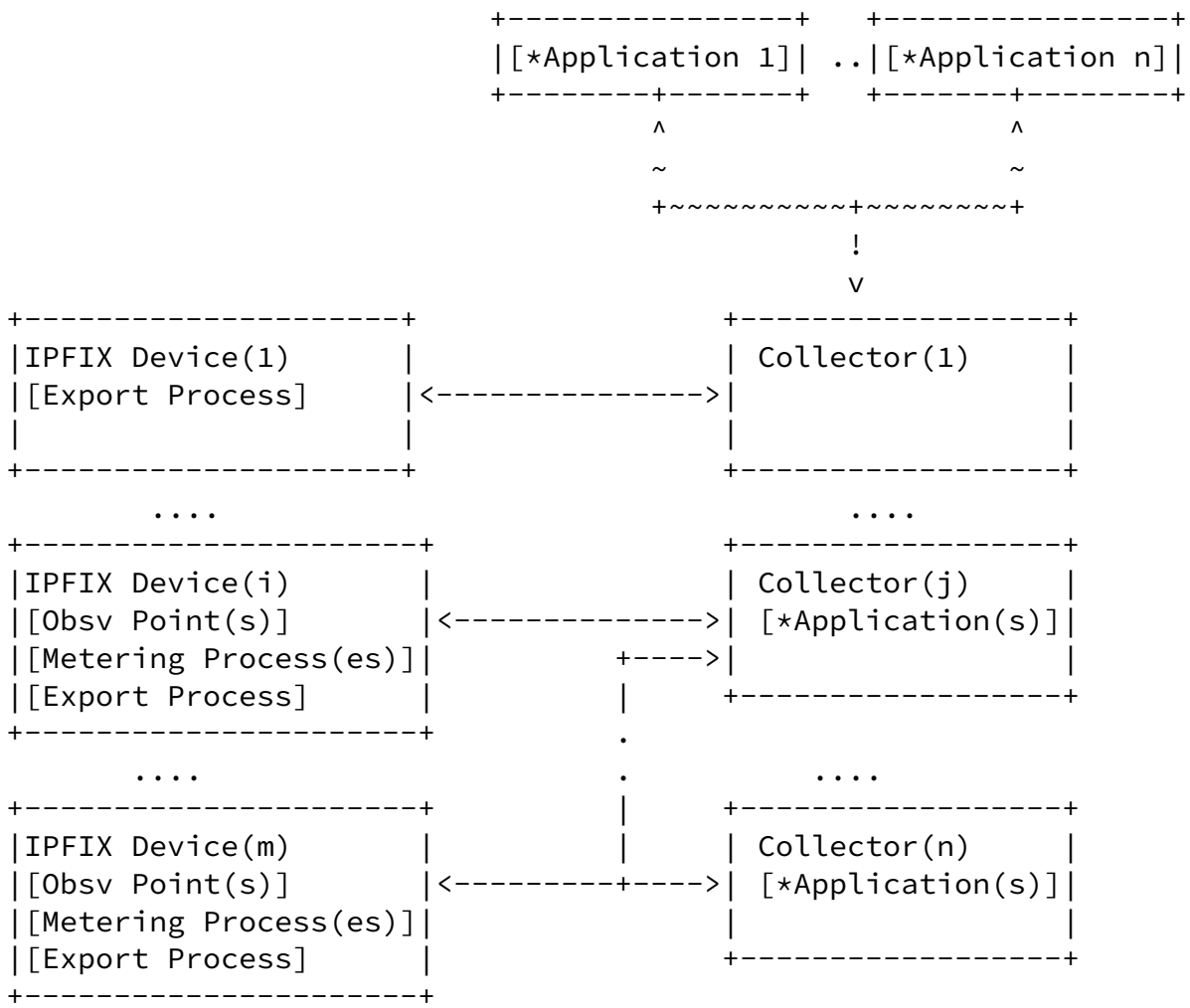
  - Flow record :
    This includes data records corresponding to the information on various observed flows at each of the observation point.This is also called as Data Stream.

The definitions in this section is intended be identical with that in the IPFIX data model [3] and in the event of a discrepancy, the definition specified in this document supercedes the one defined in

the latter.

4. IPFIX reference Model

The figure below shows the reference model for IPFIX. This figure covers the various possible scenarios that can exist in an IPFIX system.

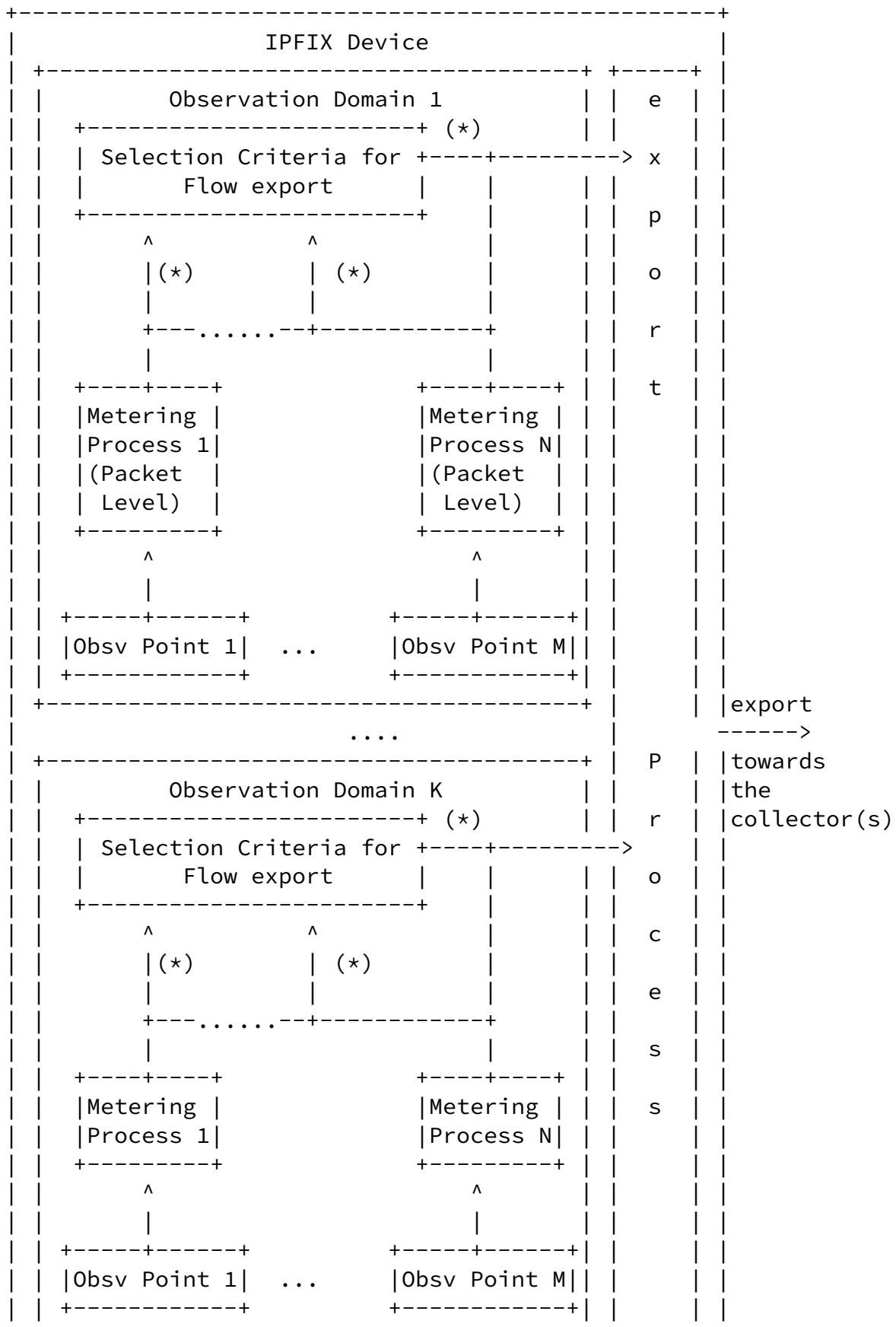


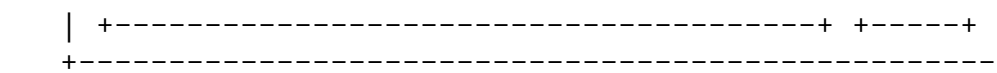
The various functional components are indicated within []. The functional components within [*] are not part of the IPFIX framework.

The interfaces shown by "<-->" are defined by the the IPFIX framework

and those shown by "<~~>" are not.

The figure below shows a typical IPFIX device.


IPFIX Device
Observation Domain 1
Selection Criteria for Flow export
(*)
(*)
(*)
Metering Process 1 (Packet Level)
Metering Process N (Packet Level)
Obsv Point 1
Obsv Point M
...
....
Observation Domain K
Selection Criteria for Flow export
(*)
(*)
(*)
Metering Process 1
Metering Process N
Obsv Point 1
Obsv Point M
...
e
x
p
o
r
t
P
r
o
c
e
s
s
export
towards
the
collector(s)



In this figure the functional blocks are shown in rectangular boxes. The interface shown by (*) is applicable only if the optional metering process at the flow level is present. Otherwise the metering process(es) at the packet level interfaces directly with the exporting function. Note that in case of multiple observation domains, a unique ID per observation domain must be transmitted as a parameters to the exporting function.

4.1. IPFIX protocol

At the IPFIX device, the protocol functionality may be split between observation domain and export process.

At a high level, IPFIX protocol at the IPFIX device does the following:

1. Encode the control information into templates.
2. Encode the flows observed at the observation points into flow records.
3. Packetize the flow record and/or control information into export packets based on the export policies.
4. Use the underlying transport layer to send the export packets to the collector.

At a high level, IPFIX protocol at the collector is responsible for the following:

1. Receive and store the control information.
2. Decode and store the flow records using the control information.

4.2. Export Process

The Export Process is the functional block that interacts with observation domain(s) on one side and collector(s) on the other side. The typical functions of an export process may include:

* Accept control information and data streams from one or more observation domains and separate them into separate export packet

streams based on observation domain.
* Run the part of the IPFIX protocol which deals with packetization and transport of flow records/control information with the collector. This involves:

  - Gather flow records from the metering process(es) and export them towards the collector(s), using the data stream.

  - Export the control information regarding the flow and metering process(es) towards the collector(s).

* Optionally monitor the status of the collector and execute a fail over in case of problem.

4.3. Observation Domain

The Observation Domain is the functional block, which MUST manage the flows generated from all the valid {Observation Point, Metering Process} combination defined within itself. The typical functions of an Observation Domain MAY include:

* Encoding the flow records and sending them to the export process.
* Encoding the control information into templates and sending them to the export process.
* Deciding which flow records/control information to export using rules based on time, thresholds, configuration events etc.
* Aggregating flow records generated by one or more metering processes.
* Flow record maintenance which may include creating new records, updating existing ones, computing flow statistics, deriving further flow properties, adding non-flow specific information (in some cases fields like AS numbers) detecting flow expiration, passing flows record to the exporting process, and deleting flow records.
* Perform appropriate middle-box functions to translate the flow information.

4.4. Metering Process Functions

4.4.1. Flow Classification

The collector MUST be able to map the flow record to the corresponding property types defined by the flow type. In addition the collector, when it receives the flow records, MAY need the following to interpret the flow records further:

a. Observation Point.
b. Selection Criteria of Packets

A flow record can be better analyzed if the Observation Point from which it is measured is known. As such it is recommended that the flow record carry the Observation Point information along with the flow records when exported. In cases where there is a single observation point or where the observation point information is not relevant, the exporter MAY choose not to add this to the flow records.

4.4.2. Selection Criteria Of Packets

The measurement device MAY define rules so that only certain packets within a flow can be chosen for measurement at an observation point. This MAY be done by one of the two types of methods defined below or a combination of them. A combination of each of these ways can be adopted to select the packets .i.e. one can define a set of methods {F1, S1, F2, S2, S3} executed in certain sequence at an observation point to collect flows of a particular type.

4.4.3. Function on properties that determines a flow type (Fi)

Packets that satisfy a function on the fields defined by the packet header fields or fields obtained while doing the packet processing or the properties of the packet itself.

Example: Mask/Match of the fields that define a filter. The filter may be defined as {Protocol == TCP, Destination Port between 80 and 120}.

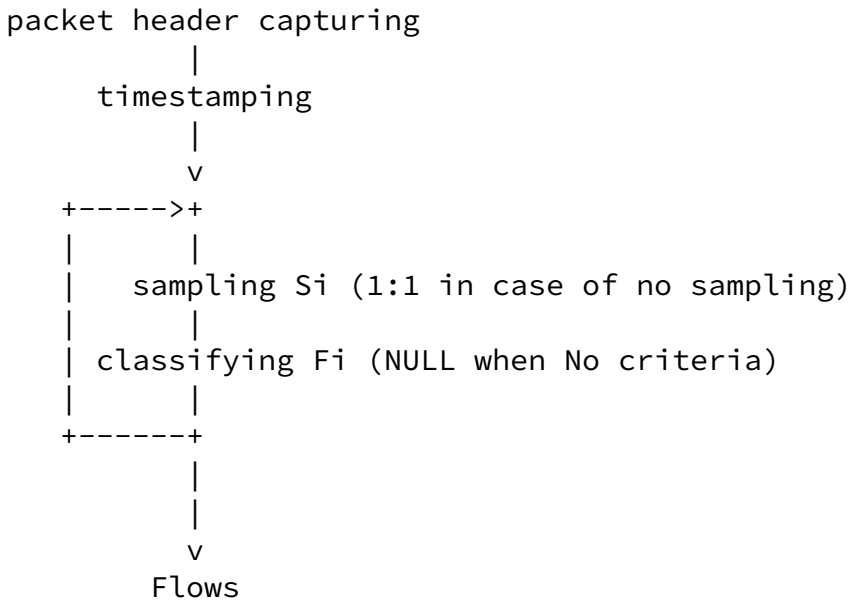
Multiple such filters could be used in any sequence to select packets.

4.4.4. Sampling packets on a flow type (Si)

Packets that satisfy the sampling criteria for this flow type.

Example: Sample every 100th packet that was received at an observation point and collect the flow information for a particular flow type. choosing all the packets is a special case where sampling rate is 1:1.

The figure below shows the operations which MAY be applied as part of a typical metering process.



4.5. Selection Criteria of flows for export

There MAY be additional rules defined within the observation domain so that only certain flows records are picked up for export. This MAY be done by either one or a combination of Si, Fi.

Example: The flow records which meet the following selection criteria are only exported.

1. All flow records whose destination IP address matches {20.3.1.5}.
2. Every other (.i.e. sampling rate 1 in 2) flow record whose destination IP address matches {160.0.1.30}.

4.6. Collector

Collector is a subsystem that interacts with one or more IPFIX devices. The functions of the collector MAY include:

* Identifying, accepting and decoding export packets from different {Export Process, Observation Domain} pairs .
* Running the IPFIX protocol.
* Storing the control information and flow records received from IPFIX device.
* Notifying the IPFIX device its status and problems.

4.7. Applications

Applications that use the information collected by IPFIX may be Billing, Intrusion Detection sub-systems, etc. These applications may be an integral part of the collector or collocated to the collector. The way by which these applications interface with IPFIX system to get the desired information is out of this document's scope.

5. IPFIX Protocol

5.1. Selection Criteria for IPFIX Protocol

There are existing standard practices in the area of flow export like Netflow, CRANE, LFAP etc. The charter mentions to choose the the protocol among these existing practices that fits the IPFIX requirements the most. There may be additions or modifications made to the chosen protocol to fit it exactly into the IPFIX architecture. The following is the list of criteria that the candidate protocol SHOULD meet in order to be the qualified into the IPFIX architecture. This is based on the requirements specified in the requirement document [2].

5.1.1. Common for IPFIX Device and Collector

1. Transparency over transport protocol. Ability to operate over a congestion aware transport like TCP or SCTP is a MUST.
2. Transparency over any underlying security protocols.
3. The protocol SHOULD be based on a flexible data model based on templates.
4. The protocol SHOULD be based on a extensible information model.

5.1.2. IPFIX Protocol on IPFIX Device (At Export Process)

1. Ability to detect loss of connectivity with the collector and trigger the appropriate action (eg. a switch over to an alternate collector.)
2. Optionally export flow records to multiple collectors.
3. Optionally re-transmit lost flow records.
4. Exchange control information from the collector, monitor export process and detect any overload in the process of exporting.

5.1.3. Intellectual Property Rights

The protocol must abide by the intellectual property rights as defined in rfc: 2026. Specifically section: 10.3.1. All

Contributions. If the protocol does not abide by this, it will not be considered.

5.1.4. IPFIX Protocol on Collector

1. Receive and decode the flow records from the IPFIX devices.
2. Ability to indicate flow record losses to the exporting IPFIX device and/or IPFIX users.
3. Optionally notify the status and overload conditions to the IPFIX device.

Once the selection is made from the set of candidate protocols, this section would be replaced by the chosen protocol.

5.2. Export Models

5.2.1. Export Model with Reliable Control Connection

As mentioned in the selection criteria, the control information and data stream MUST be transported over a congestion-aware transport protocol. If the network in which the IPFIX device and collector are located does not guarantee reliability, at least the control information SHOULD be exported over a reliable transport. There could be network security concerns between IPFIX device and collector. To avoid re-inventing the wheel, and to reduce the complexity of flow export protocol, one or a combination of the following methods MAY be adopted as a solution to achieve security :

* IP Authentication Header MAY be used when the threat environment requires stronger integrity protections, but does not require confidentiality.
* IP Encapsulating Security Payload (ESP) MAY be used to provide confidentiality and integrity.
* If the transport protocol used is TCP, optionally TCP MD5 signature option MAY be used to protect against spoofed TCP segments.
* If the transport protocol used is TCP, optionally TLS MAY be used to add integrity, authenticity and confidentiality.

The data stream MAY be exported over an reliable or unreliable transport protocol.

As explained above the transport connection (in the case of a connection oriented protocol) is pre-setup between the IPFIX device and the collector. Once connected, the collector side receives the control information and uses this information to interpret the flow records. The IPFIX device SHOULD set the keepalive (eg. keepalive timeout in the case of TCP; the HEARTBEAT interval in the case of SCTP; IPFIX protocol level Keepalive if any) to a sufficiently low value so that it can quickly detect a collector crash.

5.3. Collector Crash Detection and Recovery

5.3.1. Export Model with Reliable Control Connection

The collector crash is detected at the IPFIX device by the break in control connection (depending on the transport protocol the connection timeout mechanisms differ). On detecting a Keepalive timeout, the IPFIX device SHOULD stop sending the flow export data to the collector and try reconnecting the transport connection. This is valid for a single collector scenario. If there are multiple collectors for the same IPFIX device, the IPFIX device opens control connections to each of the collectors. But data gets sent only to one of the collectors which is chosen as the primary. There could be one or more collectors configured as secondary and a priority assigned to them. The primary collector crash is detected at the IPFIX device by the break in control connection (depending on the transport protocol the connection timeout mechanisms differ). On detecting loss of connectivity, the IPFIX device opens data stream with the secondary collector of the next highest priority. This collector now becomes the primary. The maximum export data loss would be the amount of data exported in the time between when the loss of connectivity to the collector happened, and the time at which this was detected by the IPFIX device.

5.4. Collector Redundancy

Since IPFIX protocol requires a congestion-aware transport, achieving redundancy using multicast is not an option. Multiple <control information, data stream> pairs could be setup, each to a different collector from the same IPFIX device. The control and data information are then replicated on each of the control information and data stream.

6. Security Consideration

IP flow information can be used for various purposes, such as usage accounting, traffic profiling, traffic engineering, and intrusion detection. For each application, the security requirement may differ significantly from one to another. To be able to satisfy the security needs of various IPFIX users, the architecture of IPFIX MUST provide different levels of security protection.

6.1. Data security

IPFIX data consists of control information and data stream generated by the IPFIX device.

The IPFIX data may exist in both the IPFIX device and the collector. In addition, the data is also transferred on the wire from the exporter to the collector when it is reported. To provide security, the data SHOULD be protected from adversity.

The protection of IPFIX data within the end system (IPFIX device and collector) is out of the scope. It is assumed that the end system operator will provide adequate security for the IPFIX data.

The IPFIX architecture MUST allow different levels of protection to the IPFIX data on the wire. Where ever security functions are required it is recommended to leverage to lower layers using either IPsec or TLS, if they can successfully satisfy the security requirement of IPFIX data protection.

To protect the data on the wire, three levels of granularity SHOULD be supported:

6.1.1. No security

No security is required when the transport between the IPFIX device and the collector is perceived as safe. This option allows the protocol to run most efficiently without extra overhead and an IPFIX solution MUST support it.

6.1.2. Authentication only

The authentication only protection provides the IPFIX users the assurance of data integrity and authenticity. The data exchanged between the IPFIX device and the collector is protected by authentication signature. Any modification of the IPFIX data will be

detected by the recipient, resulting in discarding of the received data. However, the authentication only option doesn't offer data confidentiality. The IPFIX user SHOULD avoid use this option when sensitive or confidential information is being exchanged. An IPFIX solution SHOULD support this option. The authentication only option SHOULD provide replay attack protection. Some means to achieve this level of security are:

* TCP with MD5 options.
* IP Authentication Header

6.1.3. Encryption

Data encryption provides the best protection for IPFIX data. The IPFIX data is encrypted at the sender and only the intended recipient can decrypt and have access to the data. This option MUST be used when the transport between the exporter and the collector are unsafe and the IPFIX data needs to be protected. It is recommended to use the underlying security layer functions for this purpose. Some means to achieve this level of security are:

* Encapsulating Security Payload.
* Transport Layer Security Protocol

The data encryption option adds overhead to the IPFIX data transfer. It may limit the rate that an export can report its flow to the collector due to the heavy resource requirement of running encryption.

6.2. IPFIX end point authentication

It is important to make sure that the IPFIX device is talking to the

"right" collector instead of a masqueraded collector. The same logic also holds true from the collector point of view that it want to make sure it is collecting the flow information from the "right" IPFIX device. The IPFIX architecture SHOULD allow the authentication capability so that either one-way or mutual authentication can be performed between the IPFIX device and collector.

The IPFIX architecture SHOULD use the existing transport protection protocols such as TLS to fulfill the authentication requirement.

6.3. Denial of service (DoS) attack prevention

Since one of the potential usages for IPFIX is for intrusion detection, it is important for the IPFIX architecture to support some kind of DoS resistance.

6.3.1. Network under attack

The Network itself may be under attack, resulting in an overwhelming number of IPFIX messages. The IPFIX SHOULD try to capture as much information as possible. However, when large amount IPFIX messages are generated in a short period of time, the IPFIX may become overloaded.

6.3.2. Generic DoS attack on the IPFIX system

The IPFIX system may subject to generic DoS attacks, just as any system on any open networks. These types of attacks are not IPFIX specific. Preventing and responding to such types of attacks are out of the scope of IPFIX WG.

6.3.3. IPFIX Specific DoS attack

There is a specific attack on the IPFIX portion of the IPFIX device or Collector.

(To be added and discussed on the general list).

7. Flow Expiration

A flow is considered to be inactive if no packets of this flow has been observed at the observation point for a given timeout interval. The flow can be exported under the following conditions:

1. If the exporter can deduce the end of a flow, the exporter SHOULD export the flow records when the end of the flow is detected. For example: flow generated by TCP type of traffic where the FIN or RST bits indicate the end of the flow

2. If the flow has been inactive for a certain period of time. This inactivity timeout SHOULD be configurable. For example: flow generated by UDP type of traffic.

3. For long aging flows, the exporter SHOULD export the flow records on regular basis, in order to:

    a. Report the flow records periodic accounting information to the collector
    b. Avoid counter wrapping This activity timeout SHOULD be configurable
    c. Prevent an attacker from indefinitely delaying the delivery of flow information to an IDS application by intentionally generating packets which fall within long aging flows.

4. If the exporter experiences resources constraints, a flow MAY be prematurely expired (example: memory)

8. IANA Consideration

Need Port number assigned from IANA [more to be written]

9. References

[1] IP Flow Information Export (IPFIX) <http://www.ietf.org/html.charters/ipfix-charter.html>

[2] J. Quittek ,T. Zseby, B. Claise,"Requirements for IP Flow Information Export", (work in progress) ,Internet Draft, Internet Engineering Task Force, <draft-ietf-ipfix-reqs-02.txt>, August 2002

[3] K.C. Norseth, Paul Calato,"Data Model for IP Flow Information Export", (work in progress) ,Internet Draft, Internet Engineering Task Force, <draft-ietf-ipfix-data-00.txt>, August 2002

10. Acknowledgements

We like to thank all the people contributing to the requirements discussion on the mailing list, and the design teams for many valuable comments.

George Carle
Tanja Zseby
Paul Calato
Dave Plonka
KC Norseth
Benoit Claise
Ganesh Sadasivan
Vamsi Valluri
Cliff Wang
Ram Gopal

Jc Martin
Carter Bullard
Juergen Quittek
Reinaldo Penno
Nevil Brownlee
Simon Leinen

11. Author's Addresses

Benoit Claise
Cisco Systems
De Kleetlaan 6a b1
1831 Diegem
Belgium
Phone: +32 2 704 5622
Email: bclaise@cisco.com

Ganesh Sadasivan
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134
USA
Phone: +1 (408) 527-0251
Email: gsadasiv@cisco.com

K.C. Norseth
Consultant
934 S. Palos Verdes Dr.
Kaysville, Utah 84037
Phone: +1 (801) 546-3316
Email: kcn@norseth.com

Juergen Quittek
NEC Europe Ltd.
Adenauerplatz 6
69115 Heidelberg
Germany
Phone: +49 6221 90511-15
EMail: quittek@ccrle.nec.de

Kevin Zhang
XACCT Technologies, Inc.
2900 Lakeside Drive

Santa Clara, CA 95054
Phone +1 301 992 4697
Email: kevin.zhang@xacct.com

12. Full Copyright Statement