IPFIX Working Group	B. Trammell
Internet-Draft	ETH Zurich
Intended status: Standards Track	A. Wagner
Expires: March 29, 2012	Consecom AG
	B. Claise
	Cisco Systems, Inc.
	September 26, 2011

Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol
draft-trammell-ipfix-a9n-04.txt

Abstract

This document describes the export of aggregated Flow information using IPFIX. An Aggregated Flow is essentially an IPFIX Flow representing packets from multiple Original Flows sharing some set of common properties. The document describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

Table of Contents

1. Introduction

The assembly of packet data into Flows serves a variety of different purposes, as noted in the requirements *[RFC3917]* and applicability statement *[RFC5472]* for the IP Flow Information Export (IPFIX) protocol *[RFC5101]*. Aggregation beyond the flow level, into records representing multiple Flows, is a common analysis and data reduction technique as well, with applicability to large-scale network data analysis, archiving, and inter-organization exchange. This applicability in large-scale situations, in particular, led to the inclusion of aggregation as part of the IPFIX Mediators Problem Statement *[RFC5982]*, and the definition of an Intermediate Aggregation Process in the Mediator framework *[RFC6183]*.

Aggregation is part of a wide variety of applications, including traffic matrix calculation, generation of time series data for visualizations or anomaly detection, or measurement data reduction. Depending on the keys used for aggregation, it may additionally have an anonymizing affect on the data: for example, aggregation operations which eliminate IP addresses make it impossible to later identify nodes using those addresses.

Aggregation as defined and described in this document covers the applications defined in [RFC5982], including 5.1 "Adjusting Flow Granularity", 5.4 "Time Composition", and 5.5 "Spatial Composition". However, this document specifies a more flexible architecture for an Intermediate Aggregation Process than that envisioned by the original Mediator work, in Section 4.2. Instead of a focus on these specific limited use cases, the Intermediate Aggregation Process is specified to cover any activity commonly described as "flow aggregation".

An Intermediate Aggregation Process may be applied to data collected from multiple Observation Points, as aggregation is natural to apply for data reduction when concentrating measurement data. This document specifically does not address the protocol issues that arise when combining IPFIX data from multiple Observation Points and exporting from a single Mediator, as these issues are general to IPFIX Mediation; they are therefore treated in detail in the Mediator Protocol *[I-D.claise-ipfix-mediation-protocol]* document.

Since Aggregated Flows as defined in the following section are essentially Flows, the IPFIX protocol [RFC5101] can be used to export, and the IPFIX File Format [RFC5655] can be used to store, aggregated data "as-is"; there are no changes necessary to the protocol. This document provides a common basis for the application of IPFIX to the handling of aggregated data, through a detailed terminology, Intermediate Aggregation Process architecture, and methods for Original Flow counting and counter distribution across intervals.

1.1. IPFIX Protocol Overview

In the IPFIX protocol, { type, length, value } tuples are expressed in templates containing { type, length } pairs, specifying which { value } fields are present in data records conforming to the Template, giving great flexibility as to what data is transmitted. Since Templates are sent very infrequently compared with Data Records, this results in significant bandwidth savings. Various different data formats may be transmitted simply by sending new Templates specifying the { type, length } pairs for the new data format. See [RFC5101] for more information.

The IPFIX information model *[RFC5102]* defines a large number of standard Information Elements which provide the necessary { type } information for Templates. The use of standard elements enables interoperability among different vendors' implementations. Additionally, non-standard enterprise-specific elements may be defined for private use.

1.2. IPFIX Documents Overview

"Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information" *[RFC5101]* and its associated documents define the IPFIX Protocol, which provides network engineers and administrators with access to IP traffic flow information.

"Architecture for IP Flow Information Export" *[RFC5470]* defines the architecture for the export of measured IP flow information out of an IPFIX Exporting Process to an IPFIX Collecting Process, and the basic terminology used to describe the elements of this architecture, per the requirements defined in "Requirements for IP Flow Information Export" *[RFC3917]*. The IPFIX Protocol document [RFC5101] then covers the details of the method for transporting IPFIX Data Records and Templates via a congestion-aware transport protocol from an IPFIX Exporting Process to an IPFIX Collecting Process.

This document specifies an Intermediate Process which may be applied at an IPFIX Mediator. "IP Flow Information Export (IPFIX) Mediation: Problem Statement" *[RFC5982]* introduces the concept of IPFIX Mediators, and defines the use cases for which they were designed; "IP Flow Information Export (IPFIX) Mediation: Framework" *[RFC6183]* then provides an architectural framework for Mediators. Protocol-level issues (e.g., template and observation domain handling across Mediators) are covered by "Specification of the Protocol for IPFIX Mediation" *[I-D.claise-ipfix-mediation-protocol]*.

2. Terminology

Terms used in this document that are defined in the Terminology section of the IPFIX Protocol *[RFC5101]* document are to be interpreted as defined there.

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 [RFC2119].
In addition, this document defines the following terms
The terminology presented herein improves the precision of, but does not supersede or contradict the terms related to mediation and aggregation defined in the problem statement *[RFC5982]* and the framework *[RFC6183]* documents. Within this document, the terminology defined in this section is to be considered normative.

3. Use Cases for IPFIX Aggregation

Aggregation, as a common data reduction method used in traffic data analysis, has many applications. When used with a regular Aggregation Interval, it generates time series data from a collection of Flows with discrete intervals. This time series data is itself useful for a wide variety of analysis tasks, such as generating input for network anomaly detection systems, or driving visualizations of volume per time for traffic with specific characteristics. As a second example, traffic matrix calculation from flow data is inherently an aggregation action, by aggregating the Flow Key down to input or output interface, address prefix, or autonomous system.
Irregular or data-dependent Aggregation Intervals and key aggregation operations can also be used to provide adaptive aggregation of network flow data. Here, full Flow Records can be kept for Flows of interest, while Flows deemed "less interesting" to a given application can be aggregated. For example, in an IPFIX Mediator equipped with traffic classification capabilities for security purposes, potentially malicious Flows could be exported directly, while known-good or probably-good Flows (e.g. normal web browsing) could be exported simply as time series volumes per web server.
Note that an Intermediate Aggregation Process which removes potentially sensitive information as identified in [RFC6235] may tend to have an anonymising effect on the Aggregated Flows, as well; however, any application of aggregation as part of a data protection scheme should ensure that all the issues raised in [RFC6235] are addressed, specifically Section 4 "Anonymization of IP Flow Data", Section 7.2 "IPFIX-Specific Anonymization Guidelines", and Section 9 "Security Considerations".
While much of the discussion in this document, and all of the examples, apply to the common case that the Original Flows to be aggregated are all of the same underlying type (i.e., are represented with identical or compatible Templates), and that each packet observed by the Metering Process on the far side of the Original Exporter is represented, this is not a necessary assumption. Aggregation can also be applied as part of a technique applying both aggregation and correlation to pull together multiple views of the same traffic from different Observation Points using different Templates. For example, consider a set of applications running at different Observation Points for different

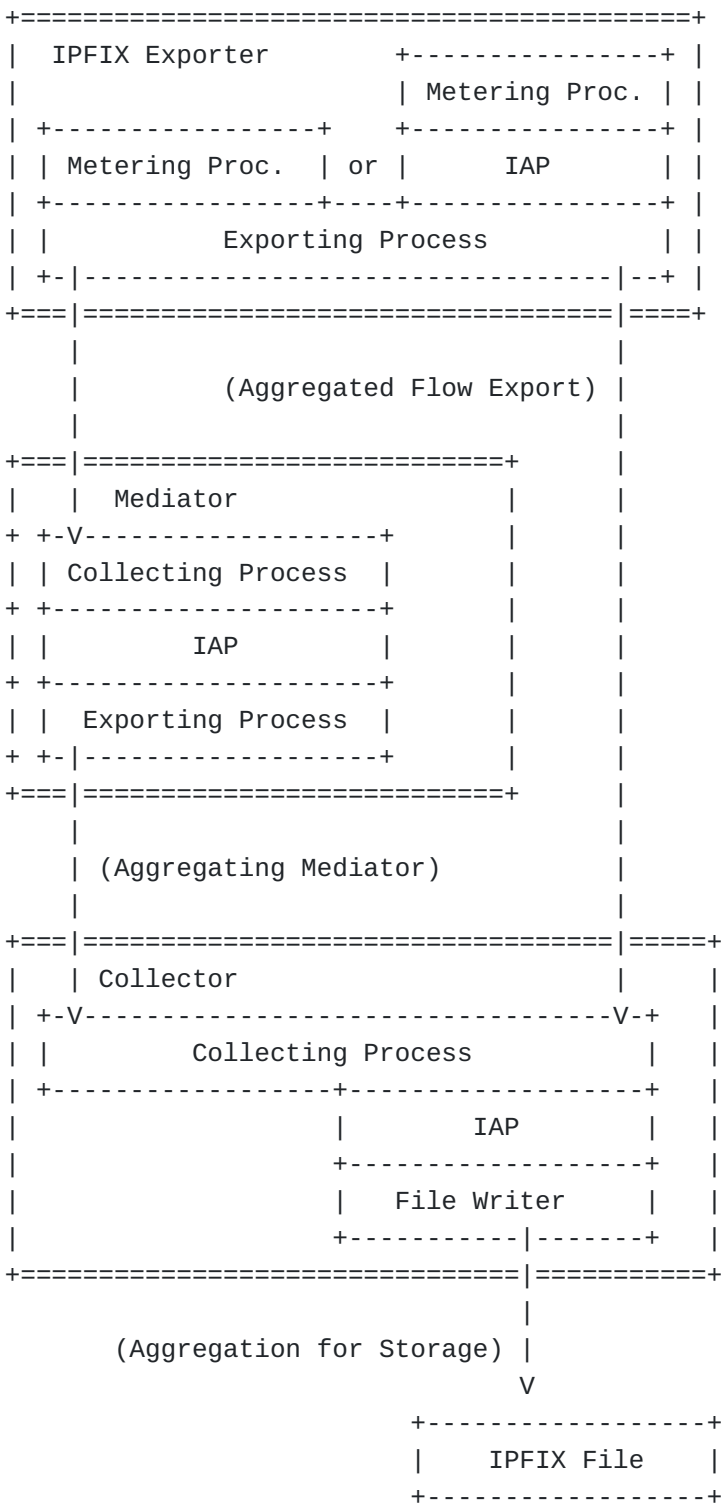
purposes -- one generating flows with round-trip-times for passive performance measurement, and one generating billing records. Once correlated, these flows could be run through an Intermediate Aggregation Process to produce Aggregated Flows containing both volume and performance information together. (Note, however, that the details of correlation are not in scope for this document.)

4. Architecture for Flow Aggregation

This section specifies how an Intermediate Aggregation Process fits into the IPFIX Architecture, and the architecture of the Intermediate Aggregation Process itself.

4.1. Aggregation within the IPFIX Architecture

An Intermediate Aggregation Process could be deployed at any of three places within the IPFIX Architecture. While aggregation is most commonly done within a Mediator which collects Original Flows from an Original Exporter and exports Aggregated Flows, aggregation can also occur before initial export, or after final collection, as shown in Figure 1. The presence of an IAP at any of these points is of course optional.



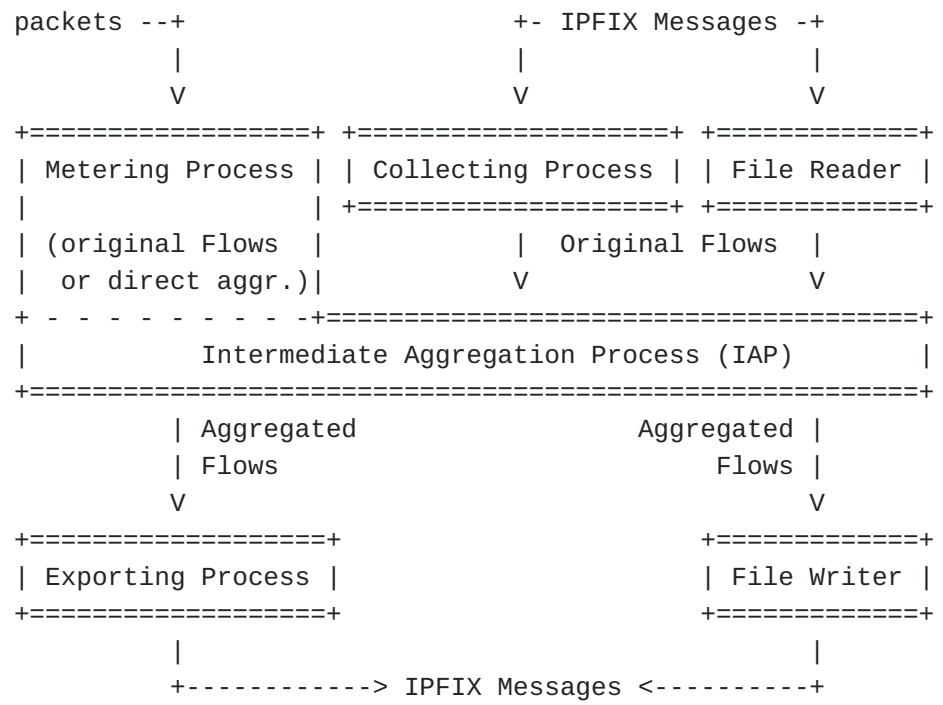
The Mediator use case is further shown in Figures A and B in [RFC6183]. Aggregation can be applied for either intermediate or final analytic purposes. In certain circumstances, it may make sense to export Aggregated Flows directly from an original Exporting Process, for example, if the Exporting Process is applied to drive a time-series visualization, or when flow data export bandwidth is restricted and flow or packet sampling is not an option. Note that this case, where

the Aggregation Process is essentially integrated into the Metering Process, is essentially covered by the IPFIX architecture *[RFC5470]*: the Flow Keys used are simply a subset of those that would normally be used, and time intervals may be chosen other than those available from the cache policies customarily offered by the Metering Process. A Metering Process in this arrangement MAY choose to simulate the generation of larger Flows in order to generate Original Flow counts, if the application calls for compatibility with an Aggregation Process deployed in a separate location.

In the specific case that an Aggregation Process is employed for data reduction for storage purposes, it can take Original Flows from a Collecting Process or File Reader and pass Aggregated Flows to a File Writer for storage.

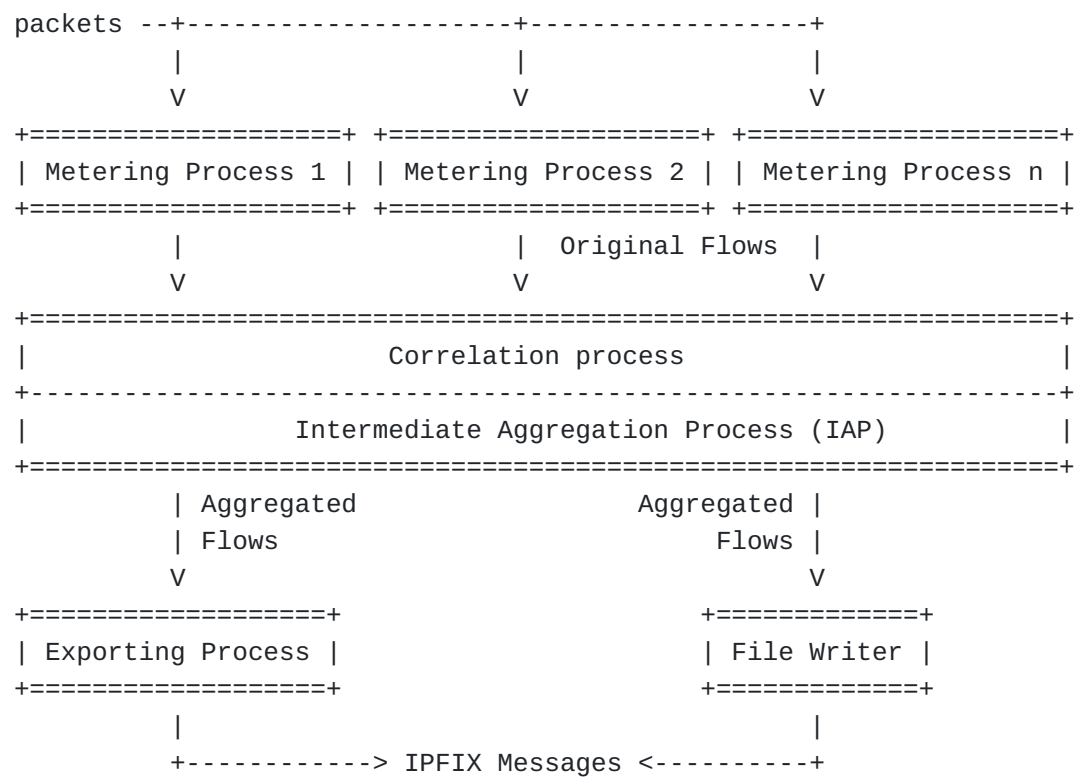
Deployment of an Intermediate Aggregation Process within a Mediator *[RFC5982]* is a much more flexible arrangement. Here, the Mediator consumes Original Flows and produces Contributing Flows; this arrangement is suited to any of the use cases detailed in Section 3. In a Mediator, aggregation can be applied as well to aggregating Original Flows from multiple sources into a single stream of Aggregated Flows; the architectural specifics of this arrangement are not addressed in this document, which is concerned only with the aggregation operation itself; see [I-D.claise-ipfix-mediation-protocol] for details.

The data paths into and out of an Intermediate Aggregation Process are shown in Figure 2.



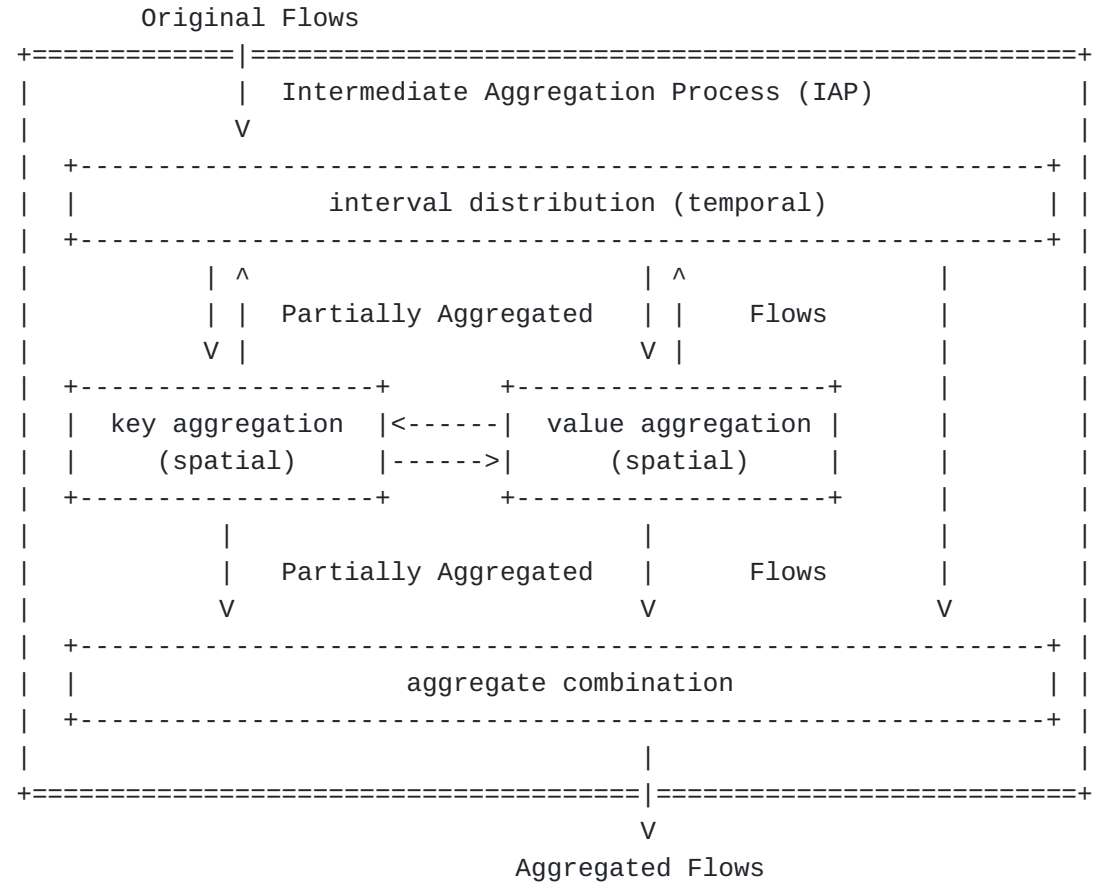
In the special case that aggregation is used together with correlation to aggregate the output from multiple Metering Processes, the

arrangement would appear as in Figure 3; the details of correlation are, as noted above, out of scope for this document.



4.2. Intermediate Aggregation Process Architecture

Within this document, an Intermediate Aggregation Process can be seen as hosting a function composed of four types of operations on Partially Aggregated Flows, as illustrated in Figure 4. "Partially Contributing Flows" as defined in Section 2 are essentially the intermediate results of aggregation, internal to the Intermediate Aggregation Process.



The first three of these operations may be carried out any number of times in any order, either on Original Flows or on the results of one of the Operations (called Partially Aggregated Flows), with one caveat: since Flows carry their own interval data, any spatial aggregation operation implies a temporal aggregation operation, so at least one interval distribution step, even if implicit, is required by this architecture. This is shown as the first step for the sake of simplicity in the diagram above. Once all aggregation operations are complete, aggregate combination ensures that for a given Aggregation Interval, set of Flow Key values, and Observation Domain, only one Flow is produced by the Intermediate Aggregation Process.

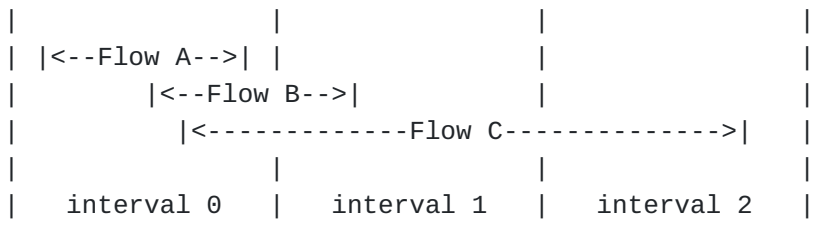
This model describes the operations within a single Intermediate Aggregation Process, and it is anticipated that most aggregation will be applied within a single process. However, as the steps in the model may be applied in any order and aggregate combination is idempotent, any number of Intermediate Aggregation Processes operating in series can be modeled as a single process. This allows aggregation operations to be flexibly distributed across any number of processes, should application or deployment considerations so dictate.

5. IP Flow Aggregation Operations

As stated in Section 2, an Aggregated Flow is simply an IPFIX Flow generated from Original Flows by an Intermediate Aggregation Process. Here, we detail the operations by which this is achieved within an Intermediate Aggregation Process.

5.1. Temporal Aggregation through Interval Distribution

Interval distribution imposes a time interval on the resulting Aggregated Flows. The selection of an interval is specific to the given aggregation application. Intervals may be derived from the Original Flows themselves (e.g., an interval may be selected to cover the entire interval containing the set of all Flows sharing a given Key, as in Time Composition describe in Section 5.1.2) or externally imposed; in the latter case the externally imposed interval may be regular (e.g., every five minutes) or irregular (e.g., to allow for different time resolutions at different times of day, under different network conditions, or indeed for different sets of Original Flows).
The length of the imposed interval itself has tradeoffs. Shorter intervals allow higher resolution aggregated data and, in streaming applications, faster reaction time. Longer intervals lead to greater data reduction and simplified counter distribution. Specifically, counter distribution is greatly simplified by the choice of an interval longer than the duration of longest Original Flow, itself generally determined by the Original Flow's Metering Process active timeout; in this case an Original Flow can contribute to at most two Aggregated Flows, and the more complex value distribution methods become inapplicable.



In Figure 5, we illustrate three common possibilities for interval distribution as applies with regular intervals to a set of three Original Flows. For Flow A, the start and end times lie within the boundaries of a single interval 0; therefore, Flow A contributes to only one Aggregated Flow. Flow B, by contrast, has the same duration but crosses the boundary between intervals 0 and 1; therefore, it will contribute to two Aggregated Flows, and its counters must be distributed among these Flows, though in the two-interval case this can be simplified somewhat simply by picking one of the two intervals, or proportionally distributing between them. Only Flows like Flow A and Flow B will be produced when the interval is chosen to be longer than

the duration of longest Original Flow, as above. More complicated is the case of Flow C, which contributes to more than two Aggregated Flows, and must have its counters distributed according to some policy as in Section 5.1.1.

5.1.1. Distributing Values Across Intervals

In general, counters in Aggregated Flows are treated the same as in any Flow. Each counter is independently calculated as if it were derived from the set of packets in the Original Flow. For the most part, when aggregating Original Flows into Aggregated Flows, this is simply done by summation.

When the Aggregation Interval is guaranteed to be longer than the longest Original Flow, a Flow can cross at most one Interval boundary, and will therefore contribute to at most two Aggregated Flows. Most common in this case is to arbitrarily but consistently choose to account the Original Flow's counters either to the first or the last Contributing Flow to which it could contribute.

However, this becomes more complicated when the Aggregation Interval is shorter than the longest Original Flow in the source data. In such cases, each Original Flow can incompletely cover one or more time intervals, and apply to one or more Aggregated Flows. In this case, the Aggregation Process must distribute the counters in the Original Flows across the multiple Aggregated Flows. There are several methods for doing this, listed here in roughly increasing order of complexity and accuracy; most of these are necessary only in specialized cases.

A method for exporting the distribution of counters across multiple Aggregated Flows is detailed in Section 7.4. In any case, counters MUST be distributed across the multiple Aggregated Flows in such a way that the total count is preserved, within the limits of accuracy of the implementation (e.g., inaccuracy introduced by the use of floating-point numbers is tolerable). This property allows data to be aggregated and re-aggregated without any loss of original count information. To avoid confusion in interpretation of the aggregated data, all the counters for a set of given Original Flows SHOULD be distributed via the same method.

More complex counter distribution methods generally require that the interval distribution process track multiple "current" time intervals at once. This may introduce some delay into the aggregation operation, as an interval should only expire and be available for export when no additional Original Flows applying to the interval are expected to arrive at the Intermediate Aggregation Process.

Note, however, that since there is no guarantee that Flows from the Original Exporter will arrive in any given order, whether for transport-specific reasons (i.e. UDP reordering) or Metering Process implementation-specific reasons, even simpler distribution methods may need to deal with flows arriving in other than start time or end time order. Therefore, the use of larger intervals does not obviate the need to buffer Partially Aggregated Flows within "current" time intervals,

to ensure it can accept flow time intervals in any arrival order. More generally, the interval distribution process SHOULD accept flow start and end times in the Original Flows in any reasonable order. The expiration of intervals in interval distribution operations is dependent on implementation and deployment requirements, and SHOULD be made configurable in contexts in which "reasonable order" is not obvious at implementation time.

5.1.2. Time Composition

Time Composition as in Section 5.4 of [RFC5982] (or interval combination) is a special case of aggregation, where interval distribution imposes longer intervals on Flows with matching keys and "chained" start and end times, without any key reduction, in order to join long-lived Flows which may have been split (e.g., due to an active timeout shorter than the actual duration of the Flow.) Here, no Key aggregation is applied, and the Aggregation Interval is chosen on a per-Flow basis to cover the interval spanned by the set of aggregated Flows. This may be applied alone in order to normalize split Flows, or in combination with other aggregation functions in order to obtain more accurate Original Flow counts.

5.2. Spatial Aggregation of Flow Keys

Key aggregation generates a new set of Flow Key values for the Aggregated Flows from the Original Flow Keys, non-Key fields in the Original Flows, or from correlation of the Original Flow information with some external source. There are two basic operations here. First, Aggregated Flow Keys may be derived directly from Original Flow Keys through reduction, or the dropping of fields or precision in the Original Flow Keys. Second, Aggregated Flow Keys may be derived through replacement, e.g. by removing one or more fields from the Original Flow and replacing them with fields derived from the removed fields. Replacement may refer to external information (e.g., IP to AS number mappings). Replacement may apply to Flow Keys as well as non-key fields. For example, consider an application which aggregates Original Flows by packet count (i.e., generating an Aggregated Flow for all one-packet Flows, one for all two-packet Flows, and so on). This application would promote the packet count to a Flow Key.
Key aggregation may also result in the addition of new non-Key fields to the Aggregated Flows, namely Original Flow counters and unique reduced key counters; these are treated in more detail in Section 5.2.1 and Section 5.2.2, respectively.
In any key aggregation operation, reduction and/or replacement may be applied any number of times in any order. Which of these operations are supported by a given implementation is implementation- and application-dependent. Key aggregation may aggregate Original Flows with different sets of Flow Keys; only the Flow Keys of the resulting Aggregated Flows

of any given Key aggregation operation need to contain the same set of fields.

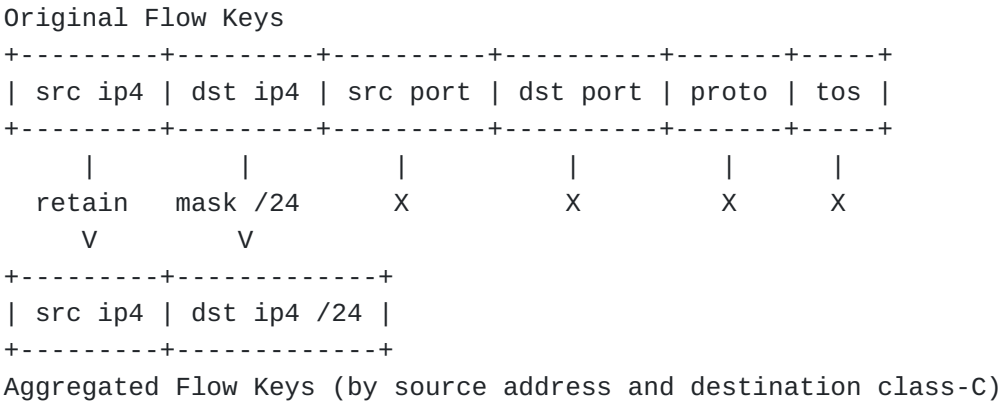


Figure 6 illustrates an example reduction operation, aggregation by source address and destination class C network. Here, the port, protocol, and type-of-service information is removed from the Flow Key, the source address is retained, and the destination address is masked by dropping the low 8 bits.

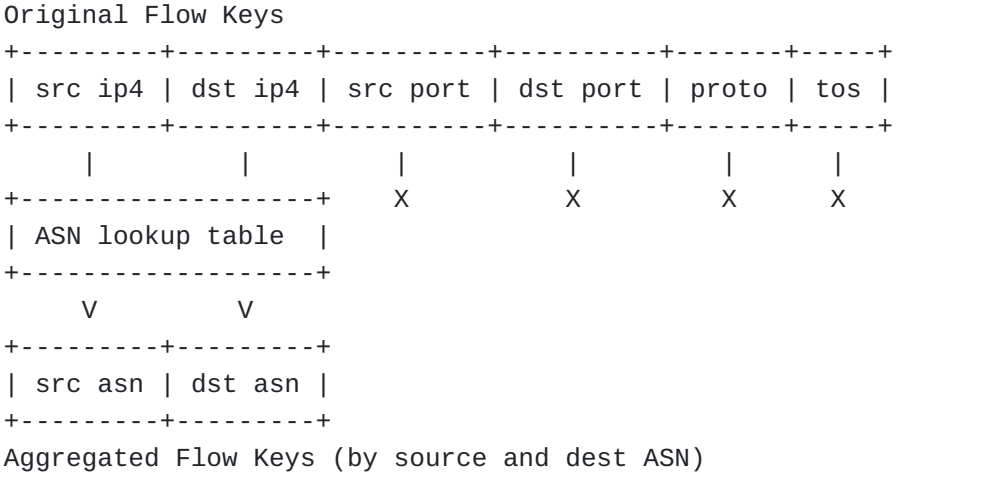


Figure 7 illustrates an example reduction and replacement operation, aggregation by source and destination Border Gateway Protocol (BGP) Autonomous System Number (ASN) without ASN information available in the Original Flow. Here, the port, protocol, and type-of-service information is removed from the Flow Keys, while the source and destination addresses are run though an IP address to ASN lookup table, and the Aggregated Flow Keys are made up of the resulting source and destination ASNs.

5.2.1. Counting Original Flows

When aggregating multiple Original Flows into an Aggregated Flow, it is often useful to know how many Original Flows are present in the

Aggregated Flow. This document introduces four new information elements in Section 7.2 to export these counters.

There are two possible ways to count Original Flows, which we call here conservative and non-conservative. Conservative flow counting has the property that each Original Flow contributes exactly one to the total flow count within a set of Contributing Flows. In other words, conservative flow counters are distributed just as any other counter during interval distribution, except each Original Flow is assumed to have a flow count of one. When a count for an Original Flow must be distributed across a set of Aggregated Flows, and a distribution method is used which does not account for that Original Flow completely within a single Aggregated Flow, conservative flow counting requires a fractional representation.

By contrast, non-conservative flow counting is used to count how many Contributing Flows are represented in an Aggregated Flow. Flow counters are not distributed in this case. An Original Flow which is present within N Aggregated Flows would add N to the sum of non-conservative flow counts, one to each Aggregated Flow. In other words, the sum of conservative flow counts over a set of Aggregated Flows is always equal to the number of Original Flows, while the sum of non-conservative flow counts is strictly greater than or equal to the number of Original Flows.

For example, consider Flows A, B, and C as illustrated in Figure 5. Assume that the key aggregation step aggregates the keys of these three Flows to the same aggregated Flow Key, and that start interval counter distribution is in effect. The conservative flow count for interval 0 is 3 (since Flows A, B, and C all begin in this interval), and for the other two intervals is 0. The non-conservative flow count for interval 0 is also 3 (due to the presence of Flows A, B, and C), for interval 1 is 2 (Flows B and C), and for interval 2 is 1 (Flow C). The sum of the conservative counts 3 + 0 + 0 = 3, the number of Original Flows; while the sum of the non-conservative counts 3 + 2 + 1 = 6.

Note that the active and inactive timeouts used to generate Original Flows, as well as the cache policy used to generate those Flows, have an effect on how meaningful either the conservative or non-conservative flow count will be during aggregation. In general, all the Original Exporters producing Original Flows to be aggregated SHOULD be aggregated using caches configured identically or similarly. Original Exporters using the IPFIX Configuration Model SHOULD be configured to export Flows with equal or similar activeTimeout and inactiveTimeout configuration values, and the same cacheMode, as defined in section 4.3 of [I-D.ietf-ipfix-configuration-model].

5.2.2. Counting Distinct Key Values

One common case in aggregation is counting distinct key values that were reduced away during key aggregation. The most common use case for this is counting distinct hosts per Flow Key; for example, in host characterization or anomaly detection, distinct sources per destination

or distinct destinations per source are common metrics. These new non-Key fields are added during key aggregation.
For such applications, Information Elements for distinct counts of IPv4 and IPv6 addresses are defined in Section 7.3. These are named distinctCountOf(KeyName). Additional such Information Elements SHOULD be registered with IANA on an as-needed basis.

5.3. Spatial Aggregation of Non-Key Fields

Aggregation operations may also lead to the addition of value fields demoted from key fields, or derived from other value fields in the Original Flows. Specific cases of this are treated in the subsections below.

5.3.1. Counter Statistics

Some applications of aggregation may benefit from computing different statistics than those native to each non-key field (i.e., union for flags, sum for counters). For example, minimum and maximum packet counts per Flow, mean bytes per packet per Contributing Flow, and so on. Certain Information Elements for these applications are already provided in the IANA IPFIX Information Elements registry (http://www.iana.org/assignments/ipfix/ipfix.html (e.g. minimumIpTotalLength). A complete specification of additional aggregate counter statistics is outside the scope of this document, and should be added in the future to the IANA IPFIX Information Elements registry on a per-application, as-needed basis.

5.3.2. Derivation of New Values from Flow Keys and non-Key fields

More complex operations may lead to other derived fields being generated from the set of values or Flow Keys reduced away during aggregation. A prime example of this is sample entropy calculation. This counts distinct values and frequency, so is similar to distinct key counting as in Section 5.2.2, but may be applied to the distribution of values for any flow field.
Sample entropy calculation provides a one-number normalized representation of the value spread and is useful for annomaly detection. The behaviour of entropy statistics is such that a small number of keys showing up very often drives the entropy value down towards zero, while a large number of keys, each showing up with lower frequency drives the entropy value up.
Entropy statistics are generally useful for address-like keys, like IP addresses, port numbers, AS numbers, etc. They can also be done on flow length, flow duration fields and the like, even if this generally yields less distinct value shifts when the traffic mix changes.
As a practical example, one host scanning a lot of other hosts will drive source IP entropy down and target IP entropy up. A similar effect can be observed for ports. This pattern can also be caused by the scan-

traffic of a fast Internet worm. A second example would be a DDoS flooding attack against a single target (or small number of targets) which drives source IP entropy up and target IP entropy down.
A complete specification of additional derived values or entropy information elements is outside the scope of this document. Any such Information Elements should be added in the future to the IANA IPFIX Information Elements registry on a per-application, as-needed basis. However, in the special case of entropy calculations, to support comparability of entropies of fields with different bit sizes, entropy SHOULD be represented as a float32 or float64 value normalized to the range [0..1].

5.4. Aggregation Combination

Interval distribution and key aggregation together may generate multiple Partially Aggregated Flows covering the same time interval with the same set of Flow Key values. The process of combining these Partially Aggregated Flows into a single Aggregated Flow is called aggregation combination. In general, non-Key values from multiple Contributing Flows are combined using the same operation by which values are combined from packets to form Flows for each Information Element. Counters are summed, averages are averaged, flags are unioned, and so on.

6. Additional Considerations and Special Cases in Flow Aggregation

6.1. Exact versus Approximate Counting during Aggregation

In certain circumstances, particularly involving aggregation by devices with limited resources, and in situations where exact aggregated counts are less important than relative magnitudes (e.g. driving graphical displays), counter distribution during key aggregation may be performed by approximate counting means (e.g. Bloom filters). The choice to use approximate counting is implementation- and application-dependent.

6.2. Considerations for Aggregation of Sampled Flows

The accuracy of Aggregated Flows may also be affected by sampling of the Original Flows, or sampling of packets making up the Original Flows. The effect of sampling on flow aggregation is still an open research question. However, to maximize the comparability of Aggregated Flows, aggregation of sampled Flows SHOULD only use Original Flows sampled using the same sampling rate and sampling algorithm, or Flows created from packets sampled using the same sampling rate and sampling algorithm. For more on packet sampling within IPFIX, see [RFC5476]. For more on Flow sampling within the IPFIX Mediator Framework, see [I-D.ietf-ipfix-flow-selection-tech].

6.3. Considerations for Aggregation of Heterogeneous Flows

Aggregation may be applied to Original Flows from different sources and of different types (i.e., represented using different, perhaps wildly-different Templates). When the goal is to separate the heterogeneous Original Flows and aggregate them into heterogeneous Aggregated Flows, each aggregation should be done at its own Intermediate Aggregation Process. The Observation Domain ID on the Messages containing the output Aggregated Flows can be used to identify the different Processes, and to segregate the output.
However, when the goal is to aggregate these Flows into a single stream of Aggregated Flows representing one type of data, and if the Original Flows may represent the same original packet at two different Observation Points, the Original Flows should be correlated to ensure that each packet is only represented in a single Aggregated Flow or set of Aggregated Flows differing only by aggregation interval.

7. Export of Aggregated IP Flows using IPFIX

In general, Aggregated Flows are exported in IPFIX as any normal Flow. However, certain aspects of Aggregated Flow export benefit from additional guidelines, or new Information Elements to represent aggregation metadata or information generated during aggregation. These are detailed in the following subsections.

7.1. Time Interval Export

Since an Aggregated Flow is simply a Flow, the existing timestamp Information Elements in the IPFIX Information Model (e.g., flowStartMilliseconds, flowEndNanoseconds) are sufficient to specify the time interval for aggregation. Therefore, this document specifies no new aggregation-specific Information Elements for exporting time interval information.
Each Aggregated Flow SHOULD contain both an interval start and interval end timestamp. If an exporter of Aggregated Flows omits the interval end timestamp from each Aggregated Flow, the time interval for Aggregated Flows within an Observation Domain and Transport Session MUST be regular and constant. However, note that this approach might lead to interoperability problems when exporting Aggregated Flows to non-aggregation-aware Collecting Processes and downstream analysis tasks; therefore, an Exporting Process capable of exporting only interval start timestamps MUST provide a configuration option to export interval end timestamps as well.

7.2. Flow Count Export

The following four Information Elements are defined to count Original Flows as discussed in Section 5.2.1.

7.2.1. originalFlowsPresent

7.2.2. originalFlowsInitiated

7.2.3. originalFlowsCompleted

7.2.4. deltaFlowCount

7.3. Distinct Host Export

The following four Information Elements represent the distinct counts of source and destination network-layer addresses, used to export distinct host counts reduced away during key aggregation.

7.3.1. distinctCountOfSourceIPAddress

7.3.2. distinctCountOfDestinationIPAddress

7.3.3. distinctCountOfSourceIPv4Address

7.3.4. distinctCountOfDestinationIPv4Address

7.3.5. distinctCountOfSourceIPv6Address

7.3.6. distinctCountOfDestinationIPv6Address

7.4. Aggregate Counter Distribution Export

When exporting counters distributed among Aggregated Flows, as described in Section 5.1.1, the Exporting Process MAY export an Aggregate Counter Distribution Record for each Template describing Aggregated Flow records; this Options Template is described below. It uses the valueDistributionMethod Information Element, also defined below. Since in many cases distribution is simple, accounting the counters from Contributing Flows to the first Interval to which they contribute, this is default situation, for which no Aggregate Counter Distribution Record is necessary; Aggregate Counter Distribution Records are only applicable in more exotic situations, such as using an Aggregation Interval smaller than the durations of Original Flows.

7.4.1. Aggregate Counter Distribution Options Template

IE	Description
templateId [scope]	The Template ID of the Template defining the Aggregated Flows to which this distribution option applies. This Information Element MUST be defined as a Scope Field.
valueDistributionMethod	

IE	Description
	The method used to distribute the counters for the Aggregated Flows defined by the associated Template.

This Options Template defines the Aggregate Counter Distribution Record, which allows the binding of a value distribution method to a Template ID. Note that this Options Template causes the valueDistributionMethod to be implicitly scoped to the Observation Domain ID of the IPFIX Message containing the Aggregate Counter Distribution Record. This is used to signal to the Collecting Process how the counters were distributed. The fields are as below:

7.4.2. valueDistributionMethod Information Element

8. Examples

In these examples, the same data, described by the same template, will be aggregated multiple different ways; this illustrates the various different functions which could be implemented by Intermediate Aggregation Processes. Templates are shown in iespec format as introduced in [I-D.trammell-ipfix-ie-doctors]. The source data format is a simplified flow: timestamps, traditional 5-tuple, and octet count. The template is shown in Figure 8.

```
flowStartMilliseconds(152)[8]
flowEndMilliseconds(153)[8]
sourceIPv4Address(8)[4]
destinationIPv4Address(12)[4]
sourceTransportPort(7)[2]
destinationTransportPort(11)[2]
protocolIdentifier(4)[1]
octetDeltaCount(1)[8]
```

The data records given as input to the examples in this section are shown below, in the format "flowStartMilliseconds-flowEndMilliseconds sourceIPv4Address:sourceTransportPort -> destinationIPv4Address:destinationTransportPort (protocolIdentifier) octetDeltaCount"; timestamps are given in H:MM:SS.sss format.

```
9:00:00.138-9:00:00.138 192.0.2.2:47113   -> 192.0.2.131:53   (17)   119
9:00:03.246-9:00:03.246 192.0.2.2:22153   -> 192.0.2.131:53   (17)    83
9:00:00.478-9:00:03.486 192.0.2.2:52420   -> 198.51.100.2:443  (6)  1637
9:00:07.172-9:00:07.172 192.0.2.3:56047   -> 192.0.2.131:53   (17)   111
9:00:07.309-9:00:14.861 192.0.2.3:41183   -> 198.51.100.67:80  (6) 16838
9:00:03.556-9:00:19.876 192.0.2.2:17606   -> 198.51.100.68:80  (6) 11538
9:00:25.210-9:00:25.210 192.0.2.3:47113   -> 192.0.2.131:53   (17)   119
9:00:26.358-9:00:30.198 192.0.2.3:48458   -> 198.51.100.133:80 (6)  2973
9:00:29.213-9:01:00.061 192.0.2.4:61295   -> 198.51.100.2:443  (6)  8350
9:04:00.207-9:04:04.431 203.0.113.3:41256 -> 198.51.100.133:80 (6)   778
9:03:59.624-9:04:06.984 203.0.113.3:51662 -> 198.51.100.3:80   (6)   883
9:00:30.532-9:06:15.402 192.0.2.2:37581   -> 198.51.100.2:80   (6) 15420
9:06:56.813-9:06:59.821 203.0.113.3:52572 -> 198.51.100.2:443  (6)  1637
9:06:30.565-9:07:00.261 203.0.113.3:49914 -> 197.51.100.133:80 (6)   561
9:06:55.160-9:07:05.208 192.0.2.2:50824   -> 198.51.100.2:443  (6)  1899
9:06:49.322-9:07:05.322 192.0.2.3:34597   -> 198.51.100.3:80   (6)  1284
9:07:05.849-9:07:09.625 203.0.113.3:58907 -> 198.51.100.4:80   (6)  2670
9:10:45.161-9:10:45.161 192.0.2.4:22478   -> 192.0.2.131:53   (17)    75
9:10:45.209-9:11:01.465 192.0.2.4:49513   -> 198.51.100.68:80  (6)  3374
9:10:57.094-9:11:00.614 192.0.2.4:64832   -> 198.51.100.67:80  (6)   138
9:10:59.770-9:11:02.842 192.0.2.3:60833   -> 198.51.100.69:443 (6)  2325
9:02:18.390-9:13:46.598 203.0.113.3:39586 -> 198.51.100.17:80  (6) 11200
9:13:53.933-9:14:06.605 192.0.2.2:19638   -> 198.51.100.3:80   (6)  2869
9:13:02.864-9:14:08.720 192.0.2.3:40429   -> 198.51.100.4:80   (6) 18289
```

8.1. Traffic Time-Series per Source

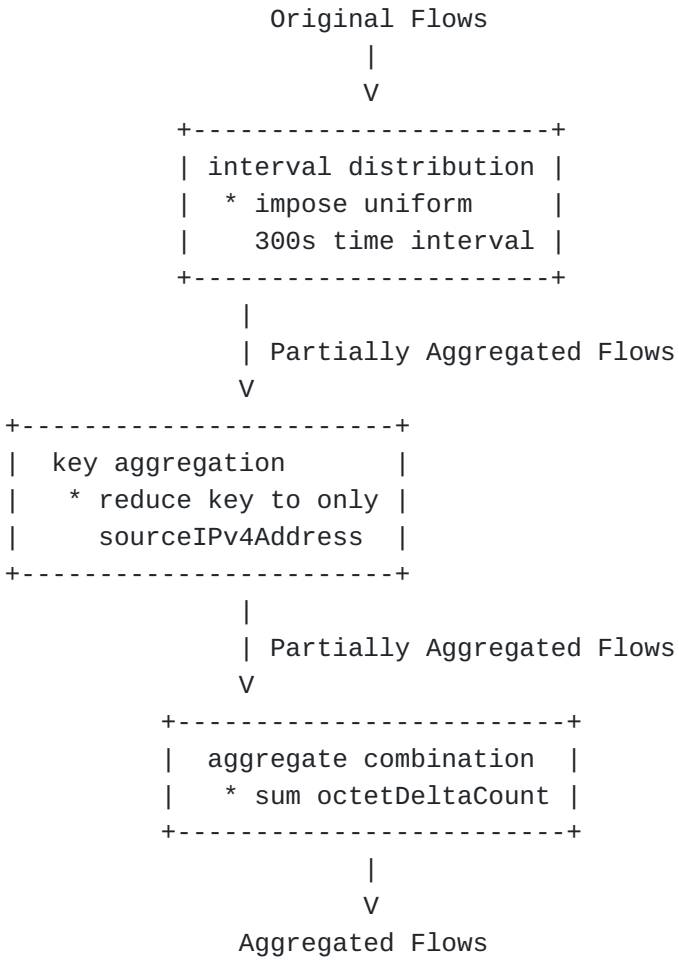
Aggregating flows by source IP address in time series (i.e., with a regular interval) can be used in subsequent heavy-hitter analysis and as a source parameter for statistical anomaly detection techniques. Here, the Intermediate Aggregation Process imposes an interval, aggregates the key to remove all key fields other than the source IP address, then combines the result into a stream of Aggregated Flows. The imposed interval of 5 minutes is longer than the majority of flows; for those flows crossing interval boundaries, the entire flow is accounted to the interval containing the start time of the flow.

In this example the Partially Aggregated Flows after each conceptual operation in the Intermediate Aggregation Process are shown. These are meant to be illustrative of the conceptual operations only, and not to suggest an implementation (indeed, the example shown here would not necessarily be the most efficient method for performing these operations). Subsequent examples will omit the Partially Aggregated Flows for brevity.

The input to this process could be any Flow Record containing a source IP address and octet counter; consider for this example the template and data from the introduction. The Intermediate Aggregation Process would then output records containing just timestamps, source IP, and octetDeltaCount, as in Figure 10.

```
flowStartMilliseconds(152)[8]
flowEndMilliseconds(153)[8]
sourceIPv4Address(8)[4]
octetDeltaCount(1)[8]
```

Assume the goal is to get 5-minute time series of octet counts per source IP address. The aggregation operations would then be arranged as in Figure 11.



After the interval distribution step, only the time intervals have changed; the Partially Aggregated flows are shown in Figure 12. Note that interval distribution follows the default Start Interval policy; that is, the entire flow is accounted to the interval containing the flow's start time.

```
9:00:00.000-9:05:00.000 192.0.2.2:47113   -> 192.0.2.131:53   (17)   119
9:00:00.000-9:05:00.000 192.0.2.2:22153   -> 192.0.2.131:53   (17)    83
9:00:00.000-9:05:00.000 192.0.2.2:52420   -> 198.51.100.2:443  (6)  1637
9:00:00.000-9:05:00.000 192.0.2.3:56047   -> 192.0.2.131:53   (17)   111
9:00:00.000-9:05:00.000 192.0.2.3:41183   -> 198.51.100.67:80  (6) 16838
9:00:00.000-9:05:00.000 192.0.2.2:17606   -> 198.51.100.68:80  (6) 11538
9:00:00.000-9:05:00.000 192.0.2.3:47113   -> 192.0.2.131:53   (17)   119
9:00:00.000-9:05:00.000 192.0.2.3:48458   -> 198.51.100.133:80 (6)  2973
9:00:00.000-9:05:00.000 192.0.2.4:61295   -> 198.51.100.2:443  (6)  8350
9:00:00.000-9:05:00.000 203.0.113.3:41256 -> 198.51.100.133:80 (6)   778
9:00:00.000-9:05:00.000 203.0.113.3:51662 -> 198.51.100.3:80   (6)   883
9:00:00.000-9:05:00.000 192.0.2.2:37581   -> 198.51.100.2:80   (6) 15420
9:00:00.000-9:05:00.000 203.0.113.3:39586 -> 198.51.100.17:80  (6) 11200
9:05:00.000-9:10:00.000 203.0.113.3:52572 -> 198.51.100.2:443  (6)  1637
9:05:00.000-9:10:00.000 203.0.113.3:49914 -> 197.51.100.133:80 (6)   561
9:05:00.000-9:10:00.000 192.0.2.2:50824   -> 198.51.100.2:443  (6)  1899
9:05:00.000-9:10:00.000 192.0.2.3:34597   -> 198.51.100.3:80   (6)  1284
9:05:00.000-9:10:00.000 203.0.113.3:58907 -> 198.51.100.4:80   (6)  2670
9:10:00.000-9:15:00.000 192.0.2.4:22478   -> 192.0.2.131:53   (17)    75
9:10:00.000-9:15:00.000 192.0.2.4:49513   -> 198.51.100.68:80  (6)  3374
9:10:00.000-9:15:00.000 192.0.2.4:64832   -> 198.51.100.67:80  (6)   138
9:10:00.000-9:15:00.000 192.0.2.3:60833   -> 198.51.100.69:443 (6)  2325
9:10:00.000-9:15:00.000 192.0.2.2:19638   -> 198.51.100.3:80   (6)  2869
9:10:00.000-9:15:00.000 192.0.2.3:40429   -> 198.51.100.4:80   (6) 18289
```

After the key aggregation step, all Flow Keys except the source IP address have been discarded, as shown in Figure 13. This leaves duplicate Partially Aggregated flows to be combined in the final operation.

```
9:00:00.000-9:05:00.000 192.0.2.2      119
9:00:00.000-9:05:00.000 192.0.2.2       83
9:00:00.000-9:05:00.000 192.0.2.2     1637
9:00:00.000-9:05:00.000 192.0.2.3      111
9:00:00.000-9:05:00.000 192.0.2.3    16838
9:00:00.000-9:05:00.000 192.0.2.2    11538
9:00:00.000-9:05:00.000 192.0.2.3      119
9:00:00.000-9:05:00.000 192.0.2.3     2973
9:00:00.000-9:05:00.000 192.0.2.4     8350
9:00:00.000-9:05:00.000 203.0.113.3    778
9:00:00.000-9:05:00.000 203.0.113.3    883
9:05:00.000-9:10:00.000 203.0.113.3   1637
9:05:00.000-9:10:00.000 203.0.113.3    561
9:05:00.000-9:10:00.000 192.0.2.2     1899
9:05:00.000-9:10:00.000 192.0.2.3     1284
9:05:00.000-9:10:00.000 203.0.113.3   2670
9:10:00.000-9:15:00.000 192.0.2.4       75
9:10:00.000-9:15:00.000 192.0.2.4     3374
9:10:00.000-9:15:00.000 192.0.2.4      138
9:10:00.000-9:15:00.000 192.0.2.3     2325
9:10:00.000-9:15:00.000 192.0.2.2     2869
9:10:00.000-9:15:00.000 192.0.2.3    18289
```

Aggregate combination sums the counters per key and interval; the summations of the first two keys and intervals are shown in detail in Figure 14.

```
  9:00:00.000-9:05:00.000 192.0.2.2      119
  9:00:00.000-9:05:00.000 192.0.2.2       83
  9:00:00.000-9:05:00.000 192.0.2.2     1637
  9:00:00.000-9:05:00.000 192.0.2.2    11538
+ 9:00:00.000-9:05:00.000 192.0.2.2    15420
                                        -----
= 9:00:00.000-9:05:00.000 192.0.2.2    28797

  9:00:00.000-9:05:00.000 192.0.2.3      111
  9:00:00.000-9:05:00.000 192.0.2.3    16838
  9:00:00.000-9:05:00.000 192.0.2.3      119
+ 9:00:00.000-9:05:00.000 192.0.2.3     2973
                                        -----
= 9:00:00.000-9:05:00.000 192.0.2.3    20041
```

Applying this to each set of Partially Aggregated Flows to produce the final Aggregated Flows shown in Figure 15 to be exported by the template in Figure 10.

```
9:00:00.000-9:05:00.000 192.0.2.2    28797
9:00:00.000-9:05:00.000 192.0.2.3    20041
9:00:00.000-9:05:00.000 192.0.2.4     8350
9:00:00.000-9:05:00.000 203.0.113.3  12861
9:05:00.000-9:10:00.000 192.0.2.2     1899
9:05:00.000-9:10:00.000 192.0.2.3     1284
9:05:00.000-9:10:00.000 203.0.113.3   4868
9:10:00.000-9:15:00.000 192.0.2.2     2869
9:10:00.000-9:15:00.000 192.0.2.3    20594
9:10:00.000-9:15:00.000 192.0.2.4     3587
```

8.2. Core Traffic Matrix

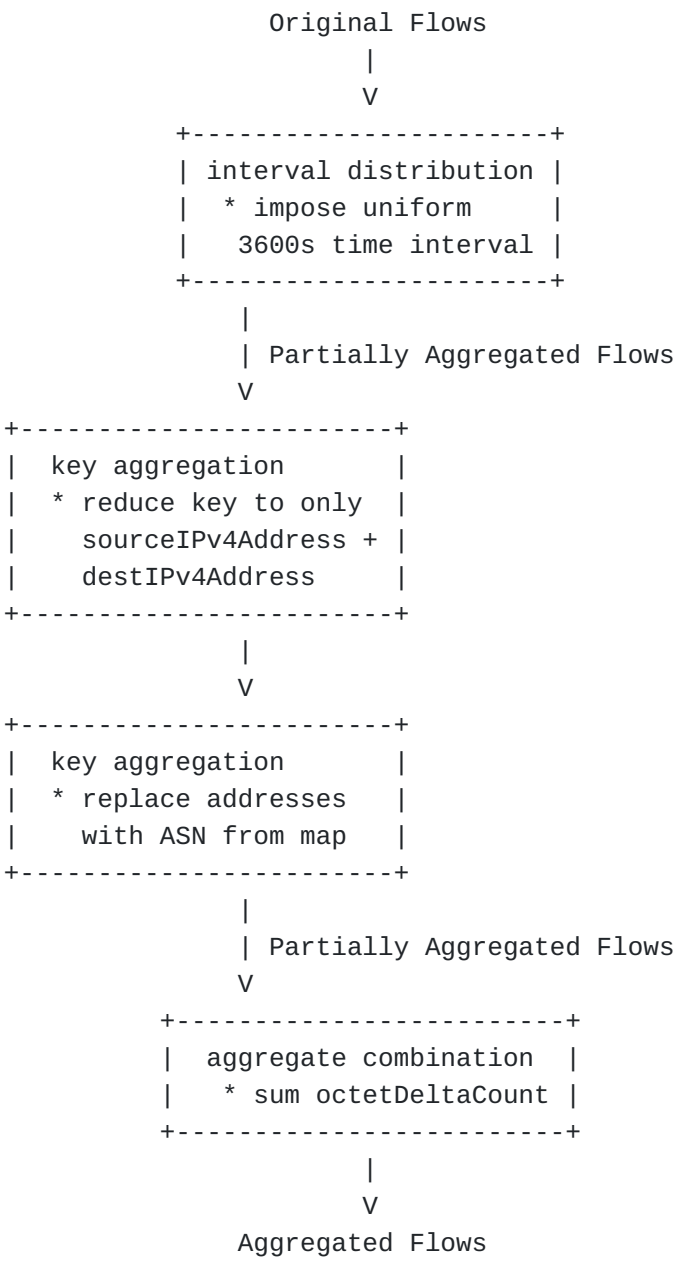
Aggregating flows by source and destination autonomous system number in time series is used to generate core traffic matrices. The core traffic matrix provides a view of the state of the routes within a network, and can be used for long-term planning of changes to network design based on traffic demand. Here, imposed time intervals are generally much longer than active flow timeouts. The traffic matrix is reported in terms of octets, packets, and flows, as each of these values may have a subtly different effect on capacity planning.
This example demonstrates key aggregation using derived keys and original flow counting. While some Original Flows may be generated by Exporting Processes on forwarding devices, and therefore contain the bgpSourceAsNumber and bgpDestinationAsNumber Information Elements, Original Flows from Exporting Processes on dedicated measurement devices will contain only a destinationIPv[46]Address. For these flows, the Mediator must look up a next hop AS from a IP to AS table, replacing source and destination addresses with AS numbers. The table used in this example is shown in Figure 16. (Note that due to limited example address space, in this example we ignore the common practice of routing only blocks of /24 or larger).

```
prefix           ASN
192.0.2.0/25     64496
192.0.2.128/25   64497
198.51.100/24    64498
203.0.113.0/24   64499
```

The template for Aggregated Flows produced by this example is shown in Figure 17.

```
flowStartMilliseconds(152)[8]
flowEndMilliseconds(153)[8]
bgpSourceAsNumber(16)[4]
bgpDestinationAsNumber(17)[4]
octetDeltaCount(1)[8]
```

Assume the goal is to get 60-minute time series of octet counts per source/destination ASN pair. The aggregation operations would then be arranged as in Figure 18.



After the interval distribution step, only the time intervals have changed; the Partially Aggregated flows are shown in Figure 19. Note that the flows are identical to those in interval distribution step in the previous example, except the chosen interval (1 hour, 3600 seconds) is different; therefore, all the flows fit into a single interval.

```
9:00:00.000-10:00:00.000 192.0.2.2:47113   -> 192.0.2.131:53   (17)   119
9:00:00.000-10:00:00.000 192.0.2.2:22153   -> 192.0.2.131:53   (17)    83
9:00:00.000-10:00:00.000 192.0.2.2:52420   -> 198.51.100.2:443  (6)  1637
9:00:00.000-10:00:00.000 192.0.2.3:56047   -> 192.0.2.131:53   (17)   111
9:00:00.000-10:00:00.000 192.0.2.3:41183   -> 198.51.100.67:80  (6) 16838
9:00:00.000-10:00:00.000 192.0.2.2:17606   -> 198.51.100.68:80  (6) 11538
9:00:00.000-10:00:00.000 192.0.2.3:47113   -> 192.0.2.131:53   (17)   119
9:00:00.000-10:00:00.000 192.0.2.3:48458   -> 198.51.100.133:80 (6)  2973
9:00:00.000-10:00:00.000 192.0.2.4:61295   -> 198.51.100.2:443  (6)  8350
9:00:00.000-10:00:00.000 203.0.113.3:41256 -> 198.51.100.133:80 (6)   778
9:00:00.000-10:00:00.000 203.0.113.3:51662 -> 198.51.100.3:80   (6)   883
9:00:00.000-10:00:00.000 192.0.2.2:37581   -> 198.51.100.2:80   (6) 15420
9:00:00.000-10:00:00.000 203.0.113.3:52572 -> 198.51.100.2:443  (6)  1637
9:00:00.000-10:00:00.000 203.0.113.3:49914 -> 197.51.100.133:80 (6)   561
9:00:00.000-10:00:00.000 192.0.2.2:50824   -> 198.51.100.2:443  (6)  1899
9:00:00.000-10:00:00.000 192.0.2.3:34597   -> 198.51.100.3:80   (6)  1284
9:00:00.000-10:00:00.000 203.0.113.3:58907 -> 198.51.100.4:80   (6)  2670
9:00:00.000-10:00:00.000 192.0.2.4:22478   -> 192.0.2.131:53   (17)    75
9:00:00.000-10:00:00.000 192.0.2.4:49513   -> 198.51.100.68:80  (6)  3374
9:00:00.000-10:00:00.000 192.0.2.4:64832   -> 198.51.100.67:80  (6)   138
9:00:00.000-10:00:00.000 192.0.2.3:60833   -> 198.51.100.69:443 (6)  2325
9:00:00.000-10:00:00.000 203.0.113.3:39586 -> 198.51.100.17:80  (6) 11200
9:00:00.000-10:00:00.000 192.0.2.2:19638   -> 198.51.100.3:80   (6)  2869
9:00:00.000-10:00:00.000 192.0.2.3:40429   -> 198.51.100.4:80   (6) 18289
```

The next step is to discard irrelevant key fields, and replace the source and destination addresses with source and destination AS numbers in the map; the results of these key aggregation steps are shown in Figure 20.

```
9:00:00.000-10:00:00.000 AS64496 -> AS64497    119
9:00:00.000-10:00:00.000 AS64496 -> AS64497     83
9:00:00.000-10:00:00.000 AS64496 -> AS64498   1637
9:00:00.000-10:00:00.000 AS64496 -> AS64497    111
9:00:00.000-10:00:00.000 AS64496 -> AS64498  16838
9:00:00.000-10:00:00.000 AS64496 -> AS64498  11538
9:00:00.000-10:00:00.000 AS64496 -> AS64497    119
9:00:00.000-10:00:00.000 AS64496 -> AS64498   2973
9:00:00.000-10:00:00.000 AS64496 -> AS64498   8350
9:00:00.000-10:00:00.000 AS64499 -> AS64498    778
9:00:00.000-10:00:00.000 AS64499 -> AS64498    883
9:00:00.000-10:00:00.000 AS64496 -> AS64498  15420
9:00:00.000-10:00:00.000 AS64499 -> AS64498   1637
9:00:00.000-10:00:00.000 AS64499 -> AS64498    561
9:00:00.000-10:00:00.000 AS64496 -> AS64498   1899
9:00:00.000-10:00:00.000 AS64496 -> AS64498   1284
9:00:00.000-10:00:00.000 AS64499 -> AS64498   2670
9:00:00.000-10:00:00.000 AS64496 -> AS64497     75
9:00:00.000-10:00:00.000 AS64496 -> AS64498   3374
9:00:00.000-10:00:00.000 AS64496 -> AS64498    138
9:00:00.000-10:00:00.000 AS64496 -> AS64498   2325
9:00:00.000-10:00:00.000 AS64499 -> AS64498  11200
9:00:00.000-10:00:00.000 AS64496 -> AS64498   2869
9:00:00.000-10:00:00.000 AS64496 -> AS64498  18289
```

Finally, aggregate combination sums the counters per key and interval. The resulting Aggregated Flows containing the traffic matrix, shown in Figure 21, are then exported using the template in Figure 17. Note that these aggregated flows represent a sparse matrix: AS pairs for which no traffic was received have no corresponding record in the output.

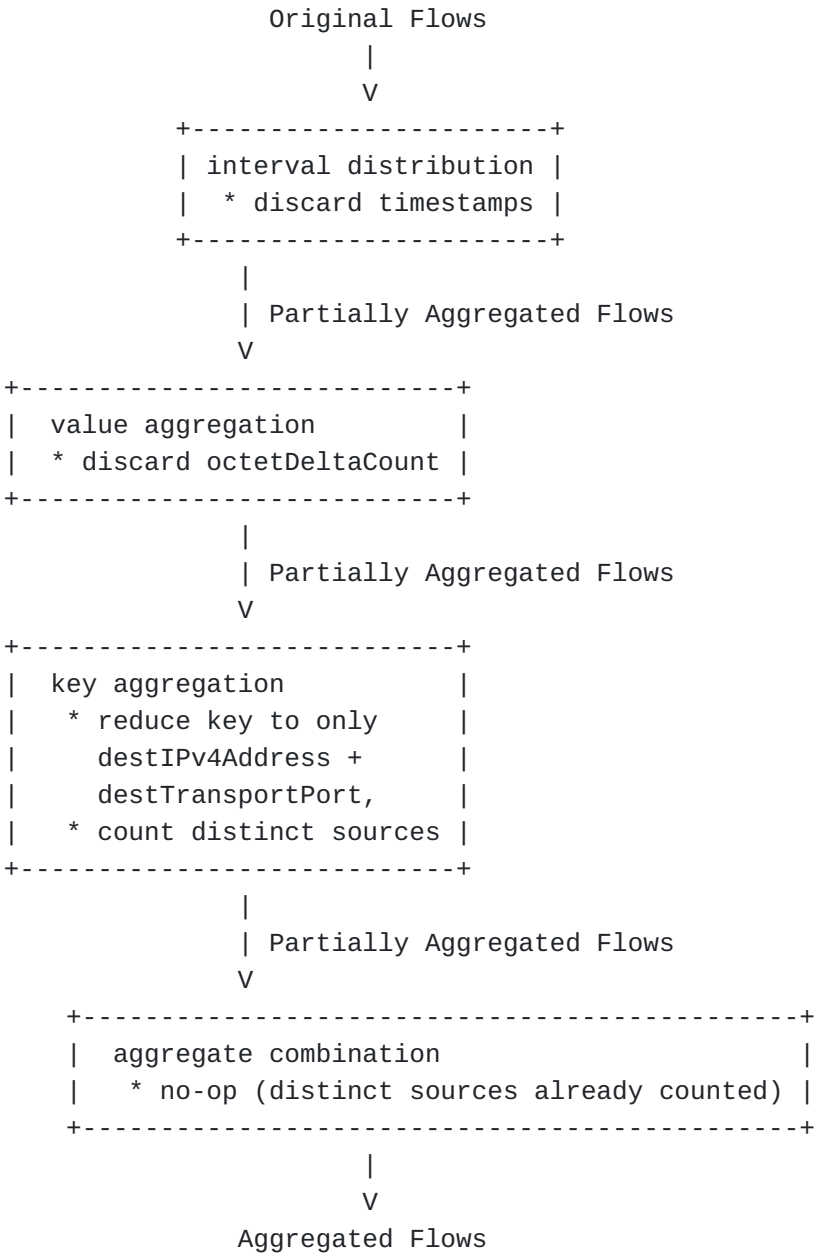
```
9:00:00.000-10:00:00.000 AS64496 -> AS64497    507
9:00:00.000-10:00:00.000 AS64496 -> AS64498  86934
9:00:00.000-10:00:00.000 AS64499 -> AS64498  17729
```

The output of this operation is suitable for re-aggregation: that is, traffic matrices from single links or observarion points can be aggregated through the same interval imposition and aggregate combination steps in order to build a traffic matrix for an entire network.

8.3. Distinct Source Count per Destination Endpoint

Aggregating flows by destination address and port, and counting distinct sources aggregated away, can be used as part of passive service inventory and host characterization approaches. This example shows aggregation as an analysis technique, performed on source data

stored in an IPFIX File. As the Transport Session in this File is bounded, removal of all timestamp information allows summarization of the entire time interval contained within the interval. Removal of timing information during interval imposition is equivalent to an infinitely long imposed time interval. This demonstrates both how infinite intervals work, and how unique counters work. The aggregation operations are summarized in Figure 22.



The template for Aggregated Flows produced by this example is shown in Figure 23.

```
destinationIPv4Address(12)[4]
destinationTransportPort(11)[2]
distinctCountOfSourceIPAddress(TBD4)[8]
```

Interval distribution, in this case, merely discards the timestamp information from the Original Flows, and as such is not shown. Likewise, the value aggregation step simply discards the octetDeltaCount value field. The key aggregation step reduces the key to the destinationIPv4Address and destinationTransportPort, counting the distinct source addresses. Since this is essentially the output of this aggregation function, the aggregate combination operation is a no-op; the resulting Aggregated Flows are shown in Figure 24.

```
destination 192.0.2.131:53      3 sources
destination 198.51.100.2:80     1 source
destination 198.51.100.2:443    3 sources
destination 198.51.100.67:80    2 sources
destination 198.51.100.68:80    2 sources
destination 198.51.100.133:80   2 sources
destination 198.51.100.3:80     3 sources
destination 198.51.100.4:80     2 sources
destination 198.51.100.17:80    1 source
destination 198.51.100.69:443   1 source
```

8.4. Traffic Time-Series per Source with Counter Distribution

Returning to the example in Section 8.1, note that our source data contains some flows with durations longer than the imposed interval of five minutes. The default method for dealing with such flows is to account them to the interval containing the flow's start time.
In this example, the same data is aggregated using the same arrangement of operations and the same output template as the as in Section 8.1, but using a different counter distribution policy, Simple Uniform Distribution, as described in Section 5.1.1. In order to do this, the Exporting Process first exports the Aggregate Counter Distribution Options Template, as in Figure 25.

```
templateId(12)[2]{scope}
valueDistribtutionMethod(TBD10)[1]
```

This is followed by an Aggregate Counter Distribution Record described by this Template; assuming the output template in Figure 10 has ID 257, this would appear as in Figure 26.

```
templateId 257: valueDistributionMethod 4 (Simple Uniform)
```

[EDITOR'S NOTE: redo these in boxdiagrams?]
Following metadata export, the aggregation steps follow as before. However, two long flows are distributed across multiple invervals in the interval imposition step, as indicated with "*" in Figure 27. Note the uneven distribution of the three-interval, 11200-octet flow into three Partially Aggregated Flows of 3733, 3733, and 3734 octets; this ensures no cumulative error is injected by the interval distribution step.

```
  9:00:00.000-9:05:00.000 192.0.2.2:47113   -> 192.0.2.131:53   (17)   119
  9:00:00.000-9:05:00.000 192.0.2.2:22153   -> 192.0.2.131:53   (17)    83
  9:00:00.000-9:05:00.000 192.0.2.2:52420   -> 198.51.100.2:443  (6)  1637
  9:00:00.000-9:05:00.000 192.0.2.3:56047   -> 192.0.2.131:53   (17)   111
  9:00:00.000-9:05:00.000 192.0.2.3:41183   -> 198.51.100.67:80  (6) 16838
  9:00:00.000-9:05:00.000 192.0.2.2:17606   -> 198.51.100.68:80  (6) 11538
  9:00:00.000-9:05:00.000 192.0.2.3:47113   -> 192.0.2.131:53   (17)   119
  9:00:00.000-9:05:00.000 192.0.2.3:48458   -> 198.51.100.133:80 (6)  2973
  9:00:00.000-9:05:00.000 192.0.2.4:61295   -> 198.51.100.2:443  (6)  8350
  9:00:00.000-9:05:00.000 203.0.113.3:41256 -> 198.51.100.133:80 (6)   778
  9:00:00.000-9:05:00.000 203.0.113.3:51662 -> 198.51.100.3:80   (6)   883
* 9:00:00.000-9:05:00.000 192.0.2.2:37581   -> 198.51.100.2:80   (6)  7710
* 9:00:00.000-9:05:00.000 203.0.113.3:39586 -> 198.51.100.17:80  (6)  3733
  9:05:00.000-9:10:00.000 203.0.113.3:52572 -> 198.51.100.2:443  (6)  1637
  9:05:00.000-9:10:00.000 203.0.113.3:49914 -> 197.51.100.133:80 (6)   561
  9:05:00.000-9:10:00.000 192.0.2.2:50824   -> 198.51.100.2:443  (6)  1899
  9:05:00.000-9:10:00.000 192.0.2.3:34597   -> 198.51.100.3:80   (6)  1284
  9:05:00.000-9:10:00.000 203.0.113.3:58907 -> 198.51.100.4:80   (6)  2670
* 9:05:00.000-9:10:00.000 192.0.2.2:37581   -> 198.51.100.2:80   (6)  7710
* 9:05:00.000-9:10:00.000 203.0.113.3:39586 -> 198.51.100.17:80  (6)  3733
  9:10:00.000-9:15:00.000 192.0.2.4:22478   -> 192.0.2.131:53   (17)    75
  9:10:00.000-9:15:00.000 192.0.2.4:49513   -> 198.51.100.68:80  (6)  3374
  9:10:00.000-9:15:00.000 192.0.2.4:64832   -> 198.51.100.67:80  (6)   138
  9:10:00.000-9:15:00.000 192.0.2.3:60833   -> 198.51.100.69:443 (6)  2325
* 9:10:00.000-9:15:00.000 203.0.113.3:39586 -> 198.51.100.17:80  (6)  3734
  9:10:00.000-9:15:00.000 192.0.2.2:19638   -> 198.51.100.3:80   (6)  2869
  9:10:00.000-9:15:00.000 192.0.2.3:40429   -> 198.51.100.4:80   (6) 18289
```

Subsequent steps are as in Section 8.1; the results, to be exported using Figure 10, are shown in Figure 28, with Aggregated Flows differing from the previous example indicated by "*".

```
* 9:00:00.000-9:05:00.000 192.0.2.2    21087
  9:00:00.000-9:05:00.000 192.0.2.3    20041
  9:00:00.000-9:05:00.000 192.0.2.4     8350
* 9:00:00.000-9:05:00.000 203.0.113.3   9394
* 9:05:00.000-9:10:00.000 192.0.2.2     9609
  9:05:00.000-9:10:00.000 192.0.2.3     1284
* 9:05:00.000-9:10:00.000 203.0.113.3   8601
  9:10:00.000-9:15:00.000 192.0.2.2     2869
  9:10:00.000-9:15:00.000 192.0.2.3    20594
  9:10:00.000-9:15:00.000 192.0.2.4     3587
* 9:10:00.000-9:15:00.000 203.0.113.3   3734
```

9. Security Considerations

This document specifies the operation of an Intermediate Aggregation Process with the IPFIX Protocol; the Security Considerations for the protocol itself in Section 11 of [RFC5101] therefore apply. In the common case that aggregation is performed on a Mediator, the Security Considerations for Mediators in Section 9 of [RFC6183] apply as well. As mentioned in Section 3, certain aggregation operations may tend to have an anyonymizing effect on flow data by obliterating sensitive identifiers. Aggregation may also be combined with anonymization within a Mediator, or as part of a chain of Mediators, to further leverage this effect. In any case in which an Intermediate Aggregation Process is applied as part of a data anonymization or protection scheme, or is used together with anonymization as described in [RFC6235], the Security Considerations in Section 9 of [RFC6235] apply.

10. IANA Considerations

This document specifies the creation of new IPFIX Information Elements in the IPFIX Information Element registry located at http://www.iana.org/assignments/ipfix, as defined in Section 7 above. IANA has assigned Information Element numbers to these Information Elements, and entered them into the registry.
[NOTE for IANA: The text TBDn should be replaced with the respective assigned Information Element numbers where they appear in this document. Note that the deltaFlowCount Information Element has been assigned the number 3, as it is compatible with the corresponding existing (reserved) NetFlow v9 Information Element. Other Information Element numbers should be assigned outside the NetFlow V9 compatibility range, as these Information Elements are not supported by NetFlow V9.]

11. Acknowledgments

Special thanks to Elisa Boschi for early work on the concepts laid out in this document. Thanks to Lothar Braun and Christian Henke for their

reviews. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

12. References

12.1. Normative References

[RFC5101]	Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
[RFC5102]	Quittek, J., Bryant, S., Claise, B., Aitken, P. and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

12.2. Informative References

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC3917]	Quittek, J., Zseby, T., Claise, B. and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
[RFC5103]	Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.
[RFC5153]	Boschi, E., Mark, L., Quittek, J., Stiemerling, M. and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC 5153, April 2008.
[RFC5470]	Sadasivan, G., Brownlee, N., Claise, B. and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.
[RFC5472]	Zseby, T., Boschi, E., Brownlee, N. and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, March 2009.
[RFC5476]	Claise, B., Johnson, A. and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.
[RFC5610]	Boschi, E., Trammell, B., Mark, L. and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, July 2009.
[RFC5655]	Trammell, B., Boschi, E., Mark, L., Zseby, T. and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.
[RFC5835]	Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

[RFC5982]	Kobayashi, A. and B. Claise, "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, August 2010.
[RFC6183]	Kobayashi, A., Claise, B., Muenz, G. and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.
[RFC6235]	Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.
[I-D.claise-ipfix-mediation-protocol]	Claise, B, Kobayashi, A and B Trammell, "Specification of the Protocol for IPFIX Mediations", Internet-Draft draft-claise-ipfix-mediation-protocol-04, July 2011.
[I-D.trammell-ipfix-ie-doctors]	Trammell, B and B Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", Internet-Draft draft-trammell-ipfix-ie-doctors-03, October 2011.
[I-D.ietf-ipfix-configuration-model]	Muenz, G, Claise, B and P Aitken, "Configuration Data Model for IPFIX and PSAMP", Internet-Draft draft-ietf-ipfix-configuration-model-10, July 2011.
[I-D.ietf-ipfix-flow-selection-tech]	D'Antonio, S, Zseby, T, Henke, C and L Peluso, "Flow Selection Techniques", Internet-Draft draft-ietf-ipfix-flow-selection-tech-09, November 2011.

Authors' Addresses

Brian Trammell Trammell Swiss Federal Institute of Technology Zurich Gloriastrasse 35 8092 Zurich, Switzerland Phone: +41 44 632 70 13 EMail: trammell@tik.ee.ethz.ch

Arno Wagner Wagner Consecom AG Bleicherweg 64a 8002 Zurich, Switzerland EMail: arno@wagner.name

Benoit Claise Claise Cisco Systems, Inc. De Kleetlaan 6a b1 1831 Diagem, Belgium Phone: +32 2 704 5622 EMail: bclaise@cisco.com