Improving Network Flow Definition: Formalization and Applicability

Petr Velan
Institute of Computer Science
Masaryk University
Brno, Czech Republic
velan@ics.muni.cz

***Abstract*—Network flow monitoring has been used for more than 20 years and has become an important part of network accounting and security. A significant effort was invested into the standardization of flow monitoring by the Internet Engineering Task Force (IETF). The flow monitoring has steadily evolved to satisfy new requirements created by the demand for increased visibility and accuracy. Therefore, it is not surprising that even the most recent flow definition created by the IETF does not consider several specifics of the flow monitoring process as it is used nowadays. This paper presents a revised flow definition that is more generic and is designed to accommodate more specific flow monitoring requirements. Moreover, we formalize our definition to avoid ambiguity and imprecision introduced by the use of natural language. One additional benefit of formalizing the flow definition is that it implicitly describes the flow creation process as well.**

***Index Terms*—network, flow, formal, definition**

I. Introduction

Network flow monitoring facilitates large scale intrusion detection and prevention systems, data analysis, capacity planning, data retention, and other operations important for network management [1], [2]. It is implemented in many commercial and open-source software products and hardware devices. The current trend in flow monitoring is to provide as much information as possible from the application layer of the observed data and provide per flow statistics to allow application performance monitoring. The interoperability of the individual implementations is limited due to the wide use of flow monitoring, the large number of measured features, and varying implementations [3].

Protocol interoperability is not the only problem that needs to be addressed. A more significant problem is that the flow creation process often differs from expectations. Hofstede et al. show that measurement artifacts that can affect data processing are often present in flow data [4]. The authors of [5] demonstrate how the active and inactive timeouts affect the flow creation process. It is clear that to be able to interpret and analyze flow data correctly, the flow creation process needs to be known and well understood.

To be able to specify the flow creation process in detail, it is necessary to agree on a common definition of what the flow is. However, the most widely used definition of flow, standardized in RFC7011 [6], does not consider several specifics of the flow monitoring process as are currently used. Moreover, the use of natural language for the flow definition leaves space for different interpretations. Therefore, this paper aims to provide more broad definition of flow which captures the idea of flow as it is widely understood by the measurement community and formalizes this definition.

The contribution of this paper is threefold. Firstly, we analyze the most recent flow definition standardized by IETF in the RFC7011 [6] and explain some of the main problems and confusions that this definition creates. Then, we provide a revised flow definition which addresses these problems. Secondly, we formalize the revised flow definition to avoid its misinterpretation. Moreover, we show that this definition is inductive and provides a recipe for the flow creation process. Lastly, we describe a universal approach for implementing the flow creation process, which is parameterized only by the choice of a flow selection function.

II. Revised Flow Definition

To be able to accurately describe the flow monitoring process, we need to have a precise definition of what a flow is. The NetFlow v9 description in [7] uses the following definition:

> *An IP Flow, also called a Flow, is defined as a set of IP packets passing an Observation Point in the network during a certain time interval. All packets that belong 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.*
>
> Cisco Systems NetFlow Services Export Version 9 [7]

The Observation Point is defined as a location where IP packets can be observed. The definition says that a flow is a set of packets within a certain time span. Furthermore, the packets in a flow have a set of common properties and these properties are either derived from data contained in the packet data or from packet treatment (e.g. next hop IP address or input interface). Since this definition is quite generic, it covers most of the common IP flow creation techniques.

The IPFIX Protocol is an internet standard [6] with its own definition of a flow that builds upon the NetFlow v9 definition. It tries to specify what “properties derived from data contained in packet data” means and differentiates two types of data. The first are the values contained in packet headers, the second type covers the characteristics of the packet itself (e.g. packet length). The definition is as follows:

> *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. Each property is defined as the result of applying a function to the values of:*
>
> 1) *one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields [5]).*
> 2) *one or more characteristics of the packet itself (e.g., number of MPLS labels)*
> 3) *one or more fields derived from packet treatment (e.g., next hop IP address, output interface)*
>
> *A packet is defined as belonging to a Flow if it completely satisfies all the defined properties of the Flow.*
>
> Specification of the IPFIX Protocol [6]

Although this definition is a part of the IPFIX internet standard, there are several problems:

1) It is not clear what a *packet header* is. One interpretation is that it includes all protocol headers in the packet up to the packet payload (i.e. application layer). However, the transport header is mentioned explicitly and the example indicates that it can also mean only network layer, in which case the data link layer is completely ignored.
2) The *characteristics of the packet* are not sufficiently described. One can interpret this as anything that cannot be computed directly from the packet header fields. The example states that a number of certain types of headers are considered as part of the packet's characteristics. The total packet length can be also included here (it was even used as an example in the early drafts in 2002).
3) The IPFIX standard limits the definition of flows only to IP traffic. However, flows are often created with the use of link layer headers. Moreover, the flow concept works even for non-IP connections, e.g. in technological networks. Therefore, the generic flow definition should allow even non-IP packets. It should be notes that the NetFlow v9 definition of flow explicitly defines IP flows, not generic flows.
4) Flows using transport header fields cannot be correctly defined for fragmented IP packets, since transport layer information is present only in the first packet fragment. Both NetFlow v9 and IPFIX definitions a set of common properties used to decide which flow the packet belongs to. This must be derived only from the single packet, which is not possible in case of fragmented packets.

In order to provide the most complete definition of flow, we must address all the above mentioned issues. The most direct solution is to start with the NetFlow v9 definition, allow non-IP packets and be clearer about deriving data from previous packets of the same flow which is used for correctly handling the packet fragmentation. Therefore, we propose a revised definition as follows:

Definition 1
A flow *is defined as a sequence of packets passing an* observation point *in the network during a certain time interval. All packets that belong to a particular* flow *have a set of common properties derived from the data contained in the packet, previous packets of the same* flow, *and from the packet treatment at the* observation point.

There are two more terms connected to flow that need to be defined: *flow key* and *flow record*. The IPFIX definition of the Flow Key needs to be adapted to our definition of flow. We can conveniently shorten the definition to the following:

Definition 2
A flow key *is a set of common properties that is used to specify a* flow.

A flow record is basically a tuple containing the flow key and other properties measured for the flow. The following definition reflects that:

Definition 3
A flow record *is a tuple which describes a particular* flow *containing values of:*

1) *the* flow key *used to specify the* flow,
2) *other properties of the* flow *derived from:*
   a) *data contained in the packets of the* flow,
   b) *the packet treatment of the* flow *at the* observation point.

To make the definitions above clearer, we provide an example of concrete properties that might be contained in a flow record in Table I. The table shows examples of flow record properties that can be derived from packet data and packet treatment. The properties can be aggregated when the derived value differs between individual packets of the flow or where counters such as the number of packets are involved. A summation function is usually applied to the number of bytes in each packet, TCP flags are aggregated using a logical OR function, the flow start timestamp is derived using a minimum function on each packet timestamp. The non-aggregated properties may be used as part of a flow key.

III. Formalization of the Flow Definition

Definition 1 states what the flow is. Although we tried to be as explicit as possible, the definition is informal and therefore subject to different interpretations. For this reason we now provide a formal definition of flow, which not only refines the informal definition, but also provides a guide to the construction of the flows.

TABLE I
EXAMPLES OF FLOW PROPERTIES

	Aggregated properties	**Non-aggregated properties**
Packet data	Number of bytes TCP flags Time to Live	Source IP address Destination port Transport protocol
Packet treatment	Number of packets Flow start timestamp	Input interface number Next-Hop IP address

Definition 4
Let P be a set of all packets. Let T be a set of packet treatment information. We define a set of extended packets

$$\widehat{P} = P \times T,$$

so that $\widehat{p} \in \widehat{P}$ denotes a packet p together with its packet treatment information. Let $\mathbb{S}$ be a set of indexes of packets observed at an observation point*:*

$$\mathbb{S} = \{1, \ldots, n\} \vee \mathbb{N},$$

where $n \in \mathbb{N}$ is the number of observed packets when the number is finite.

We denote sequence of packets and extended packets observed at an observation point *respectively:*

$$\mathcal{P} = (p_i)_{i\in\mathbb{S}},\, p_i \in P,$$
$$\widehat{\mathcal{P}} = (\widehat{p}_i)_{i\in\mathbb{S}},\, \widehat{p}_i \in \widehat{P}.$$

Both sequences are of size $|\mathbb{S}|$.

Let us now define a *flow selection function* φ which takes a sequence of extended packets and a new extended packet and decides whether they form a flow. We will use this function to determine whether a newly observed packet belongs to an existing flow.

Definition 5
Let $\widehat{P}^*$ be a set of all finite sequences of extended packets, $\widehat{P}$ be a set of extended packets. We say that a function of type

$$\varphi : \widehat{P}^* \times \widehat{P} \to \{true, false\}$$

is a flow selection function.

Before we give a formal definition of a flow, we provide the following intuition for our definition. A flow $\mathcal{F}$ is a sequence of packets defined by a sequence of extended packets with indexes in $\mathbb{S}$ and a *flow selection function* φ. We require that a packet belongs to a flow if it is determined by all previous packets of that flow. Therefore we construct the flow by induction as described in Algorithm 1.

We shall now define set $\mathbb{I}$ of indexes from $(\widehat{p}_i)$ selected using *flow selection function* φ, and flow $\mathcal{F}$ so that it conforms with the Definition 1 as follows:

```
1: Denote $\mathbb{I}$ the set of packet indexes that belong to the flow $\mathcal{F}$
2: Start with $\mathbb{I} = \emptyset$
3: while An index $k$ of the first extended packet $\widehat{p}_k$ for which $\varphi((\widehat{p}_n)_{n\in\mathbb{I}}, \widehat{p}_k) = true$ exists do
4:    Add $k$ to $\mathbb{I}$
5: end while
6: The flow $\mathcal{F}$ is a sequence of packets with indexes from $\mathbb{I}$
```

Algorithm 1: Construction of a flow.

```
1: Denote $S_1 = \mathbb{S}$
2: Set counter $i = 1$
3: repeat
4:    Apply the flow selection function $\varphi$ to extended packets with indexes in $S_i$
5:    Denote indexes of matching extended packets $\mathbb{I}_i$
6:    Flow $\mathcal{F}_i$ is a sequence of packets with indexes from $\mathbb{I}_i$
7:    Remove indexes in $\mathbb{I}_i$ from $S_i$, denote the new sequence $S_{i+1}$
8:    Increment counter $i = i + 1$
9: until $\mathcal{S}_i$ is empty
```

Algorithm 2: Construction of a sequence of flows.

Definition 6
Let $(p_i)_{i\in S}, (\widehat{p}_i)_{i\in S}, S \subseteq \mathbb{S}$ be (possibly finite) mutually corresponding sequences of packets and extended packets respectively, φ a flow selection function.

We define a flow index set $\mathbb{I} = \mathbb{I}\left((\widehat{p}_i)_{i\in S},\, \varphi\right)$ *as*

$$\mathbb{I} = \lim_{i\to\infty} J_i, \text{ where } J_i \text{ is defined inductively over } i \in \mathbb{N} \text{ as:}$$

$$J_i = \begin{cases} \{\min\{\alpha \in S \mid \varphi(\widehat{p}_\alpha) = true\}\} & \text{for } i = 1, \\ J_{i-1} \cup \{\min\{\alpha \in S \mid \alpha > \sup(J_{i-1}), \\ \quad \varphi((\widehat{p}_n)_{n\in J_{i-1}}, \widehat{p}_\alpha) = true\}\} & \text{for } i > 1. \end{cases}$$

Finally, we define flow $\mathcal{F} = \mathcal{F}\left((p_i)_{i\in S},\, \mathbb{I}\right)$ as:

$$\mathcal{F} = (p_i)_{i\in\mathbb{I}},\, p_i \in \mathcal{P}.$$

Since we need the min function to be defined for an empty set (the cases where no flow is defined and where we have already added all possible indexes from $\mathbb{S}$), we define

$$\{\min\ \emptyset\} = \emptyset$$

Definition 6 of flow creates a single flow for a sequence of extended packets $\widehat{\mathcal{P}}$ and a *flow selection function* φ. The flow $\mathcal{F}$ is selected based on the first extended packet accepted by φ. Since we naturally expect that every packet is a part of only a single flow, we can construct a sequence of flows $(\mathcal{F}_i)_{i\in\mathbb{N}}$ by induction as described in Algorithm 2.

Let us now provide a more formal definition of a sequence of flows $(\mathcal{F}_i)_{i\in\mathbb{N}}$.

Definition 7
Let $\mathcal{P}, \widehat{\mathcal{P}}$ be sequences of packets and extended packets re-

spectively, φ a flow selection function. *We define the sequence $(\mathcal{F}_i)_{i\in\mathbb{N}}$ of flows inductively:*

$$\begin{aligned}\mathcal{F}_i &= \mathcal{F}_i\left((p_j)_{j\in S_i},\, \mathbb{I}_i\right),\ \textit{where}\\ S_1 &= \mathbb{S},\\ S_i &= S_{i-1}\setminus \mathbb{I}_{i-1},\\ \mathbb{I}_i &= \mathbb{I}_i\left((\widehat{p}_j)_{j\in S_i},\, \varphi\right).\end{aligned}$$

Definition 7 provides a guide to constructing a sequence of flow records. The procedure can be easily modified to run in real time so that each newly observed extended packet can be added to the appropriate flow. In our definition, we want every packet to be a part of only a single flow. Therefore, we apply the *flow selection function φ* to each pair of existing flow (enriched by packet treatment information) and the new extended packet. Then, we add the packet to the first flow that matches. If none of the existing flows match, we apply the function φ to this packet only and start a new flow if necessary.

From this, we can see that the flow creation process depends solely upon the implementation of the *flow selection function*. We will shortly discuss common implementations in the following section.

IV. Flow Creation Process

This section describes the flow creation process using an implementation of the *flow creation function*. Before the flow records can be constructed, the packets need to be captured and processed. The process of *packet capture*, also called *packet observation* is out of the scope of this paper. A good overview of the process is provided by [8]. The goal of the packet processing is to extract the values of chosen properties of individual packets and the corresponding packet treatment information. Attributes such as IP addresses, transport protocol, and ports are used as part of *flow keys* often. The specific flow key used depends on the applied flow selection function, which is used to decide to which flow the packet belongs. Other attributes of the packets such as TCP flags or the number of bytes, are extracted as well for further analysis. We call the extracted properties *packet metadata*.

The packet metadata are aggregated to create flow records. The definition of the flow selection function requires all preceding extended packets of the same flow to determine whether a new packet belongs to particular flow. Since it is not viable to keep all packets of a flow in memory, only selected information is stored in real world implementations. All active flows have a flow record with all the necessary information stored in a *flow cache*. When a new packet arrives, the flow selection function is called for each stored flow record and the metadata of the new packet to determine, to which flow the new packet belongs. If a matching flow record is found, it is updated using the packet metadata (e.g. packet and byte counters are incremented, and the flow end timestamp is updated). If no such record exists, the packet is considered to be the first packet of a new flow and a corresponding flow record is created in the flow cache. Algorithm 3 illustrates the flow creation process. It is worth noting that the flow selection function is denoted ϕ as we refer to a concrete implementation here instead of the formal definition.

```
1:  loop
2:      Get new packet $P$
3:      Extract packet metadata $M$
4:      Set found = false
5:      for all flow record $\mathcal{F}$ in flow cache do
6:          Apply flow selection function $\phi$ to $\mathcal{F}$ and $M$
7:          if $\phi(\mathcal{F}, M) = true$ then
8:              Aggregate $M$ to $\mathcal{F}$
9:              Set found = true;
10:             break
11:         end if
12:     end for
13:     if not found then
14:         Create new flow record $\mathcal{F}$ from $M$
15:         Insert $\mathcal{F}$ into flow cache
16:     end if
17: end loop
```

Algorithm 3: Construction of flow records.

V. Conclusions

The main goal of this paper was to point out the deficiencies of the current, widely-used flow definitions and to introduce a revised flow definition to address the identified deficiencies. The main problem of the current flow definition is that it allows only IP flows, is ambiguous, and does not allow for the proper measurement of fragmented traffic.

We have proposed a revised flow definition which does not limit flow measurement to IP traffic and allows flow keys to be derived based on information contained in the previous packets of the flow as well. To address the ambiguity issues that are caused by the use of natural language for the definition, we have provided a formalized definition of the flow as well. The formal definition is inductive in its nature and provides a recipe for the flow construction process.

The presented revised flow definition and its formalization can serve as a basis for the unified description of novel flow creation processes as well as for clarifying the currently used processes. Under the revised definition, flow monitoring is no longer limited to IP packets and information from network and transport layers. Instead, it is a complex process that utilizes data from link layer and packet treatment as well as from the application payload. Therefore, we need to be more careful than ever to ensure that the flow monitoring process is well defined and understood.

Acknowledgment

This research was supported by the Security Research Programme of the Czech Republic 2015 - 2020 (BV III / 1 VS) granted by the Ministry of the Interior of the Czech Republic under No. VI20162019029 The Sharing and analysis of security events in the Czech Republic.

References

[1] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, “Review: A Survey of Network Flow Applications,” *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 567–581, Mar. 2013.

[2] M. F. Umer, M. Sher, and Y. Bi, “Flow-based intrusion detection: Techniques and challenges,” *Computers & Security*, vol. 70, pp. 238 – 254, 2017.

[3] B. Trammell. (2011) DEMONS IPFIX Interoperability Event - Final Report. [Online]. Available: https://www.ietf.org/proceedings/80/slides/ipfix-4.pdf

[4] R. Hofstede, I. Drago, A. Sperotto, R. Sadre, and A. Pras, “Measurement Artifacts in Netflow Data,” in *Proceedings of the 14th International Conference on Passive and Active Measurement*, ser. PAM’13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 1–10.

[5] J. M. Rodriguez, V. C. Español, P. B. Ros, R. Hoffmann, and K. Degner, “Empirical Analysis of Traffic to Establish a Profiled Flow Termination Timeout,” in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, July 2013, pp. 1156–1161.

[6] B. Claise, B. Trammell, and P. Aitken, “Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information,” RFC 7011 (Internet Standard), pp. 1–76, Sep. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7011.txt

[7] B. Claise, “Cisco Systems NetFlow Services Export Version 9,” RFC 3954 (Informational), pp. 1–33, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3954.txt

[8] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, “Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX,” *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 2037–2064, 2014.