# Research on Networks 2017

# Final report – University of Twente

## Overview

This document describes the findings done within Research on Networks in 2017, where we focused on two topics: we investigated more detailed flow export of IPv6-specific network features, along with ways to visualize these measurements, and, we continued work on and using the Anycast testbed. The latter resulted in a scientific publication and a M.Sc. thesis, which are provided in separate documents; the remainder of this document describes the finding of the IPv6-specific measurements.

Code produced is available on https://github.com/ut-dacs/ipfix-viz

## Table of Contents

## Enhancing IPFIX-based IPv6 measurements

Last year in Research on Networks, we started research on IPFIX measurements with an IPv6 focus. Specifically, we were interested to which extend Extension Headers affected flow measurements.

Results from last year's efforts resulted in a publication on the TMA Network Traffic Measurement and Analysis Conference in Dubin/Maynooth, and showed the potential of enhancing flow exporters (in our case, INVEA-TECH flow probes) using plugins. The measurements showed a significant amount of legitimate traffic to be 'hidden' behind Extension Headers. Analyzing the source/destination port distribution of this traffic hinted at large shares of DNS and HTTP(S). As these protocols directly affect end-user quality of experience, simply dropping all traffic containing Extension Headers was recommended against. Dropping based on Extension Headers is, however, a popular approach for network operators to secure their networks from these 'scary' Extension Headers.

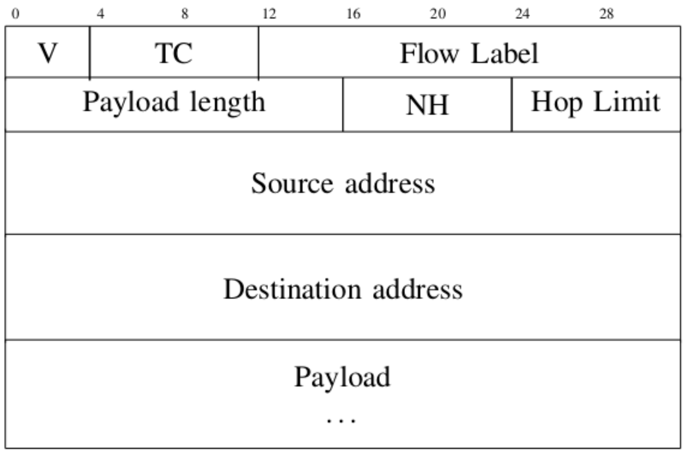
With these insights, we narrowed our focus in an attempt to extract more details from flows containing Extension Headers. Specifically, we extended the exporter plugin to enable export of ICMP information when Extension Headers are present, as well as TCP flag information. As both protocols are known to allow misuse (albeit not for IPv6 per se in the case of TCP, as e.g. a TCP SYN flood is network layer agnostic), we aimed at enabling operators to do more detailed analysis on their collected flows.

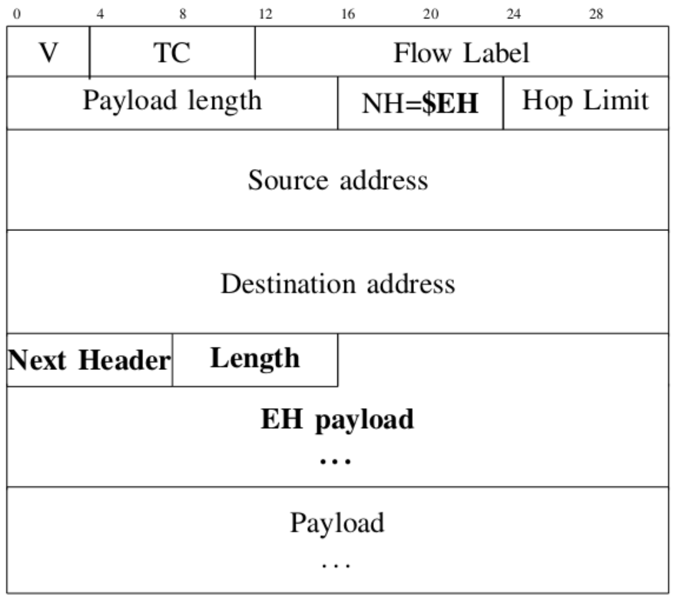
Part of the TMA publication consisted of analysis of a large amount of collected data in the search for misuse. We learned that retroactive querying on such large amounts of data is cumbersome, and finding small anomalies is not trivial. Therefore, we expanded our efforts from only collecting detailed flow data to include visualizations of said data. Combining the newly implemented export features (ICMP Type and Codes, and TCP flags) with experimental visualization allows operators to better explore flows and get a better understanding of these 'strange Extension Headers'.

## Extension Headers & Flow Measurements

Described in more detailed in the TMA publication, this section highlights the challenges of flow-based measurements on traffic containing Extension Headers.

Shown below are two abstract IPv6 packets: the first one without any Extension Header, the second one. As flow metering (the process in the flow exporter aggregating packets into flow records) uses the Next Header field in its accounting process, clearly, having an Extension Header affects this: where in the ‘normal’ packet the Next Header field will contain e.g. 6 or 17 (for respectively TCP or UDP traffic), it contains the protocol number of the Extension Header when one is present. Even though the actual upper layer protocol, thus TCP or UDP in this example, is present further down the packet, flow exporters do not extract and use it in their metering process. This results in a distorted or inaccurate view on the traffic: as transport ports are also ‘hidden behind the Extension Header’, multiple flows between a pair of hosts are possibly aggregated into a single flow. Other upper layer features, such as ICMP Type and Code, and TCP flags, are naturally hidden as well.





## Implementation

Leveraging the code from last year's development, we already have the traversal of one or more Extension Headers covered in our plugin's functionality. The plugin framework within the INVEA-TECH Flowmon exporters enables adapting the flow cache key, which is a necessity in case of the ICMP Type and Code: in many flow exporter implementations, the ICMP Type and Code carry the function of the transport protocol ports, in the sense that they distinguish multiple ICMP flows. Analog to the port numbers, the Type and Code are included in the flow cache key. This improves the accuracy of the resulting exported flow records, as multiple smaller ICMP flows are not aggregated in one flow record incorrectly describing a single flow. Naturally, this also exports in the Type and Code as features (Information Elements in IPFIX terminology), enabling operators to do detailed analysis for performance and security ends.

## ICMP Type + Code

In terms of security-related issues, one can think of ICMP-based flooding attacks that remain hidden when naive flow export (i.e., not taking into account Extension Headers and the actual upper layer features like ICMP Type and Code). Specifically for IPv6, attacks based on false ICMP Packet Too Big messages might trigger malign MTU reductions on paths, impairing quality of service on the network. However, a far more common use-case for extraction of ICMP information is basic network troubleshooting. Especially in the area of IPv6 where some operators tend to treat the network and the devices on it as if it were IPv4, and e.g. configure firewalls to drop all ICMP traffic, unexpected scenarios occur. Operators have to deal with subtle observations at best, thus requiring all details possibly available. For example, a ICMPv6 Type 1 (*Destination Unreachable*) with any Code (e.g. 3, *Address Unreachable*) with a Hop-by-Hop Extension Header will show up as a flow containing the IP address of the router sending the ICMP message, but with protocol 0 and no ports nor ICMP type/code information. Without enhanced export as we present and propose, troubleshooting is unnecessarily hard.

### Information Elements

As the ICMP Type and Code behind any possible Extension Header do not differ from those in ICMP traffic without Extension Headers, we can reuse the normal Information Elements as specified by IANA. The Code and Type are combined into a single value (the Type is shifted 8 bits to the left, then the Code is simply added), which is defined as e0id139 named 'icmpTypeCodeIPv6'.

## TCP Flags

In the same vein of troubleshooting subtle problems, TCP flags can be a valuable piece of information to operators when determining (up to a certain extent) why, when or how connections are lost. Though in the case of TCP flags hidden behind Extension Headers, certain types of attacks become invisible as

well. Detection of a SYN-flood is trivial, as one simply looks for ‘a large number of flows towards a single system, containing only the TCP SYN-flag’. When Extension Headers are present in the traffic though, a naive flow exporter will produce a flow record showing a single TCP flow because the port information is hidden in the actual upper layer, and moreover, it does not tell us anything about the flags observed in the packets. The result is ‘a single, large flow’ instead of ‘many small flows’, ergo completely different from what one would base the detection on.
Other scenarios emphasizing the usefulness of TCP flag information in flow export include investigating TCP-RST flags, send in either malicious context or caused by end system misbehavior.

## Information Elements

Similarly to the ICMP Type and Code, the TCP flags can be exported in the Information Element defined by IANA. This is the ‘tcpControlBits’ field, e0id6.

## Visualization

With the enhanced flow export available, measuring on the University of Twente network (being IPv6-enabled for years) results in a significant amount of data, both in terms of traffic and in number of flows. As we focused on subtle aspects, i.e. ICMP Types and Codes and TCP flags, going through all this data while not always knowing what to look for is cumbersome. Especially when analyzing flow data spanning a long time window, think multiple weeks or months, any possible occurrences of subtle anomalies do not pop up from the large collection of flow records.

In an attempt to combat this, we developed a small framework to aid in exploratory visualizations. The main goal is to visualize specific (small) flow features over time, while being able to easily add new visualizations based on query syntax operators are used to. Shortly put, the main requirements are:

- Loosely coupled, allowing easily switch to a different output (plotting) format
- Extensible, allowing new graphs 'on-the-fly'
- Built with/on FOSS in order to prevent any vendor lock-in or other restrictions

The development process resulted in two iterations: firstly, an RRDtool based output was implemented, but it's limitations resulted in the second, more flexible setup based on OpenTSDB and Grafana.

## Iteration #1: RRDtool

Maybe the most seen graph in anything related to networking, RRDtool is a suite allowing one to append a Round-Robin Database with values over time. With RRD graphs as our ‘plotting output’, we created to following set of components to go from IPFIX-based measurements to graphs useful for exploratory analysis of network data:

**Components:**

- Existing FlowMon/INVEA-TECH exporter, combined with the open-source IPFIXcol collector. Analogue to nfdump in the well-known and often used NfSen suite, IPFIXcol comes with fbitdump, providing equivalent ways to query collected flow data. Note that both the exporter and collector feature our enhancements to export and handle our newly implemented Information Elements.
- Configuration file describing the desired graphs in terms of fbitdump queries.
- Python-based providing the functionality to create the RRD database files based on the configuration file, as well as update existing databases with newly collected flow-data.
- Crontab to trigger both the updates and creation of new databases in case of alterations of the configuration file, thus automating everything.

## Screenshots

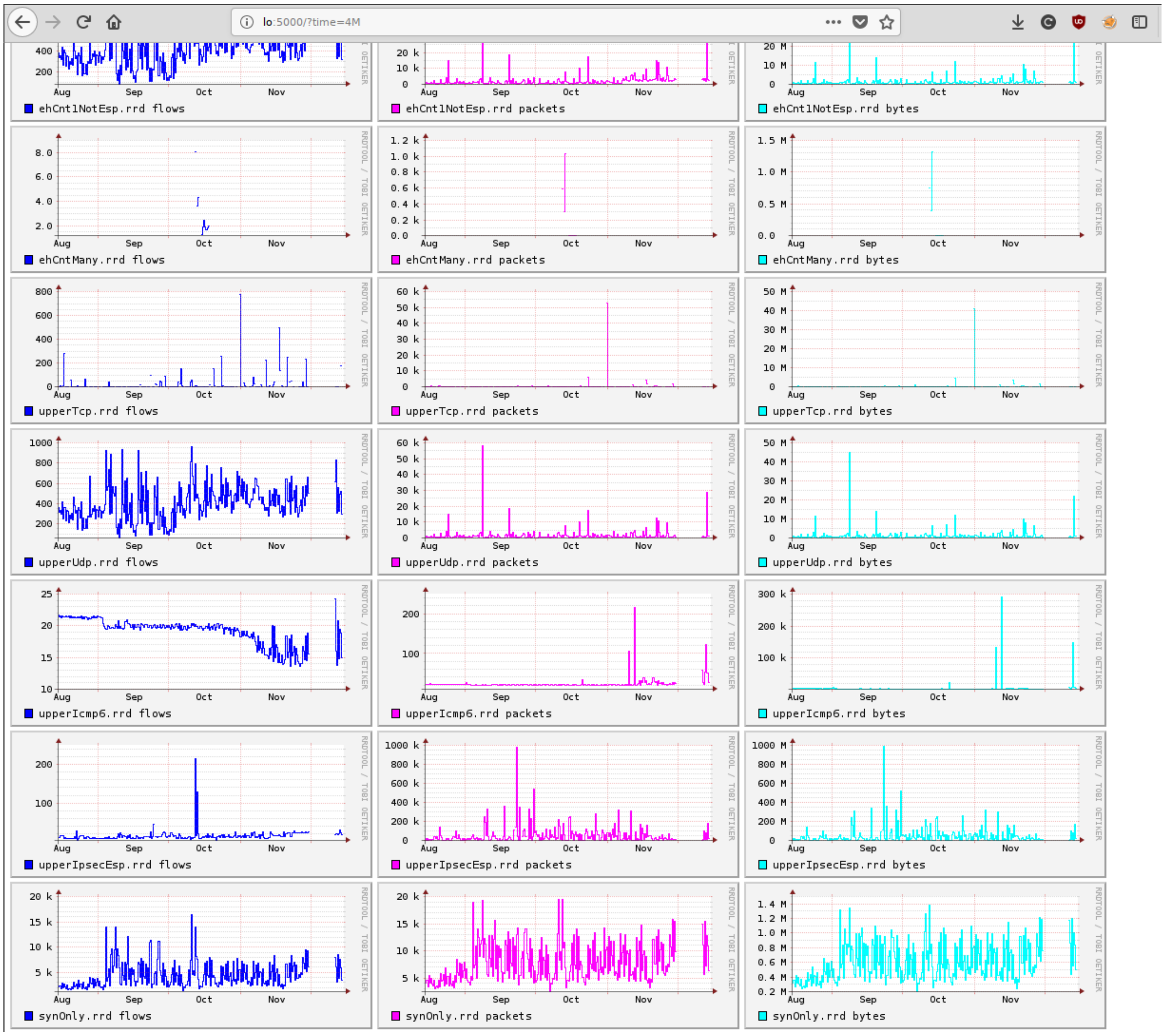


*Illustration 1: Overview of all plots, showing flows/packets/bytes per network feature*

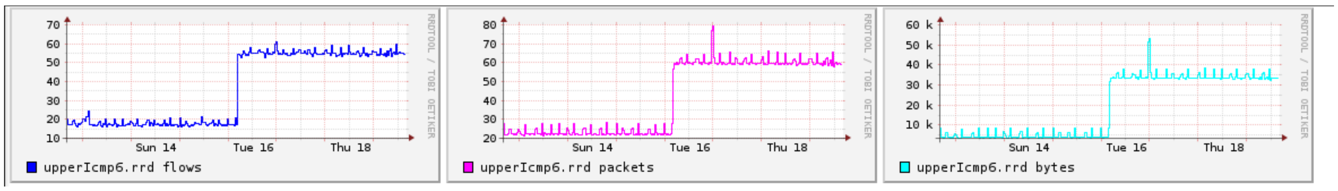


*Illustration 2: ICMPv6 behind Extension Headers: though only a handful of flows, we can still see something happened on the network*

## Take aways

We quickly learned that a brute-force approach to plotting, i.e. plotting the number of packets, bytes and flows for many queries, reveals patterns and outliers quickly. Having many plots on a single page, though structured in a certain order, we can leverage the human eye to spot any peaks, anomalies or differences over time. This, without pro-actively configuring the setup to plot specifically for these patterns. When we found anything interesting but needed a more fine-grained plot, simply adding an extra query to the configuration file proved to work. An example for this was plotting ‘everything that contains one or more extension headers’. As appeared to be the case on the network of the University of Twente, IPSEC is used in sizable numbers. In IPv6, IPSEC is specified as an Extension Header. We quickly learned that we were interested in ‘one or more extension headers but not IPSEC’, for which we had graphs directly after altering the configuration file. In other words, having the opportunity to quickly add in new graphs based on query-syntax is valuable.

## Shortcomings

While pleasing some people with their old-school aesthetics, the looks of RRD graphs can be considered outdated. But more importantly, functionally speaking, RRD databases and graphs have certain shortcomings: one can only add datapoints with timestamps later than the last datapoint, i.e. backfilling a database is not possible without complete destroying and re-creating the database. Combining multiple databases into one graph is possible, though requires significant efforts. Other features like zooming in graphs or manipulating the plotting time frame are not trivial as well.

Because we observed the value of having multiple graphs close together, showing different network traffic features thus allowing to spot relations between different features, the next iteration focuses on increasing the flexibility of the plots, to further enhance the opportunity for exploratory analysis.

## Iteration #2: OpenTSDB + Grafana

In addition to the attempt to introduce more graphing flexibility as aforementioned, a second iteration also tests how flexible the framework itself is: we replace the output (RRDtool) with more modern approaches to time-series databases and pair it with a separate graphing engine. With a strong preference towards FOSS solutions, we chose OpenTSDB for storing the time-series datapoints, and coupled it with Grafana, a popular, modern graphing engine.

With OpenTSDB, many of the shortcomings in RRD are resolved: the back-filling problem, and adding in data points (possibly from multiple sources) in general is more flexible process , and together with Grafana the combining of multiple data sources into a single graph is simple. This allows for more in-depth analysis based on our plots, as now we can:

- easily compare network traffic features from multiple sources, e.g. different IPFIX exporters on different networks
- combine multiple network traffic features within a single graph, enabling for a more aimed way to find patterns or relations between those features.

### Alterations

Input-wise, instead of writing the datapoints to RRD database files, all datapoints (describing multiple network data features) go into one OpenTSDB instance. OpenTSDB requires Hbase to store its values, which is a great future outlook when scaling up to storing many datapoints from many networks, as Hbase will allow for a Hadoop-based clustered approach.

Output wise, instead of using RRDgraph with the simple HTML wrapper, Grafana allows for direct coupling with an OpenTSDB instance and provides customizable dashboards out of the box.

## Screenshots

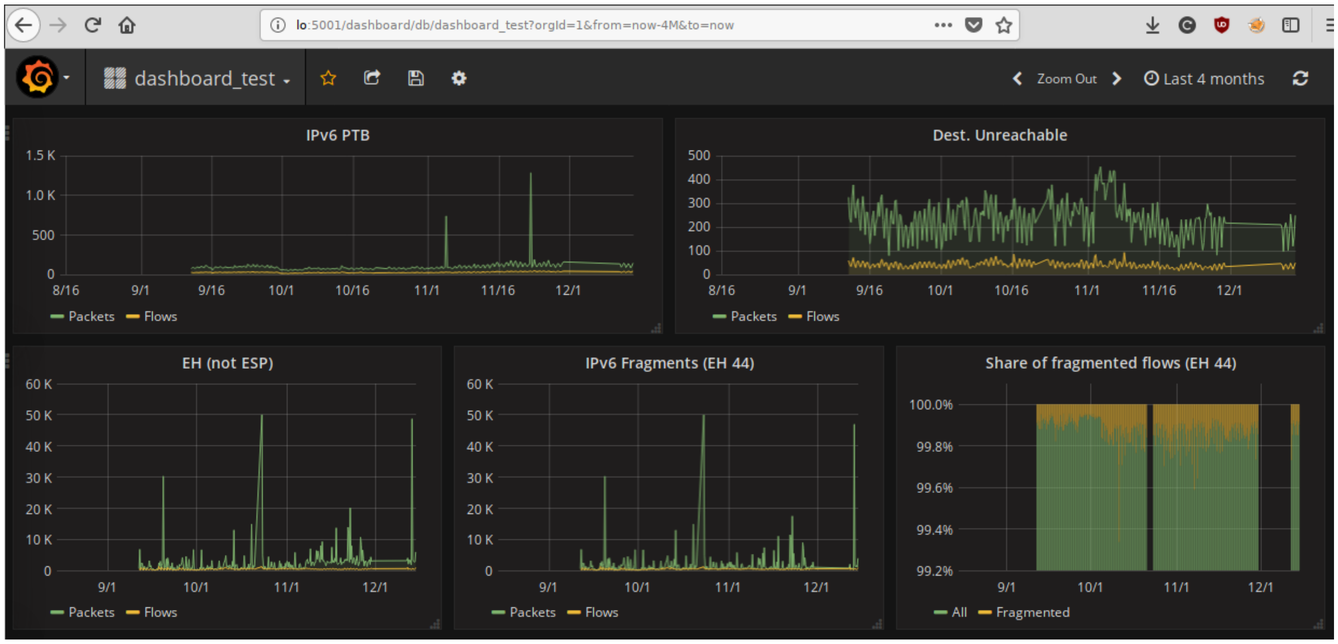


*Illustration 3: Overview of a Grafana-based dashboard, showing plots related to ICMPv6 and Fragmentation*

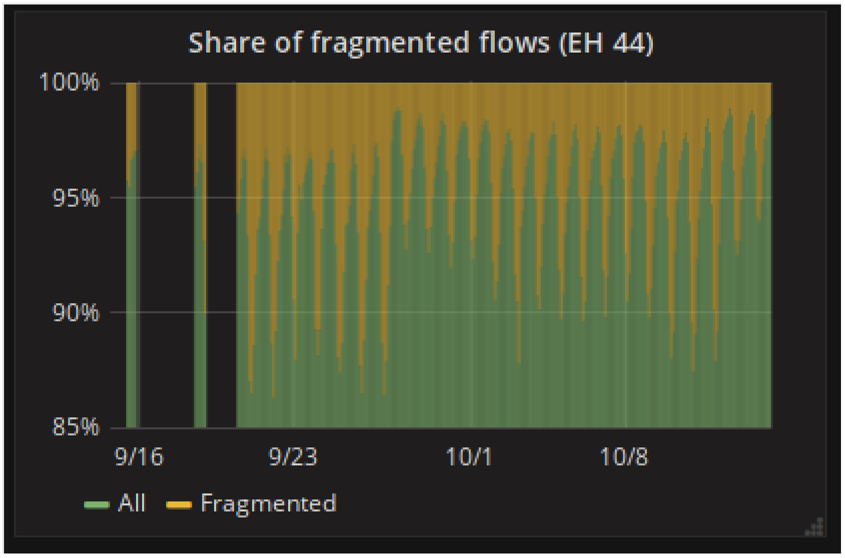


*Illustration 4: Detailed view of fragmented traffic on a different network (AS1101), showing vastly different characteristics from the UT network.*

## Take aways:

Where RRD database files were just that, simply files on the filesystem, introducing OpenTSDB and thus Hbase certainly adds complexity to the system. Deploying the system on a system limited in resources requires detailed fine-tuning, because failure of one component (say, Hbase running out of disk space) will cause the entire system to enter a failed state. Troubleshooting the issue is now a matter of going through log files of multiple applications/services and possibly restarting multiple services as well, in order to get back to a working graphing setup.

The additional flexibility in plotting we were after, does however justify these additional efforts. This iteration brought us dynamic dashboards and combined plots, which can be customized by the operator for troubleshooting ends, or a security officer for spotting anomalies or threats, in any way they see fit for their job at hand.

## Lessons learned & possible improvements

Concluding, we find that visualization works. Especially, when we do not know where to look for exactly, which can be the case with new IPv6 concepts, plotting for exploratory analysis quickly yields insight in what is going on on the network.

When working in such a way, the ability of creating new plots or combining multiple graphs in a single plot is key, as the analysis of the network data is often an iterative process as well: one pattern or relation leads to a more fine-grained query highlighting more specific network traffic, whether it’s security related or functional troubleshooting.

While IPFIX itself is flexible enough to allow fine-grained measurements (and thus visualization), we notice a certain degree of vendor-lockin because of extensions being vendor-specific. With the uprise of P4, we hope to be able to perform measurements in a more generically applicable way. With the created visualization setup, switching from IPFIX to P4 for input should be easy.