

28/02/06

Deliverable DJ2.2.2: User and Test Report on NetFLOW Collector



Deliverable DJ2.2.2

Contractual Date:	30/11/05
Actual Date:	28/02/06
Contract Number:	511082
Instrument type:	Integrated Infrastructure Initiative (I3)
Activity:	JRA2
Work Item:	2
Nature of Deliverable:	R (Report)
Dissemination Level	PU (Public)
Lead Partner	SURFnet
Document Code	GN2-06-004v3
Authors:	Ladislav Lhotka (CESNET)

Abstract

Flow-based monitoring using Cisco Netflow has been identified by GN2 Joint Research Activity 2 as the principal source of data for security analysis and intrusion detection. This deliverable reports on first user experiences and tests performed with the hardware-accelerated Netflow probe that is being developed as part of the JRA2 toolset. The deliverable is also intended as an aid for future users and testers of the probe.

Table of Contents

0	Executive Summary	iv
1	Introduction	1
	1.1 Terminology	2
2	The Netflow Probe	3
	2.1 Hardware	3
	2.2 Firmware	4
	2.3 Device Driver	6
	2.4 Flow Exporter Program	6
3	Probe installation and setup	8
	3.1 Host computer	8
	3.2 Network connection	8
	3.3 Software prerequisites	9
	3.4 Compiling the software	9
	3.5 Compiling the driver	9
	3.6 Loading kernel modules	10
	3.7 Booting the firmware and running the exporter	10
	3.8 Logging	11
4	Tests 13	
	4.1 Performance	13
	4.2 Netflow version 9 conformance tests	15
	4.3 Tests with production traffic	15
5	Conclusions	21
6	References	22
7	Acronyms	23

Project:	GN2
Deliverable Number:	DJ2.2.2
Date of Issue:	28/02/06
EC Contract No.:	511082
Document Code:	GN2-06-004v3

List of Figures

Figure 1. Netflow hardware accelerator: COMBO6 motherboard (bottom) and COMBO-4SFP interface card with optical SFP transceivers (top).	4
Figure 2. Firmware block diagram	5
Figure 3. Throughput of the Netflow probe as a function of packet size	14
Figure 4. Number of flows in the cache	16
Figure 5. Rates of flows exported by the firmware	17
Figure 6. Spectrum of transport layer protocols	18
Figure 7. Top sources and services	19
Figure 8. Cumulative volumes of most frequent services	20
Figure 9. BGP sessions	20
Figure 10. Structure of the Netflow v9 Template FlowSet used by the flow exporter	24

0 Executive Summary

Given the focus of the GN2 project and most of its participants on backbone networks, the Joint Research Activity 2 (JRA2, Security) decided to concentrate on flow-based monitoring as the single most important source of data for security analysis and intrusion detection. The current de facto standard in this area is the Cisco Netflow technology. This deliverable reports on first experiences and tests performed with the hardware-accelerated Netflow probe that is being developed within JRA2 Work Item 2 as part of the security analysis toolset. The probe currently supports export of flow records in the Netflow v9 format.

The deliverable first discusses general advantages of using a stand-alone probe over the standard method of Netflow data acquisition from IP routers. Following overall characterisation of the probe, the steps for connecting the probe to the network, initialising the firmware and running the flow exporter program are described in some detail. Finally the results of various tests performed by CESNET both with laboratory equipment and on live production network are shown.

It is planned to distribute the probe to other WI2 partners (this process has already started) so that it can be tested in diverse network environments and the exported data processed with multiple collector programs. In the future versions of this deliverable the results of these additional tests will be included.

Project:	GN2
Deliverable Number:	DJ2.2.2
Date of Issue:	28/02/06
EC Contract No.:	511082
Document Code:	GN2-06-004v3

1 Introduction

State-of-the-art methods for network security analysis and intrusion detection usually rely on combining and correlating data from various sources such as link utilisation, statistics about IP flows and detailed packet payload inspection. In the early phases of the JRA2 activity, such an integrated approach has also been discussed. However, given the focus of the GN2 project and most of its partners on backbone network operation, it was eventually realised that payload inspection on backbone links is beyond the reach of the existing technology:

- Software tools such as *Snort*¹ can handle, depending on the processor speed, traffic rates in the range of few hundred megabits per second. This is clearly insufficient for network links with capacities of 1 Gb/s and more.
- Methods for hardware-accelerated payload inspection and pattern matching using Field-Programmable Gate Arrays (FPGA) are a subject of an intensive research. However, neither commercial products nor functional prototypes are available yet.

As a result, JRA2 decided to abandon the field of payload inspection and concentrate instead on collection and analysis of data about IP flows using the de facto standard technology -- Cisco Netflow. Unlike payload inspection, flow-based monitoring is best deployed in backbone networks and methods for collecting and processing flow data from multi-gigabit links are relatively well understood.

So far, Netflow data are typically acquired and exported by IP routers. Such a setup has several drawbacks, especially for security-related monitoring:

1. Routers are by definition visible Layer 3 devices that can be easily discovered by simple tools such as *traceroute*. Consequently, they often become targets for all kinds of attacks and intrusions.
2. The main task of routers is to forward datagrams and exchange routing information with their neighbours. The processing power available to other tasks is thus rather limited and so operations on Netflow data do not usually go much beyond simple export of raw flow records.

¹ <http://www.snort.org>

3. As a special case of the previous item, some routers impose sampling on the input traffic. Even if sampling is not required, it is often the only way for keeping the router operational, especially on high speed interfaces.

This deliverable describes the hardware-accelerated Netflow probe that is being developed within JRA2 as a part of the Work Item 2 toolset for security monitoring. It is essentially a stealth device invisible at both Layer 3 and Layer 2. At the moment, it is only able to monitor Gigabit Ethernet links. Support for other link technologies (10GE, STM-16 PoS) are under development, although the throughput of the current hardware is limited to 1,6 Gb/s.

The contents of this deliverable are organised as follows: In the next section we will give an overview of the probe from the user perspective, without going into much technical detail. The hardware and firmware design of the probe is described in [ZPK05]. Section 3 contains step-by-step instructions for connecting the probe into the network and getting it up and running. Finally, in section 4 we describe the results of initial laboratory tests and first experiences with using the probe in a production network.

1.1 Terminology

The following terms are used throughout the report.

IP traffic flow	Subset of IP datagrams passing an observation point during a certain time period that share a specified set of common properties – key fields – taken either from IP and transport headers or from additional context (e.g., input interface, source autonomous system)
exporter	Device sending export packets that contain records about observed flows to one or more collectors. It could be an IP router or a specialised probe.
collector	Computer running a program that receives flow records from one or more exporter(s). It may either simply store the records in a database or provide additional functions such as further processing or viewing the data.
sampling	Process in which a subset of the incoming packets is selected – only these packets are then taken into account in the flow records. The simplest case is <i>statistical sampling</i> : Each incoming packet is selected with probability p_s . It is often described in terms of sampling rate R , which is the reciprocal value of the sampling probability, i.e., $R = 1/p_s$.
active timeout	Time period after which data about an ongoing flow are exported. This way, information about long-term flows is not lost in case of exporter failure.
inactive timeout	If no packets for a given flow are observed for this time period, the flow is terminated and the corresponding flow record exported. Note that termination of flows can sometimes be based on protocol-related events such as presence of FIN or RST flags in TCP segments.

2 The Netflow Probe

The Netflow probe is a standard PC running Linux that is equipped with an add-on hardware card for accelerated flow processing. In a typical case, the host PC will also have a separate network interface card used for sending Netflow records to a collector over the network and also for remote configuration and management. However, it is also possible to avoid any standard network connection altogether and run the collector and presentation backend directly on the PC hosting the probe.

2.1 Hardware

The Netflow hardware accelerator is based on the COMBO family of programmable hardware cards that were originally developed for the 6NET project² and later also reused by the SCAMPI project³. It is a sandwich of two interconnected cards that fits into a single PCI slot (see Figure 1):

² FP5-IST-2001-32603, <http://www.6net.org>

³ F55-IST-2001-32404, <http://www.ist-scampi.org>

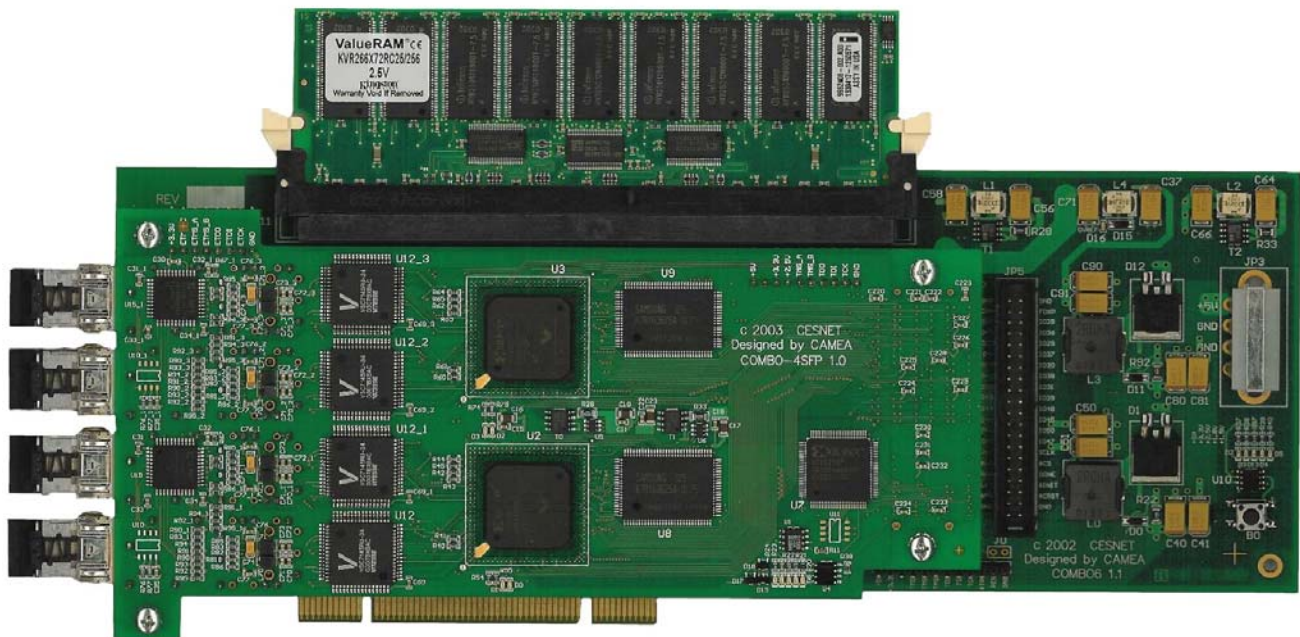


Figure 1. Netflow hardware accelerator: COMBO6 motherboard (bottom) and COMBO-4SFP interface card with optical SFP transceivers (top).

- COMBO6 motherboard houses most of the processing power and connects the accelerator to the PCI bus.
- Through the ports on the *interface card* the accelerator is connected to the network. Two Gigabit Ethernet interface cards are currently available: (i) COMBO-4MTX with four metallic ports, and (ii) COMBO-4SFP with four cages for standard SFP transceivers.

2.2 Firmware

Firmware for the Netflow hardware accelerator is an entirely new design written in the VHDL language. It is able to process both IPv4 and IPv6 traffic simultaneously. From the viewpoint of the monitored link, the probe acts as a repeater: The ingress traffic received on one port of a COMBO interface card is immediately transmitted to another port while a copy of the traffic is passed to the firmware for flow processing. The firmware is able to maintain records about up to $2^{16} = 65536$ flows at the same time.

Project:	GN2
Deliverable Number:	DJ2.2.2
Date of Issue:	28/02/06
EC Contract No.:	511082
Document Code:	GN2-06-004v3

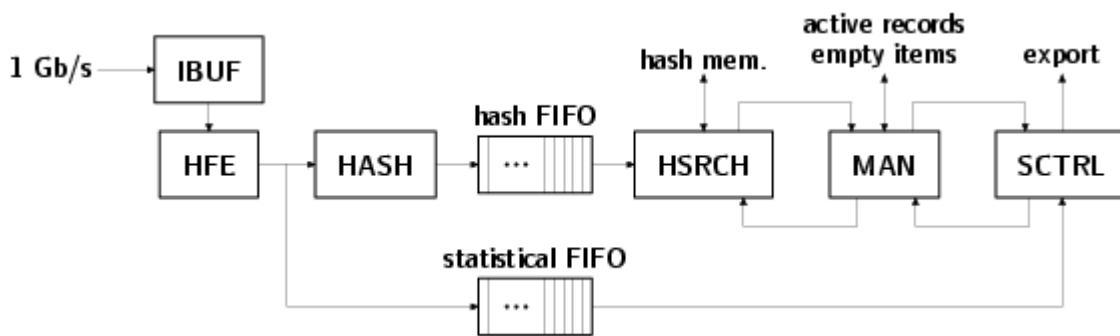


Figure 2. Firmware block diagram

The processing pipeline is depicted in Figure 2 and works as follows:

Packets received by the Input Buffer (IBUF) are passed to the Header Field Extractor (HFE). This unit parses L2, L3 and L4 headers, extracts all relevant fields and records them in a fixed data structure named Unified Header (UH) that is stored in RAM-based queue (Statistical FIFO). In parallel, certain *key fields* from the UH are used as input to the CRC-64 hash function implemented by the HASH unit. From the 64-bit result of the CRC-64 function, we use 57 bits as the unique identifier of the flow. By default the key field are:

- source and destination IP addresses
- source and destination port numbers
- L3 protocol number

The firmware can be configured to out an arbitrary selection of bits from these key fields so that only the remaining bits are used for computing the hash function. The flows would be more aggregated in this case.

The use of a 57-bit hash value as the flow identifier means that two packets with differing key fields may occasionally map to the same hash value so that the packets are incorrectly classified as belonging to the same flow. However, given the statistically uniform distribution of CRC-64 values, the probability of an *undetected* collision of flows is $N \cdot 2^{-57}$ where N is the actual number of flows in the cache. As the maximum number of flows in the cache is $N_{\max} = 2^{16}$, the collision probability cannot be higher than $2^{-41} \approx 4,55 \cdot 10^{-13}$. With the current maximum throughput of half million packet per second it means 7 collisions per year on the average. While this probability is sufficiently low for most purposes, we have also taken into account the possibility of an attacker generating a hash collision on purpose, e.g., by injecting carefully forged traffic first and then launching another (hostile) traffic that will be classified as the previous forged flow due to the hash collision. This scenario is not extremely difficult to realise given the fact that the hash function is known. To oppose this threat, the HASH unit is initialised with a random seed so that the values of the hash function are not predictable.

The hash value is stored in the Hash FIFO. On the opposite end of this queue, the Hash Search Unit (HSRCH) takes the hash values one-by-one and searches the Hash Memory in order to find out whether the hash value is already present. If it is the case, the SCTRL unit is instructed to update the statistics of the existing flow. Otherwise, new entries in both the Hash Memory and the SCTRL unit are created.

The Manager unit (MAN) looks after all flows entries in the Hash Memory and also manages the list of free memory locations. The flow entries are kept in a bidirectional list sorted by the timestamp of the flow entries. This way, inactive flows are easily recognised when their age exceeds a given threshold (inactive timeout).

Finally, the Storage unit (SCTRL) collects the statistics about active flows and exports flow records according to instructions obtained from the MAN and HSRCH units.

The current firmware version also includes a special experimental sampling procedure: *sample-and-hold* [EV02]. Its operation is similar to the standard statistical sampling with the following significant difference: sampling is always avoided for flows that already have entries in the flow cache. This way, one can get a very precise information about large flows. The plain statistical sampling scheme is not yet supported but is planned for the next version.

Several parameters governing firmware behaviour can be modified on the fly, without disrupting its operation:

- active timeout in the range 0-1200 seconds
- inactive timeout in the range 0-60 seconds
- sampling rate for sample-and-hold between 1 and 65535
- threshold for sample-and-hold – sampling is not started unless the number of entries in the flow cache is higher than this value.

2.3 Device Driver

Device driver for the Netflow probe is available for Linux 2.4 and 2.6 kernels. It is quite sophisticated in that it allows concurrent access of multiple applications to the flow records. A single shared memory block is used for storing up to 16384 flow records in a logical structure of a circular buffer. When the buffer becomes full, the oldest records are rewritten by the new ones. Each application keeps its own pointer to the buffer and can also lock up to 1024 records in order to prevent these records from rewriting before the application is finished with reading them.

Applications are allowed to access the driver only through a special low-level library, *libcsflow*. This library implements certain common functions and, for debugging purposes, it also enables application testing without access to the Netflow probe itself – the flow records can be supplied from a disk file.

2.4 Flow Exporter Program

First application to use the Netflow probe is the flow exporter program for Netflow version 9 [RFC3954]. It is currently able to send export packets to a single collector via IPv4/UDP transport. The collector IP address and destination UDP port can be configured through command line parameters.

Project:	GN2
Deliverable Number:	DJ2.2.2
Date of Issue:	28/02/06
EC Contract No.:	511082
Document Code:	GN2-06-004v3

Six Netflow v9 templates are available for all combinations of IPv4 or IPv6 and the most common L3 protocols TCP, UDP and ICMP. Two additional templates (IPv4/OTHER and IPv6/OTHER) are used for all other L3 protocols. All templates share the following seven fields:

- FIRST_SWITCHED – timestamp of the first packet in a flow
- LAST_SWITCHED – timestamp of the last packet in a flow
- OUT_PKTS – number of packets associated with a flow
- OUT_BYTES – number of octets associated with a flow
- IP_SRC_ADDR or IPV6_SRC_ADDR – source IP address
- IP_DST_ADDR or IPV6_DST_ADDR – destination IP address
- PROT – Layer 3 protocol number

Additional fields present in protocol-specific templates are shown in Table 1.

Field	TCP	UDP	ICMP	Description
L4_SRC_PORT	•	•		source port number
L4_DST_PORT	•	•		destination port number
DST_TOS	•	•		type of service octet
TCP_FLAGS	•			TCP flags
ICMP_FLAGS			•	ICMP flags

Table 1: Fields in protocol-specific templates

The templates contain vital information for the collectors to be able to interpret the data flowsets. It is thus important to resend the template descriptions every once in a while. The period for sending template flowsets can be configured through a command line parameter.

Netflow v9 options are not supported yet.

3 Probe installation and setup

The distribution package containing the firmware and software necessary for running the Netflow probe is currently available to JRA2 partners and other interested parties upon registration⁴. Although the probe is still in the stage of a prototype, even a non-expert should be able to install and operate it. The following sections contain simple step-by-step instructions. Additional details can be found in README files that are part of the distribution package.

3.1 Host computer

The probe can be hosted essentially by any stock PC. In order to guarantee sufficient heat dissipation, a full-size case with good fans is recommended. The sandwich consisting of the COMBO6 card and an interface card may be plugged into any PCI slot.

In most cases, another (standard) network interface card will be needed for sending the export packets to a remote collector and also to be able to establish a remote connection to the host computer in order to configure and manage the probe.

3.2 Network connection

For monitoring a single link, the cables should be connected as follows:

- The ingress traffic goes to port 0 – in Figure 1 it is the bottommost port adjacent to the PCI connector.
- Port 1 retransmits the data received at port 0.

Only Gigabit Ethernet is supported at the moment – in particular, plain Ethernet or Fast Ethernet will not work.

The repeater functionality of the probe is still experimental, so it is probably not wise to insert the probe into production lines with high reliability requirements. For the initial JRA2 tests, we recommend to connect port 0 to a switch/router port receiving mirrored traffic from other interface(s). Port 1 can then be connected, for example, to a network tester in order to have an independent record of the traffic processed by the probe.

⁴ URL: <http://www.liberouter.org/clients/>

3.3 Software prerequisites

All software for the probe (driver, control utilities and the flow exporter) is currently supported only for Linux. So far most tests have been performed using the Debian Linux but any reasonably recent distribution should be fine as well. The software is distributed as a source package and must be compiled on the host computer. To be able to do this, the C development environment must be installed. The GNU C Compiler of major version 2 or 3 is recommended. In addition, *autoconf* and *automake* programs are also required. Most Linux distributions provide them as add-on packages.

The device driver has been tested with recent Linux kernels from the 2.4 and 2.6 series. It is important though to compile the driver module with exactly the same GCC version as the kernel proper.

3.4 Compiling the software

The name of the distribution package is of the form `NETFLOW_XX_YY.tgz` where `XX` and `YY` are major and minor versions of the package, respectively. In general, the most recent version should always be used as older versions may not have all features described in the documentation and/or contain bugs.

The distribution tarball can be unpacked in any convenient location. In order to compile the flow exporter program and control utilities, go to the `NETFLOW_XX_YY/base` directory (replacing `XX_YY` with the actual version number) and run the setup script:

```
$ ./setup
```

After the script is done, run

```
$ make
```

This will compile all necessary programs and utilities. It is not necessary to install them into the system directories such as `/usr/local/bin` – the probe and flow exporter program can be started by running shell scripts that use the programs directly from the build tree as described in the following subsections.

3.5 Compiling the driver

The device driver has to be compiled separately. To do that, perform the following steps:

```
$ cd sys_sw/drivers/linux  
$ ./cvscompile
```

Again, it is not necessary to install the compiled kernel modules into the system directories.

Project:	GN2
Deliverable Number:	DJ2.2.2
Date of Issue:	28/02/06
EC Contract No.:	511082
Document Code:	GN2-06-004v3

Finally, the driver uses two device files, `/dev/combosix/0` and `/dev/scampi/0` that can be created by running the following script as root:

```
# ./flowdevices
```

The permissions of the device files must be arranged so that the non-privileged user that will execute the `netflow_ph1` script below has permissions to read and write both files. For example, one can create a new group, say `combo-rw`, add the user in question to this group and change the ownership and permissions of the device files as follows:

```
# chgrp combo-rw /dev/combosix/0 /dev/scampi/0  
# chmod 664 /dev/combosix/0 /dev/scampi/0
```

3.6 Loading kernel modules

As the first step in the process of activating the probe, the kernel modules have to be loaded under the superuser privileges:

```
# cd NETFLOW_XX_YY/base/vhdl_design/projects/netflow_ph1/test  
# ./netflow_ph1_modules -l 0
```

The command line option “-l 0” (ell zero) results in taking the kernel modules directly from the build tree instead of the `/lib/modules` tree.

3.7 Booting the firmware and running the exporter

All the remaining steps – namely downloading and booting the firmware and starting the flow exporter program – are now handled by the `netflow_ph1` shell script that is located in the same directory as the previous `netflow_ph1_modules` script. This script should be executed by a non-privileged user that has read/write access to the device files as discussed above:

```
$ ./netflow_ph1 -c <collector_name_or_ip>:<collector_port>
```

Specifically, this script downloads and boots the firmware and starts the flow exporter program.

As the value of the `-c` option, supply the DNS name or IPv4 address of a running Netflow v9 collector and the UDP port where it is listening.

The probe should now start analyzing the traffic arriving at port 0 and the flow exporter sending Netflow v9 packets to the configured collector. As a basic test, one can use a packet sniffer such as *tcpdump*⁵ or *Ethereal*⁶. For the former, the following command can be used to see just the Netflow export packets:

```
# tcpdump 'udp dst port <collector_port>'
```

Apart from the `-c` option, the `netflow_ph1` script accepts few other options that can be used to control the behaviour of the firmware and software (change active and inactive timeouts etc.). To see all available options, run the script with the `-h` switch.

In all subsequent invocations of the `netflow_ph1` script, i.e., when restarting the flow exporter or even after rebooting the host computer, you will probably want to skip the card booting step by including the `-r` switch. This way, the losses in flow records are minimized. Under normal circumstances, the firmware must be booted only after the host computer has been switched off.

3.8 Logging

The flow exporter program logs important events to syslog using the *user* facility. To have these log messages saved to a file such as `/var/log/flowexp.log`, you must configure the syslog daemon appropriately. In the case of the classical Unix `syslogd` daemon, add the following line to the `/etc/syslog.conf` configuration file:

```
user.*                /var/log/flowexp.log
```

The probe firmware is also able to report its status periodically to a log file. To get this information, run the `NETFLOW_XX_YY/base/vhdl_design/projects/netflow_ph1/test/netflow_ph1_log` script. It accepts the following command line options:

- `-l <filename>` Specifies the name of the log file. The default log file is `netflow.log` in the current directory.
- `-p <period>` Specifies reporting period in seconds. The default is 60 seconds.
- `-h` Prints a help message and exits.

Every periodical report appends one line to the log file in the following format:

```
YYYY-MM-DD HH:MM:SS, <cache_size>, <total_flows>
```

The two numbers following the timestamp are

⁵ <http://www.tcpdump.org>

⁶ <http://www.ethereal.com>

<cache_size> Current number of flows in the flow cache.
<total_flows> Accumulated number of flows that have been exported by the firmware. It is a 32-bit number so that $2^{32} = 4294967296$ wraps back to zero.

4 Tests

We performed tests of the Netflow probe covering the following three operational aspects:

- Tests of *raw performance* yielded estimated throughput of the device for different packet sizes.
- In *conformance* tests we used a representative set of packets with varying properties (L2 and L3 protocols, ICMP types, TCP flags etc.) and checked whether these packets were handled correctly by the device.
- In *quasi-production* tests we exposed the probe to a relatively massive stream of real network data and analysed the exported data using standard collector and visualisation software.

4.1 Performance

The aim of performance tests was to estimate the throughput of the Netflow probe for Gigabit Ethernet frame sizes ranging from 66 to 1500 bytes. For this purpose, throughput is defined as the highest frame rate processed by the device without any packet loss. A series of tests was run, each consisting of 511 different IPv4/UDP flows with packets of varying size. The flows were generated by the Spirent AX/4000 network analyser. In each test session we were gradually increasing the frame rate and checked whether the Netflow engine reported exactly the number of packets that were sent to the probe. The exact point where the probe starts losing packets was then determined by bisecting the borderline interval of frame rates.

The results of these tests are given in Table 2.

Frame size	Datagram size	Packet count	Packets/sec	% Bandwidth
66	48	1000000	494962,65	26,53
128	110	1000000	381640,57	39,08
256	238	1000000	256396,54	52,51
512	494	1000000	154785,21	63,40
1024	1006	1000000	86047,32	70,49
1500	1482	1000000	61191,67	73,43

Table 2: Throughput of the Netflow probe for different packet sizes

The plot in Figure 3 shows the throughput in Mb/s for different packet sizes.

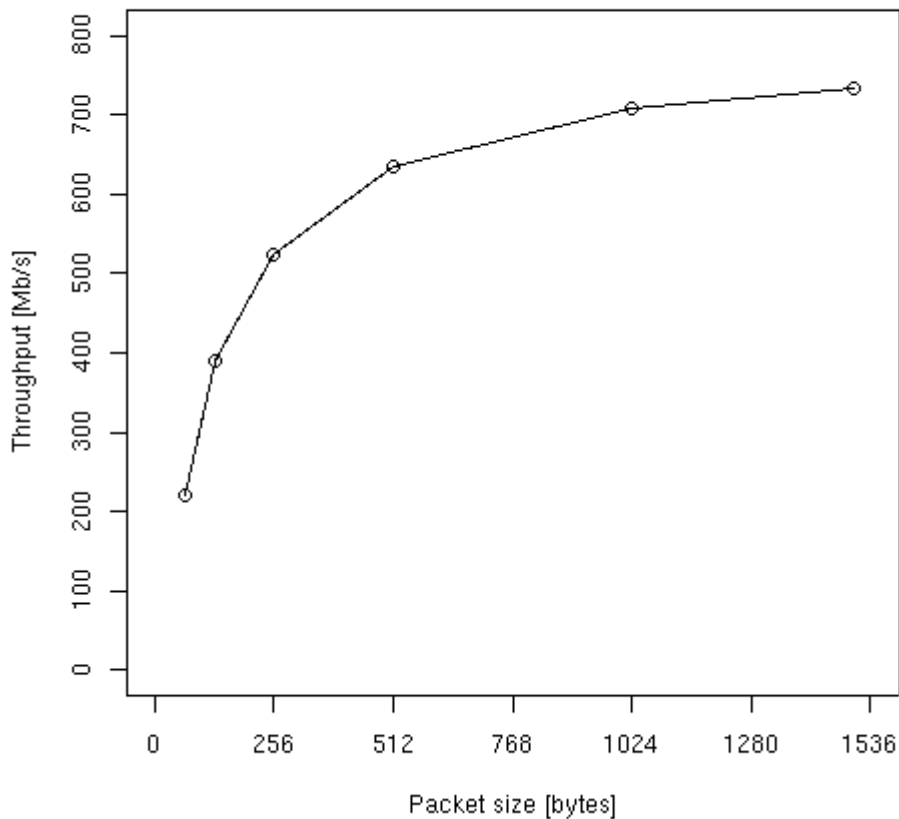


Figure 3. Throughput of the Netflow probe as a function of packet size

From these results it is apparent that the Netflow engine has two independent bottlenecks:

- The firmware currently runs at clock rate 50 Mhz, which means that with 16-bit wide data path the theoretical throughput limit is 800 Mb/s. This bottleneck limits the throughput of big packets at the upper-right end of the curve in Figure 3.
- Header Field Extractor cannot process more than approximately 500 thousand packets per second. This affects especially the throughput of small packets at the lower-left end of the curve in Figure 3.

Both bottlenecks will be addressed in the future versions of hardware and firmware:

1. With the new COMBO6X motherboard the clock rate will be increased to 100 MHz meaning theoretical throughput of 1,6 Gb/s.

2. A new implementation of HFE is now being tested that was designed to process at least 2 million packets per second.

After these improvements are implemented, we will be able to monitor 1 Gb/s circuits at line rate for any mix of packet sizes.

4.2 Netflow version 9 conformance tests

In this test we used the Spirent AX/4000 network analyser to generate single-packet flows with known characteristics for all eight combinations of the supported L2 (IPv4 and IPv6) and L3 (TCP, UDP, ICMP and OTHER) protocols and captured the export packets sent by the flow exporter program. We used Ethereal for dissecting the export packets. However, it turned out the Netflow v9 support in Ethereal version 0.10.13 is not complete in that it is only able to recognise the first template in a Template FlowSet containing multiple template specification, even though such Template FlowSets are perfectly legal, see [RFC3954], section 5.2. Moreover, Ethereal is not able to decode some fields in Data FlowSets such as OUT_BYTES or OUT_PKTS. Because of these bugs, we had to complement Ethereal by inspecting directly hexadecimal dumps of the export packets.

The flow exporter program periodically sends all eight templates in a single Template FlowSet. The period is configurable and the default behaviour is to send the templates after every 10 export packets with Data FlowSets. We verified that the Template FlowSet as well as the individual templates are correct. Appendix A contains the details about the structure of the Template FlowSet. One thing that should be improved in the future version of the flow exporter program is the inconsistent layout of the templates. For example, all templates except one have the IP addresses as the last two fields, whereas the IPv6/ICMP template has them as the first two fields. Strictly speaking, this is not a problem and any Netflow-v9-compliant collector should be able to handle them. Nonetheless, we think such a complication is unnecessary and should be avoided.

Similarly, we also verified that the Data FlowSets corresponding to all test flows were well-formed and contained correct data.

4.3 Tests with production traffic

The Netflow probe has been tested for almost two months in a configuration that was essentially identical to a production deployment. The probe was connected to a router port that is receiving network traffic mirrored from one of the busiest CESNET2 backbone lines – access line of the Brno campus. Generated Netflow v9 data are processed by the FTAS software [Koš04] consisting of a collector and visualisation software.

We monitored the performance of the probe and recorded the values of two parameters that are critical for its operation, namely the number of flows in the cache and the rate of flow records exported from the hardware. Their values are read from the corresponding firmware registers in one-minute intervals. Figure 4 and Figure 5 show typical 3-hour time series of these parameter taken during work hours.

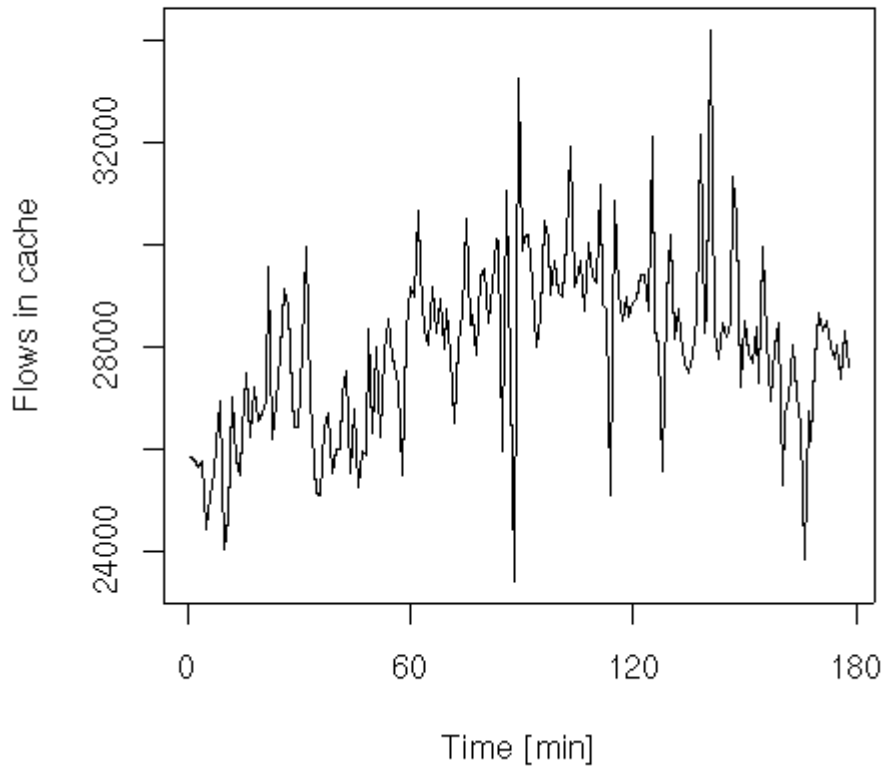


Figure 4. Number of flows in the cache

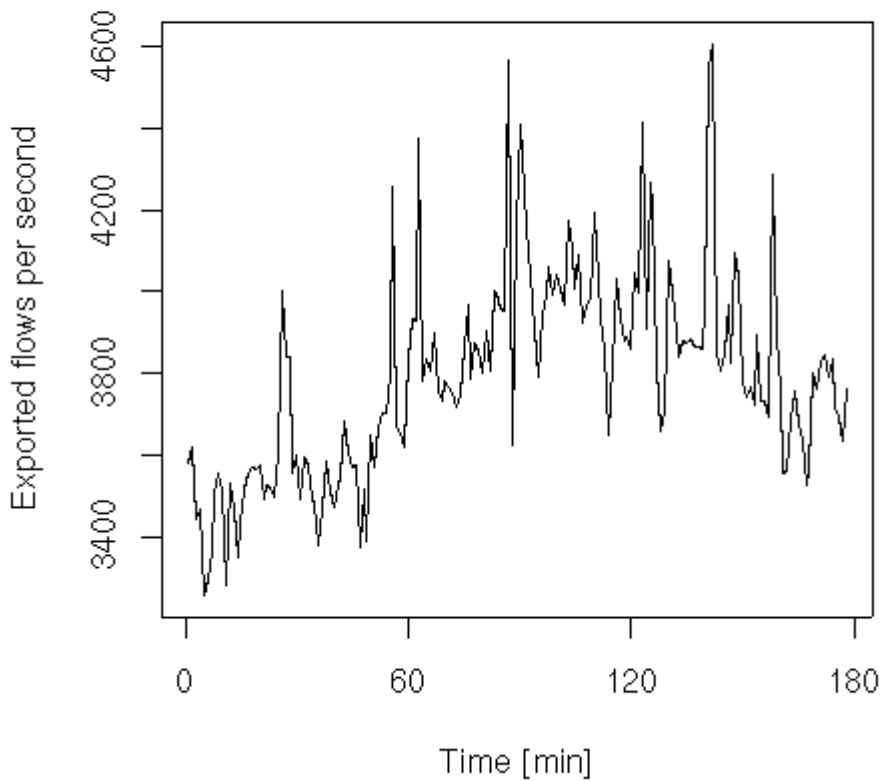


Figure 5. Rates of flows exported by the firmware

The arithmetic mean of the cache occupancy in Figure 4 is 27994 while the geometric mean of the export rates in Figure 5 is 3797 flows per second. From the graphs it is clear that during the observed period the cache was always far from filling up and the flows were mainly exported due to active timeout expirations.

We used the selection and query possibilities of the FTAS software and analysed the available data in many different ways. The following figures show typical results of such an analysis. All examples use the same ten-minute traffic frame. Our primary goal here was to verify that the observed qualitative patterns do not deviate from the expected outcomes and thus get more confidence that the Netflow probe operates properly and reliably in real-life circumstances.

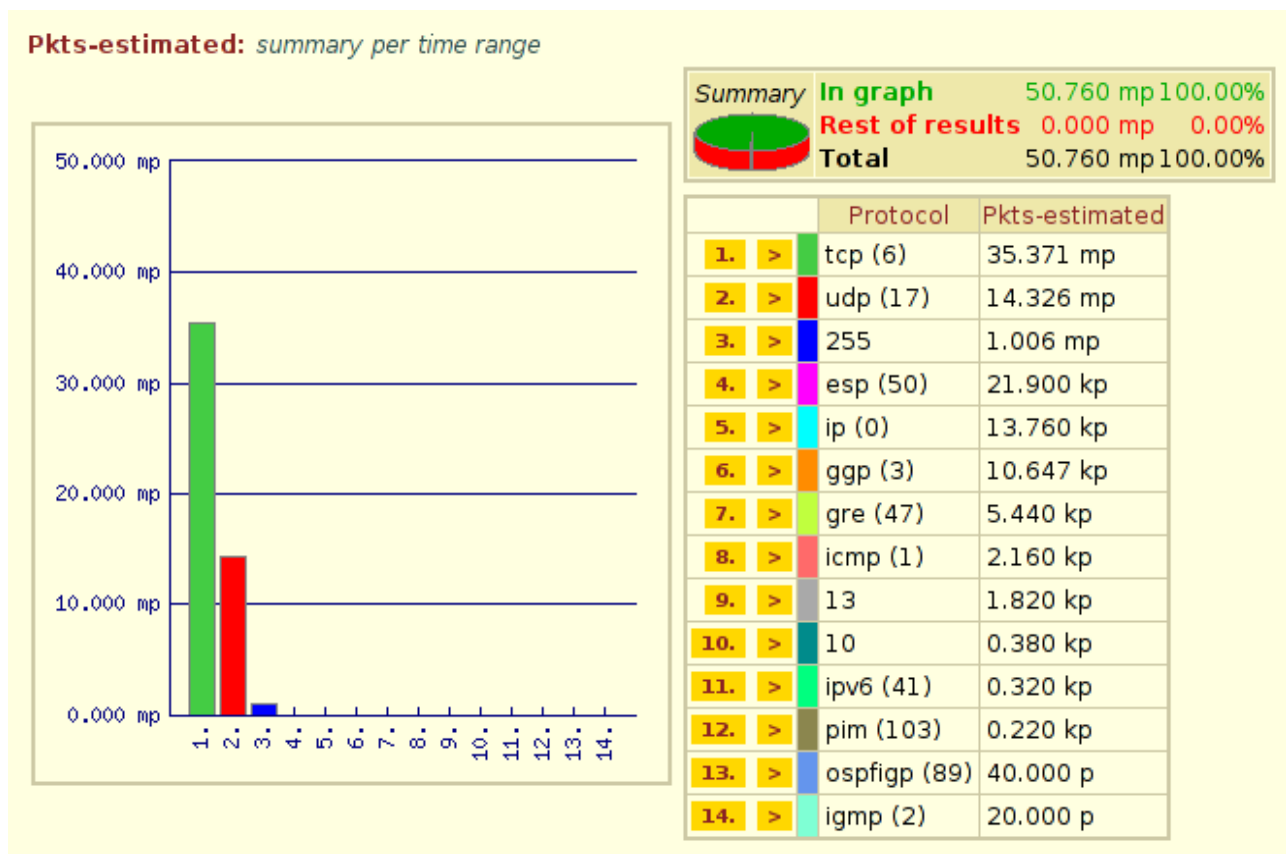


Figure 6. Spectrum of transport layer protocols

In Figure 6 we see traffic shares of the most significant transport layer protocols in terms of estimated packet counts⁷. Apart from TCP and UDP, the relatively massive traffic with protocol number 255 probably indicates the presence of compromised host(s).

⁷ The visualisation software tries to estimate the actual packet counts by multiplying observed packet counts with known sampling rates.

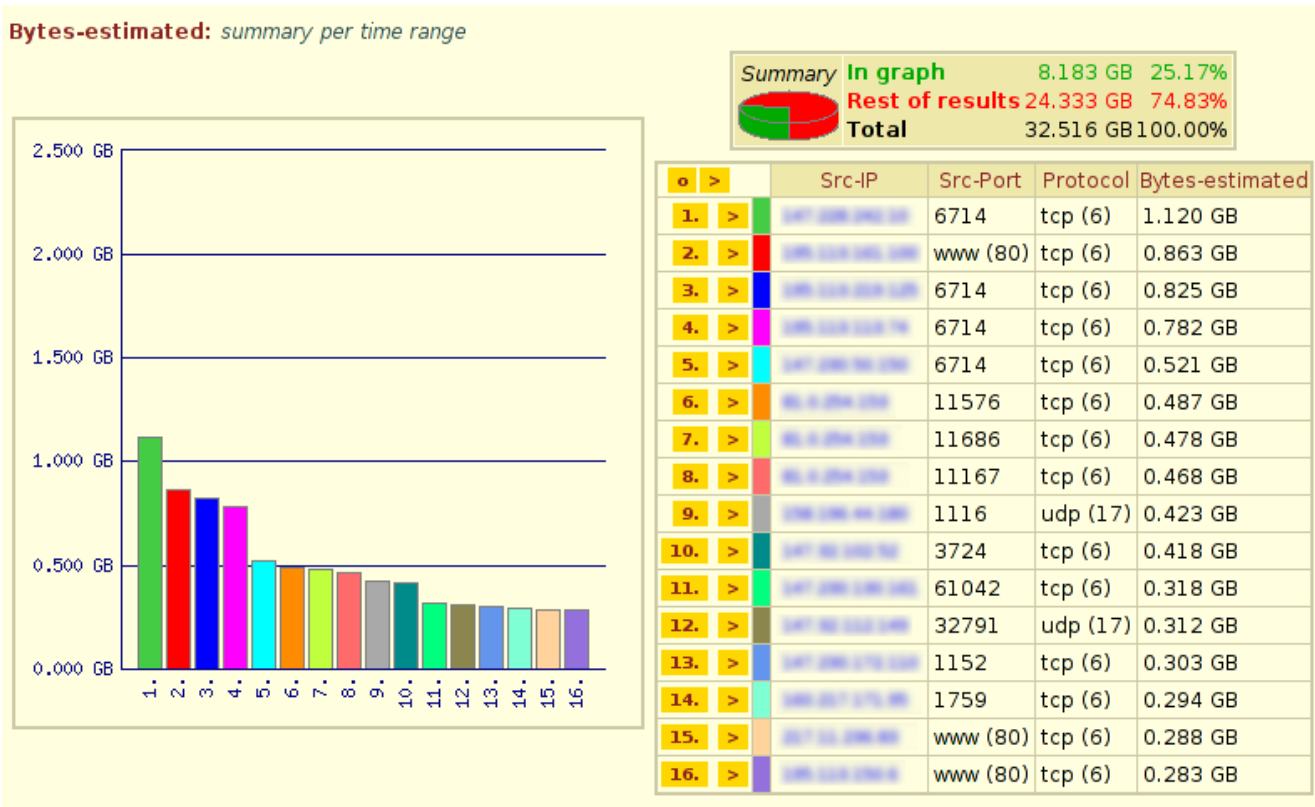


Figure 7. Top sources and services

Another useful insight that can be obtained from flow data is a list of most important data sources and services as shown in Figure 7. For privacy protection, all IP addresses were blurred in the table. Interestingly enough, the standard WWW service (port 80) is rather rare among the top 16 sources. Note, however, that almost three quarters of the traffic is not represented in this view – see the pie chart at the top right.

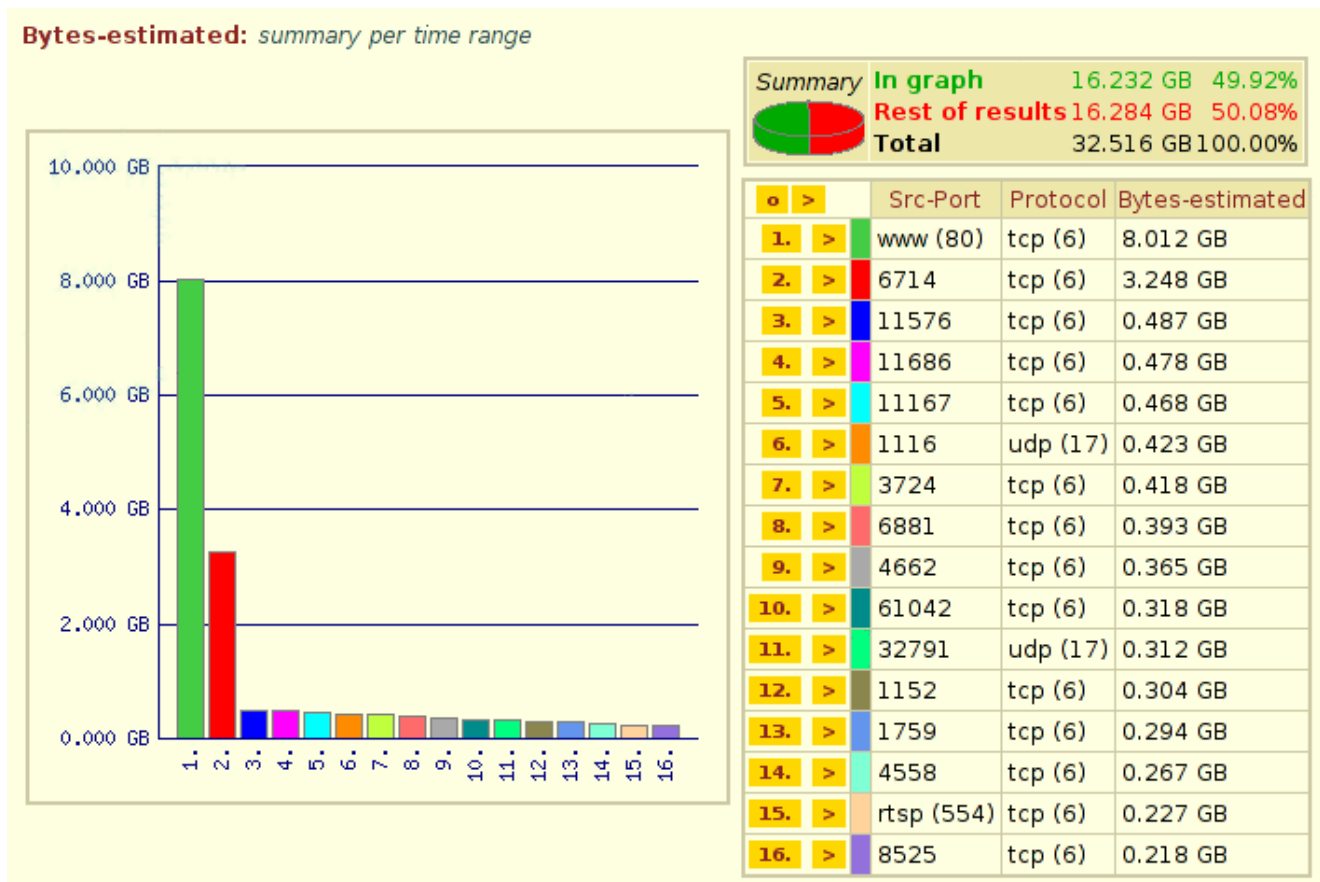


Figure 8. Cumulative volumes of most frequent services

In terms of cumulative volumes of the services, the WWW service still keeps the largest traffic share, as demonstrated in Figure 8.

Finally, Figure 9 shows a table of all BGP connections that are established over the monitored link. Note that one of them (line 2) is an IPv6 peering.

Results

	Src-IP	Dst-IP	Pkts-estimated	Bytes-estimated	Src-Port	Dst-Port	Protocol
1.	193.50.135.179	193.50.135.179	40.000 p	3.418 KB	16174	bgp (179)	tcp (6)
2.	2001:193:50:135::179	2001:193:50:135::179	20.000 p	1.895 KB	19431	bgp (179)	tcp (6)
3.	193.50.135.179	193.50.135.179	20.000 p	1.523 KB	11027	bgp (179)	tcp (6)
4.	193.50.135.179	193.50.135.179	20.000 p	1.523 KB	16174	bgp (179)	tcp (6)

Figure 9. BGP sessions

5 Conclusions

The tests presented in this report indicate that the existing Netflow probe prototype satisfies the requirements for Phase 1 specification with one exception: the throughput does not yet reach the projected 1 Gb/s line rate. This performance issue should be resolved soon though by using the new COMBO6X card that will allow to increase the clock rate to 100 MHz and also by improving the design of the Header Field Extractor.

The main weakness of the tests is the fact that they were performed all at the same site, essentially by the same team that developed the probe. Further independent tests are thus being prepared. One prototype of the probe has already been installed by SURFnet and additional ones will be distributed to other partners as soon as new COMBO cards are delivered from the factory and activated. Another obstacle to effective testing is the fact that most Netflow collectors either lack support for Netflow version 9 outright or, as it turned out, contain serious bugs in the modules that do version 9 data processing. To alleviate this problem, we decided to implement the most common version 5 in the next release of the flow exporter program.

Security-related applications of Netflow tend to avoid sampling in order to keep track of every single packet. Therefore, sampling was not among the required features in the JRA2 specification of the probe. Nevertheless, it turns out that the probe can be useful for JRA1 (QoS monitoring) and possibly other purposes. Statistical sampling will be available in the next release of the firmware (scheduled for January 2006). With this addition, the functionality of the Netflow probe will be fully comparable to the existing Netflow engines. We will then concentrate on further improvements on the performance side and also on novel techniques such as adaptive sampling.

6 References

- [EV02]** Estan, C. and Varghese, G. New directions in traffic measurement and accounting. In: *Proceedings of the 2001 ACM SIGCOMM Internet Measurement Workshop*, San Francisco, California, 2001, pp. 75-80.
- [Koš04]** Košnar, T. Flow-Based Traffic Analysis System – Architecture Overview. Technical Report 15/2004, CESNET, Praha, 2004. URL: <http://www.cesnet.cz/doc/techzpravy/2004/ftas-arch/>
- [RFC3954]** Claise, B. (Ed.). *Cisco Systems NetFlow Services Export Version 9*. RFC 3954, IETF, October 2004. URL: <http://www.ietf.org/rfc/rfc3954.txt>
- [ZPK05]** Žádník, M., Pečenka, T. and Kořenek, J. NetFlow probe intended for high-speed networks. In: Rissa, T., Wilton, S. and Leong, P. (Eds.), *Proceedings of the 15th International Conference on Field-Programmable Logic and Applications (FPL05)*, Tampere, Finland. IEEE CS, 2005, pp. 695-698.

7 Acronyms

CRC	Cyclic Redundancy Check
FIFO	First-In-First-Out
FPGA	Field-Programmable Gate Array
FTAS	Flow-based Traffic Analysis System
GCC	GNU C Compiler
GE	Gigabit Ethernet
GNU	GNU is Not Unix
HFE	Header Field Extractor
HSRCH	Hash Search Unit
IBUF	Input Buffer
ICMP	Internet Control Message Protocol
JRA	Joint Research Activity
MAN	Manager Unit
POS	Packet over SDH
RAM	Random Access Memory
SCTRL	Storage Control Unit
SDH	Synchronous Digital Hierarchy
TCP	Transmission Control Protocol
TOS	Type of Service
UDP	User Datagram Protocol
UH	Unified Header
WI	Work Item
10GE	Ten-Gigabit Ethernet

Appendix A Netflow v9 Templates

This appendix shows the structure of all Netflow v9 templates that are used by the flow exporter program. All eight templates in use are periodically sent inside a single Template FlowSet [RFC3954]. The period can be configured via a command-line parameter to the *flowexporter* program or by changing the `NUM_TEMPLATES` variable inside the *netflow_ph1* script, see section 3.7.

The general structure of the Template FlowSet used by the flow exporter program is outlined in Figure 10.

15	31
FlowSet ID = 0	Length
Template ID = 257	Field Count = 11
Field Type 1	Field Length 1
Field Type 2	Field Length 2
...	
Field Type 11	Field Length 11
Template ID = 258	Field Count = 10
Field Type 1	Field Length 1
Field Type 2	Field Length 2
...	
Field Type 10	Field Length 10
...	
Template ID = 264	Field Count = 7
...	

Figure 10. Structure of the Netflow v9 Template FlowSet used by the flow exporter

The following sections contain tables describing the layout of all the templates, i.e., the blocks that start with "Template ID" in Figure 10. Details about individual fields can be found in [RFC3954].

A.1 IPv4/TCP

- Template ID = 257
- Field Count = 11

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
DST_TOS (55)	1
LAST_SWITCHED (21)	4
L4_DST_PORT (11)	2
L4_SRC_PORT (7)	2
TCP_FLAGS (6)	1
PROT (4)	1
IP_SRC_ADDR (8)	4
IP_DST_ADDR (12)	4

A.2 IPv4/UDP

- Template ID = 258
- Field Count = 10

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
DST_TOS (55)	1
LAST_SWITCHED (21)	4
L4_DST_PORT (11)	2
L4_SRC_PORT (7)	2
PROT (4)	1
IP_SRC_ADDR (8)	4
IP_DST_ADDR (12)	4

A.3 IPv4/ICMP

- Template ID = 259
- Field Count = 8

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
LAST_SWITCHED (21)	4
ICMP_TYPE (32)	2
PROT (4)	1
IP_SRC_ADDR (8)	4

Field Type (Decimal Code)	Field Length
IP_DST_ADDR (12)	4

A.4 IPv4/OTHER

- Template ID = 260
- Field Count = 7

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
LAST_SWITCHED (21)	4
PROT (4)	1
IP_SRC_ADDR (8)	4
IP_DST_ADDR (12)	4

A.5 IPv6/TCP

- Template ID = 261
- Field Count = 11

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
DST_TOS (55)	1
LAST_SWITCHED (21)	4
L4_DST_PORT (11)	2
L4_SRC_PORT (7)	2
TCP_FLAGS (6)	1
PROT (4)	1
IPV6_DST_ADDR (28)	16
IPV6_SRC_ADDR (27)	16

A.6 IPv6/UDP

- Template ID = 262
- Field Count = 10

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
DST_TOS (55)	1
LAST_SWITCHED (21)	4
L4_DST_PORT (11)	2
L4_SRC_PORT (7)	2
PROT (4)	1
IPV6_DST_ADDR (28)	16
IPV6_SRC_ADDR (27)	16

A.7 IPv6/ICMP

- Template ID = 263
- Field Count = 8

Field Type (Decimal Code)	Field Length
IPV6_SRC_ADDR (27)	16
IPV6_DST_ADDR (28)	16
OUT_PKTS (24)	5
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5

Field Type (Decimal Code)	Field Length
LAST_SWITCHED (21)	4
PROT (4)	1
ICMP_TYPE (32)	2

A.8 IPv6/OTHER

- Template ID = 264
- Field Count = 7

Field Type (Decimal Code)	Field Length
OUT_PKTS (24)	4
FIRST_SWITCHED (22)	4
OUT_BYTES (23)	5
LAST_SWITCHED (21)	4
PROT (4)	1
IPV6_DST_ADDR (28)	16
IPV6_SRC_ADDR (27)	16