



# nProbe User's Guide

Open Source Software and Hardware NetFlow v5/v9 Probe

Version 6.16  
April 2014

© 2002-14

# 1. Table of Contents

1. Introduction .....	3
1.1. Main Features .....	4
1.2. What's New? .....	4
2. Using nProbe .....	6
2.1. Compiling nProbe Source Code .....	6
2.2. Installing a Binary nProbe .....	6
2.3. nProbe Command Line Options .....	7
2.4. nProbe on Windows .....	18
2.5. Licenses Installation .....	19
2.6. Tuning nProbe Performance .....	19
2.7. Using nProbe with ntopng .....	20
2.8. Frequently Asked Questions .....	20
3. nProbe Plugins .....	22
3.1. BGP Plugin .....	22
3.2. DNS Plugin .....	23
3.3. GTPv0 Plugin .....	23
3.4. GTPv1 Plugin .....	24
3.5. GTPv2 Plugin .....	24
3.6. HTTP Plugin .....	25
3.7. IMAP Plugin .....	25
3.8. MySQL Plugin .....	26
3.9. Oracle Plugin .....	26
3.10. POP3 Plugin .....	27
3.11. Radius Plugin .....	27
3.12. RTP Plugin .....	27
3.13. SIP Plugin .....	28
3.14. SMTP Plugin .....	28
3.15. NetFlow-Lite Plugin .....	28
4. Developing nProbe Plugins .....	29
5. References .....	34
5.1. Credits .....	34
6. Appendix A: BPF Packet Filtering Expressions .....	35
6.1. Examples .....	38
7. Appendix B: Flow Information Elements .....	40
8. Appendix C: nProbe Usage Modes .....	44
9. Appendix D: nProbe License .....	45
10. Appendix E: EULA .....	45

## 2.Introduction

Traffic measurements are necessary to operate all types of IP networks. Networks admin need a detailed view of network traffic for security, accounting and management reasons. The compositions of the traffic have to be analyzed accurately when estimating traffic metrics or when finding network problems. All of these measurements have to be made by analyzing all the packets flowing to the central points in the network (such as router and/or switches). The analysis could be done on the fly or by logging all the packets and than post-processing them. But with the increasing network capacities and traffic volumes this kind of approach is not very efficient. Instead similar packets (packets with a set of common properties) can be grouped together composing flows. As an example, a flow can be composed of all flowing packets that share the same source and destination address so a flow can be derived using only some fields of a network packet. This way, similar types of traffic can be stored in a more compact format without losing the information we are interested in. This information can be aggregated in a flow datagram and exported to a collector able to report network metrics in a user-friendly format.

When collected this information provides a detailed view of the network traffic.

Precise network metric measurements is a challenging task so a lot of work has been done in this field. In commercial environments, NetFlow is probably the de-facto standard for network traffic accounting and billing. NetFlow is a technology originally created by Cisco in 1996 and is now standardized as Internet Protocol Flow Information eXport (IPFIX – RFC 3917). NetFlow is based on the probe/collector paradigm. The probe, usually part of network appliance such as a router or a switch, is deployed on the measured network segment, it sends traffic information in NetFlow format towards a central collector.

nProbe is a software NetFlow v5/v9/IPFIX probe able to collect, analyze and export network traffic reports using the standard Cisco NetFlow v5/v9/IPFIX format. It is available for most of the OSs on the market (Windows, BSD, Linux, MacOSX). When installed on a PC, nProbe turn it into a Network-aware monitoring appliance.

This manual aims at describing how to use nProbe, deploy it in networks, and how to develop plugins for extending it functionalities.

There are two main version of the nProbe probe:

- The one that is public available and distributed in both source and binary format (see Appendix for the license information) with only installation support.
- The binary nProbe distributed only in binary format that comes with additional features with respect to the previous version.

The first version of nProbe is available for use with no further configuration. On the other end the Pro version requires a per-server valid license in order to work.

## 2.1. Main Features

Some of the nProbe features include:

- Limited memory footprint (regardless of the network size and speed) and CPU savvy.
- Designed for running on environments with limited resources (the nProbe binary).
- Fully user configurable.
- Fully NetFlow v4/v5/v9 IPFIX compliant.
- High-performance probe: commercial probes included those embedded on routers and switches are often not able to keep up with high-speeds or, when able, their performance decreases dramatically handling small size packets.
- Ability to work as a NetFlow proxy.
- Support for disk-dump flow, either text files or SQLite files, and MySQL database server dump flow.

## 2.2. What's New?

Release 6.16 (April 2014)

- Updated nProbe with 6.16 features.

Release 6.12 (January 2014)

- Updated nProbe with 6.12 features.

Release 5.0 (February 2008)

- Updated nBox firmware
- Updated nProbe with latest features.
- Updated ntop with latest 3.3.X version.

Release 4.0 (July 2007)

- Updated nBox with latest 2.6 kernel series image
- Updated nProbe with 4.9 version coverage.

Release 3.9 (April 2005)

- Updated nBox section

Release 3.0.1 (February 2004)

- Updated nBox section

Release 3.0 (January 2004)

- Added nProbe 3.0 coverage

Release 2.2 (October 2003)

- Added nBox coverage

Release 2.1 (June 2003)

- Added nFlow support

Release 2.0.1 (February 2003)

- Added the ability to save flows on disk (-P flag)

Release 2.0 (January 2003)

- Added the ability to select multiple NetFlow collectors.
- Added -p flag for ignoring TCP/UDP ports.
- Added -e flag for slowing down flow export speed.
- Added -u flag for identifying input NetFlow devices into emitted flows.
- Added -z flag for preventing nProbe from emitting tiny flows.
- Added -a flag for selecting the way flows are exported to several collectors (if defined).
- Added the ability to control an LCD display where the probe can report traffic statistics.
- Enhanced TCP flags support in exported flows.

Release 1.3 (July 2002)

- First public release.

## 3. Using nProbe

The nProbe probe has to be activated on a PC from which it is possible to see/capture all the traffic you are interested in. For this reason, in case of switched networks, it is necessary to either mirror traffic (VLAN or port mirror) or place the probe on a location (e.g. by the border gateway) where most of the traffic flows.

When activated, nProbe will collect traffic data and emit NetFlow v4/v5/v9/IPFIX flows towards the specified collector. A set of packets with the same (src ip & port, dst ip & port, protocol #) is called flow (note that some protocols such as ICMP have no concept of ports). Every flow, even a very long-standing ISO CD image download, has a limited lifetime; this is because the flow collector should periodically receive flow chunks for accounting traffic precisely.

In the following sections, we discuss all the command line options and how to efficiently configure nProbe to run on your network.

### 3.1. Compiling nProbe Source Code

The nProbe source code (if you have decided to compile nProbe from source instead of using a binary package, on Unix it can be compiled as follows:

```
cd <nprobe source code directory>
./autogen.sh
make
```

On Windows the compilation is much more complicated as .NET compiler is needed and all the code dependencies must be satisfied. For this reason ntop releases a pre-built nProbe binary for the windows platform. Please note that the nProbe source code compiles both on Unix and Windows.

### 3.2. Installing a Binary nProbe

The windows version of nProbe comes in a standard installer package that can be installed using the wizard. On Linux, we pre-build two packages for the two most popular platforms Ubuntu Server LTE x64 and CentOS x64. We always build binaries for the latest server versions. Such packages can be installed from:

- <http://apt.ntop.org> (Ubuntu)
- <http://rpm.ntop.org> (CentOS)

Often the above packages can be installed on "sister" distributions such as Debian and RedHat/Fedora, although we cannot guarantee that they will work or install properly.

Once the installation is completed it is necessary to create the nProbe license otherwise the probe will operate in demo mode.

### 3.3. nProbe Command Line Options

nProbe allows network administrators to precisely tune the flow generation policy. In particular, it is possible to specify a lot of command line options.

Below are listed the available options and a detailed explanation of each option:

```
# nprobe -h

Welcome to nprobe v.6.16.140318 ($Revision: 4095 $) for x86_64-apple-darwin13.1.0

Copyright 2002-14 ntop.org

SystemID: 1FE719B8-0B82-5C67-9AE6-990B5030479F
WARNING: Invalid nProbe license (/etc/nprobe.license) [License mismatch error]

Usage:
nprobe -n <host:port|none> [-i <interface|dump file>] [-t <lifetime timeout>]
      [-d <idle timeout>] [-l <queue timeout>] [-s <snaplen>]
      [-p <aggregation>] [-f <filter>] [-a] [-b <level>] [-G] [-O <# threads>]
      [-P <path>] [-F <dump timeout>] [-D <format>]
      [-u <in dev idx>] [-Q <out dev idx>]
      [-I <probe name>] [-v] [-w <hash size>] [-e <flow delay>] [-B <packet count>]
      [-z <min flow size>] [-M <max num flows>]
      [-x <payload policy>] [-E <engine>] [-C <flow lock file>]
      [-m <min # flows>] [-R <cmd>]
      [-S <sample rate>] [-A <AS list>] [-g <PID file>]
      [-T <flow template>] [-U <flow template id>]
      [-o <v9 templ. export policy>] [-L <local nets>] [-c] [-r]
      [-1 <interface nets>] [-2 <number>] [-3 <port>] [-4] [-5 <port>] [-6]
      [-9 <path>] [--black-list <networks>] [--pcap-file-list <filename>]
      [-N <biflows export policy>] [--dont-drop-privileges]

[--collector|-n] <host:port|none> | Address of the NetFlow collector(s).
                                | Multiple collectors can be defined using
                                | multiple -n flags. In this case flows
                                | will be sent in round robin mode to
                                | all defined collectors if the -a flag
                                | is used. Note that you can specify
                                | both IPv4 and IPv6 addresses.
                                | If you specify none as value,
                                | no flow will be export; in this case
                                | the -P parameter is mandatory.
                                | Note that you can specify the protocol
                                | used to send packets. Example:
                                | udp://192.168.0.1:2055,tcp://10.1.2.3:2055

[--interface|-i] <iface|pcap> | Interface name from which packets are
                              | captured, or .pcap file (debug only).

[--lifetime-timeout|-t] <timeout> | It specifies the maximum (seconds) flow
                                  | lifetime [default=120]

[--idle-timeout|-d] <timeout> | It specifies the maximum (seconds) flow
                              | idle lifetime [default=30]

[--queue-timeout|-l] <timeout> | It specifies how long expired flows
                              | (queued before delivery) are emitted
                              | [default=30]

[--snaplen|-s] <snaplen> | Packet capture snaplen [default 128 bytes]

[--aggregation|-p] <aggregation> | It specifies the flow aggregation level:
                                  | <VLAN Id>/<proto>/<IP>/<port>/<TOS>/<AS>
                                  | where each element can be set to 0=ignore
                                  | or 1=take care. Example '-p 1/0/1/1/1/1'
                                  | ignores the protocol, whereas
                                  | '-p 0/0/1/0/0/0' ignores everything
                                  | but the IP

[--bpf-filter|-f] <BPF filter> | BPF filter for captured packets
                              | [default=no filter]

[--all-collectors|-a] | If several collectors are defined, this
                              | option gives the ability to send all
                              | collectors all the flows. If the flag is
                              | omitted collectors are selected in
                              | round robin.

[--verbose|-b] <level> | Verbose output:
                              | 0 - No verbose logging
                              | 1 - Limited logging (traffic statistics)
                              | 2 - Full verbose logging

[--daemon-mode|-G] | Start as daemon.

[--num-threads|-O] <# threads> | Number of packet fetcher threads
                              | [default=1]. Use 1 unless you know
                              | what you're doing.

[--dump-path|-P] <path> | Directory where dump files will
                              | be stored.

[--exec-cmd-dump|-R] <cmd> | Execute the specified command for each
                              | file dump on disk (including plugins).

[--dump-frequency|-F] <dump timeout> | Dump files dump frequency (sec).
                              | Default: 60

[--dump-format|-D] <format> | <format>: flows are saved as:
                              | b : raw/uncompressed flows
                              | B : raw core flow fields (152 bytes)
                              | t : text flows
                              | d : SQLite
                              | Example: -D b. Note: this flag has no
                              | effect without -P.

[--in-iface-idx|-u] <in dev idx> | Index of the input device used in the
                              | emitted flows (incoming traffic). Default
                              | value is 0. Use -1 as value to dynamically
```

```

| set to the last two bytes of
| the MAC address of the flow sender.
|--out-iface-idx|-Q] <out dev idx> | Index of the output device used in the
| emitted flows (outgoing traffic). Default
| value is 0. Use -1 as value to dynamically
| set to the last two bytes of
|--vlanid-as-iface-idx] <mode> | the MAC address of the flow receiver.
| Use vlanId (0 for untagged traffic)
| as interface index. Mode specifies with
| stacked VLANs which vlanId to choose. Values
| are 'inner', 'outer', 'single', or 'dual':
| inner = use the most inner VLAN tag
| outer = use the first (the one close to ether) VLAN tag
| single = for even outer VLAN tags 'E',
|         where E={2,4,6...4094},
|         ifIdx is set to IN='0',OUT='E'.
|         For odd outer VLAN tags 'O',
|         where O={3,5,7...4095},
|         ifIdx is set to IN='O-1',OUT='0'
| double = for even outer VLAN tags 'E',
|         where E={2,4,6...4094}, ifIdx
|         is set to IN='E+1',OUT='E'.
|         For odd outer VLAN tags 'O',
|         where O={3,5,7...4095},
|         ifIdx are set to IN='O-1',OUT='O'
| Note that this option
| supersedes the --in/out-iface-idx options
|--discard-unknown-flows] <mode> | In case you enable L7 proto detection
| (e.g. add %L7_PROTO to the template)
| this options enables you not to export
| flows for which nDPI has not been able
| to detect the proto. Mode values:
| 0 - Export known/unknown flows (default)
| 1 - Export only known flows (discard
|     flows with unknown protos)
| 2 - Export only unknown flows (discard
|     flows with known protos)
|--nprobe-version|-v] | Prints the program version.
|--flow-lock|-C] <flow lock> | If the flow lock file is present no flows
| are emitted. This facility is useful to
| implement high availability by means of
| a daemon that can create a lock file
| when this instance is in standby.
|--help|-h] | Prints this help.
--interpret-flow-packets | Interpret received packets to see
| if they contain flows (development only).
--debug | Enable debugging (development only).
--json-labels | In case JSON label is used (e.g. with ZMQ)
| labels instead of numbers are used as keys.
--quick-mode | Micro-nprobe: use if need speed
| and do not need advanced traffic analysis.
--fake-capture | Fake packet capture (development only).
--drop-flow-no-plugin | Drop flows that have not processed by a plugin.
--dont-nest-dump-dirs | Dump files won't be saved on nested dirs.
--performance | Enable performance tracing (debug only).
[--syslog|-I] <probe name> | Log to syslog as <probe name>
| [default=stdout]
[--hash-size|-w] <hash size> | Flows hash size [default=131072]
[--no-ipv6|-W] | IPv6 packets will not be accounted.
[--flow-delay|-e] <flow delay> | Delay (in ms) between two flow
| exports [default=1]
[--count-delay|-B] <packet count> | Send this many packets before
| the -e delay [default=1]
[--min-flow-size|-z] <min flow size> | Minimum TCP flow size (in bytes).
| If a TCP flow is shorter than the
| specified size the flow is not
| emitted [default=unlimited]
[--max-num-flows|-M] <max num flows> | Limit the number of active flows. This is
| useful if you want to limit the memory
| or CPU allocated to nProbe in case of non
| well-behaved applications such as
| worms or DoS. [default=524288]
[--netflow-engine|-E] <type:id> | Specify the engine type and id.
| The format is engineType:engineId.
| [default=0:66] where engineId is a
| random number.
[--min-num-flows|-m] <min # flows> | Minimum number of flows per packet
| unless an expired flow is queued
| for too long (see -l) [default=30
| for v5, dynamic for v9]
[--sender-address|-q] <host:port> | Specifies the address:port of the flow
| sender. This option is useful for hosts
| with multiple interfaces or if flows
| must be emitted from a static port/IP.
[--sample-rate|-S] <pkt rate>:<flow rate> | Packet capture sampling rate and flow
| sampling rate. If <pkt rate> starts with
| '@' it means that nprobe will report
| the specified sampling rate but will
| not sample itself as incoming packets
| are already sampled on the specified
| capture device at the specified rate.
| Default: 1:1 [no sampling]
|--as-list|-A] <AS list> | GeoIP file containing with known ASs.
| Example: GeoIPASNum.dat
--city-list <city list> | GeoIP file containing the city/IP mapping.
| Note that nProbe will load the IPv6 file
| equivalent if present. Example:
| --city-list GeoLiteCity.dat will also

```

```

|--pid-file|-g] <PID file> | attempt to load GeoLiteCityv6.dat
|--flow-templ|-T] <flow template> | Put the PID in the specified file
|--flow-templ-id|-U] <templ. id> | Specify the NFv9/IPFIX template.
| Specify the NFv9/IPFIX template identifier
| [default: 257]
|--flow-version|-V] <version> | NetFlow Version: 5=NFv5, 9=NFv9, 10=IPFIX
|--flows-intra-templ|-o] <num> | Specify how many flow pkts are exported
| between template exports [default: 10]
|--local-networks|-L] <nets> | Specify the list of local networks whose
| format is <net>/<mask> (if multiple use comma).
|--local-hosts-only|-c] | All the IPv4 hosts outside the local
| network lists will be set to 0.0.0.0
| (-L must be specified before -c).
| This reduces the load on the probe
| instead of discarding flows on the
| collector side.

|--local-traffic-direction|-r] | All the traffic going towards
| the local networks (-L must also be
| specified before -r) is assumed incoming
| traffic all the rest is assumed outgoing
| (see also -u and -Q).

|--max-flow-size|-0] <size> | Specify the maximum flow size. NOTE:
| This parameter has influence on -m.
|--if-networks|-1] <nets> | Specify the binding between interfaceId
| and a network (see below).
|--count|-2] <number> | Capture a specified number of packets
| and quit (debug only)
|--collector-port|-3] <port> | NetFlow/IPFIX/sFlow collector flows port
|--tunnel|-5] | Compute flows on tunneled traffic rather than
| on the external envelope
|--no-promisc|-6] | Capture packets in non-promiscuous mode
|--smart-udp-frag|-7] | Ignore UDP fragmented packets with fragment offset
| greater than zero, and compute the fragmented
| packet length on the initial fragment header.
|--ipsec-auth-data-len|-8] <len> | Length of the authentication data of IPsec
| in tunnel mode. If not set, IPsec will not be decoded
|--dump-stats|-9] <path> | Periodically dump traffic stats into the
| specified file
--black-list <networks> | All the IPv4 hosts inside the networks
| black-list will be discarded.
| This reduces the load on the probe
| instead of discarding flows on the
| collector side.
--pcap-file-list <filename> | Specify a filename containing a list
| of pcap files.
| If you use this flag the -i option will be
| ignored.
|--biflows-export-policy|-N] <pol> | Bi-directional flows export policy:
| 0 - export all flows
| 1 - export bi-directional flows only
| 2 - export mono-directional flows only
--csv-separator <separator> | Specify the text files separator (see -P)
| Default is '|' (pipe)
--dont-drop-privileges | Do not drop privileges changing to user nobody
--bi-directional | Force flows to be bi-directional. This option
| is not supported by NetFlow V5 that by nature
| supports only mono-directional flows
--account-l2 | NetFlow accounts IP traffic only, not counting
| L2 headers. Using this option the L2 headers
| are also accounted
--dump-metadata <file> | Dump flow metadata into the specified file
| and quit. Useful for knowing the IE handled.
--dump-pkts <.pcap file> | Dump incoming packets on the specified dump
--max-log-lines <num> | Maximum number of lines on a dump file. Default: 10000.
--timestamp-format <mode> | Specified the timestamp format on dump files. Value:
| 0 - Unix Epoch
| 1 - Unix Epoch with microseconds
| 2 - Human readable timestamp
--ndpi-proto <proto> | Comma separated list of NDPI protocols to enable. If
| not specified, all known protocols are detected.
--account-imsi-traffic | When used with GTP traffic and --redis, the user traffic
| is accounted per IMSI/NSAPI (mobile traffic only)
--event-log <file> | Dump relevant activities into the specified log file
--collection-filter <filter> | Filter applied to collected filters only (-3). Filter format:
| [!]<asX | network/mask> (! means discard flows matching filter)
| Example: !as12345, 192.168.0.0/24, !10.0.0.0/8
--imsi-aggregation | Aggregate IMSI traffic (GTP traffic only)
--simulate-storage | Simulate storage to disk (debug only)
--zmq <socket> | Deliver flows to subscribers connected to the specified endpoint.
| Example tcp://*:5556 or ipc://flows.ipc
--tcp <server:port> | Deliver flows in JSON format to the specified server via TCP.
--dump-bad-packets <file> | Dump bad/undecodeable packets into the specified pcap file
--lru-cache-size <size> | Users and protocol cache size. Default 16384
--enable-throughput-stats | Compute throughput stats that can be dumped when -P is used
--ndpi-proto-ports <file> | Read custom ports definitions for NDPI (see NDPI/example/protos.txt)
--disable-l7-protocol-guess | When NDPI is enabled, in case a protocol is not recognized,
| nProbe guesses the protocol based on ports. This option disables
| this feature and uses only strict payload dissection
--original-speed | When using -i with a pcap file, instead of reading packets
| as fast as possible, the original speed is preserved (debug only)
--dont-reforge-timestamps | Disable nProbe to reforge timestamps with -i <pcap file> (debug only)
--db-engine <database engine> | Define the DB engine type (example MyISAM, InnoDB).
| This information is used by the database plugin.
| Default MyISAM.
--unprivileged-user <name> | Use <name> instead of nobody when dropping privileges
--disable-cache | Disable flow cache for avoid merging flows. This option
| is available only in collector/proxy mode
| (i.e. use -i none)
--redis <host>[:<port>] | Connected to the specified redis server

```

```

--use-redis-proxy      | Example --redis localhost
                       | Use a redis proxy (e.g.
                       | https://github.com/twitter/twemproxy)
--ucloud              | Enable the nProbe micro-cloud
--show-system-id      | Print the system identifier
--check-license       | Checks if the license is present and valid
--dump-plugin-families | Dump all available plugin families

```

#### -n: collector addresses

This specifies the NetFlow collectors addresses to which nProbe will send the flows. If more than one is specified, they need to be separated with a comma or the `-n` flag can be repeated several times (e.g. `-n 172.22.3.4:33,172.22.3.4:34` and `-n 172.22.3.4:33 -n 172.22.3.4:34` are equivalent). When multiple collectors are defined, you can control the way flows are exported using the `-a` option (see below); if on a collector address the destination port is omitted, flows are sent to 2055 port and whereas if all the option is not specified, by default, flows are sent to the loop back interface (127.0.0.1) on port 2055. If this parameter is used, nProbe exports flows towards collector running at 127.0.0.1:2055. By default the UDP protocol is used but also TCP and SCTP (Linux only when nProbe is compiled with SCTP support and the kernel supports it). In this case you can specify the collector address as `udp://<host>:<port>`, `tcp://<host>:<port>`, and `sctp://<host>:<port>`,

#### -i: interface name

It specifies the interface from which packets are captured. If `-i` is not used, nProbe will use the default interface (if any). In case a user needs to activate nProbe on two different interfaces, then he/she needs to activate multiple nProbe instances once per interface. For debugging purposes it is possible to pass nProbe a `.pcap` file from which packets will be read. If nProbe is compiled and activated with `PF_RING` support, you can specify multiple interfaces from which packets are captured. Example `"-i eth0,eth1"`

#### -t: maximum flow lifetime

Regardless of the flow duration, a flow that has been active for more that the specified maximum lifetime is considered expired and it will be emitted. Further packets belonging to the same flow will be accounted on a new flow.

#### -d: maximum flow idle lifetime

A flow is over when the last packet received is older that the maximum flow idle lifetime. This means that whenever applicable, (e.g. SNMP walk) UDP flows will not be accounted on 1 packet/1 flow basis, but on one global flow that accounts all the traffic. This has a benefit on the total number of generated flows and on the overall collector performance.

#### -l: maximum queue timeout

It specifies the maximum amount of time that a flow can be queued waiting to be exported. Use this option in order to try to pack several flows into fewer packets, but at the same time have an upper bound timeout for queuing flows into the probe.

#### -s: snaplen

This flag specifies the portion of the packet (also called snaplen) that will be captured by nProbe. By default nprobe sets the snaplen automatically according to its configuration, but you can override its value using thia flag.

**-p: flow aggregation**

Flows can be aggregated both at collector and probe side. However probe allocation is much more effective as it reduces significantly the number of emitted flows hence the work that the collector has to carry on. nProbe supports various aggregation levels that can be selected specifying with the `-p` flag. The aggregation format is `<vlanid>/<proto>/<IP>/<port>/<TOS>/<AS>` where each option can be set to 0 (ignore) or 1 (take care). Ignored fields are set to a null value. For instance the value `0/0/1/0/0/0` is useful for creating a map of who's talking to who (network conversation matrix).

**-f: packet capture filter**

This BPF filter (see the appendix for further information about BPF filters) allows nProbe to take into account only those packets that match the filter (if specified).

**-a: select flow export policy**

When multiple collectors are defined (see `-n` option), nProbe sends them flows in round robin. However it is possible to send the same flow to all collectors as a flow redirector does if the `-a` option is used.

**-b: enable verbose logging**

Using this flag, nProbe generates verbose output that can be used to tune its performance (see chapter 2.4). Zero is the lowest level (little information is printed), 1 displays traffic statistics, 2 is really verbose. Example of traffic statistics:

```
04/Jul/2007 18:16:00 [nprobe.c:1129] Average traffic: [1.7 pkt/sec][1 Kb/sec]
04/Jul/2007 18:16:00 [nprobe.c:1134] Current traffic: [1.9 pkt/sec][1 Kb/sec]
04/Jul/2007 18:16:00 [nprobe.c:1140] Current flow export rate: [0.9 flows/sec]
04/Jul/2007 18:16:00 [nprobe.c:1144] Buckets:
[active=13][allocated=21][free=8][toBeExported=0][frags=0]
04/Jul/2007 18:16:00 [nprobe.c:1149] Fragment queue: [len=0]
04/Jul/2007 18:16:00 [nprobe.c:1153] Num Packets: 111 (max bucket search: 0)
04/Jul/2007 18:16:00 [nprobe.c:1170] 115 pkts rcvd/0 pkts dropped
```

**-G: start nprobe as a daemon.**

Useful when starting nprobe as daemon.

**-O: set the number of threads that fetch packets out of the network interface.**

In general: the more threads are available, the better is the performance. However it is not suggested to have too many threads as in some platforms this can slow down the probe. Start with 1 and increase it if necessary. We suggest to run nprobe as single threaded application and distribute the traffic across multiple probes using PF\_RING (e.g. PF\_RING cluster or libzero). In fact adding threads you will end up spending a lot of time on synchronization without improving the performance. Please refer to this post <http://www.ntop.org/nprobe/10-gbit-line-rate-netflow-traffic-analysis-using-nprobe-and-dna/> for more information.

**-P: dump flows**

This path specifies the directory where flows will be dumped. The dump format is text and it depends on the nProbe template specified with `-T`.

**-F:**

It specifies the frequency at which files are dumped on disk

**-D: dump flows format**

Flows stored on disks can be stored in two formats: text with user-specified format or SQLite format (availability depends on the platform and if nProbe has been compiled with it). Using flow SQLite format (-D d) can significantly reduce the size of stored files, although all the collectors might not support this format. Text flows (-D t) are the safest setting if you want to use a standard collector able to read flows dump on disk. You can also export core flow fields (-D B) in binary format for post-processing by binary applications. Note that this flag has no effect unless -P is used.

-u: input device index

The NetFlow specification contains a numeric index in order to identify flows coming from different interfaces of the same probe. As multiple nProbe instances can be started on the same host but on different devices, the collector to divide flows according to interface number can use this flag. If -u is not used, then nprobe will use 0 as interface index, instead of -1 is used the last two bytes of the mac address of the flow sender will be used as index.

-Q: output device index

Similar to -u but for the output interface.

--vlanid-as-iface-idx <mode: inner | outer>

nProbe can use the VLAN tag as interface identifier. Using this flag you enable this feature. As VLAN tags can be stacked you need to specify if the inner or outer tag will be used for the interface identifier.

--discard-unknown-flows <mode:0 | 1 | 2>

nProbe includes nDPI support for analyzing packet contents in order to detect application protocol. The mode value can be used to:

- 0: Export all know (i.e. those whose application protocol has been detected) and unknown (i.e. the application protocol is unknown)
- 1: Export only know flows, discarding unknown flows.
- 2: Export only unknown flows, discarding known flows.

-v: print version

This flag is used to print the nProbe version number and date.

-C: flow export lock

This is a simple way to implement high-availability. Start two probes capturing the same data. The master probe emit flows, the slave probe is started with -C <path>. As long as <path> exists, the slave works but no flow is emitted. If the <path> file is deleted (e.g. using an external program for controlling the master/slave such as heartbeat) the slave starts emitting flows. If the file is restored, the slave is silent again.

-h: print help

Prints the nProbe help.

--quick-mode

nProbe is computing many statistics, but if you care just about basic netflow (i.e. V5 or V9/IPFIX flows with standard fields) you can use this flag to expedite operations telling nProbe to avoid doing many unnecessary things (e.g. handle L2 traffic). Use this option if you care about speed.

--dont-nest-dump-dirs

nProbe dumps data on disk (e.g. with -P) using a nested directory. In essence the base directory will be partitioned in sub-directories with <year>/<month>/<day>/<hour>/<min> structure. use this option is you want nProbe to dump all data in the base directory without creating this nested directory tree.

-l: log to syslog <probe name>

nProbe logs on stdout unless the -g flag (see above) is used. If the syslog needs to be used instead of a file, this flag instruments nProbe to log on it using the specified name (this is useful when multiple nProbe instances are active on the same host). Please note that -g is ignored if -l is used, and this option is not available on nProbe for Win32.

-w: size of the hash that stores the flows

The default size is 131072 and it should be enough for most of networks. In case flows are not emitted often and with strong traffic conditions it would be necessary to increase the hash. See later in this manual for knowing more about nProbe tuning.

-W: Discard IPv6 traffic

Use this flag if you want nProbe not to account IPv6 traffic.

-e: flow export delay

Some collectors cannot keep up with nProbe export speed. This flag allows flows to be slow down by adding a short delay (specified in ms) between two consecutive exports. The maximum allowed delay is 1000 ms.

-B: packet count delay

It specified how many flow packets need to be sent before -e is applied,

-z: minimum TCP flow size

Peer-to-peer applications, attacks or misconfigured applications often generate a lot of tiny TCP flows that can cause significant load on the collector side. As most collector setups often discarded those flows, it is possible to instrument nProbe via the -z flag not to emit such flows. Note that the -z flag affects only the TCP protocol (i.e. UDP, ICMP and other protocols are not affected).

-M: maximum number of active flows

It is used to limit the maximum number of concurrent flows that the probe can sustain. This is useful for preventing the probe from creating as many flows as needed and hence to take over all the available resources.

-E: netflow engine

Specify the netflow engineType:engineId into the generated flows.

-m: minimum number of flows per packet

In order to minimize the number of emitted packets containing flows, it is possible to specify the minimum number of flows that necessarily need to be contained in a packet. This means that the packet is not emitted until the specified number of flows is reached.

-q: flow sender address

This option is used to specify the address and port from which the packets containing flows are coming from. Usually the operating systems prevents people from sending packets from addresses different from those assigned to the network interfaces.

-S: sample rate <packet rate>:<flow rate>

nProbe uses all the captured packets for calculating flows. In some situations (e.g. strong traffic conditions) it is necessary to reduce the number of packets that need to be handled by nProbe. This option specifies the sampling rate, i.e. the number of packets that are discarded between two packets used to produce flows. You can also specify the flow sample rate that reduce the egress flow rate thus lowering the load on collectors. The default value is 1:1 (no packet sample, no flow sample).

-A: AS file

Network probes are usually installed on systems where the routing information is available (e.g. via BGP) in order to specify the AS (Autonomous System) id of the flow peer. As nProbe has no access to BGP information unless you enable the BGP plugin, users need to provide this information by means of a static file whose format is <AS>:<network>. The file can be stored in both plain text and gzip format.

--city-list: City List

With this option you can enable geolocation of IP addresses at city/country detail level. Here you need to specify the GeolP city database (e.g. GeoliteCity.dat)

-g:

It specifies the path where nProbe will save the process PID.

-T: flow template definition

Contrary to NetFlow v5 where the flow format is fixed, NetFlow V9 and IPFIX flows have a custom format that can be specified at runtime using this option as specified in appendix.

-U: flow template id

NetFlow v9 and IPFIX flows format is specified in a template whose definition is sent by nProbe before to start sending flows. The flow format is defined by -T, where -U is used to set the template identifier. This option should not be used unless the default template value (257) needs to be changed. As based on -T nProbe can define several templates, this value is the one used for the first defined template.

-V: flow export version

It is used to specify the flow version for exported flows. Supported versions are 5 (v5), 9 (v9) and 10 (IPFIX).

-o: intra templates packet export.

It specifies the number of flow packets that are exported between two templates export.

-L: local networks

Use this flag to specify (format network/mask, e.g. 192.168.0.10/24) the list of networks that are considered local (see -c).

-c: track local hosts only

It allows nProbe to set to 0.0.0.0 all those hosts that are considered non-local (see -L). This is useful when it is necessary to restrict the traffic analysis only to local hosts.

-r: set traffic direction

When this option is used (-L must be specified before -r), all the traffic that goes towards the local networks is considered incoming, all the rest is outgoing. This has effect on the -u/-Q that are then forced with -r.

--if-networks: specify a mapping between MAC address/Interface index

Flags -u and -Q are used to specify the SNMP interface identifiers for emitted flows. In mirrored environments, it is possible to simulate a switched environment by playing with MAC addresses. This option allows users to bind a MAC or IP address to a specified interface. The syntax of --if-networks is <MAC|IP/mask>@<interface> where multiple entries can be separated by a comma (,). Example: --if-networks "AA:BB:CC:DD:EE:FF@3,192.168.0.0/24@2" or --if-networks @<filename> where <filename> is a file path containing the networks specified using the above format.

--count: debug only

Let the probe capture only up to the specified number of packets.

--collector-port: specifies the NetFlow collector port

It is now possible to use the nProbe as NetFlow proxy. With --collector-port we can see the incoming NetFlow port on which flows are received instead of sniffing packets. nProbe is able to convert flows from various versions. For instance "nprobe --collector-port 2055 -i 192.168.0.1:2056 -V 10" converts each flow received on port 2055 to IPFIX and sends them to 192.168.0.1:2056.

--tunnel:

Let the probe decode tunneled traffic (e.g. GTP or GRE traffic) and thus extract traffic information from such traffic rather than from the external envelope.

--no-promisc:

With this option nProbe does not use promiscuous mode to capture packets.

--smart-udp-frags:

Ignore UDP fragmented packets with fragment offset greater than zero, and compute the fragmented packet length on the initial fragment header. This flag might lead to inaccuracy in measurement but it speeds up operations with fragmented traffic.

--ipsec-auth-data-len

Length of the authentication data of IPsec in tunnel mode. If not set, IPsec will not be decoded but just accounted.

--dump-stats: dump some flow statistics on file

Periodically dump NetFlow statistics on the specified file. Note that when using nProbe over PF\_RING, nProbe dumps statistics on /proc/net/pf\_ring/stats/<nprobe stats file>.

--black-list

With this option you can specify a list of networks or hosts from which all the incoming packets will be discarded by the probe. The accepted notation can be CIDR format or the classical network/netmask format.

--pcap-file-list <file>

The specified file path contains a list of pcap files to be read in sequence by nProbe. Use this option when you want nProbe to read a list of pcap files (e.g. when generated using tcpdump).

--biflows-export-policy <policy>

Bi-directional flows are such when there is traffic in both direction of the flow (i.e. source->dest and dest->source). As mono-directional flows might indicate suspicious activities, this flag is used to determine the export policy:

- 0: Export all know (i.e. mono and bi-directional flows)
- 1: Export only bi-directional flows, discarding mono-directional flows.
- 2: Export only mono-directional flows, discarding bi-directional flows.

--csv-separator <separator>

Override the default 'l' separator in dumps with the specified one.

--dont-drop-privileges

Do not drop root privileges to user 'nobody' when this option is specified. See al --unprivileged-user later int this manual.

--bi-directional

Force flows to be bi-directional. This option is not supported by NetFlow V5 that by nature supports only mono-directional flows

--account-l2

NetFlow accounts IP traffic only, not counting layer 2 headers. Using this option the layer 2 headers are also accounted in flow traffic statistics.

--dump-metadata <file>

Dump metadata information into the specified file and quit. This option is useful when users want to know the type of each information element exported by nProbe so that (for instance) they can properly import into a database.

--event-log <file>

Dump relevant activities (e.g. nProbe start/stop or packet drop) onto the specified file.

--enable-throughput-stats

When -P is used, with this option is also possible to generate throughput information. The file has the following format: <epoch> <bytes> <packets>. Each line is printed every second and it contains the number of bytes and packets observed within minute.

--ndpi-proto-ports <file>

Read the nDPI custom protocol and ports configuration from the specified file. Please refer to the nDPI manual for further information about the format of this file.

--disable-l7-protocol-guess

When nDPI is unable to detect a protocol, nProbe uses the port information to guess the protocol. This flag prevents nProbe from doing that, so protocols are detected only by nDPI without relying on default ports.

--db-engine <database engine>

In case flows are dumped on a MySQL database (see later on this manual) the default database engine used by nProbe is MyISAM. With this option you can use another engine (e.g. InnoDB).

--unprivileged-user <name>

When nprobe drops privileges (unless `--dont-drop-privileges` is used) the user nobody is used. It is possible to use another user by using this option.

#### `--disable-cache`

nProbe implements a flow cache for merging packets belonging to the same flow. In proxy/collector mode, nProbe can disable this feature so that incoming flows are not put in cache but immediately exported.

#### `--redis <host>[:<port>]`

The redis database (when nProbe is compiled with it) is used to implement a data cache and for aggregating flow information. This option specifies the host (and optionally the port) where redis is listening. nProbe opens several connections to redis (not just one) in order to maximize performance.

#### `--ucloud`

This option enables the micro-cloud concept. Please refer to <http://www.ntop.org/nprobe/monitoring-on-the-microcloud/> for more information.

#### `--show-system-id`

Shown the systemId where nProbe is running (for binary nProbe's only).

#### `--check-license`

Checks if the configured license is valid (for binary nProbe's only).

#### `--dump-plugin-families`

Dump installed plugin family names.

As some people prefer to have a configuration file containing the options that otherwise would be specified on the command line, it is also possible to start nProbe as follows:

```
nprobe <configuration file path>
```

where the configuration file contains the same options otherwise specified on the command line. The only difference between the command line and the configuration file is that different options need to be specified on different lines. For instance:

```
nprobe -n 127.0.0.1:2055 -i en0 -a -p
```

is the same as:

```
nprobe /etc/nprobe.conf
```

where `/etc/nprobe.conf` contains the following lines:

```
# cat /etc/nprobe.conf
```

```
-n=127.0.0.1:2055
-i=en0
-a=
-p=
```

Note that flags with no parameter associated (e.g. `-a`) also need to have '=' specified. Any standard NetFlow collector (e.g. ntop) can be used to analyze the flows generated by nProbe. When used with ntop, the nProbe can act as a remote and light traffic collector and

ntop as a central network monitoring console. See chapter 3 for further information about this topic.

### 3.4. nProbe on Windows

nProbe is activated as service or application (i.e. you can start it from cmd.exe). The nProbe installer registers the service and creates an entry on the Start menu. Example:

```
E:\ntop\Source\nprobe\Debug>nprobe /h
```

Available options:

```
/i [nprobe options] - Install nprobe as service
/c [nprobe options] - Run nprobe on a console
/r                   - Deinstall the service
```

Example:

Install nprobe as a service: 'nprobe /i -i 0 -n 192.168.0.1:2055'

Remove the nprobe service: 'nprobe /r'

Notes:

1. Type 'nprobe /c -h' to see all options
2. In order to reinstall a service with new options it is necessary to first remove the service, then add it again with the new options.
3. Services are started/stopped using the Services control panel item.

If nProbe is started on the console, the /c flag needs to be used (e.g. nprobe /c -n 127.0.0.1:2055). If used as service, the command line options need to be specified at service registration and can be modified only removing and adding the service. The nProbe installer registers nProbe as a service with the default options. If you need to change the nProbe setup, you need to do as follows:

nprobe /r	Remove the service
nprobe /i <put your options here>	Install the service with the specified options.

Services are started and stopped using the Services application part of the Windows administrative tools.

As network interfaces on Windows can have long names, a numeric index is associated to the interface in order to ease the nProbe configuration. The association interface name and index is shown by typing the 'nprobe /c -h'

```
C:\ntop\nprobe\Debug>nprobe.exe/c -h
Running nProbe for Win32.
```

```
Welcome to nprobe v.4.9.2 for Win32
Built on 05/03/07 10:35:28
Copyright 2002-07 by Luca Deri <deri@ntop.org>
```

[...]

Available interfaces:

```
[index=0] 'Adapter for generic dialup and VPN capture'
[index=1] 'Realtek 8139-series PCI NIC'
```

[...]

For instance, in the above example the index 1 is associated to the interface Realtek 8139-series PCI NIC, hence in order to select this interface nprobe needs to be started with `-i 1` option.

### 3.5. Licenses Installation

Binary nProbe instances require a per-server license that is released according to the EULA (End User License Agreement) as specified in the appendix. Each license is perpetual (i.e. it does not expire) and it allows to install updates for one year since purchase/license issue. This means that a license generated on 1/1/2013 will be able to activate new versions of the software until 1/1/2014. If you want to install new versions of the software release after that date, you need to purchase a new license or avoid further updating the software. For source-based nProbes you still have to obey to the nProbe license listed in appendix.

nProbe licenses are generated using the orderId and email you provided when the license has been purchased on <http://shop.ntop.org/>. The licenses are generated at <http://www.nmon.net/mklicense/>.

### 3.6. Tuning nProbe Performance

As nProbe can be deployed on very different environments, it is necessary to tune it according to the network where is active. In order to achieve a good probe setup, it is necessary to understand how nProbe is working internally. Each captured packet is analyzed, associated to a flow, and stored onto a hash. Periodically, the hash is analyzed and expired flows are emitted<sup>1</sup>. The hash size is static (`-w` flag)<sup>2</sup> as this allows nProbe to:

- Allocate all the needed memory at startup (this is compulsory on embedded systems where memory is limited and it is necessary to know at startup whether a certain application can operate with the available resources).
- Avoid exhausting all the available memory in case of attacks that can produce several flows.

Selecting the hash size is a matter of trade-off between efficiency (an efficient hash is at least 1/3 empty) and memory usage. This statement does not mean that a huge hash is always the solution as the flow export process can be slower (and more CPU cycles are needed) as a large hash needs to be explored.

On the other hand, the hash size is just a part of the problem. In fact, the hash fill percentage can be also controlled by other factors such as:

- Reducing the flow lifetime (`-t`)
- Reducing the maximum flow idle time (`-d`)
- Increasing how often the hash is walked searching expired flows (`-s`)

nProbe allows users to ease the tuning process by printing the status of internal hashes using the `-b` flag. Users who experience severe nProbe performance problems, packet loss

<sup>1</sup> It is worth to remark that packets are captured while nProbe performs flow export (i.e. packet capture is not stopped during flow export).

<sup>2</sup> Note that the basic hash has a static size specified by `-w` that can grow as needed according to traffic conditions.

or high CPU usage, should start nProbe with `-b` in order to find out whether their probe setup is optimal.

### 3.7. Using nProbe with ntopng

On the Internet there are several NetFlow collectors (see Reference paragraph) that can be used to handle flows generated by nProbe. Among them ntopng is included. This section explains how to configure ntopng to take advantage of nProbe.

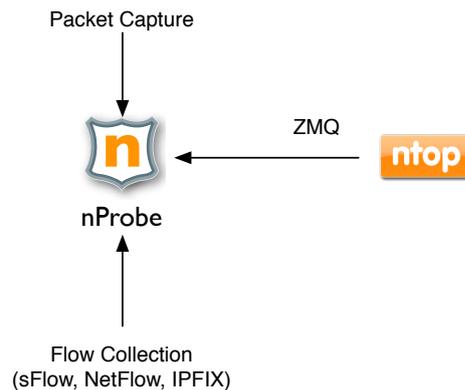


Fig. 1– Using ntop with nProbe

ntopng is using nProbe for converting flows from their native format to the ntop format (JSON based). Flows are delivered by nProbe to ntopng via ZMQ. In this case ntopng connects to nProbe via ZMQ, contrary to what happens with NetFlow probes that send flows to the collector. Supposing to have the probe active on host whose IP is 1.2.3.4 and ntopng on host 1.2.3.5, the configuration to be used are (in case host apps are on the same host, please replace the IPs with 127.0.0.1):

- `ntopng -i tcp://1.2.3.4:5556`
- `nprobe --zmq "tcp://*:5556" -i ethX -n none -b 2`

### 3.8. Frequently Asked Questions

1. Q: I'm sending 60 bytes ping packets using `'ping -s 60'` but nProbe reports 92 bytes packets.  
A: nProbe counts the packet size at IP level. An ICMP Echo Request packet with 60 bytes payload is 92 bytes long.
2. Q: I need to capture traffic from several interfaces but nProbe allows just one interface to be used. What can I do?  
A: You can start several instances of nProbe, each on a different network interface.
3. Q: nProbe is exporting flows too fast and my collector cannot keep up with it. How can I slow down nProbe export rate?  
A: nProbe has been for high-speed networks (1Gb and above) so its export rate can be high due to traffic conditions. There are several solutions available:

- a. Specify a minimum intra-flow delay (-e flag)
- b. Use several collectors and send them flows in round robin (-n flag) in order to balance load among the collectors.

## 4. nProbe Plugins

nProbe has been designed as an engine that processes packets and compute basic statistics, and plugins that extend the core with additional capabilities. Each plugin dissects a specific traffic (e.g. SMTP email traffic), but you can enable the use of multiple plugins simultaneously. nProbe based on the template configuration (-T) will selectively enable plugins and define as many templates as necessary. Their number depends on the plugins enabled and on the fact that you might enable IPv4 and/or IPv6 traffic support.

The following sections cover the configuration and information elements provided by each individual plugin. Most plugins are available also in source format, but sometimes due to license restrictions (e.g. the plugin has been sponsored by a company that does not want others to access the source code) we are unable to release all plugins in source format.

### 4.1. BGP Plugin

This plugin is used in combination with the `bgp_probe_client.pl` script for receiving BGP information and updates from a router. In order to use it you need to:

- Edit the `bgp_probe_client.pl` file and configure the IP address of the machine where the script is listening (`$local_ip`) and its AS (`$local_as`), the IP address of the router (`$remote_ip`) and its AS (`$remote_as`). Of course you better define a private AS for doing all this.

```
# BGP
my $local_ip = '192.168.48.2';
my $local_as = 65498;
my $remote_ip = '192.168.48.1';
my $remote_as = 2597;

# nProbe
my $nprobe_ip = '127.0.0.1';
my $nprobe_port = 4096;
```

- Start the script and configure the router to connect to the script (that acts as a server). The router will initially send its BGP table, and then periodically send BGP updates.
- Start nProbe on the same machine where the script is active with the option `--bgp-port <port>` where `<port>` is set to the value of `$nprobe_port`.

With this plugin nProbe will emit AS information with exported flows using the information exported by the router via BGP. If the plugin is not active, nProbe will use information from GeolIP if configured.

This plugin defines the following information elements used to export not just the AS to which flows belong to, but also the whole AS path.

<code>%SRC_AS_PATH_1</code>	Src AS path position 1
<code>%SRC_AS_PATH_2</code>	Src AS path position 2
<code>%SRC_AS_PATH_3</code>	Src AS path position 3
<code>%SRC_AS_PATH_4</code>	Src AS path position 4
<code>%SRC_AS_PATH_5</code>	Src AS path position 5
<code>%SRC_AS_PATH_6</code>	Src AS path position 6
<code>%SRC_AS_PATH_7</code>	Src AS path position 7

%SRC_AS_PATH_8	Src AS path position 8
%SRC_AS_PATH_9	Src AS path position 9
%SRC_AS_PATH_10	Src AS path position 10
%DST_AS_PATH_1	Dest AS path position 1
%DST_AS_PATH_2	Dest AS path position 2
%DST_AS_PATH_3	Dest AS path position 3
%DST_AS_PATH_4	Dest AS path position 4
%DST_AS_PATH_5	Dest AS path position 5
%DST_AS_PATH_6	Dest AS path position 6
%DST_AS_PATH_7	Dest AS path position 7
%DST_AS_PATH_8	Dest AS path position 8
%DST_AS_PATH_9	Dest AS path position 9
%DST_AS_PATH_10	Dest AS path position 10

## 4.2. DNS Plugin

This plugin dissects DNS traffic and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%DNS_QUERY	DNS query
%DNS_QUERY_ID	DNS query transaction Id
%DNS_QUERY_TYPE	DNS query type (e.g. 1=A, 2=NS..)
%DNS_RET_CODE	DNS return code (e.g. 0=no error)
%DNS_NUM_ANSWERS	DNS # of returned answers

Using `--dns-dump-dir <dump dir>` it is possible to specify where the DNS dump files will be saved. Each file is up to 1000 lines long and when is completed a new file will be created.

## 4.3. GTPv0 Plugin

This plugin dissects GTPv0 signaling information (GTP-C) and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%GTPV0_REQ_MSG_TYPE	GTPv0 Request Msg Type
%GTPV0_RSP_MSG_TYPE	GTPv0 Response Msg Type
%GTPV0_TID	GTPv0 Tunnel Identifier
%GTPV0_APN_NAME	GTPv0 APN Name
%GTPV0_END_USER_IP	GTPv0 End User IP Address
%GTPV0_END_USER_MSISDN	GTPv0 End User MSISDN
%GTPV0_RAI_MCC	GTPv0 Mobile Country Code
%GTPV0_RAI_MNC	GTPv0 Mobile Network Code
%GTPV0_RAI_CELL_LAC	GTPv0 Cell Location Area Code
%GTPV0_RAI_CELL_RAC	GTPv0 Cell Routing Area Code
%GTPV0_RESPONSE_CAUSE	GTPv0 Cause of Operation

The plugin supports the following command line options that are used to specify where the (optional) GTP log file is saved. As previously described for `-P`, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--gtpv0-dump-dir <dump dir> Directory where GTP logs will be dumped
--gtpv0-exec-cmd <cmd> Command executed whenever a directory has been dumped
```

Please note that GTP-U is not handled by this plugin but rather by the nProbe core when the `--tunnel` option is used.

## 4.4. GTPv1 Plugin

This plugin dissects GTPv1 signaling information (GTP-C) and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%GTPV1_REQ_MSG_TYPE	GTPv1 Request Msg Type
%GTPV1_RSP_MSG_TYPE	GTPv1 Response Msg Type
%GTPV1_C2S_TEID_DATA	GTPv1 Client->Server TunnelId Data
%GTPV1_C2S_TEID_CTRL	GTPv1 Client->Server TunnelId Control
%GTPV1_S2C_TEID_DATA	GTPv1 Server->Client TunnelId Data
%GTPV1_S2C_TEID_CTRL	GTPv1 Server->Client TunnelId Control
%GTPV1_END_USER_IP	GTPv1 End User IP Address
%GTPV1_END_USER_IMSI	GTPv1 End User IMSI
%GTPV1_END_USER_MSISDN	GTPv1 End User MSISDN
%GTPV1_END_USER_IMEI	GTPv1 End User IMEI
%GTPV1_APN_NAME	GTPv1 APN Name
%GTPV1_RAI_MCC	GTPv1 RAI Mobile Country Code
%GTPV1_RAI_MNC	GTPv1 RAI Mobile Network Code
%GTPV1_RAI_LAC	GTPv1 RAI Location Area Code
%GTPV1_RAI_RAC	GTPv1 RAI Routing Area Code
%GTPV1_ULI_MCC	GTPv1 ULI Mobile Country Code
%GTPV1_ULI_MNC	GTPv1 ULI Mobile Network Code
%GTPV1_ULI_CELL_LAC	GTPv1 ULI Cell Location Area Code
%GTPV1_ULI_CELL_CI	GTPv1 ULI Cell CI
%GTPV1_ULI_SAC	GTPv1 ULI SAC
%GTPV1_RESPONSE_CAUSE	GTPv1 Cause of Operation

The plugin supports the following command line options that are used to specify where the (optional) GTP log file is saved. As previously described for -P, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--gtpv1-dump-dir <dump dir> Directory where GTP logs will be dumped
--gtpv1-exec-cmd <cmd> Command executed whenever a directory has been dumped
```

Please note that GTP-U is not handled by this plugin but rather by the nProbe core when the --tunnel option is used.

## 4.5. GTPv2 Plugin

This plugin dissects GTPv2 signaling information (GTP-C) and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%GTPV2_REQ_MSG_TYPE	GTPv2 Request Msg Type
%GTPV2_RSP_MSG_TYPE	GTPv2 Response Msg Type
%GTPV2_C2S_S1U_GTPU_TEID	GTPv2 Client->Svr S1U GTPU TEID
%GTPV2_C2S_S1U_GTPU_IP	GTPv2 Client->Svr S1U GTPU IP
%GTPV2_S2C_S1U_GTPU_TEID	GTPv2 Srv->Client S1U GTPU TEID
%GTPV2_S2C_S1U_GTPU_IP	GTPv2 Srv->Client S1U GTPU IP
%GTPV2_END_USER_IMSI	GTPv2 End User IMSI
%GTPV2_END_USER_MSISDN	GTPv2 End User MSISDN
%GTPV2_APN_NAME	GTPv2 APN Name
%GTPV2_ULI_MCC	GTPv2 Mobile Country Code
%GTPV2_ULI_MNC	GTPv2 Mobile Network Code
%GTPV2_ULI_CELL_TAC	GTPv2 Tracking Area Code
%GTPV2_ULI_CELL_ID	GTPv2 Cell Identifier
%GTPV2_RESPONSE_CAUSE	GTPv2 Cause of Operation

The plugin supports the following command line options that are used to specify where the (optional) GTP log file is saved. As previously described for `-P`, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--gtpv2-dump-dir <dump dir> Directory where GTP logs will be dumped
--gtpv2-exec-cmd <cmd>    Command executed whenever a directory has been dumped
```

Please note that GTP-U is not handled by this plugin but rather by the nProbe core when the `--tunnel` option is used.

## 4.6. HTTP Plugin

This plugin dissects HTTP traffic information (https can be decoded if the plugin is compiled with CyaSSL and the private SSL key is available and configured in the plugin) and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%HTTP_URL	HTTP URL
%HTTP_RET_CODE	HTTP return code (e.g. 200, 304...)
%HTTP_REFERER	HTTP Referer
%HTTP_UA	HTTP User Agent
%HTTP_MIME	HTTP Mime Type
%HTTP_HOST	HTTP Host Name
%HTTP_FBOOK_CHAT	HTTP Facebook Chat

The plugin supports the following command line options that are used to specify where the (optional) HTTP log file is saved. As previously described for `-P`, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--http-dump-dir <dump dir>  Directory where HTTP logs will be dumped
--ssl-config-file <path>    Configuration file for SSL certificate decoding
--ssl-debug                Enables ssl tracing (highly verbose)
--http-exec-cmd <cmd>      Command executed whenever a directory has been dumped
--dont-hash-cookies        Dump cookie string instead of cookie hash
--max-http-log-lines        Max number of lines per log file (default 10000)
--http-dump-timeout        After that timeout (in sec) the log file will be closed (default: 60 sec)
--http-ports                List of ports used for http protocol (default: 80)
--https-ports               List of ports used for https protocol (default: 443)
--proxy-ports               List of ports used for proxy protocol (default: 3128, 8080)
```

## 4.7. IMAP Plugin

This plugin dissects IMAP traffic information and saves it in dump files as well export the information via NetFlow/IPFIX using the following information element.

%IMAP_LOGIN	Mail sender
-------------	-------------

The plugin supports the following command line options that are used to specify where the (optional) log file is saved. As previously described for `-P`, dumps are nested in directories. It

is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--imap-dump-dir <dump dir>    Directory where IMAP logs will be dumped
--imap-exec-cmd <cmd>        Command executed whenever a directory has been dumped
--imap-peek-headers          Dump both emails body and headers (default: body only)
```

## 4.8. MySQL Plugin

This plugin dissects MySQL (unencrypted) traffic information and saves the queries log in dump files as well export the information via NetFlow/IPFIX using the following information elements.

```
%MYSQL_SERVER_VERSION    MySQL server version
%MYSQLE_USERNAME         MySQL username
%MYSQLE_DB               MySQL database in use
%MYSQLE_QUERY            MySQL Query
%MYSQLE_RESPONSE        MySQL server response
%MYSQLE_APPL_LATENCY_USEC MySQL request->response latency (usec)
```

The plugin supports the following command line options that are used to specify where the (optional) log file is saved. As previously described for -P, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--mysql-dump-dir <dump dir>    Directory where MySQL logs will be dumped
--mysql-exec-cmd <cmd>        Command executed whenever a directory has been dumped
--max-mysql-log-lines        Max number of lines per log file (default 10000)
```

## 4.9. Oracle Plugin

This plugin dissects Oracle (unencrypted) traffic information and saves the queries log in dump files as well export the information via NetFlow/IPFIX using the following information elements.

```
%ORACLE_USERNAME        Oracle Username
%ORACLE_QUERY            Oracle Query
%ORACLE_RSP_CODE         Oracle Response Code
%ORACLE_RSP_STRING       Oracle Response String
%ORACLE_QUERY_DURATION   Oracle Query Duration (msec)
```

The plugin supports the following command line options that are used to specify where the (optional) log file is saved. As previously described for -P, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--oracle-dump-dir <dump dir>    Directory where Oracle logs will be dumped
--oracle-exec-cmd <cmd>        Command executed whenever a directory has been dumped
--max-oracle-log-lines        Max number of lines per log file (default 10000)
```

Note that not all Oracle DB version might be supported by this plugin.

## 4.10. POP3 Plugin

This plugin dissects POP3 traffic information and saves it in dump files as well export the information via NetFlow/IPFIX using the following information element.

```
%POP_USER          POP3 user login
```

The plugin supports the following command line options that are used to specify where the (optional) log file is saved. As previously described for -P, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) if fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--pop-dump-dir <dump dir>  Directory where POP3 logs will be dumped
--pop-exec-cmd <cmd>       Command executed whenever a directory has been dumped
```

## 4.11. Radius Plugin

This plugin dissects Radius (unencrypted) traffic information and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

```
%RADIUS_REQ_MSG_TYPE    RADIUS Request Msg Type
%RADIUS_RSP_MSG_TYPE    RADIUS Response Msg Type
%RADIUS_USER_NAME       RADIUS User Name (Access Only)
%RADIUS_CALLING_STATION_ID  RADIUS Calling Station Id
%RADIUS_CALLED_STATION_ID  RADIUS Called Station Id
%RADIUS_NAS_IP_ADDR     RADIUS NAS IP Address
%RADIUS_NAS_IDENTIFIER   RADIUS NAS Identifier
%RADIUS_USER_IMSI       RADIUS User IMSI (Extension)
%RADIUS_USER_IMEI       RADIUS User MSISDN (Extension)
%RADIUS_FRAMED_IP_ADDR  RADIUS Framed IP
%RADIUS_ACCT_SESSION_ID  RADIUS Accounting Session Name
%RADIUS_ACCT_STATUS_TYPE RADIUS Accounting Status Type
%RADIUS_ACCT_IN_OCTETS   RADIUS Accounting Input Octets
%RADIUS_ACCT_OUT_OCTETS  RADIUS Accounting Output Octets
%RADIUS_ACCT_IN_PKTS    RADIUS Accounting Input Packets
%RADIUS_ACCT_OUT_PKTS   RADIUS Accounting Output Packets
```

The plugin supports the following command line options that are used to specify where the (optional) log file is saved. As previously described for -P, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) if fully dumped (i.e. nProbe has moved to the next directory in time order).

```
--radius-dump-dir <dump dir>  Directory where Radius logs will be dumped
--radius-exec-cmd <cmd>       Command executed whenever a directory has been dumped
```

Note that 3GPP radius extensions are supported by the plugin.

## 4.12. RTP Plugin

This plugin dissects RTP traffic information and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

```
%RTP_FIRST_SSRC        First flow RTP Sync Source ID
%RTP_FIRST_TS          First flow RTP timestamp
%RTP_LAST_SSRC         Last flow RTP Sync Source ID
%RTP_LAST_TS          Last flow RTP timestamp
%RTP_IN_JITTER         RTP Jitter (ms * 1000)
%RTP_OUT_JITTER        RTP Jitter (ms * 1000)
```

%RTP_IN_PKT_LOST	Packet lost in stream
%RTP_OUT_PKT_LOST	Packet lost in stream
%RTP_IN_PAYLOAD_TYPE	RTP payload type
%RTP_OUT_PAYLOAD_TYPE	RTP payload type
%RTP_IN_MAX_DELTA	Max delta (ms*100) between consecutive pkts
%RTP_OUT_MAX_DELTA	Max delta (ms*100) between consecutive pkts
%RTP_SIP_CALL_ID	SIP call-id corresponding to this RTP stream

### 4.13. SIP Plugin

This plugin dissects SIP traffic information and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%SIP_CALL_ID	SIP call-id
%SIP_CALLING_PARTY	SIP Call initiator
%SIP_CALLED_PARTY	SIP Called party
%SIP_RTP_CODECS	SIP RTP codecs
%SIP_INVITE_TIME	SIP SysUptime (msec) of INVITE
%SIP_TRYING_TIME	SIP SysUptime (msec) of Trying
%SIP_RINGING_TIME	SIP SysUptime (msec) of RINGING
%SIP_INVITE_OK_TIME	SIP SysUptime (msec) of INVITE OK
%SIP_INVITE_FAILURE_TIME	SIP SysUptime (msec) of INVITE FAILURE
%SIP_BYE_TIME	SIP SysUptime (msec) of BYE
%SIP_BYE_OK_TIME	SIP SysUptime (msec) of BYE OK
%SIP_CANCEL_TIME	SIP SysUptime (msec) of CANCEL
%SIP_CANCEL_OK_TIME	SIP SysUptime (msec) of CANCEL OK
%SIP_RTP_IPV4_SRC_ADDR	SIP RTP stream source IP
%SIP_RTP_L4_SRC_PORT	SIP RTP stream source port
%SIP_RTP_IPV4_DST_ADDR	SIP RTP stream dest IP
%SIP_RTP_L4_DST_PORT	SIP RTP stream dest port
%SIP_FAILURE_CODE	SIP failure response code
%SIP_REASON_CAUSE	SIP Cancel/Bye/Failure reason cause

### 4.14. SMTP Plugin

This plugin dissects IMAP traffic information and saves it in dump files as well export the information via NetFlow/IPFIX using the following information elements.

%SMTP_MAIL_FROM	Mail sender
%SMTP_RCPT_TO	Mail recipient

The plugin supports the following command line options that are used to specify where the (optional) log file is saved. As previously described for -P, dumps are nested in directories. It is possible to instruct nProbe to execute a command when a directory (not a log file) is fully dumped (i.e. nProbe has moved to the next directory in time order).

- smtp-dump-dir <dump dir> Directory where SMTP logs will be dumped
- smtp-exec-cmd <cmd> Command executed whenever a directory has been dumped

### 4.15. NetFlow-Lite Plugin

This plugin collects NetFlow-Lite flows and uses them as (simulated) packets as if they were received from a captured device. As the plugin acts as a collector for flows sent in NF-Lite format, you need to specify the listening port and an optional number of sequential ports to which flows will be sent. The more ports the more performance can be achieved.

- nflite <flow listen port low>[:<num ports>]> | Specify NetFlow-Lite listen port(s) (max 32)

## 5. Developing nProbe Plugins

Each nProbe plugin is implemented as shared library to be loaded at runtime by nProbe. The probe comes with several plugins that can be used as example for this activity. Below we list the main concepts you need to know if you plan to develop nProbe plugins.

Each plugin has to defined a plugin entry point as follows

```
static PluginEntryPoint dbPlugin = {
    NPROBE_REVISION,
    "My Plugin Name",
    "shortName", NULL,
    "version",
    "Plugin string description",
    "author email",
    0 /* always enabled */, 1, /* enabled */
    PLUGIN_DONT_NEED_LICENSE,
    myPlugin_init,
    NULL, /* Term */
    myPlugin_conf,
    myPlugin_delete,
    1, /* call packetFlowFctn for each packet */
    NULL /* myPlugin_packet */,
    myPlugin_get_template,
    myPlugin_export,
    myPlugin_print,
    NULL,
    NULL,
    myPlugin_help,
    NULL, 0, 0
};
```

and a function with the following format

```
#ifdef MAKE_STATIC_PLUGINS
PluginEntryPoint* myPluginEntryFctn(void)
#else
PluginEntryPoint* PluginEntryFctn(void)
#endif
{
    return(&myPlugin);
}
```

The fields of the PluginEntryPoint function have the following meaning:

- char \*nprobe\_revision;  
String to be defined as NPROBE\_REVISION.
- char \*name;  
Extended plugin name.
- char \*short\_name;  
Short plugin name.
- char \*family;  
Plugin family name (if any) or NULL to use the short plugin name.
- char \*version;  
Plugin version (e.g. 1.0)

- `char *descr;`  
Plugin description in plain English.
- `char*author;`  
Plugin author name and email.
- `u_int8_t always_enabled;`  
Set it to 1 to enable the plugin permanently regardless of its use in the template (-T command line option).
- `u_int8_t enabled;`  
Do not touch it and set it to 0 (used by nProbe).
- `u_int8_t need_license;`  
Set it to 1 if a license for this plugin is needed, or 0 if is not needed.
- `PluginInitFctn initFctn;`  
Plugin initialization function called when the plugin is loaded in memory. This function is called regardless of the fact that the plugin will later be used or not.
- `PluginTermFctn termFctn;`  
Plugin termination function called when the plugin is terminated during nProbe shutdown.
- `PluginConf pluginFlowConf;`  
Function that returns the flow configuration (see below).
- `PluginFctn deleteFlowFctn;`  
Flow callback that is called for flows handled by this plugin whenever a flow has been exported. This function is used to free memory of resources associated to the flow. Set it to NULL if no function will be defined,
- `u_int8_t call_packetFlowFctn_for_each_packet;`  
Set it to 1 to ask nProbe to call the `packetFlowFctn` callback for every packet belonging to this flow, or 0 for calling it only for the first flow packet.
- `PluginPacketFctn packetFlowFctn;`  
Callback called whenever nProbe has a packet belonging to the flow to be processed by the plugin.
- `PluginGetTemplateFctn getTemplateFctn;`  
Function used to return the template Element for the specified information element passed as parameter.
- `PluginExportFctn pluginExportFctn;`  
Callback called whenever the flow handled by this plugin is going to be exported.
- `PluginPrintFctn pluginPrintFctn;`  
Function that is called when `nprobe -P` is used, and that is supposed to print flow information into text files.
- `PluginStatsFctn pluginStatsFctn;`  
Function that is called (when not set to NULL) whenever nProbe prints periodic information (-b 1 or -b 2).
- `PluginSetupFctn setupFctn;`  
Function called after plugin initialization (when not set to NULL), if according to the specified template, this plugin will be used.
- `PluginHelpFctn helpFctn;`  
Function that is called when `nprobe -h` is executed, and that is supposed to print plugin information.
- `PluginIdleTaskFctn idleFctn;`  
If not set to NULL, this function will be periodically called by the nProbe core to execute (if any) housekeeping activities.

- `u_int8_t v4TemplateIdx, v6TemplateIdx;`

Used by nProbe. Set them to 0.

Each plugin must define a template with the following format

```
static V9V10TemplateElementId myPlugin_template[] = {
.....
  { 0, BOTH_IPV4_IPV6, FLOW_TEMPLATE, LONG_SNAPLEN, NTOP_ENTERPRISE_ID, 0, STATIC_FIELD_LEN, 0, 0, 0,
  NULL, NULL, NULL }
};
```

what will be then used by the following functions

```
static V9V10TemplateElementId* myPlugin_get_template(char* template_name) {
  int i;

  for(i=0; myPlugin_template[i].templateElementId != 0; i++) {
    if(!strcmp(template_name, myPlugin_template[i].netflowElementName)) {
      return(&myPlugin_template[i]);
    }
  }

  return(NULL); /* Unknown */
}

static V9V10TemplateElementId* myPlugin_conf(void) {
  return(myPlugin_template);
}
```

In the file `template.h` are specified the flow identifiers to be used in `V9V10TemplateElementId` that is defined as follows:

- `u_int8_t isInUse;`  
Always set it to 1, or 0 if it is the last template element to indicate that no further element will be defined.
- `u_int8_t protoMode;`  
Set it to `BOTH_IPV4_IPV6` or `ONLY_IPV4`, `ONLY_IPV6` if this element is for both IPv4 and IPv6 flows, just for IPv4 flows, or just for IPv6 flows.
- `const u_int8_t isOptionTemplate;`  
Set it to 0 if this is a flow template (default), or 1 if it used as option template.
- `const u_int8_t useLongSnaplen;`  
Set it to 1 if this plugin requires nProbe to capture packets with long snaplen that are needed when the plugin has to perform payload analysis.
- `const u_int32_t templateElementEnterpriseId;`  
Specify the IANA defined enterprise Id for this custom field. ntop uses `NTOP_ENTERPRISE_ID` for the proprietary ones.
- `const u_int16_t templateElementId;`  
Used by nProbe, leave it to 0.
- `u_int8_t variableFieldLength;`  
Set it to 1 to indicate that if nProbe exports flows in IPFIX format (-V 10) this field will have a variable field size.
- `u_int16_t templateElementLen;`  
Specify the static field size (-V 9) or max field size (-V 10)
- `const ElementFormat elementFormat;`  
Specify the format of the element. This information will be used when this data is

printed into MySQL. The supported format types are: `ascii_format`, `hex_format`, `numeric_format`, `ipv6_address_format`.

- `const ElementDumpFormat fileDumpFormat;`  
Specify the field format when the nProbe metadata information is printed (`--metadata`). The supported format types are: `dump_as_uint`, `dump_as_formatted_uint`, `dump_as_ip_port`, `dump_as_ip_proto`, `dump_as_ipv4_address`, `dump_as_ipv6_address`, `dump_as_mac_address`, `dump_as_epoch`, `dump_as_bool`, `dump_as_tcp_flags`, `dump_as_hex`, `dump_as_ascii`
- `const char *netflowElementName;`  
String with the symbolic network element name used in NetFlow (-V 9).
- `const char *ipfixElementName;`  
String with the symbolic network element name used in IPFIX (-V 10).
- `const char *templateElementDescr;`  
String that describes the element information type used by nProbe when the help (-h) is printed.

Most plugin callbacks are straightforward and its logic can be understood simply having a look at examples of existing plugins. The only function worth to describe is the one that processes packets as it is the most complex one.

```
static void myPlugin_packet(u_char new_bucket,
    int packet_if_idx /* -1 = unknown */,
    void *pluginData,
    FlowHashBucket* bkt,
    FlowDirection flow_direction,
    u_int16_t ip_offset, u_short proto,
    u_char isFragment,
    u_short numPkts, u_char tos,
    u_short vlanId, struct eth_header *ehdr,
    IpAddress *src, u_short sport,
    IpAddress *dst, u_short dport,
    u_int len, u_int8_t flags,
    u_int32_t tcpSeqNum, u_int8_t icmpType,
    u_short numMplsLabels,
    u_char mplsLabels[MAX_NUM_MPLS_LABELS (MPLS_LABEL_LEN)],
    const struct pcap_pkthdr *h, const u_char *p,
    u_char *payload, int payloadLen) {
    ...
}
```

This function processes a packet belonging to a flow handled by this plugin. nProbe has no clue what plugins are doing, this whenever a new flow is created (`new_bucket` is set to 1 for the first packet of the flow, or 0 for the following packets), it calls all active plugins to tell that a new flow is active in cache. The plugin will then decide if the packet can be handled by the plugin or not. This is done by looking at the packet header fields passed to the function, or inspecting the packet payload (payload point whose length is specified by `payloadLen`). If a plugin decides that the packet cannot be handled by the plugin (for instance because the packet protocol is not managed by the plugin) no action is needed and the function must simply return. Instead if the plugin can handle the packet, at the beginning of the function the following code-like must be specified in order to add the plugin to the list of plugins (it should usually be 1 or 0 element long) handling this flow.

```

if(new_bucket /* This bucket has been created recently */) {

    info->pluginPtr = (void*)&myPlugin;
    pluginData = info->pluginData = (struct my_plugin_info*)malloc(sizeof(struct my_plugin_info));

    if(info->pluginData == NULL) {
        traceEvent	TRACE_ERROR, "Not enough memory?");
        free(info);
        return; /* Not enough memory */
    } else {
        struct my_plugin_info *myinfo = (struct my_plugin_info*)pluginData;

        /* Reset fields */
        memset(myinfo, 0, sizeof(struct my_plugin_info));

        info->next = bkt->ext->plugin;
        info->plugin_used = 0;
        bkt->ext->plugin = info;
    }
}
}
}

```

Once a plugin is defined, it must be placed into the nProbe/plugins directory so that the nProbe build process will detect and compile it.

## 6. References

1. Introduction to Cisco NetFlow, [http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps\\_wp.htm](http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm)
2. ntop, <http://www.ntop.org/>
3. nProbe, <http://www.ntop.org/nprobe.html>
4. nBox, <http://www.nmon.net/nBox.html>
5. Linux Debian, <http://www.debian.org/>
6. tcpdump, <http://www.tcpdump.org/>
7. Extreme Happy Netflow Tool, <http://ehnt.sourceforge.net/>
8. Libpcap, <http://www.tcpdump.org/>
9. Winpcap, <http://winpcap.polito.it/>
10. PC Engines, <http://www.pcengines.ch/>
11. SQLite, <http://www.sqlite.org>
12. IPerf, <http://dast.nlanr.net/Projects/lperf>

### 6.1. Credits

- NetFlow is a trademark of Cisco Systems.
- Windows is a trademark of Microsoft Corporation.

## 7. Appendix A: BPF Packet Filtering Expressions

This section has been extracted from the `tcpdump` man page and it describes the syntax of BPF filters you can specify using the `-f` flag.

The expression consists of one or more primitives. Primitives usually consist of an id (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

### type

qualifiers say what kind of thing the id name or number refers to. Possible types are `host`, `net` and `port`. E.g., `host foo`, `net 128.3`, `port 20`. If there is no type qualifier, `host` is assumed.

### dir

qualifiers specify a particular transfer direction to and/or from id. Possible directions are `src`, `dst`, `src or dst` and `src and dst`. E.g., `src foo`, `dst net 128.3`, `src or dst port ftp-data`. If there is no `dir` qualifier, `src` or `dst` is assumed.

### proto

qualifiers restrict the match to a particular protocol. Possible protos are: `ether`, `fdi`, `ip`, `arp`, `rarp`, `decnet`, `lat`, `moprc`, `mopdl`, `tcp` and `udp`. E.g., `ether src foo`, `arp net 128.3`, `tcp port 21`. If there is no `proto` qualifier, all protocols consistent with the type are assumed. E.g., `src foo` means `(ip or arp or rarp) src foo` (except the latter is not legal syntax), `net bar` means `(ip or arp or rarp) net bar` and `port 53` means `(tcp or udp) port 53`.

[`fdi` is actually an alias for `ether`; the parser treats them identically as meaning "the data link level used on the specified network interface." FDDI headers contain Ethernet-like source and destination addresses, and often contain Ethernet-like packet types, so you can filter on these FDDI fields just as with the analogous Ethernet fields. FDDI headers also contain other fields, but you cannot name them explicitly in a filter expression.]

In addition to the above, there are some special 'primitive' keywords that don't follow the pattern: `gateway`, `broadcast`, `less`, `greater` and arithmetic expressions. All of these are described below.

More complex filter expressions are built up by using the words `and`, `or` and `not` to combine primitives. E.g., `host foo and not port ftp and not port ftp-data`. To save typing, identical qualifier lists can be omitted. E.g., `tcp dst port ftp or ftp-data or domain` is exactly the same as `tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain`.

Allowable primitives are:

### dst host host

True if the IP destination field of the packet is `host`, which may be either an address or a name.

### src host host

True if the IP source field of the packet is `host`.

**host host**

True if either the IP source or destination of the packet is host. Any of the above host expressions can be prepended with the keywords, ip, arp, or rarp as in: ip host host

which is equivalent to: ether proto \ip and host host

If host is a name with multiple IP addresses, each address will be checked for a match.

**ether dst ehost**

True if the ethernet destination address is ehost. Ehost may be either a name from /etc/ethers or a number.

**ether src ehost**

True if the ethernet source address is ehost.

**ether host ehost**

True if either the ethernet source or destination address is ehost.

**gateway host**

True if the packet used host as a gateway. I.e., the ethernet source or destination address was host but neither the IP source nor the IP destination was host. Host must be a name and must be found in both /etc/hosts and /etc/ethers. (An equivalent expression is ether host ehost and not host host

which can be used with either names or numbers for host / ehost.)

**dst net net**

True if the IP destination address of the packet has a network number of net, which may be either an address or a name.

**src net net**

True if the IP source address of the packet has a network number of net.

**net net**

True if either the IP source or destination address of the packet has a network number of net.

**dst port port**

True if the packet is ip/tcp or ip/udp and has a destination port value of port. The port can be a number or a name used in /etc/services. If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (e.g., dst port 513 will print both tcp/login traffic and udp/who traffic, and port domain will print both tcp/domain and udp/domain traffic).

**src port port**

True if the packet has a source port value of port.

**port port**

True if either the source or destination port of the packet is port. Any of the above port expressions can be prepended with the keywords, tcp or udp, as in: tcp src port port which matches only tcp packets.

less length

True if the packet has a length less than or equal to length. This is equivalent to: len <= length.

greater length

True if the packet has a length greater than or equal to length. This is equivalent to: len >= length.

ip proto protocol

True if the packet is an ip packet of protocol type protocol. Protocol can be a number or one of the names icmp, udp, nd, or tcp. Note that the identifiers tcp, udp, and icmp are also keywords and must be escaped via backslash (\), which is \\ in the C-shell.

ether broadcast

True if the packet is an ethernet broadcast packet. The ether keyword is optional.

ip broadcast

True if the packet is an IP broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the local subnet mask.

ether multicast

True if the packet is an ethernet multicast packet. The ether keyword is optional. This is shorthand for ether[0] & 1 != 0'.

ip multicast

True if the packet is an IP multicast packet.

ether proto protocol

True if the packet is of ether type protocol. Protocol can be a number or a name like ip, arp, or rarp. Note these identifiers are also keywords and must be escaped via backslash (\). [In the case of FDDI (e.g., fddi protocol arp'), the protocol identification comes from the 802.2 Logical Link Control (LLC) header, which is usually layered on top of the FDDI header. ntop assumes, when filtering on the protocol identifier, that all FDDI packets include an LLC header, and that the LLC header is in so-called SNAP format.]

decnet src host

True if the DECNET source address is host, which may be an address of the form "0.123", or a DECNET host name. [DECNET host name support is only available on Ultrix systems that are configured to run DECNET.]

decnet dst host

True if the DECNET destination address is host.

dechnet host host

True if either the DECNET source or destination address is host.

ip, arp, rarp, dechnet

Abbreviations for: ether proto p where p is one of the above protocols.

lat, moprc, mopdl

Abbreviations for: ether proto p where p is one of the above protocols. Note that ntop does not currently know how to parse these protocols.

tcp, udp, icmp

Abbreviations for: ip proto p where p is one of the above protocols.

expr relop expr

True if the relation holds, where relop is one of >, <, >=, <=, =, !=, and expr is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+ , - , \* , / , & , |], a length operator, and special packet data accessors. To access data inside the packet, use the following syntax: proto [ expr : size ] Proto is one of ether, fddi, ip, arp, rarp, tcp, udp, or icmp, and indicates the protocol layer for the index operation. The byte offset, relative to the indicated protocol layer, is given by expr. Size is optional and indicates the number of bytes in the field of interest; it can be either one, two, or four, and defaults to one. The length operator, indicated by the keyword len, gives the length of the packet.

For example, ``ether[0] & 1 != 0`` catches all multicast traffic. The expression ``ip[0] & 0xf != 5`` catches all IP packets with options. The expression ``ip[6:2] & 0x1fff = 0`` catches only unfragmented datagrams and frag zero of fragmented datagrams. This check is implicitly applied to the tcp and udp index operations. For instance, `tcp[0]` always means the first byte of the TCP header, and never means the first byte of an intervening fragment.

Primitives may be combined using:

- A parenthesized group of primitives and operators
- (parentheses are special to the Shell and must be escaped).
- Negation (`!`` or ``not``).
- Concatenation (`&&`` or ``and``).
- Alternation (`|`` or ``or``).

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit and tokens, not juxtaposition, are now required for concatenation. If an identifier is given without a keyword, the most recent keyword is assumed. For example, `not host vs and ace` is short for `not host vs and host ace` which should not be confused with `not ( host vs or ace )`. Expression arguments can be passed to nProbe as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

## 7.1. Examples

To select all packets arriving at or departing from sundown:

`nprobe -f "host sundown"`

To select traffic between helios and either hot or ace:

`nprobe -f "host helios and ( hot or ace )"`

To select all IP packets between ace and any host except helios:

`nprobe -f "ip host ace and not helios"`

To select all traffic between local hosts and hosts at Berkeley:

`nprobe -f "net ucb-ether"`

To select all ftp traffic through internet gateway snup: (note that the expression is quoted to prevent the shell from (mis-)interpreting the parentheses):

`nprobe -f "gateway snup and (port ftp or ftp-data)"`

To select traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this stuff should never make it onto your local net).

`nprobe -f " ip and not net localnet"`

To select the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host.

`nprobe -f "tcp[13] & 3 != 0 and not src and dst net localnet"`

To select IP packets longer than 576 bytes sent through gateway snup:

`nprobe -f "gateway snup and ip[2:2] > 576"`

To select IP broadcast or multicast packets that were not sent via ethernet broadcast or multicast:

`nprobe -f "ether[0] & 1 = 0 and ip[16] >= 224"`

To select all ICMP packets that are not echo requests/replies (i.e., not ping packets):

`nprobe -f "icmp[0] != 8 and icmp[0] != 0"`

## 8. Appendix B: Flow Information Elements

The `-T` flag enabled users to specify the format of NetFlow v9/IPFIX flows. The format options currently supported by nProbe are those specified in the NetFlow v9 RFC, namely (in square brackets it is specified the field Id as defined in the RFC). As nProbe can be extended by means of plugins, further information elements can be defined based on plugin presence.

[ 1]	%IN_BYTES	%octetDeltaCount	Incoming flow bytes (src->dst)
[ 2]	%IN_PKTS	%packetDeltaCount	Incoming flow packets (src->dst)
[ 3]	%FLOWS		Number of flows
[ 4]	%PROTOCOL	%protocolIdentifier	IP protocol byte
[58500]	%PROTOCOL_MAP		IP protocol name
[ 5]	%SRC_TOS	%ipClassOfService	Type of service byte
[ 6]	%TCP_FLAGS	%tcpControlBits	Cumulative of all flow TCP flags
[ 7]	%L4_SRC_PORT	%sourceTransportPort	IPv4 source port
[58503]	%L4_SRC_PORT_MAP		Layer 4 source port symbolic name
[ 8]	%IPV4_SRC_ADDR	%sourceIPv4Address	IPv4 source address
[ 9]	%IPV4_SRC_MASK	%sourceIPv4PrefixLength	IPv4 source subnet mask (/<bits>)
[10]	%INPUT_SNMP	%ingressInterface	Input interface SNMP idx
[11]	%L4_DST_PORT		IPv4 destination port
[58507]	%L4_DST_PORT_MAP		Layer 4 destination port symbolic name
[58508]	%L4_SRV_PORT		Layer 4 server port
[58509]	%L4_SRV_PORT_MAP		Layer 4 server port symbolic name
[12]	%IPV4_DST_ADDR	%destinationIPv4Address	IPv4 destination address
[13]	%IPV4_DST_MASK	%destinationIPv4PrefixLength	IPv4 dest subnet mask (/<bits>)
[14]	%OUTPUT_SNMP	%egressInterface	Output interface SNMP idx
[15]	%IPV4_NEXT_HOP	%ipNextHopIPv4Address	IPv4 next hop address
[16]	%SRC_AS	%bgpSourceAsNumber	Source BGP AS
[17]	%DST_AS	%bgpDestinationAsNumber	Destination BGP AS
[21]	%LAST_SWITCHED	%flowEndSysUpTime	SysUptime (msec) of the last flow pkt
[22]	%FIRST_SWITCHED	%flowStartSysUpTime	SysUptime (msec) of the first flow pkt
[23]	%OUT_BYTES	%postOctetDeltaCount	Outgoing flow bytes (dst->src)
[24]	%OUT_PKTS	%postPacketDeltaCount	Outgoing flow packets (dst->src)
[27]	%IPV6_SRC_ADDR	%sourceIPv6Address	IPv6 source address
[28]	%IPV6_DST_ADDR	%destinationIPv6Address	IPv6 destination address
[29]	%IPV6_SRC_MASK	%sourceIPv6PrefixLength	IPv6 source mask
[30]	%IPV6_DST_MASK	%destinationIPv6PrefixLength	IPv6 destination mask
[32]	%ICMP_TYPE	%icmpTypeCodeIPv4	ICMP Type * 256 + ICMP code
[34]	%SAMPLING_INTERVAL		Sampling rate
[35]	%SAMPLING_ALGORITHM		Sampling type (deterministic/random)
[36]	%FLOW_ACTIVE_TIMEOUT	%flowActiveTimeout	Activity timeout of flow cache entries
[37]	%FLOW_INACTIVE_TIMEOUT	%flowIdleTimeout	Inactivity timeout of flow cache entries
[38]	%ENGINE_TYPE		Flow switching engine
[39]	%ENGINE_ID		Id of the flow switching engine
[40]	%TOTAL_BYTES_EXP	%exportedOctetTotalCount	Total bytes exported
[41]	%TOTAL_PKTS_EXP	%exportedMessageTotalCount	Total flow packets exported
[42]	%TOTAL_FLOWS_EXP	%exportedFlowRecordTotalCount	Total number of exported flows
[52]	%MIN_TTL	%minimumTTL	Min flow TTL
[53]	%MAX_TTL	%maximumTTL	Max flow TTL
[56]	%IN_SRC_MAC	%sourceMacAddress	Source MAC Address
[58]	%SRC_VLAN	%vlanId	Source VLAN
[59]	%DST_VLAN	%postVlanId	Destination VLAN
[60]	%IP_PROTOCOL_VERSION	%ipVersion	[4=IPv4][6=IPv6]
[61]	%DIRECTION	%flowDirection	It indicates where a sample has been taken (always 0)
[62]	%IPV6_NEXT_HOP	%ipNextHopIPv6Address	IPv6 next hop address
[70]	%MPLS_LABEL_1	%mplsTopLabelStackSection	MPLS label at position 1
[71]	%MPLS_LABEL_2	%mplsLabelStackSection2	MPLS label at position 2
[72]	%MPLS_LABEL_3	%mplsLabelStackSection3	MPLS label at position 3
[73]	%MPLS_LABEL_4	%mplsLabelStackSection4	MPLS label at position 4
[74]	%MPLS_LABEL_5	%mplsLabelStackSection5	MPLS label at position 5
[75]	%MPLS_LABEL_6	%mplsLabelStackSection6	MPLS label at position 6
[76]	%MPLS_LABEL_7	%mplsLabelStackSection7	MPLS label at position 7
[77]	%MPLS_LABEL_8	%mplsLabelStackSection8	MPLS label at position 8
[78]	%MPLS_LABEL_9	%mplsLabelStackSection9	MPLS label at position 9
[79]	%MPLS_LABEL_10	%mplsLabelStackSection10	MPLS label at position 10
[80]	%OUT_DST_MAC	%destinationMacAddress	Destination MAC Address
[95]	%APPLICATION_ID	%application_id	Cisco NBAR Application Id
[102]	%PACKET_SECTION_OFFSET		Packet section offset
[103]	%SAMPLED_PACKET_SIZE		Sampled packet size
[104]	%SAMPLED_PACKET_ID		Sampled packet id
[130]	%EXPORTER_IPV4_ADDRESS	%exporterIPv4Address	Exporter IPv4 Address
[131]	%EXPORTER_IPV6_ADDRESS	%exporterIPv6Address	Exporter IPv6 Address
[148]	%FLOW_ID	%flowId	Serial Flow Identifier
[150]	%FLOW_START_SEC	%flowStartSeconds	Seconds (epoch) of the first flow packet
[151]	%FLOW_END_SEC	%flowEndSeconds	Seconds (epoch) of the last flow packet
[152]	%FLOW_START_MILLISECONDS	%flowStartMilliseconds	Msec (epoch) of the first flow packet
[153]	%FLOW_END_MILLISECONDS	%flowEndMilliseconds	Msec (epoch) of the last flow packet
[239]	%BIFLOW_DIRECTION	%biflow_direction	1=initiator, 2=reverseInitiator
[277]	%OBSERVATION_POINT_TYPE		Observation point type
[300]	%OBSERVATION_POINT_ID		Observation point id
[302]	%SELECTOR_ID		Selector id
[304]	%IPFIX_SAMPLING_ALGORITHM		Sampling algorithm
[309]	%SAMPLING_SIZE		Number of packets to sample
[310]	%SAMPLING_POPULATION		Sampling population
[312]	%FRAME_LENGTH		Original L2 frame length
[318]	%PACKETS_OBSERVED		Tot number of packets seen
[319]	%PACKETS_SELECTED		Number of pkts selected for sampling
[335]	%SELECTOR_NAME		Sampler name

[NFv9 57552] [IPFIX 35632.80] %FRAGMENTS	Number of fragmented flow packets
[NFv9 57554] [IPFIX 35632.82] %CLIENT_NW_DELAY_SEC	Network latency client <=> nprobe (sec) [deprecated]
[NFv9 57555] [IPFIX 35632.83] %CLIENT_NW_DELAY_USEC	Network latency client <=> nprobe (residual usec)
[NFv9 57595] [IPFIX 35632.123] %CLIENT_NW_DELAY_MS	Network latency client <=> nprobe (msec)
[NFv9 57556] [IPFIX 35632.84] %SERVER_NW_DELAY_SEC	Network latency nprobe <=> server (sec) [deprecated]
[NFv9 57557] [IPFIX 35632.85] %SERVER_NW_DELAY_USEC	Network latency nprobe <=> server (residual usec)
[NFv9 57596] [IPFIX 35632.124] %SERVER_NW_DELAY_MS	Network latency nprobe <=> server (residual msec)
[NFv9 57558] [IPFIX 35632.86] %APPL_LATENCY_SEC	Application latency (sec) [deprecated]
[NFv9 57559] [IPFIX 35632.87] %APPL_LATENCY_USEC	Application latency (residual usec) [deprecated]
[NFv9 57597] [IPFIX 35632.125] %APPL_LATENCY_MS	Application latency (msec)
[NFv9 57560] [IPFIX 35632.88] %NUM_PKTS_UP_TO_128_BYTES	# packets whose size <= 128
[NFv9 57561] [IPFIX 35632.89] %NUM_PKTS_128_TO_256_BYTES	# packets whose size > 128 and <= 256
[NFv9 57562] [IPFIX 35632.90] %NUM_PKTS_256_TO_512_BYTES	# packets whose size > 256 and < 512
[NFv9 57563] [IPFIX 35632.91] %NUM_PKTS_512_TO_1024_BYTES	# packets whose size > 512 and < 1024
[NFv9 57564] [IPFIX 35632.92] %NUM_PKTS_1024_TO_1514_BYTES	# packets whose size > 1024 and <= 1514
[NFv9 57565] [IPFIX 35632.93] %NUM_PKTS_OVER_1514_BYTES	# packets whose size > 1514
[NFv9 57570] [IPFIX 35632.98] %CUMULATIVE_ICMP_TYPE	Cumulative OR of ICMP type packets
[NFv9 57573] [IPFIX 35632.101] %SRC_IP_COUNTRY	Country where the src IP is located
[NFv9 57574] [IPFIX 35632.102] %SRC_IP_CITY	City where the src IP is located
[NFv9 57575] [IPFIX 35632.103] %DST_IP_COUNTRY	Country where the dst IP is located
[NFv9 57576] [IPFIX 35632.104] %DST_IP_CITY	City where the dst IP is located
[NFv9 57577] [IPFIX 35632.105] %FLOW_PROTO_PORT	L7 port that identifies the flow protocol or 0 if unknown
[NFv9 57578] [IPFIX 35632.106] %UPSTREAM_TUNNEL_ID	Upstream tunnel identifier (e.g. GTP TEID) or 0 if unknown
[NFv9 57579] [IPFIX 35632.107] %LONGEST_FLOW_PKT	Longest packet (bytes) of the flow
[NFv9 57580] [IPFIX 35632.108] %SHORTEST_FLOW_PKT	Shortest packet (bytes) of the flow
[NFv9 57581] [IPFIX 35632.109] %RETRANSMITTED_IN_PKTS	Number of retransmitted TCP flow packets (src->dst)
[NFv9 57582] [IPFIX 35632.110] %RETRANSMITTED_OUT_PKTS	Number of retransmitted TCP flow packets (dst->src)
[NFv9 57583] [IPFIX 35632.111] %OORDER_IN_PKTS	Number of out of order TCP flow packets (dst->src)
[NFv9 57584] [IPFIX 35632.112] %OORDER_OUT_PKTS	Number of out of order TCP flow packets (src->dst)
[NFv9 57585] [IPFIX 35632.113] %UNTUNNELED_PROTOCOL	Untunneled IP protocol byte
[NFv9 57586] [IPFIX 35632.114] %UNTUNNELED_IPV4_SRC_ADDR	Untunneled IPv4 source address
[NFv9 57587] [IPFIX 35632.115] %UNTUNNELED_L4_SRC_PORT	Untunneled IPv4 source port
[NFv9 57588] [IPFIX 35632.116] %UNTUNNELED_IPV4_DST_ADDR	Untunneled IPv4 destination address
[NFv9 57589] [IPFIX 35632.117] %UNTUNNELED_L4_DST_PORT	Untunneled IPv4 destination port
[NFv9 57590] [IPFIX 35632.118] %L7_PROTO	Layer 7 protocol (numeric)
[NFv9 57591] [IPFIX 35632.119] %L7_PROTO_NAME	Layer 7 protocol name
[NFv9 57592] [IPFIX 35632.120] %DOWNSTREAM_TUNNEL_ID	Downstream tunnel identifier (e.g. GTP TEID) or 0 if unknown
[NFv9 57593] [IPFIX 35632.121] %FLOW_USER_NAME	Flow username of the tunnel (if known)
[NFv9 57594] [IPFIX 35632.122] %FLOW_SERVER_NAME	Flow server name (if known)
[NFv9 57598] [IPFIX 35632.126] %PLUGIN_NAME	Plugin name used by this flow (if any)
[NFv9 57819] [IPFIX 35632.347] %NUM_PKTS_TTL_EQ_1	# packets with TTL = 1
[NFv9 57818] [IPFIX 35632.346] %NUM_PKTS_TTL_2_5	# packets with TTL > 1 and TTL <= 5
[NFv9 57806] [IPFIX 35632.334] %NUM_PKTS_TTL_5_32	# packets with TTL > 5 and TTL <= 32
[NFv9 57807] [IPFIX 35632.335] %NUM_PKTS_TTL_32_64	# packets with TTL > 32 and <= 64
[NFv9 57808] [IPFIX 35632.336] %NUM_PKTS_TTL_64_96	# packets with TTL > 64 and <= 96
[NFv9 57809] [IPFIX 35632.337] %NUM_PKTS_TTL_96_128	# packets with TTL > 96 and <= 128
[NFv9 57810] [IPFIX 35632.338] %NUM_PKTS_TTL_128_160	# packets with TTL > 128 and <= 160
[NFv9 57811] [IPFIX 35632.339] %NUM_PKTS_TTL_160_192	# packets with TTL > 160 and <= 192
[NFv9 57812] [IPFIX 35632.340] %NUM_PKTS_TTL_192_224	# packets with TTL > 192 and <= 224
[NFv9 57813] [IPFIX 35632.341] %NUM_PKTS_TTL_224_255	# packets with TTL > 224 and <= 255
[NFv9 57821] [IPFIX 35632.349] %IN_SRC_OSI_SAP	OSI Source SAP (OSI Traffic Only)
[NFv9 57822] [IPFIX 35632.350] %OUT_DST_OSI_SAP	OSI Destination SAP (OSI Traffic Only)
Plugin BGP Update Listener templates:	
[NFv9 57762] [IPFIX 35632.290] %SRC_AS_PATH_1	Src AS path position 1
[NFv9 57763] [IPFIX 35632.291] %SRC_AS_PATH_2	Src AS path position 2
[NFv9 57764] [IPFIX 35632.292] %SRC_AS_PATH_3	Src AS path position 3
[NFv9 57765] [IPFIX 35632.293] %SRC_AS_PATH_4	Src AS path position 4
[NFv9 57766] [IPFIX 35632.294] %SRC_AS_PATH_5	Src AS path position 5
[NFv9 57767] [IPFIX 35632.295] %SRC_AS_PATH_6	Src AS path position 6
[NFv9 57768] [IPFIX 35632.296] %SRC_AS_PATH_7	Src AS path position 7
[NFv9 57769] [IPFIX 35632.297] %SRC_AS_PATH_8	Src AS path position 8
[NFv9 57770] [IPFIX 35632.298] %SRC_AS_PATH_9	Src AS path position 9
[NFv9 57771] [IPFIX 35632.299] %SRC_AS_PATH_10	Src AS path position 10
[NFv9 57772] [IPFIX 35632.300] %DST_AS_PATH_1	Dest AS path position 1
[NFv9 57773] [IPFIX 35632.301] %DST_AS_PATH_2	Dest AS path position 2
[NFv9 57774] [IPFIX 35632.302] %DST_AS_PATH_3	Dest AS path position 3
[NFv9 57775] [IPFIX 35632.303] %DST_AS_PATH_4	Dest AS path position 4
[NFv9 57776] [IPFIX 35632.304] %DST_AS_PATH_5	Dest AS path position 5
[NFv9 57777] [IPFIX 35632.305] %DST_AS_PATH_6	Dest AS path position 6
[NFv9 57778] [IPFIX 35632.306] %DST_AS_PATH_7	Dest AS path position 7
[NFv9 57779] [IPFIX 35632.307] %DST_AS_PATH_8	Dest AS path position 8
[NFv9 57780] [IPFIX 35632.308] %DST_AS_PATH_9	Dest AS path position 9
[NFv9 57781] [IPFIX 35632.309] %DST_AS_PATH_10	Dest AS path position 10
Plugin DHCP Protocol templates:	
[NFv9 57825] [IPFIX 35632.353] %DHCP_CLIENT_MAC	MAC of the DHCP client
[NFv9 57826] [IPFIX 35632.354] %DHCP_CLIENT_IP	DHCP assigned client IPv4 address
[NFv9 57827] [IPFIX 35632.355] %DHCP_CLIENT_NAME	DHCP client name
Plugin DNS Protocol templates:	
[NFv9 57677] [IPFIX 35632.205] %DNS_QUERY	DNS query
[NFv9 57678] [IPFIX 35632.206] %DNS_QUERY_ID	DNS query transaction Id
[NFv9 57679] [IPFIX 35632.207] %DNS_QUERY_TYPE	DNS query type (e.g. 1=A, 2=NS..)
[NFv9 57680] [IPFIX 35632.208] %DNS_RET_CODE	DNS return code (e.g. 0=no error)
[NFv9 57681] [IPFIX 35632.209] %DNS_NUM_ANSWERS	DNS # of returned answers
[NFv9 57824] [IPFIX 35632.352] %DNS_TTL_ANSWER	TTL of the first A record (if any)
Plugin FTP Protocol templates:	
[NFv9 57828] [IPFIX 35632.356] %FTP_LOGIN	FTP client login
[NFv9 57829] [IPFIX 35632.357] %FTP_PASSWORD	FTP client password
[NFv9 57830] [IPFIX 35632.358] %FTP_COMMAND	FTP client command
[NFv9 57831] [IPFIX 35632.359] %FTP_COMMAND_RET_CODE	FTP client command return code

## Plugin GTPv0 Signaling Protocol templates:

[NFv9 57793]	[IPFIX 35632.321]	%GTPV0_REQ_MSG_TYPE	GTPv0 Request Msg Type
[NFv9 57794]	[IPFIX 35632.322]	%GTPV0_RSP_MSG_TYPE	GTPv0 Response Msg Type
[NFv9 57795]	[IPFIX 35632.323]	%GTPV0_TID	GTPv0 Tunnel Identifier
[NFv9 57798]	[IPFIX 35632.326]	%GTPV0_APN_NAME	GTPv0 APN Name
[NFv9 57796]	[IPFIX 35632.324]	%GTPV0_END_USER_IP	GTPv0 End User IP Address
[NFv9 57797]	[IPFIX 35632.325]	%GTPV0_END_USER_MSISDN	GTPv0 End User MSISDN
[NFv9 57799]	[IPFIX 35632.327]	%GTPV0_RAI_MCC	GTPv0 Mobile Country Code
[NFv9 57800]	[IPFIX 35632.328]	%GTPV0_RAI_MNC	GTPv0 Mobile Network Code
[NFv9 57801]	[IPFIX 35632.329]	%GTPV0_RAI_CELL_LAC	GTPv0 Cell Location Area Code
[NFv9 57802]	[IPFIX 35632.330]	%GTPV0_RAI_CELL_RAC	GTPv0 Cell Routing Area Code
[NFv9 57803]	[IPFIX 35632.331]	%GTPV0_RESPONSE_CAUSE	GTPv0 Cause of Operation

## Plugin GTPv1 Signaling Protocol templates:

[NFv9 57692]	[IPFIX 35632.220]	%GTPV1_REQ_MSG_TYPE	GTPv1 Request Msg Type
[NFv9 57693]	[IPFIX 35632.221]	%GTPV1_RSP_MSG_TYPE	GTPv1 Response Msg Type
[NFv9 57694]	[IPFIX 35632.222]	%GTPV1_C2S_TEID_DATA	GTPv1 Client->Server TunnelId Data
[NFv9 57695]	[IPFIX 35632.223]	%GTPV1_C2S_TEID_CTRL	GTPv1 Client->Server TunnelId Control
[NFv9 57696]	[IPFIX 35632.224]	%GTPV1_S2C_TEID_DATA	GTPv1 Server->Client TunnelId Data
[NFv9 57697]	[IPFIX 35632.225]	%GTPV1_S2C_TEID_CTRL	GTPv1 Server->Client TunnelId Control
[NFv9 57698]	[IPFIX 35632.226]	%GTPV1_END_USER_IP	GTPv1 End User IP Address
[NFv9 57699]	[IPFIX 35632.227]	%GTPV1_END_USER_IMSI	GTPv1 End User IMSI
[NFv9 57700]	[IPFIX 35632.228]	%GTPV1_END_USER_MSISDN	GTPv1 End User MSISDN
[NFv9 57701]	[IPFIX 35632.229]	%GTPV1_END_USER_IMEI	GTPv1 End User IMEI
[NFv9 57702]	[IPFIX 35632.230]	%GTPV1_APN_NAME	GTPv1 APN Name
[NFv9 57703]	[IPFIX 35632.231]	%GTPV1_RAI_MCC	GTPv1 RAI Mobile Country Code
[NFv9 57704]	[IPFIX 35632.232]	%GTPV1_RAI_MNC	GTPv1 RAI Mobile Network Code
[NFv9 57814]	[IPFIX 35632.342]	%GTPV1_RAI_LAC	GTPv1 RAI Location Area Code
[NFv9 57815]	[IPFIX 35632.343]	%GTPV1_RAI_RAC	GTPv1 RAI Routing Area Code
[NFv9 57816]	[IPFIX 35632.344]	%GTPV1_ULI_MCC	GTPv1 ULI Mobile Country Code
[NFv9 57817]	[IPFIX 35632.345]	%GTPV1_ULI_MNC	GTPv1 ULI Mobile Network Code
[NFv9 57705]	[IPFIX 35632.233]	%GTPV1_ULI_CELL_LAC	GTPv1 ULI Cell Location Area Code
[NFv9 57706]	[IPFIX 35632.234]	%GTPV1_ULI_CELL_CI	GTPv1 ULI Cell CI
[NFv9 57707]	[IPFIX 35632.235]	%GTPV1_ULI_SAC	GTPv1 ULI SAC
[NFv9 57804]	[IPFIX 35632.332]	%GTPV1_RESPONSE_CAUSE	GTPv1 Cause of Operation

## Plugin GTPv2 Signaling Protocol templates:

[NFv9 57742]	[IPFIX 35632.270]	%GTPV2_REQ_MSG_TYPE	GTPv2 Request Msg Type
[NFv9 57743]	[IPFIX 35632.271]	%GTPV2_RSP_MSG_TYPE	GTPv2 Response Msg Type
[NFv9 57744]	[IPFIX 35632.272]	%GTPV2_C2S_S1U_GTPU_TEID	GTPv2 Client->Svr S1U GTPU TEID
[NFv9 57745]	[IPFIX 35632.273]	%GTPV2_C2S_S1U_GTPU_IP	GTPv2 Client->Svr S1U GTPU IP
[NFv9 57746]	[IPFIX 35632.274]	%GTPV2_S2C_S1U_GTPU_TEID	GTPv2 Srv->Client S1U GTPU TEID
[NFv9 57747]	[IPFIX 35632.275]	%GTPV2_S2C_S1U_GTPU_IP	GTPv2 Srv->Client S1U GTPU IP
[NFv9 57748]	[IPFIX 35632.276]	%GTPV2_END_USER_IMSI	GTPv2 End User IMSI
[NFv9 57749]	[IPFIX 35632.277]	%GTPV2_END_USER_MSISDN	GTPv2 End User MSISDN
[NFv9 57750]	[IPFIX 35632.278]	%GTPV2_APN_NAME	GTPv2 APN Name
[NFv9 57751]	[IPFIX 35632.279]	%GTPV2_ULI_MCC	GTPv2 Mobile Country Code
[NFv9 57752]	[IPFIX 35632.280]	%GTPV2_ULI_MNC	GTPv2 Mobile Network Code
[NFv9 57753]	[IPFIX 35632.281]	%GTPV2_ULI_CELL_TAC	GTPv2 Tracking Area Code
[NFv9 57754]	[IPFIX 35632.282]	%GTPV2_ULI_CELL_ID	GTPv2 Cell Identifier
[NFv9 57805]	[IPFIX 35632.333]	%GTPV2_RESPONSE_CAUSE	GTPv2 Cause of Operation

## Plugin HTTP Protocol templates:

[NFv9 57652]	[IPFIX 35632.180]	%HTTP_URL	HTTP URL
[NFv9 57832]	[IPFIX 35632.360]	%HTTP_METHOD	HTTP METHOD
[NFv9 57653]	[IPFIX 35632.181]	%HTTP_RET_CODE	HTTP return code (e.g. 200, 304...)
[NFv9 57654]	[IPFIX 35632.182]	%HTTP_REFERER	HTTP Referer
[NFv9 57655]	[IPFIX 35632.183]	%HTTP_UA	HTTP User Agent
[NFv9 57656]	[IPFIX 35632.184]	%HTTP_MIME	HTTP Mime Type
[NFv9 57659]	[IPFIX 35632.187]	%HTTP_HOST	HTTP Host Name
[NFv9 57660]	[IPFIX 35632.188]	%HTTP_FB00K_CHAT	HTTP Facebook Chat
[NFv9 57833]	[IPFIX 35632.361]	%HTTP_SITE	HTTP server without host name

## Plugin IMAP Protocol templates:

[NFv9 57732]	[IPFIX 35632.260]	%IMAP_LOGIN	Mail sender
--------------	-------------------	-------------	-------------

## Plugin MySQL Plugin templates:

[NFv9 57667]	[IPFIX 35632.195]	%MYSQL_SERVER_VERSION	MySQL server version
[NFv9 57668]	[IPFIX 35632.196]	%MYSQL_USERNAME	MySQL username
[NFv9 57669]	[IPFIX 35632.197]	%MYSQL_DB	MySQL database in use
[NFv9 57670]	[IPFIX 35632.198]	%MYSQL_QUERY	MySQL Query
[NFv9 57671]	[IPFIX 35632.199]	%MYSQL_RESPONSE	MySQL server response
[NFv9 57792]	[IPFIX 35632.320]	%MYSQL_APPL_LATENCY_USEC	MySQL request->response latency (usec)

## Plugin Oracle Protocol templates:

[NFv9 57672]	[IPFIX 35632.200]	%ORACLE_USERNAME	Oracle Username
[NFv9 57673]	[IPFIX 35632.201]	%ORACLE_QUERY	Oracle Query
[NFv9 57674]	[IPFIX 35632.202]	%ORACLE_RSP_CODE	Oracle Response Code
[NFv9 57675]	[IPFIX 35632.203]	%ORACLE_RSP_STRING	Oracle Response String
[NFv9 57676]	[IPFIX 35632.204]	%ORACLE_QUERY_DURATION	Oracle Query Duration (msec)

## Plugin POP3 Protocol templates:

[NFv9 57682]	[IPFIX 35632.210]	%POP_USER	POP3 user login
--------------	-------------------	-----------	-----------------

## Plugin Radius Protocol templates:

[NFv9 57712]	[IPFIX 35632.240]	%RADIUS_REQ_MSG_TYPE	RADIUS Request Msg Type
[NFv9 57713]	[IPFIX 35632.241]	%RADIUS_RSP_MSG_TYPE	RADIUS Response Msg Type
[NFv9 57714]	[IPFIX 35632.242]	%RADIUS_USER_NAME	RADIUS User Name (Access Only)
[NFv9 57715]	[IPFIX 35632.243]	%RADIUS_CALLING_STATION_ID	RADIUS Calling Station Id
[NFv9 57716]	[IPFIX 35632.244]	%RADIUS_CALLED_STATION_ID	RADIUS Called Station Id
[NFv9 57717]	[IPFIX 35632.245]	%RADIUS_NAS_IP_ADDR	RADIUS NAS IP Address
[NFv9 57718]	[IPFIX 35632.246]	%RADIUS_NAS_IDENTIFIER	RADIUS NAS Identifier
[NFv9 57719]	[IPFIX 35632.247]	%RADIUS_USER_IMSI	RADIUS User IMSI (Extension)
[NFv9 57720]	[IPFIX 35632.248]	%RADIUS_USER_IMEI	RADIUS User MSISDN (Extension)
[NFv9 57721]	[IPFIX 35632.249]	%RADIUS_FRAMED_IP_ADDR	RADIUS Framed IP
[NFv9 57722]	[IPFIX 35632.250]	%RADIUS_ACCT_SESSION_ID	RADIUS Accounting Session Name
[NFv9 57723]	[IPFIX 35632.251]	%RADIUS_ACCT_STATUS_TYPE	RADIUS Accounting Status Type
[NFv9 57724]	[IPFIX 35632.252]	%RADIUS_ACCT_IN_OCTETS	RADIUS Accounting Input Octets
[NFv9 57725]	[IPFIX 35632.253]	%RADIUS_ACCT_OUT_OCTETS	RADIUS Accounting Output Octets

[NFv9 57726] [IPFIX 35632.254]	%RADIUS_ACCT_IN_PKTS	RADIUS Accounting Input Packets
[NFv9 57727] [IPFIX 35632.255]	%RADIUS_ACCT_OUT_PKTS	RADIUS Accounting Output Packets
Plugin RTP Plugin templates:		
[NFv9 57622] [IPFIX 35632.150]	%RTP_FIRST_SSRC	First flow RTP Sync Source ID
[NFv9 57623] [IPFIX 35632.151]	%RTP_FIRST_TS	First flow RTP timestamp
[NFv9 57624] [IPFIX 35632.152]	%RTP_LAST_SSRC	Last flow RTP Sync Source ID
[NFv9 57625] [IPFIX 35632.153]	%RTP_LAST_TS	Last flow RTP timestamp
[NFv9 57626] [IPFIX 35632.154]	%RTP_IN_JITTER	RTP jitter (ms * 1000)
[NFv9 57627] [IPFIX 35632.155]	%RTP_OUT_JITTER	RTP jitter (ms * 1000)
[NFv9 57628] [IPFIX 35632.156]	%RTP_IN_PKT_LOST	Packet lost in stream
[NFv9 57629] [IPFIX 35632.157]	%RTP_OUT_PKT_LOST	Packet lost in stream
[NFv9 57633] [IPFIX 35632.161]	%RTP_IN_PAYLOAD_TYPE	RTP payload type
[NFv9 57630] [IPFIX 35632.158]	%RTP_OUT_PAYLOAD_TYPE	RTP payload type
[NFv9 57631] [IPFIX 35632.159]	%RTP_IN_MAX_DELTA	Max delta (ms*100) between consecutive pkts
[NFv9 57632] [IPFIX 35632.160]	%RTP_OUT_MAX_DELTA	Max delta (ms*100) between consecutive pkts
[NFv9 57820] [IPFIX 35632.348]	%RTP_SIP_CALL_ID	SIP call-id corresponding to this RTP stream
[NFv9 57842] [IPFIX 35632.370]	%RTP_MOS	RTP MOS (value * 100)
[NFv9 57843] [IPFIX 35632.371]	%RTP_R_FACTOR	RTP R_FACTOR (value * 100)
[NFv9 57852] [IPFIX 35632.380]	%RTP_RTT	RTP Round Trip Time (ms)
Plugin SIP Plugin templates:		
[NFv9 57602] [IPFIX 35632.130]	%SIP_CALL_ID	SIP call-id
[NFv9 57603] [IPFIX 35632.131]	%SIP_CALLING_PARTY	SIP Call initiator
[NFv9 57604] [IPFIX 35632.132]	%SIP_CALLED_PARTY	SIP Called party
[NFv9 57605] [IPFIX 35632.133]	%SIP_RTP_CODECS	SIP RTP codecs
[NFv9 57606] [IPFIX 35632.134]	%SIP_INVITE_TIME	SIP SysUptime (epoch) of INVITE
[NFv9 57607] [IPFIX 35632.135]	%SIP_TRYING_TIME	SIP SysUptime (epoch) of Trying
[NFv9 57608] [IPFIX 35632.136]	%SIP_RINGING_TIME	SIP SysUptime (epoch) of RINGING
[NFv9 57609] [IPFIX 35632.137]	%SIP_INVITE_OK_TIME	SIP SysUptime (epoch) of INVITE OK
[NFv9 57610] [IPFIX 35632.138]	%SIP_INVITE_FAILURE_TIME	SIP SysUptime (epoch) of INVITE FAILURE
[NFv9 57611] [IPFIX 35632.139]	%SIP_BYE_TIME	SIP SysUptime (epoch) of BYE
[NFv9 57612] [IPFIX 35632.140]	%SIP_BYE_OK_TIME	SIP SysUptime (epoch) of BYE OK
[NFv9 57613] [IPFIX 35632.141]	%SIP_CANCEL_TIME	SIP SysUptime (epoch) of CANCEL
[NFv9 57614] [IPFIX 35632.142]	%SIP_CANCEL_OK_TIME	SIP SysUptime (epoch) of CANCEL OK
[NFv9 57615] [IPFIX 35632.143]	%SIP_RTP_IPV4_SRC_ADDR	SIP RTP stream source IP
[NFv9 57616] [IPFIX 35632.144]	%SIP_RTP_L4_SRC_PORT	SIP RTP stream source port
[NFv9 57617] [IPFIX 35632.145]	%SIP_RTP_IPV4_DST_ADDR	SIP RTP stream dest IP
[NFv9 57618] [IPFIX 35632.146]	%SIP_RTP_L4_DST_PORT	SIP RTP stream dest port
[NFv9 57619] [IPFIX 35632.147]	%SIP_FAILURE_CODE	SIP failure response code
[NFv9 57620] [IPFIX 35632.148]	%SIP_REASON_CAUSE	SIP Cancel/Bye/Failure reason cause
[NFv9 57834] [IPFIX 35632.362]	%SIP_C_IP	SIP C IP addresses
[NFv9 57835] [IPFIX 35632.363]	%SIP_CALL_STATE	SIP Call State
Plugin SMTP Protocol templates:		
[NFv9 57657] [IPFIX 35632.185]	%SMTP_MAIL_FROM	Mail sender
[NFv9 57658] [IPFIX 35632.186]	%SMTP_RCPT_TO	Mail recipient
Plugin Whois Protocol templates:		
[NFv9 57823] [IPFIX 35632.351]	%WHOIS_DAS_DOMAIN	Whois/DAS Domain name

For instance if you want to specify NetFlow v9 flows in a format similar to v5 flows you can do as follows:

```
nprobe -T "%IPV4_SRC_ADDR %IPV4_DST_ADDR %IPV4_NEXT_HOP %INPUT_SNMP %OUTPUT_SNMP %IN_PKTS
%IN_BYTES %FIRST_SWITCHED %LAST_SWITCHED %L4_SRC_PORT %L4_DST_PORT %TCP_FLAGS %PROTOCOL
%SRC_TOS %SRC_AS %DST_AS %SRC_MASK %DST_MASK"
```

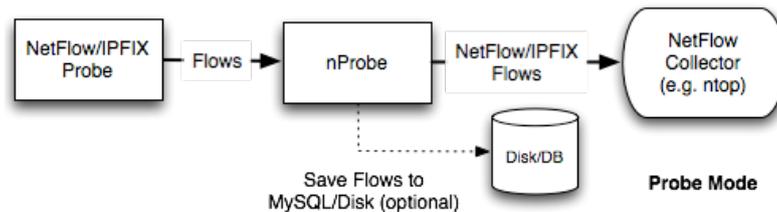
Note that the fields start with a % and are separated by a space.

## 9. Appendix C: nProbe Usage Modes

nProbe can be used in three modes:

- Probe (default)
- Collector: Flow collection only, no Probe
- Proxy: Receive flows via NetFlow and emit them (optionally combining with captured traffic) to a remote collector.

### 1. Probe mode (default)



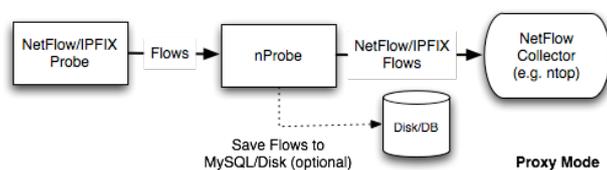
Command: `"nprobe -i eth0 -n collector_ip:2055"`

### 2. Collector mode



Command: `"nprobe -nf-collector-port 2055"`

### 3. Proxy mode



Command: `"nprobe -nf-collector-port 2055 -n collector_ip:2055 -V 9"`

In proxy mode you can convert from/to IPFIX/NetFlow v5/v9 in order to smoothly upgrade to newer netflow protocol versions while capitalizing on previous protocol versions. So you can for instance convert flows coming from your v5 router into IPFIX and vice-versa. Note that with some combinations (e.g. from v9 to v5) you might lose some flow information.

## 10. Appendix D: nProbe License

nProbe is (C) 2002-14 by ntop di Luca Deri that is distributed under GNU GPL who owns the nProbe intellectual property and copyright (beside the sFlow collector code).

If you wish to integrate or embed nProbe into a proprietary software or an embedded software/hardware solution such as the nBox, be aware that we offer alternative licenses that allow you to do that. Please contact [license@ntop.org](mailto:license@ntop.org).

GPL requires that any work derived from a GPL licensed work (as nProbe) must also be distributed under GPL. As the term "derivative work" (see [http://en.wikipedia.org/wiki/Derivative\\_work](http://en.wikipedia.org/wiki/Derivative_work)) is not entirely clear we want to clarify this concept in the case of nProbe. We consider a derivated work of nProbe for the purpose of this license if it does any of the following:

- Integrates (even partially) nProbe source code
- Includes (even partially) nProbe copyrighted data files
- Integrates/embeds nProbe into a binary installer/application
- Includes the nProbe into an appliance, router or similar device
- Links (even through nProbe's plugins) to a library that is not available under GPL
- Executes nProbe and uses the produced results (usually flows either on the network, disk or database)

Note that the list applies to both nProbe "as a whole" and also to portions of it. The above list is not exhaustive but it's used to clarify the term derivative work with respect to nProbe. This means that:

- nothing prevents you from distributing a proprietary product (either appliance, GUI front-end, or application) based on nProbe. Just sell/distribute it \*without\* nProbe, and point your customers to <http://www.ntop.org/products/nprobe/> in order to have access to nProbe.
- you cannot include nProbe into a non-GPL derivative work.

Note that the above is a clarification of what we mean for "derivative work" in the scope of the GPL-licensed nProbe. This interpretation applies only to nProbe and not to any other GPL products.

if you have questions about nProbe licensing please contact [license@ntop.org](mailto:license@ntop.org)

See COPYING and EULA files for more details.

Note that the EULA applies \*only\* to nProbe derived work.

## 11. Appendix E: EULA

### NTOP END USER LICENSE AGREEMENT

THIS "END USER LICENSE AGREEMENT" ("EULA") IS A BINDING LEGAL AGREEMENT. BY USING THE "SOFTWARES PROVIDED TOGETHER WITH THIS EULA" (THE "SOFTWARE"), OR USING ANY AUTHORIZATION CODE PROVIDED BY THE LICENSOR, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS EULA, THAT YOU UNDERSTAND IT, AND THAT YOU AGREE TO BE BOUND BY ITS TERMS. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS EULA, DO NOT MAKE ANY USE OF THE SOFTWARE OR ANY USE OF AUTHORIZATION CODES. YOUR USE OF THE SOFTWARE AND YOUR USE OF ANY AUTHORIZATION CODE IS SUBJECT AT ALL TIMES TO

THE TERMS AND CONDITIONS HERE, INCLUDING ALL PROVISIONS REGARDING THE LIMITATIONS OF THE LICENSOR LIABILITY.

1. Grant of License in Favour of Registered Users. "NTOP di Deri Luca" ("Licensor") grants you a non-exclusive license to use the Software (including any updates of the Software that the Licensor may make available to you at his own discretion) and any "documentation files pertaining to the Software" ("Documentation"), only in connection with a single hardware unit with a unique "Media Access Control" ("MAC") address or unique system identifier of any kind. All copyright notices in the Software and Documentation must be retained at all times. The Software and Documentation shall be used only for your own personal, non-commercial use and not for the benefit of any other person or entity. You may also make one copy of the Software in machine-readable form only for back-up purposes, provided you properly reproduce also both any Licensor's copyright notices/credits and any proprietary legends/disclaimers.

2. IP Ownership. All the intellectual property rights pertaining to Software and Documentation shall be reserved to the Licensor, at all times. Provided the Section 1 above, any other use of the Software by any person or entity of any kind is strictly forbidden and is a violation of this EULA. All the rights not expressly granted to you herein are reserved to the Licensor. You are not allowed to remove any Licensor's copyright notices/credits and any proprietary legends/disclaimers from any copy of Software, Documentation or any component thereof.

3. Restrictions. You are not allowed to or permit/assist any third party to: (a) publish, display, disclose, rent, lease, modify, copy, loan, distribute, or create derivative works based on the Software or any part thereof; (b) reverse engineer, decompile, translate, adapt, or disassemble the Software or any part thereof; (c) attempt to create or otherwise reproduce in any form the source code from the object code of any portion or component of the licensed Software; (d) sublicense the Software or permit the exploitation of the Software by more than a single hardware unit with a unique MAC address or unique system identifier of any kind; (e) attempt to disable or circumvent any technological protection measure of the Software or assist third parties to do so.

4. Confidentiality. You agree to maintain the confidentiality of Software and Documentation: you will not disclose Software or Documentation, or any information or materials related thereto, to any third party without the express written consent of NTOP di Deri Luca. You further agree to take all reasonable measures to limit access to Software and Documentation only to those of your employees who reasonably require such access to perform their employment obligations and who are bound by confidentiality agreements with you (in order to maintain the confidentiality of Software and Documentation).

5. No Warranty. Software is provided "as it is". To the maximum extent permitted by applicable laws, Licensor disclaims warranties of any kind, either explicit or implied, including, without limitation, implied warranties of merchantability and fitness for a particular purpose. Licensor does not warrant that the functions contained in the Software will meet any requirements or needs you may have, or that the Software will operate error free, or in an uninterrupted fashion, or that any defects or errors in the Software will be corrected, or that the Software is compatible with any particular platform. Licensor is not obligated to provide any updates to the Software.

6. Limitation of Liability. To the extent not prohibited by applicable laws, in no event shall the Licensor be liable to you or any third party for any incidental or consequential damages (including - without limitation - indirect, special, punitive or exemplary damages for loss of business, loss of profits, business interruption or loss of business information) arising out of or related to the use of or the inability to use the Software, or for any claim by any other party, even if the Licensor has been advised of the possibility of such damages. In no event shall Licensor's total liability to you for all damages exceed the amount of the license fee paid by you to Licensor for Software and Documentation.

7. Export Restrictions. You may not export Software and Documentation in violation of applicable laws and regulations.

8. High Risk Activities. The licensed Software is not fault-tolerant and is not designed, manufactured or intended for any kind of use with on-line control equipment in hazardous environments requiring fail-safe performance (such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapon systems in which the failure of the Software could lead directly to death, personal injury or severe physical or environmental damage: all the so called "High Risk Activities"). Accordingly, Licensor specifically disclaims any express or implied warranty of fitness for High Risk Activities.

9. Ethics. The licensor commits itself to use the Software in compliance with all applicable local, national and international laws, rules and regulations, including any laws regarding the transmission of technical data exported from its country of residence. In no case the licensed Software can be used to track, spy, intercept or collect evidence of network communications to be used against individuals or organizations, to prosecute individuals or organizations, or to restrict their freedom.

10. Termination. Licensor may terminate this Agreement at any time if you violate its terms. Upon termination, you must immediately destroy or return to Licensor the Software and Documentation. The provisions of Sections 2 (IP Ownership), 3 (Restrictions), 4 (Confidentiality), 5 (No Warranty), 6 (Limitation of Liability) and the provisions of this Section 9 (Termination) shall in any case survive the termination or expiration hereof.

11. General Provisions, Governing Law, Jurisdiction. This EULA shall be governed by, construed and interpreted in accordance with Italian Law. The licensee agrees that any dispute arising from or connected to this EULA shall be submitted to the exclusive jurisdiction of the Italian Specialized IP Courts. Therefore, the jurisdiction of any other court is expressly excluded. This EULA shall constitute the entire agreement between the parties; any waiver or modification of this EULA shall be effective only if it is in writing and signed by both parties. Should any part of this EULA be found invalid or unenforceable by an Italian Specialized IP Court, the remainder of this EULA shall be interpreted so as to reasonably effect the intention of the parties.

EACH PARTY IS HEREBY CONFIRMING ITS AGREEMENT WITH THE FOREGOING BY SIGNING AND RETURNING ONE COPY OF THIS EULA TO THE OTHER PARTY.

PURSUANT TO ARTICLES 1341 AND 1342 OF THE ITALIAN CIVIL CODE, THE PARTIES HEREBY ACKNOWLEDGE AND EXPRESSLY APPROVE SECTIONS 1 (Grant of License in Favour of Registered Users), 2 (IP Ownership), 3 (Restrictions), 5 (No Warranty), 6 (Limitation of Liability), 8 (High Risk Activities), 9 (Termination) AND 10 (General Provisions, Governing Law, Jurisdiction).