n2disk™

Ultra-high-speed packet recorder with realtime indexing.

User’s Guide

n2disk v.3.1

Dec 2017

1. Introduction

n2disk is a network traffic recorder application. With n2disk you can capture full-sized network packets at multi-Gigabit rate (above 10 Gigabit/s on adequate hardware) from a live network interface, and write them into files without any packet loss. n2disk has been designed to write files into disks for very long periods, you have to specify a maximum number of distinct file that may be written during the execution, and if n2disk reaches the maximum number of files, it will start recycling the files from the oldest one. This way you can have a complete view of the traffic for a fixed temporal window, knowing in advance the amount of disk space needed.

n2disk uses the industry standard PCAP file format (regular and nanosecond) to dump packets into files so the resulting output can be easily integrated with existing third party or even open/source analysis tools (like Wireshark). Alternatively n2disk can produce compressed PCAP files (*.npcap*) to reduce disk space and optimise disk throughput.

n2disk has been designed and developed mainly because most network security systems rely on capturing all packets (both header and payload), since any packets may have been responsible for the attack or could contain the problems that we are trying to find. Netflow information is more manageable and requires less disk space to be stored, but in some cases, like deep-packet-inspection analysis or controlled traffic regeneration, it is not useful. When we need to collect the entire packet, because we need all the information, n2disk has to be used.

n2disk can be effectively used to perform numerous activities, among these:

- Off-line network packets analysis by feeding a specialised tools like Snort.
- Reconstruct particular communication flows or network activities.
- Reproduce the previous captured traffic to a different network interface.

2. Main Features

Some of the n2disk features include:

- Fully user configurable.
- Use of the standard PCAP file format (regular and nanosecond).
- Ability to compress PCAP on-the-fly producing *.npcap* files.
- High-performance packet to disk recording.
- BPF filters supports (using the same format as in the popular tcpdump tool) to filter out the unwanted network packets from the recording process.
- Optimised BPF-like filters support, a faster replacement for BPF filters (a subset of the BPF syntax is supported).
- Multi-core support. n2disk has been designed with multicore architectures in mind. It uses at least 2 threads (one for the packet capture and one for the disk writing) and it is possible to further parallelise packet capture using multiple threads. The communication between threads has been carefully optimised.
- PF_RING acceleration. n2disk exploit the packet capture acceleration offered both by standard PF_RING and PF_RING ZC.
- Direct-IO disk access. n2disk uses the Direct IO access to the disks in order to obtain maximum disk-write throughput.
- Real-Time indexing. n2disk is able to produce an index on-the-fly during packet capture. The index can be queried using a BPF-like syntax to quickly retrieve interesting packets in a specified time interval.

Dump Format

Capture files are saved in the standard PCAP format with timestamps in microsecond resolution. The PCAP format with nanosecond resolution is also an option when using the PF_RING support and network cards with hardware timestamp.

Capture files are stored in sequential order with a per-file limit in duration or size. It is also possible to specify the maximum number of files: when the limit is reached, n2disk recycles the files already written, starting from the oldest one.

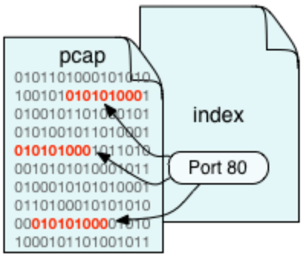
Packet Capture Filters

n2disk uses the same syntax for capture filters as tcpdump and any other program that uses the libpcap library. In fact two type of capture filters are available:

- Standard BPF filters.
- BPF-like filters (a subset of the BPF syntax), a faster replacement for BPF. See Appendix B.

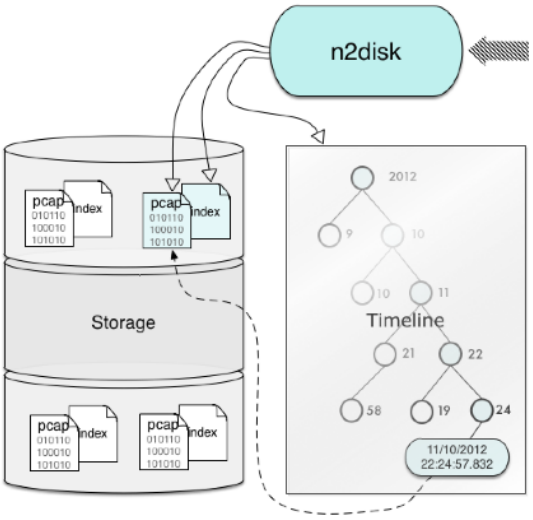
Index and Post-Capture Filters

n2disk produces an index on-the-fly during packet capture. The index is in a separate file associated with the capture file and houses all the flows information and packets offsets.
As soon as the dump file is ready, using the utilities provided with n2disk, it is possible to query the index for specific packets in a time interval using a BPF-like syntax. The packets matching the filter are returned in PCAP format.



Timeline

Besides the per-dump-file index, n2disk can also produce a timeline, a way of keeping the whole captured traffic in chronological order.
Using the utilities provided with n2disk, it is possible to query the timeline for specific packets belonging to the whole dump set in a given time interval.



3. Expected n2disk Performance

The n2disk performance is influenced by two factors:

- Packet capture performance
- Disk write performance

Unless you have both fast capture and write performance, n2disk will not operate properly. In the appendix you can learn how to use ZC drivers to accelerate packet capture performance.

We have implemented in n2disk a testing facility for allowing you to determine if your system has optimal performance. We suggest you to run this performance test prior to use n2disk. n2disk comes with the `-e` command line option

```
[--simulation-mode|-e] <mode> | Simulation mode (debug only)
                              | 1 - Test max dump speed (simulate capture)
                              | 2 - Test max capture speed (simulate storage)
```

The -e option can have two mode:

- 1: packet capture is simulated. Namely n2disk simulates packet capture so that n2disk does not spend time on capturing and thus the capture speed is virtually unlimited. In this mode we can evaluate the disk write performance as packet capture costs virtually zero and it has virtually and infinite speed.
- 2: write to disk is simulated. Namely n2disk does not save packets on disk but it reports that the pcap file has been saved. This way no time is spent writing on disk, and thus we can measure the packet capture performance. Note that in this setup, you must inject traffic on the ingress (-i) n2disk monitoring interface.

In general, disk-write speed should exceed network speed (or packet capture speed) in order to guarantee that all captured packets can be written to disk. Note that disk performance does not depend just on hardware, but also on filesystem type and configuration. Please refer to the appendix for more details on this subject.

4. Versions

The n2disk software comes in four versions: n2disk1g, n2disk5g, n2disk (10g and above).

N2disk1g (1 Gbit/s)

This is the entry-level version, enough for rates up to 1 Gigabit.

n2disk5g (5 Gbit/s)

This is the standard version, with two threads: one for packet capture and one for disk writing. This version can handle rates up to 5 Gigabit.

n2disk (10/40 Gbit/s)

This version includes support for ZC and multithreaded packet capture, with an internal architecture able to balance the load across up to 32 threads in a zero-copy fashion, and the ability to use multiple threads also in IPC mode (acting as consumer of another ZC process, attaching to a sw queue). This version can handle 10 Gigabit and above.

5. Usage

In order to save all the traffic into disks, the n2disk application has to be activated on an interface from which it is possible to capture all the traffic you are interested in. Once activated, n2disk will save the traffic data into the specified directory recycling the files already written, starting from the oldest one, this in case the maximum number of created files is reached.

In the following sections, we discuss all the n2disk 2.x command line options and how to efficiently configure n2disk to capture all the traffic flowing in your network.

Command Line Options

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

```
CAPTURE SETTINGS

[--interface|-i] <device>           | Ingress packet device.
[--active-wait|-g]                  | Active packet polling.
[--poll-duration|-q] <duration>     | Poll duration (usec). Lower more CPU is used,
                                    | better response time. Default: 1 usec.
[--snaplen|-s] <len>                | Max packet capture length. Default: 1536.
[--packet-slicing] <header level>   | Slice packet after the specified header.
[--strip-header-bytes|-s] <len>     | Strip the specified number of bytes from packet header.
[--sample-rate|-y] <rate>           | Packet sample rate (e.g. 100 means 1:100 sampling).
[--capture-direction|-2] <dir>      | Capture direction: 0=RX+TX, 1=RX only (default), 2=TX only
[--not-promisc|-3]                  | Do not set the interface in promiscuous mode.

FILTERING

[--filter|-f] <filter>              | BPF (tcpdump-like) ingress packet filter.
[--fast-filter|-F] <filter>         | Faster replacement for BPF ingress packet filter.

TIMESTAMPING

[--hw-timestamp|-J] <mode>    | Hardware timestamp packet strip. Supported modes:
                              | ixia - Timestamped packets by ixiacom.com hardware devices
                              | NOTE: Silicom Hw TS NICs are supported automatically.
[--time-pulse|-S <id>]        | Enable time pulse thread (optimise sw packet timestamping)
                              | and bind it to the specified core.
[--nanoseconds|-d]            | Dump timestamps in nanosecond format.
[--no-timestamp|-T]           | Do not compute timestamps.

DUMP SETTINGS

[--chunk-len|-C] <len>        | Size (KB) of the chunk written to disk (must be
                              | multiple of 4096). Default: 64 KB.
[--journaling|-j]             | Enable journaling.
[--buffer-len|-b] <len>       | Buffer length (MBytes).
```

```
[--dump-directory|-o] <dir>          | Directory where dump files will be saved.
[--archive-directory|-O] <dir>       | Directory where dump files will be archived (slower
                                     | disks). The -a option is overwritten when using it.
[--max-file-len|-p] <len>            | Max pcap file length (MBytes).
[--max-file-duration|-t] <secs>      | Max pcap file duration (sec).
[--max-file-packets|-N] <num>        | Max packets per pcap file. Default: 0 (no max duration).
[--archive|-a]                       | Archive pcap file (rename to .old) instead
                                     | of overwriting if already present on disk.
[--disk-limit|-6] <size>             | Max amount of disk space to use (MBytes or %).
[--max-num-files|-m] <num>           | Max number of files before restarting file name.
[--max-nested-dirs|-n] <num>         | Max number of nested dump sub-directories.
[--file-prefix|-x] <prefix>          | Dump file prefix.
[--hugepages|-U]                     | Use hugepages for memory allocation.
[--disable-direct-io|-r]             | Disable Direct I/O (experts only).
[--stop-on-limit|-L]                 | Use max len/duration/packets as limit to stop capture.
[--remove-ahead|-H]                  | Delete in advance dumped files that will soon be
                                     | overwritten. Use this option to reduce file deletion cost.
[--pcap-compression|-M]              | Compress pcap files (produces .npcap compressed files)

INDEX SETTINGS

[--index|-I]                         | Create pcap-index on-the-fly.
[--index-version|-1] <version>       | Specify the index version:
                                     | 1 - Standard index (default)
                                     | 2 - Flow-based index
[--timeline-dir|-A] <dir>            | Time-arranged directory for produced pcaps and indexes.
[--extended-index|-E] <type>         | Extended index type: 1 - add per-packet timestamp.
[--index-tunnel-content|-G] <m>      | Compute index on tunnel content (GTP-U)
                                     | 1 - Index both inner and outer headers
                                     | 2 - Use inner header as outer header
[--index-on-compressor-threads|-Z]   | Compute index on the thread(s) used for compression (-z)
                                     | instead of using the capture thread(s).
[--exec-cmd] <cmd>                   | Execute the specified command when a pcap has been dumped.


MULTITHREADING

[--reader-threads|-R] <ids>          | Enable multithread capture support and bind reader threads
                                     | to the specified core ids (e.g. 0,1,2,3).
[--reader-threads-queue-len] <len>   | Reader threads queue length.
[--cluster-id|-X] <id>               | Specify the ZC cluster id for multithreaded support
[--cluster-ipc-queues|-W] <ids>      | In case of external ZC cluster (IPC), specify the queue ids
                                     | to use from the cluster for distributing packets across
                                     | reader threads
[--cluster-ipc-pool|-B] <id>  | In case of external ZC cluster (IPC), specify the pool id to
                                     | use for allocating buffers for distribution across reader
                                     | threads
```

```
CPU AFFINITY

[--reader-cpu-affinity|-c] <id>       | Bind the reader thread to the specified core.
[--writer-cpu-affinity|-w] <id>       | Bind the writer thread to the specified core.
[--compressor-cpu-affinity|-z] <ids>  | Enable multithread compression and/or indexing and
                                      | bind thread(s) to the specified core ids (e.g. 0,1,2,3)
                                      | (mandatory with indexing on Napatech cards)

MICRO-BURST DETECTION
[--uburst-detection]                  | Enable microburst detection.
[--uburst-log]                        | Microbursts log file.
[--uburst-win-size] <usec>            | Window size for microburst check.
[--uburst-link-speed] <mbit/s>        | Link speed.
[--uburst-threshold] <percent>        | Traffic threshold (link speed percentage).

OTHER

[--daemon]                         | Daemonize this app at startup
[--export-flow-offload] <cluster>  | Enable flows offload and export flows using an ipc queue
[--unprivileged-user|-u] <user>    | Use <user> instead of nobody when dropping privileges.
[--pid|-P] <path>                  | Save the pid into the specified path.
[--version|-V]                     | Print application version.
[--help|-h]                        | Help.
[--verbose|-v]                     | Verbose.
[--show-system-id]                 | Print the system identifier.
[--check-license]                  | Checks if the license is present and valid.
[--check-maintenance]              | Checks the maintenance status for the specified license
[--syslog|-l]                      | Dump trace messages to syslog.
[--event-log|-Q] <file>            | Save relevant events (i.e. drops) onto the specified file.
```

6. Tuning n2disk performance

In order to achieve a good n2disk setup able to obtain the maximum performance, it is important to take into account the following aspects.

Besides libpcap, n2disk can take advantage from the PF_RING (both standard and ZC) acceleration to capture packets from a live network interface.

In case standard PF_RING is used it is particularly important to reserve enough ring buffer space inside the kernel. Furthermore, in order to reduce the number of clock-cycles needed to capture the packets and cross the network stack, it is possible to turn off the PF_RING transparent mode setting the transparent_mode option to 2.

For further info about PF_RING and ZC please have a look at Appendix A.

Regarding the n2disk start-up parameters particularly important are the following options:

- The buffer length (-b) has to be big enough. 1 GB is sufficient in most cases.
- The write chunk size (-C) has to be greater than or equal to 64 Kbytes.
- The maximum file size (-p) should not be very small. A good value has to be more than 64 Mbytes.
- The core binding for the reader (-c) and writer (-w) thread. It is highly recommended to bind those threads to different core of the same physical CPU (according to the system topology, you should choose the physical CPU where the network card is closer/directly connected). The same applies when using multithreaded packet capture (-R).

Examples

Basic example:

```
n2disk -i zc:eth1 -o /storage/eth1/ -b 1024 -C 1024 -p 512 -q 1 -S 0 -
c 1 -w 2
```

Packet indexing:

```
n2disk -i zc:eth1 -o /storage/eth1/ -I -A /index/eth1/ -b 1024 -C 1024
-p 512 -q 1 -S 0 -c 1 -w 2 -z 3,4,5 -Z
```

7. Utilities

The following utilities are provided with n2disk.

disk2n

This utility replays network traffic previously captured with n2disk on live networks observing the original inter-packet time. The industry standard PCAP file format (both regular and nanosecond) is supported.

disk2n has been designed to replay multiple pcap files of any size with limited memory usage. You can specify a playlist of pcap files and instruct n2disk to continue from the first file when the last one is reached. It is possible to determine in advance the amount of memory used by disk2n, in fact it uses a memory buffer of arbitrary size to cache the next packets to replay, achieving good transmit performance with limited memory usage.

disk2n is multithreaded and uses 3 threads: one for reading packets from disk, one for packet transmission and one for precise time generation.

```
TX SETTINGS

[--interface|-i] <device>               | Egress interface.
[--pcap-file|-f] <pcap file path>       | Pcap file to replay.
[--multiple-pcap-files|-m] <txt path>   | Text file containing a list of pcap files to replay
                                        | in sequence.
[--timeline-dir|-A] <dir>               | Timeline directory created by n2disk (-A).
[--begin-time|-B] <epoch time>          | Begin time on which packets will be selected.
[--end-time|-E] <epoch time>            | End time on which packets will be selected.
[--coherent-ts-across-pcaps|-H]         | Time coherence across pcap files (do not collapse
                                        | inter-pcap gap).
[--inter-packet-time|-I] <device>       | Inter-packet time nsec (ignore pcap timestamps).
[--precise-timing|-p]                   | High time accuracy (lower throughput, forces
                                        | active wait).
[--active-wait|-g]                      | Active wait (higher cpu load).
[--one-shot|-O]                         | Send pcap(s) once.


BUFFERING SETTINGS

[--chunk-len|-C] <len>                  | (KBytes) Size of the chunk read from disk.
[--buffer-len|-b] <len>                 | (MBytes) Buffer length.
[--hugepages|-U]                        | Use hugepages for memory allocation.
[--disable-direct-io|-r]                | Disable Direct I/O (experts only).


CPU AFFINITY

[--reader-cpu-affinity|-c] <id>         | Binds the reader thread to the specified core.
[--sender-cpu-affinity|-w] <id>         | Binds the sender thread to the specified core.
[--time-pulse|-S] [<id>]                | Bind time pulse thread to the specified core.

PACKET REFORGING

[--src-mac] <mac>                       | Reforge source MAC address.
[--dst-mac] <mac>                       | Reforge destination MAC address.
[--src-ip] <ip>                         | Reforge source IP address.
[--dst-ip] <ip>                         | Reforge destination IP address.
[--src-port] <port>                     | Reforge source port.
[--dst-port] <port>                     | Reforge destination port.
                                        | (comma-separated list in case of multiple egress
                                        | interfaces, one per interface)

OTHER

[--simulation-mode|-e]                  | Simulation mode (disk simulation, synthetic traffic is
                                        | generated).
[--unprivileged-user|-u] <username>     | Use <username> instead of nobody when dropping
                                        | privileges.
[--pid|-P] <path>                       | Save the pid into the specified path.
```

```
[--version|-V]                          | Print application version.
[--help|-h]                             | Help.
[--verbose|-v]                          | Verbose.
[--show-system-id]                      | Print the system identifier.
[--check-license]                       | Checks if the license is present and valid.
[--syslog|-l]                           | Dump trace messages to syslog.
[--event-log|-Q] <file>                 | Save relevant events onto the specified file.
```

Example:

```
disk2n -i zc:eth1 -m playlist.txt -C 4096 -b 2000 -S 0 -c 1 -w 2
```

npcapindex

This utility produces an index file (with the same format of the index produced by n2disk) from a pcap file. Available options are:

```
[-i ] <pcap file>                        | Pcap file to index
[-o] <index file>                        | Index file
[-f ] <bpf>                              | BPF filter to select packets to index
[-G] <mode>                              | Compute index on tunnel content (GTP-U)
                                         | 1 - Index both inner and outer headers
                                         | 2 - Use inner header as outer header
[-I] <version>                           | Specify the index version:
                                         | 1 - Standard index (default)
                                         | 2 - Flow-based index
[-v]                                     | Verbose
[-h]                                     | Help
```

Example:

```
npcapindex -i dummy.pcap -o dummy.index
```

npcapextract

This utility, given a pcap file and its index, or alternatively a “timeline” tree created by n2disk, produces a new pcap file (or several pcap files according to the specified file limit) with the packets matching the provided filter in BPF-like syntax. Available options are:

```
FIND FROM TIMELINE

[-t] <timeline dir>               | Timeline directory created by n2disk with -A (multiple -t are
allowed)
[-b] <begin time>                 | Begin time on which packets will be selected
[-e] <end time>                   | Endtime on which packets will be selected

FIND FROM SINGLE FILE/INDEX

[-a] <pcap file>                  | Pcap file from which extract packets
[-i] <index file>                 | Index corresponding to the pcap file

FILTERING

[-f] <filter>                     | BPF ingress packet filter (tcpdump-like)
                                  | NOTE: -b/-e time must specified as YYYY-MM-DD hh:mm:ss

DUMP SETTINGS

[-o] <pcap file>                  | Pcap file or directory where matched packets will be saved
[-s] <len>                        | Cut packets to len (Bytes)
[-P] <len>                        | Max pcap file length (MBytes)
[-N] <number>                     | Max packets per pcap file. (Default: 0 - no max duration)
[-x] <prefix>                     | Dump file prefix (to be used with -P or -N)

OTHER

[-g] <core_id>                    | CPU core affinity
[-u] <username>                   | Use <username> instead of nobody when dropping privileges
[-O]                              | Write output to stdout (npcapextract -O .. | tshark -i -)
[-l]                              | List matching pcap files only (no packet extraction)
[-T]                              | Test index scan performance (no packet extraction)
[-0]                              | Write empty pcap on no match
[-v] <level>                      | Verbosity level: 0..6 (default = 2)
[-h]                              | Help
```

Example of extraction from single PCAP/Index file:

```
npcapextract -a dummy.pcap -i dummy.pcap.idx -o output.pcap -f
"host 192.168.1.1 or port 80”
```

Example of extraction from a timeline/index produced by n2disk:

```
npcapextract -t /tmp/n2disk/timeline -b "2012-10-02 12:00:00" -e
"2012-10-03 00:00:00" -o output.pcap -f "host 192.168.1.1 and
port 80”
```

Since version 3.0 npcapextract features PAM support. With PAM it is possible to integrate multiple authentication schemes (including LDAP) for granting traffic retrieval capabilities to selected users/groups. By default the user running npcapextract (or the one specified with -u) should have read permissions on the dump set in order to extract/retrieve traffic. In order to enable PAM support, you should set the SUID to the npcapextract executable, and configure /etc/pam.d/npcapextract with your preferred authentication scheme (by default npcapextract uses the unix authentication!).

npcapdecompress

This utility decompress compressed *.npcap* files to standard pcap files. Available options are:

```
[-v]                            | Verbose
[-h]                            | Help
[-i ] <.npcap file>             | Compressed npcap file to decompress
[-o] <.pcap file>               | Decompressed pcap file (output)
```

Example:

```
npcapdecompress -i dummy.npcap -o dummy.pcap
```

8. Netflow and Traffic Recording

nProbe Cento to n2disk Raw Traffic Forwarding

nProbe Cento, the 1/10/40/100 Gbit NetFlow/IPFIX Probe, Traffic Classifier, and Packet Shunter, integrates straightforwardly with n2disk to generate Netflow while selectively writing raw data to disk.

Both nProbe Cento and nProbe have been designed to fully exploit the resources provided by multi-core systems, aggregating traffic that comes from multiple, independent input queues in the former, scattering packet processing on multiple cores in the latter.

Let’s say for instance that nProbe Cento has to be used as a flow exporter for a v9 NetFlow collector listening on 192.168.2.221 port 2055 and, at the same time, has to aggregate monitored traffic received from four different eth1 queues into a single aggregated egress queue. Then the following command can be issued

```
cento-ids -i zc:eth1@[0-3] —aggregated-egress-queue --v9 
192.168.2.221:2055
```

Upon successful startup, nProbe Cento will output the ZC egress queue identifier that needs to be passed to n2disk, in the format `zc:<cluster id>:<queue id>`. Assuming nProbe Cento assigned identifier `zc:10@0` to the `aggregated-egress-queue`, it is possible to record egress traffic via n2disk using the following command:

```
n2disk -i zc:10@0 --dump-directory /storage -—max-file-len 1024 --
buffer-len 2048 --chunk-len 4096 --reader-cpu-affinity 4 --writer-cpu-
affinity 5 --index --index-version 2 --timeline-dir /storage
```

Where `-i zc:10@0` is the n2disk ingress queue, that is the egress queue of nProbe Cento.

The user has a fine-grained control over the traffic that gets actually forwarded to the egress queue by nProbe Cento, for instance to avoid forwarding encrypted traffic (e.g., SSL, HTTPS) to n2disk for recording, or to forward only the traffic that comes from a suspicious range of addresses. This is enforced in nProbe Cento through a set of rules that can be applied at different levels (per queue, per subnet, per application protocol) specified using a configuration file (—egress-conf option) that follows the INI standard. Example:

```
cento-ids -i zc:eth1@[0-3] --aggregated-egress-queue —egress-conf 
egress.example --dpi-level 2 --v9 192.168.2.221:2055
```

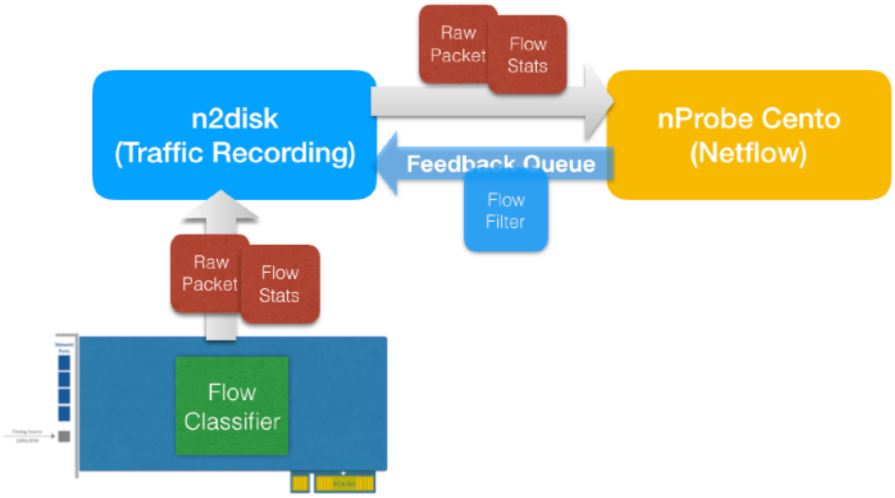
To learn more about aggregates egress queues and configurations please refer to the nProbe Cento User’s Guide.

n2disk to nProbe Cento Flow Updates with Hardware Offload

Since PF_RING version 7.0 and n2disk version 3.0, it is possible to offload flow processing to the network card (on supported adapters) reducing CPU utilisation. Using flow offload the network card can be programmed to:

1. Keep flow state, doing flow classification in hw.
2. Periodically provide informations like hash, packets, bytes, first/last packet timestamp, tcp flags, to the application.
3. Drop/bypass/mark flow packets.

n2disk is able to record raw data while feeding nProbe Cento with flow updates. In addition to flow updates, when nDPI is enabled in nProbe Cento for L7 protocol detection, n2disk can also forward raw traffic, using a feedback queue for shunting flow packets as soon as the nDPI engine detects the protocol.



By leveraging on this technology, it is possible to do traffic recording and Netflow generation at high speed on the same box with low CPU utilization.

For example, if we want to run n2disk on an Accolade card featuring flow offload support, and export flow updates to nProbe Cento, we need to add the —export-flow-offload option, specifying a cluster id that is used for allocating a ZC queue for forwarding flow updates and raw data from n2disk to nProbe Cento:

```
n2disk -i anic:0 -o /storage/n2disk -p 1024 -b 4096 -C 4096 -c 0 -
w 1 --disk-limit 80% --export-flow-offload 99
```

After n2disk is running, we can attach nProbe Cento to the ZC queue for receiving the flow updates using zc:<cluster id> as interface name, and adding the —flow-offload option:

```
cento -i zc:99@0 --flow-offload -P /storage/cento
```

Appendix A
PF_RING Support

PF_RING download instructions can be found in http://www.ntop.org/get-started/download/.

PF_RING kernel module installation

Please note that for some Linux distributions an installation package is provided (http://packages.ntop.org/). If you choose to install from package please read the section “Configuring a PF_RING Deb/RPM package” below.

In order to compile the PF_RING kernel module from source code you need to have the linux kernel headers installed.

```
cd <PF_RING PATH>/kernel
make
```

The kernel module installation requires root capabilities:

```
cd <PF_RING PATH>/kernel
make install
```

Running n2disk on top of PF_RING

Before using n2disk on top of PF_RING, the pf_ring kernel module should be loaded.

```
insmod <PF_RING PATH>/kernel/pf_ring.ko
```

A few configuration options are available via insmod parameters:

- *min_num_slots*: Min number of ring slots
- *enable_tx_capture*: 1 to capture outgoing packets, 0 otherwise
- *enable_ip_defrag*: 1 to enable IP defragmentation, 0 otherwise
- *quick_mode*: 1 to run at full speed but with up to one socket per interface

ZC drivers

If you want to achieve line-rate packet capture even at 10 Gigabit, you should use ZC drivers, implementing zero-copy kernel-bypass packet capture. ZC drivers are part of the PF_RING distribution and can be found in “<PF_RING PATH>/drivers/”.

Currently available ZC drivers are:

- e1000
- e1000e
- igb
- ixgbe
- i40e
- fm10k

Please note that:

- the PF_RING kernel module must be loaded before the ZC driver

- in order to correctly configure the device, it is highly recommended to use the *load_driver.sh* script provided with the drivers (take a look at the script to fine-tune the configuration)
- ZC drivers need hugepages, the *load_driver.sh* script takes care of hugepages configuration, for further information please read Appendix E

Example loading PF_RING and the ixgbe-zc driver:

```
cd <PF_RING PATH>/kernel
insmod pf_ring.ko

cd PF_RING/drivers/intel/ixgbe/ixgbe-X.X.X-zc/src
make
./load_driver.sh
```

Configuring a PF_RING Deb/RPM package

Another option for installing PF_RING is using the installation packages provided at http://packages.ntop.org/.
Once the "pfring" package, and optionally the ZC drivers, is installed following the procedure on the web page, it is possible to use the init script under /etc/init.d/pf_ring to automate the kernel module and drivers loading. The init script acts as follows:

1. loads the pf_ring.ko kernel module.
2. scans folders `/etc/pf_ring/zc/{e1000e,igb,ixgbe,i40e,fm10k}/` searching:
   - `{e1000e,igb,ixgbe,i40e,fm10k}.conf` containing the driver parameters
   - `{e1000e,igb,ixgbe,i40e,fm10k}.start` that should be just an empty file
3. loads the drivers whose corresponding `{e1000e,igb,ixgbe,i40e,fm10k}.start` file is present, unloading the vanilla driver.
4. configures hugepages if a ZC driver has been loaded, reading the configuration from `/etc/pf_ring/hugepages.conf`. Each line (one per CPU) of the configuration file should contain:
```
   node=<NUMA node id> hugepagenumber=<number of pages>
```
4. changes the MTU for the devices addresses listed in `/etc/pf_ring/mtu.conf.` Each line should contain MAC address and MTU:
```
   00:11:22:33:44:55 1500
```

Example of a minimal configuration for a dual-port ixgbe card on a uniprocessor:

```
mkdir -p /etc/pf_ring/zc/ixgbe
echo "RSS=1,1" > /etc/pf_ring/zc/ixgbe/ixgbe.conf
touch /etc/pf_ring/zc/ixgbe/ixgbe.start
echo "node=0 hugepagenumber=1024" > /etc/pf_ring/hugepages.conf
```

In order to run the init script, after all the files have been configured:

```
/etc/init.d/pf_ring start
```

Appendix B
BPF-Like Packet Filtering Expressions

BPF-like filters can be specified using a subset of the BPF syntax.
As the filter expression complexity affects:

- packet capture speed when used for filtering incoming traffic
- index complexity and speed when used for filtering dumped traffic

we will define a set of constraints and allowed expressions.

An expression consists of one or more primitives.
Complex filter expressions are built by using AND, OR and NOT operators.

Allowed qualifiers for primitive expressions:

Protocol: ether, ip, ip6, tcp, udp, sctp
Direction: src, dst, src or dst, src and dst
Type: host, net, port

Allowed qualifiers in index filters when enabling —index-tunnel-contentl-G: inner, outer

Other allowed primitives: vlan

Additional constraints for packet capture filters:

- it is not possible to use the NOT operator.
- it is possible to use up to two level of nesting with parenthesis
- on the same nesting level, and inside the same parenthesis, it is not possible to mix different operators

Tricks to speed up index filtering:

- Use /32 and /24 IPv4
- Use /128 only IPv6
- Use exact ports (no port range)
- Do not use protocol identifiers

Primitive filter examples:

```
ether host 00:11:22:33:44:55
ether src host 00:11:22:33:44:55

ip host 192.168.0.1
ip dst host 192.168.0.1

ip6 host 2001:0db8:85a3:0042:0000:8a2e:0370:7334
ip6 src host 2001:0db8:85a3:0042:0000:8a2e:0370:7334

ip net 192.168.1.0/24
ip src net 192.168.1

port 80
udp port 9000
```

```
tcp src port 80
vlan 32
```

Complex capture filter examples:

```
ip host 192.168.1.1 and 192.168.1.2
ip src 192.168.1.1 and dst 192.168.1.2

ip host 192.168.1.1 and tcp port (80 or 443)
(ip host 192.168.1.1 or 192.168.1.2) and (port 80 or 443)
```

Appendix C
Disk Partitioning and Formatting

n2disk writes files sequentially. In our experience the XFS filesystem is the best option you can select although other filesystems such as EXT4 can also be used. Supposing that you have created the /dev/sda1 disk partition, you can format it as follows:

```
# mkfs.xfs -f -d sunit=128,swidth=1024,agcount=6000 -l size=64m /dev/sda1
```

Once you have formatted the disk you can mount it as follows:

```
# mount -o
noatime,nodiratime,attr2,nobarrier,logbufs=8,logbsize=256k,osyncisdsync /dev/
sda1 /storage/
```

Note that for small partition sizes, you might need to adapt some of the above parameters.

You can test the disk write performance as explained in chapter 4. Please note that depending on the disk types and controller, your write performance can be influenced by the chunk size (-C option) that specifies. the minimal unit of data written to disk. Values in the range between 64 and 512 KBytes should guaranteed adequate performance, although some combination of disks/controllers operate better with 1024 KBytes or more. You can check that with “-e 1”.

Appendix D
Installing n2disk

n2disk needs a license in order to operate permanently (i.e. not in demo mode). In order to obtain a license, you need to go to http://shop.ntop.org and order it. Once you have done that, you need to activate your n2disk license as follows:

1. Go to http://packages.ntop.org and select your platform (currently CentOS and Ubuntu), then download the PF_RING and n2disk packages.
2. Install the packages (as root). First you need to install the PF_RING package then the n2disk package. Example:
    a. Ubuntu
        i. dpkg -i pfring-<version>.deb
        ii. dpkg -i n2disk-<version>.deb
    b. CentOS
        i. rpm -i pfring-<version>.rpm
        ii. rpm -i n2disk-<version>.rpm
3. Identity your system Id

    ```
    # n2disk -V
    n2disk 3.0 [SystemID: 1234567890ABCDEF]
    ```
4. Generate your license using the above system identifier and the credentials you have received.
5. Install the license in /etc/n2disk.license

You can now start your licensed n2disk copy. To start the copy, you can either use the scripts provided under /etc/init.d/ already designed to control the start/stop of the process. Alternatively, you can start n2disk in foreground, or as a daemon by specifying option --daemon.

Remember that for good results, you need to use n2disk over ZC. In order to do that please refer to the PF_RING manual (see http://www.ntop.org/products/pf_ring/).

Appendix E
Hugepages Support

Hugepages can be enabled in n2disk using the -U option. This section describes how to enable hugepages into your system, a mandatory step for running n2disk with hugepages.

For the impatient

In order to reserve and make available 2 GBytes (1024 pages of 2 MBytes) of memory for hugepages without any special NUMA node affinity you can use:

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
mount -t hugetlbfs nodev /mnt/huge
```

The whole story

Linux typically uses memory pages of 4 KBytes, but provides an explicit interface to allocate pages with bigger size called hugepages. It is up to developers/administrators to decide when they have to be used.

Hugepages advantages:

1. Large amounts of physical memory can be reserved for memory allocation, that otherwise would fail especially when physically contiguous memory is required.
2. Reduced overhead: as the TLB (Translation Lookaside Buffer) contains per-page virtual to physical address mappings, using a large amount of memory with the default page size leads to processing overhead for managing the TLB entries.

The default hugepage size is usually 2 MBytes. The hugepage size can be found in /proc/meminfo:

```
cat /proc/meminfo | grep Hugepagesize
  Hugepagesize: 2048 kB
```

Hugepages can be dinamically reserved with:

```
echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
```

The above pages are allocated by the system without node affinity. If you want to force allocation on a specific NUMA node you have to do:

```
echo 1024 > /sys/devices/system/node/node0/hugepages/
hugepages-2048kB/nr_hugepages
echo 1024 > /sys/devices/system/node/node1/hugepages/
hugepages-2048kB/nr_hugepages
```

It is possible to change the default hugepages size and reserve large amounts of memory at boot time using the following kernel options:

```
default_hugepagesz=1G hugepagesz=1G hugepages=4
```

If this commands returns a non-empty string, 2MB pages are supported:

```
cat /proc/cpuinfo | grep pse
```

If this commands returns a non-empty string, 1GB pages are supported:

```
cat /proc/cpuinfo | grep pdpe1gb
```

In order to make the hugepages available for use, Linux provides a RAM-based filesystem called "hugetlbfs" that have to be mount'ed with:

```
mount -t hugetlbfs none /mnt/hugepages
```

With no options the default hugepage size is used. To use a different size it is possible to specify the "pagesize=" option. In order to control the maximum amount of memory bound to a mount point it is possible to specify the "size=" option (size is rounded down to the nearest hugepage size).
Example:

```
mount -t hugetlbfs -o pagesize=1G,size=2G none /mnt/hugepages
```

It is possible to see what pages are currently in use using the following command:

```
cat /sys/devices/system/node/node*/meminfo | grep Huge
Node 0 HugePages_Total:  1024
Node 0 HugePages_Free:   1024
Node 0 HugePages_Surp:      0
```

Appendix F
n2disk requirements on Napatech cards

Specific setup is needed in order to have n2disk working with the highest performances with Napatech (www.napatech.com) cards.
Once installed Napatech software, user should customise the “/opt/napatech3/config/ntservice.ini” modifying the parameter as follows:

TimestampFormat = PCAP

if needed standard pcap timestamp

TimestampFormat = PCAP_NS

if needed nanosecond pcap timestamp

Packet descriptor should be set as follows:

PacketDescriptor = PCAP

Segment size should be set to maximum 4MB:

HostBufferSegmentSizeRx = 4

Optionally, user can decide to increase RX host buffer to 128 MB

HostBuffersRx = [4,128,0]

Once completed the customisation, Napatech driver can be started with:

```
    $ /opt/napatech3/bin/ntstart.sh
```

Upon successful driver startup, PF_RING driver can be loaded using the specific pf_ring startup script:

```
    $ /etc/init.d/pfring start
```

After this command, user can start using n2disk for example in a setup like this:

```
    $ n2disk -o /storage -p 1000 -b 4000 -i nt:0 -q 1 -C 1024 -c
1 -z 2 -w 3 -n 50 -m 100
```

where nt:0 means Napatech port 0.

The Napatech card can be instructed to create a stream bound to two ports, then it is possible to open the stream in n2disk. Example using the nptl tool (under /opt/napatech/bin/):

```
     $ ntpl -e "Assign[streamid=0] = port == 0,1"

     $ n2disk -o /storage -p 1000 -b 4000 -i nt:stream0 -q 1 -C
1024 -c 1 -z 2 -w 3 -n 50 -m 100
```

Appendix G

LICENSE

Licensee's use of this software is conditioned upon acceptance of the terms specified in https://svn.ntop.org/svn/ntop/trunk/legal/LicenseAgreement