

n2disk™

Ultra-high-speed packet recorder with realtime indexing.



User Guide

n2disk v.2.7

March 2017

© 2002-17



1. Changelog	4
2. Introduction	5
3. Main Features	6
Dump Format	6
Packet Capture Filters	6
Index and Post-Capture Filters	7
Timeline	7
4. Expected n2disk Performance	8
5. Versions	9
n2disk1g	9
n2disk5g	9
n2disk	9
6. Usage	10
Command Line Options	10
7. Tuning n2disk performance	13
Examples	13
8. Utilities	14
disk2n	14
npcapindex	16
npcapextract	17
npcapdecompress	18
Appendix A	19
PF_RING Support	19
PF_RING kernel module installation	19
Running n2disk on top of PF_RING	19
ZC drivers	19
Configuring a PF_RING Deb/RPM package	20
Appendix B	21
BPF-Like Packet Filtering Expressions	21
Appendix C	23
Disk Partitioning and Formatting	23
Appendix D	24
Installing n2disk	24
Appendix E	25
Hugepages Support	25
For the impatient	25

The whole story	25
Appendix F	27
.....	
n2disk requirements on Napatech cards	27
.....	
Appendix G	28

1. Changelog

- n2disk (recording)
 - Cento integration for metadata import (including L7 proto and flow-ID)
 - Added L7 protocol support to the index (when used in combination with Cento)
 - New flow-based index (-1 2) including support for flow-ID (64-bit)
 - New --not-promisc|-3 flag to capture traffic without promisc mode
 - New --capture-direction|-2 for specifying the capture direction
 - New --packet-slicing <level> option for cutting packets after the specified header
 - Extended -n/-m options: -n/-m -1 means unlimited number of folders/files
 - Support for Ubuntu 16
 - Removed n2disk10gdna, n2disk10gzc is now n2disk
- npcapextract (extraction)
 - Extended Fast-BPF filters with L7 support (syntax: l7proto <nDPI protocol>)
 - New -g option to set core affinity for the extraction thread
 - New -s option to set extraction snaplen
 - Filtering improvements: falling back to standard BPF in case of estraction filter not supported by Fast-BPF
 - New -O option to write pcap to stdout (i.e. pipeline to tshark -i/ wireshark -k -i -)
 - New -0 option to write an empty file on empty result (useful with -O))
 - Support for legacy and new index (both standard with L7 support and flow-based index)
 - Improved extraction with O_DIRECT support
 - Compressed .npcap extraction fix
 - Index file descriptors leak fix
 - Memory leak fix
- Tools
 - New n2membenchmark tool for benchmarking system performance

2. 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 specialized tools like Snort.
- Reconstruct particular communication flows or network activities.
- Reproduce the previous captured traffic to a different network interface.

3. 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.
- Optimized 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 parallelize packet capture using multiple threads. The communication between threads has been carefully optimized.
- 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.

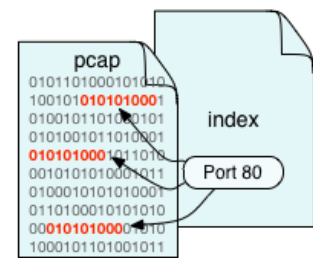
Packet Capture Filters

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.

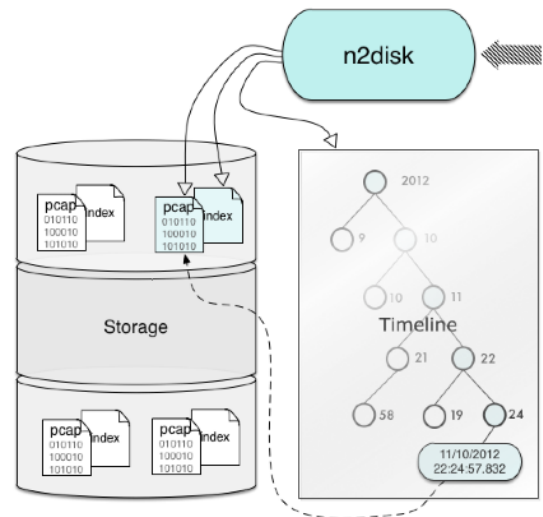
Index and Post-Capture Filters

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.



4. 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.

5. Versions

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

n2disk1g

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

n2disk5g

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

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.

6. 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

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

FILTERING

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

TIMESTAMPING

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

DUMP SETTINGS

<code>--chunk-len -C</code> <len>	Size (KB) of the chunk written to disk (must be multiple of 4096). Default: 64 KB.
<code>--journaling -j</code>	Enable journaling.
<code>--buffer-len -b</code> <len>	Buffer length (MBytes).
<code>--dump-directory -o</code> <dir>	Directory where dump files will be saved.
<code>--archive-directory -O</code> <dir>	Directory where dump files will be archived (slower disks). The -a option is overwritten when using it.
<code>--max-file-len -p</code> <len>	Max pcap file length (MBytes).
<code>--max-file-duration -t</code> <secs>	Max pcap file duration (sec).
<code>--max-file-packets -N</code> <num>	Max packets per pcap file. Default: 0 (no max duration).
<code>--archive -a</code>	Archive pcap file (rename to .old) instead of overwriting if already present on disk.
<code>--max-num-files -m</code> <num>	Max number of files before restarting file name.
<code>--max-nested-dirs -n</code> <num>	Max number of nested dump sub-directories.
<code>--file-prefix -x</code> <prefix>	Dump file prefix.
<code>--hugepages -U</code>	Use hugepages for memory allocation.
<code>--disable-direct-io -r</code>	Disable Direct I/O (experts only).
<code>--stop-on-limit -L</code>	Use max len/duration/packets as limit to stop capture.
<code>--remove-ahead -H</code>	Delete in advance dumped files that will soon be overwritten. Use this option to reduce file deletion cost.
<code>--pcap-compression -M</code>	Compress pcap files (produces .npcap compressed files)

INDEX SETTINGS

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

MULTITHREADING (n2disk only)

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

CPU AFFINITY

<code>[--reader-cpu-affinity -c] <id></code>	Bind the reader thread to the specified core.
<code>[--writer-cpu-affinity -w] <id></code>	Bind the writer thread to the specified core.
<code>[--compressor-cpu-affinity -z] <ids></code>	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)

OTHER

<code>[--simulation-mode -e] <mode></code>	Simulation mode (debug only). 1 - Test max dump speed (simulate capture). 2 - Test max capture speed (simulate storage). 3 - Test max capture speed (simulate storage, produce real index).
<code>[--daemon]</code>	Daemonize this app at startup
<code>[--unprivileged-user -u] <user></code>	Use <user> instead of nobody when dropping privileges.
<code>[--pid -P] <path></code>	Save the pid into the specified path.
<code>[--version -V]</code>	Print application version.
<code>[--help -h]</code>	Help.
<code>[--verbose -v]</code>	Verbose.
<code>[--show-system-id]</code>	Print the system identifier.
<code>[--check-license]</code>	Checks if the license is present and valid.
<code>[--check-maintenance]</code>	Checks the maintenance status for the specified license
<code>[--syslog -l]</code>	Dump trace messages to syslog.
<code>[--event-log -Q] <file></code>	Save relevant events (i.e. drops) onto the specified file.

7. 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
```

Multithreaded packet capture (3 threads):

```
./n2disk -i zc:eth1 -o /storage/eth1/ -b 1024 -C 1024 -p 512 -q
1 -S 0 -c 1 -R 3,4,5 -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
```

8. 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.

n2disk is multithreaded and uses 3 threads: one for reading packets from disk, one for packet transmission and one for precise time generation.

TX SETTINGS

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

BUFFERING SETTINGS

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

CPU AFFINITY

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

PACKET REFORGING

<code>--src-mac] <mac></code>	Reforge source MAC address.
<code>--dst-mac] <mac></code>	Reforge destination MAC address.
<code>--src-ip] <ip></code>	Reforge source IP address.

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

OTHER

<code>[--simulation-mode -e]</code>	Simulation mode (disk simulation, synthetic traffic is generated).
<code>[--unprivileged-user -u] <username></code>	Use <username> instead of nobody when dropping privileges.
<code>[--pid -P] <path></code>	Save the pid into the specified path.
<code>[--version -V]</code>	Print application version.
<code>[--help -h]</code>	Help.
<code>[--verbose -v]</code>	Verbose.
<code>[--show-system-id]</code>	Print the system identifier.
<code>[--check-license]</code>	Checks if the license is present and valid.
<code>[--syslog -l]</code>	Dump trace messages to syslog.
<code>[--event-log -Q] <file></code>	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:

[-v]	Verbose
[-h]	Help
[-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
[-l] <version>	Specify the index version:
	1 - Standard index (default)
	2 - Flow-based index

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

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

FIND FROM SINGLE FILE/INDEX

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

FILTERING

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

DUMP SETTINGS

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

OTHER

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

Examples:

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

```
./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"
```

npcapdecompress

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

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

Example:

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

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 (default: 4096)
- *enable_tx_capture*: 1 to capture outgoing packets, 0 otherwise (default: 1)
- *enable_ip_defrag*: 1 to enable IP defragmentation, 0 otherwise (default 0)
- *quick_mode*: 1 to run at full speed but with up to one socket per interface (default: 0)

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

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 guarantee 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 2.0 [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 pdpelgb
```

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

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/):

```
$ nptl -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>