ntopng User's Guide

High-Speed Web-based Traffic Analysis and Flow Collection

Version 1.2
August 2014

1.Table of Contents

1.1. What's New?

Release 1.2 (August 2014)

- Fixed some bugs that caused crashes in particular on 32 bit platforms.
- Updated web GUI with the use of Bootstrap 3 and many new reports.
- Added support for system probe analysis (via nProbe with processPlugin).
- Interface names are now identified with a symbolic name that can be changed by the user. Network interfaces can be enabled/disabled at runtime.
- Starting ntopng without -i now causes it to open all the network interfaces present on the system.
- Added support for hardware timestamped packets produced by IXIA devices (--hw-timestamp-mode <mode>).
- Added coded to reconnect to redis automatically in case of redis restart.
- Added changes for running ntopng on SecurityOnion.
- Added -W flag for enable the HTTPS Server on a specific port.
- Updated -i flag for aggregate multiple collectors traffic, specifying multiple pcap file simultaneously and aggregating their traffic.
- Added --json-label flag for using labels instead of numbers are used as keys to saving flows.
- Added -I flag for exporting flows using the specified ZMQ endpoint in order to create and hierarchies of ntopng Instances.
- Added -A flag for data aggregations for clustering information based on homogeneous information.
- Added -g flag for bind the capture/processing thread to a specific CPU Core (Linux only).
- Added the concept of runtime preferences and added a new menu in the web GUI for handling them.
- Added -k flag for enable the http:bl service[6] that can be used to trigger alerts for hosts that have been put on blacklist based on their bad behaviour.
- Added alerts system with many customisable thresholds that can be logged on syslog.
- Extended host reporting information with new reports and enhancements to existing ones.
- Added the Historical Interface for loading flows saved in SQLite format, specifying an time and date interval.
- Added VLAN support.
- Performance improvements both in nDPI and the ntopng engine.
- Improved host correlation techniques.

Release 1.1 (November 2013)

- Updated web GUI.
- Updated -i flag for specifying multiple interfaces simultaneously.
- Added -F flag for saving flows in SQLite format.
- Added -A flag for data aggregations for clustering information based on homogeneous information.
- Added activity map for having at 1-second visibility of hosts activities.
- Extended host reporting information with new reports and enhancements to existing ones.

- Added Google Maps support and HTML 5 map geolocation support.
- Performance improvements both in nDPI and the ntopng engine.
- Implemented passive OS detection by dissecting, via nDPI, HTTP request headers.
- All objects are JSON-friendly.

Release 1.0 (May 2013)

- First public release.

2.It's time for a completely new ntop.

15 years are past since the first version of ntop. In 1998 network monitoring requirements were very different from today: few protocols (mostly in plain text) to monitor, IP was not yet "the only protocol", low network speed, very few connected hosts, no iPhones yet, raspberry was still a fruit, Linux was still for geeks. In 2013 the whole picture is very different.
One gigabit links are now commodity (10 Gbit is around the corner), (too?) many hosts interconnected and mobile, application protocols (e.g. Spotify or Skype) are "the" protocols (TCP is a generic protocol) so we need nDPI to figure out what is happening on the network.

The way the original ntop was designed was IMHO very advanced for that time, but today is no longer so for many reasons. Today people want to have a flexible network monitoring engine able to scale at multi-Gbit, using limited memory, immune to crashes "no matters what", scriptable and extensible, able to see what's happening in realtime with 1-second accuracy, capable of characterising hosts (call it host reputation if you wish) and storing monitoring data on the cloud for (de-)centralised monitoring even of those devices that have no disk space. Over the past years we have tried to address ntop open issues, but the code base was too old, complicated, bug-prone. In essence it was time to start over, preserve the good things of ntop, and learn from mistakes. So basically looking forward by creating a new ntop, able to survive (hopefully) 15 more years and set new monitoring standards.

This has the motivation behind what I temporarily call ntopng (ntop next generation). The work to do is huge but as you can see many things are already working.

3. Introduction

ntopng is the next generation version of the original ntop, a network traffic probe that shows the network usage, similar to what the popular top Unix command does. It displays a list of hosts that are currently using the network and reports information concerning the (IP and non-IP) traffic generated and received by each host. ntopng may operate as a front-end collector or as a stand-alone collector/display program.

ntopng is based on libpcap and it has been written in a portable way in order to virtually run on every Unix platform, MacOSX and on Win32 as well.

ntopng is a network traffic monitor, by default it uses the layer 2 Media Access Control (MAC) addresses AND the layer 3 tcp/ip addresses. ntopng is capable of associating the two, so that ip and non-ip traffic (e.g. arp, rarp) are combined for a complete picture of network activity.

ntopng users must use a web browser to navigate through ntopng (that acts as a web server) traffic information and get a dump of the network status. In the latter case, ntopng can be seen as a simple RMON-like agent with an embedded web interface. The use of:

- a web interface.
- limited configuration and administration via the web interface.
- reduced CPU and memory usage (they vary according to network size and traffic).

3.1. The main design principles

- Open source, self-contained with zero configuration, just like the original ntop.
- ntopng is a cache, just like the original ntop, but contrary to its predecessor we leverage on Redis for implementing multi-level caching:
- ntopng keeps in memory the current network traffic
- Redis keeps the "recent network history"
- (Optionally) Persistently dump traffic history on disk for long term traffic analysis.
- nDPI centric: ports are no longer enough, as we want to identify application protocols even on non standard ports.
- Ability to leverage on PF_RING for monitoring million packets/second with no drops.
- Written in C++, with clean code layout. Occasionally some routines from the original ntop will be ported to ntopng, but the idea is to write everything from scratch on a clean room. The ntop code didn't have a real API and it was so complicated after years of patches, that people were scared of touching me.
- The web GUI is based on Twitter Bootstrap for modern, consistent, and mobile-friendly GUI.
- The ntopng engine is scriptable in LuaJIT.
- Web pages are written in Lua: everyone can write its on pages without having to code in C.
- ntopng, as well ntopng, leverage on the MicroCloud for creating a comprehensive network view.

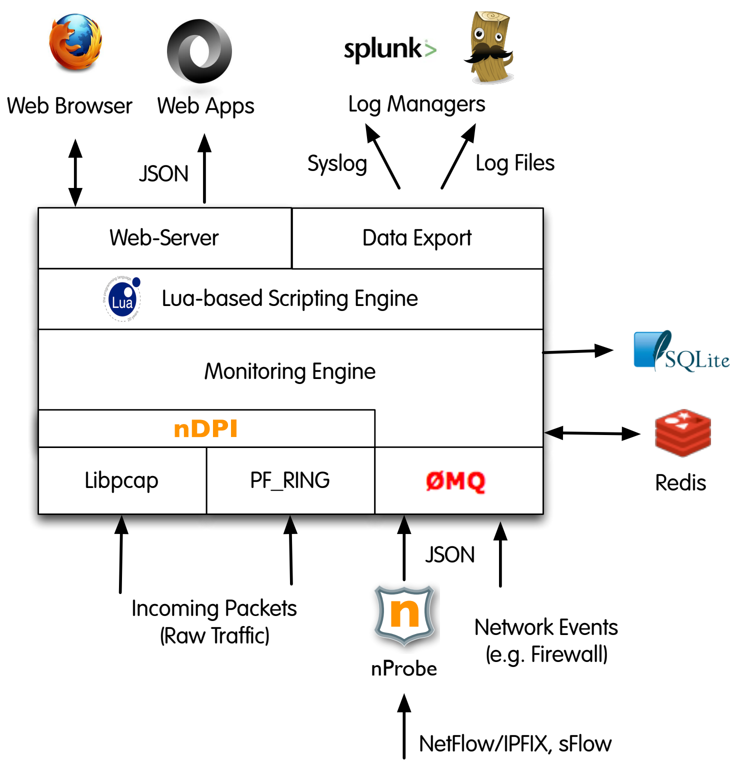
3.2. What ntopng can do for me?

- Sort network traffic according to many protocols
- Show network traffic and IPv4/v6 active hosts
- Store on disk persistent traffic statistics in RRD format
- Geolocate hosts
- Discover application protocols by leveraging on nDPI, ntop's DPI framework.
- Characterise HTTP traffic by leveraging on characterisation services provided by block.si. ntopng comes with a demo characterisation key, but if you need a permanent one, please mail info@block.si.
- Show IP traffic distribution among the various protocols
- Analyse IP traffic and sort it according to the source/destination
- Display IP Traffic Subnet matrix (who's talking to who?)
- Report IP protocol usage sorted by protocol type

Platforms	• Unix (including Linux, *BSD, and MacOSX) • Windows (including the latest Windows 7/8)
Web GUI	A modern HTML 5 browser is needed to visualise ntopng traffic statistics.
Requirements	• Memory Usage It depends on the ntop configuration, number of hosts, and number of active TCP sessions. In general it ranges from a few MB (little LAN) to 100 MB for a WAN. • CPU Usage It depends on the ntop configuration, and traffic conditions. On a modern PC and large LAN, it is less than 10% of overall CPU load.
Protocols	• IPv4/IPv6 • All IP protocols supported by nDPI (~170 and counting) • …and many more
Extensibility	ntopng engine is scripted using the LuaJIT language. Users can extend the web interface as well modify it in realtime without having to code into the ntopng C++ engine.
Additional Features	• sFlow, NetFlow (including v5 and v9) and IPFIX support through ntopng • Network Flows • Local Traffic Analysis • Lua lightweight API for extending ntop via scripts • Support of both NetFlow andsFlow as flow collector. ntop can collect simultaneously from multiple probes. • Traffic statistics are saved into RRD databases for long-run traffic analysis. • Internet Domain, AS (Autonomous Systems), VLAN (Virtual LAN) Statistics. • Protocol decoders for all application protocols supported by nDPI. • Advanced HTTP password protection with encrypted passwords • RRD support for persistently storing per-host traffic information

3.3. ntopng Architecture

The figure below depicts the may ntop components



ntopng Monitoring Engine

- Coded in C++ and based the concept of flow (set of packets with the same 6-tuple).
- Flows are inspected with a home-grown DPI- library named nDPI aiming to discover the "real" application protocol (no ports are used).
- Information is clustered per:
  - (Capture) Network Device
  - Flow
  - Host
  - High-level Aggregations

Information Lifecycle

- All information (e.g. hosts and flows) is stored in memory.
- Using command line options, users can specify how many hosts/flows can be kept in memory.
- Idle flows are periodically purged in order to free memory.
- Hosts are serialised and stored in JSON format in Redis for 1 hour, so that in case new traffic is detected ntopng can restore them from cache.

Packet Processing Journey

- Packet capture: PF_RING (Linux) or libpcap.
- Packet decoding: no IP traffic is accounted.
- IPv4/v6 Traffic only:
  - Map the packet to a 6-tuple flow and increment stats.
  - Identify source/destination hosts and increment stats.
  - Use nDPI to identify the flow application protocol
    - UDP flows are identified in no more than 2 packets.
    - TCP Flows can be identified in up to 15 packets in total, otherwise the flow is marked as "Unknown".
- Move to the next packet.

3.4. Download ntopng

Have a look at the download page.

4.Using ntopng

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

4.1. Compiling ntopng Source Code

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

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

and it can be installed as follow:

```
make install
make uninstall
```

Please note that the ntopng source code compiles both on Unix and Windows. If you are looking for the main dependence and how to install it please read the appendix A at the end of this document.

4.2. Installing a Binary ntopng

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://packages.ntop.org

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.

- OSX

  http://sourceforge.net/projects/ntop/files/ntopng/
  Homebrew
  ```
  brew update
  brew install ntopng
  ```

- Windows

  http://shop.ntop.org or build it from source

4.3. ntopng Command Line Options

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

```
# ntopng -h
ntopng x86_64 v.1.1.99 (r8109) - (C) 1998-14 ntop.org

Usage:
  ntopng <configuration file>
  or
  ntopng [-m <local nets>] [-d <data dir>] [-e] [-n mode] [-i <iface|pcap file>]
         [-w <http port>] [-W <https port>] [-p <protos>] [-P] [-d <path>]
         [-c <categorization key>] [-k <httpbl key>] [-r <redis>]
         [-l] [-U <sys user>] [-s] [-v] [-C]
         [-F] [-D <mode>] [-E <mode>]
         [-B <filter>] [-A <mode>]

Options:
[--dns-mode|-n] <mode>              | DNS address resolution mode
                                    | 0 - Decode DNS responses and resolve
                                    |     local numeric IPs only (default)
                                    | 1 - Decode DNS responses and resolve all
                                    |     numeric IPs
                                    | 2 - Decode DNS responses and don't
                                    |     resolve numeric IPs
                                    | 3 - Don't decode DNS responses and don't
                                    |     resolve numeric IPs
[--interface|-i] <interface|pcap>   | Input interface name (numeric/symbolic)
                                    | or pcap file path
[--data-dir|-d] <path>              | Data directory (must be writable).
                                    | Default: /var/tmp/ntopng
[--daemon|-e]                       | Daemonize ntopng
[--httpdocs-dir|-1] <path>          | HTTP documents root directory.
                                    | Default: httpdocs
[--scripts-dir|-2] <path>           | Scripts directory.
                                    | Default: scripts
[--callbacks-dir|-3] <path>         | Callbacks directory.
                                    | Default: scripts/callbacks
[--dump-timeline|-C]                | Enable timeline dump.
[--categorization-key|-c] <key>     | Key used to access host categorization
                                    | services (default: disabled).
                                    | Please read README.categorization for
                                    | more info.
[--httpbl-key|-k] <key>             | Key used to access httpbl
                                    | services (default: disabled).
                                    | Please read README.httpbl for
                                    | more info.
[--http-port|-w] <http port>        | HTTP port. Set to 0 to disable http server. Default: 3000
[--https-port|-W] <http port>       | HTTPS port. Default: 3001
[--local-networks|-m] <local nets>  | Local nets list (default: 192.168.1.0/24)
                                    | (e.g. -m "192.168.0.0/24,172.16.0.0/16")
[--ndpi-protocols|-p] <file>.protos | Specify a nDPI protocol file
                                    | (eg. protos.txt)
[--disable-host-persistency|-P]     | Disable host persistency
[--redis|-r] <redis host[:port]>    | Redis host[:port]
[--user|-U] <sys user>              | Run ntopng with the specified user
                                    | instead of nobody
[--dont-change-user|-s]             | Do not change user (debug only)
[--disable-autologout|-q]           | Disable web interface logout for inactivity
[--disable-login|-l]                | Disable user login authentication
[--max-num-flows|-X] <num>          | Max number of active flows
                                    | (default: 131072)
[--max-num-hosts|-x] <num>          | Max number of active hosts
                                    | (default: 65536)
[--users-file|-u] <path>            | Users configuration file path
                                    | Default: ntopng-users.conf
[--pid|-G] <path>                   | Pid file path
[--disable-alerts|-H]               | Disable alerts generation
[--packet-filter|-B] <filter>       | Ingress packet filter (BPF filter)
[--enable-aggregations|-A] <mode>   | Setup data aggregation:
                                    | 0 - No aggregations (default)
                                    | 1 - Enable aggregations, no timeline dump
                                    | 2 - Enable aggregations, with timeline
                                    |     dump (see -C)
[--dump-flows|-F]                   | Dump expired flows.
[--export-flows|-I] <endpoint>      | Export flows using the specified endpoint.
[--dump-hosts|-D] <mode>            | Dump hosts policy (default: none).
                                    | Values:
                                    | all    - Dump all hosts
                                    | local  - Dump only local hosts
                                    | remote - Dump only remote hosts
[--dump-aggregations|-E] <mode>     | Dump aggregations policy (default: none).
                                    | Values:
                                    | all    - Dump all hosts
                                    | local  - Dump only local hosts
                                    | remote - Dump only remote hosts
[--sticky-hosts|-S] <mode>          | Dont flush hosts (default: none).
                                    | Values:
                                    | all    - Keep all hosts in memory
                                    | local  - Keep only local hosts
```

```
                                  | remote - Keep only remote hosts
                                  | none   - Flush hosts when idle
--json-labels                     | In case JSON label is used (e.g. with
                                  | ZMQ/Sqlite) labels instead of numbers are
                                  | used as keys.
--hw-timestamp-mode <mode>        | Enable hw timestamping/stripping.
                                  | Supported TS modes are:
                                  | ixia - Timestamped packets by ixiacom.com
                                  |        hardware devices
[--verbose|-v]                    | Verbose tracing
[--version|-V]                    | Print version and quit
[--help|-h]                       | Help
```

filename

The text of filename is copied - ignoring line breaks and comment lines (anything following a #) - into the command line. ntopng behaves as if all of the text had simply been typed directly on the command line. For example, if the command line is "ntopng s.conf" and file s.conf contains just the line '-s', then the effective command line is "ntopng -s". In case you use a configuration file, the following options on the command line will be ignored. Example "ntopng /etc/ntopng/ntopng.conf -v" the -v option is ignored.

The configuration file is similar to the command line, with the exception that an equal sign '=' must be used between key and value. Example: -i=p1 or --interface=p1.

Remember, most ntopng options are "sticky", that is they just set an internal flag. Invoking them multiple times doesn't change the ntopng's behavior. However, options that set a value, such as --trace-level, will use the LAST value given: -w 8000 -w 8080 will run as -w 8080.

-n: DNS mode

This specifies the DNS address resolution mode. Ntopng provides four modes:
0 - Decode DNS responses and resolve local numeric IPs only,
1 - Decode DNS responses and resolve all numeric IPs;
2 - Decode DNS responses and don't resolve numeric IPs;
3 - Don't decode DNS responses and don't resolve numeric IPs.
By default the DNS mode is set to 0.

-i: interface

Specifies the network interface or collector endpoint to be used by ntopng for network monitoring.
On Unix you can specify both the interface name (e.g. lo) or the numeric inter-face id as shown by ntopng -h. On Windows you must use the interface number instead.
In case a user needs to activate ntopng on two or more different interfaces, then he/she needs to repeater the -i flag once for interface (e.g -i eth0 -i eth1).

If a collector endpoint is specified, ntopng open a ZeroMQ connection to the specified endpoint as a subscriber. Example of collector endpoints are "tcp://127.0.0.1:5556" or ipc://flows.ipc . Note that you can specify multiple endpoint, commas separated list, in order to instruct ntopng to aggregate it in a single interface. (e.g -i tcp://127.0.0.1:5556,ipc://flows.ipc)

If you want you can pass a path of a pcap file (e.g. -i dummy.pcap) or a path of a list file contains a path of a pcap file for each line (e.g. -i pcap.list) and ntopng will read packets from the specified pcap file/s.

nProbe can be instructed to act as a publisher delivering flows to a ZeroMQ endpoint using the --zmq <endpoint> parameter.

--daemon | -e : start ntopng as a daemon
Useful when starting ntopng as daemon.
This parameter causes ntop to become a daemon, i.e. a task which runs in the background without connection to a specific terminal. To use ntop other than as a casual monitoring tool, you probably will want to use this option.

--data-dir | -d: data directory
Specifies the data directory. It must be writable. Default directory is / var/tmp/ntopng .

--httpdocs-dir | -1: HTTP documents root directory
This specifies the path of HTTP documents root directory. By default it is httpdocs.

--scripts-dir | -2: Scripts directory
This specifies the path of scripts directory. By default it is scripts.

--callbacks-dir | -3: Callbacks directory
This specifies the path of callbacks directory. By default it is scripts/ callbacks.

—disable-alerts | -H
Disable the generation of alerts.

--httpbl-key | -k <key>
Set the key used to access httpbl services (default: disabled). Please read README.httpbl for more info.

--categorization-key | -c <key>
Set key used to access host categorization services (default: disabled). Please read README.categorization for more info.

--http-port | -w <http port>
Sets the HTTP port of the embedded web server. If set to 0, the http server will be disable.

--https-port | -W <https port>
Sets the HTTPS port of the embedded web server. If not set, it will be set to the value of -w plus one.

--local-networks | -m <local nets>
Set the ip addresses and netmasks for each active interface. Any traffic on those networks is considered local. This parameter allows the user to define additional networks and subnetworks whose traffic is also considered local in ntopng reports. All other hosts are considered remote. If not specified the default is set to 192.168.1.0/24.

Commas separate multiple network values. Both netmask and CIDR notation may be used, even mixed together, for instance "131.114.21.0/24,10.0.0.0/255.0.0.0".

--ndpi-protocols|-p <file>.protos
This parameter is used to specify a nDPI protocol file. The format is <tcp|udp>:<port>,<tcp|udp>:<port>,.....@<proto> where <port> is a port number and <proto> is a name of a protocol supported by nDPI protocol, or host:"<string>"@<proto> where string is part of an

host:"<string>"@<proto> where string is part of an host name. As example see https://svn.ntop.org/svn/ntop/trunk/nDPI/example/protos.txt

--disable-host-persistency | -P
Disable host persistency

--redis|-r] <redis host[:port]>
Specifies the redis database host and port. For more information about redis, please refer to http://redis.io/

--user | -U <sys user>
Run ntopng with the specified user instead of nobody

--dont-change-user | -s
Do not change user (debug only)

--disable-login | -l
Disable user login. This is useful for debug purposes or if you want to let everyone access the web gui.

—disable-autologout | -q
Disable web interface logout for inactivity.

--max-num-flows | -X <num>
Specify the maximum number of active flows that ntopng will handle. If more flows are detected they will be discarded (default: 131072)

--max-num-hosts | -x <num>
Specify the maximum number of active hosts that ntopng will handle. If more hosts are detected they will be discarded (default: 65536)

--users-file | -u <path>
Users configuration file path Default: ntopng-users.conf

--pid | -G <path>
Specifies the path where the PID (process ID) is saved.

--packet-filter | -B <filter>
Specifies the packet filter for all packet capture interfaces. For pcap/PF_RING interfaces the filter has to be specified in BPF format (Berkeley Packet Filter).

--enable-aggregations | -A <mode>
Setup data aggregations (e.g. Operating System, DNS etc). The available modes are:
0 - No aggregations (default),
1 - Enable aggregations, no timeline dump on disk,
2 - Enable aggregations, with timeline dump on disk (see -C)

--dump-timeline | -C
Enable timeline dump on disk (default: disabled).

--dump-flows | -F
Dump expired flows. If ntopng is compiled with sqlite support, flows can dumped persistently on disk using this option. Databases are created daily under <data directory>/<interface>db.

--dump-hosts | -D <mode>
Dump hosts policy (default: none). If ntopng is compiled with sqlite support, hosts contacts can dumped persistently on disk using this option. Databases are created daily under <data directory>/<interface>/contacts. This options supports three dump modes:
all - Dump all hosts;
local - Dump only local hosts;
remote - Dump only remote hosts.

```
--dump-aggregations | -E  <mode>
        Dump aggregations policy (default: none). Values:
        all    - Dump all hosts;
        local  - Dump only local hosts;
        remote - Dump only remote hosts.
--sticky-hosts | -S <mode>
        Dont flush hosts (default: none). Values:
        all - Keep all hosts in memory;
        local - Keep only local hosts;
        remote - Keep only remote hosts;
        none - Flush hosts when idle.
--json-labels
        Using this option in case JSON label is used (e.g. with ZMQ/Sqlite)
        labels instead of numbers are used as keys.
--hw-timestamp-mode
        Enable hw timestamping/stripping. Supported TS modes are:
        ixia - Timestamped packets by ixiacom.com hardware devices .
—version | -V
        Print ntopng version and quit.
--verbose | -v
        Verbose tracing
--help | -h
        Help
```

4.4. ntopng on Windows

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

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

Available options:
/i [ntopng options]       - Install ntopng as service
/c [ntopng options]       - Run ntopng on a console
/r                    - Deinstall the service

Example:
Install ntopng as a service:        ntopng /i -i 0 -n 192.168.0.1:2055
Remove the ntopng service:          ntopng /r

Notes:
Type 'ntopng /c -h' to see all options
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.
Services are started/stopped using the Services control panel item.
```

If ntopng is started on the console, the /c flag needs to be used (e.g. ntopng /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 ntopng installer registers ntopng as a service with the default options. If you need to change the ntopng setup, you need to do as follows:

ntopng /r	Remove the service
ntopng /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 ntopng configuration. The association interface name and index is shows typing the ‘ntopng /c -h’

```
C:\ntop\ntopng\Debug>ntopng.exe/c -h
Running ntopng.

Welcome to ntopng v.x.x.x
Built on dd/mm/yy hh:mm:ss
Copyright 2002-14 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 ntopng needs to be started with -i 1 option.

5. API Scripting Lua

In the following sections, we discuss the main API Lua of ntopng.

5.1. Introduction

A design principle of ntopng has been the clean separation of the GUI from engine (in ntop it was all mixed).

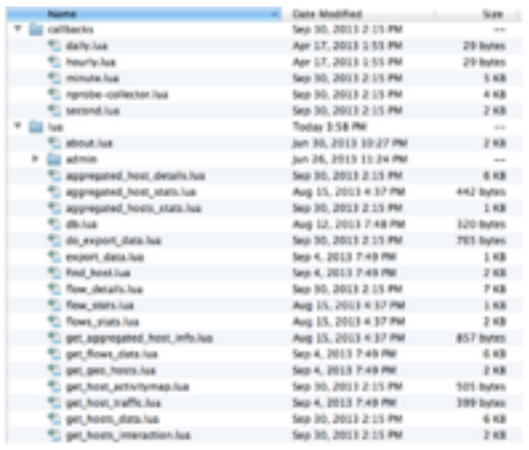
This means that ntopng can (also) be used (via HTTP) to feed data into third party apps such as Nagios or OpenNMS. All data export from the engine happens via Lua API.
Lua methods invoke the ntopng C++ API in order to interact with the monitoring engine.

5.2. How ntopng and Lua working together

The Lua script are divided in two directory:

<ntopng directory>/scripts/callback/
scripts are executed periodically to perform specific actions.

<ntopng directory>/scripts/lua/
scripts are executed only by the web GUI. Example:



Name	Date Modified	Size
callbacks	Sep 30, 2013 2:15 PM	--
daily.lua	Apr 17, 2013 1:55 PM	29 bytes
hourly.lua	Apr 17, 2013 1:55 PM	29 bytes
minute.lua	Sep 30, 2013 2:15 PM	5 KB
nprobe-collector.lua	Sep 30, 2013 2:15 PM	4 KB
second.lua	Sep 30, 2013 2:15 PM	2 KB
lua	Today 3:58 PM	--
about.lua	Jun 30, 2013 10:27 PM	2 KB
admin	Jun 26, 2013 11:24 PM	--
aggregated_host_details.lua	Sep 30, 2013 2:15 PM	6 KB
aggregated_host_stats.lua	Aug 15, 2013 4:37 PM	442 bytes
aggregated_hosts_stats.lua	Sep 30, 2013 2:15 PM	1 KB
db.lua	Aug 12, 2013 7:48 PM	320 bytes
do_export_data.lua	Sep 30, 2013 2:15 PM	705 bytes
export_data.lua	Sep 4, 2013 7:49 PM	1 KB
find_host.lua	Sep 4, 2013 7:49 PM	2 KB
flow_details.lua	Sep 30, 2013 2:15 PM	7 KB
flow_stats.lua	Aug 15, 2013 4:37 PM	1 KB
flows_stats.lua	Aug 15, 2013 4:37 PM	2 KB
get_aggregated_host_info.lua	Aug 15, 2013 4:37 PM	857 bytes
get_flows_data.lua	Sep 4, 2013 7:49 PM	6 KB
get_geo_hosts.lua	Sep 4, 2013 7:49 PM	2 KB
get_host_activitymap.lua	Sep 30, 2013 2:15 PM	505 bytes
get_host_traffic.lua	Sep 4, 2013 7:49 PM	399 bytes
get_hosts_data.lua	Sep 30, 2013 2:15 PM	6 KB
get_hosts_interaction.lua	Sep 30, 2013 2:15 PM	2 KB

http://localhost:3000/lua/flow_stats.lua

Each scripts uses the API Lua to access to interface or ntopng informations.

5.3. Developing with the API Lua

ntopng defines (in C++) two Lua classes:

- interface: This class provides to hook to objects that describe flows and hosts and it allows you to access to live monitoring data.

- ntop: This class provides a set of general functions used to interact with ntopng configuration.

If you are looking for a complete description of the API Lua, please read the appendix B at the end of this document.

- C++ sets their data, Lua reads data (e.g. host.name).
- Some Lua methods (e.g.interface.restoreHost()) can however modify the information stored in the engine.

5.4. Example of Lua scripting

In order to explain the main concept of Lua scripting, we created a few sample script.

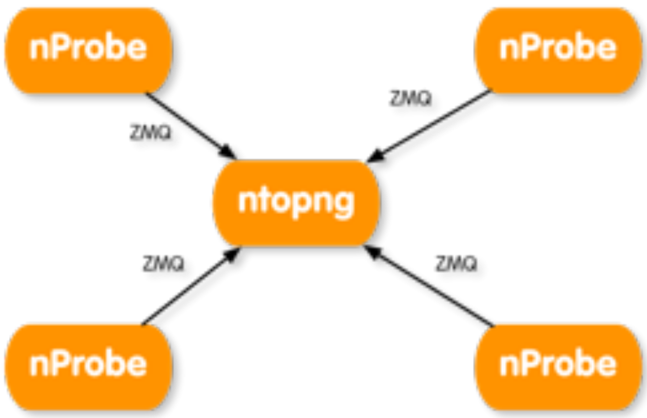
You can find this sample script in <ntopng directory>/scripts/lua/esamples/ and execute each scripts by the web gui (with ntopng active) .

- ntop lua class – http://localhost:3000/lua/examples/ntop.lua
- interface lua class – http://localhost:3000/lua/examples/interface.lua
- debug lua class – http://localhost:3000/lua/examples/debug.lua

6.Use Cases

6.1. Using ntopng as Flow Collector

In ntopng we have decided to collect flows through nprobe that can act as probe/proxy. This is because we wanted to keep the ntopng engine simple and clean from flow-based application needs. The communication between nprobe and ntopng happens through ZeroMQ that decouples ntopng from nprobe.



In order to use ntopng as a flow collector with nprobe you need to start the apps as follows:

1. Start the remote nProbe instances as follows
   - [host1] nprobe –zmq "tcp://*:5556" –i ethX
   - [host2] nprobe –zmq "tcp://*:5556" –i ethX
   - [host3] nprobe –zmq "tcp://*:5556" –i ethX
   - [host4] nprobe –zmq "tcp://*:5556" –i ethX

2. If you want to merge all nProbe traffic into a single ntopng interface do:
   - ntopng –i tcp://host1:5556,tcp://host2:5556,tcp://host3:5556,tcp://host4:5556

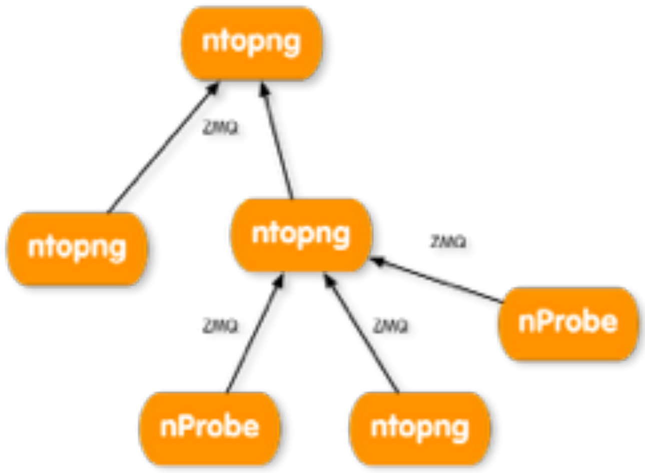
3. If you want to keep each nProbe traffic into a separate ntopng interface do:
   - ntopng –i tcp://host1:5556 –i tcp://host2:5556 –i tcp://host3:5556 –i tcp://host4:5556

Always remember that is ntopng that connects to nProbe instances and polls flows and not the other way round as happens with NetFlow for instance.

Flows exchanged between nprobe and ntopng are formatted in JSON and not on standard sFlow/NetFlow format.

6.2. Creating Hierarchies of ntopng Instances

Suppose that you want to create install a ntopng instance on each of your remote locations, and want to have a ntopng instance that merges the traffic coming from various locations as depicted below.
In order to do this you can use the -I parameter in ntopng to tell a specific ntopng instance to create a ZMQ endpoint to which another ntopng instance can poll traffic flows. Note that you can create this hierarchy by mixing nProbe and ntopng instances, or by using just ntopng instances (this in case



you do not need to handle NetFlow/sFlow flows).

You can create a hierarchy of ntopng's (e.g. on a star topology, where you have many ntopng processes on the edge of a network and a central collector) as follows:

1. Remote ntopng's
   - [Host 1.2.3.4] ntopng -i ethX -I "tcp://*:3456"
   - [Host 1.2.3.5] ntopng -i ethX -I "tcp://*:3457"
   - [Host 1.2.3.6] ntopng -i ethX -I "tcp://*:3458"

2. Central ntopng
   - ntopng -i tcp://1.2.3.4:3456 -i tcp://1.2.3.5:3457 -i tcp://1.2.3.6:3458

Note that:

- On the central ntopng you can also add "-i ethX" if you want the central ntopng to monitor a local interface as well.
- The same ntopng instance (this also applies to nProbe) via -I can serve multiple clients. In essence you can create a fully meshed topology and not just a tree topology.

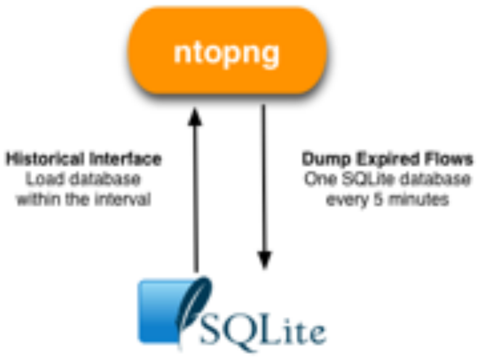
Let's list some use cases:

2. You have a more layered network where each regional office has under it other remote sub-offices.

Please note that the deeper is your hierarchy, the more (likely) the central ntopng will receive traffic from remote peers. So this topology is necessary only if you want to see in realtime what is happening in your network by merging traffic into various locations. If this is not your need, but you just need to supervise the traffic on each remote office from a central location, you can avoid sending all your traffic to the central ntopng and access each remote ntopng instance via HTTP without having to propagate traffic onto a sophisticated hierarchy of instances.

6.3. Load Historical Data

As you know via the option -F you can asked to ntopng to dump expired flows in SQLite format. Now suppose that you have to check the activity of your network in a specific time interval, you can decide to load those SQLite database into a dedicated ntopng interface called "Historical".



Through the Historical Interface configuration page, you can specify the ntopng interface, for which you want to load the historical data, and the date and time interval to be loaded.

Ntopng will load sequentially each SQLite database, previously saved in your data directory, into the time interval for the selected ntopng interface.

Note that:

- Before a new load, ntopng will clean up the Historical interface state.
- Some statistics and error information (i.e number of loaded flows, corrupted or missing files) about the loading process will be show you into the footer.

6.4. Accessing ntopng URLs from command line tools

You need to specify the user and password as specified below (please note the space in the cookie).

Note that you can optionally also specify the interface name.

```
curl --cookie "user=admin; password=admin" "http://127.0.0.1:3000/lua/network_load.lua?ifname=ethX"
```

7. References

1. ntopng, http://www.ntop.org/ntopng.html
2. redis, http://redis.io
3. sqlite, http://www.sqlite.org
4. lua, http://www.lua.org
5. http:bl, http://www.projecthoneypot.org/httpbl.php

8. Tutorials

1. Video, Youtube Channel

Appendix A: Source compilation dependencies

This section explain how to install all ntopng dependencies for the main operation system.
You must install the following packages:

Centos

```
yum groupinstall Development tools
yum install autoconf automake libpcap-devel GeoIP-devel hiredis-devel redis glib2-devel libxml2-devel sqlite-devel gcc-c++ libtool wget
```

Ubuntu/Debian

```
apt-get install build-essential libglib2.0 libxml2-dev libpcap-dev libtool rrdtool autoconf automake autogen redis-server wget sqlite-dev libhiredis-dev libgeoip-dev
```

Mac OSX

```
brew install XXX (Please install homebrew)
```

In any case if you want an updated list of dependencies, you must use the following command:

Centos	yum deplist ntopng
Debian	apt-rdepends ntopng

Appendix B: Doxygen

This section explain how to configure and generate the ntopng documentation as doxygen style.

The ntopng documentation, on Unix it can be compiled as follows:

```
cd <ntopng source code directory>
./configure
cd doc/
doxygen doxygen.conf
```

Or by using the Makefile as follows:

```
cd <ntopng source code directory>
./configure
make docs
```

Documentation is generated using the doxygen tool (1.8 or higher) that uses Graphviz for drawing graphs.

In order to generate the documentation for the Lua API you need to install the Lua plugin for doxygen. You must install the following dependencies:

Debian	apt-get install doxygen graphviz perl lua5.1 lua5.1-doc
Mac OSX	port install XXX (Please install macports)

Once the dependencies has been installed, you can run the following command to install the Lua plugin.

```
cpan App::cpanminus
cpan inc::Module::Install
git clone https://github.com/alecchen/doxygen-lua.git
cd doxygen-lua/
perl Makefile.PL
make
sudo make install
sudo cp bin/lua2dox /opt/local/bin/
```

Appendix C: SSL

In order to use SSL with ntopng (i.e. HTTPS) you need to:

1. Install OpenSSL

   ```
   On OSX do: sudo port install openssl or brew install openssl
   ```

2. Create your SSL certificate

   ```
   make cert
   ```

3. Copy in the directory where ntopng is running the shared libraries needed to enable SSL.

   3.1. Linux

   ```
   ln -s /usr/lib/x86_64-linux-gnu/libssl.so .
   ```

   3.2. OSX

   ```
   ln -s /opt/local/lib/libssl.dylib /opt/local/lib/libcrypto.dylib .
   ```

4. Start ntopng

If you have installed ntopng via the pre-build packages, you can enable SSL as follow:

```
cd /tmp/
openssl req -new -x509 -sha1 -extensions v3_ca -nodes -days 365 -out cert.pem
cat privkey.pem cert.pem > /usr/local/share/ntopng/httpdocs/ssl/ntopng-
cert.pem
/bin/rm -f privkey.pem cert.pem
cd /usr/local/bin/
ln -s /usr/lib/x86_64-linux-gnu/libssl.so .
ntopng
```

If you have installed ntopng via the homebrew formula, you can enable SSL as follow:

```
cd /tmp/
openssl req -new -x509 -sha1 -extensions v3_ca -nodes -days 365 -out cert.pem
cat privkey.pem cert.pem > /usr/local/share/ntopng/httpdocs/ssl/ntopng-
cert.pem
/bin/rm -f privkey.pem cert.pem
cd /usr/local/bin/
ln -s /opt/local/lib/libssl.dylib /opt/local/lib/libcrypto.dylib .
ntopng
```

Appendix D: Data Management

Default ntopng data directory is /var/tmp/ntopng .

It is possible cleanup all historical data and preference (i.e custom interface name, default number of rows on table ecc) as follow:

```
$ redis-cli flushall
$ rm -rf /var/tmp/ntopng/
```

Appendix E: Users Management

Default ntopng user is "admin" with password "admin".

It is possible to provide ntopng a list of users credentials in a text file
with the format:

```
user.user.password=5f4dcc3b5aa765d61d8327deb882cf99
user.user.full_name=Nuovo Utente
user.user.group=administrator
```

The md5 sum of the password has to be provided.
In the same way it is possible to overwrite the default "admin" password.

Users can be added/deleted also from the web gui. Changes are written back to the users file, if any.

It is also possible to add an user via Redis using the command:

```
redis-cli SET user.<username>.password <password md5>
```

Example of adding a user "user" with password "password":

```
$ echo -n "password" | md5
5f4dcc3b5aa765d61d8327deb882cf99
$ redis-cli SET user.user.password 5f4dcc3b5aa765d61d8327deb882cf99
```

It is possible to list all users via Redis using:

```
$ redis-cli KEYS user*
1) "user.admin.password"
2) "user.user.password"
```