High-Speed Network Traffic Monitoring Using ntopng

Luca Deri <deri@ntop.org>
@lucaderi

© 2014 - ntop.org

LISA14
November 9–14, 2014

Outlook

- What are the main activities of ntop.org ?
- ntop's view on network monitoring.
- From ntop to ntopng.
- ntopng architecture and design.
- ntopng as a flow collector
- Exploring system activities using ntopng
- Using ntopng.
- Advanced monitoring with ntopng.
- Future roadmap items.

© 2014 - ntop.org

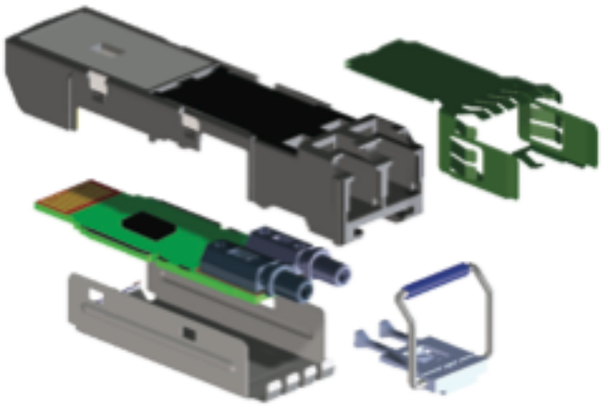
About ntop.org [1/2]

- ntop develops of open source network traffic monitoring applications.
- ntop (circa 1998) is the first app we released and it is a web-based network monitoring application.
- Today our products range from traffic monitoring, high-speed packet processing, deep-packet inspection, and IDS/IPS acceleration (snort and suricata).

© 2014 - ntop.org

About ntop.org [2/2]

- Our software is powering many commercial products...



Integrated ASIC with JDSU technology







ntop Goals

- Provide better, yet price effective, traffic monitoring solution by enabling users to have increased traffic visibility.
- Go beyond standard metrics and increase traffic visibility by analysing key protocols in detail.
- Provide users comprehensive and accurate traffic reports able to offer at a fraction of price what many commercial products do together.
- Promote open-source software, while protecting selected IPRs.

© 2014 - ntop.org

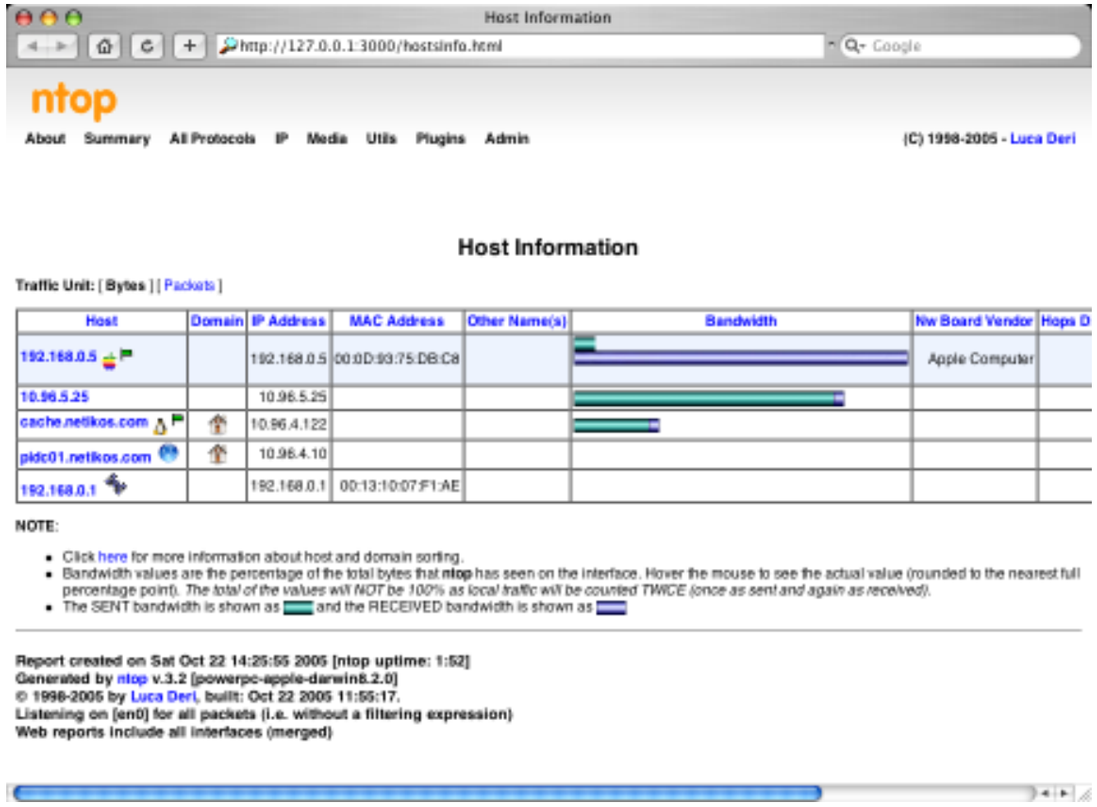
ntop's Approach to Traffic Monitoring

- Ability to capture, process and (optionally) transmit traffic at line rate, any packet size.
- Leverage on modern multi-core/NUMA architectures in order to promote scalability.
- Use commodity hardware for producing affordable, long-living (no vendor lock), scalable (use new hardware by the time it is becoming available) monitoring solutions.
- Use open-source to spread the software, and let the community test it on unchartered places.

© 2014 - ntop.org

Some History

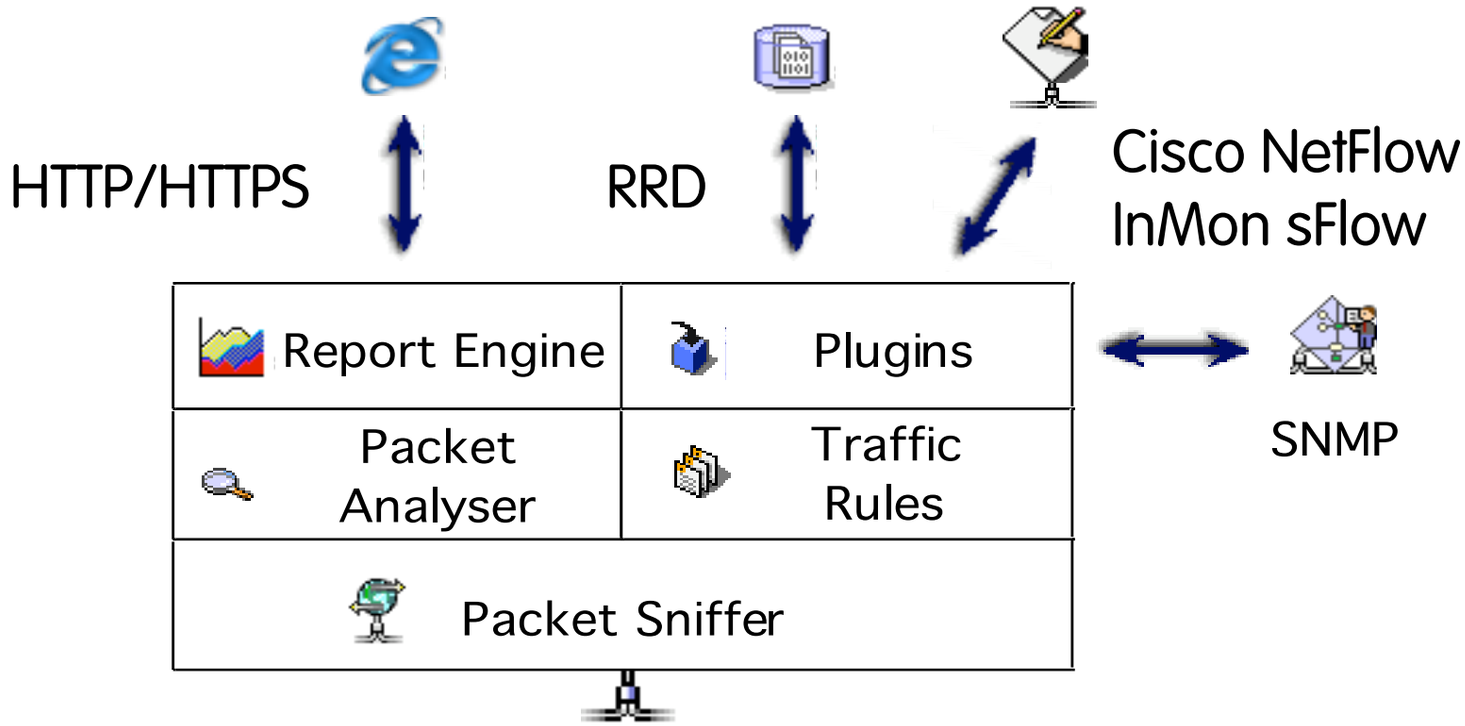
- In 1998, the original ntop has been created.
- It was a C-based app embedding a web server able to capture traffic and analyse it.



- Contrary to many tools available at that time, ntop used a web GUI to report traffic activities.
- It is available for Unix and Windows under GPL.

© 2014 - ntop.org

ntop Architecture





© 2014 - ntop.org

Why was ntop obsolete?

- Its original LAN-oriented design prevented ntop from handling more than a few hundred Mbit.
- The GUI was an old (no fancy HTML 5) monolithic piece written in C so changing/ extending a page required a programmer.
- ntop could not be used as web-less monitoring engine to be integrated with other apps.
- Many components were designed in 1998, and it was time to start over (spaghetti code).

© 2014 - ntop.org

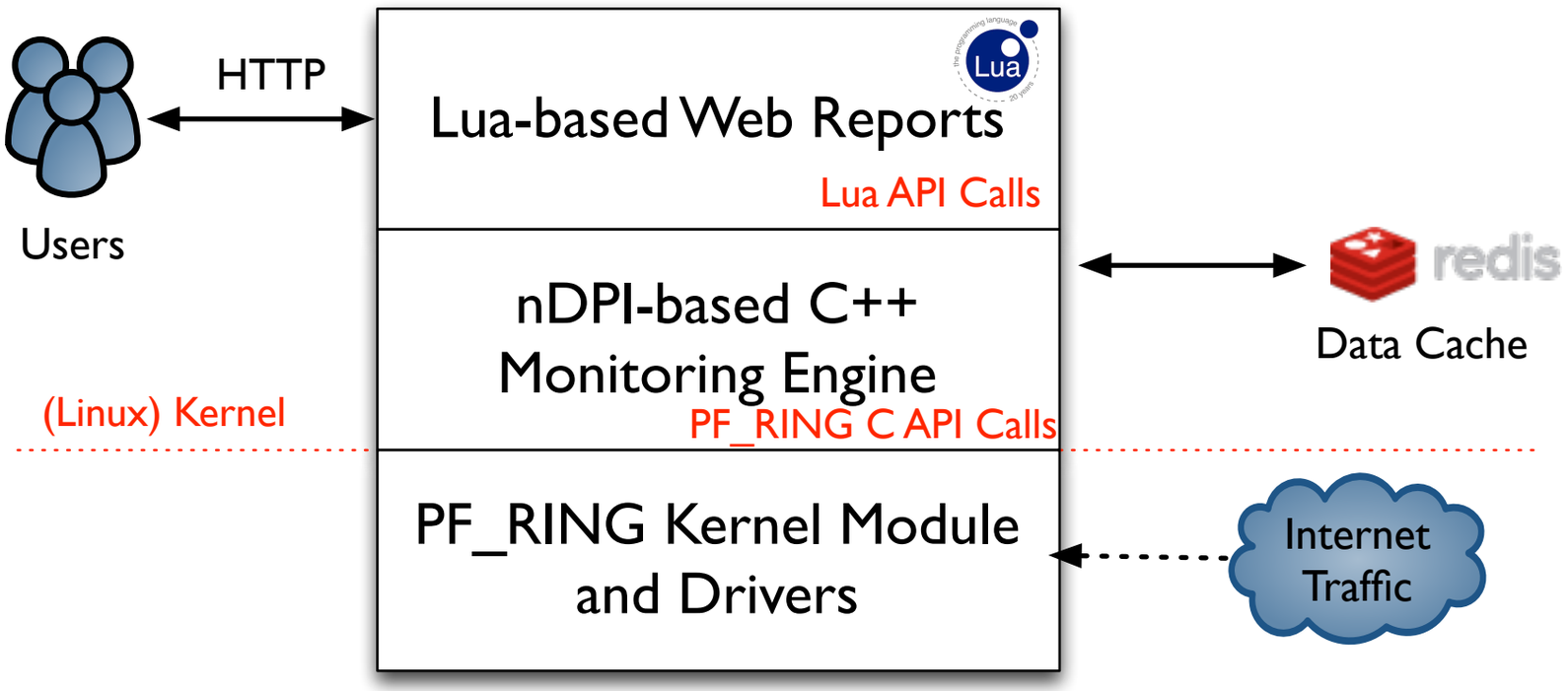
ntopng Design Goals

- Clean separation between the monitoring engine and the reporting facilities.
- Robust, crash-free engine (ntop was not really so).
- Platform scriptability for enabling extensions or changes at runtime without restart.
- Realtime: most monitoring tools aggregate data (5 mins usually) and present it when it's too late.
- Many new features including HTML 5-based dynamic GUI, categorisation, DPI.

© 2014 - ntop.org

ntopng Architecture

- Three different and self-contained components, communicating with clean API calls.



© 2014 - ntop.org

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

© 2014 - ntop.org

Local vs Remote Hosts [1/2]

- ntopng keeps information in memory at different level of accuracy in order to save resources for hosts that are not "too relevant".
- For this reason at startup hosts are divided in:
  - Local hosts
    The local host where ntopng is running as well the hosts belonging to some "privileged" IPv4/v6 networks. These hosts are very relevant and thus ntopng keep full statistics.
  - Remote hosts
    Non-local hosts for which we keep a minimum level of detail.

© 2014 - ntop.org

Local vs Remote Hosts [2/2]

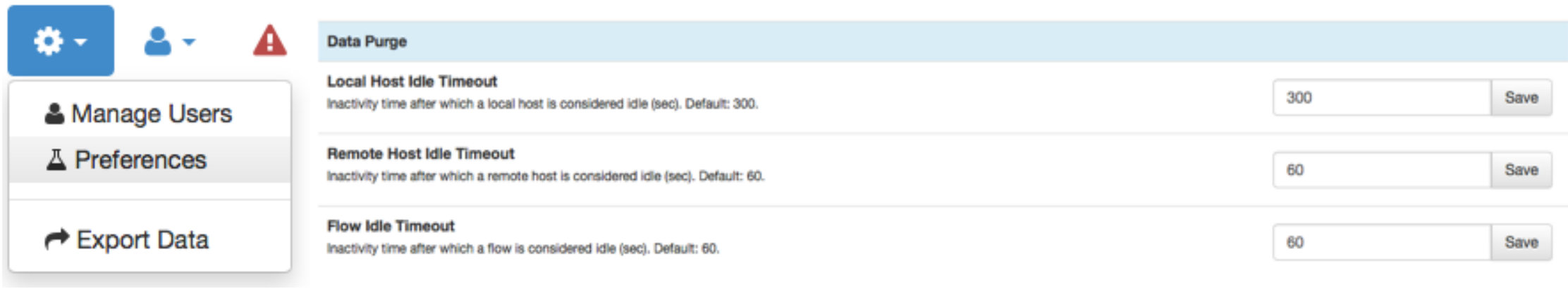
- For local hosts (unless disabled via preferences) are kept all L7 protocol statistics, as well basic statistics (e.g. bytes/packets in/out).
- No persistent statistics are saved on disk.
- A system host is the host where ntopng is running and it is automatically considered local as well the networks of its ethernet interfaces.

IP Address	192.12.193.11 [192.12.193.11/32] [Pisa]
ASN	2597 [Registry of ccTLD it - IIT-CNR]
Name	pc-deri.nic.it Local System

© 2014 - ntop.org

Information Lifecycle

- ntopng keeps in memory live information such as flows and hosts statistics.
- As the memory cannot be infinite, periodically non-recent information is harvested.
- Users can specify preferences for data purge:



© 2014 - ntop.org

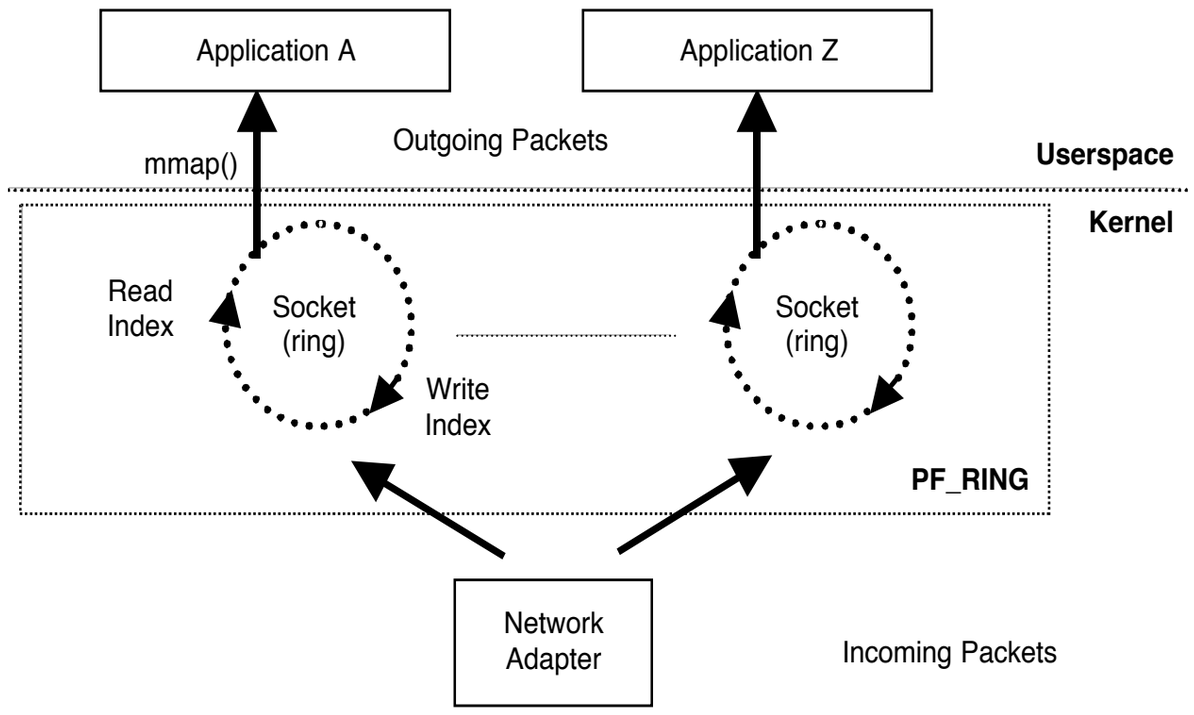
Packet Processing Journey

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

© 2014 - ntop.org

PF_RING [1/2]

- In 2004 we realised the the Linux kernel was not efficient enough to fulfil our packet capture requirements and thus we have written a in-kernel circular buffer named PF_RING.



© 2014 - ntop.org

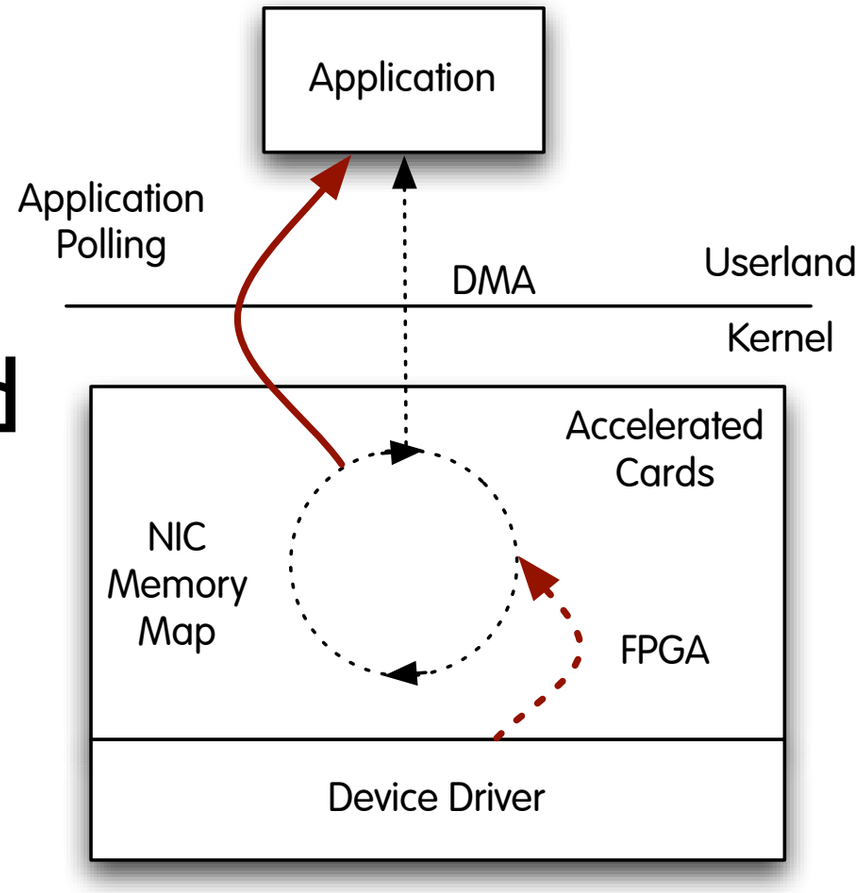
PF_RING [2/2]

- It creates a straight path for incoming packets accessed from user-space applications with memory mapping.
- No need to use custom network cards: any card is supported.
- Transparent to applications: legacy applications need to be recompiled in order to use it (pcap-over-PF_RING).
- Developers familiar with network applications can immediately take advantage of it without having to learn new APIs.
- Acceleration support for many popular open-source applications including Wireshark, Suricata and Snort.

© 2014 - ntop.org

Moving towards 10 Gbit and above [1/2]

- The original PF_RING is a good solution up to 3/5 Gbit but not above as the cost of packet copy into the ring is overkilling.
- PF_RING ZC (Zero Copy) is an extension that allows packets to received/transmitted in zero copy similar to what FPGA-accelerated cards (e.g. Napatech) do in hardware.



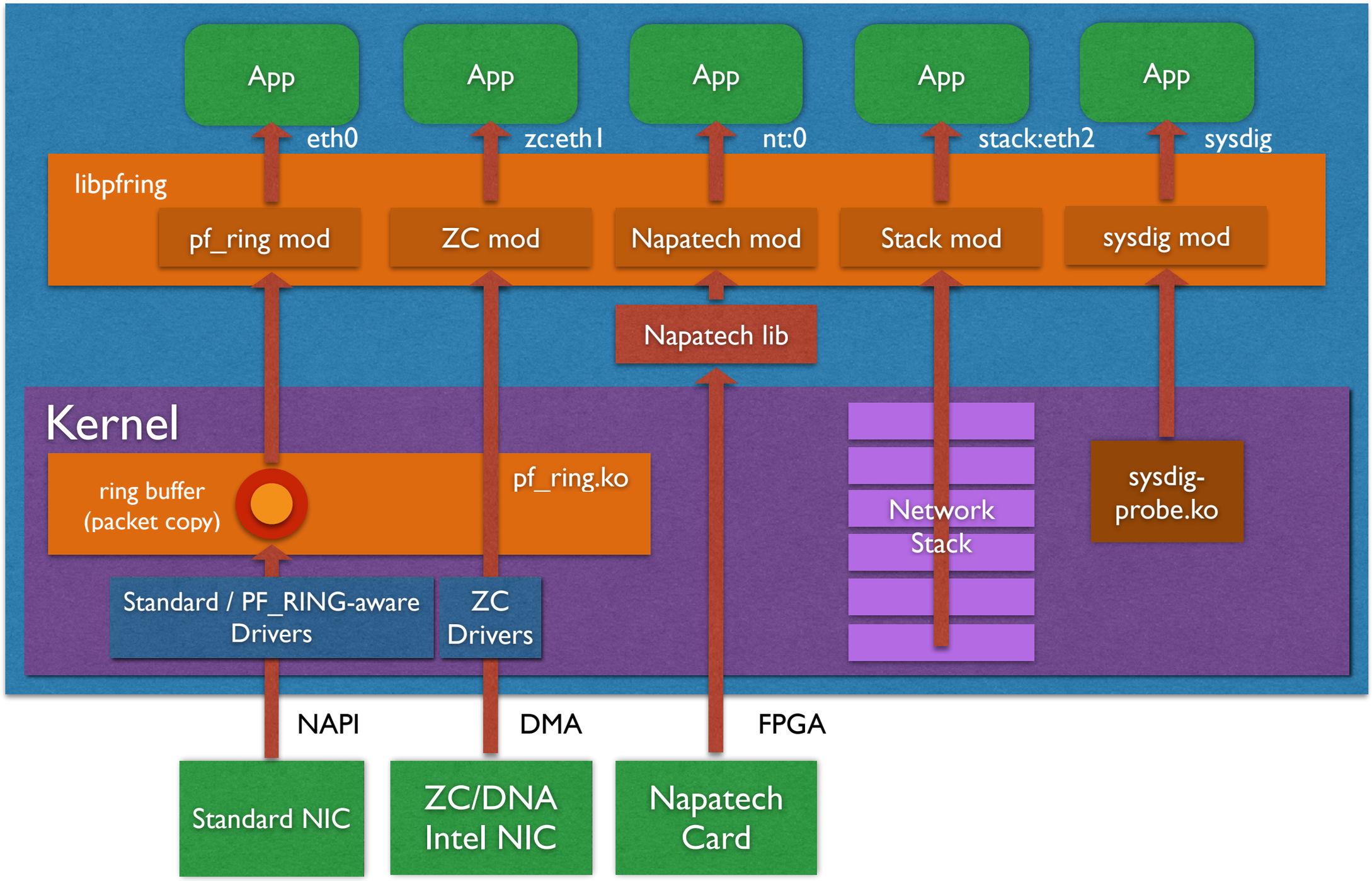
© 2014 - ntop.org

Moving towards 10 Gbit and above [2/2]

- In ZC a packet is put by the ingress NIC into a shared memory buffer, and it hop across applications (and VMs) by exchanging the buffer pointer (packets don't move).
- Thanks to this solution it is possible to create arbitrary packet processing topologies at multi-10 Gbit line rate using commodity hardware x86 servers and adapters (ZC natively supports Intel ethernet adapters).

© 2014 - ntop.org

PF_RING ZC Architecture



© 2014 - ntop.org

PF_RING ZC [1/2]

- The idea behind ZC is to create a playground for processing information (and in particular network packets) in zero copy.
- In order to implements this, ZC comes with 0-copy user-space drivers (for 1 and 10G Intel NICs) that allow packets to be read in 0-copy.
- 1-copy packets (e.g. received on non-Intel NICs or WiFi/Bluetooth devices) can be injected in ZC and from that time onwards, be used in 0-copy.

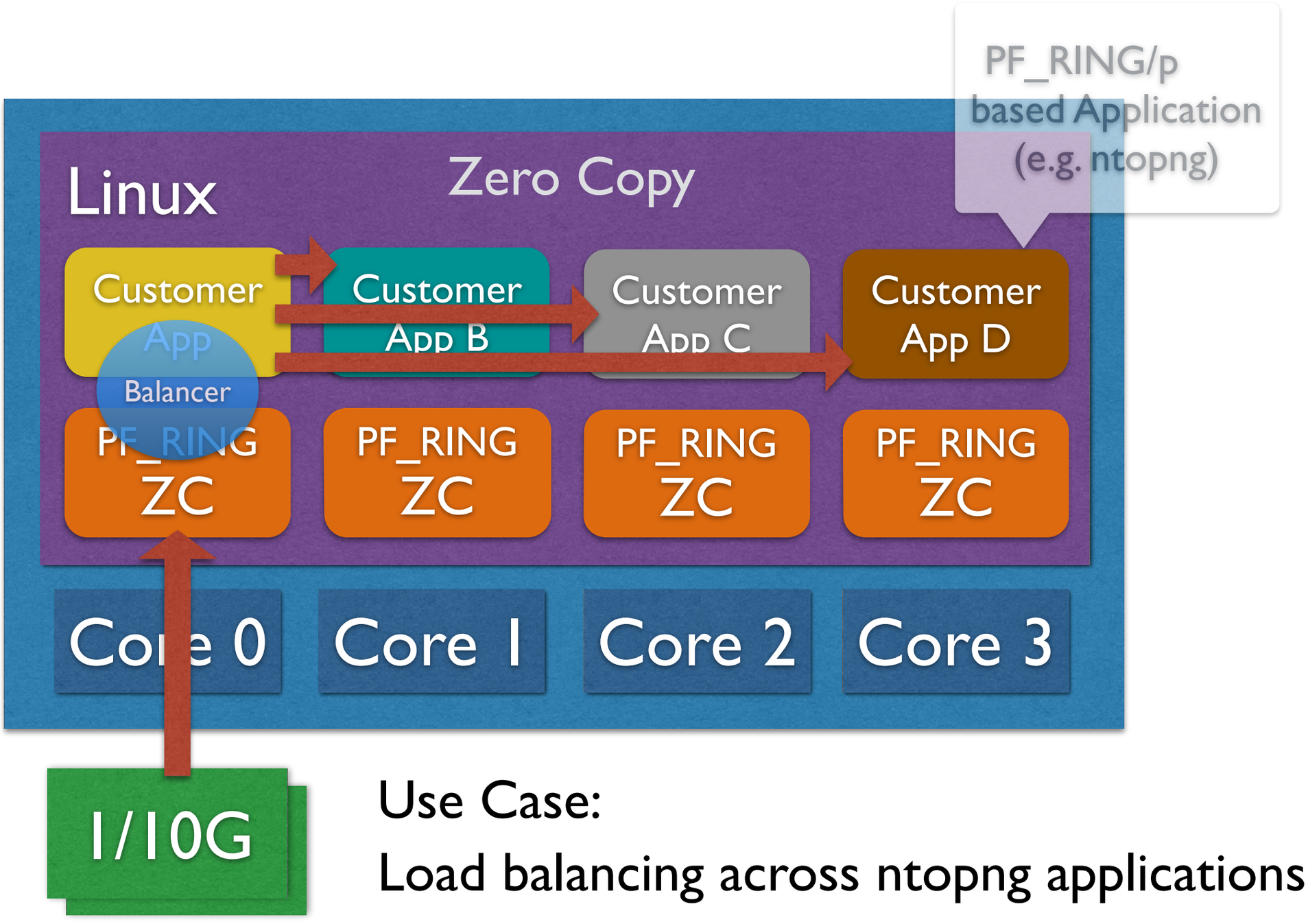
© 2014 - ntop.org

PF_RING ZC [2/2]

- PF_RING ZC has a clean and simple API that hides many low-level configuration details.
- Support of legacy pcap-based applications.
- ZC has simple components: queue and worker.
- KVM support: ability to setup Inter-VM communication.
- Native PF_RING ZC support in many open-source applications such as Snort, Suricata, Bro, Wireshark.
- Ability to operate on top of sysdig.org for dispatching system events to PF_RING applications.

© 2014 - ntop.org

PF_RING ZC Network Topologies [1/2]

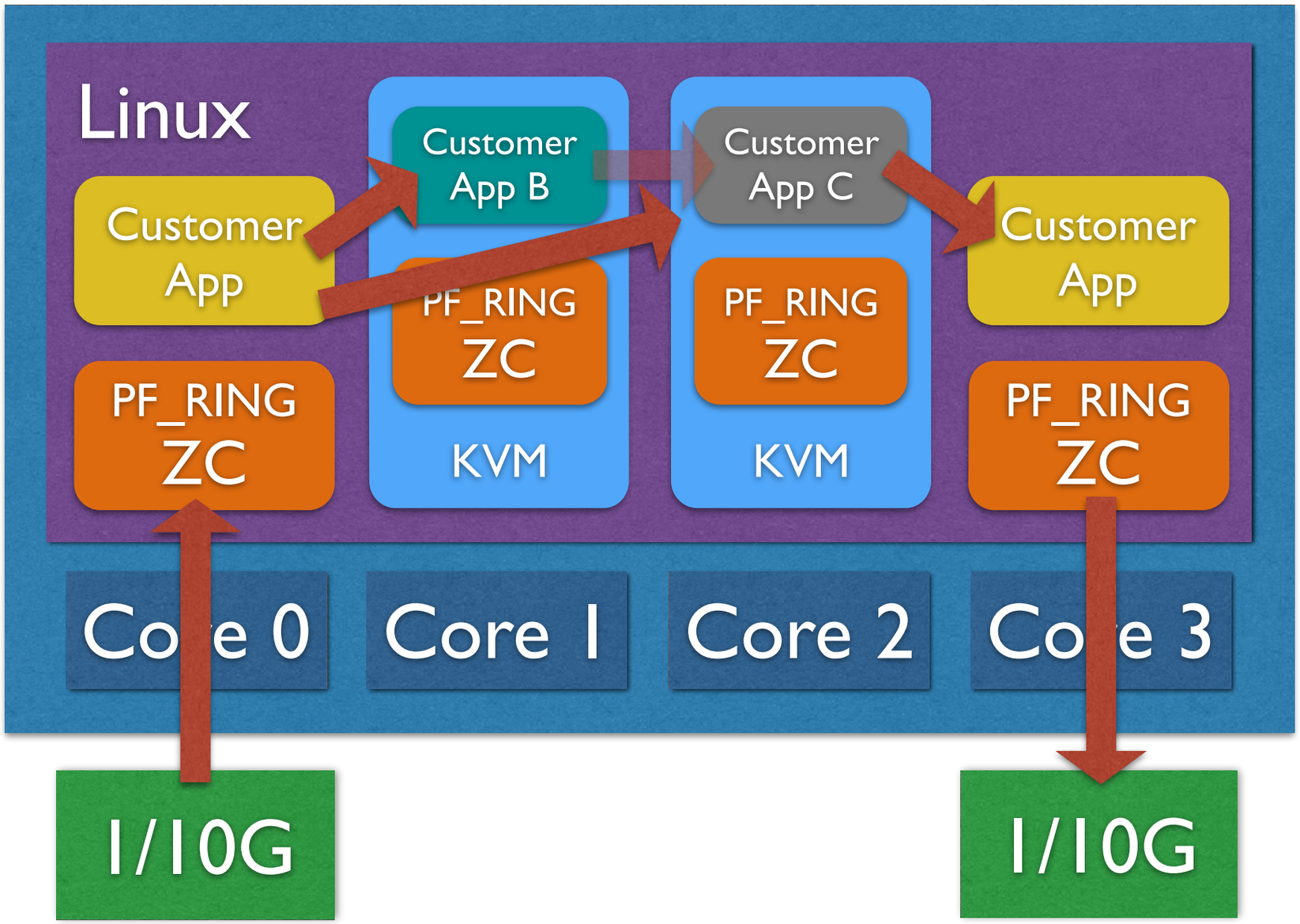


Use Case:

Load balancing across ntopng applications.

© 2014 - ntop.org

PF_RING ZC Network Topologies [2/2]



Use Case:

Application pipeline or run multiple apps (e.g. ntopng) in VMs to insulate them.

© 2014 - ntop.org

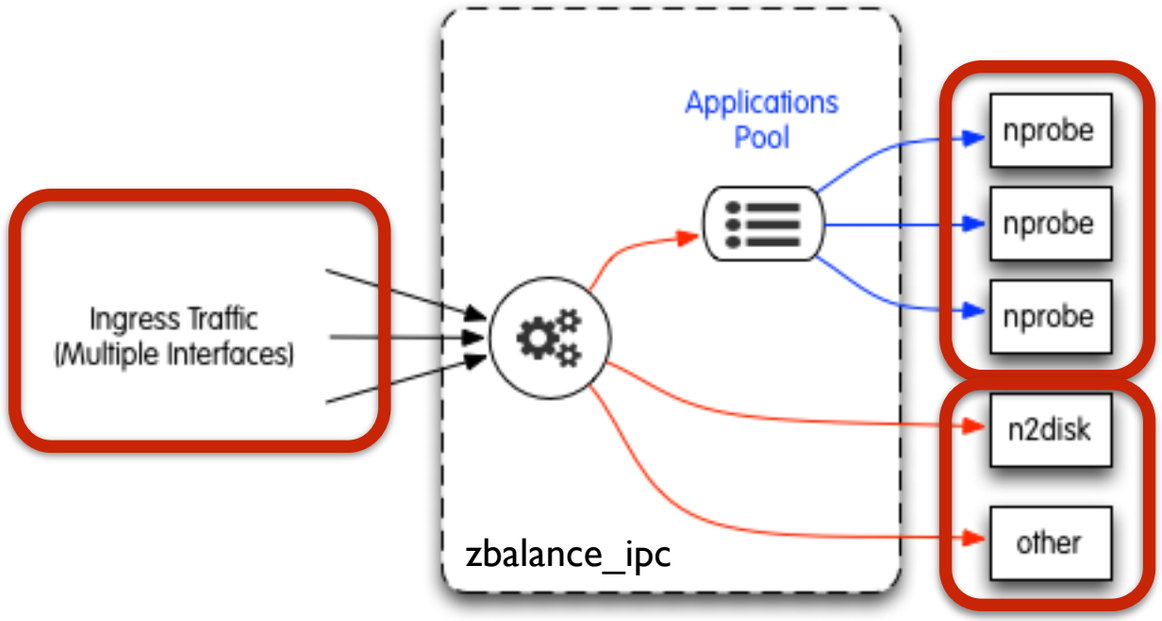
PF_RING (ZC) and ntopng

- Using PF_RING (ZC) with ntopng has several benefits:
  - ntopng can scale to 10 Gbit and above by spawning several ntopng instances each bound to a (few) core(s).
  - It is possible to send the same packet to multiple apps. For instance it is possible to send the same packet to ntopng (for accounting purposes) and n2disk (ntop's application for dumping packet-to-disk at multi-10G) and/or and IDS (e.g. Suricata and snort).

© 2014 - ntop.org

Traffic Balancing with PF_RING ZC

Ingress
Packet Aggregation



Balancing

0-Copy
Packet Fanout

Traffic Fan-Out Across Applications

Traffic Balancing Across Applications

© 2014 - ntop.org

The need for DPI in Monitoring [2/2]

- DPI (Deep Packet Inspection) is a technique for inspecting the packet payload for the purpose of extracting metadata (e.g. protocol).
- There are many DPI toolkits available but they are not what we looked for as:
  - They are proprietary (you need to sign an NDA to use them), and costly for both purchase and maintenance.
  - Adding a new protocol requires vendor support (i.e. it has a high cost and might need time until the vendor supports it) = you're locked-in.
- On a nutshell DPI is a requirement but the market does not offer an alternative for open-source.

© 2014 - ntop.org

Say hello to nDPI



- ntop has decided to develop its own GPL DPI toolkit in order to build an open DPI layer for ntop and third party applications.
- Supported protocols (> 180) include:
  - P2P (Skype, BitTorrent)
  - Messaging (Viber, Whatsapp, MSN, The Facebook)
  - Multimedia (YouTube, Last.gm, iTunes)
  - Conferencing (Webex, CitrixOnLine)
  - Streaming (Zattoo, Icecast, Shoutcast, Netflix)
  - Business (VNC, RDP, Citrix, *SQL)

© 2014 - ntop.org

nDPI Overview

- Portable C library (Win and Unix, 32/64 bit).
- Designed for user and kernel space
  - Linux ndpi-netfilter implements L7 kernel filters
- Used by many non-ntop projects (eg. xplico.org) and part of Linux distributions (e.g. Debian).
- Able to operate on both plain ethernet traffic and encapsulated (e.g. GTP, GRE…).
- Ability to specify at runtime custom protocols (port or hostname - dns, http, https -based).

© 2014 - ntop.org

nDPI API

- The nDPI API is pretty simple
  - `ndpi_init_detection_module()`
    `ndpi_exit_detection_module()`
    Init/term the nDPI library.
  - `ndpi_load_protocols()`
    Load custom protocol definitions.
  - `ndpi_detection_process_packet()`
    Process the packet in nDPI and return the L7 protocol or NDPI_UNKNOWN (too early or detection failed).
  - `ndpi_guess_protocol()`
    Guess a L7 protocols when DPI fails.

© 2014 - ntop.org

nDPI on ntopng

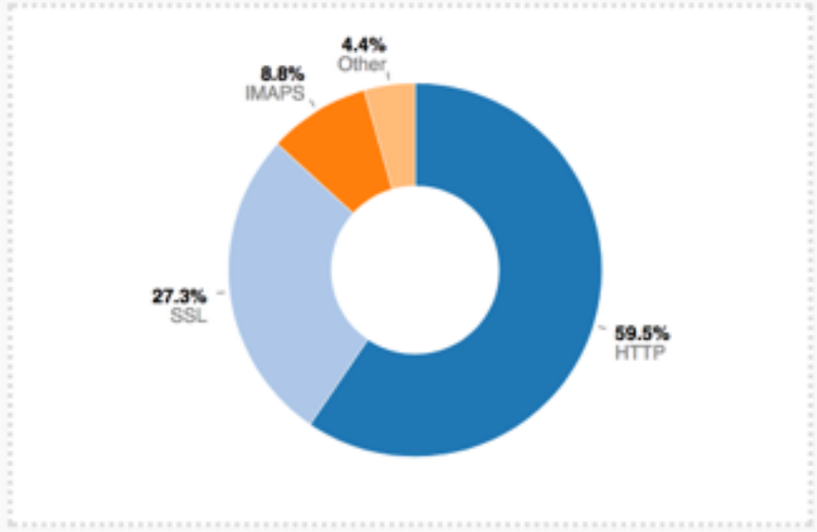
- In ntopng all flows are analysed through nDPI to associate an application protocol to them.
- L7 statistics are available per flow, host, and interface (from which monitoring data is received).
- For network interfaces and local hosts, nDPI statistics are saved persistently to disk (in RRD format).

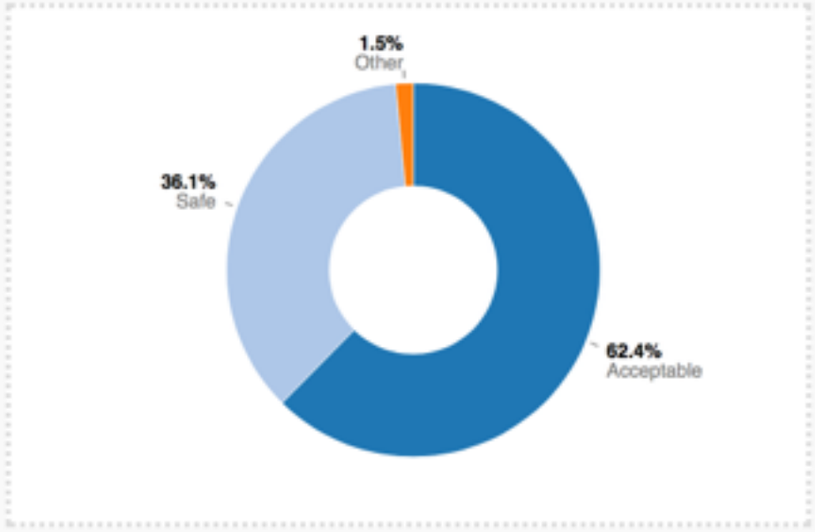
© 2014 - ntop.org

nDPI on ntopng: Interface Report [1/2]

Interface: en0 | Overview | Packets | Protocols | Historical Activity

Protocol Overview

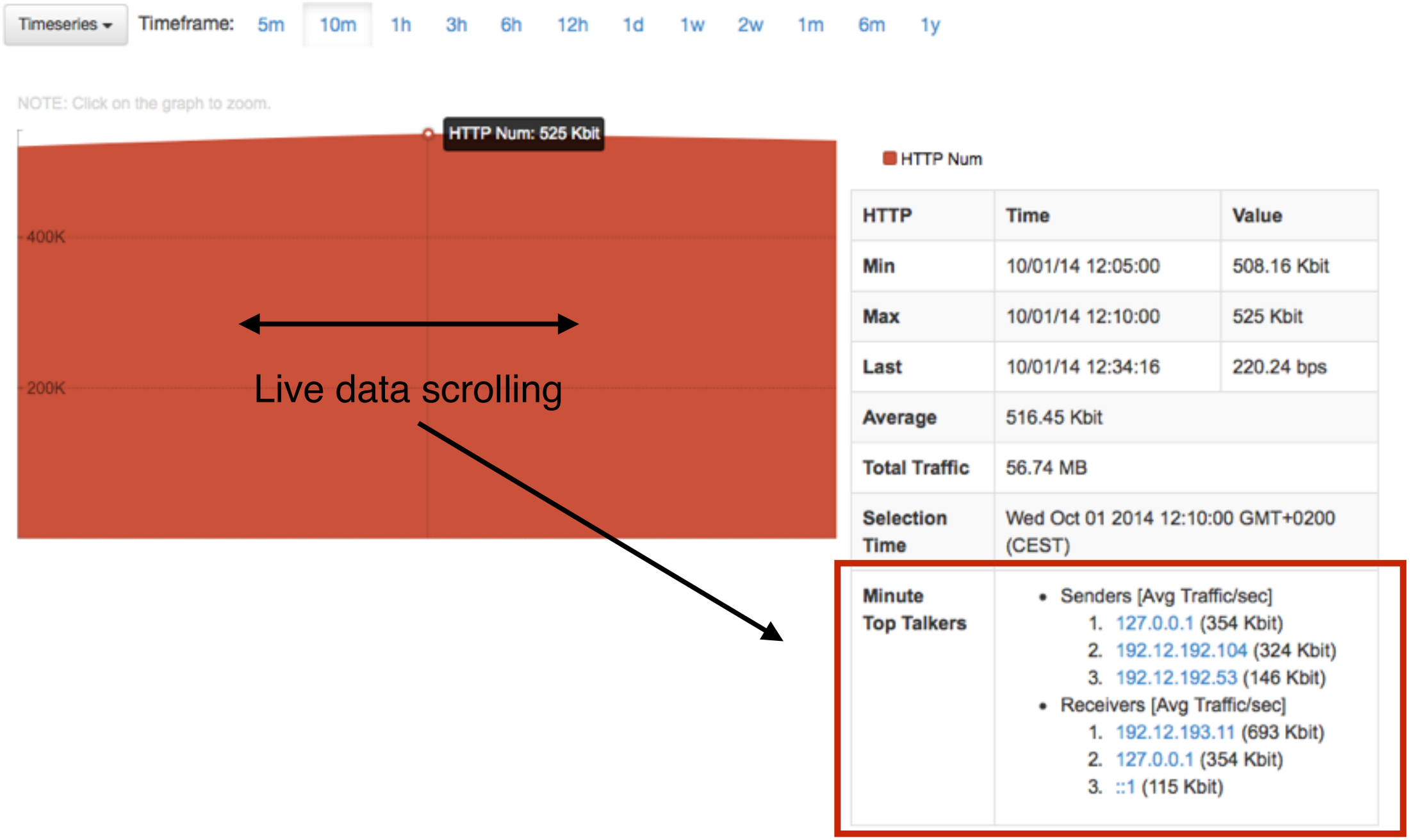




Application Protocol	Total (Since Startup)	Percentage
AppleiCloud	8.43 KB	0.33 %
DNS	4.19 KB	0.17 %
DropBox	6.15 KB	0.24 %
Google	9.04 KB	0.36 %
HTTP	1.43 MB	57.8 %
ICMP	280 Bytes	0.01 %
IMAPS	216.79 KB	8.56 %

© 2014 - ntop.org

nDPI on ntopng: Interface Report [2/2]



HTTP	Time	Value
Min	10/01/14 12:05:00	508.16 Kbit
Max	10/01/14 12:10:00	525 Kbit
Last	10/01/14 12:34:16	220.24 bps
Average	516.45 Kbit	
Total Traffic	56.74 MB	
Selection Time	Wed Oct 01 2014 12:10:00 GMT+0200 (CEST)	
Minute Top Talkers	• Senders [Avg Traffic/sec] 1. 127.0.0.1 (354 Kbit) 2. 192.12.192.104 (324 Kbit) 3. 192.12.192.53 (146 Kbit) • Receivers [Avg Traffic/sec] 1. 192.12.193.11 (693 Kbit) 2. 127.0.0.1 (354 Kbit) 3. ::1 (115 Kbit)	

© 2014 - ntop.org

ntopng and Redis [1/2]

- Redis is an open source key-value in-memory database.
- ntop uses it to cache data such as:
  - Configuration and user preferences information.
  - DNS name resolution (numeric to symbolic).
  - Volatile monitoring data (e.g. hosts JSON representation).
- Some information is persistent (e.g. preferences) and some is volatile: ntopng can tell redis how long a given value must be kept in cache.

© 2014 - ntop.org

ntopng and Redis [2/2]

- Redis is also used as a (persistent) queue for requests towards external applications.
  - If configured (-F command line option), periodically flow status is saved onto a redis queue, requests are packed, and send to a remote BigData system.

- In essence Redis is used by ntopng to store information that might take too much memory (if kept on ntopng memory space), or to pile up list of things that are executed periodically or that require interaction with remote applications that might be slow or temporary unavailable.

© 2014 - ntop.org

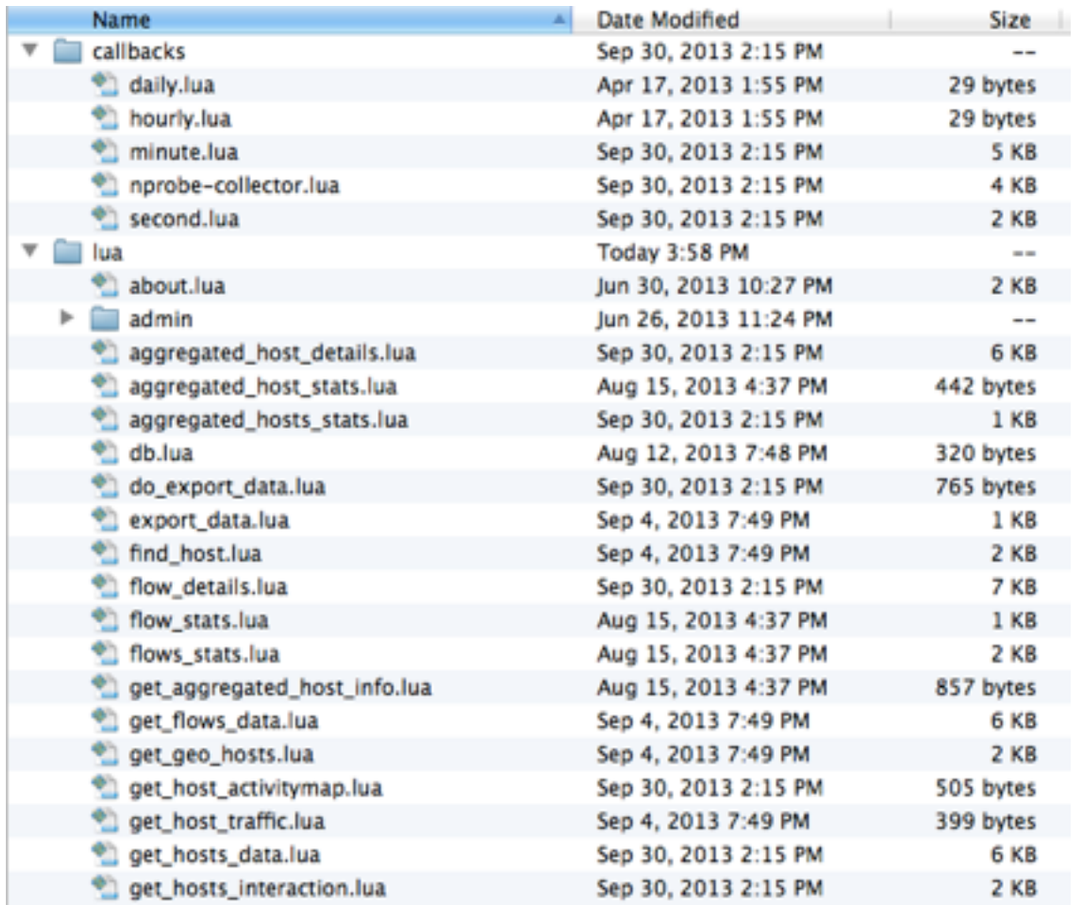
Lua-based ntopng Scriptability [1/3]

- 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.
- Lua methods invoke the ntopng C++ API in order to interact with the monitoring engine.

© 2014 - ntop.org

Lua-based ntopng Scriptability [2/3]

- `/scripts/callback/` scripts are executed periodically to perform specific actions.
- `/scripts/lua/` scripts are executed only by the web GUI.
- Example: `http://ntopng:3000/lua/flow_stats.lua`



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	765 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

© 2014 - ntop.org

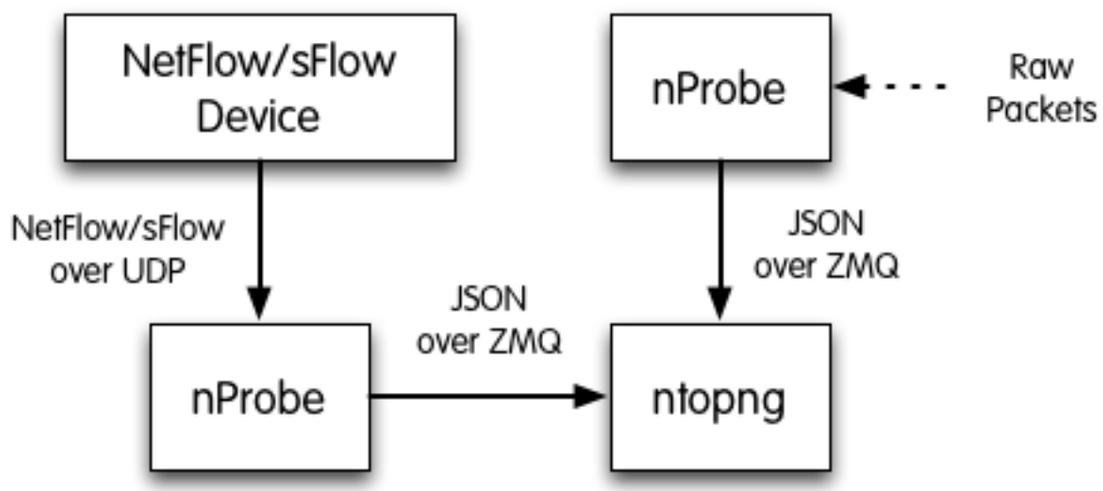
Lua-based ntopng Scriptability [3/3]

- ntopng defines (in C++) two Lua classes:
  - `interface`
    - Hook to objects that describe flows and hosts.
    - Access to live monitoring data.
  - `ntop`
    - General functions used to interact with ntopng configuration.
- Lua objects are usually in "read-only" mode
  - 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.

© 2014 - ntop.org

ntopng as a NetFlow/sFlow Collector [1/3]

- The "old" ntop included a NetFlow/sFlow collector. Considered the effort required to support all the various NetFlow dialects (e.g. Cisco ASA flows are not "really" flows), in ntopng we have made a different design choice.



© 2014 - ntop.org

ntopng as a NetFlow/sFlow Collector [2/3]

- nProbe (a home-grown NetFlow/sFlow collector/probe) is responsible for collecting/generating flows and convert them to JSON so that ntopng can understand it.
- The communication ntopng <-> nProbe is over ØMQ a simple/fast messaging system that allows the two peers to be decoupled while:
  - Avoiding "fat" communication protocols such as HTTP.
  - Relying on a system that works per message (no per packet) and handles automatic reconnection if necessary.

© 2014 - ntop.org

ntopng as a NetFlow/sFlow Collector [3/3]

Flows are sent in the following format

- {“8”:"192.12.193.11","12":"192.168.1.92","15":"0.0.0.0","10":0,"14":0,"2":5,"1":406,"22":1412183096,"21":1412183096,"7":3000,"11":55174,"6":27,"4":6,"5":0,"16":2597,"17":0,"9":0,"13":0,"42":4}
- Where:
  - “<Element ID>”: <value> (example 8 = IPV4_SRC_ADDR)

- Contrary to what happens in NetFlow/sFlow ntopng (collector) connects to nProbe (probe) and fetches the emitted flows. Multiple collectors can connect to the same probe. No traffic is created when no collector is attached to the probe.

© 2014 - ntop.org

Flow Collection Setup: an Example

Flow collection/generation (nProbe)

- Probe mode

```
nprobe --zmq "tcp://*:5556" -i eth1 -n none
```

- sFlow/NetFlow collector mode

```
nprobe --zmq "tcp://*:5556" -i none -n none --collector-port 2055
```

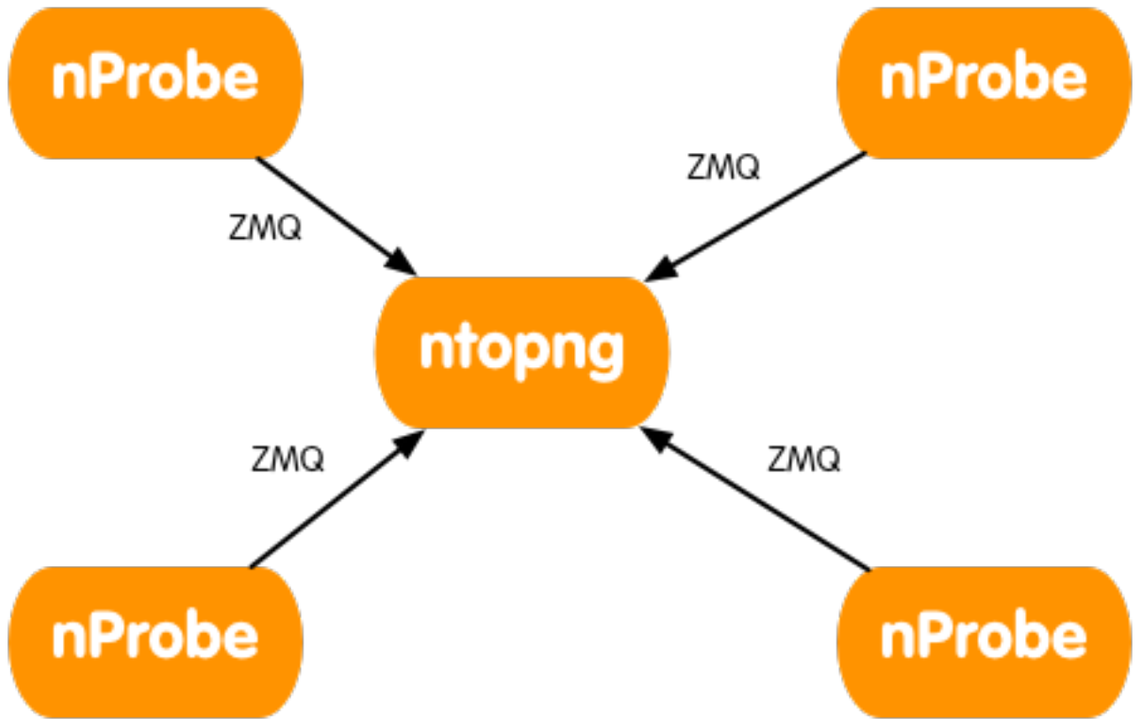
Data Collector (ntopng)

- `ntopng -i tcp://127.0.0.1:5556`

© 2014 - ntop.org

Creating ntopng Clusters [1/3]

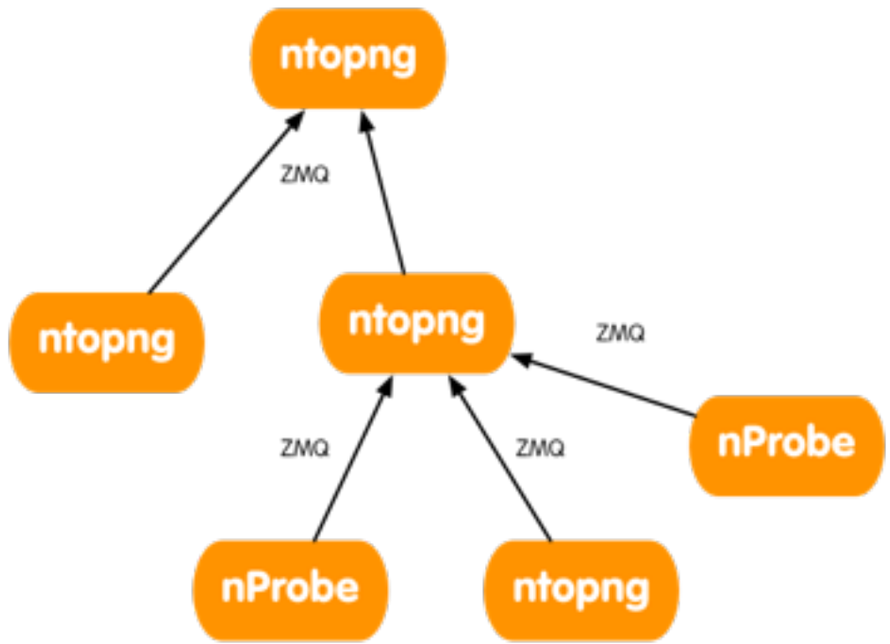
- ntopng is not only a flow collector, but it can export flows in the same JSON format used in the received flows.
- This allows complex clusters to be created:



© 2014 - ntop.org

Creating ntopng Clusters [2/3]

- In many companies, there are many satellite offices and a few central aggregation points.
- Using ØMQ (both ntopng and nProbe flows are in the same format) it is possible to create a hierarchy of instances.
- Each node aggregates the traffic for the instances "below" it, so that at each tree layer you have a summarised view of the network activities.



© 2014 - ntop.org

Creating ntopng Clusters [3/3]

Example

- 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`

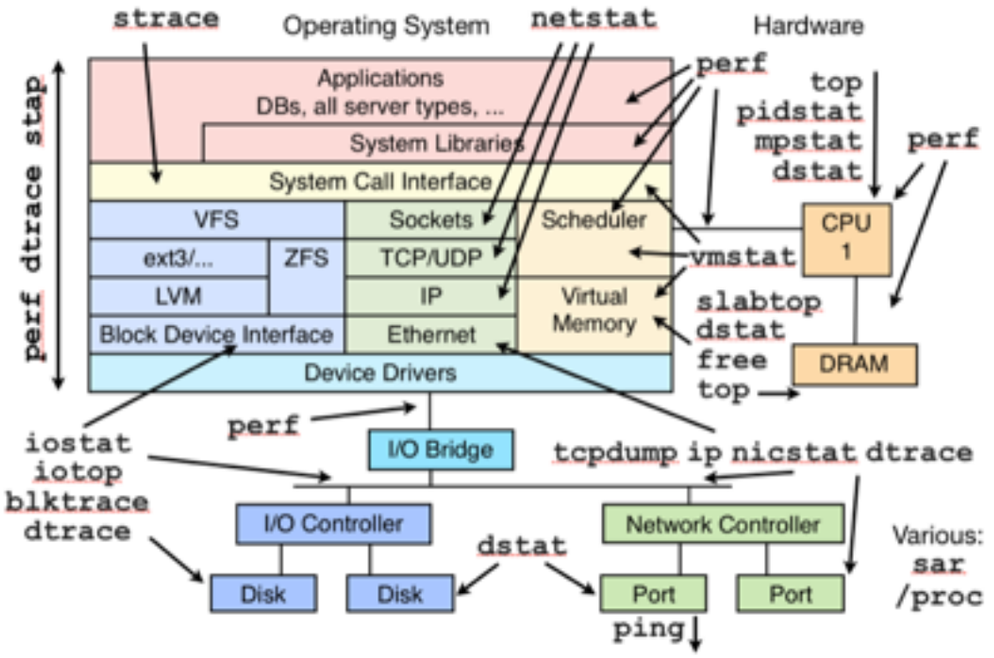
- If you want to merge all nProbe traffic into a single ntopng interface do:
  - `ntopng -i tcp://host1:5556,tcp://host2:5556,tcp://host3:5556,tcp://host4:5556`

- 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`

© 2014 - ntop.org

System+Network Monitoring [1/3]

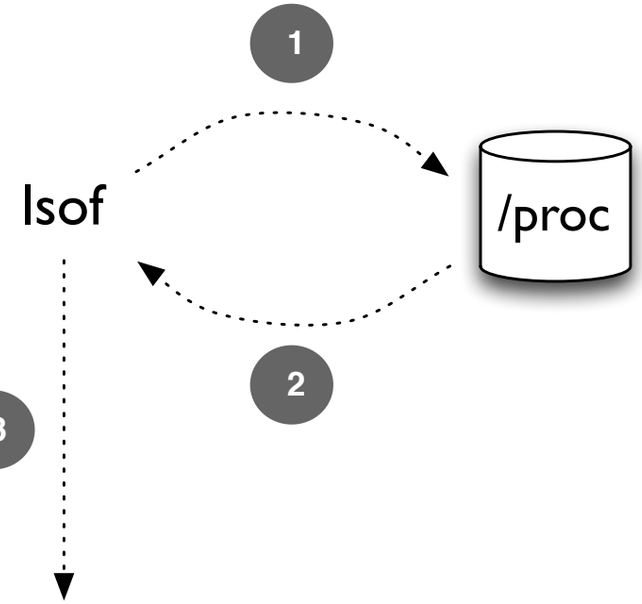
- Historically on Unix there are many tools for system monitoring.
- Like when we started the development of ntop, all these tools are nice per-se, but are not integrated with the rest of the environment.
- ntopng/nProbe monitor network activities, but have no visibility of the processes that are originating the observed network activities.

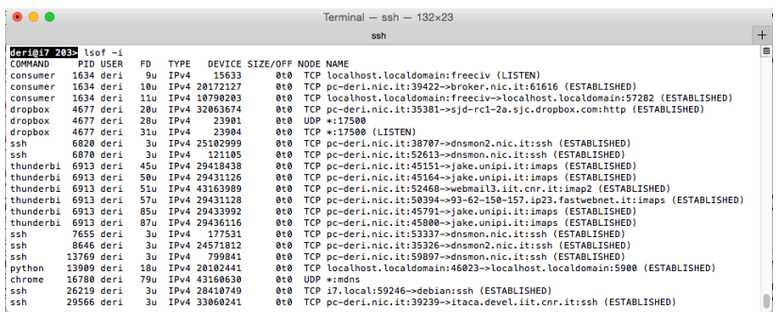


© 2014 - ntop.org

System+Network Monitoring [3/3]

How most system management tools work on Linux:

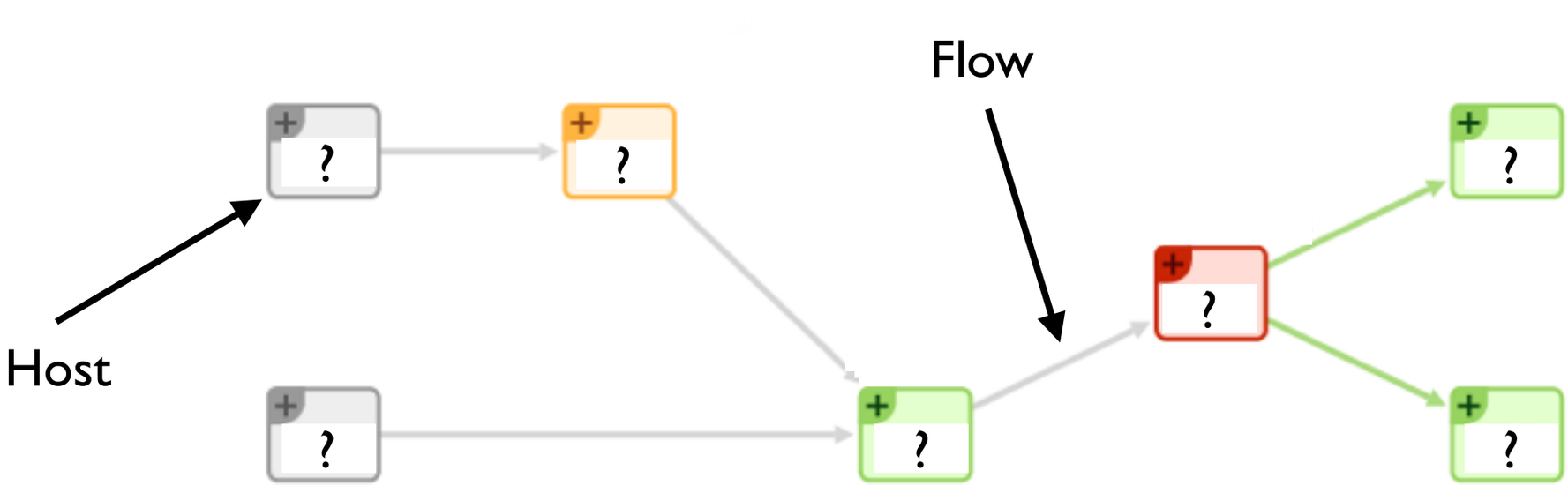




© 2014 - ntop.org

System+Network Monitoring [3/3]

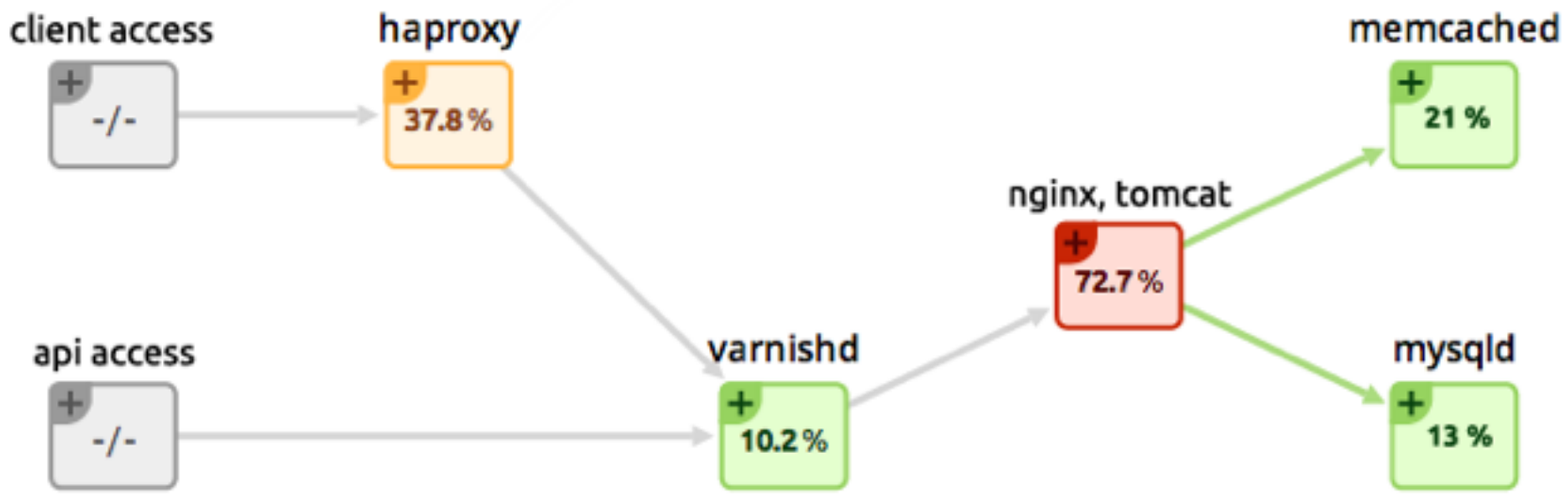
- Using ntopng/nProbe you can see the flows that are are being exchanged across systems but it is not possible to know more than that.



© 2014 - ntop.org

System+Network Monitoring [3/3]

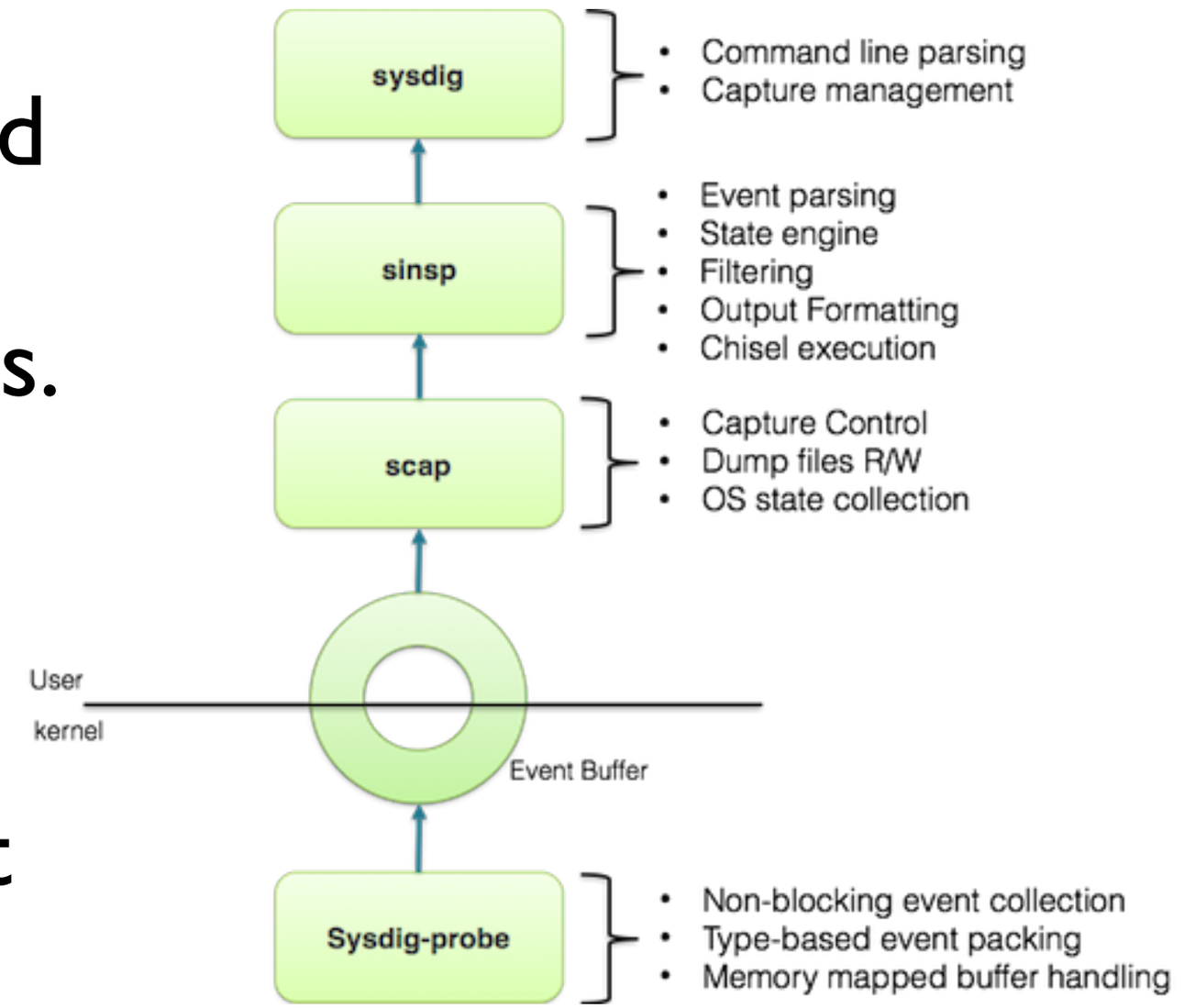
- It would be desirable to know exactly what is the process originating the traffic observed and what resources the process is using while generating such traffic.
- In essence we would like to see this picture:



© 2014 - ntop.org

Welcome to Sysdig

- Sysdig is a Linux framework developed by Draios Inc for capturing system calls.
- The kernel module intercepts the calls.
- The user-space libs receive and interpret the received calls.



© 2014 - ntop.org

Why Sysdig?

- Contrary to all other tools available for system monitoring, sysdig implements the "packet paradigm" applied to system events:
  - Events are received in a way similar to what happens with packet capture.
  - It is possible to store events on pcap-like files and reply them later on.
- To simplify things, instead of using the sysdig API, we added native sysdig support in PF_RING so that all apps (e.g. ntopng) can use it.

© 2014 - ntop.org

Integrating sysdig in nProbe [1/3]

- Instead of complicating the design of ntopng with sysdig support, we have decided to extend nProbe with system visibility.
- nProbe monitors both the network interfaces and the system events via PF_RING.
- Network and system information is then combined and exported in standard network flows over NetFlow v9/IPFIX and in JSON to ntopng for data visualization.

© 2014 - ntop.org

Integrating sysdig in nProbe [2/3]

- The current system information elements include:

```
[NFv9 57640][IPFIX 35632.168] %SRC_PROC_PID                 Src process PID
[NFv9 57641][IPFIX 35632.169] %SRC_PROC_NAME                Src process name
[NFv9 57844][IPFIX 35632.372] %SRC_PROC_USER_NAME           Src process user name
[NFv9 57845][IPFIX 35632.373] %SRC_FATHER_PROC_PID          Src father process PID
[NFv9 57846][IPFIX 35632.374] %SRC_FATHER_PROC_NAME         Src father process name
[NFv9 57855][IPFIX 35632.383] %SRC_PROC_ACTUAL_MEMORY       Src process actual memory (bytes)
[NFv9 57856][IPFIX 35632.384] %SRC_PROC_PEAK_MEMORY         Src process peak memory (bytes)
[NFv9 57857][IPFIX 35632.385] %SRC_PROC_AVERAGE_CPU_LOAD    Src process avg load (% * 100)
[NFv9 57858][IPFIX 35632.386] %SRC_PROC_NUM_PAGE_FAULTS     Src process num pagefaults
[NFv9 57865][IPFIX 35632.393] %SRC_PROC_PCTG_IOWAIT         Src process iowait time % (% * 100)
[NFv9 57847][IPFIX 35632.375] %DST_PROC_PID                 Dst process PID
[NFv9 57848][IPFIX 35632.376] %DST_PROC_NAME                Dst process name
[NFv9 57849][IPFIX 35632.377] %DST_PROC_USER_NAME           Dst process user name
[NFv9 57850][IPFIX 35632.378] %DST_FATHER_PROC_PID          Dst father process PID
[NFv9 57851][IPFIX 35632.379] %DST_FATHER_PROC_NAME         Dst father process name
[NFv9 57859][IPFIX 35632.387] %DST_PROC_ACTUAL_MEMORY       Dst process actual memory (bytes)
[NFv9 57860][IPFIX 35632.388] %DST_PROC_PEAK_MEMORY         Dst process peak memory (bytes)
[NFv9 57861][IPFIX 35632.389] %DST_PROC_AVERAGE_CPU_LOAD    Dst process avg load (% * 100)
[NFv9 57862][IPFIX 35632.390] %DST_PROC_NUM_PAGE_FAULTS     Dst process num pagefaults
[NFv9 57866][IPFIX 35632.394] %DST_PROC_PCTG_IOWAIT         Src process iowait time % (% * 100)
```


© 2014 - ntop.org

Integrating sysdig in nProbe [3/3]

- Using sysdig, nProbe is able to bind a (local) process to a network flow, and to monitor its I/O activities, CPU and memory utilisation.
- This way we know for sure what network activities are performed by processes, including those activities performed by trojans and malware that start up, send the packet-of-death and then disappear.
- Thanks to the PID/father-PID hierarchy it is possible to know an any time the exact activation chain.

© 2014 - ntop.org

ntopng+nProbe+sysdig [1/2]

- In order to activate system+network monitoring, it is necessary to load the sysdig kernel module and start nProbe (flow probe) as follows:

```
nprobe -T "%IPV4_SRC_ADDR %L4_SRC_PORT  %IPV4_DST_ADDR %L4_DST_PORT %IN_PKTS
%IN_BYTES %FIRST_SWITCHED %LAST_SWITCHED" %TCP_FLAGS %PROTOCOL @PROCESS@ %L7_PROTO
--zmq "tcp://*:1234" -i any --dont-drop-privileges -t 5 -b 2
```

- Then start ntopng (flow collector) as follows:

```
ntopng -i tcp://nprobe1.ntop.org:1234 -i tcp://nprobe2.ntop.org:1234 …
```


© 2014 - ntop.org

ntopng+nProbe+sysdig [2/2]

- When ntopng receives flow enriched with system information, it interprets it, and depicts:
  - The process-to-flow association.
  - For flows whose peers are hosts monitored by nProbe instances, it "glues" the flows together.
  - The process call father/process hierarchy is depicted.
  - The overall system process view including the process relationships.

© 2014 - ntop.org

Process Network Communications



Flow/Process Drill-down [1/2]

Active Flows

10 ▾ Applications▾

Info	Application	L4 Proto	Client Process	Client Peer	Server Process	Server Peer	Duration	Breakdown	Total Bytes
Info	SSH	TCP		dnsmon.nic.it :22		pc-deri.nic.it :46861	1 day, 6 h, 12 min, 6 sec	Client S	5.41 GB
Info	Redis	TCP	ntopng	localhost.localdomai... :53452	redis-server	localhost.localdomai... :6379	1 day, 6 h, 12 min, 5 sec	Client Server	3.8 GB

Flow: localhost.localdomain:53452 ⇄ localhost.localdomain:6379 | Overview | ↩

Flow Peers	localhost :53452 ⇄ localhost :6379	
Protocol	TCP / Redis	
First / Last Seen	30/09/2014 15:01:34 [1 day, 6 h, 13 min, 5 sec ago]	01/10/2014 21:14:33 [6 sec ago]
Total Traffic Volume	3.81 GB —	
Client vs Server Traffic Breakdown	localhost.localdomain:53452 / localhost.localdomain:6379	
Client to Server Traffic	7,677,287 Pkts / 1.85 GB —	
Server to Client Traffic	6,754,744 Pkts / 1.95 GB —	
TCP Flags	SYN PUSH ACK This flow is active.	



© 2014 - ntop.org

Flow/Process Drill-down [2/2]

Client Process Information	
User Name	deri
Process PID/Name	13058/ntopng [son of 11235/tcsh]
Average CPU Load	0.71 %
I/O Wait Time Percentage	0 %
Memory Actual / Peak	1.4 MB / 1.46 MB [95.7%]
VM Page Faults	0

Flow-to-Process binding

} Dynamically Updated

Server Process Information	
User Name	redis
Process PID/Name	1769/redis-server [son of 1/init]
Average CPU Load	0.12 %
I/O Wait Time Percentage	0 %
Memory Actual / Peak	344.13 KB / 344.13 KB [100%]
VM Page Faults	0

Flow-to-Process binding

} Dynamically Updated

© 2014 - ntop.org

Active Process Network Connections

Info	Application	L4 Proto	Client Process	Client Peer	Server Process	Server Peer	Duration	Breakdown	Total Bytes
Info	HTTP	TCP	chrome	i7 :50540	java	i7 :9200	1 h, 39 min, 2 sec	Server	37.88 MB
Info	HTTP	TCP	chrome	i7 :45671	java	i7 :9200	1 h, 11 min, 2 sec	Server	24.03 MB
Info	HTTP	TCP	ntopng	i7 :48526	java	i7 :9200	1 sec	Client S	7.62 KB
Info	HTTP	TCP	ntopng	i7 :48528	java	i7 :9200	1 sec	Client S	7.27 KB
Info	HTTP	TCP	ntopng	i7 :48529	java	i7 :9200	1 sec	Client S	6.71 KB
Info	HTTP	TCP	ntopng	i7 :48527	java	i7 :9200	1 sec	Client S	6.69 KB
Info	HTTP	TCP	ntopng	i7 :48525	java	i7 :9200	1 sec	Client S	6.48 KB
Info	HTTP	TCP	chrome	i7 :48461	java	i7 :9200	1 sec	Cl Server	5.2 KB

Showing 1 to 8 of 8 rows

© 2014 - ntop.org

Process Network Traffic

Active Processes

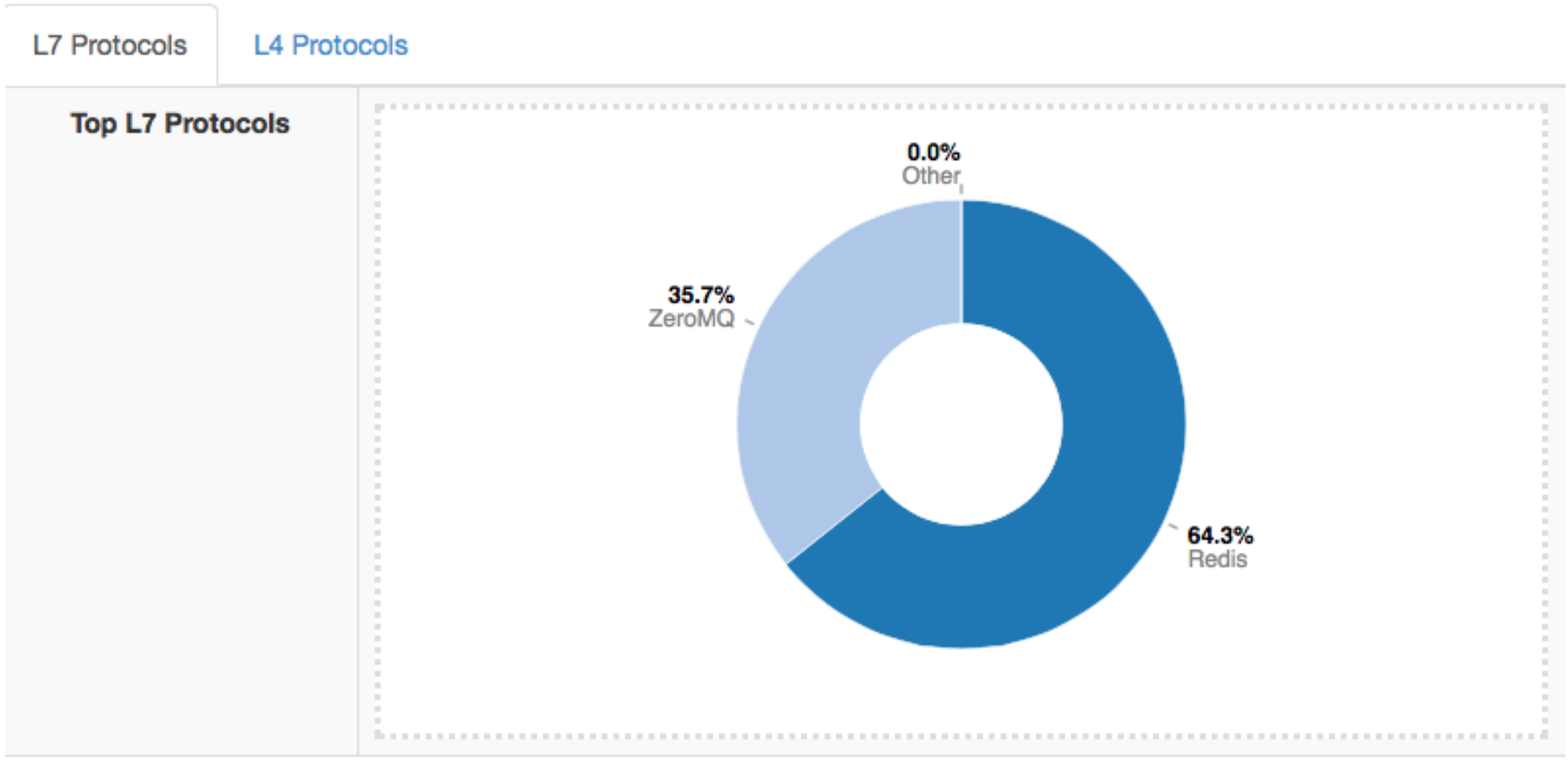
10

Name	Flows Count	Active Since	Traffic Sent	Traffic Rcvd
sshd	1 –	19 min, 42 sec	39.99 KB –	20.87 KB –
redis-server	1 –	1 day, 6 h, 19 min, 50 sec	1.96 GB ↑	1.86 GB ↑
ntopng	40 ↑	1 day, 6 h, 19 min, 50 sec	2.05 GB ↑	3.89 GB ↑
nprobe	2 –	1 day, 6 h, 19 min, 55 sec	2.72 GB ↑	198.96 MB ↑
java	13 ↓	1 h, 41 min, 32 sec	63.57 MB ↑	554.87 KB ↑
dropbox	1 –	20 min, 27 sec	28.79 KB –	6.36 KB –
chrome	17 ↑	14 h, 18 min, 2 sec	11.58 MB ↑	790.09 MB ↑

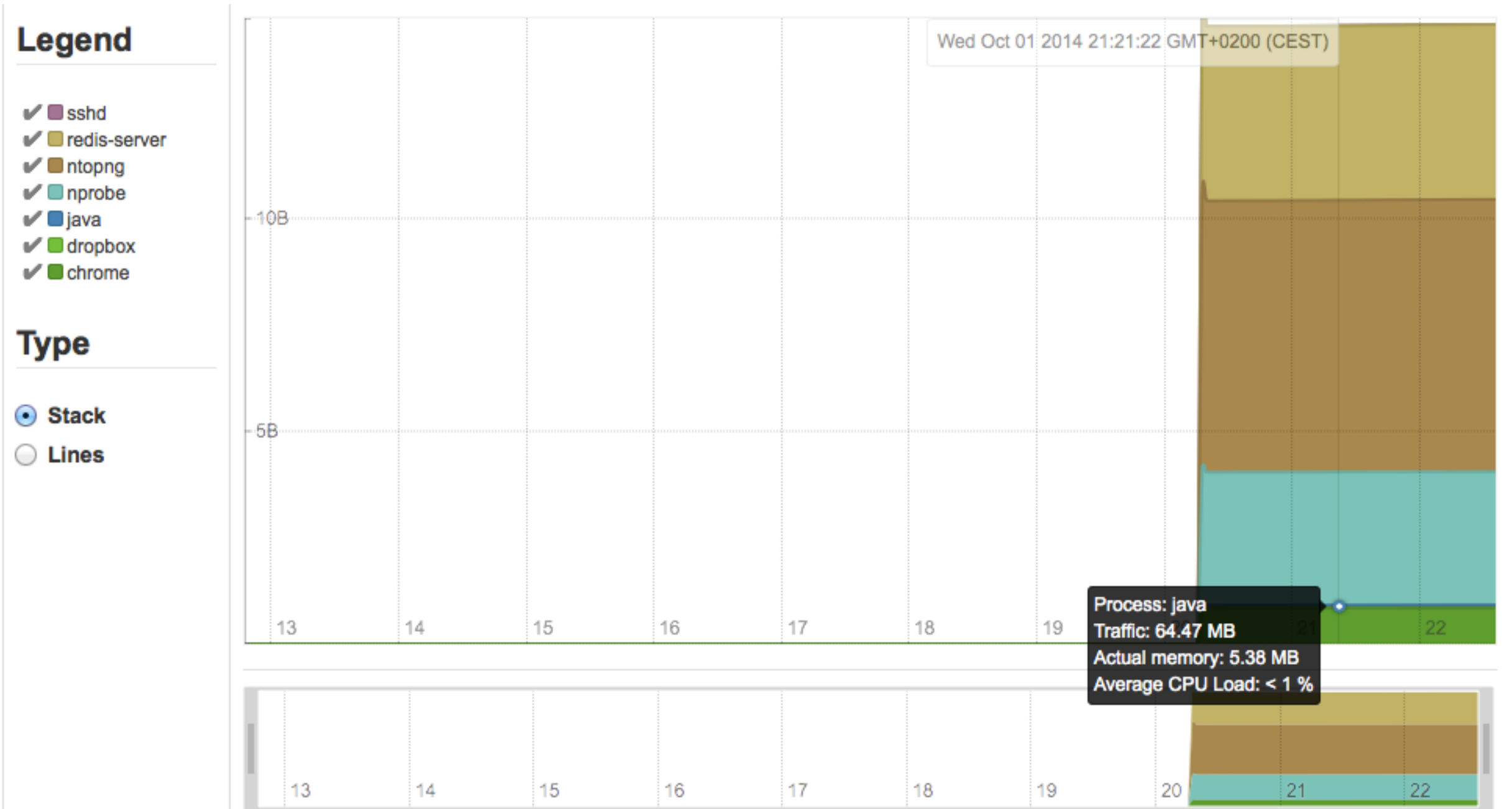
Showing 1 to 7 of 7 rows

© 2014 - ntop.org

Process Protocols Drill-Down



Processes Timeline



User Flows and Processes

elasticsearch - i7 | Applications | Protocols | Flow

Active Flows

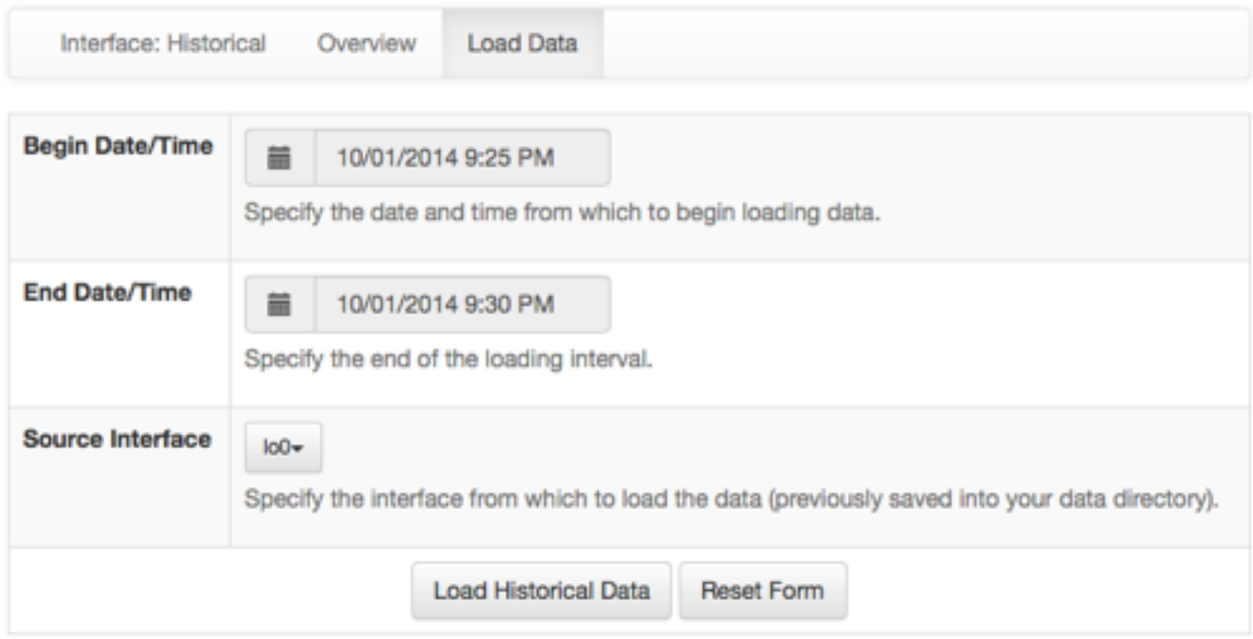
10 | Applications

Info	Application	L4 Proto	Client Process	Client Peer	Server Process	Server Peer	Duration	Breakdown	Total Bytes
Info	HTTP	TCP	chrome	i7 :50540	java	i7 :9200	1 h, 54 min, 2 sec	Server	43.81 MB
Info	HTTP	TCP	chrome	i7 :45671	java	i7 :9200	1 h, 26 min, 2 sec	Server	26.99 MB
Info	HTTP	TCP	chrome	i7 :48461	java	i7 :9200	15 min, 1 sec	Server	3.82 MB
Info	HTTP	TCP	ntopng	i7 :33419	java	i7 :9200	1 sec	Client	8.13 KB
Info	HTTP	TCP	ntopng	i7 :33418	java	i7 :9200	1 sec	Client	6.72 KB
Info	HTTP	TCP	ntopng	i7 :33417	java	i7 :9200	1 sec	Client	6.71 KB

© 2014 - ntop.org

Historical Flow Navigation

- ntopng can store (-F) persistently on disk networks flows in SQLite format on 5 min DBs.
- Inside ntopng there is a (virtual) Historical interface that retrieves such flows and allows them to be navigated using the ntopng GUI.



© 2014 - ntop.org

ntopng and Big Data

- Using SQLite to save flows persistently is good when flows are not too many and the system that runs ntopng has storage.
- For large deployments or disk-less systems (e.g. ARM-based PCs) it is desirable to upload flows on remote, cloud-based, systems able to scale with the number of flows.
- In essence ntopng has been opened to what is currently defined as "big data" systems that can scale with data in volume and speed.

© 2014 - ntop.org

Integrating ntopng with ElasticSearch [1/2]

- An emerging Big Data system is ElasticSearch that is used by a large community because of its flexibility and user interface (Kibana) that allow visual applications to be developed in minutes.
- Although we do not want to bind ntopng only with ES, we believe that its integration is a good starting point for:
  - Opening ntopng to the Big Data world.
  - Allowing people to use ntopng as data source and let them use ES for long-term data storage and develop custom dashboards using Kibana.

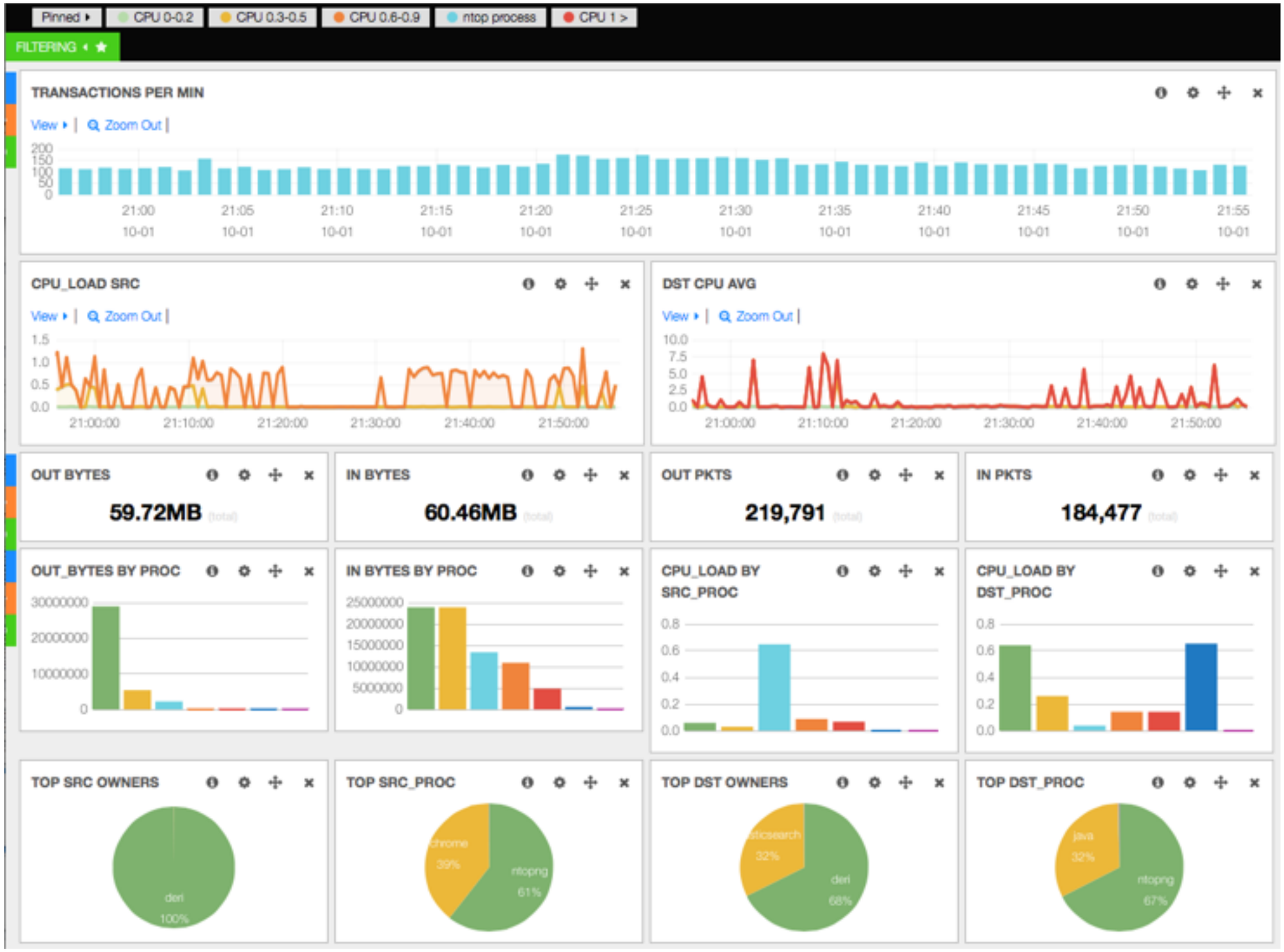
© 2014 - ntop.org

Integrating ntopng with ElasticSearch [2/2]

- ntopng dumps exported flows in JSON format onto a Redis queue enriched with some specified ES attributes (e.g. @timestamp that specifies the time such flow has been exported).
- As soon as there is a minimum number of flows in queue, a ntopng thread packs them together and sends them to ES using the ES bulk API.
- ES indexes the received flows and make them available to external applications such as the Kibana dashboard.

© 2014 - ntop.org

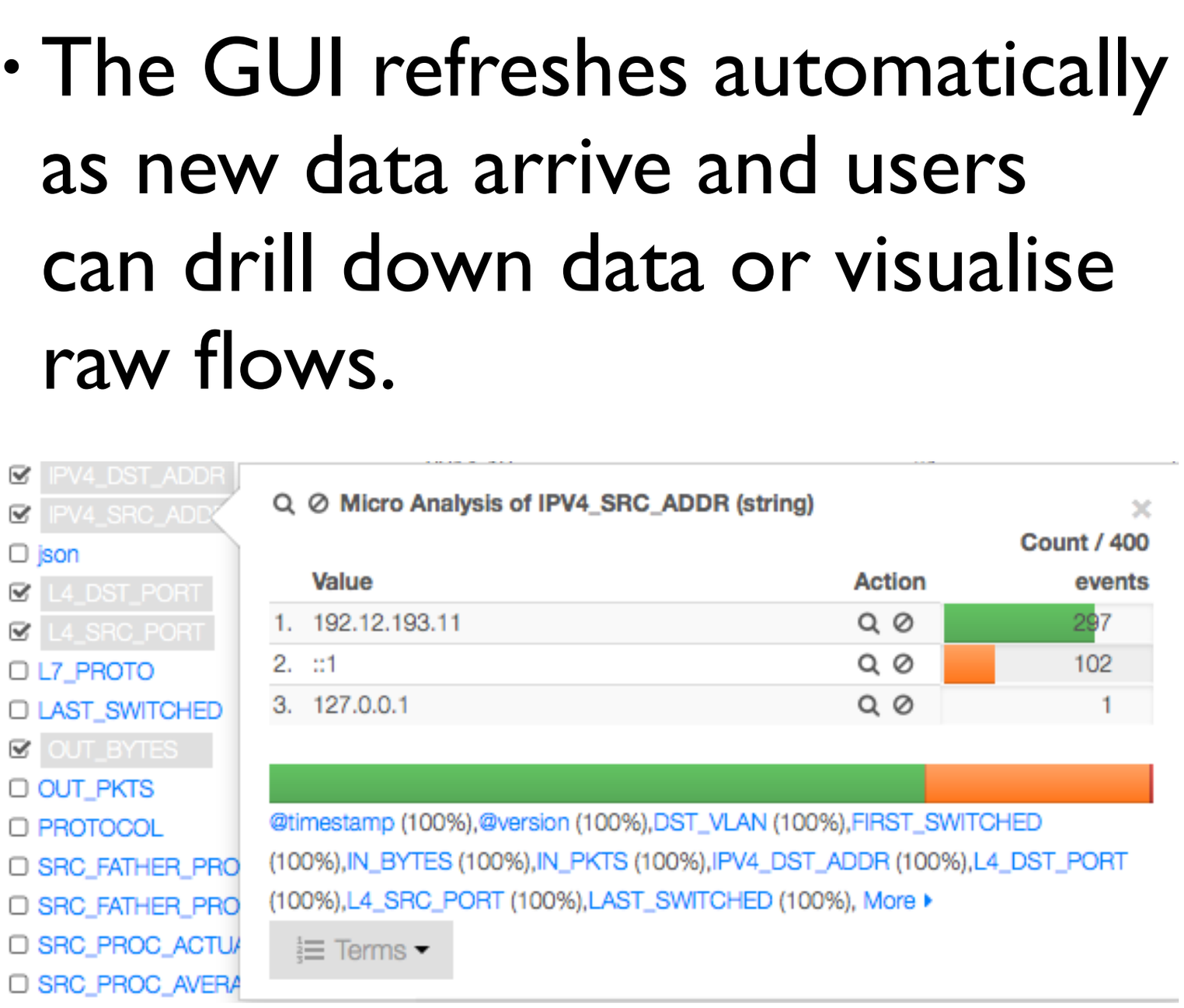
ntopng Process Dashboard in Kibana [1/2]

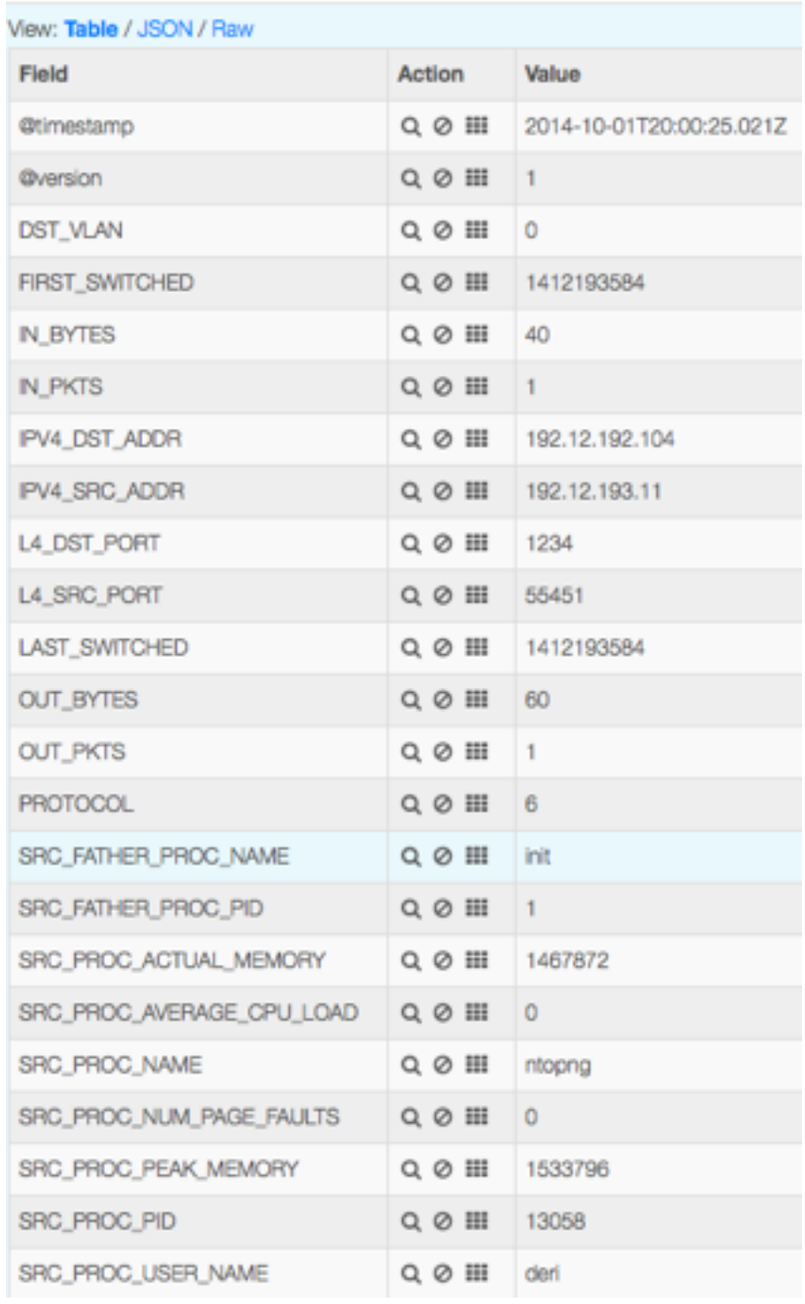


© 2014 - ntop.org

ntopng Process Dashboard in Kibana [2/2]

- The GUI refreshes automatically as new data arrive and users can drill down data or visualise raw flows.



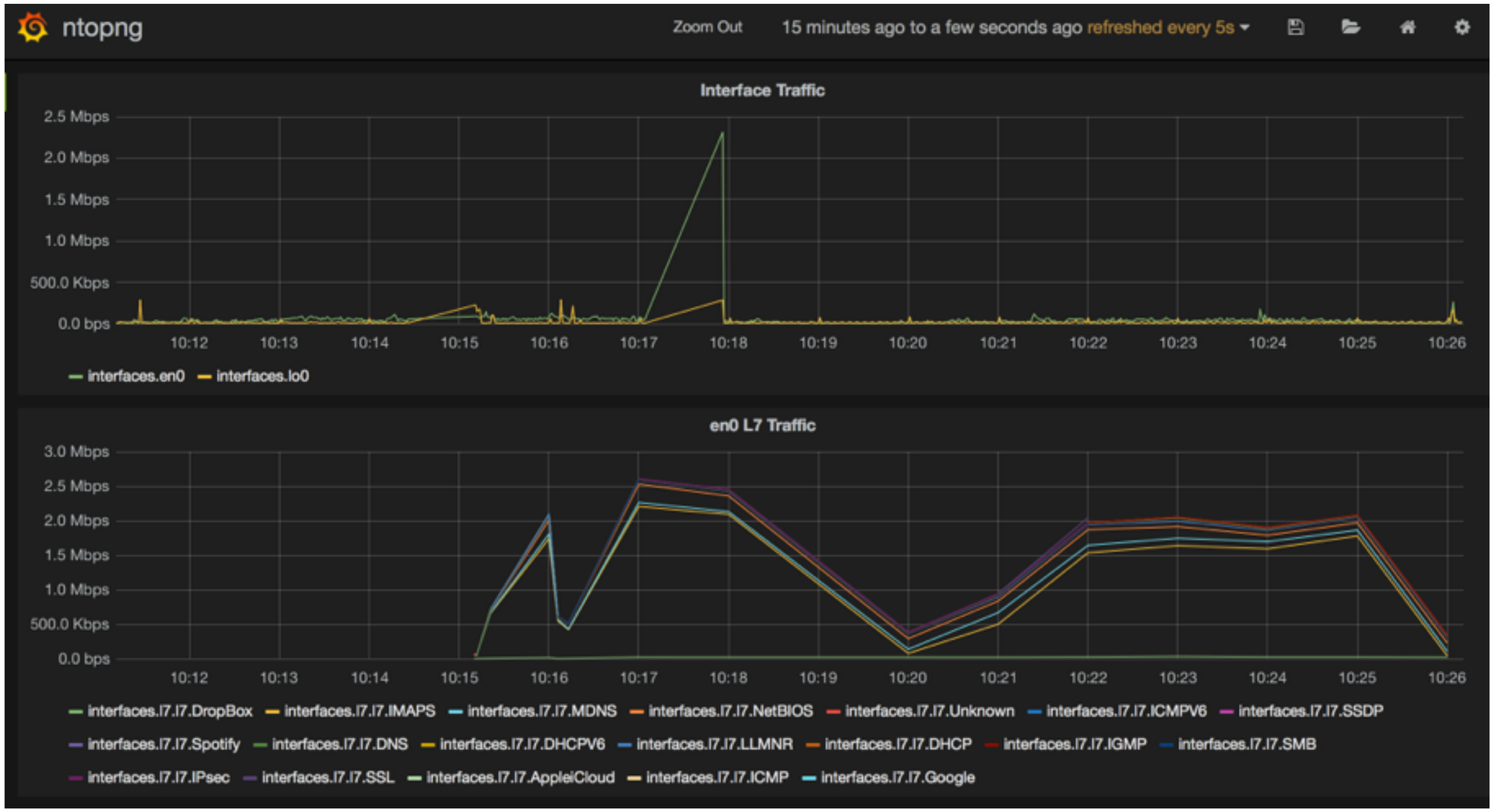


View: **Table** / JSON / Raw

Field	Action	Value
@timestamp		2014-10-01T20:00:25.021Z
@version		1
DST_VLAN		0
FIRST_SWITCHED		1412193584
IN_BYTES		40
IN_PKTS		1
IPV4_DST_ADDR		192.12.192.104
IPV4_SRC_ADDR		192.12.193.11
L4_DST_PORT		1234
L4_SRC_PORT		55451
LAST_SWITCHED		1412193584
OUT_BYTES		60
OUT_PKTS		1
PROTOCOL		6
SRC_FATHER_PROC_NAME		init
SRC_FATHER_PROC_PID		1
SRC_PROC_ACTUAL_MEMORY		1467872
SRC_PROC_AVERAGE_CPU_LOAD		0
SRC_PROC_NAME		ntopng
SRC_PROC_NUM_PAGE_FAULTS		0
SRC_PROC_PEAK_MEMORY		1533796
SRC_PROC_PID		13058
SRC_PROC_USER_NAME		deri

© 2014 - ntop.org

Integrating ntopng with InfluxDB



© 2014 - ntop.org

What's Next on Big Data and ntopng

- We believe that the big data world is still very liquid and it is not clear what the emerging technology will be.
- We believe ntopng should be just a data source without being tightly integrated with any external tool (ntopng speaks JSON and HTTP so we can cover most of them pretty easily).
- We are experimenting with other big data technologies (e.g. druid.io) and we plan to open it to all the emerging technologies available.

© 2014 - ntop.org

ntopng on Virtual Environments

- ntopng has been packaged for major Linux distributions such as Debian/Ubuntu, CentOS/RedHat and also FreeBSD and OSX (brew): installation couldn't be simpler.
- However the current trend is going towards virtualised environments (not just VMs such as VMware) and IaaS (Infrastructure as a Service) and thus we need to support them.





© 2014 - ntop.org

ntopng on Docker [1/5]

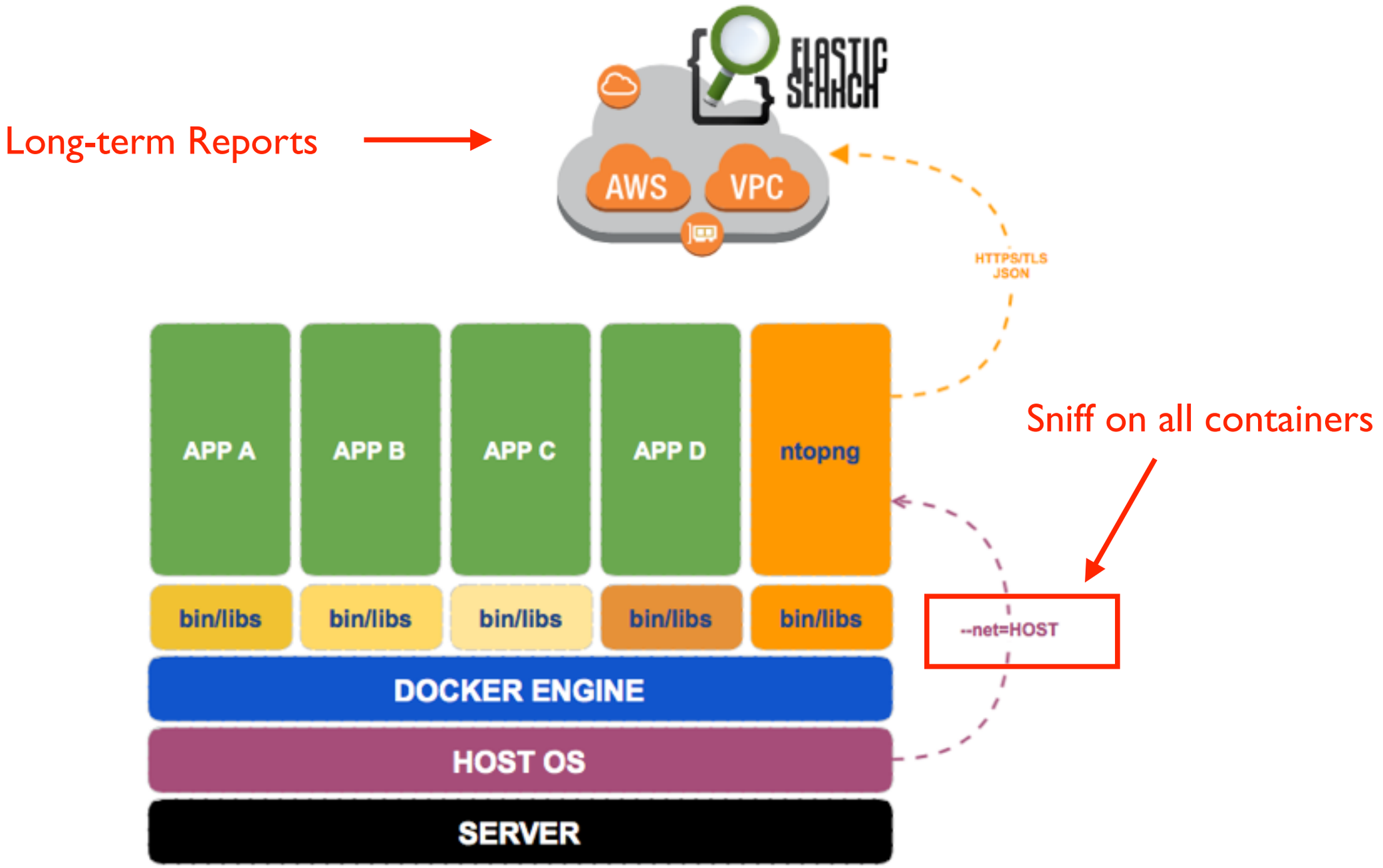
- In essence there are two types of virtualisation:
  - Virtual Machine: emulation of a particular computer system, including its devices (network, storage, USB etc).
  - Operating-system level virtualisation: run multiple isolated user-space instances (often called containers) that look like a real server.



- Docker is an open-source software that automates the deployment of applications inside software containers. Each container runs within a single Linux instance without the overhead of starting VMs.

© 2014 - ntop.org

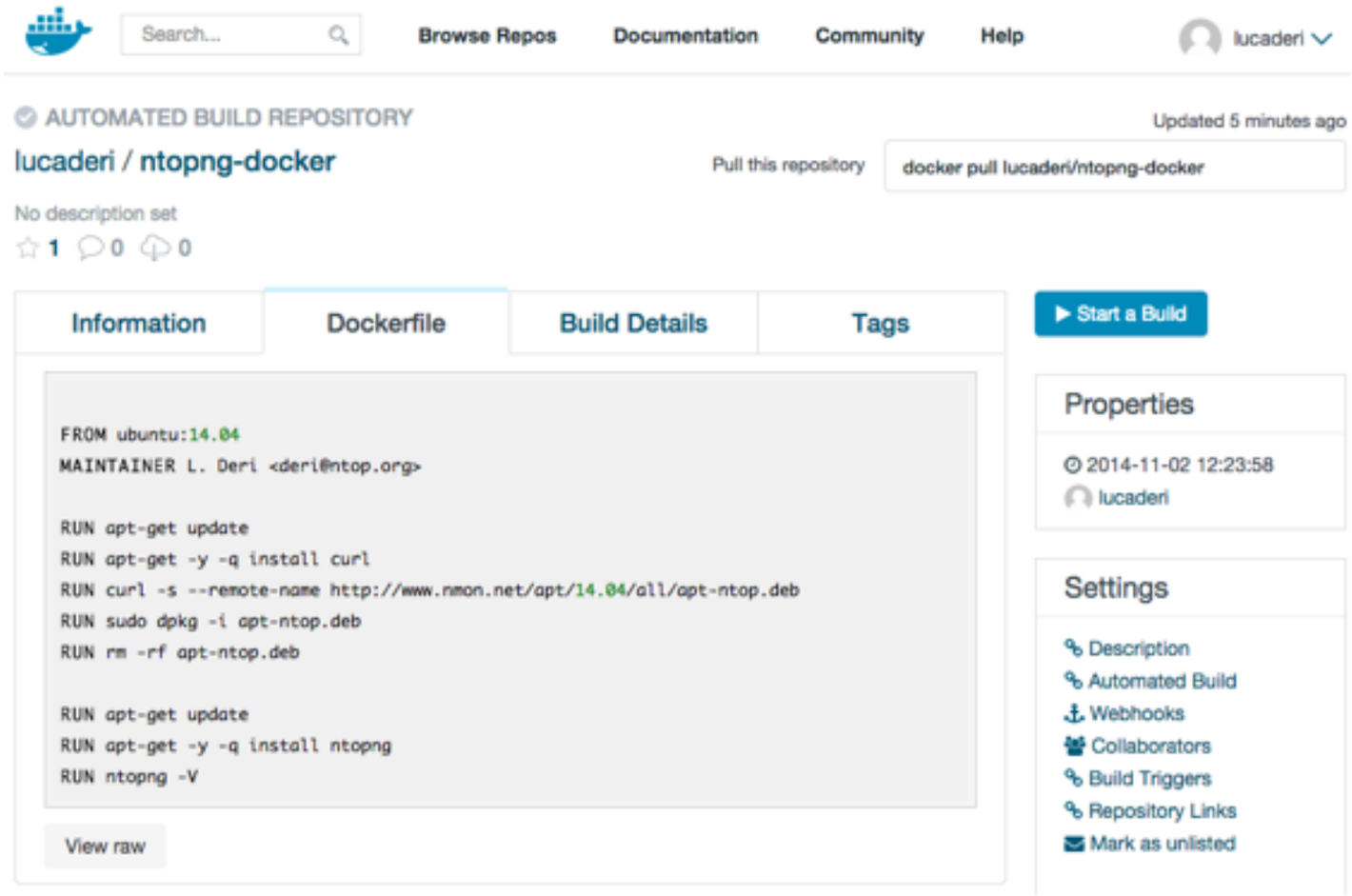
ntopng on Docker [2/5]



© 2014 - ntop.org

ntopng on Docker [3/5]

- A ntopng container allows you to run ntopng on a clean and isolated environment.
- Building a dock is can be done in a few clicks on hub.docker.com



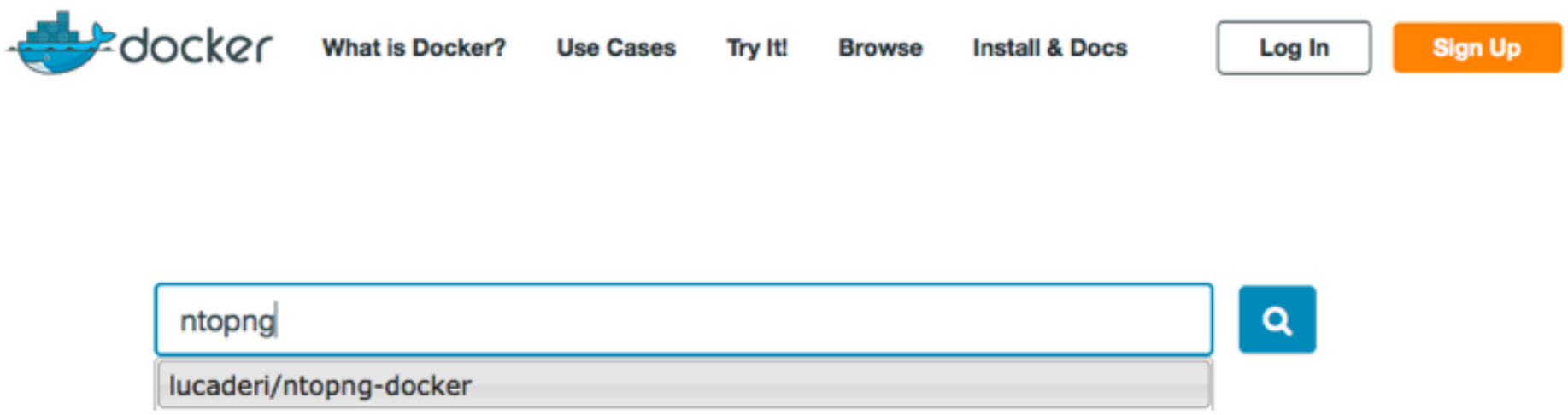
© 2014 - ntop.org

ntopng on Docker [4/5]

- Install docker (http://docs.docker.com/installation/ubuntulinux/)

```
$ sudo apt-get update
$ sudo apt-get install docker.io
$ sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
$ sudo sed -i '$acomplete -F _docker docker' /etc/bash_completion.d/docker.io
$ source /etc/bash_completion.d/docker.io
$ sudo sh -c "echo deb https://get.docker.com/ubuntu docker main > /etc/apt/sources.list.d/docker.list"
$ sudo apt-get update
$ sudo apt-get install lxc-docker
```

- Go do docker.com and search for ntopng



© 2014 - ntop.org

ntopng on Docker [5/5]

- Pull the ntopng container

```
root@ubuntu:/home/deri# docker pull lucaderi/ntopng-docker
Pulling repository lucaderi/ntopng-docker
8077c18a90a8: Download complete
511136ea3c5a: Download complete
d497ad3926c8: Download complete
ccb62158e970: Download complete
e791be0477f2: Download complete
…
e072f31bb2a5: Download complete
9e52f4c92f80: Download complete
ecc46895937f: Download complete
3a3f2545e225: Download complete
4f1229fadea7: Download complete
5b5364929cbf: Download complete
Status: Downloaded newer image for lucaderi/ntopng-docker:latest
```

- Run ntopng on a container

```
root@ubuntu:/home/deri# docker run --net=host --name ntopng -t -i lucaderi/ntopng-docker ntopng -v
….
02/Nov/2014 12:55:20 [main.cpp:183] PID stored in file /var/tmp/ntopng.pid
02/Nov/2014 12:55:20 [HTTPserver.cpp:374] HTTPS Disabled: missing SSL certificate /usr/share/ntopng/httpdocs/ssl/ntopng-cert.pem
02/Nov/2014 12:55:20 [HTTPserver.cpp:376] Please read https://svn.ntop.org/svn/ntop/trunk/ntopng/README.SSL if you want to enable SSL.
02/Nov/2014 12:55:20 [HTTPserver.cpp:420] Web server dirs [/usr/share/ntopng/httpdocs][/usr/share/ntopng/scripts]
02/Nov/2014 12:55:20 [HTTPserver.cpp:423] HTTP server listening on port 3000
02/Nov/2014 12:55:20 [main.cpp:231] Working directory: /var/tmp/ntopng
02/Nov/2014 12:55:20 [main.cpp:233] Scripts/HTML pages directory: /usr/share/ntopng
02/Nov/2014 12:55:20 [Ntop.cpp:218] Welcome to ntopng x86_64 v.1.2.2 (r8539) - (C) 1998-14 ntop.org
```


© 2014 - ntop.org

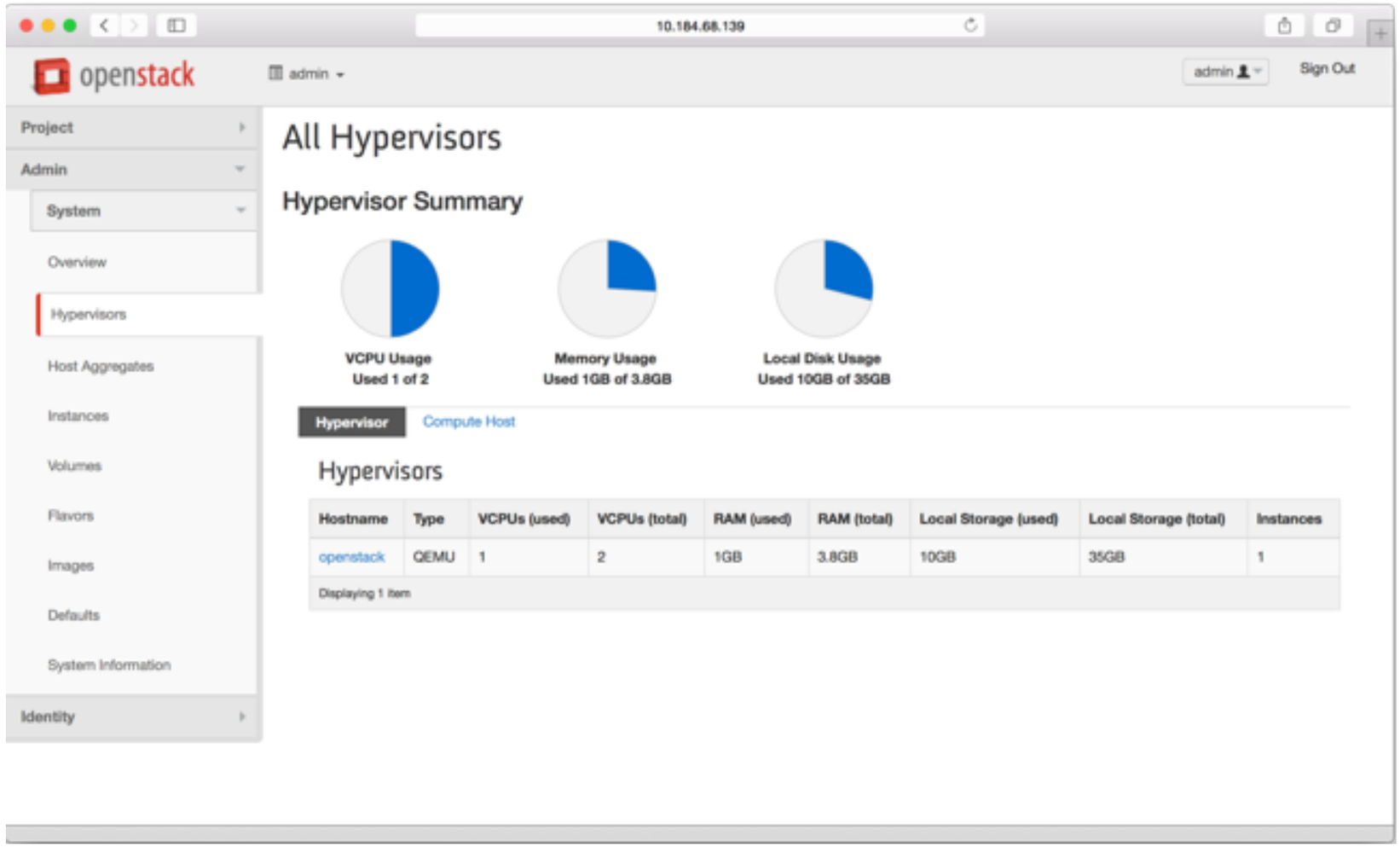
ntopng on OpenStack [1/6]

- OpenStack is a technology that allows to deploy and control resources on a data center (VMs, storage, networking).
- Our interest in OpenStack is manyfold:
  - Create an OpenStack VM image for enabling people to easily deploy ntop monitoring apps on datacenter.
  - Exploit ntop's PF_RING open-source packet processing technology for bringing packets in 0-copy at 10 Gbit on a VM managed by OpenStack. This is to enable efficient traffic monitoring on a data center.

© 2014 - ntop.org

ntopng on OpenStack [2/6]

- In OpenStack, VMs are KVM-based and are managed though the OpenStack controller.



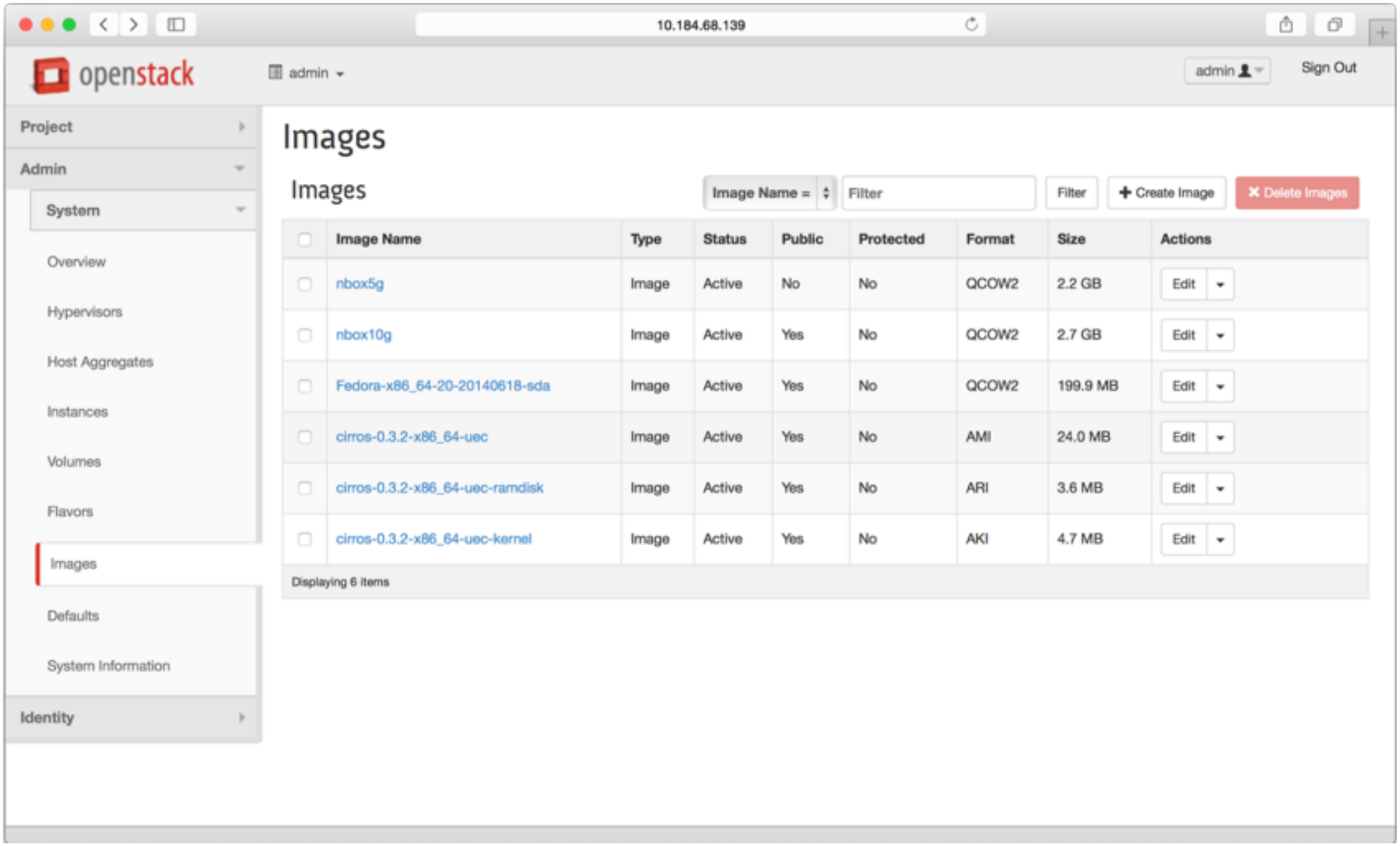
© 2014 - ntop.org

ntopng on OpenStack [3/6]

- Through OpenStack we want to be able to deploy VMs with ntopng and attach them to virtual controllers (Open vSwitch) or 0-copy PF_RING ZC-based packet sources.
- With ZC, packets are captured in 0-copy from network adapters and deployed in 0-copy to VMs.
- ZC packets are deployed on the VM using virtual adapters attached dynamically to the VM though a ntop-developed kernel module based on PCI hotplug.

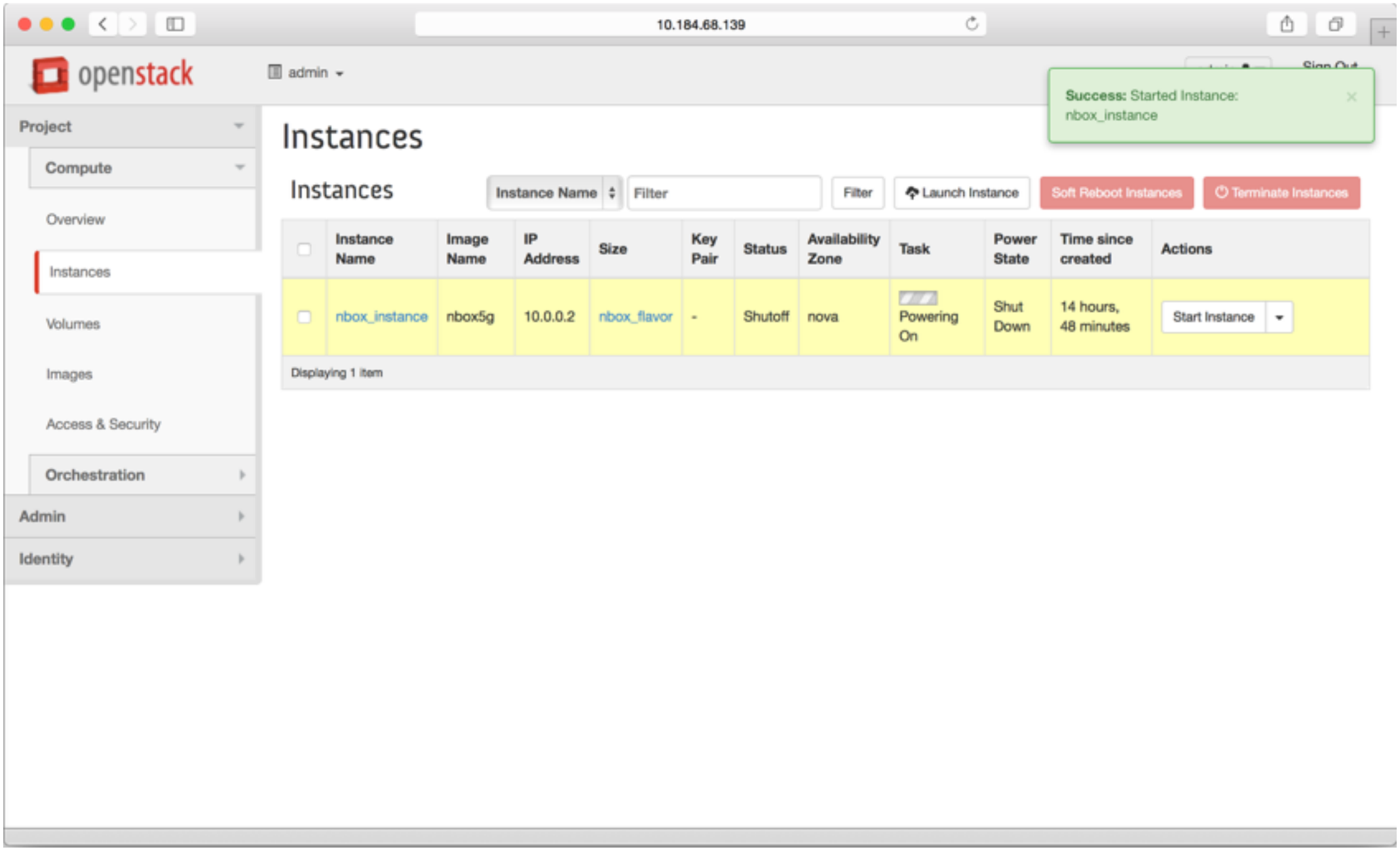
© 2014 - ntop.org

ntopng on OpenStack [4/6]

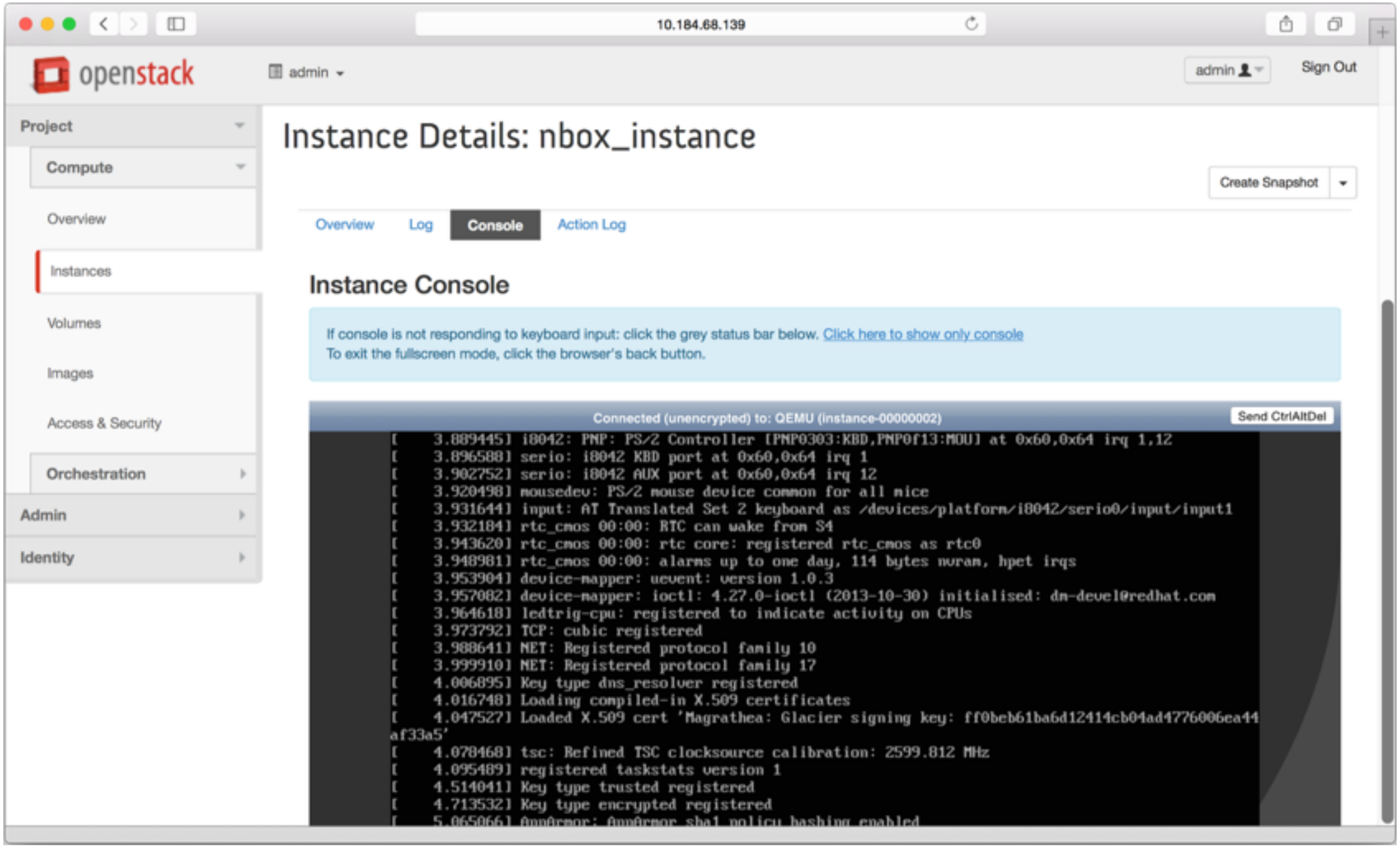


NOTE: OpenStack image available from the ntop web site

ntopng on OpenStack [5/6]



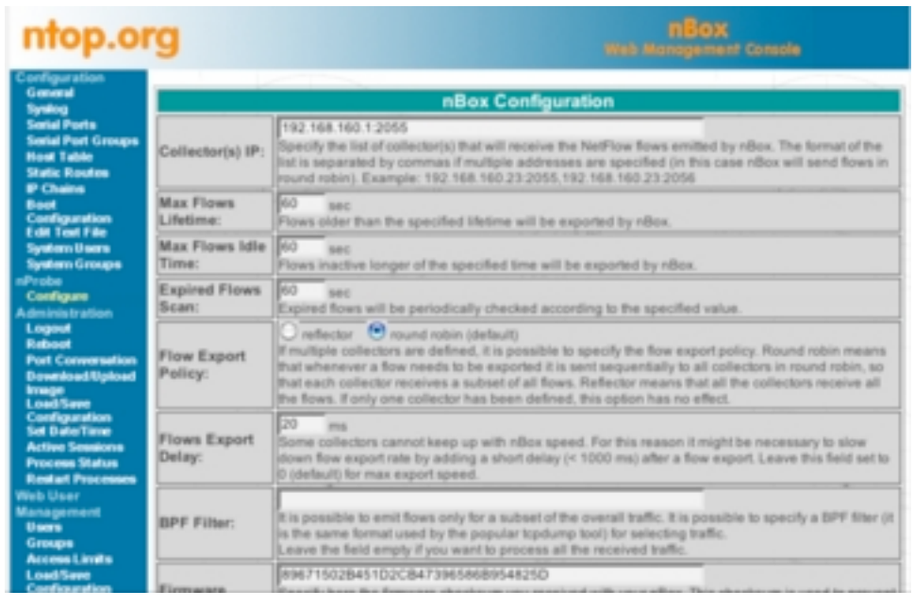
ntopng on OpenStack [6/6]



Embedding ntopng [1/4]

- Historically we have started our first embed attempt in 2003 with the Cyclades TS100.
- The nBox was used to analyse traffic then sent to ntop for representation.
- After 10 years we have tried again with ntopng.

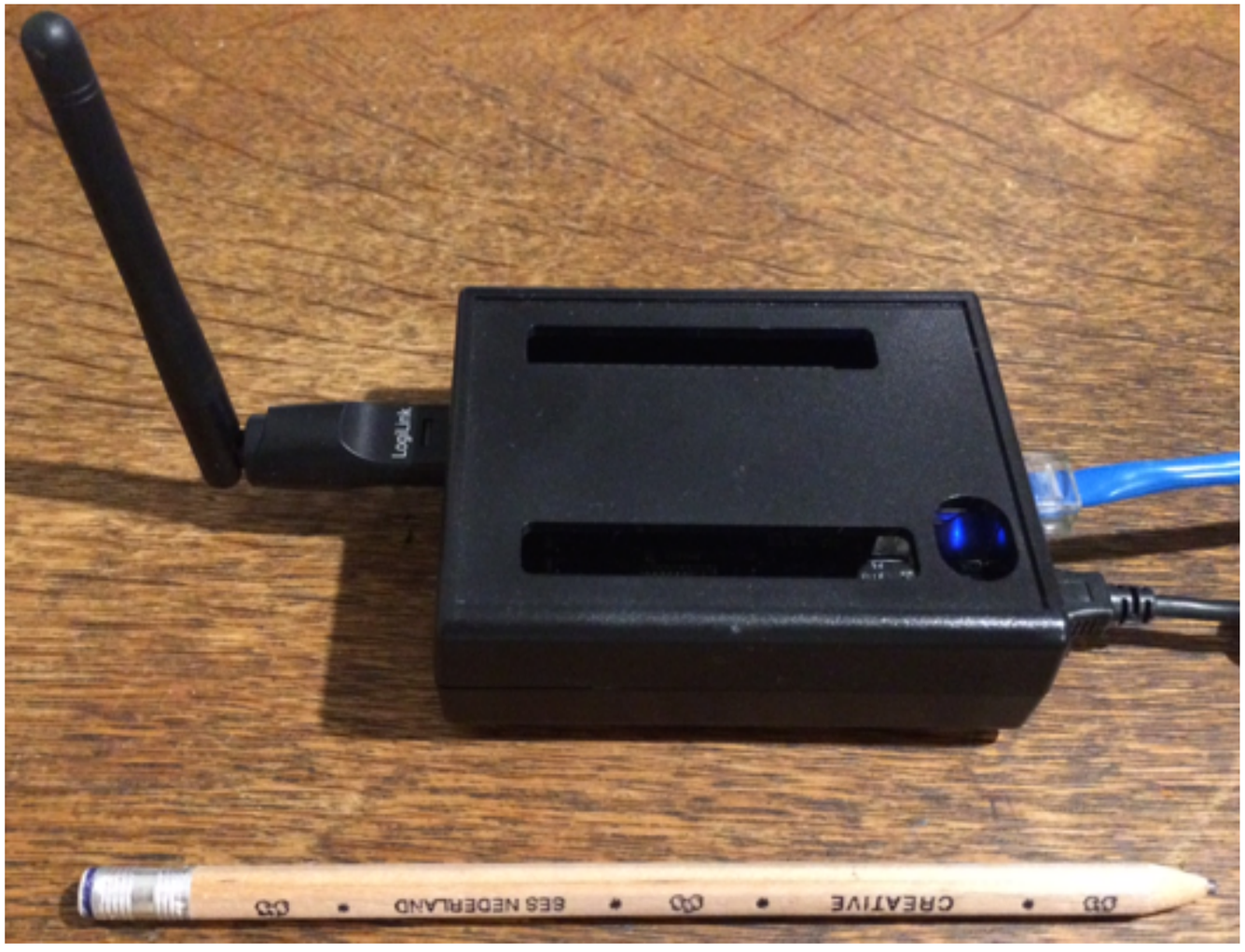




© 2014 - ntop.org

Embedding ntopng [2/4]

- It is a while that we are working towards a cheap platform for everyone…



BeagleBoard Black

© 2014 - ntop.org

Embedding ntopng [3/4]

- Main issue with boards like BeagleBoard/Raspberry: only one ethernet interface built-in (extra ports via USB).
- Boxes like Ubiquity Networks EdgeRouter are also an option but we're basically jeopardising a box designed for other tasks (issues with hardware guarantee, GUI etc.).



- Open issues: how to monitor traffic? Port mirror or tap?

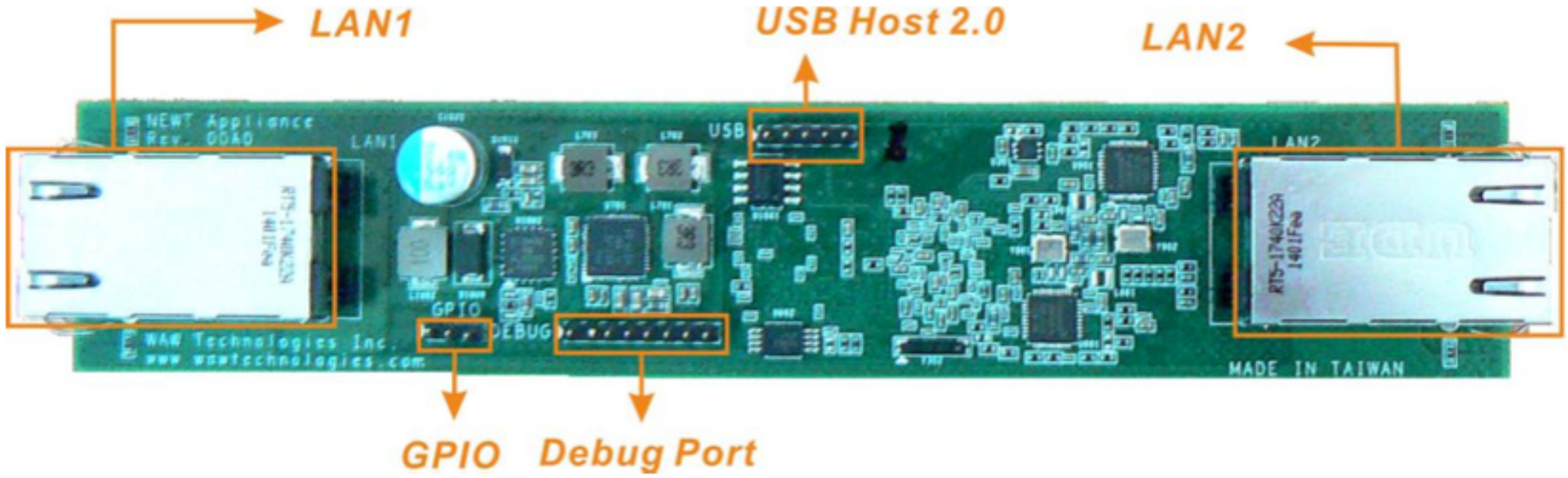
© 2014 - ntop.org

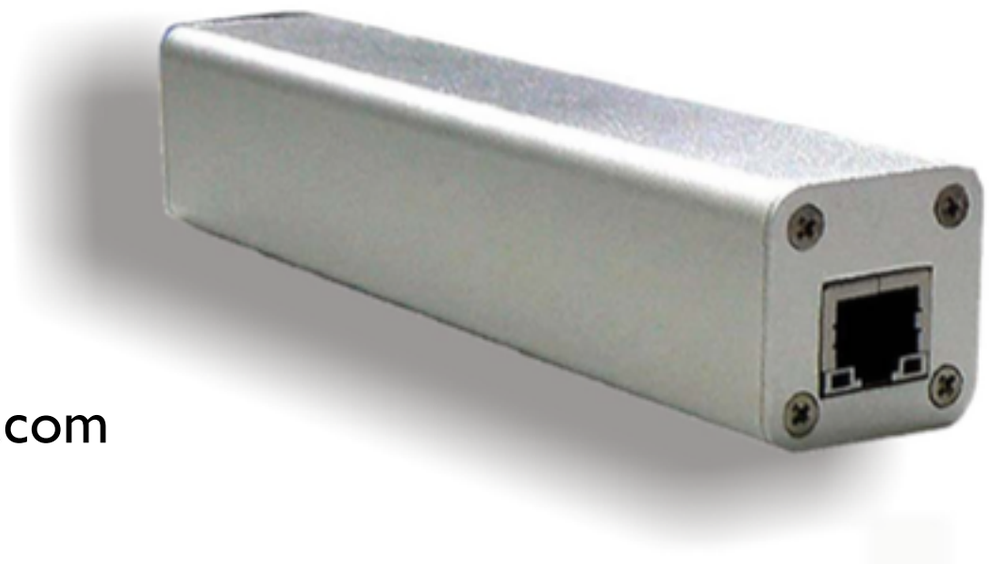
Embedding ntopng [4/4]

- We're trying to find the third way…
  - Rely on a hardware company to build a cheap ARM-based box suitable for network monitoring (ntop is making software no hardware).
  - Two ethernet interfaces to be used as either a bump-in-the-wire or 2 x independent interfaces.
  - Built-in hardware tap with bypass.
  - Able to monitor xDSL/cable and up.
  - Power-over-Ethernet (POE).

© 2014 - ntop.org

Current Prototype [1/3]





www.wawtechnologies.com

© 2014 - ntop.org

Current Prototype [2/3]

- Hardware Roadmap
  - Two models to be released in 2015.
    - Entry level model (cheap, no POE, tap w/o bypass).
    - Enterprise model (POE, tap+bypass, copper/optical).
- ntopng Software Roadmap
  - Ability to control various distributed ntopng instances from a central location.
  - Cloud-based/like storage of distributed probes.
  - Passive and active (basic connectivity tests) edge monitoring for measuring quality-of-experience at the customer site.

© 2014 - ntop.org

Current Prototype [3/3]

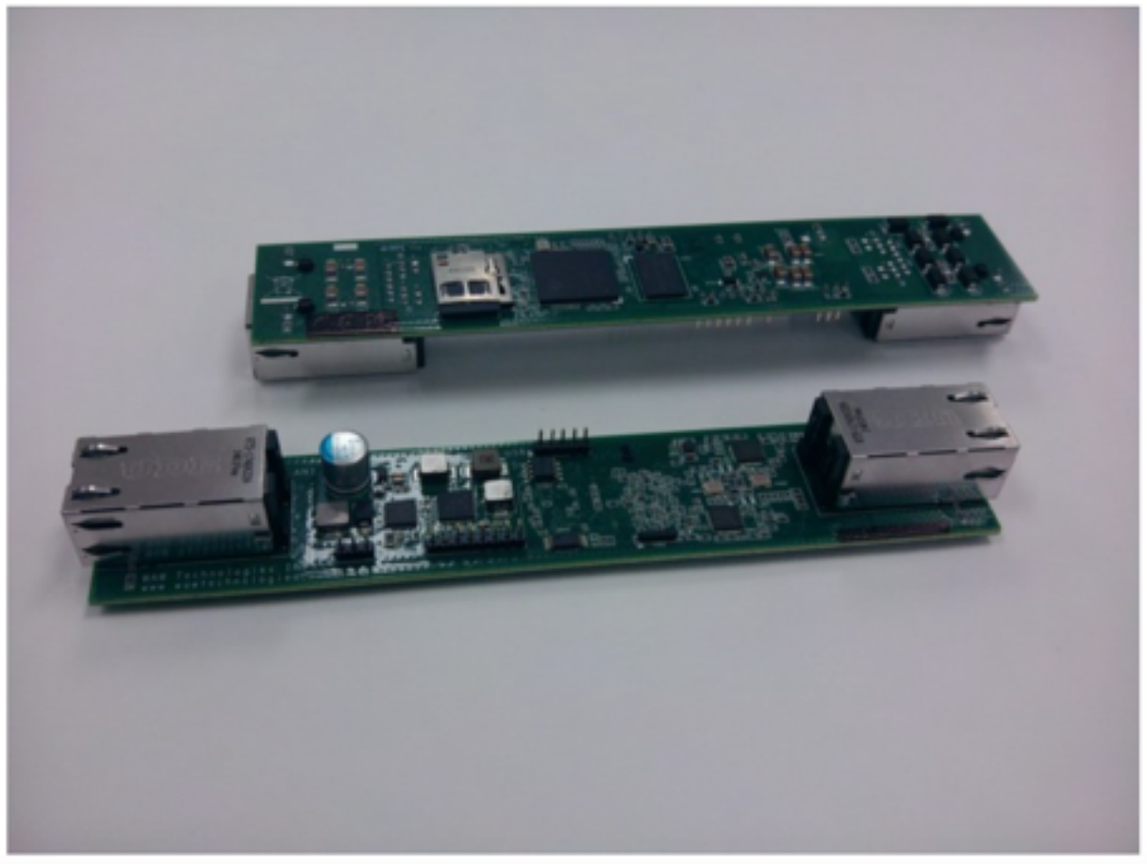
KICKSTARTER

NEWT - Network Explorer Wire TAP

by WAW Technologies Inc.

Home Updates 0 Backers 0 Comments 0 Ottawa, Canada Hardware

THIS PROJECT IS NOT LIVE This is only a draft that the creator has chosen to share.



0
Backers

$0.00 CAD
pledged of $12,000 goal

Project by

WAW Technologies Inc.
Ottawa, Canada

First created · 0 backed

Has not connected Facebook

wawtechnologies.com

See full bio | Contact me

© 2014 - ntop.org

Final Remarks

- Over the past 16 years ntop created a software framework for efficiently monitoring traffic.
- "We have a story to tell you, not just hacks".
- Commodity hardware, with adequate software, can now match the performance and flexibility that markets require. With the freedom of open source.
- ntopng is available under GNU GPLv3 from http://www.ntop.org/.

© 2014 - ntop.org