PERFECTLY IMPERFECT

19-20-21 agosto
2016
Parco Ex Caserma Cocco
Pescara

# NtopNg e il monitoraggio del traffico di rete (in high-speed network)


**Giuseppe Augiero**

<talk@augiero.it> - @GiuseppeAugiero

**Luca Deri**

<deri@ntop.org> - @lucaderi

# Outlook

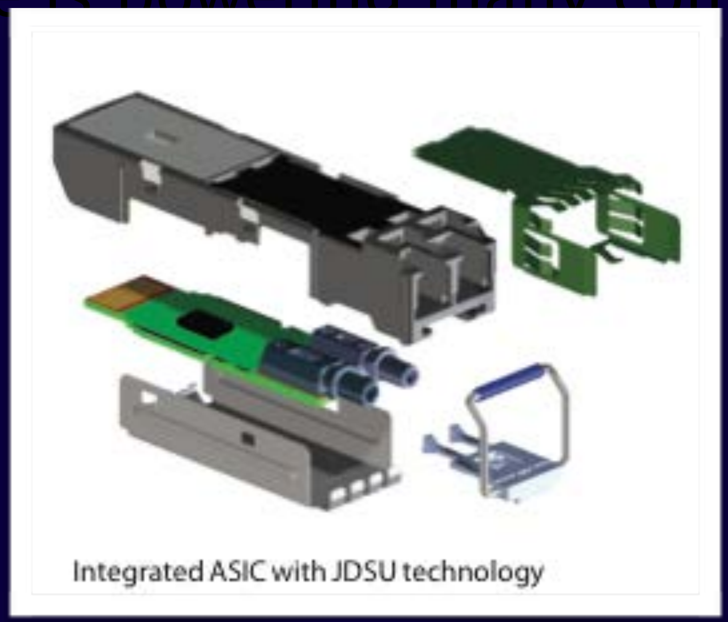
- What are the main activities of ntop.org ?
- Ntop’s view on network monitoring.
- From ntop to ntopng.
- Ntopng architecture and design.
- Ntopng as a flow collector
- Exploring system activities using ntopng
- Using ntopng.
- Advanced monitoring with ntopng.
- Future roadmap items.

# 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).

# About ntop.org [2/2]

Our software is powering many commercial products...





# Ntop Goals

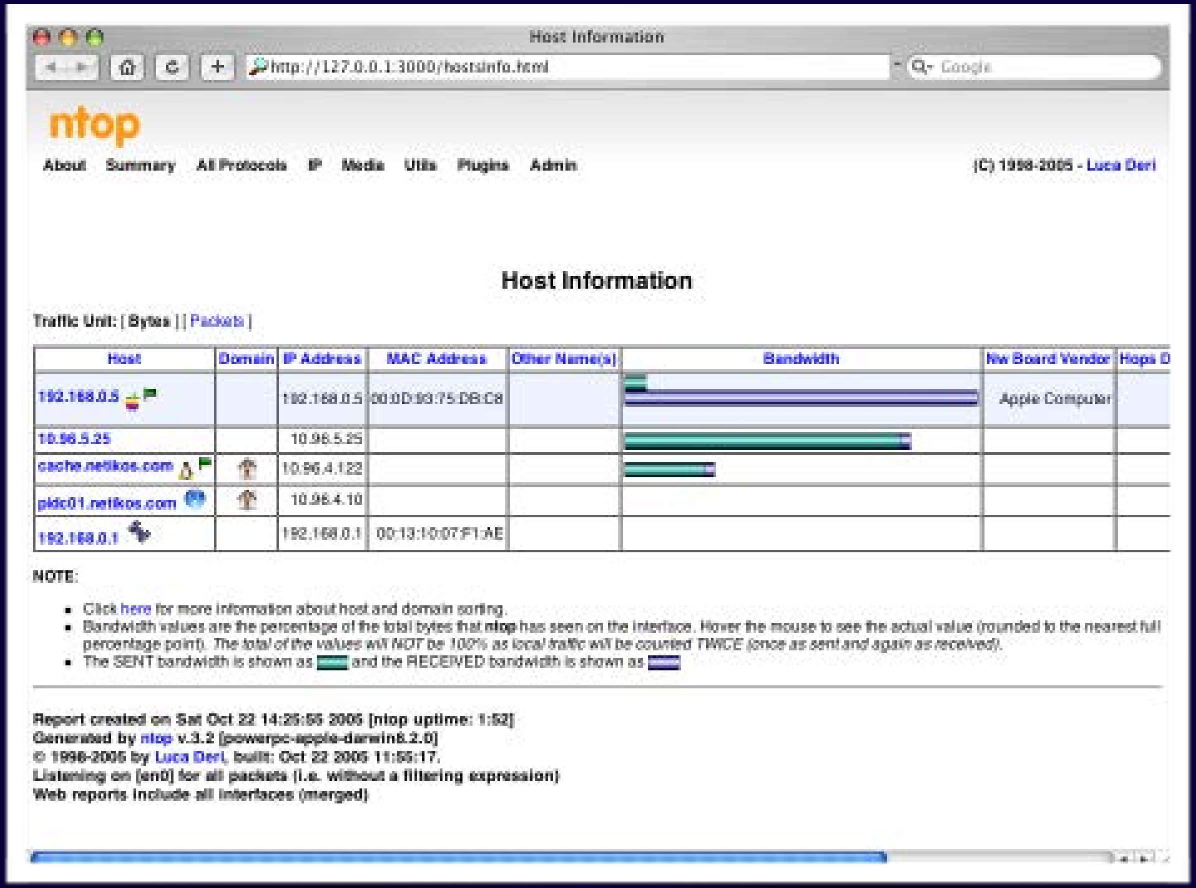
- Provide better, yet price effective, traffic monitoring solution by enabling users to have increased traffic visibility.
- Go beyond standard metrics and increase traffic visibility by analysing key protocols in detail.
- Provide users comprehensive and accurate traffic reports able to offer at a fraction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

# 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.

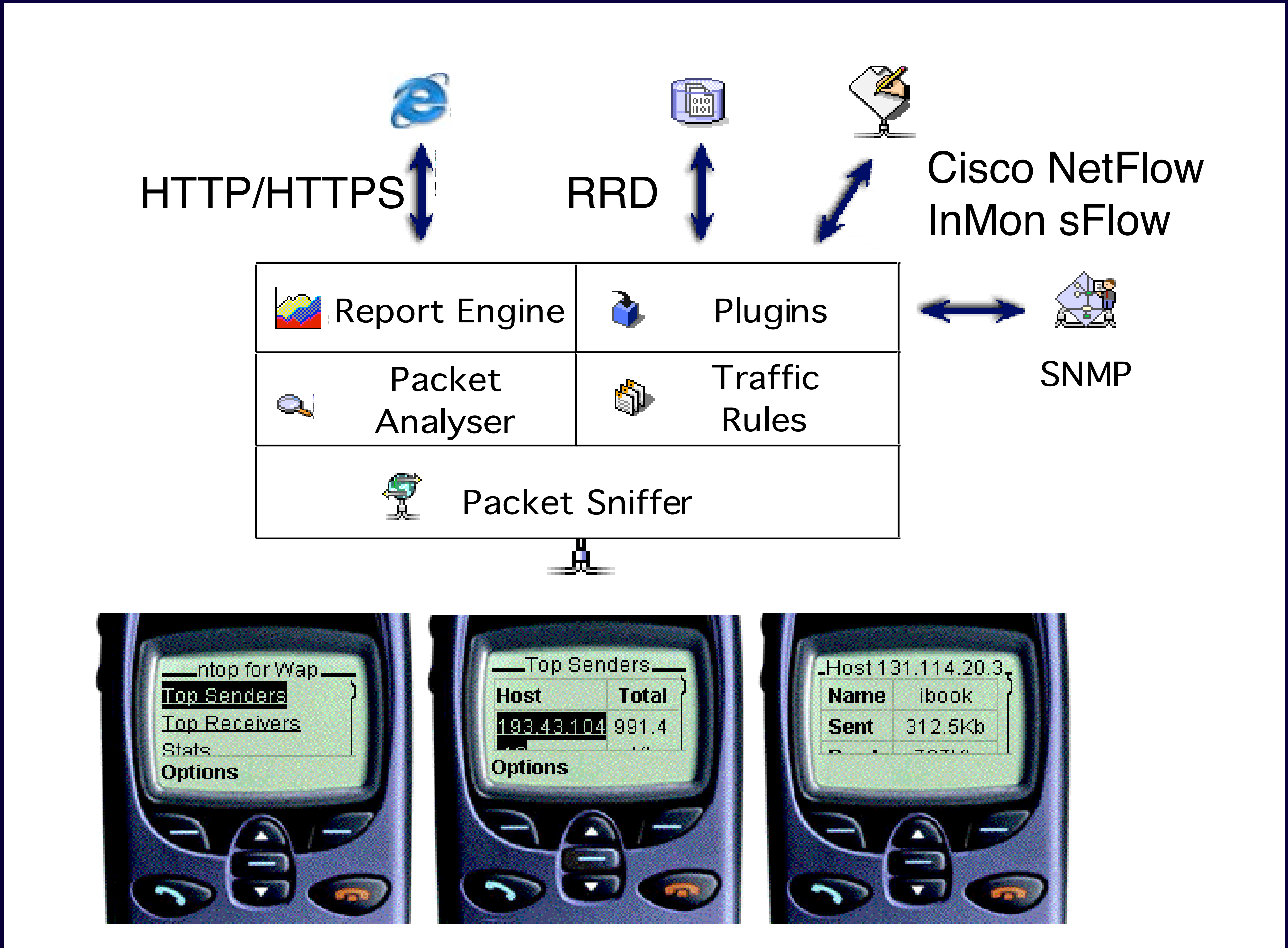
# Some History

- 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.

# Ntop Architecture



# Why was Ntop obsolete?

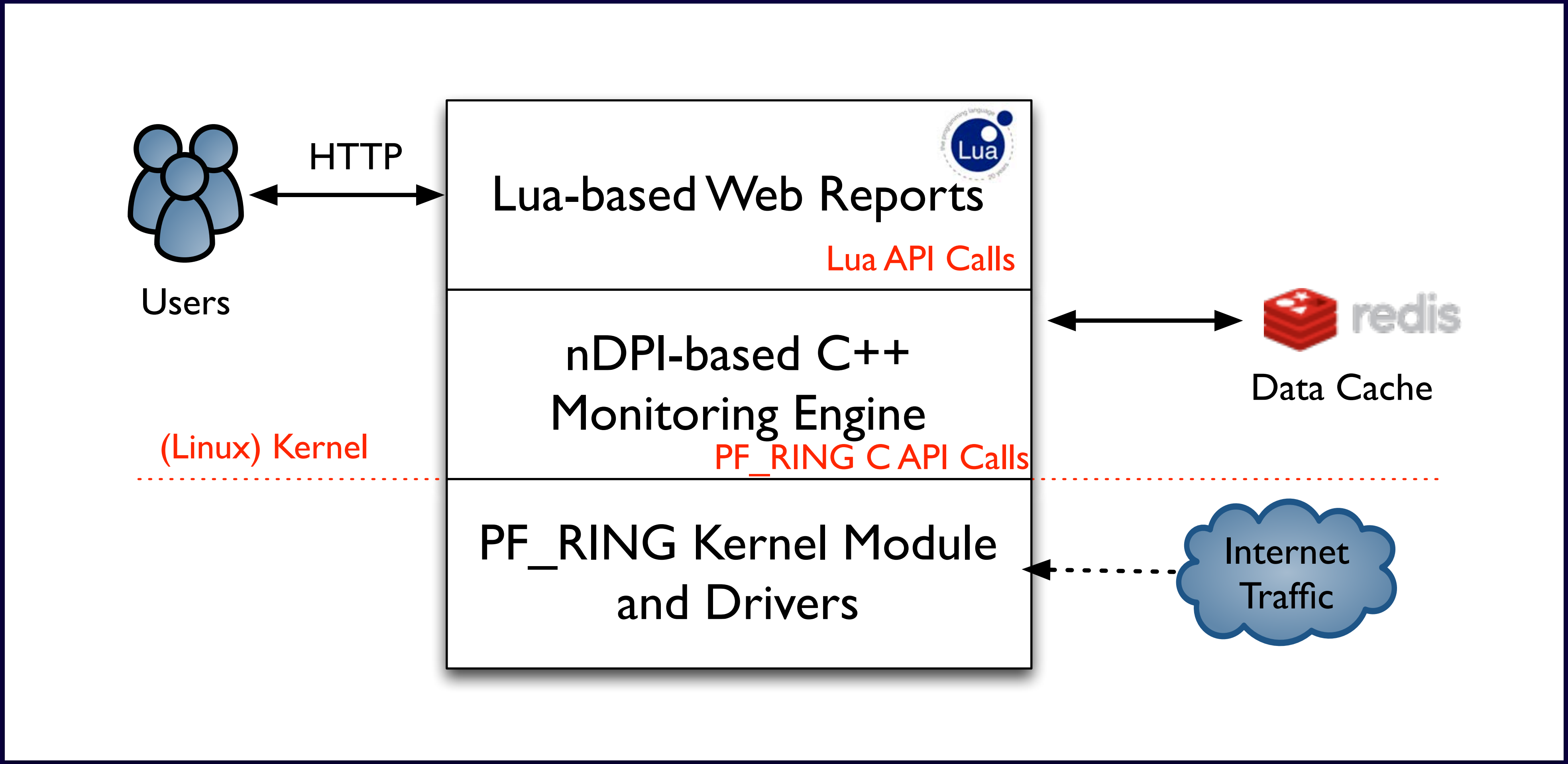
- Its original LAN-oriented design prevented ntop from handling more than a few hundred Mbit.
- The GUI was an old (no fancy HTML 5) monolithic piece written in C so changing/extending a page required a programmer.
- Ntop could not be used as web-less monitoring engine to be integrated with other apps.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

# 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.

# Ntopng Architecture

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



# Network Traffic & Probe

- Where to place the probe?
  - Near the router?

- And the Network Edge?

- Probe:
  - Passive (only analysis).
  - Active (analysis and block).

# Network measures

- Quantitative:
  - Top Talkers.
  - Protocols or applications.
  - Destinations.
  - Host counters.
- Qualitative:
  - Traffic not allowed.
  - Errors.

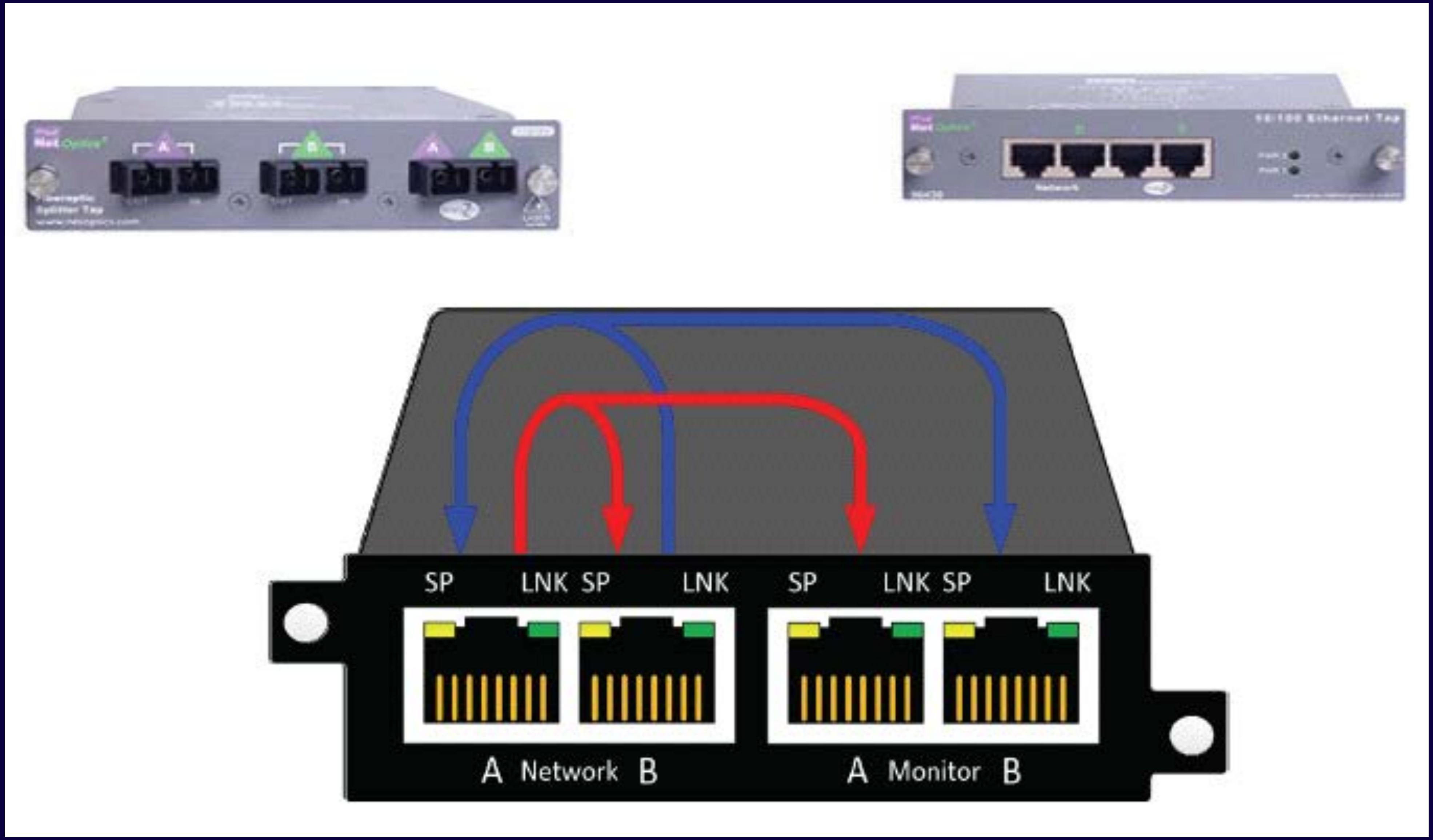
# Some problems

- Security issues
  - All the network traffic is captured and not just the one sent to the sniffing host.
  - If there is a switched network it is captured only a part of traffic.
  - Usability limited to those who have root capabilities.
- Performance
  - Sniffer implies also the cpu load because all the captured packets must be analysed by the program and not just those directed to the host.

# Traffic mirror

- Hardware:
  - Hub (Copper Ethernet).
  - Optical Splitter (Optical Fibers).
  - Tap (Copper/Fiber).
- Software:
  - Switch Port Mirror (1:1, 1:N).
  - Switch VLAN Mirror (N:1).
  - Switch Traffic Filter/Mirroring (Packet Brokers).

# Network Taps



# 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

# 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.

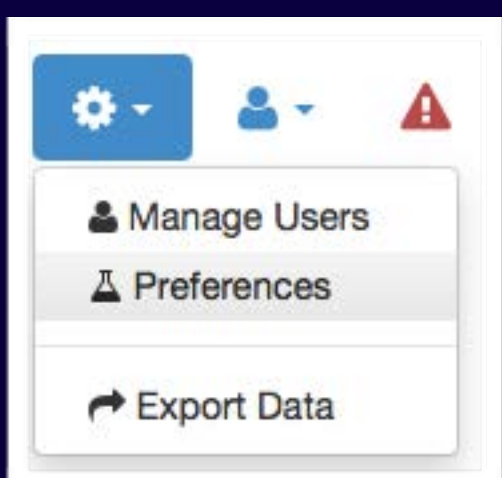
# Local vs Remote Hosts [2/2]

- 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 |

# 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:

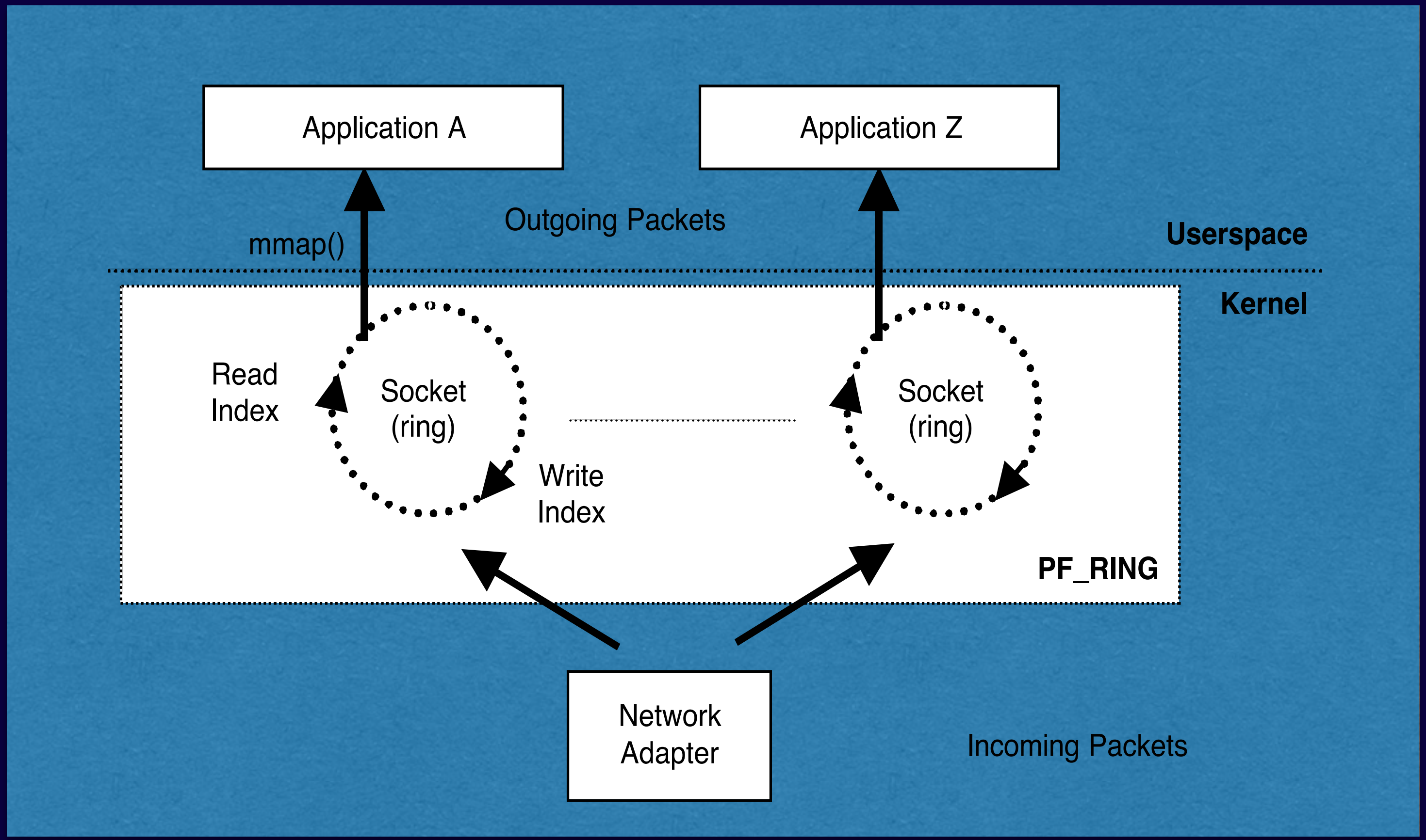



# Packet Processing Journey

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.

# 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.

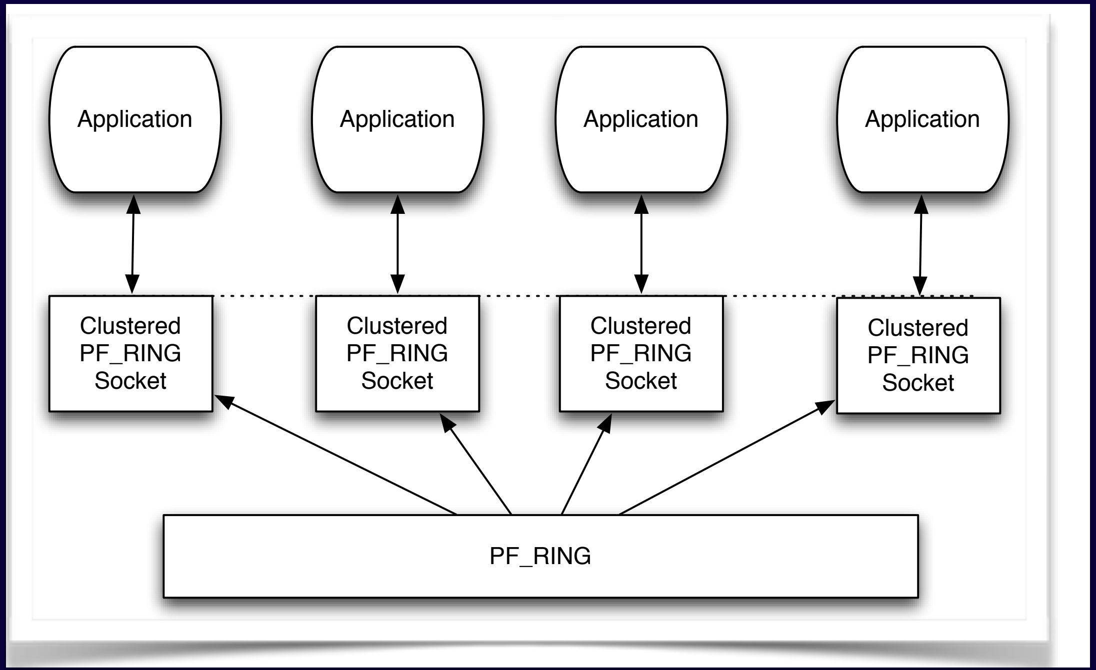


# PF_RING [2/2]

- 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.

# Balancing Traffic with PF_RING

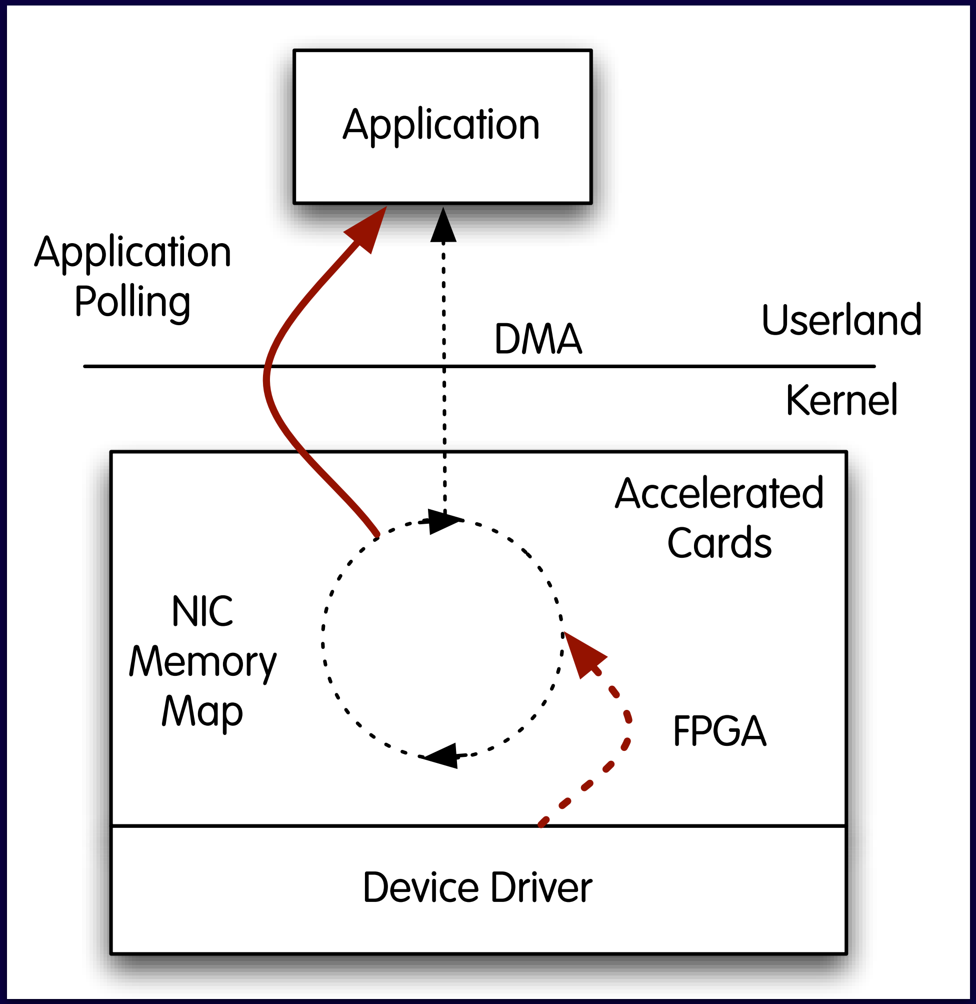
- At high speed on modern multi-core systems, it is a good idea to improve the overall system performance by balancing traffic across cores.




- PF_RING shares ingress packets across multiple consumer applications (e.g. ntopng) by hashing them (tunnels are supported) so that they are balanced to multiple consumer applications via virtual PF_RING network interfaces.

# 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 be received/transmitted in zero copy similar to what FPGA-accelerated cards (e.g. Napatech) do in hardware.



# Moving towards 10 Gbit and above [2/2]

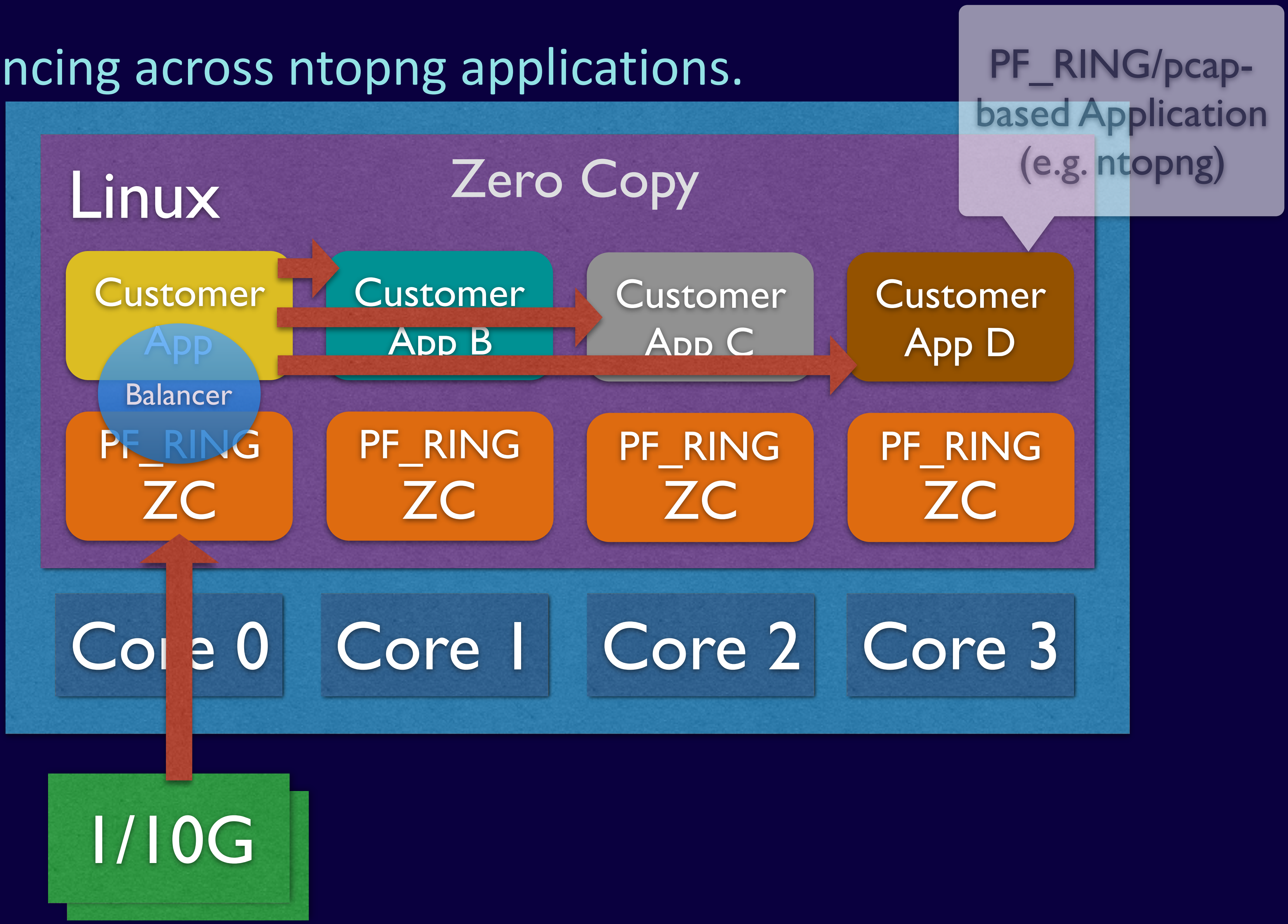
- In ZC a packet is put by the ingress NIC into a shared memory buffer, and it hop across applications (and VMs) by exchanging the buffer pointer (packets don’t move).

- 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).

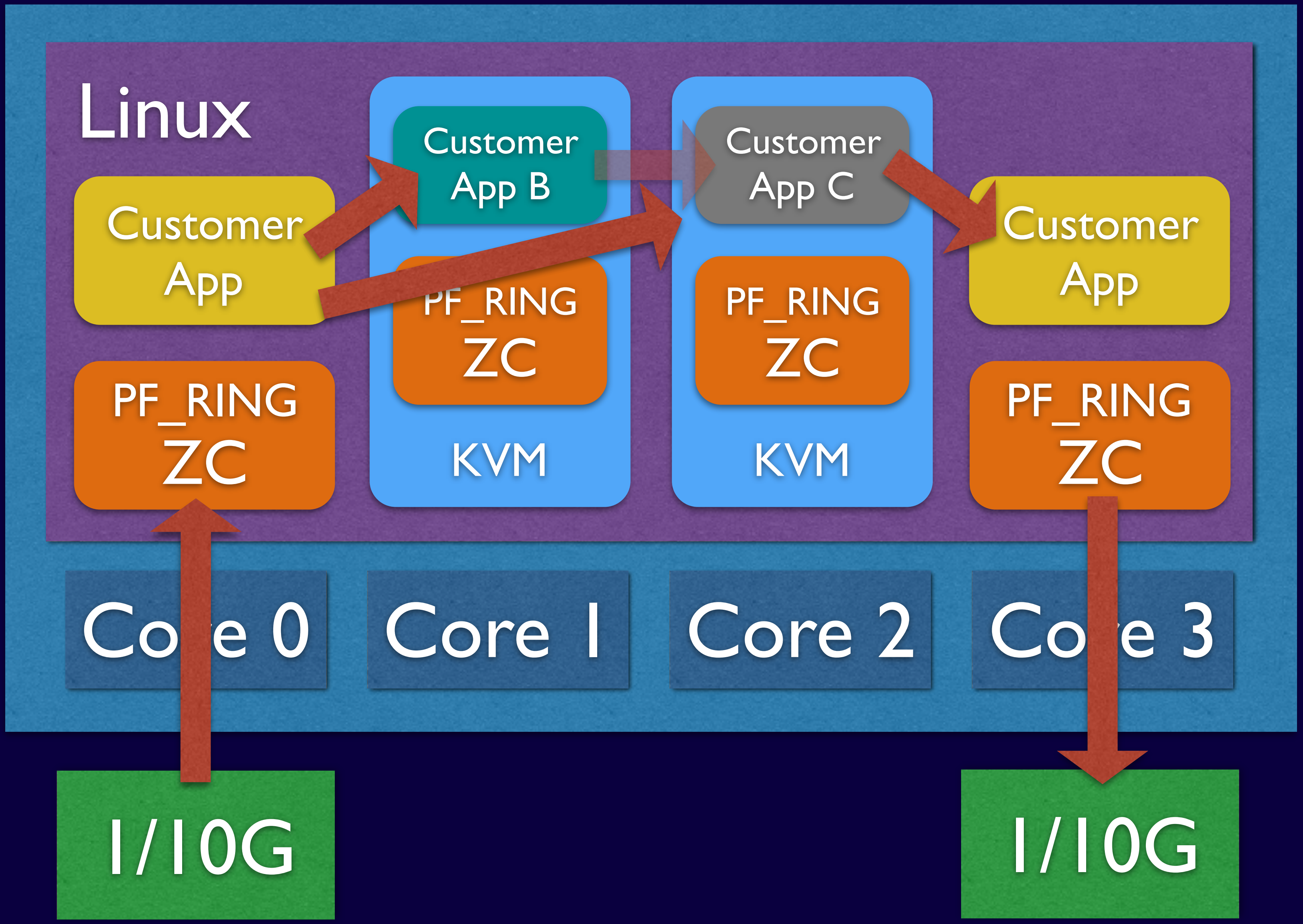
# PF_RING ZC Network Topologies [1/2]

Use Case:

Load balancing across ntopng applications.



# PF_RING ZC Network Topologies [2/2]




Use Case:

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

# 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).

# The need for DPI in Monitoring [1/2]

- Limit traffic analysis at packet header level it is no longer enough (nor cool).
- Network administrators want to know the real protocol without relying on the port being used.
- Selected protocols can be "precisely dissected" (e.g. HTTP) in order to extract information, but on the rest of the traffic it is necessary to tell network administrators what is the protocol flowing in their network.

# 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.

# 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 (> 220) 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)

# 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).

# nDPI API

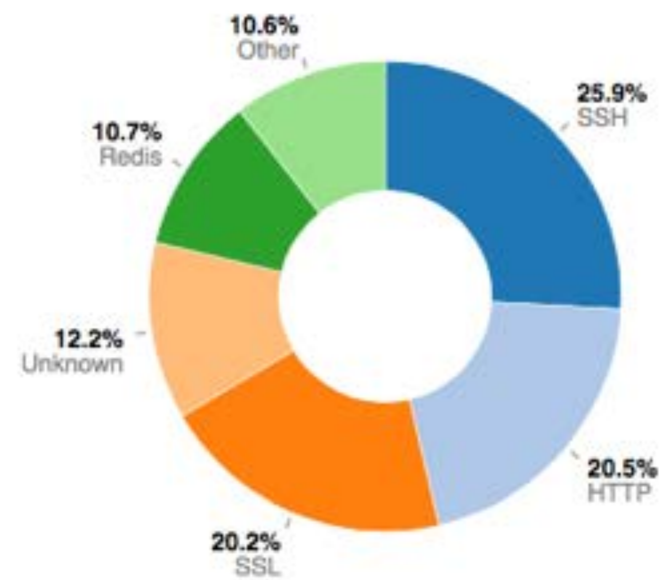
- The nDPI API is pretty simple
  - `ndpi_init_detection_module()`
    `ndpi_exit_detection_module()`
    Init/term the nDPI library.
  - `ndpi_load_protocols()`
    Load custom protocol definitions.
  - `ndpi_detection_process_packet()`
    Process the packet in nDPI and return the L7 protocol or NDPI_UNKNOWN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

# 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).

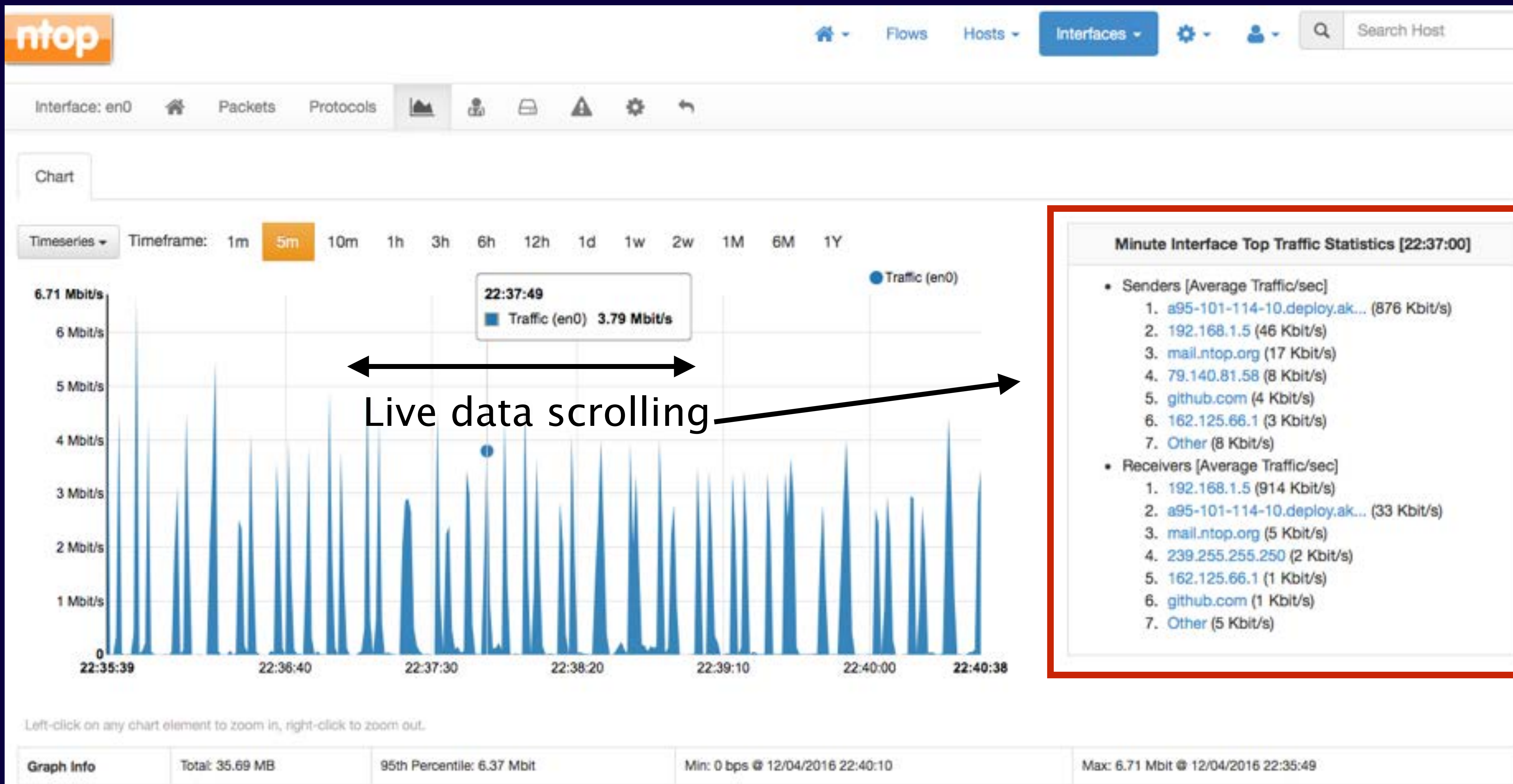
# nDPI on ntopng: Interface Report [1/2]

Protocol Overview




| Application Protocol | Total (Since Startup) | Percentage |
|---|---|---|
| Apple | 17.94 KB | 0 % |
| BitTorrent | 90.59 KB | 0 % |
| CiscoVPN | 560 Bytes | 0 % |
| DCE_RPC | 2.65 KB | 0 % |
| DHCP | 1.09 MB | 0 % |
| DHCPV6 | 3.38 KB | 0 % |

# nDPI on ntopng: Interface Report [2/2]



# Ntopng and Redis [1/2]

- Redis is an open source key-value in-memory database.
- Ntopng 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.

# 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.

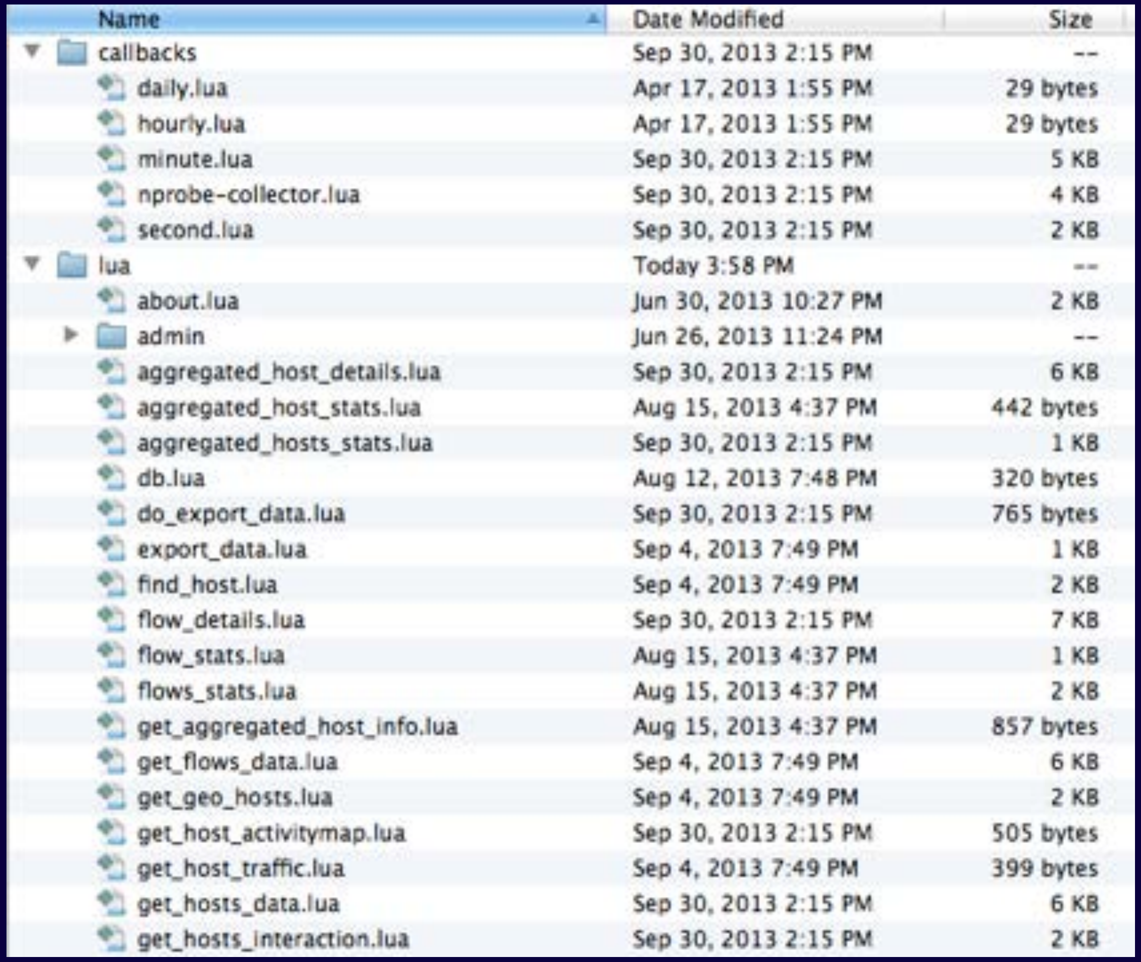
- In essence Redis is used by ntopng to store information that might take too much memory (if kept on ntopng memory space), or to pile up list of things that are executed periodically or that require interaction with remote applications that might be slow or temporary unavailable.

# 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.

# 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.



| 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 |

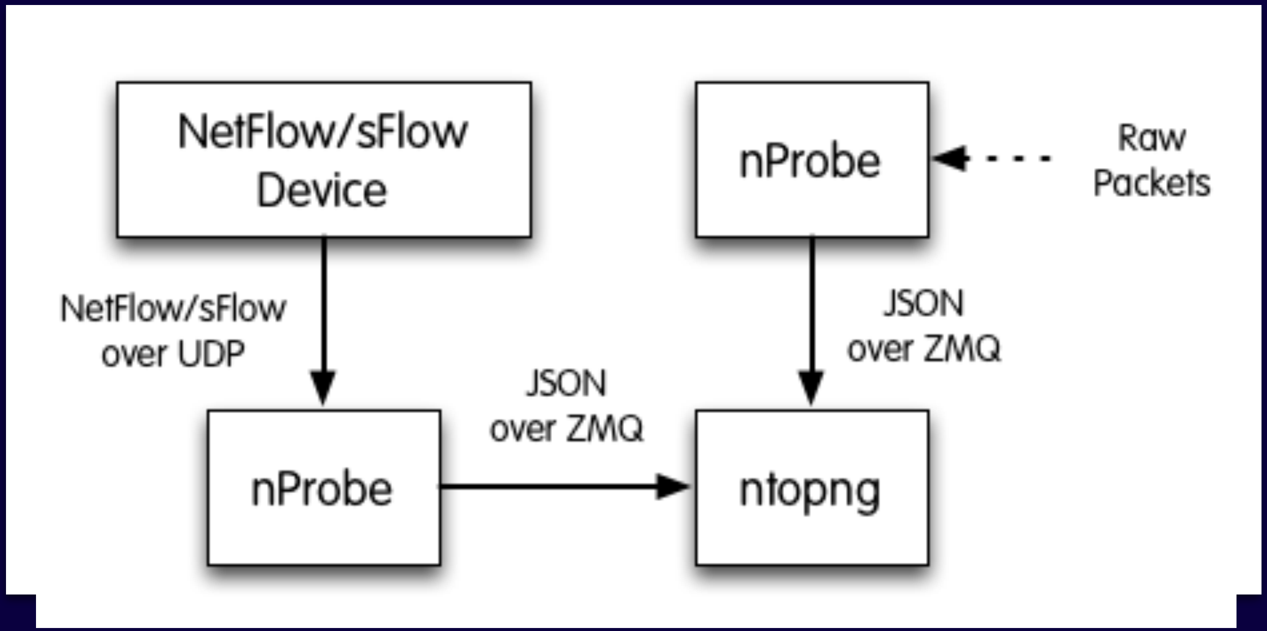
- Example:
  `http://ntopng:3000/lua/flow_stats.lua`

# 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.

# 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.



# 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.

# Ntopng as a NetFlow/sFlow Collector [3/3]

Flows are sent in the following format (gzip+encryption)

- {“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.

# 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`

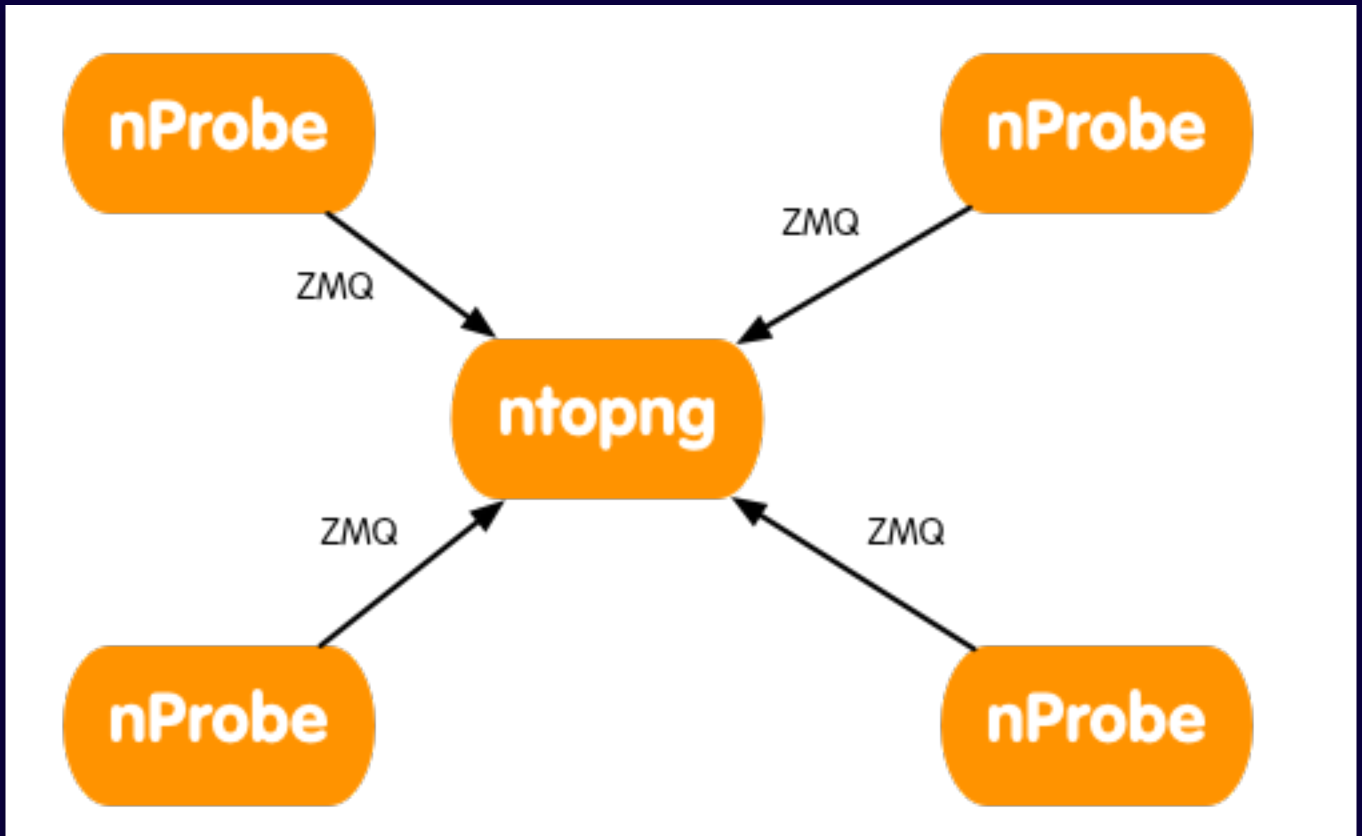
# Flow Collection: Pull vs Poll Mode

- Poll Mode
  - host X> ntopng -i "tcp://Y:1234" --zmq-encrypt-pwd myencryptionkey
  - host Y> nprobe -n none --zmq "tcp://*:1234" --zmq-encrypt-pwd myencryptionkey

- Push Mode
  - host X> ntopng -i "tcp://Y:1234" --zmq-collector-mode --zmq-encrypt-pwd myencryptionkey
  - host Y> nprobe -n none --zmq "tcp://*:1234" --zmq-probe-mode --zmq-encrypt-pwd myencryptionkey

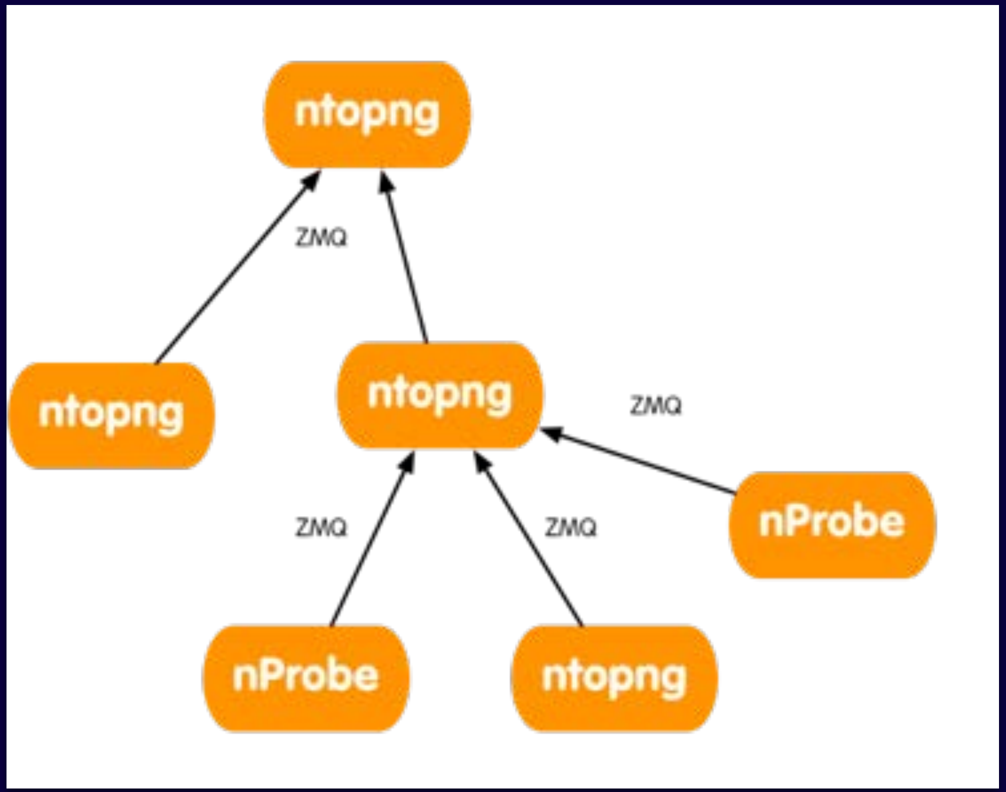
# Creating ntopng Clusters [1/3]

- 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:



# 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.



# 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`

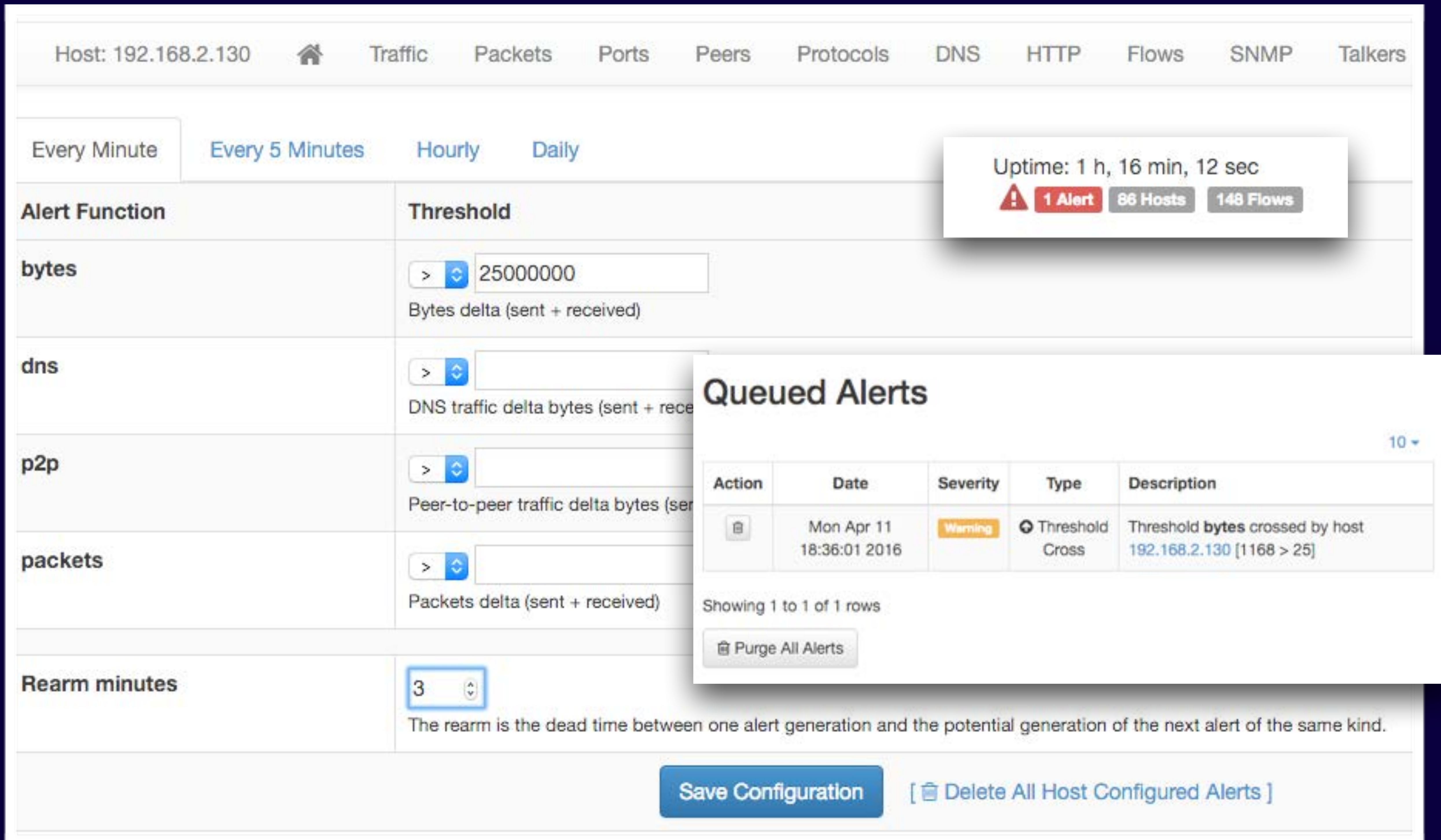
- If you want to merge all nProbe traffic into a single ntopng interface do:
  - `ntopng -i tcp://host1:5556,tcp://host2:5556,tcp://host3:5556,tcp://host4:5556`

- 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`

# Managing Alerts [1/2]

- In many situations it is fundamental to set alerts that can signal anomalous conditions
- Ntopng handles host/interface/network alerts hooked to multiple metrics
- Metrics include bytes/packets received/generated
- User-submitted alerts are continuously monitored in the background

# Managing Alerts [2/2]



# Sending ntopng Alerts to Nagios [1/2]

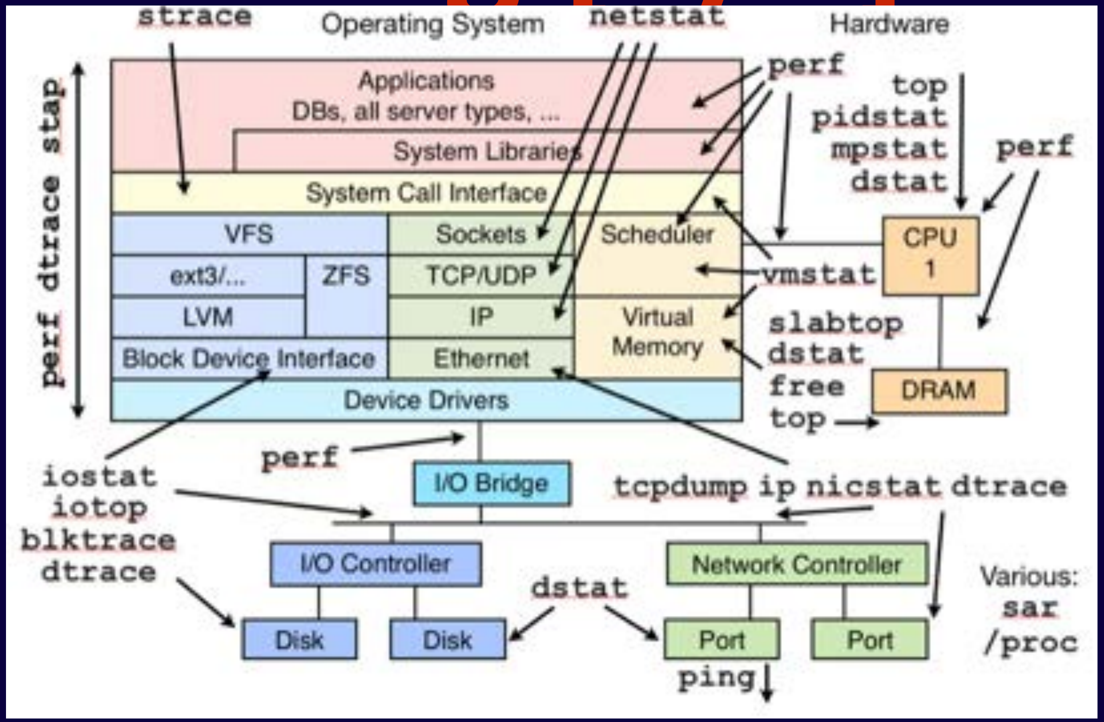
- Nagios is the de-facto standard in infrastructure monitoring
- Ntopng features alert propagation to Nagios

# Sending ntopng Alerts to Nagios [2/2]

- Alerts are sent to Nagios via NSCA
- Nagios will intercept all alerts that are explicitly declared as passive services
- Passive service description format is:
  - NtopngAlert_<host/network/interface>_<timespan>_<metric>

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ntopng-host | NtopngAlert | ? | OK | 12-23-2015 15:25:50 | 0d 17h 27m 59s | 1/1 | Alert for host Y! |
| | NtopngAlert_192.168.1.15_min_bytes | ? | OK | 12-23-2015 09:13:22 | 0d 6h 47m 34s | 1/1 | OK, alarm deactivated |
| | NtopngAlert_192.168.2.0/24 | ? | OK | 12-23-2015 11:02:34 | 0d 4h 33m 4s | 1/1 | OK, alarm deactivated |
| | NtopngAlert_192.168.70.0/24_min_egress | ? | WARNING | 12-23-2015 15:33:01 | 0d 0h 6m 5s | 1/1 | Threshold egress crossed by network 192.168.70.0/24 [1180 > 10] |
| | NtopngAlert_192.168.70.0/24_min_ingress | ? | WARNING | 12-23-2015 15:33:01 | 0d 0h 2m 5s | 1/1 | Threshold ingress crossed by network 192.168.70.0/24 [11241211 > 10] |

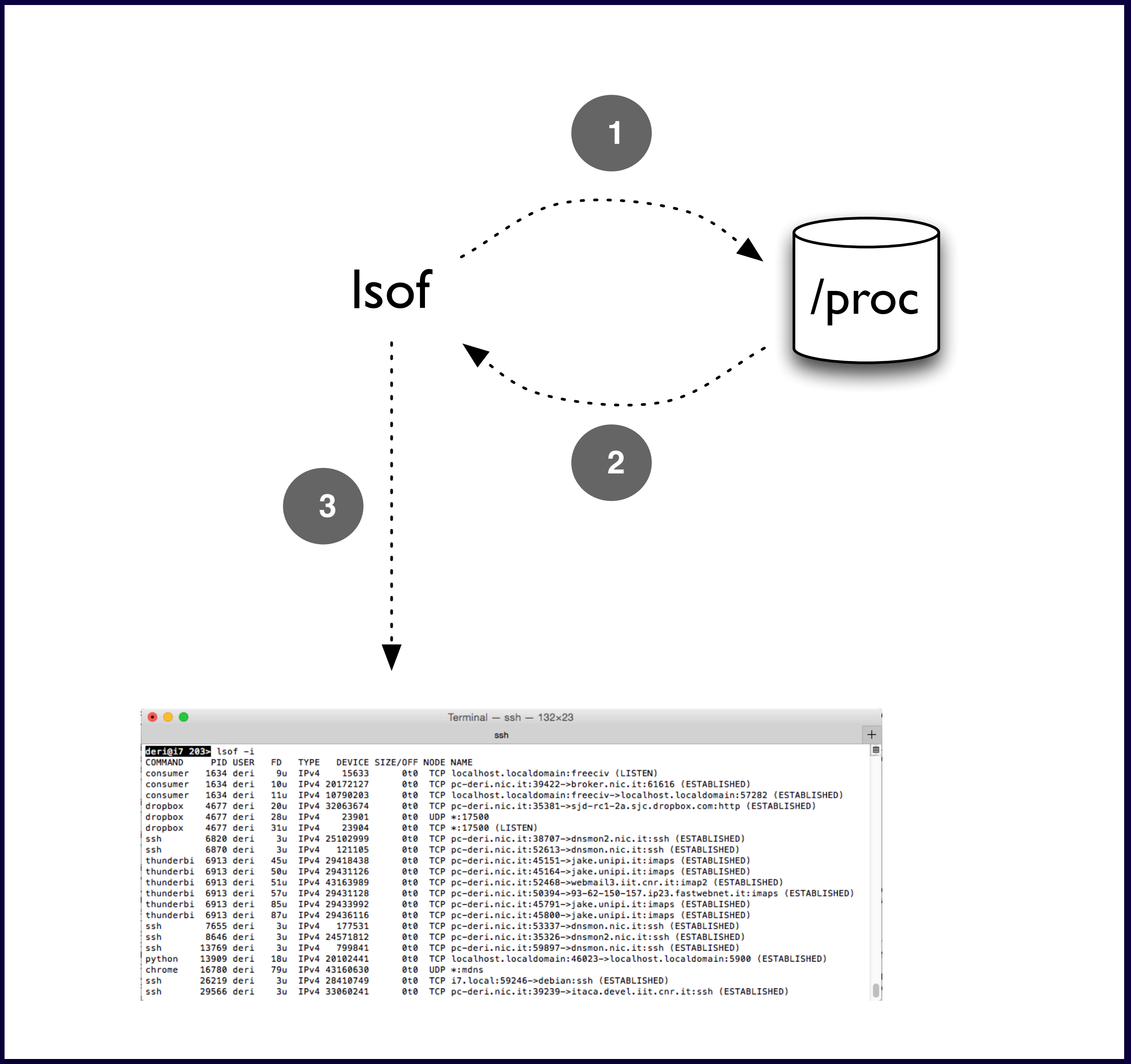
# System+Network Monitoring [1/3]




- 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.

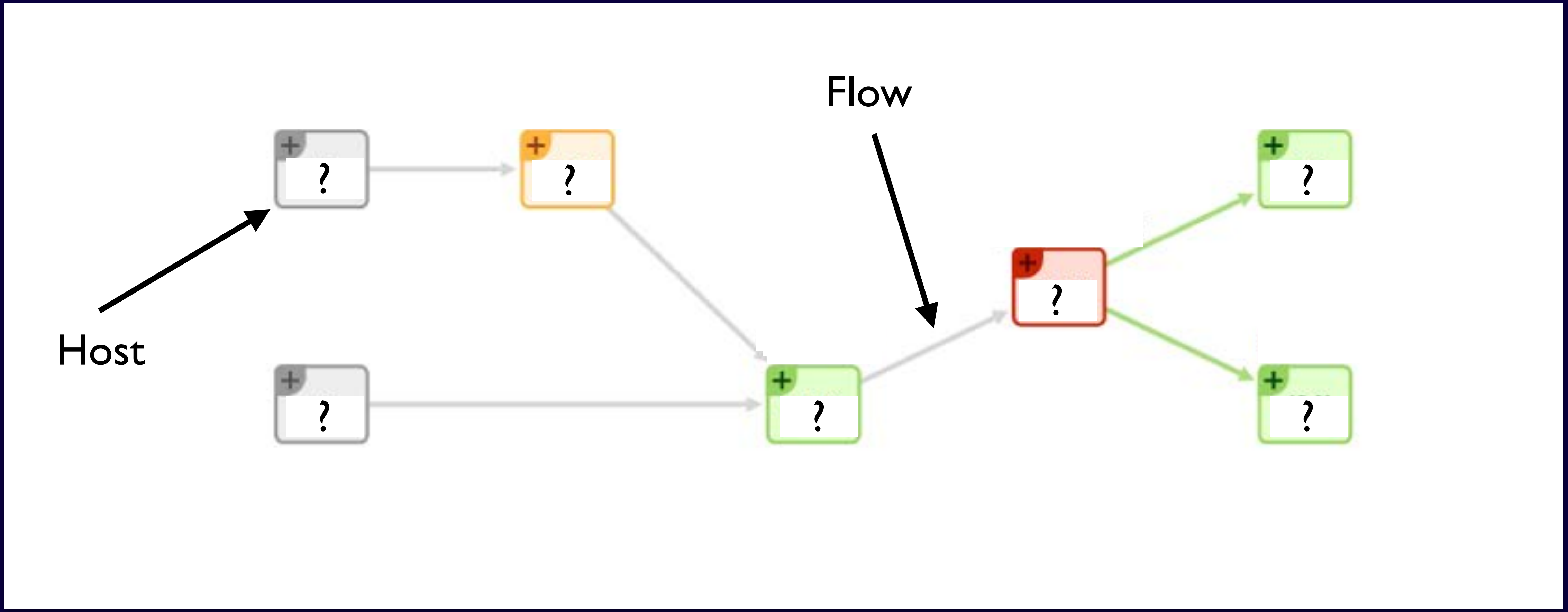
# System+Network Monitoring [2/3]

How most system management tools work on Linux:



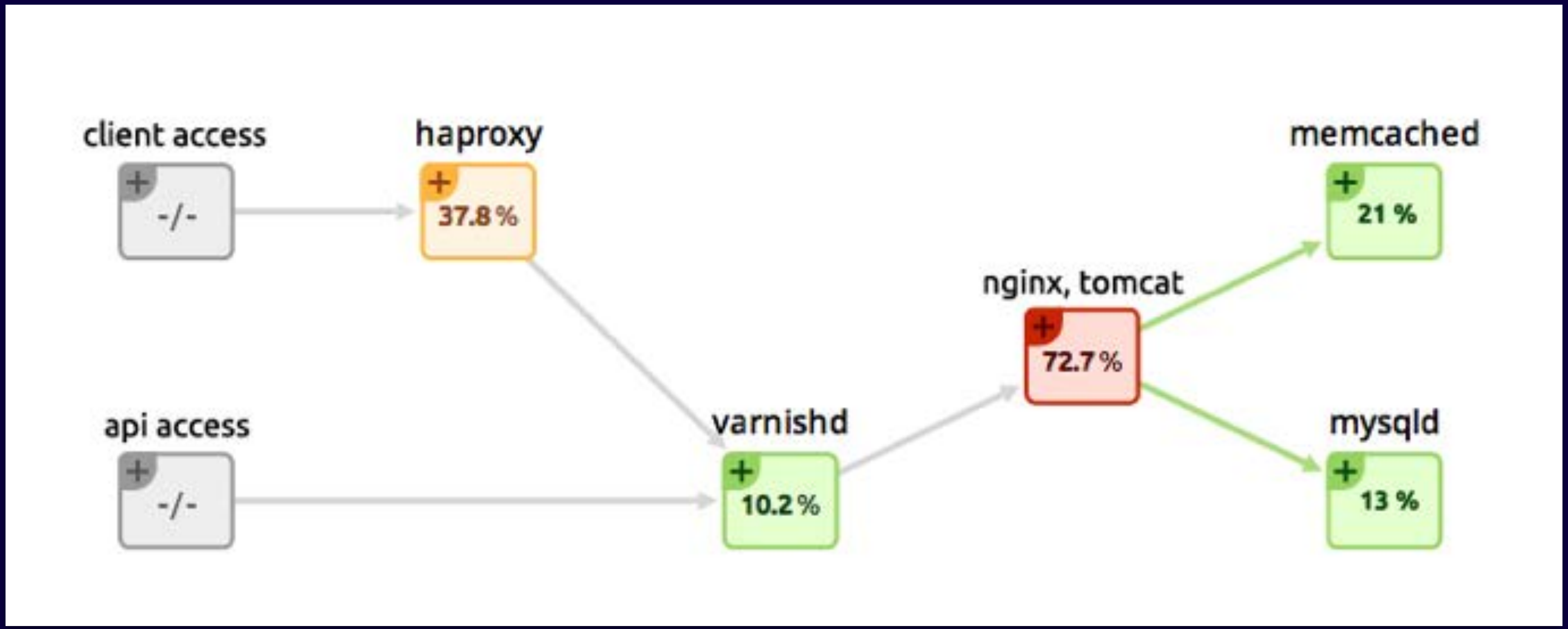
# System+Network Monitoring [3/3]

- 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.



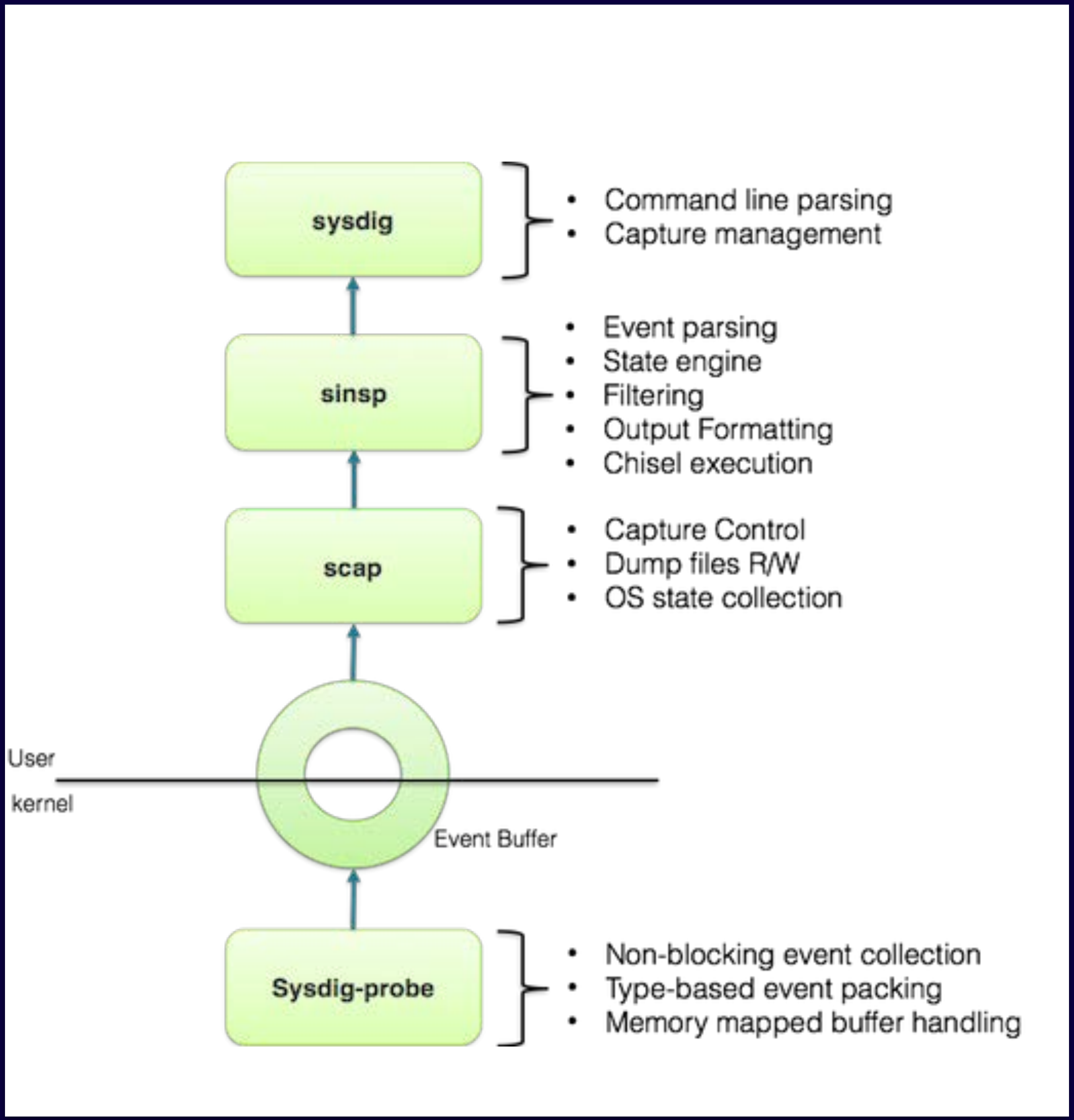
# System+Network Monitoring [3/3]

- 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:



# 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.



# 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.

# Integrating sysdig in nProbe [1/2]

- 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.

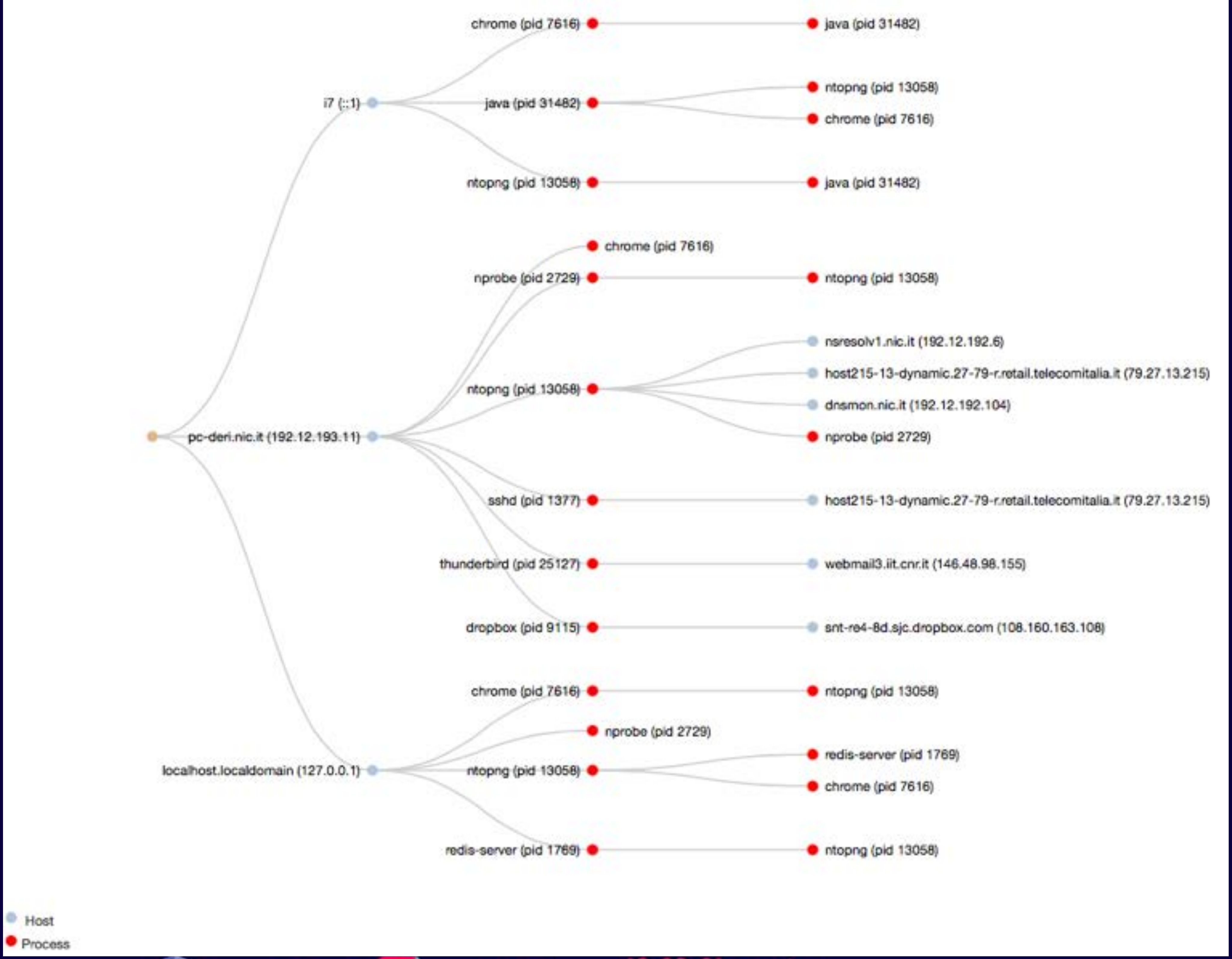
# Integrating sysdig in nProbe [2/2]

- 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.

# Ntopng+nProbe+sysdig

- 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.

# Process Network Communications



# Flow/Process Drill-down [1/2]

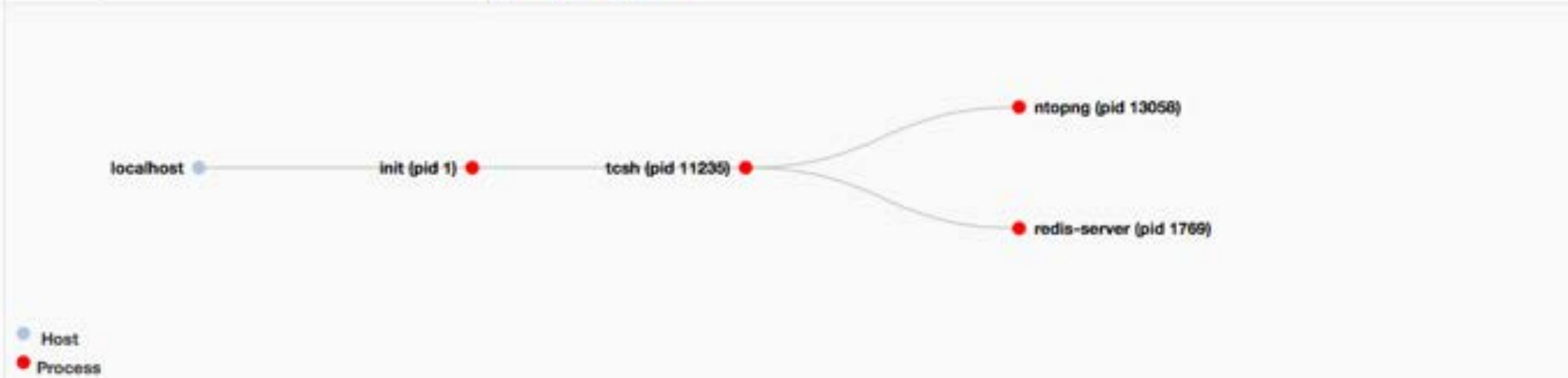
## Active Flows

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. | |



# 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 |
| **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

Flow-to-Process binding

Dynamically Updated

# 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

# Process Network Traffic

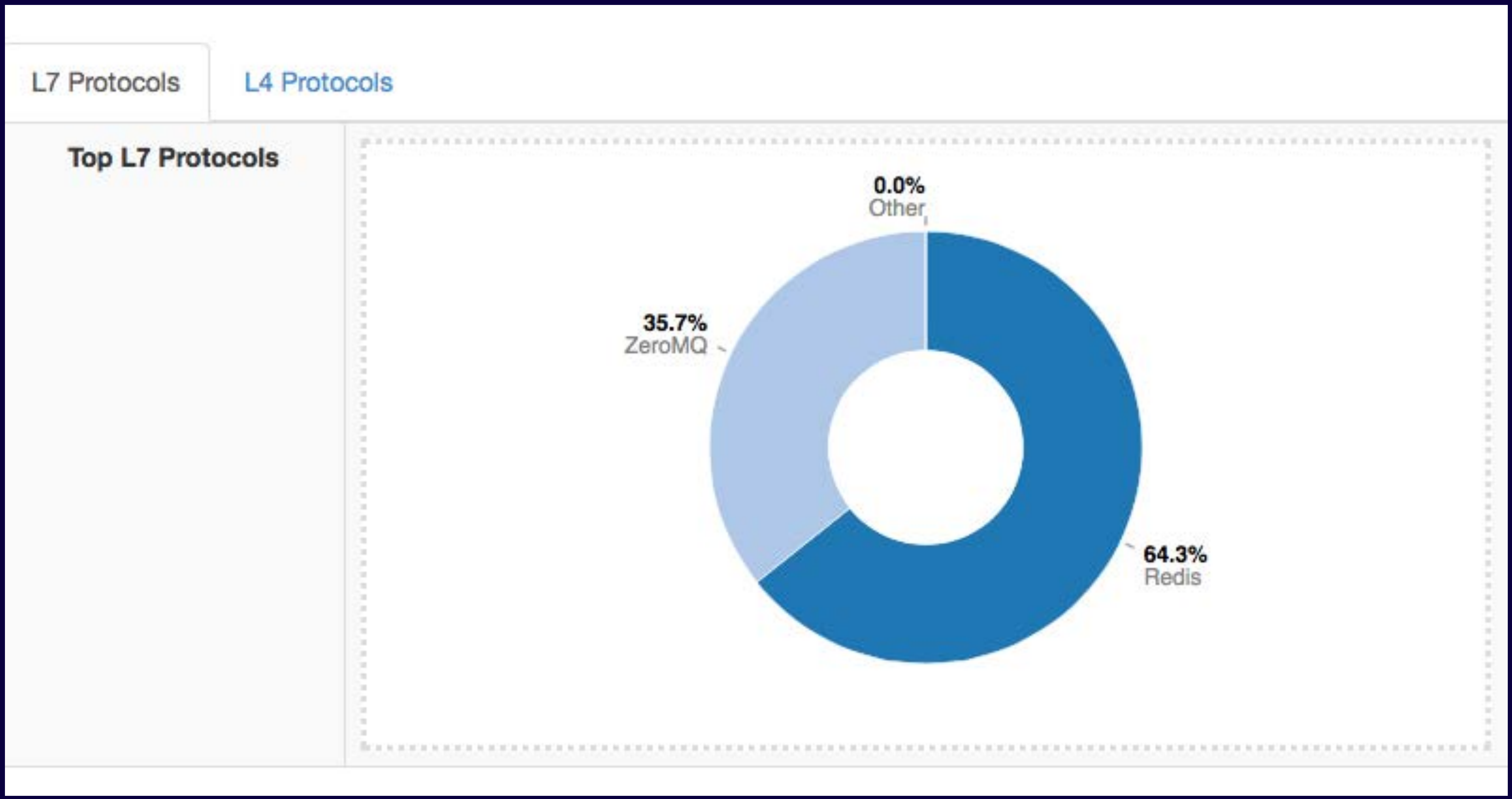
## Active Processes

10 ▾

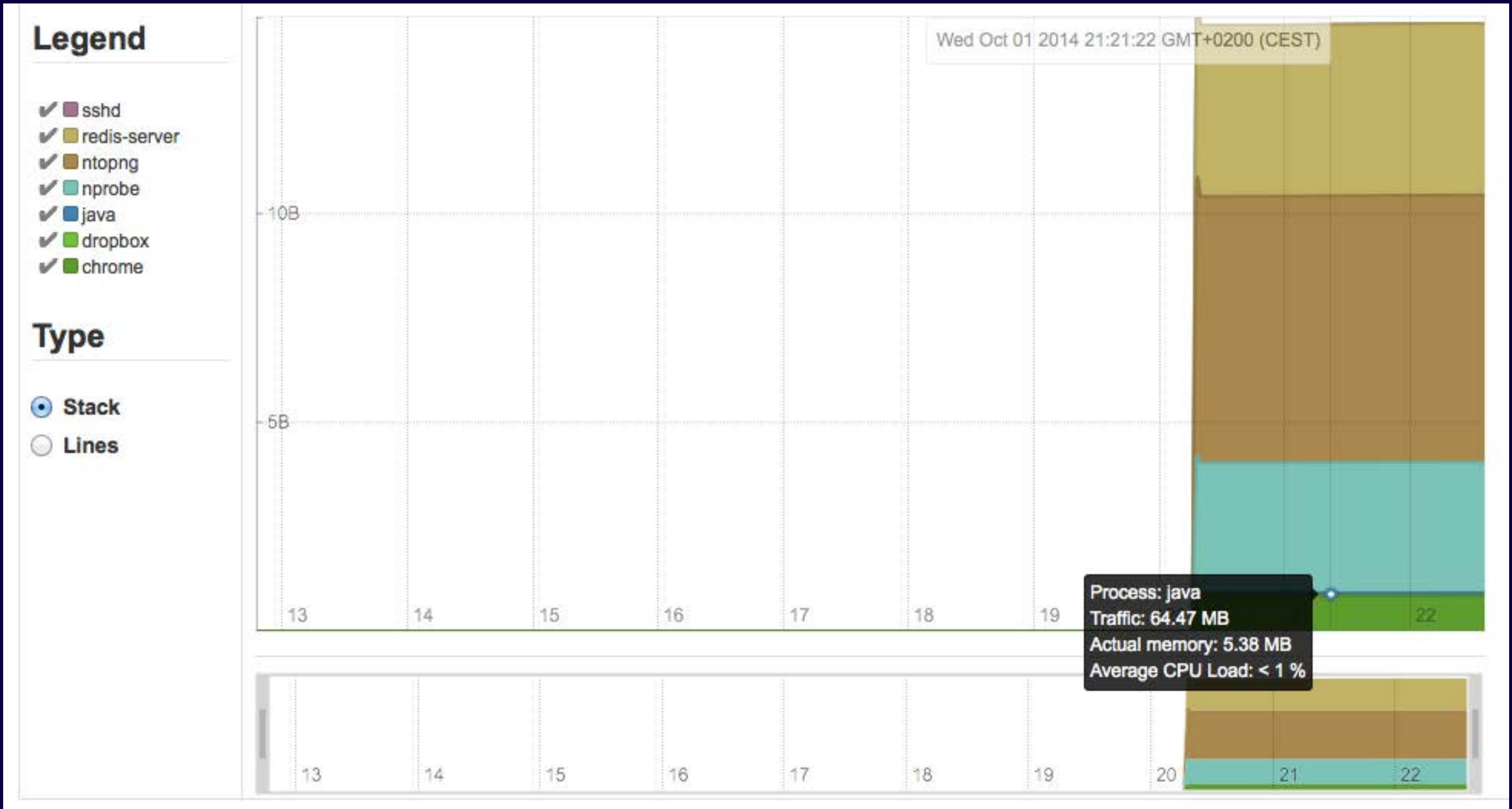
| Name | Flows Count | Active Since | Traffic Sent | Traffic Rcvd |
|---|---|---|---|---|
| sshd | 1 — | 19 min, 42 sec | 39.99 KB — | 20.87 KB — |
| redis-server | 1 — | 1 day, 6 h, 19 min, 50 sec | 1.96 GB ↑ | 1.86 GB ↑ |
| ntopng | 40 ↑ | 1 day, 6 h, 19 min, 50 sec | 2.05 GB ↑ | 3.89 GB ↑ |
| nprobe | 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

# Process Protocols Drill-Down



# Processes Timeline



# User Flows and Processes

elasticsearch - i7 | Applications | Protocols | Flow

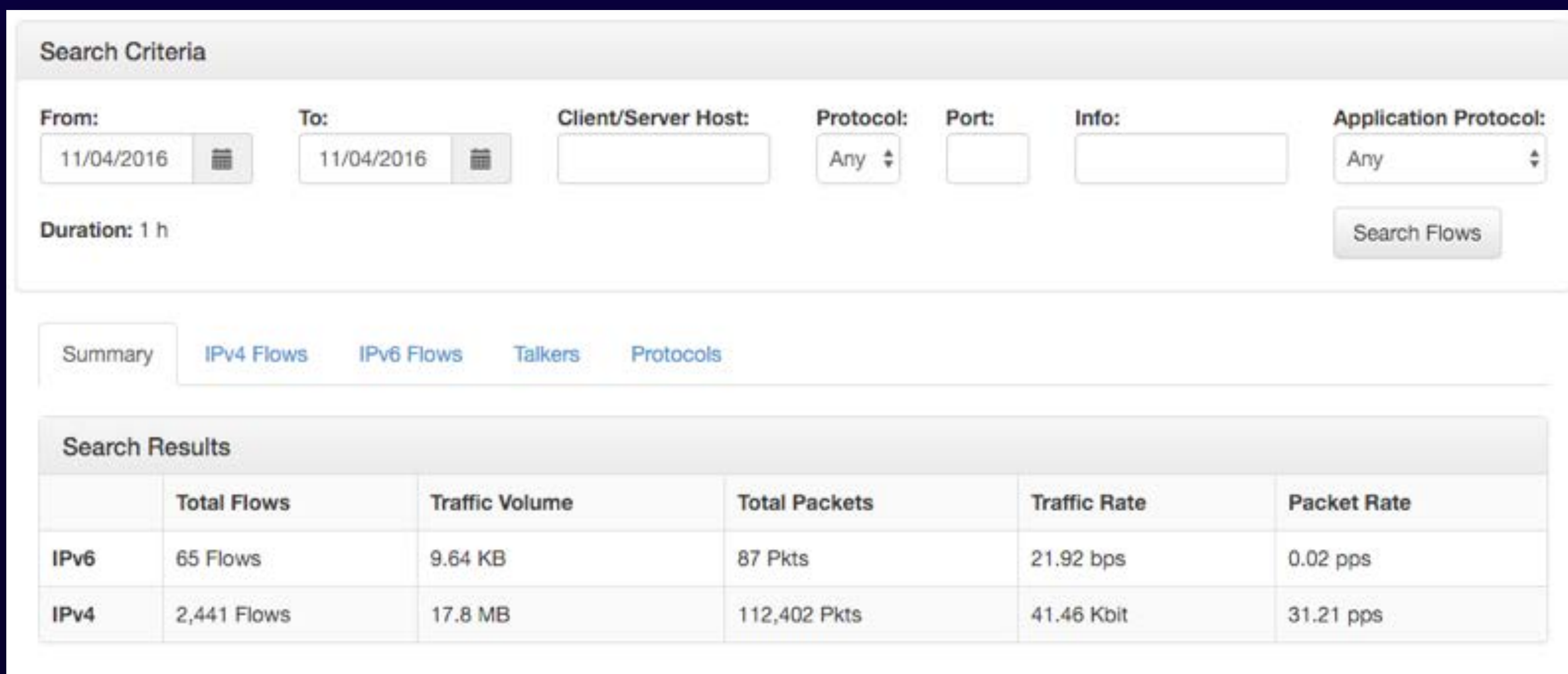
## Active Flows

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 S | 8.13 KB |
| Info | HTTP | TCP | ntopng | i7 :33418 | java | i7 :9200 | 1 sec | Client S | 6.72 KB |
| Info | HTTP | TCP | ntopng | i7 :33417 | java | i7 :9200 | 1 sec | Client S | 6.71 KB |

# Historical Flow Navigation [1/2]

- Ntopng can send (-F) network flows to MySQL
  - A built-in database explorer retrieves such flows and allows them to be navigated and searched



# Historical Flow Navigation [2/2]

Summary | IPv4 Flows | IPv6 Flows | Talkers | Protocols

## IPv6 Top Flows [11/04/2016 17:56:35 - 11/04/2016 18:56:35]

5

| | Application | L4 Proto | Client | Server | Begin | End | Bytes | Avg Thpt |
|---|---|---|---|---|---|---|---|---|
| Info | ? Unknown | UDP | simones-macbook-pro.loca...:mdns | ff02::fb:mdns | 11/04/2016 18:22:02 | 11/04/2016 18:22:03 | 811 B | 3.24 Kbit |
| Info | ? Unknown | UDP | simones-macbook-pro.loca...:mdns | ff02::fb:mdns | 11/04/2016 18:22:02 | 11/04/2016 18:22:03 | 811 B | 3.24 Kbit |
| Info | ? Unknown | UDP | fe80::3e15:c2ff:feb7:720...:mdns | ff02::fb:mdns | 11/04/2016 18:39:30 | 11/04/2016 18:39:30 | 613 B | 4.9 Kbit |
| Info | ? Unknown | UDP | fe80::b675:eff:fe92:8917...:dhcpv6-client | ff02::1:2:dhcpv6-server | 11/04/2016 18:50:40 | 11/04/2016 18:50:43 | 324 B | 648 bps |
| Info | ? Unknown | UDP | fe80::b675:eff:fe92:8917...:dhcpv6-client | ff02::1:2:dhcpv6-server | 11/04/2016 18:41:55 | 11/04/2016 18:41:58 | 324 B | 648 bps |

Showing 1 to 5 of 65 rows

« < 1 2 3 4 5 > »

Download flows: IPv4 IPv6   Extract pcap:

Bulk download and full pcap extraction options

# Historical Talkers [1/2]

- Top Talkers can be automatically extracted from flows.
- Every top talker can be clicked to inspect its peers.
- Every peer can be clicked to inspect L7 application protocols.

# Historical Talkers [2/2]

Summary | IPv4 Flows | IPv6 Flows | Talkers | Protocols

Interface en4

50

| Host Name | IP Address | Total Traffic | Total Packets | Ingress Traffic | Ingress Packets | Egress Traffic | Egress Packets | Flows |
|---|---|---|---|---|---|---|---|---|
| 192.168.2.130 | 192.168.2.130 | 18.27 MB | 119,364 | 9.02 MB | 86,911 | 9.25 MB | 32,453 | 2,320 |

Summary | IPv4 Flows | IPv6 Flows | Talkers | Protocols

Interface en4 / Talkers with 172.217.16.5

50

| Host Name | IP Address | Total Traffic | Total Packets | Traffic Sent | Packets Sent | Traffic Received | Packets Received | Flows |
|---|---|---|---|---|---|---|---|---|
| 192.168.2.130 | 192.168.2.130 | 1.68 MB | 3,317 | 0 B | 0 | 1.68 MB | 3,317 | 12 |

Summary | IPv4 Flows | IPv6 Flows | Talkers | Protocols

Interface en4 / Talkers with 172.217.16.5 / Applications between 172.217.16.5 and 192.168.2.130

50

| Application | Traffic Volume | Packets | Flows |
|---|---|---|---|
| Quic | 1.68 MB | 3,317 | 12 |

# Historical Applications [1/2]

- Top Applications can be automatically extracted from flows as well.
- Every top application can be clicked to inspect hosts that have used it.
- Every host can be clicked to inspect peers that have used a given application to communicate with the host.

# Historical Applications [2/2]

Chart | IPv4 Flows | IPv6 Flows | Talkers | Protocols

Interface en4

protocols — all — Select saved...

host peers by protocol — all — Select saved...

50

| Application | Traffic Volume | Packets | Flows |
|---|---|---|---|
| AppleiTunes | 471 B | 2 | 1 |
| IGMP | 600 B | 10 | 10 |
| NTP | 1.05 KB | 12 | 6 |

Chart | IPv4 Flows | IPv6 Flows | Talkers | Protocols

Interface en4 / AppleiTunes talkers

protocols — all — Select saved...

host peers by protocol — all — Select saved...

50

| Host Name | Address | Traffic Volume | Packets | Flows |
|---|---|---|---|---|
| 192.168.2.130 | 192.168.2.130 | 471 B | 2 | 1 |

Chart | IPv4 Flows | IPv6 Flows | Talkers | Protocols

Interface en4 / AppleiTunes talkers / AppleiTunes talkers with 192.168.2.130

protocols — all — Select saved...

host peers by protocol — all — Select saved...

50

| Host Name | Address | Traffic Volume | Packets | Flows |
|---|---|---|---|---|
| jake.unipi.it | 131.114.18.19 | 471 B | 2 | 1 |

# 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.

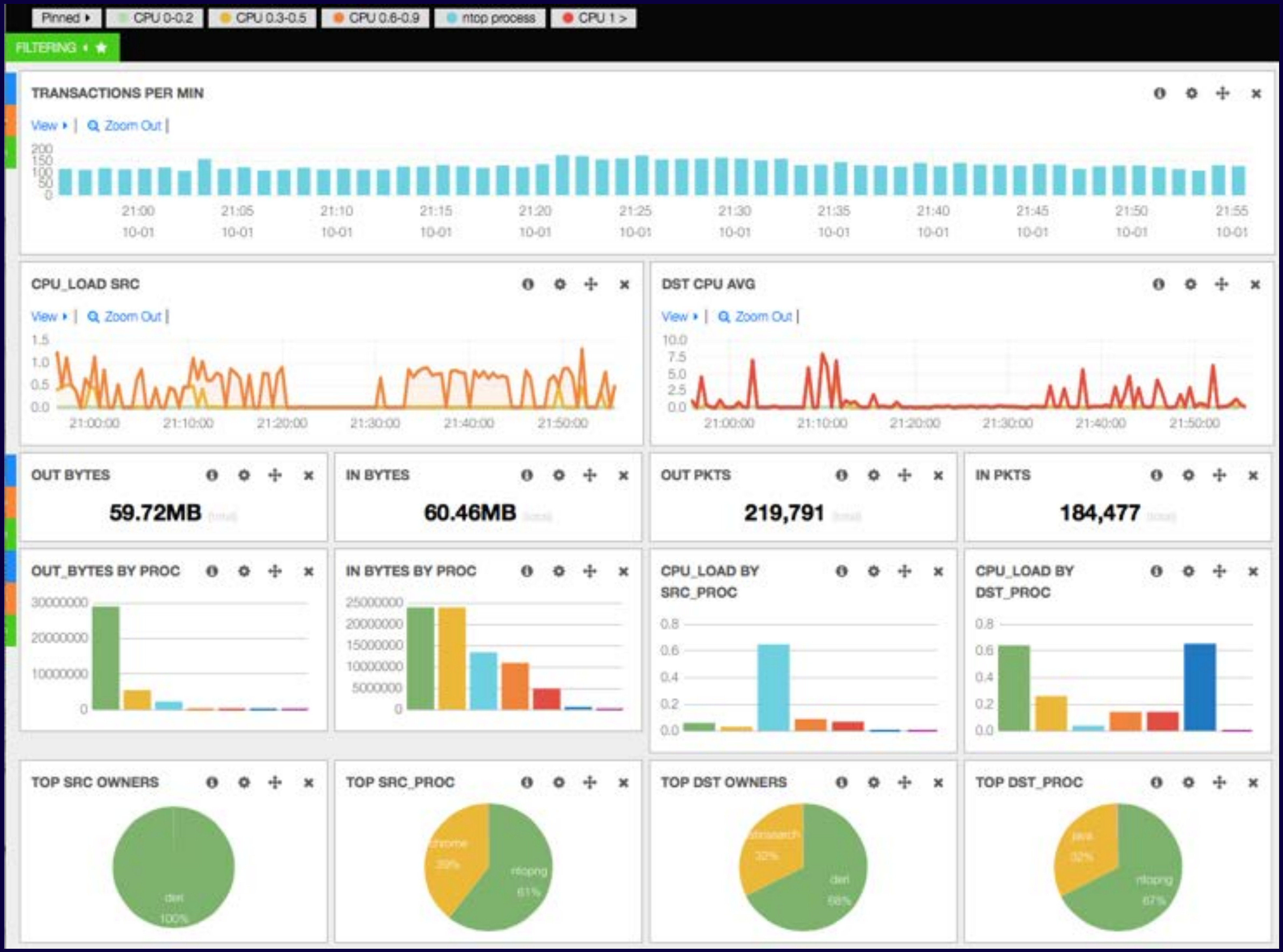
# Integrating Ntopng with ElasticSearch [1/2]

- 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.

# 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.

# Ntopng Process Dashboard in Kibana [1/2]

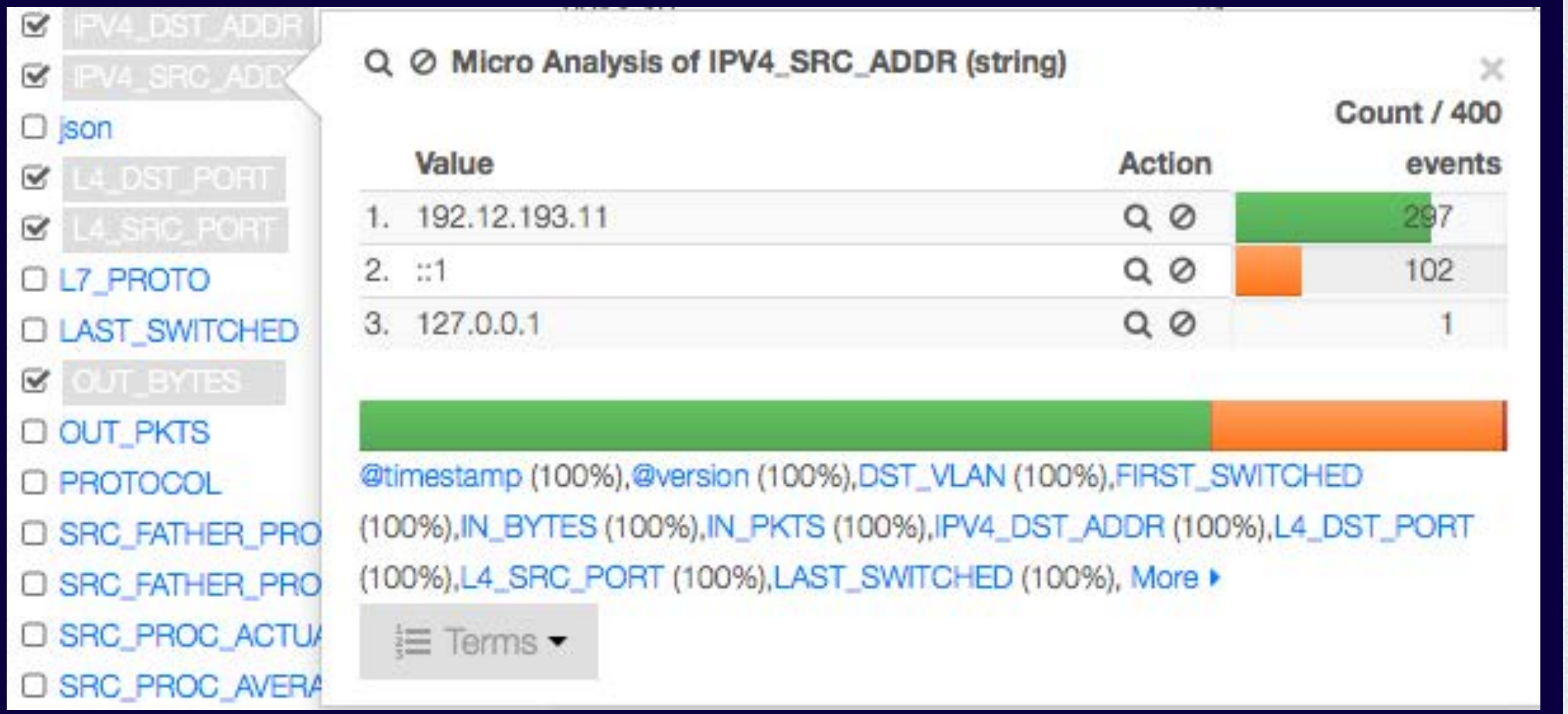


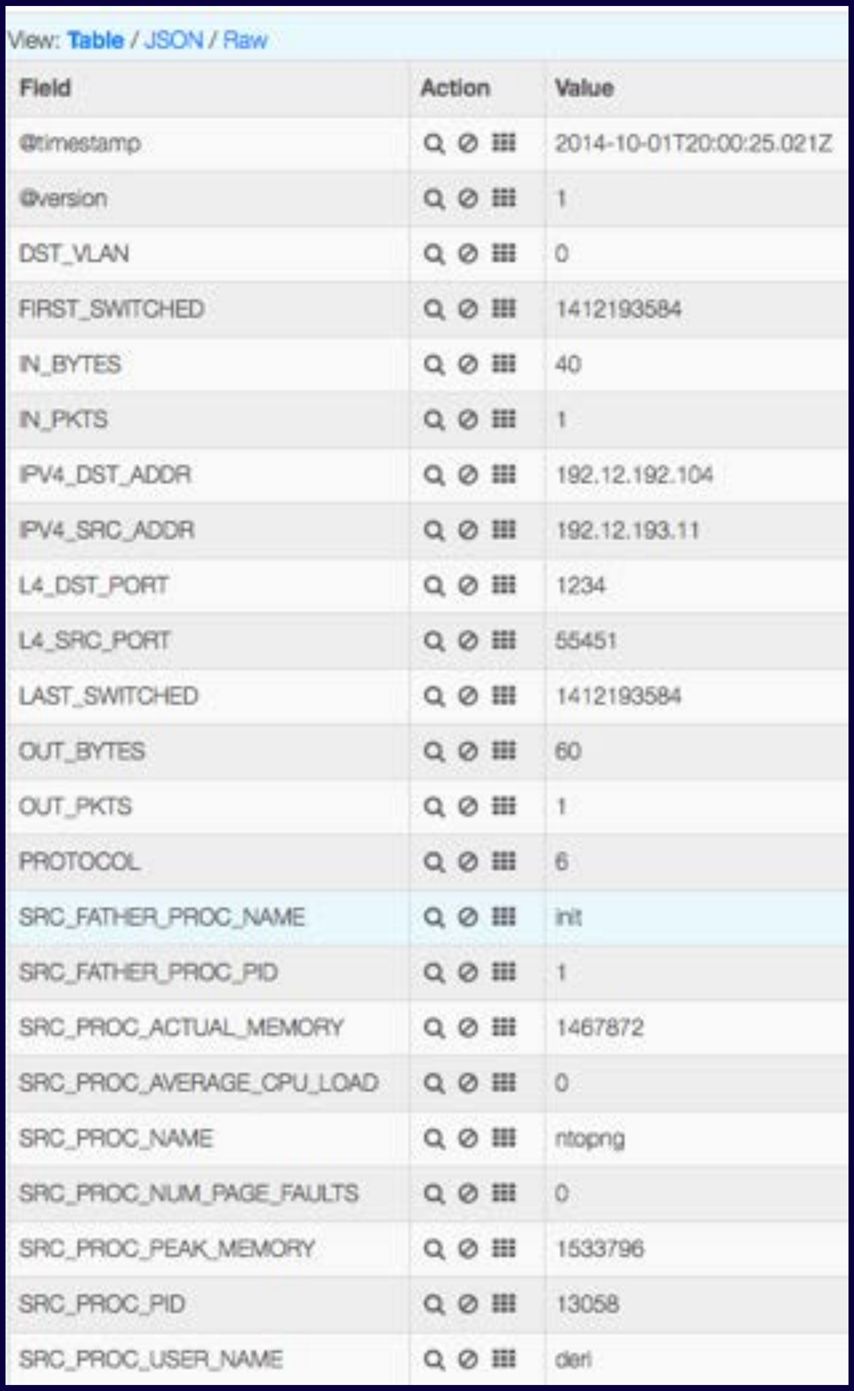

19-20-21 agosto 2016

Parco Ex Caserma Cocco
Pescara

# 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.



# 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.

# 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.

docker

openstack™ CLOUD SOFTWARE

# 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.


# Embedding Ntopng [2/4]

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

BeagleBoard Black

# 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?

# 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).

# Final Remarks

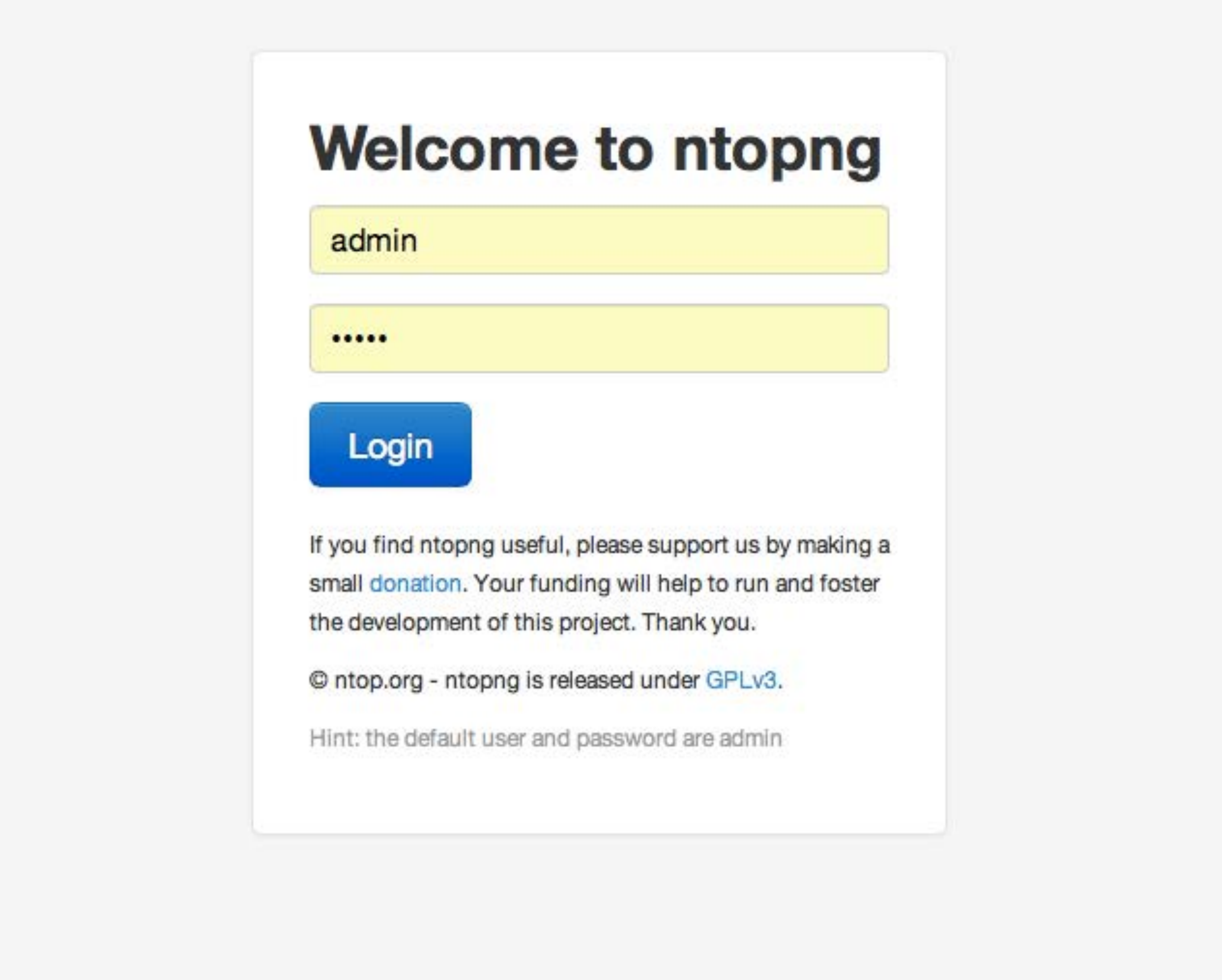
- Over the past 18 years ntop created a software framework for efficiently monitoring traffic.
- “We have a story to tell you, not just hacks”.
- Commodity hardware, with adequate software, can now match the performance and flexibility that markets require. With the freedom of open source.
- Ntopng is available under GNU GPLv3 from http://www.ntop.org/.

PERFECTLY IMPERFECT

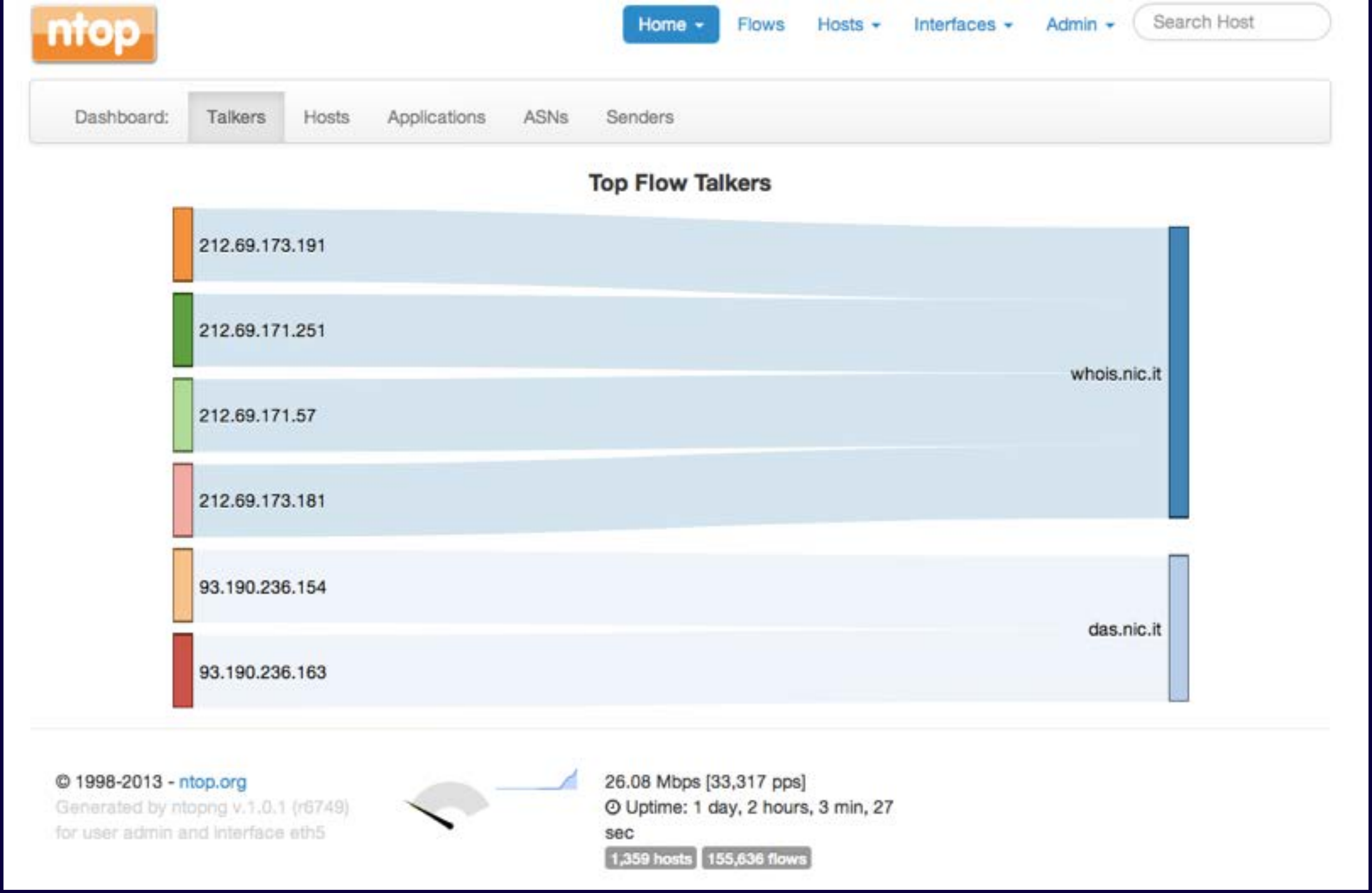
19-20-21 agosto
2016
Parco Ex Caserma Cocco
Pescara

# Using Ntopng

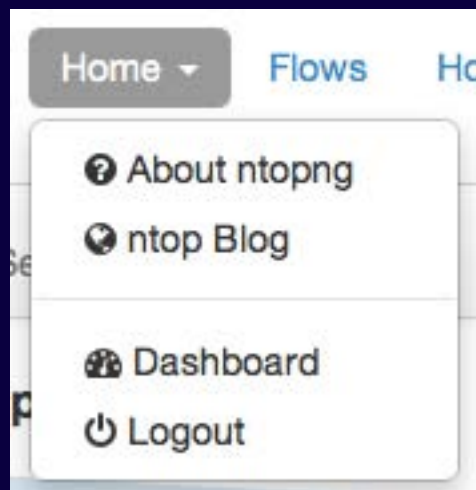
# Logging into ntopng

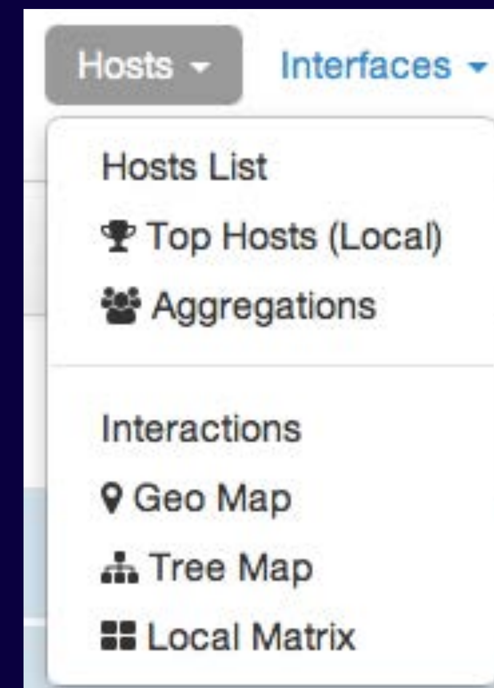


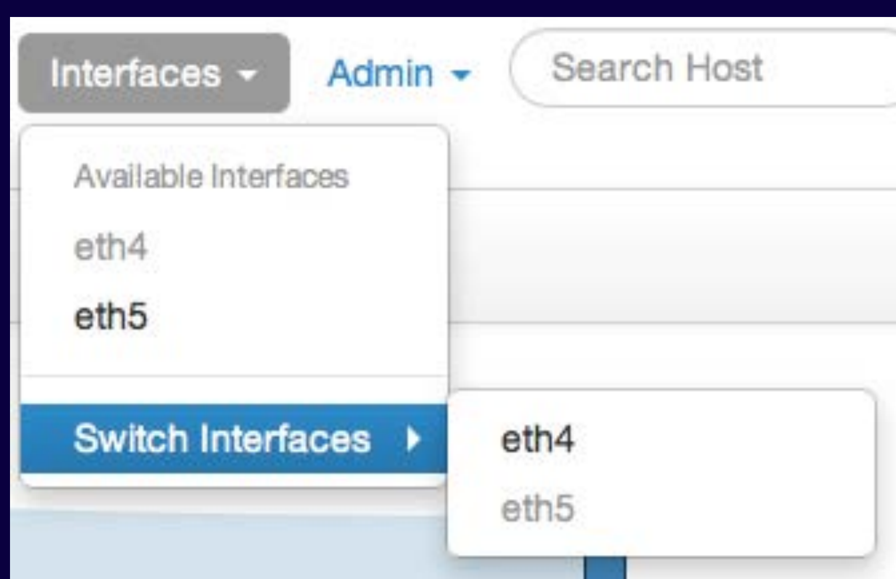
# ntopng Dashboard

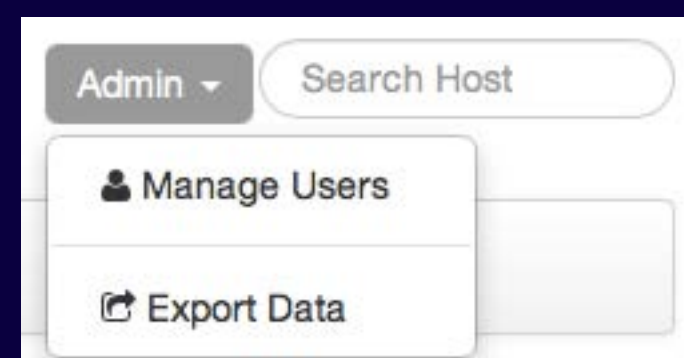



# Available Menu Items

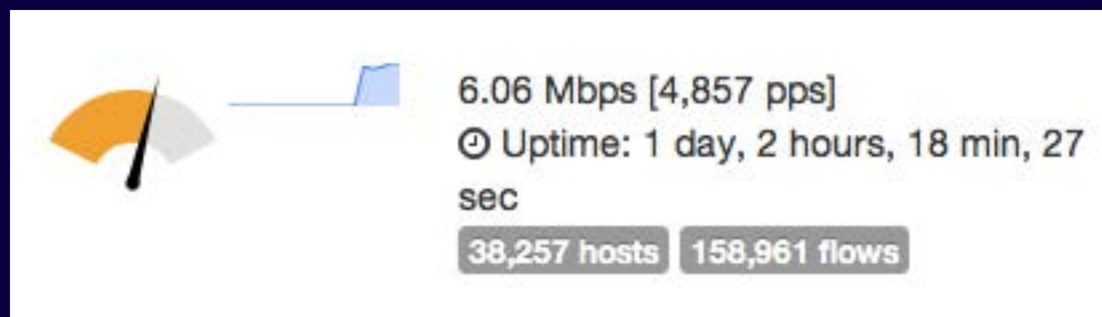


# Dynamic Web Interface




Applications▾

- DHCP
- DHCPV6
- HTTP
- ICMP
- ICMPV6
- IGMP
- MDNS
- OSPF
- Unknown
- VRRP
- Whois-DAS

| Throughput | Total Bytes |
|---|---|
| 8.09 Kbit ↓ | 94.23 MB |
| 5.59 Kbit ↑ | 60.15 MB |
| 5.16 Kbit ↓ | 60.15 MB |

# Flows Monitoring [1/2]

## Active Flows

⚙ 10 ▾ ⤢ Applications▾

| Info | Application | L4 Proto | Client | Server | Duration | Breakdown | Throughpu | |
|---|---|---|---|---|---|---|---|---|
| Info | VRRP | VRRP | fe80::192:12:192:7 | ff02::12 | 1 day, 2 hours, 4 min, 19 sec | Client | 8.09 Kbit ↓ | |
| Info | VRRP | VRRP | 192.12.192.7 | 224.0.0.18 | 1 day, 2 hours, 4 min, 19 sec | Client | 5.59 Kbit ↑ | |
| Info | VRRP | VRRP | 192.168.18.7 | 224.0.0.18 | 1 day, 2 hours, 4 min, 19 sec | Client | 5.16 Kbit ↓ | |
| Info | DHCP | UDP | 0.0.0.0:68 | 255.255.255.255:67 | 1 day, 2 hours, 3 min, 57 sec | Client | 0 bps — | |
| Info | OSPF | 89 | 192.12.192.7 | 224.0.0.5 | 1 day, 2 hours, 4 min, 13 sec | Client | 0 bps ↓ | |
| Info | OSPF | 89 | 192.168.18.7 | 224.0.0.5 | 1 day, 2 hours, 4 min, 7 sec | Client | 0 bps — | |
| Info | OSPF | 89 | 192.168.18.9 | 224.0.0.5 | 1 day, 2 hours, 4 min, 14 sec | Client | 359.83 bps ↑ | |
| Info | OSPF | 89 | 192.12.192.9 | 224.0.0.5 | 1 day, 2 hours, 4 min, 16 sec | Client | 359.83 bps ↑ | |
| Info | OSPF | 89 | 192.168.18.34 | 224.0.0.5 | 1 day, 2 hours, 4 min, 7 sec | Client | 0 bps ↓ | 1 MB |
| Info | OSPF | 89 | 192.12.192.34 | 224.0.0.5 | 1 day, 2 hours, 4 min, 7 sec | Client | 0 bps — | 1 MB |

- DHCP
- DHCPV6
- HTTP
- ICMP
- ICMPV6
- IGMP
- MDNS
- OSPF
- Unknown
- VRRP
- Whois-DAS

Showing 1 to 10 of 151325 rows

← First | Prev | 1 | 2 | 3 | 4 | 5 | Next | Last →

# Flows Monitoring [2/2]

Flow: 192.12.192.237:53060 ⇄ whois.nic.it:5043 | Overview

| | |
|---|---|
| Client | web-r1.nic.it:53060 |
| Server | whois.nic.it:5043 |
| Application Protocol | HTTP |
| First Seen | 11/10/2013 13:45:26 [6 min, 54 sec ago] |
| Last Seen | 11/10/2013 13:52:12 [8 sec ago] |
| Total Traffic Volume | 7.03 KB — |
| Client vs Server Traffic Breakdown | 192.12.192.237 |
| Client to Server Traffic | 63 Pkts / 7.03 KB — |
| Server to Client Traffic | 0 Pkts / 0 Bytes — |
| Actual Throughput | 0 bps — |
| TCP Flags | SYN PUSH ACK |



193.98 Kbps [260 pps]
Uptime: 1 day, 2 hours, 4 min, 49 sec
1,272 hosts 153,747 flows

# Host Monitoring [1/3]

## Hosts List

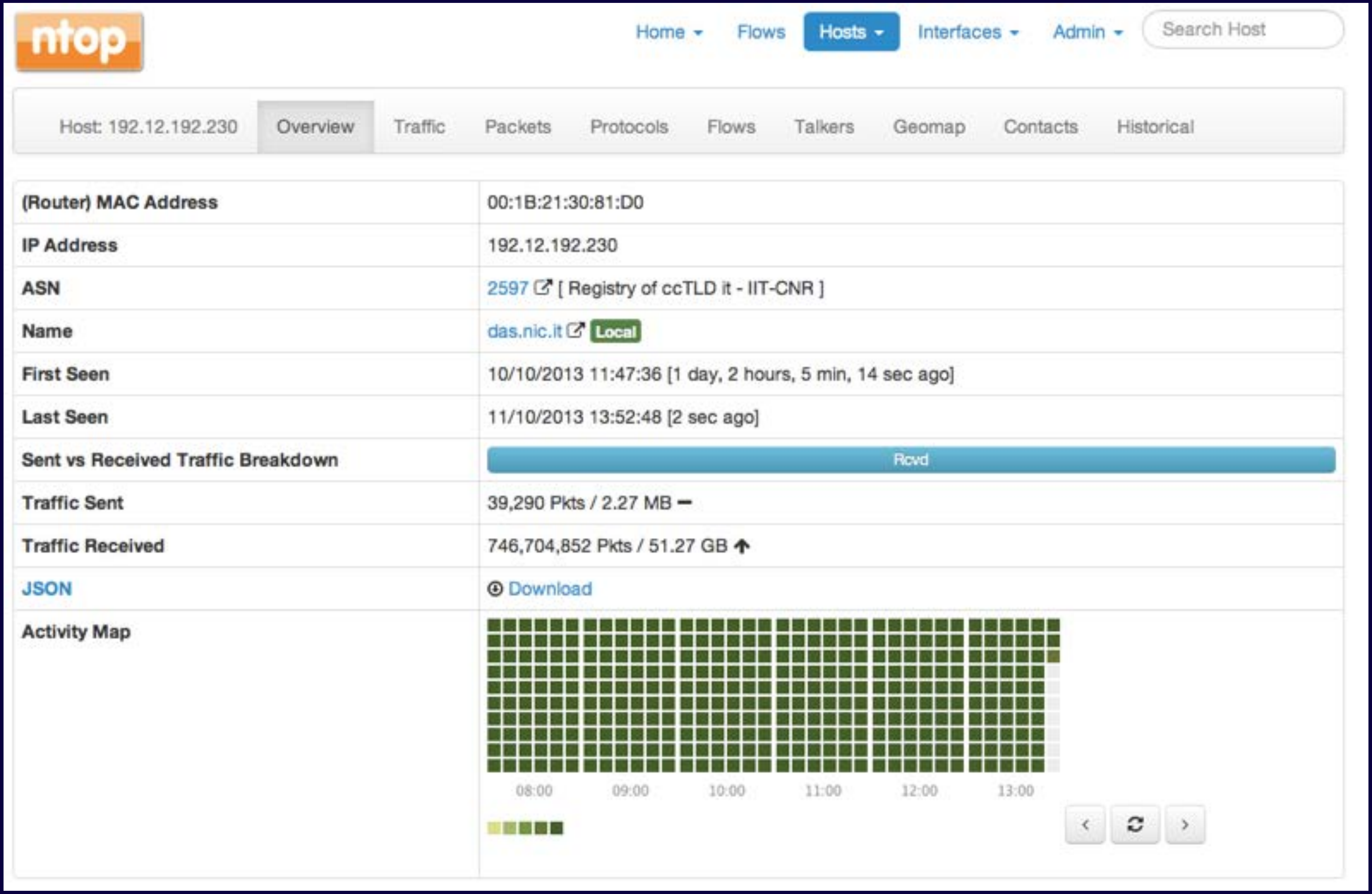
⚙ 10 ▾

| IP Address | Location | Symbolic Name | Seen Since | ASN | Breakdown | Throughput | Traffic |
|---|---|---|---|---|---|---|---|
| 192.12.192.230 | Local | das.nic.it | 1 day, 2 hours, 4 min, 49 sec | 2597 | Rcvd | 13.57 Kbit | 51.27 GB |
| 192.165.67.192 | Remote | 192.165.67.192 | 1 day, 2 hours, 4 min, 31 sec | 34971 | Sent | 0 bps | 9.62 GB |
| 192.165.67.166 | Remote | 192.165.67.166 | 1 day, 2 hours, 4 min, 31 sec | 34971 | Sent | 659.95 bps | 9.18 GB |
| 78.46.216.98 | Remote | 78.46.216.98 | 1 day, 2 hours, 4 min, 48 sec | 24940 | Sent | 219.98 bps | 7.87 GB |
| 192.165.67.22 | Remote | 192.165.67.22 | 1 day, 2 hours, 4 min, 30 sec | 34971 | Sent | 0 bps | 7.81 GB |
| 78.47.50.132 | Remote | 78.47.50.132 | 1 day, 2 hours, 4 min, 48 sec | 24940 | Sent | 879.93 bps | 7.18 GB |
| 62.149.189.11 | Remote | 62.149.189.11 | 1 day, 2 hours, 4 min, 35 sec | 31034 | Sent | 0 bps | 1.44 GB |
| 192.12.192.242 | Local | whois.nic.it | 1 day, 2 hours, 4 min, 49 sec | 2597 | Rcvd | 84.86 Kbit | 964.02 MB |
| 224.0.0.18 | Remote | vrrp.mcast.net | 1 day, 2 hours, 4 min, 49 sec | | Rcvd | 8.81 Kbit | 120.35 MB |
| 213.154.243.80 | Remote | 213.154.243.80 | 18 hours, 57 min, 57 sec | 12859 | Sent | 4.51 Kbit | 116.72 MB |

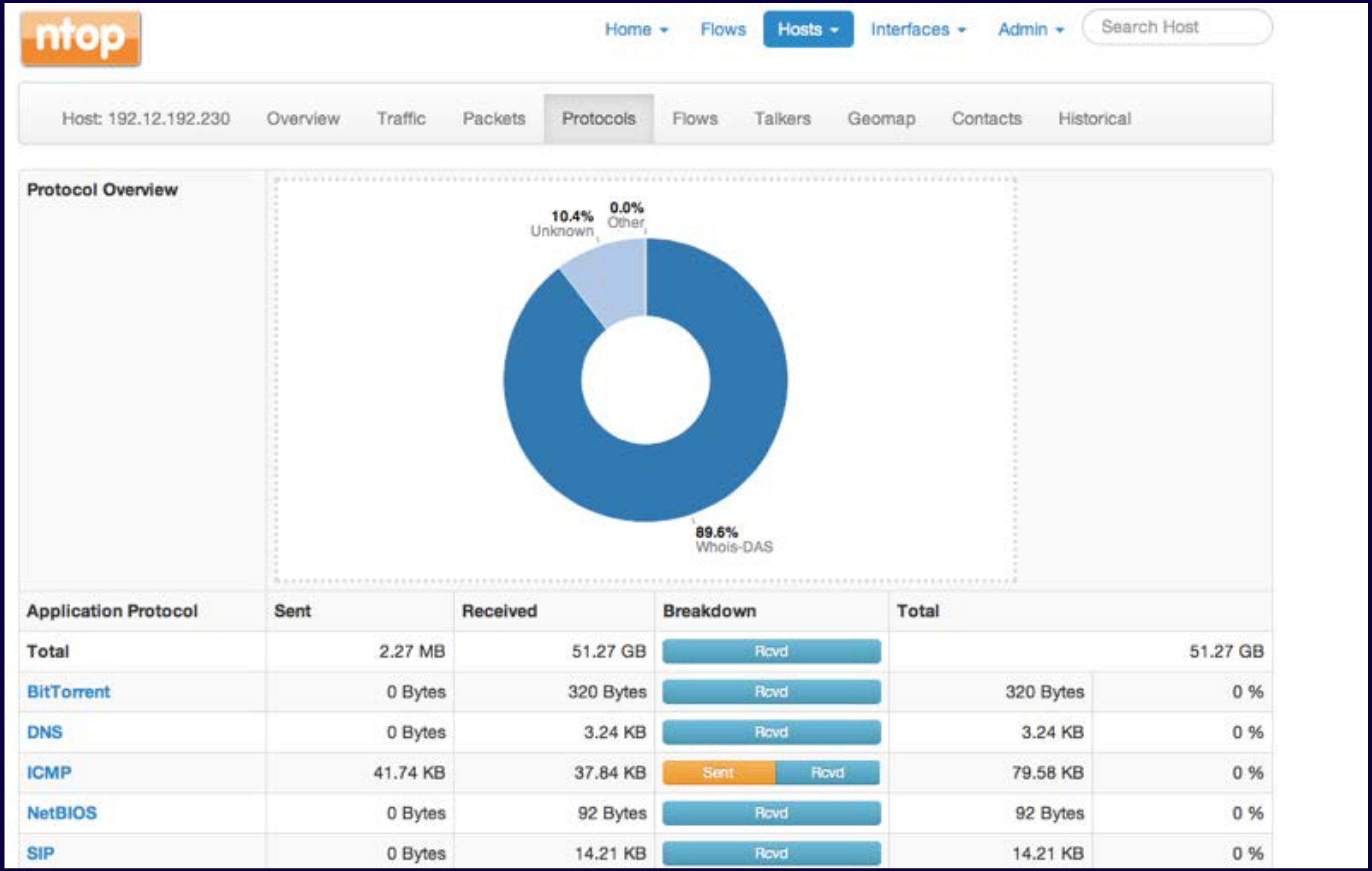
Showing 1 to 10 of 1275 rows

← First | Prev | 1 | 2 | 3 | 4 | 5 | Next | Last →

# Host Monitoring [2/3]




# Host Monitoring [3/3]

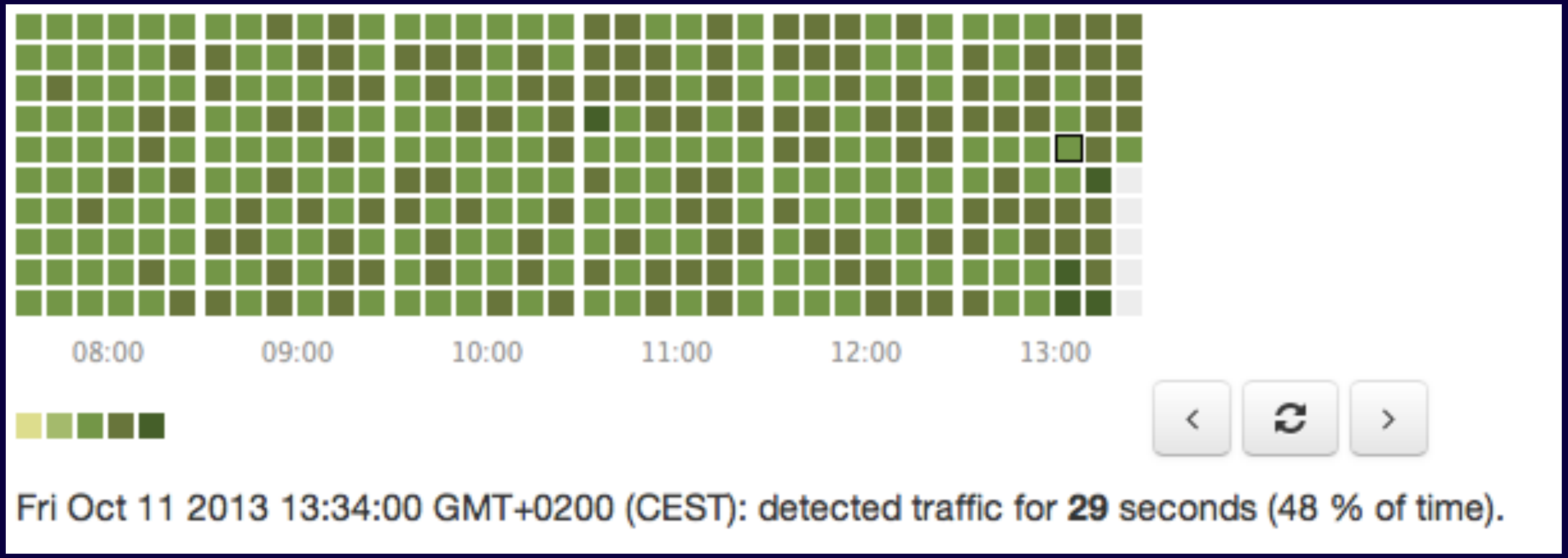


# Activity Map

- 1 second resolution host and aggregation activity
- Compressed bitmap

```
> ls -l client14.dropbox.com
4 -rw-rw-rw- 1 nobody nogroup 24 Oct 11 02:31 client14.dropbox.com
```

- Saved persistently on disk (Local Hosts only)



# Traffic Aggregations [1/2]

- nDPI extracts specific attributes from traffic that ntopng aggregates (if configured):
  - DNS/Whois responses
  - HTTP host names
  - Operating System (from HTTP headers)
- Aggregations can be enabled (they are off by default) and are handled just as flows and hosts.

# Traffic Aggregations [2/2]

## Aggregations

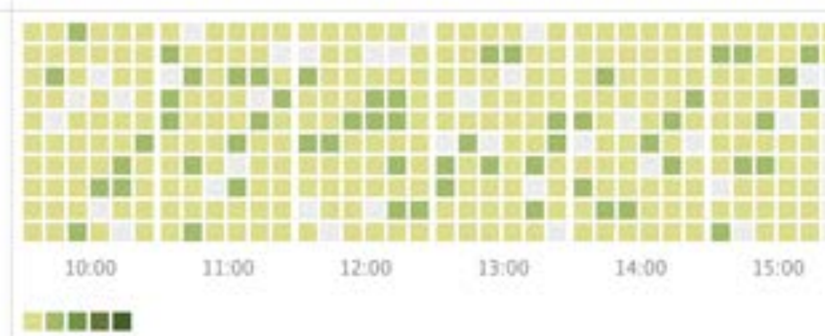
10 · Aggregations

- All
- DNS
- Operating System
- HTTP

| Name | Protocol | Seen Since | Last Seen | Qu |
|---|---|---|---|---|
| dnsmon.nic.it | HTTP | 1 day, 46 min, 20 sec | 4 sec | |
| Linux x86_64 | Operating System | 1 day, 46 min, 20 sec | 4 sec | |
| daisy.ubuntu.com | DNS | 1 day, 46 min, 16 sec | 28 sec | 13,613 — |
| i7.ntop.org | HTTP | 11 sec | 1 sec | 26 — |
| Intel Mac OS X 10_8_5 | Operating System | 11 sec | 1 sec | 26 — |
| www.google.com | DNS | 1 min, 30 sec | 39 sec | 15 — |
| pnnptfiomq.nic.it | DNS | 39 sec | 39 sec | 2 — |
| tdkoxonujj.nic.it | DNS | 40 sec | 40 sec | 2 — |
| ilkomppxne.nic.it | DNS | 39 sec | 39 sec | 2 — |
| checkip.dyndns.com | DNS | 40 sec | 40 sec | 2 — |

Showing 1 to 10 of 20 rows

← First · Prev · 1 · 2 · Next · Last →

| | |
|---|---|
| **Name** | daisy.ubuntu.com |
| **Family** | DNS |
| **First Seen** | 10/10/2013 15:05:05 [1 day, 46 min, 43 sec ago] |
| **Last Seen** | 11/10/2013 15:51:33 [30 sec ago] |
| **Contacts Received** | 13,622 — |
| **Activity Map** |  |

# Hosts and Aggregations Interaction


# Geolocation

| Host: 192.12.193.5 | Overview | Traffic | Packets | Protocols |
|---|---|---|---|---|

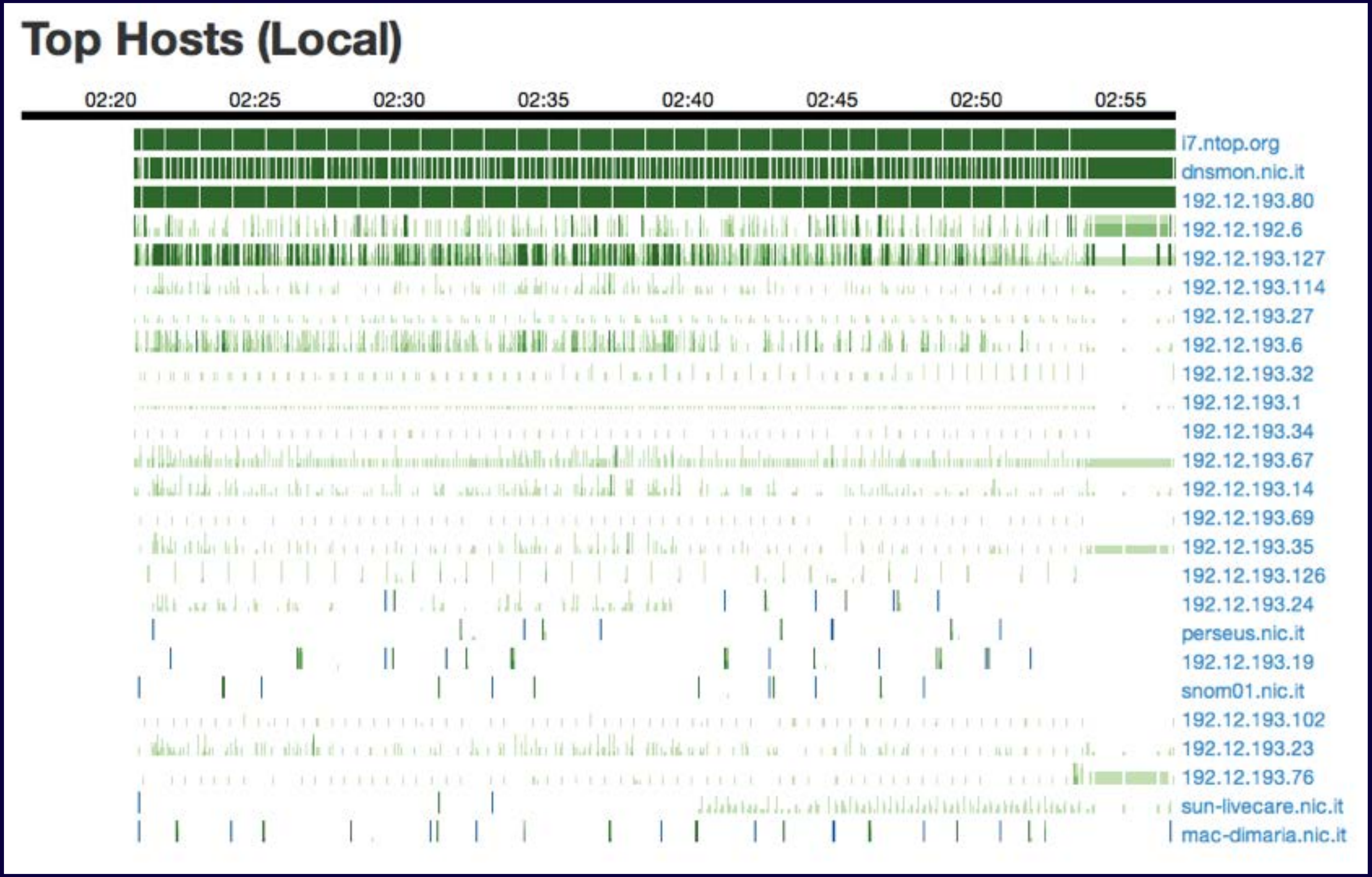
| | |
|---|---|
| **(Router) MAC Address** | 78:AC:C0:A7:0D:4C |
| **IP Address** | 192.12.193.5 [ Pisa 🇮🇹 ] |

## Hosts GeoMap


**NOTE**

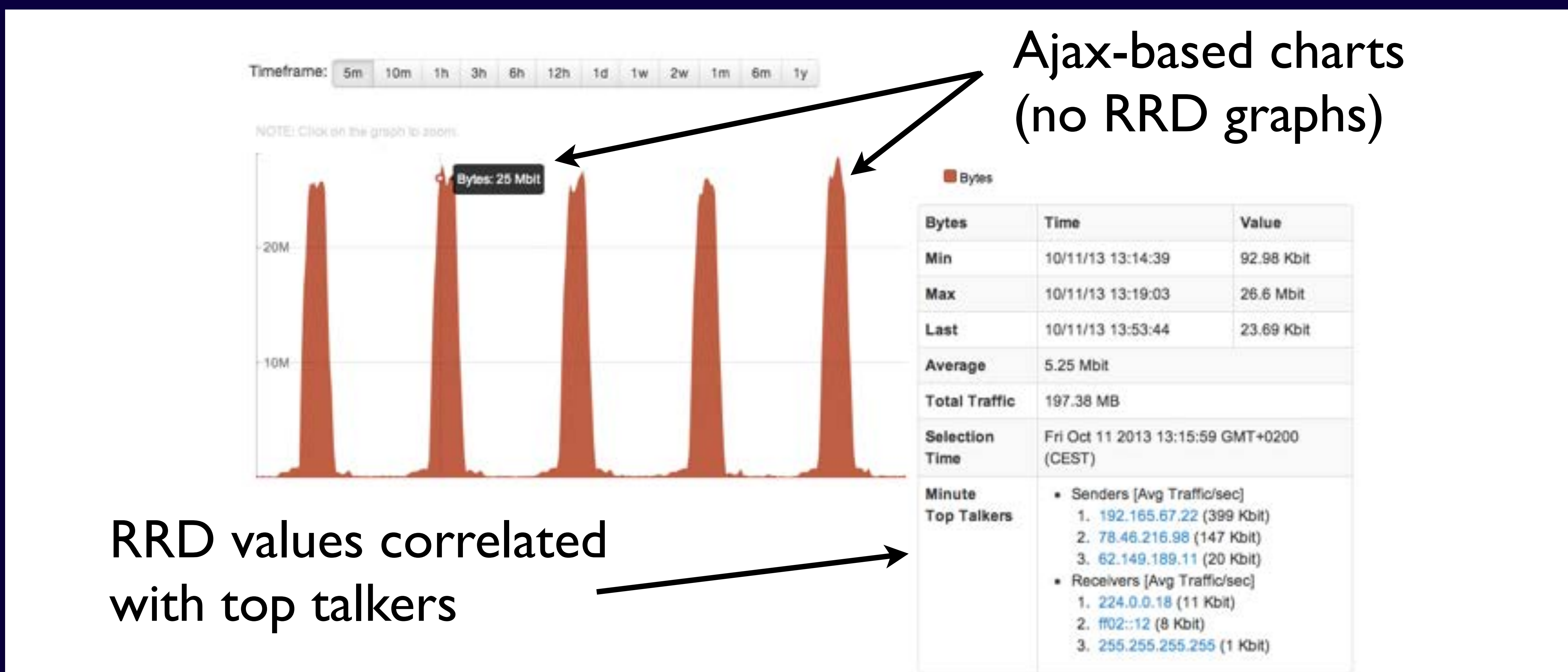
1. Browser reported home map location [Latitude: 43.71949459086955, Longitude: 10.4219399273913]
2. In order to visualize maps you must:
   1. Have a working Internet connection.
   2. Have compiled ntopng with geolocation and started with it.
   3. Have active flows between peers with public IP addresses.
3. HTML browser geolocation is used to place on map hosts based on unknown locations.

# Live Host Activities



# Historical Activities

- All relevant counters are saved on disk in RRD.
- Interface counters are saved with 1 second resolution. Hosts counters every 5 minutes.

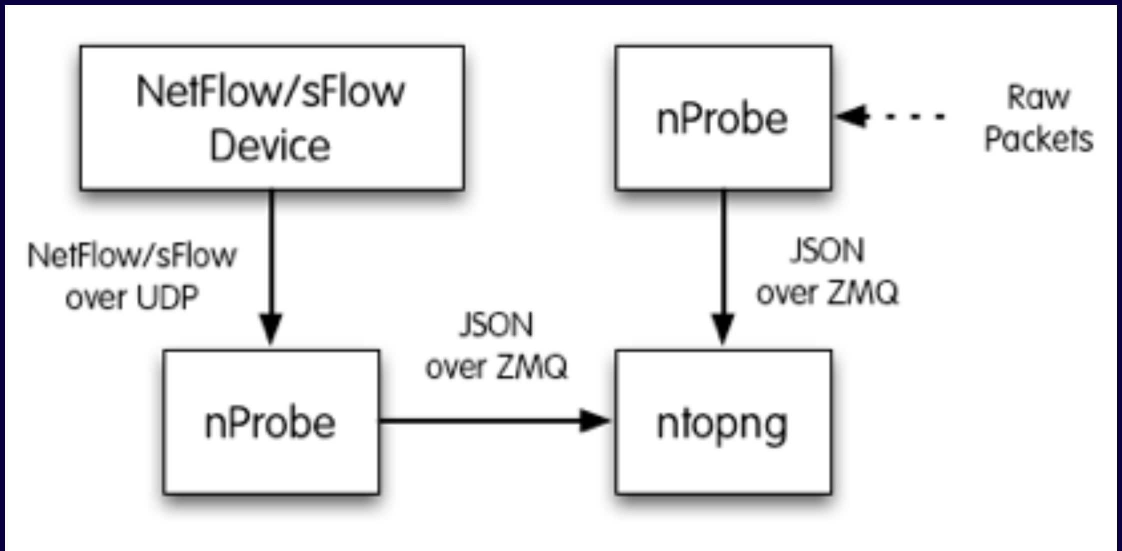


# Using Ntopng as a Live Data Source

- ntopng is a server able to serve data to third party applications via HTTP.
- Data is exported via JSON.
- This mechanism can be extended via Lua scripts.


# Using Ntopng with NetFlow/sFlow

- ntopng can handle flows (Net/sFlow) via nProbe.




- Data Collector (ntopng)
  - `ntopng -i tcp://127.0.0.1:5556`
- Probe (nProbe)
  - `nprobe --zmq "tcp://*:5556" -i eth1 -n none` (probe mode)
  - `nprobe --zmq "tcp://*:5556" -i none -n none --collector-port 2055` (sFlow/NetFlow collector mode)

PERFECTLY IMPERFECT

19-20-21 agosto
2016
Parco Ex Caserma Cocco
Pescara

# NtopNg e il monitoraggio del traffico di rete (in high-speed network)


**Giuseppe Augiero**

<talk@augiero.it> - @GiuseppeAugiero

**Luca Deri**

<deri@ntop.org> - @lucaderi