Network Traffic and Security Monitoring Using ntopng and InfluxDB

27
JUN

Wednesday, June 27, 2018

Monitoring with Time Series

Hosted by Katy F.
From Time Series SF
Public group ?

Luca Deri <deri@ntop.org>
@lucaderi

Part I:
Welcome to ntopng

About Me

- (1997) Founder of the ntop.org project with the purpose of creating a simple, and open source web-based traffic monitoring application.
- Lecturer at the University of Pisa, Italy.
- Author of various open source projects
  - n2n: peer-to-peer layer 2 VPN.
  - nDPI: deep-packet-inspection library.
  - PF_RING: high-speed packet capture and transmission.

About ntop.org

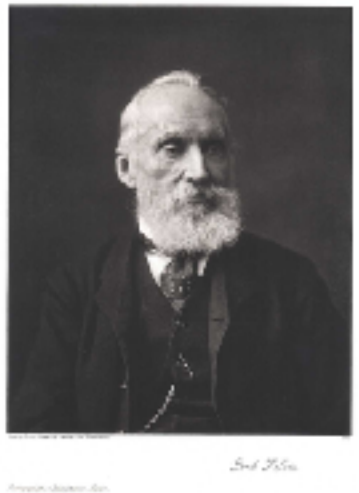
- ntop develops open source network traffic monitoring applications.
- ntop (circa 1998) is the first app we released and it is a web-based network monitoring application.
- Today our products range from traffic monitoring, high-speed packet processing, deep-packet inspection, and IDS/IPS acceleration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

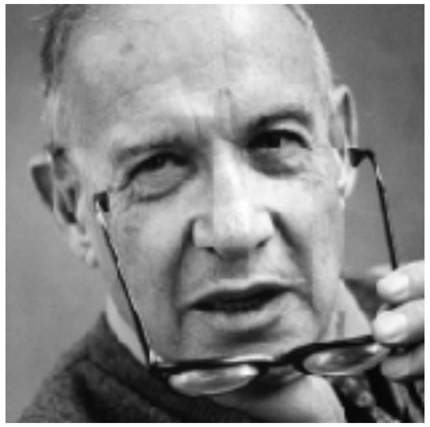
ntop's Approach to Traffic Monitoring

- 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.

Motivation For Traffic Monitoring

If you can’t measure it, you can’t improve it
(Lord Kelvin, 1824 – 1907)





If you can’t measure it, you can’t manage it
(Peter Drucker, 1909 – 2005)

What Happens in Our Network?

- Do we have control over our network?
- It’s not possible to imagine a healthy network without a clear understanding of traffic flowing on our network.
- Knowledge is the first step towards evaluation of potential network security issues.
- Event correlation can provide us timely information about our network health.

How To Install ntopng

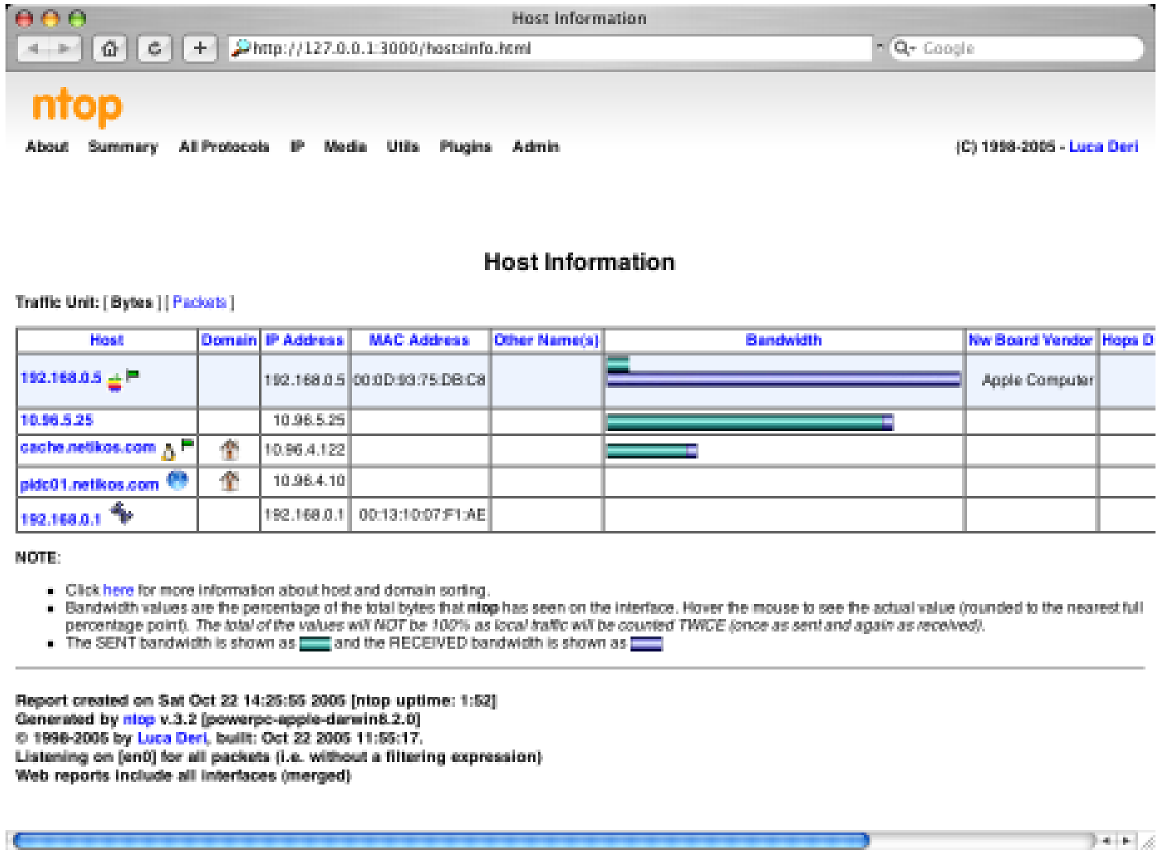
- Source code
  https://github.com/ntop/ntopng

- Distributions
  Ubuntu/Debian, FreeBSD…. (included in the distro)
  OSX (brew)

- Binary Packages (nightly + stable)
  http://packages.ntop.org (Debian/Ubuntu/CentOS, OSX, RaspberryPI/ARM)

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.



Host Information

Traffic Unit: [Bytes] [Packets]

Host	Domain	IP Address	MAC Address	Other Name(s)	Bandwidth	Nw Board Vendor	Hops D
192.168.0.5		192.168.0.5	00:0D:93:75:DB:C8			Apple Computer	
10.96.5.25		10.96.5.25					
cache.netikos.com		10.96.4.122					
pidc01.netikos.com		10.96.4.10					
192.168.0.1		192.168.0.1	00:13:10:07:F1:AE				

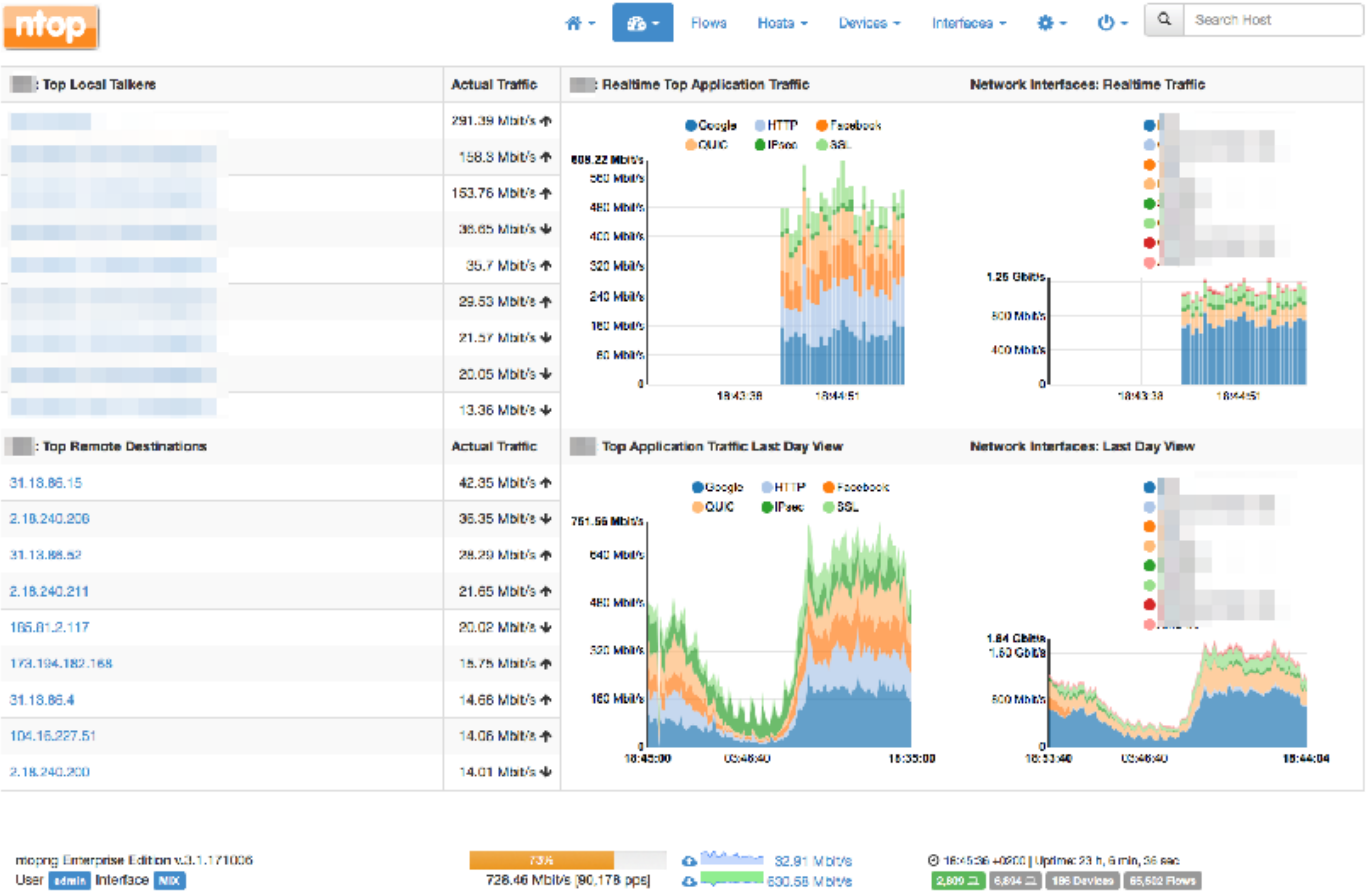
NOTE:

- Click here for more information about host and domain sorting.
- Bandwidth values are the percentage of the total bytes that **ntop** has seen on the interface. Hover the mouse to see the actual value (rounded to the nearest full percentage point). *The total of the values will NOT be 100% as local traffic will be counted TWICE (once as sent and again as received).*
- The SENT bandwidth is shown as 🟩 and the RECEIVED bandwidth is shown as 🟦

Report created on Sat Oct 22 14:25:55 2005 [ntop uptime: 1:52]
Generated by ntop v.3.2 [powerpc-apple-darwin8.2.0]
© 1998-2005 by Luca Deri, built: Oct 22 2005 11:55:17.
Listening on [en0] for all packets (i.e. without a filtering expression)
Web reports include all interfaces (merged)

- 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.

Welcome to ntopng (2013-)



: Top Local Talkers	Actual Traffic
[redacted]	291.39 Mbit/s ↑
[redacted]	158.3 Mbit/s ↑
[redacted]	153.76 Mbit/s ↑
[redacted]	38.65 Mbit/s ↓
[redacted]	35.7 Mbit/s ↑
[redacted]	29.53 Mbit/s ↑
[redacted]	21.57 Mbit/s ↓
[redacted]	20.05 Mbit/s ↓
[redacted]	13.36 Mbit/s ↓

: Top Remote Destinations	Actual Traffic
31.13.86.15	42.35 Mbit/s ↑
2.18.240.206	36.35 Mbit/s ↓
31.13.88.52	28.29 Mbit/s ↑
2.18.240.211	21.65 Mbit/s ↑
185.61.2.117	20.02 Mbit/s ↓
173.194.182.168	15.75 Mbit/s ↑
31.13.86.4	14.66 Mbit/s ↑
104.16.227.51	14.06 Mbit/s ↑
2.18.240.200	14.01 Mbit/s ↓

ntopng Enterprise Edition v.3.1.171006
User admin Interface NIX

73%
728.46 Mbit/s [90,178 pps]

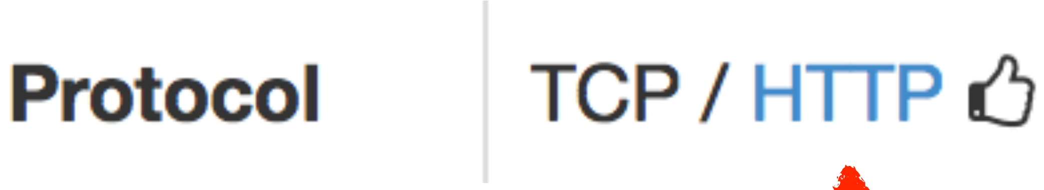
32.91 Mbit/s
630.58 Mbit/s

16:45:36 +0200 | Uptime: 23 h, 6 min, 36 sec
2,809 | 6,804 | 186 Devices | 65,502 Flows

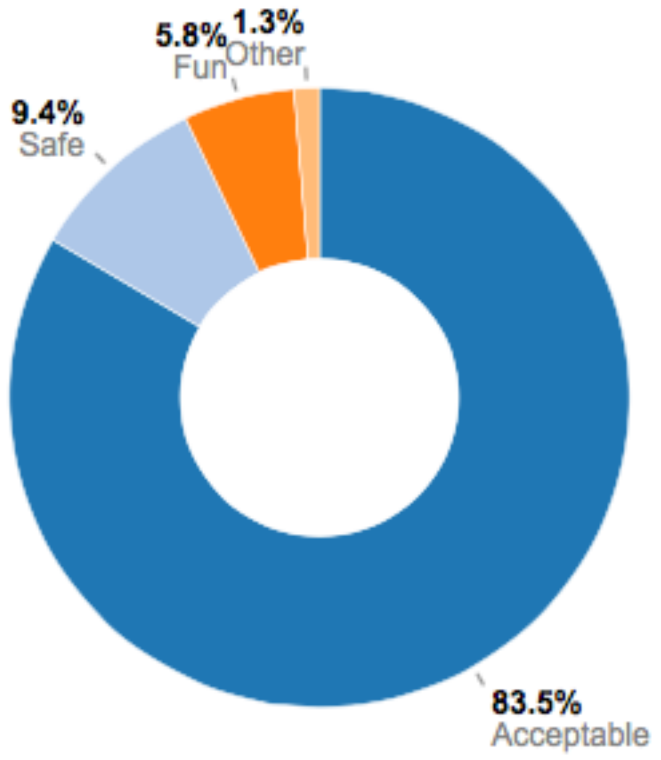
ntopng Walk Through: Interpreting Data

Layer 4 Protocol

Good or Bad?

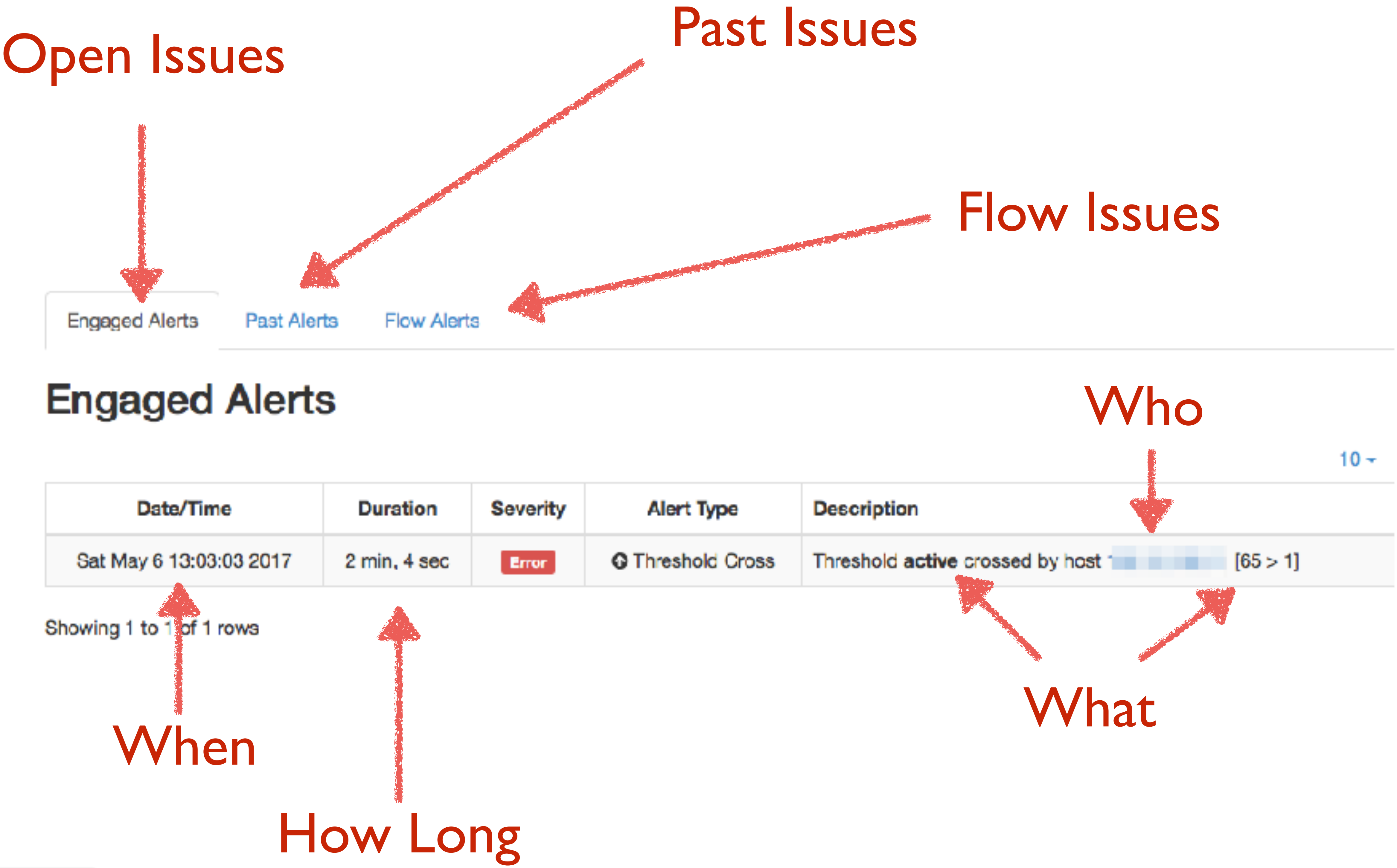


Protocol | TCP / HTTP 👍



Layer 7 Protocol

ntopng Walk Through: Alerts



Open Issues

Past Issues

Flow Issues

Engaged Alerts | Past Alerts | Flow Alerts

Engaged Alerts

Who

10

Date/Time	Duration	Severity	Alert Type	Description
Sat May 6 13:03:03 2017	2 min, 4 sec	Error	Threshold Cross	Threshold **active** crossed by host [redacted] [65 > 1]

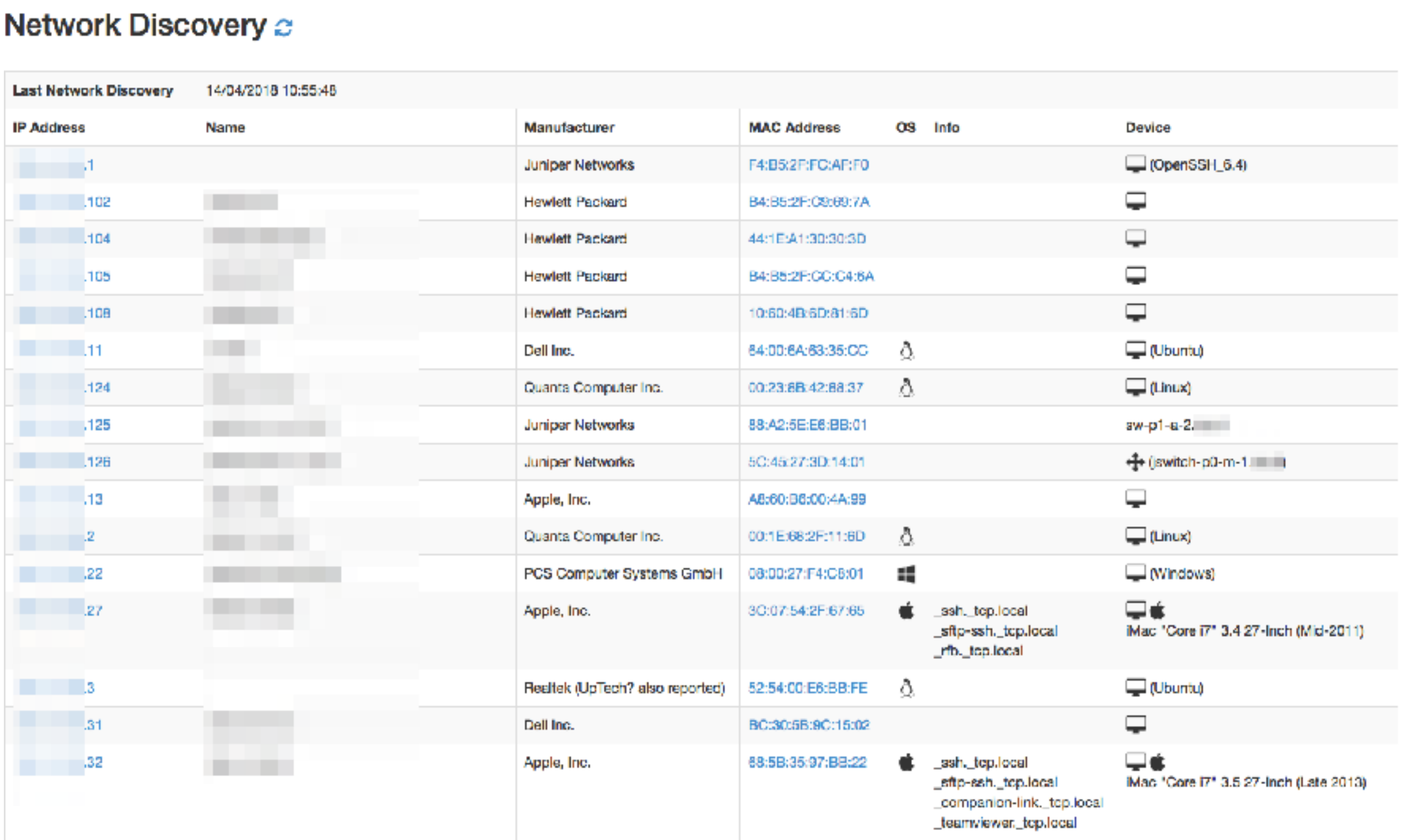
Showing 1 to 1 of 1 rows

What

When

How Long

ntopng Walk Through: Discovery

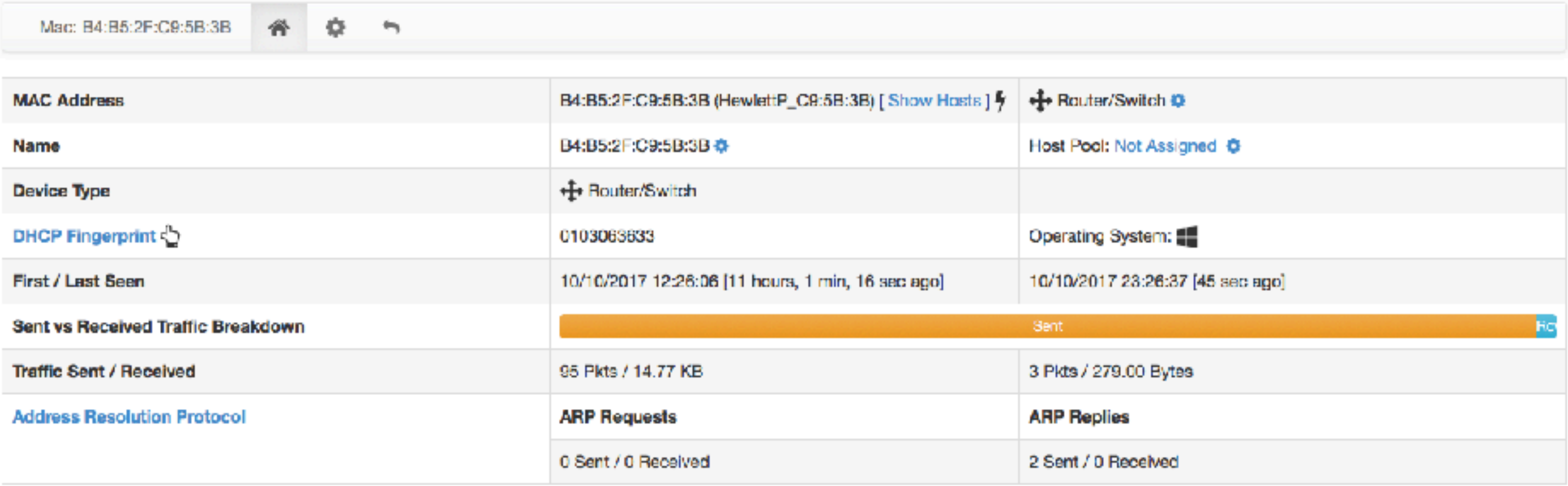


Network Discovery ⟳

Last Network Discovery 14/04/2018 10:55:48

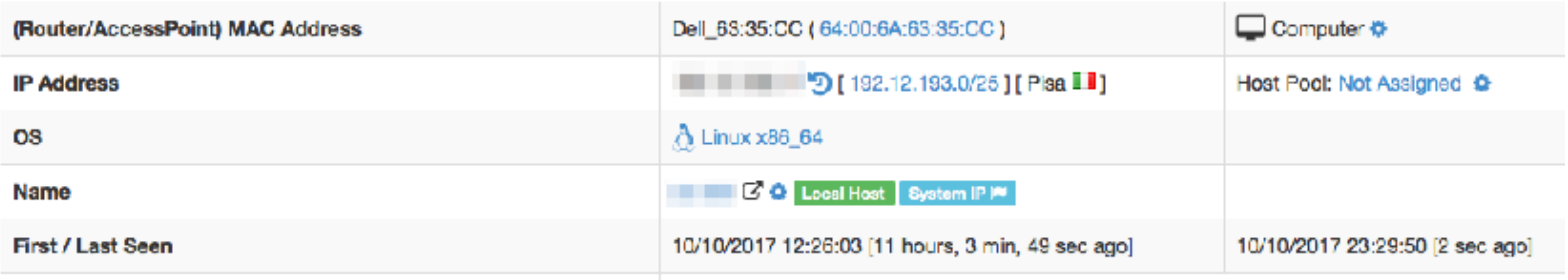
IP Address	Name	Manufacturer	MAC Address	OS	Info	Device
[redacted].1		Juniper Networks	F4:B5:2F:FC:AF:F0			🖥 (OpenSSH_6.4)
[redacted].102	[redacted]	Hewlett Packard	B4:B5:2F:C8:69:7A			🖥
[redacted].104	[redacted]	Hewlett Packard	44:1E:A1:30:30:3D			🖥
[redacted].105	[redacted]	Hewlett Packard	B4:B5:2F:CC:C4:6A			🖥
[redacted].108	[redacted]	Hewlett Packard	10:60:4B:6D:81:6D			🖥
[redacted].11	[redacted]	Dell Inc.	84:00:6A:63:35:CC	🐧		🖥 (Ubuntu)
[redacted].124	[redacted]	Quanta Computer Inc.	00:23:8B:42:88:37	🐧		🖥 (Linux)
[redacted].125	[redacted]	Juniper Networks	88:A2:5E:E6:BB:01			sw-p1-a-2.[redacted]
[redacted].126	[redacted]	Juniper Networks	5C:45:27:3D:14:01			✣ (switch-p0-m-1.[redacted])
[redacted].13	[redacted]	Apple, Inc.	A8:60:B6:00:4A:99			🖥
[redacted].2	[redacted]	Quanta Computer Inc.	00:1E:68:2F:11:6D	🐧		🖥 (Linux)
[redacted].22	[redacted]	PCS Computer Systems GmbH	08:00:27:F4:C8:01	⊞		🖥 (Windows)
[redacted].27	[redacted]	Apple, Inc.	3C:07:54:2F:67:65		_ssh._tcp.local _sftp-ssh._tcp.local _rfb._tcp.local	🖥 iMac "Core i7" 3.4 27-Inch (Mid-2011)
[redacted].3		Realtek (UpTech? also reported)	52:54:00:E6:BB:FE	🐧		🖥 (Ubuntu)
[redacted].31	[redacted]	Dell Inc.	BC:30:5B:8C:15:02			🖥
[redacted].32	[redacted]	Apple, Inc.	88:5B:35:97:B8:22		_ssh._tcp.local _sftp-ssh._tcp.local _companion-link._tcp.local _teamviewer._tcp.local	🖥 iMac "Core i7" 3.5 27-Inch (Late 2013)

ntopng Walk Through: Drill Down



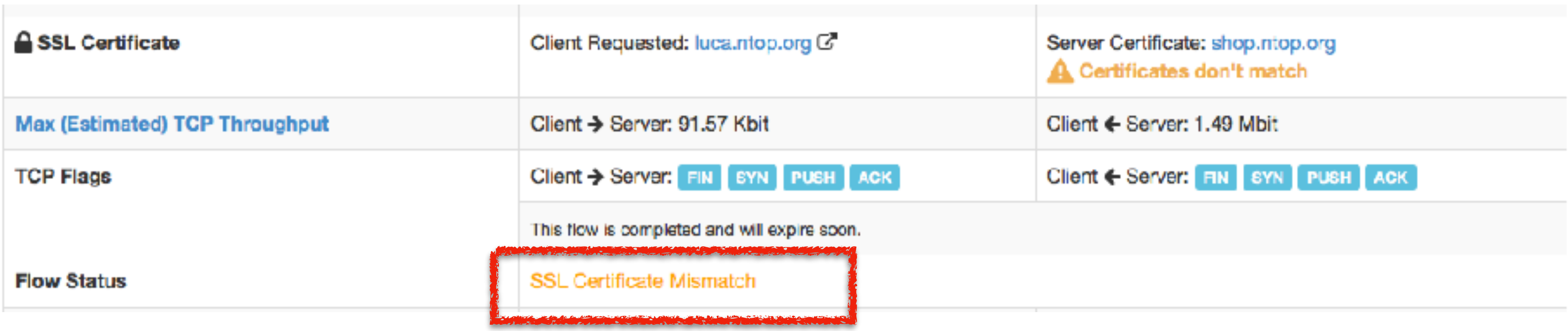
Mac: B4:B5:2F:C9:5B:3B

MAC Address	B4:B5:2F:C9:5B:3B (HewlettP_C9:5B:3B) [Show Hosts]	Router/Switch
Name	B4:B5:2F:C9:5B:3B	Host Pool: Not Assigned
Device Type	Router/Switch	
DHCP Fingerprint	0103063633	Operating System:
First / Last Seen	10/10/2017 12:26:06 [11 hours, 1 min, 16 sec ago]	10/10/2017 23:26:37 [45 sec ago]
Sent vs Received Traffic Breakdown	Sent Rc	
Traffic Sent / Received	95 Pkts / 14.77 KB	3 Pkts / 279.00 Bytes
Address Resolution Protocol	**ARP Requests**	**ARP Replies**
	0 Sent / 0 Received	2 Sent / 0 Received



(Router/AccessPoint) MAC Address	Dell_63:35:CC (64:00:6A:63:35:CC)	Computer
IP Address	[redacted] [192.12.193.0/25] [Pisa]	Host Pool: Not Assigned
OS	Linux x86_64	
Name	[redacted] Local Host System IP	
First / Last Seen	10/10/2017 12:26:03 [11 hours, 3 min, 49 sec ago]	10/10/2017 23:29:50 [2 sec ago]

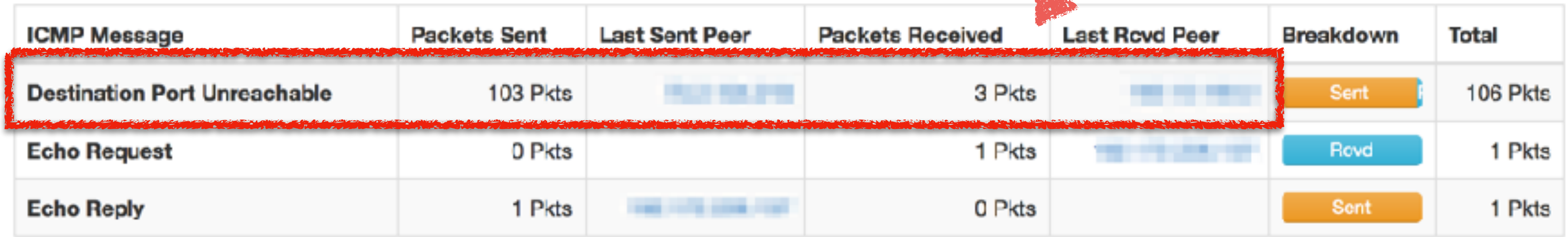
ntopng Walk Through: Monitoring



SSL Certificate	Client Requested: luca.ntop.org	Server Certificate: shop.ntop.org ⚠ Certificates don't match
Max (Estimated) TCP Throughput	Client → Server: 91.57 Kbit	Client ← Server: 1.49 Mbit
TCP Flags	Client → Server: FIN SYN PUSH ACK	Client ← Server: FIN SYN PUSH ACK
Flow Status	This flow is completed and will expire soon. SSL Certificate Mismatch	

Invalid Configuration or Threat ?

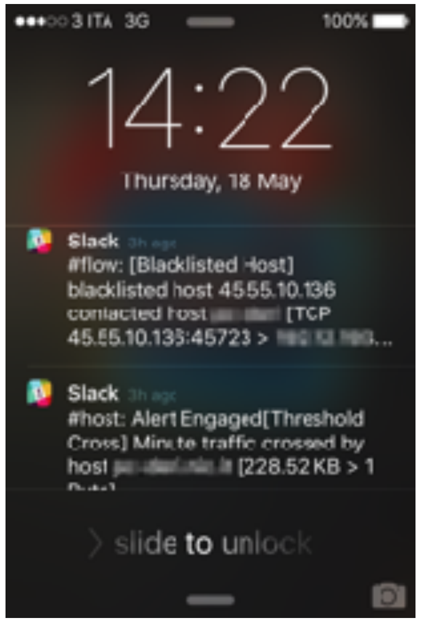
Service Down or Scan?



ICMP Message	Packets Sent	Last Sent Peer	Packets Received	Last Rcvd Peer	Breakdown	Total
Destination Port Unreachable	103 Pkts	[redacted]	3 Pkts	[redacted]	Sent	106 Pkts
Echo Request	0 Pkts		1 Pkts	[redacted]	Rcvd	1 Pkts
Echo Reply	1 Pkts	[redacted]	0 Pkts		Sent	1 Pkts

Yes You Can

- Embedded alerting system pluggable with nagios and messaging systems.



●●●○○ 3 ITA 3G 100%

14:22

Thursday, 18 May

Slack 3h ago
#flow: [Blacklisted Host] blacklisted host 4555.10.136 contacted host [redacted] [TCP 45.55.10.135:45723 > [redacted]...

Slack 3h ago
#host: Alert Engaged[Threshold Cross] Minute traffic crossed by host [redacted] [228.52 KB > 1 Byte]

> slide to unlock

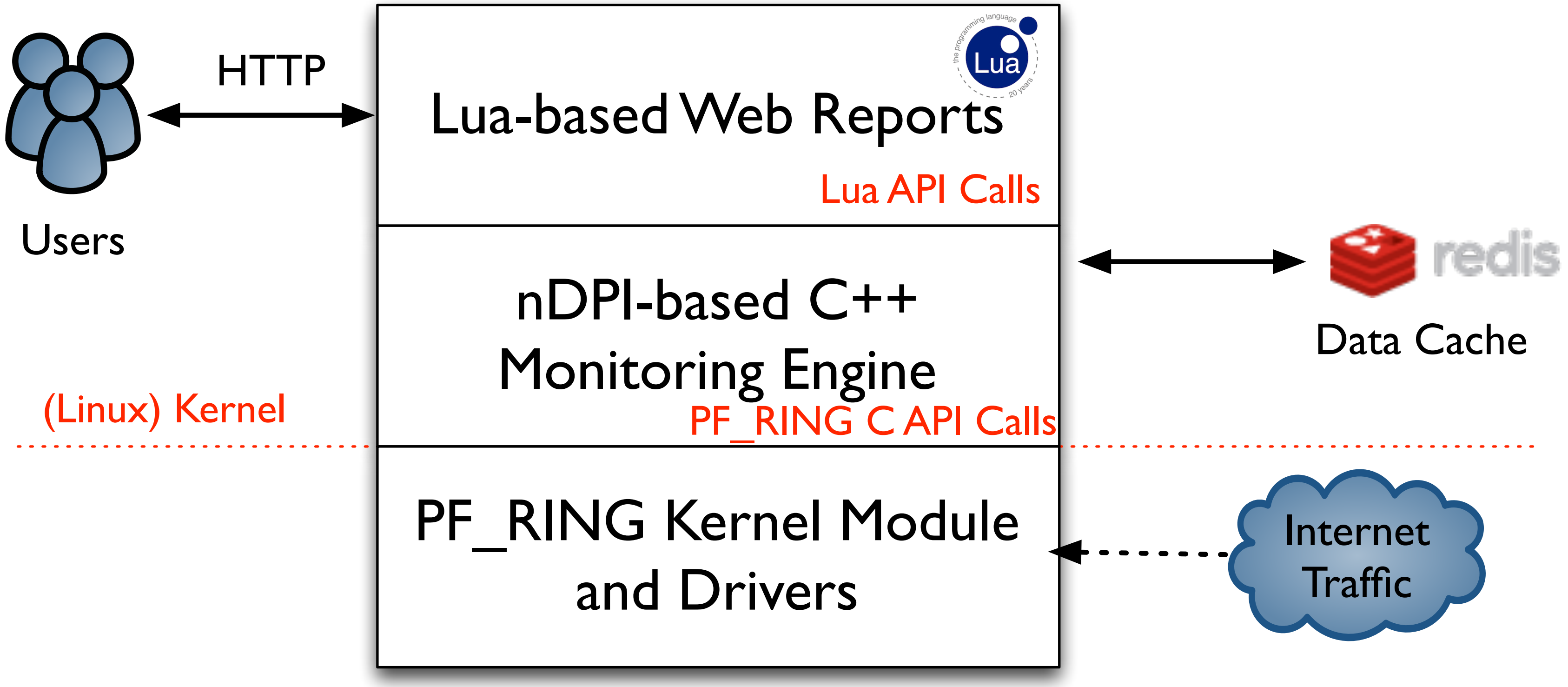
- Use it as Grafana datasource

ntopng by ntop
DATA SOURCE
ntopng datasource

- Ready for openstack WIRESHARK VAGRANT docker elastic
- nDPI: passive mode = monitoring, inline = IPS
- Support for NetFlow/sFlow/SNMP.
- Passive/Active Network Device Discovery.
- Traffic Behaviour Analysis.

ntopng Architecture

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



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

Part II:
Network Monitoring and TimeSeries

Monitoring Granularity

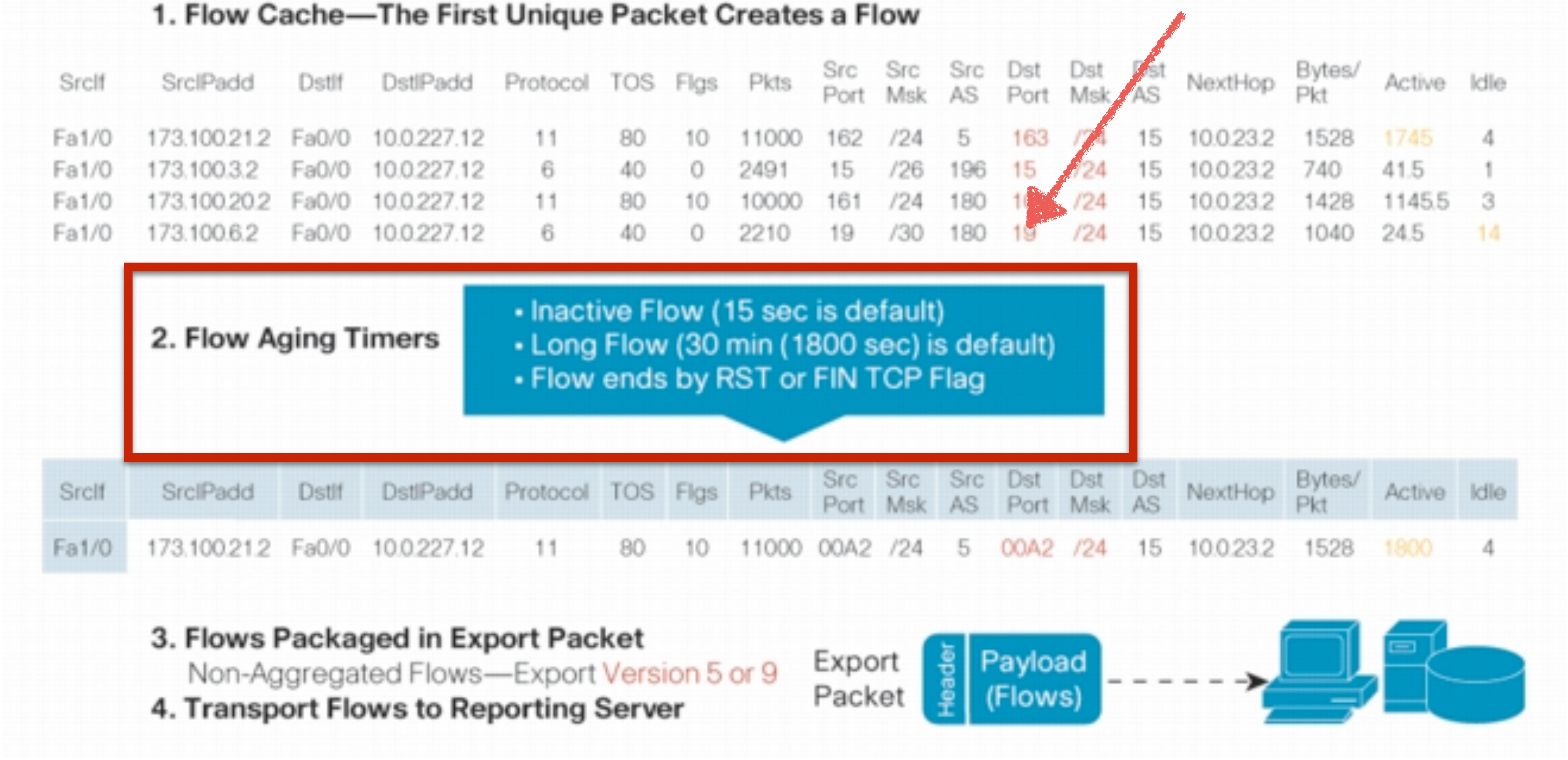
- Historically network devices such as routers and switches produce monitoring information based on the traffic that traverses such devices.
- Various techniques are used to avoid exhausting network device resources, in particular CPU and memory, including:
  - Packet sampling (i.e. consider one packet in N and not all traffic) that is native in sFlow for instance.
  - Limit counter polling. For instance SNMP device counters are updated every X seconds (e.g. 3 sec), so polling them too fast won't help to produce fine grained data.

Average Values As Default Option [1/2]

- Most monitoring protocols have been designed to produce average data.
- If you wish limited SNMP polling frequency produces average data.
- NetFlow, the leading monitoring protocols for Internet traffic is even worse. Default flow duration is set in minutes, thus making measurements even less granular.

Average Values As Default Option [2/2]

Netflow RFC



1. Flow Cache—The First Unique Packet Creates a Flow

SrcIf	SrcIPadd	DstIf	DstIPadd	Protocol	TOS	Flgs	Pkts	Src Port	Src Msk	Src AS	Dst Port	Dst Msk	Dst AS	NextHop	Bytes/ Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	162	/24	5	163	/24	15	10.0.23.2	1528	1745	4
Fa1/0	173.100.3.2	Fa0/0	10.0.227.12	6	40	0	2491	15	/26	196	15	/24	15	10.0.23.2	740	41.5	1
Fa1/0	173.100.20.2	Fa0/0	10.0.227.12	11	80	10	10000	161	/24	180	[illegible]	/24	15	10.0.23.2	1428	1145.5	3
Fa1/0	173.100.6.2	Fa0/0	10.0.227.12	6	40	0	2210	19	/30	180	19	/24	15	10.0.23.2	1040	24.5	14

2. Flow Aging Timers

- Inactive Flow (15 sec is default)
- Long Flow (30 min (1800 sec) is default)
- Flow ends by RST or FIN TCP Flag

SrcIf	SrcIPadd	DstIf	DstIPadd	Protocol	TOS	Flgs	Pkts	Src Port	Src Msk	Src AS	Dst Port	Dst Msk	Dst AS	NextHop	Bytes/ Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	00A2	/24	5	00A2	/24	15	10.0.23.2	1528	1800	4

3. Flows Packaged in Export Packet

Non-Aggregated Flows—Export Version 5 or 9

4. Transport Flows to Reporting Server

ntopng Realtime Measurement

- The ntopng engine has been designed for realtime traffic measurement. It can report packet-based information in realtime and the engine can be polled continuously while processing traffic.
- While the user interface can be refreshed from 1-5 sec via Ajax/WebSocket, writing time series to disk can be difficult when the number of metrics is high in cardinality.

Metrics Cardinality: SNMP

- Each interface has 5 counters (ifIn/outPackets, ifIn/outBytes, ifInErrors).
- Switches can have have a few ports (e.g. 24) or 500+ ports on core switches.
- SMEs can have a few (e.g. 4) switches, while ISP can have hundred of them.
- In summary ntopng has been designed to poll thousand ports across multiple switches. For this reason, unless sFlow is used, it is not possible to implement sub-minute polling (small networks) or sub-5 minute on larger networks in order to avoid putting too much load on devices or producing too much network traffic due to SNMP.

Metrics Cardinality: Internet Traffic [1/3]

- Usually traffic counters are kept one per local host (i.e. those hosts that belong to the company being monitored).
- SMEs can have as low as 20 hosts (today this is the typical number of connected devices of a home, when considering IoT, tables, phones etc) or thousand of them.
- Even though nDPI supports ~240 protocols, most hosts do not use all of them, and usually they are limited to about 30 protocols/host.

Metrics Cardinality: Internet Traffic [2/3]

- In total a host has:
  - 4 counters in/out packets + in/out bytes per nDPI protocol (so in average 4 x 30 = 120)
  - About 20 counters for other metrics such as retransmission, packets-out-of-order,…
  - In total in average 120+20 counters/host.
- In addition we need to add counters for additional elements such as visited autonomous systems, networks, countries etc.

Metrics Cardinality: Internet Traffic [3/3]

- In total on a network with 254 hosts (a /24 CIDR) we have ~35-50k counters to save every minute.
- Further challenges:
  - The number of these counters changes overtime (e.g. when a host is disconnected, no traffic is reported, so no counters are produced).
  - Counters granularity is not homogeneous: interface counters are per second, host per minute, SNMP every 5 mins.
  - It would be desirable to have more find grained counters (a value every 5/10 sec) but this can significantly increase the complexity of our problem.

ntopng and Timeseries [1/4]

- ntopng has an embedded "crontab" that executes tasks:
  - Per system (singleton), monitored interface (physical 1-3, logical up to 128).
  - Every second, minute, 5 minute, hour, day.
- Tasks
  - Second: Interface counters in/out packets/bytes, drops, packets out-of-order….
  - 5 Minutes: hosts and SNMP counters.

ntopng and Timeseries [2/4]

- RRD-based system used for historical reasons (ntop used it) and because it is file based (pro) that allows us to run it on the same host where ntopng runs with no external service dependency.
- Cons:
  - Unable to cope with large number of hosts (5 mins are not enough) so we are unable to save all the data to disk.
  - High load on the filesystem in particular on low-end/ embedded devices that feature slow disks.

ntopng and Timeseries [3/4]

- RRD cons (cont).
  - …old programming library designed as a tool rather than a library: every function call uses argc/argv.
  - Thread support works but it is mostly a hack.
  - Too many library dependencies even if ntopng uses it as time series database with no graph generation.
  - The library is not moving forward since a long time (5+ years) in terms of new developments and we do not expect this to change anytime soon. In essence this is at the moment a project in maintenance mode.

ntopng and Timeseries [4/4]

- Beside all these limitations in RRD design, the main driving force for replacing it in ntopng are:
  - Inability to easily compare timeseries
    - Select all hosts that in the past hour has made 20% up of traffic with respect to the previous hour.
    - Trigger an alert when host X is sending/receiving double the traffic of the second top-sender in the last hour.
  - Inability to handle long-term measurements without aggregation and "native" loss of precision (RRD basically implements always an average and if you insert 1 you will extract 0.999999999 that can be annoying sometimes).

What Alternatives to RRD? [1/2]

- Since a few years, timeseries databases are becoming popular, and there are many interesting projects.
- In 2014 ntop started to evaluate InfluxDB 0.x but we decided to wait a bit longer before integrating it into the project as it was not mature enough (the engine was under heavy changes, we had data loss during developments).
- In 2017, we decided that RRD was becoming a real bottleneck to ntopng evolution so we started to find a solution to this problem.

What Alternatives to RRD? [2/2]

- Out to the many timeseries databases we believe that the only two real options are InfluxDB and Prometheus, otherwise better to stay with RRD.
- ntopng currently supports both of them for data dump, but we have decided to bet in InfluxDB for a few reasons:
    - Ecosystem (Chronograf….), vibrant community and company behind the project.
    - Flexibility: we prefer to push data to Influx, rather than ntopng to be pulled as Prometheus does.
    - Other ntop users use it in various projects and they are happy of performance and stability.
    - Generic DB, not targeting a specific domain such as Prometheus.

Influx Migration Implications [1/2]

- Create a timeseries Lua layer inside ntopng to allow people to use both RRD and Influx.
- Mask RRD/Influx differences during data extraction:
    - RRD has automatic data consolidation/rollup that instead we need to configure in Influx with continuous queries.
    - RRD handles natively counters/gauges whereas on Influx we need to do that in queries.

Influx Migration Implications [2/2]

- Mask RRD/Influx differences (cont)
  - In RRD we set time the policy and data value boundaries at archive creation: in case our data is out of range, measurements are automatically discarded. In Influx we need to do that during data extraction so applications need to be aware of that (hint: our driver will handle that).
  - RRD normalises data automatically (and sometimes this is not a plus) at a specific resolution based on the query time-range: with Influx we have more precise measurements that are a good thing, but that have to be properly handled otherwise we end up having too many data points in graphs that will result in slow Ajax-based rendering.

Migration Progress: Completed

- Created a timeseries API that supports both RRD and InfluxDB for (seamlessly) migrating our users to the Influx.
  - Abstraction layer over the timeseries data sources so that from the ntopng perspective there are no differences across timeseries DBs.
  - The new timeseries API uses schemas, a collection of tags and metrics, to declare the structure of the data and defines a generic timeseries driver API (so far for RRD and InfluxDB) to manipulate timeseries data.
  - All metrics are exported to Influx via HTTP: in order to avoid making too many HTTP POSTs, we cache updates for a few seconds and push them in small batches.

Migration Progress: Ongoing

- ntopng need some rework to avoid periodic tasks and thus move from 5-min host counters to 1 minute or less. On small/medium networks we would like to go down to 5/10 seconds.
- Configurable timeseries data sources and export endpoints: due to some existing design limitations we are currently unable to mix timeseries coming from different ntopng's or network interfaces.
- Completely independent from RRD: the current GUI relies on some RRD features, so we are working at creating a ntopng that is RRD-free (too many dependencies).
- More dynamic and responsive charts exploiting native platform facilities (e.g. topX InfluxDB has)
- Use ntopng as a data source for third-party apps, via a generic TS query URL `http://ntopng_host:3000/lua/get_ts.lua?schema=host:traffic&query=ifid:2,host=192.168.1.1`

How to Further Improve Influx

- Change the application design in LibInflux + InfluxDB (that uses libInflux and adds an ingest layer).
- Let local applications push data through LibInflux using a C/C++ API:
  - No need of an intermediate text-based format.
  - Less security headaches as there is no need to perform DB authentication or data encryption when pushing data to the database.
  - Less overhead due to HTTP, no data loss when using UDP.
- Some words about Chronograf (biased by Grafana)…

Final Remarks

- InfluxDB is definitively a step forward with respect to RRD.
- The changes we had to do in our code to handle it were mostly due to the fact that ntopng was not sitting on top of a generic timeseries layer as it assumed RRD was living underneath.
- Removing the inability to monitor large networks with many counters and with low granularity is compulsory, and InfluxDB is definitively adequate for this task.