RIPE NCC
RIPE NETWORK COORDINATION CENTRE

Webinar: RIPE Atlas Tools hackathon

Vesna Manojlovic
Massimo Candela
Christopher Amin
Christian Teuschel

Amsterdam | 6 November 2015

Goals for the hackathon

- HAVE FUN!!!!11
- Create tools for operators
- Give the code to the community
- Meet colleagues & create new connections

Goals for this webinar

- Learn about RIPE Atlas
- Discuss hackathon challenges
- Get your questions answered by developers

Prerequisites

- We assume you have already used RIPE Atlas
- Do you have a RIPE NCC Access account?
  - **If not - create one here: ripe.net/register**
- Want to start more measurements? We'll give you extra credits…

Overview

- Introduction to RIPE Atlas
- Data structures
- APIs
- Available tools
- Other related projects
- Challenges
- Previous RIPE Atlas hackathon experience
- Questions & answers

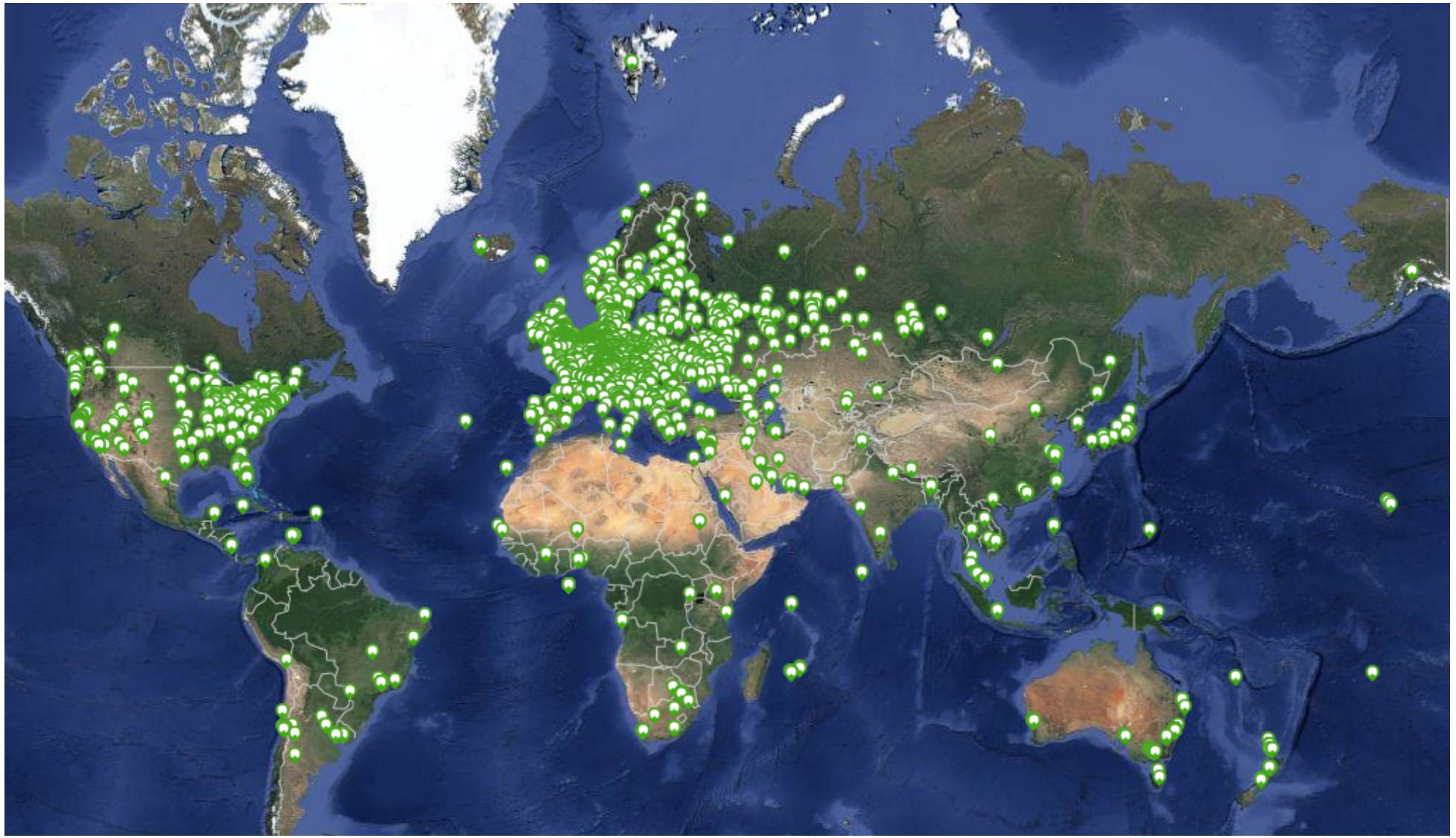
More material

- Hackathon info
  - https://labs.ripe.net/Members/becha/join-the-ripe-atlas-tools-hackathon
  - https://atlas.ripe.net/hackathon/tools-2015/#!attendee-information
- Basic RIPE Atlas course: http://www.ripe.net/lir-services/training/courses/tailor-made-workshops/#tools
- Advanced course - webinar material: https://www.ripe.net/support/training/learn-online/webinars/advanced-ripe-atlas-usage-webinar

Introduction to RIPE Atlas

- RIPE Atlas is a global active measurements platform
- Goal: view Internet reachability
- Probes hosted by volunteers
- Data publicly available

RIPE Atlas coverage



RIPE Atlas results

- Ongoing global measurements towards root nameservers
  - Visualised as Internet traffic maps
- Ongoing regional measurements towards “anchors”
- Users can run customised measurements
  - ping, traceroute, DNS, SSL/TLS, NTP & http

Numbers for October 2015

- 8,900+ probes connected

- 5,000+ active users in the last quarter
- 2,500+ results collected per second
- 35,000+ customised measurements weekly

Data structures

JSON data


```
[{“af":6,"avg" 61.32,

”dst_addr":"2a00:1450:4004:802::1014",“dst_name":"www.google.com",

“dup":0,

“from":"2001:8a0:7f00:b201:220:4aff:fec5:5b5b",

“fw":4660,“lts":411,

“max”:62.148,"min":60.372,

“msm_id":1004005,"msm_name":"Ping",

"prb_id":722,"proto":"ICMP","rcvd":10,

“result":[{"rtt":62.148},{"rtt":61.437},{"rtt":61.444},{"rtt":61.448},{"rtt":61.794},
{"rtt":61.533},{"rtt":60.372},{"rtt":60.373},{"rtt":61.384},{"rtt":61.267}],

“sent":10,"size"64,

“src_addr":"2001:8a0:7f00:b201:220:4aff:fec5:5b5b",

"step":240,"timestamp":1410220847,"ttl":54,"type":"ping"},
```


Destination (IP & name)

Source (probe public IP address)

Packet loss: difference between sent & received!

Reference (msm ID)

Examples of result analysis code

Python:

Parse json and find total avg:

```
import json
f = open(“measurement.json”,”r”)
measurements = json.load(f)
for m in measurements:
    for r in m[“result”]:
        rtt = r[“rtt”]
ifrtt >60: i += 1
```

i must be > than 14563.

Javascript:

```
<script>
var dataAPIUrl = "https://atlas.ripe.net/api/v1/
measurement/1004005/result/?start=1410220800";
jQuery.ajax({

url: dataAPIUrl, error: function() {

alert("error"); },

success: function( response ) { var i = 0;

for ( var i = 0, n = response.length; i < n; i++) { var
measurement = response[i];

for ( var j = 0, m = measurement.result.length; j < m; j++) {
var rtt = measurement.result[j].rtt;
console.log(rtt);
if (rtt > 60)

i++; }

}
jQuery("p").html("The RTT has been above 60ms for " + i
+ " times");

},

dataType: "jsonp" });

</script>
```

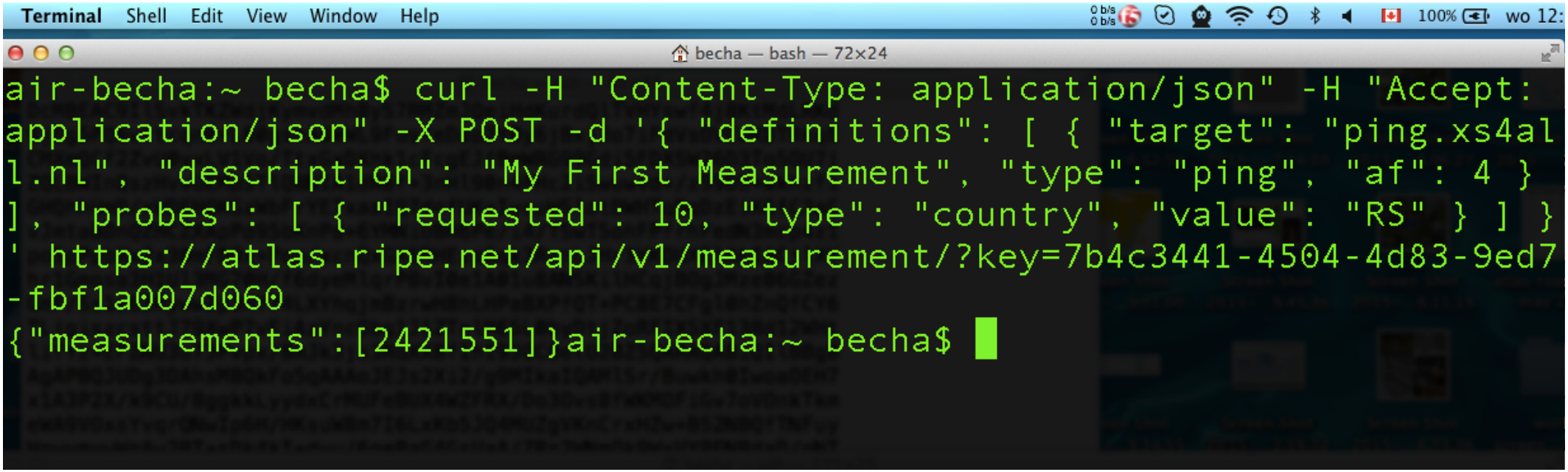
APIs

- API documentation:
  - https://atlas.ripe.net/docs/measurement-creation-api/
  - https://atlas.ripe.net/docs/rest/
  - https://atlas.ripe.net/doc/credits
  - https://atlas.ripe.net/doc/udm
  - https://atlas.ripe.net/keys/
  - https://atlas.ripe.net/docs/keys2/
  - https://atlas.ripe.net/docs/status-checks/
  - https://stat.ripe.net/docs/data_api

Using API to create measurement

- Example:

```
$ curl -H "Content-Type: application/json" -H "Accept: application/json" -X POST
-d '{ "definitions": [ { "target": "ping.xs4all.nl", "description": "My First API
Measurement", "type": "ping", "af": 4 } ], "probes": [ { "requested": 10, "type":
"country", "value": "RS" } ] }' https://atlas.ripe.net/api/v1/measurement/?
key=YOUR_API_KEY
```



Existing tools & use cases

Links to the existing tools

- Contributed by the community:
  - https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/README.md
- CLI toolset https://github.com/RIPE-NCC/ripe-atlas-tools/
- Sagan https://github.com/RIPE-NCC/ripe.atlas.sagan
- Cousteau https://github.com/RIPE-NCC/ripe-atlas-cousteau
- Eyeballtrace
  - https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/emileaben/eyeballtrace

Why CLI RIPE Atlas tools

- Access RIPE Atlas from the terminal / shell console
- Quick & dirty shortcuts for network troubleshooting

- FLOSS (open source) tools
  - written & maintained by RIPE NCC
  - open for contributions by the community

- Before you can use the toolset
  - download the tools, install, configure

Use cases for CLI tools

1. create a measurement
2. generate a simple report abut ongoing measurement
3. look at the results
4. collect results from the ongoing measurement (streaming)

Examples of ping

- Geo-specific from 20 probes from Canada:

```
$ ripe-atlas measure ping --target example.com
--probes 20 --from-country ca
```

- 20 Canadian probes that definitely support IPv6:

```
$ ripe-atlas measure ping --target example.com \
-- probes 20 --from-country ca --include-tag \
system-ipv6-works
```

- Create a recurring measurement:

```
$ ripe-atlas measure ping \ —target
example.com --interval 3600
```

CLI toolset - links

- Source:
  - https://github.com/RIPE-NCC/ripe-atlas-tools/
- Documentation:
  - https://ripe-atlas-tools.readthedocs.org/
- How to contribute:
  - https://github.com/RIPE-NCC/ripe-atlas-tools/blob/master/CONTRIBUTING.rst

Integration with Network Monitoring Systems

- Nagios & Icinga can receive input from RIPE Atlas via the API
  - based on the ping measurement from up to 500 probes
- Status checks work via RIPE Atlas' RESTful API
  - https://atlas.ripe.net/api/v1/status-checks/MEASUREMENT_ID/
- You define the alert parameters
- Icinga configuration code using the built-in “check_http” plugin
  - https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/scripts_for_nagios_icinga_alerts
- Documentation: https://atlas.ripe.net/docs/status-checks/

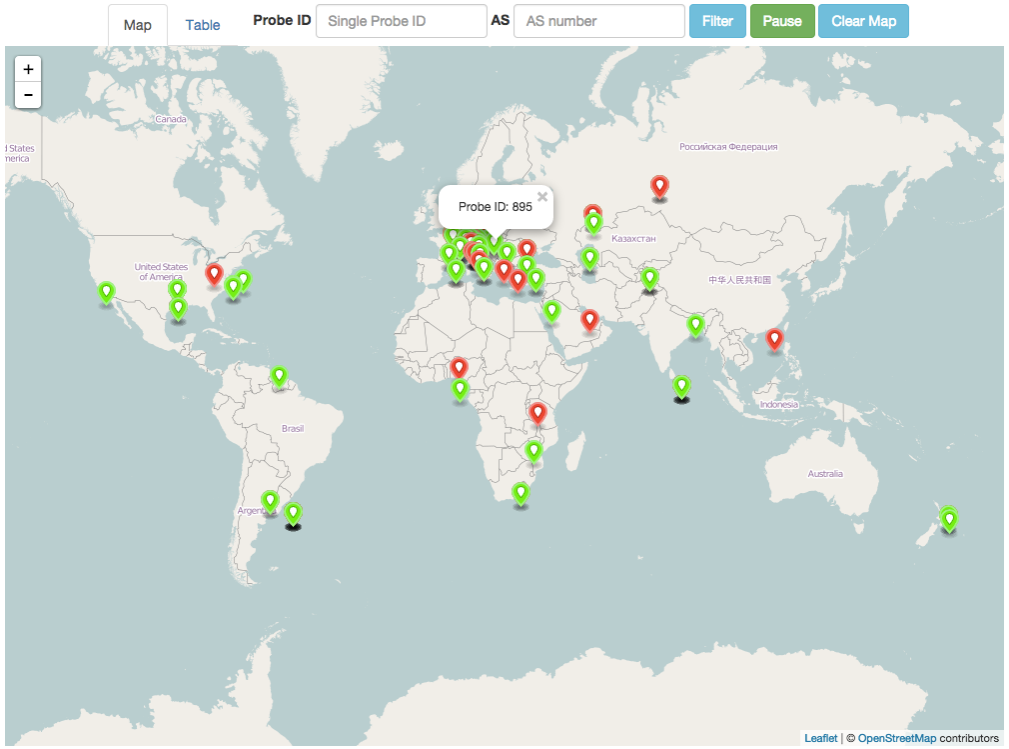
Real-time performance monitoring

- RIPE Atlas streaming is an architecture that allows users to receive the measurement results as soon as they are sent by the probes
  - In real time
  - Publish/subscribe through web sockets
- Documentation:
  - https://atlas.ripe.net/docs/result-streaming/
  - https://labs.ripe.net/Members/suzanne_taylor_muzzin/data-streaming-in-ripe-atlas

Monitoring outages

- Probe connection status events are also published as streaming

- https://labs.ripe.net/Members/kistel/the-ams-ix-outage-as-seen-with-ripe-atlas



https://labs.ripe.net/Members/andreas_strikos/amsterdam-power-outage-as-seen-by-ripe-atlas

Other projects

Crowdsourcing IP geolocation

- OpenIPMap
- Goal: improve infrastructure geolocation
- Method:
  - import & decode existing geoloc data
  - visualize RIPE Atlas traceroutes
  - provide interface to network operators to contribute data
  - share collected data
- https://marmot.ripe.net/openipmap/
  - https://github.com/RIPE-Atlas-Community/openipmap

Measuring Impact of IXPs on Keeping Traffic Local

- https://github.com/emileaben/ixp-country-jedi
- Benefits:
  - Routing and traffic optimisation
  - Shows how IXPs help keep traffic local and regional
  - Comparing IPv4 and IPv6 paths
  - How much traffic stays within the country? Where do the paths go?
  - Comparing countries with each other
  - Use case for improving data quality

IXP-Country-Jedi Method

- traceroute measurements using RIPE Atlas probes
- Steps:
  - Identify ASNs in the country using RIPEstat
  - Identify IXPs and IXP LANs using PeeringDB
  - Construct mesh: from all (*) country's probes to each other

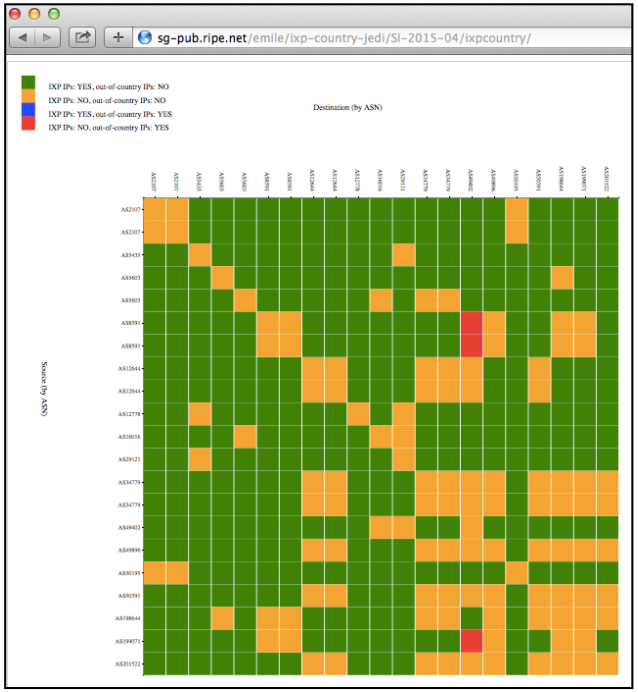
*Maximum of two probes per ANS and only "public" probes with "good" geolocation

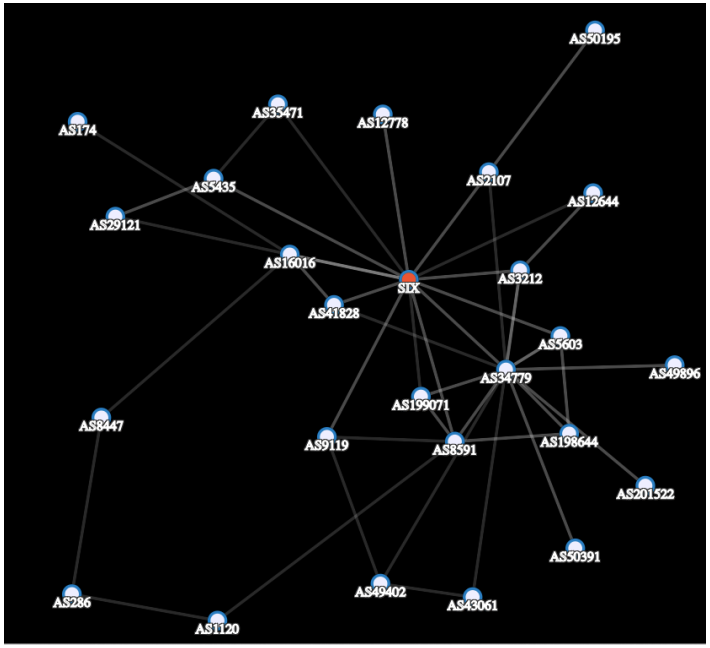
- Hops geolocated using "OpenIPMap" database

- Examples:
  - https://labs.ripe.net/Members/emileaben/measuring-ixps-with-ripe-atlas
  - https://labs.ripe.net/Members/emileaben/measuring-countries-and-ixps-in-the-see-region
- Latest data:
  - http://sg-pub.ripe.net/emile/ixp-country-jedi/latest/

Paths going via an IXP?

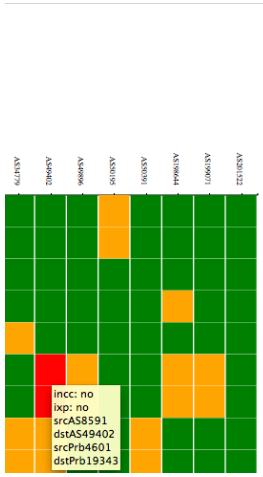
http://sg-pub.ripe.net/emile/ixp-country-jedi/SI-2015-04/ixpcountry/





Interactive diagnostic tool

- Green: “good”, as far as we can see it
  - Not a judgment, only one way of visualising data
- Red or blue: path is going out of country
  - If this is a surprise: talk to your upstream(s)
- Yellow: path is not going via a local IXP
  - If this is undesired: make a new peering agreement




```
## msm_id:1962254 prb_id:4601 dst:193.169.48.40 ts:2015-04-16 09:01:06 -00:00
1 (AS8591) maribor10-ge-2-20-v987.amis.net [1.593, 1.602, 2.292] |Maribor,Maribor,SI
2 (AS8591) mx-mbl-te-1-2-0.amis.net [1.619, 1.697, 1.944] ||
3 (AS8591) mx-ljl-te-2-3-1.amis.net [3.599, 3.865, 5.148] ||
4 (AS8591) mx-zgl-xe-2-0-1.amis.net [5.568, 5.576, 5.69] ||
5 () 75.64-127.15.192.193.in-addr.arpa [5.955, 5.98, 5.985] |Zagreb,Grad Zagreb,HR|
6 (AS9119) 212.13.240.249 [5.778, 5.83, 5.935] ||
7 (AS9119) 212.103.133.4 [7.099, 7.84, 7.926] ||
8 (AS9119) 212.13.240.62 [6.597, 7.674, 7.696] ||
9 (AS9119) hsl.gw0.hsl.eu [5.833, 6.079, 6.368] ||
10 (AS49402) ntp.hsl.eu [6.657, 7.273, 8.155] ||
11 (AS49402) 193.169.48.40 [6.661, 6.691, 6.872] ||
```

http://sg-pub.ripe.net/emile/ixp-country-jedi/SI-2015-04/ixpcountry/

Hackathon Challenges

Hackathon challenges

- Modify existing tools in order to improve them
- Create a new stand-alone tool
  (a working prototype)
- Make code to be integrated with RIPE Atlas
- Yet another thing that does not fit categorization?!

Criteria for “winning”

- Application and usefulness for the operators
- Solving one or more use cases
- Make code flexible & reusable
- Give source code back to the community
- Using more than one data sets

Requirements for integration into the RIPE Atlas system

- Code must:
  - Be written in Python if you require server-side development
  - Be written in JavaScript (or an established JS framework) for client-side development.
- Code must not:
  - Make use of proprietary technologies like Flash or Silverlight
  - Be precompiled
  - Be obfuscated

Some ideas for use cases

- - SSL Watch
- - Comparing IPv6 and IPv4 reachability
- - Integrating RIPE Atlas with IXP manager
  - https://github.com/inex/IXP-Manager/wiki
- - Correlating BGP & Atlas data
  - https://www.ripe.net/data-tools/stats/ris/ris-raw-data
  - https://stat.ripe.net/data-sources
- Play with HTTP measurements to anchors

Other use cases

- Helping ISPs monitor the reachability of the websites most visited by their users

- Hosting companies with multiple POPs around the world that need to be **alerted** of high latency from a certain part of the world

- The integration of existing web front-end tools (based on RIPE Atlas APIs) into a network operator's "dashboard" view

What I'll be working on… do join :)

- Improve multilingual documentation
  - translate exiting documentation to other languages
  - add info about RIPE Atlas in other languages to GitHub
  - https://github.com/RIPE-Atlas-Community/multi-lingual-docs

- Improve Wikipedia page
  - https://en.wikipedia.org/wiki/RIPE_Atlas
  - translate to other languages & make new wiki pages

Previous hackathon

DataViz hackathon, March 2015

- Summary of results:
  - More than 70 applications for participation
  - Present: 24 participants (& one no-show), 6 jury members and 11 support staff (coincidentally: 42!)
  - Six out of 25 participants were female (24%)
  - Three jury members were female (50%!)
  - Ten final projects were presented
  - Four projects were awarded prizes
  - 14 free & open source software projects were published on GitHub

DataViz hackathon, March 2015

- Links:
  - https://labs.ripe.net/Members/becha/ripe-atlas-hackathon-results
  - Info - including licences https://atlas.ripe.net/hackathon/2015/#!attendee-information
  - Code on GitHub https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/README.md
  - Presentations: https://labs.ripe.net/Members/becha/ripe-atlas-hackathon-presentations

Contact us

- https://atlas.ripe.net
- Users' mailing list: ripe-atlas@ripe.net
- Articles and updates: https://labs.ripe.net/atlas
- Questions and bugs: atlas@ripe.net
- Twitter: @RIPE_Atlas and #RIPEAtlas
- HACKATHON specific:
  - IRC #ripeatlas at Freenode
  - Vesna's phone number: +31 6 21 25 81 91

Extra: Existing RIPE Atlas Visualizations

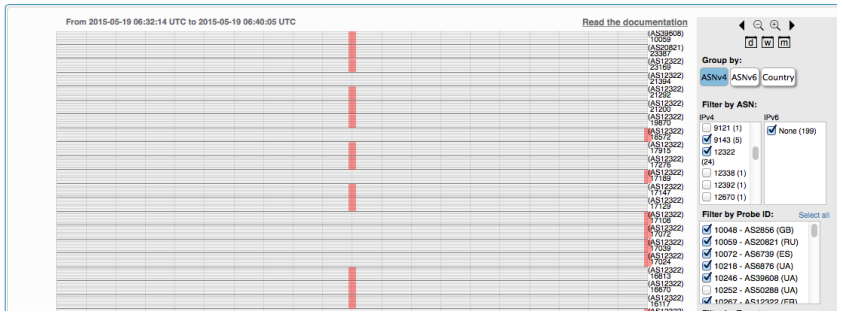
Available visualisations: ping

- List of probes: sortable by RTT

Probe	ASN (v4)	ASN (v6)	Time	RTT
6019	3333	3333	2015-05-19 09:23	1.157
6069	59469	59469	2015-05-19 09:23	15.253
6111	198068	198068	2015-05-19 09:23	37.760
6112	197216	197216	2015-05-19 09:23	35.494
10008	3851		2015-05-19 09:23	24.664
10218	6876		2015-05-19 09:23	37.952
10246	39608		2015-05-19 09:23	36.313
10252	50288		2015-05-19 09:23	62.441
10267	12322		2015-05-19 09:23	31.498
10296	51214		2015-05-19 09:23	Unreachable

- Map: colour-coded by RTT

- Seismograph: stacked multiple pings with packet loss



Internet maps (DNS rootnameservers)

DNS Root Instances

Shows, for each probe, which root DNS server instance the probe ends up querying, when they ask a particular root server. In other words, it shows the "gravitational radius" for root DNS server instances.

Comparative DNS Root RTT

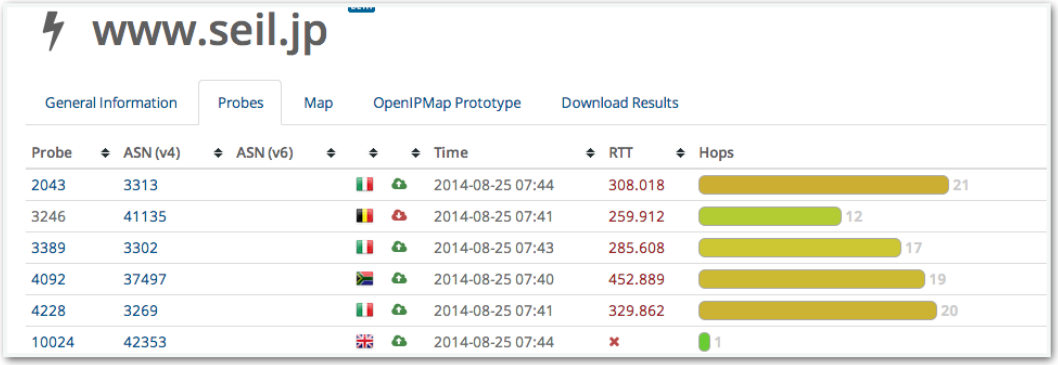
Shows a comparison of response time for DNS SOA queries to all the root DNS servers. For each probe, a marker shows the "best" root server with colour identifying the related minimum response time.

Root Server Performance

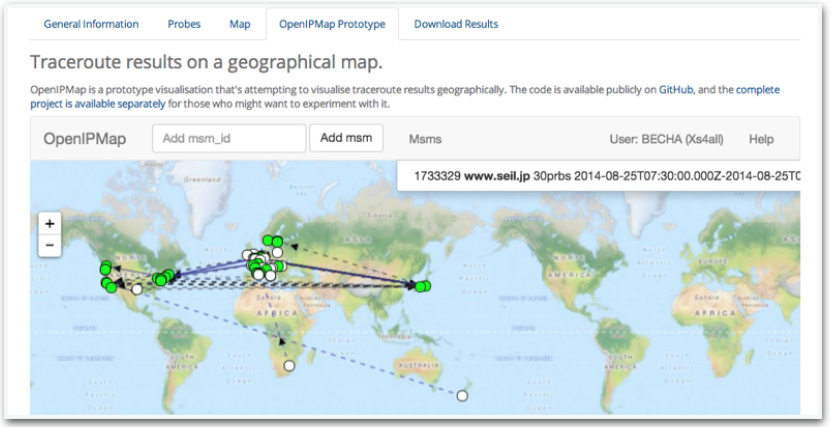
This map shows the reply time to the SOA query of a particular root DNS server, over the selected transport protocol (UDP, TCP or comparison of the two) for each probe.

Available visualisations: traceroute

- List of probes, colour-coded number of hops



- Map
- Traceroute paths map, geolocation using OpenIPMap: https://github.com/RIPE-Atlas-Community/openipmap



Available visualisations: DNS

- Map, colour-coded response time or diversity

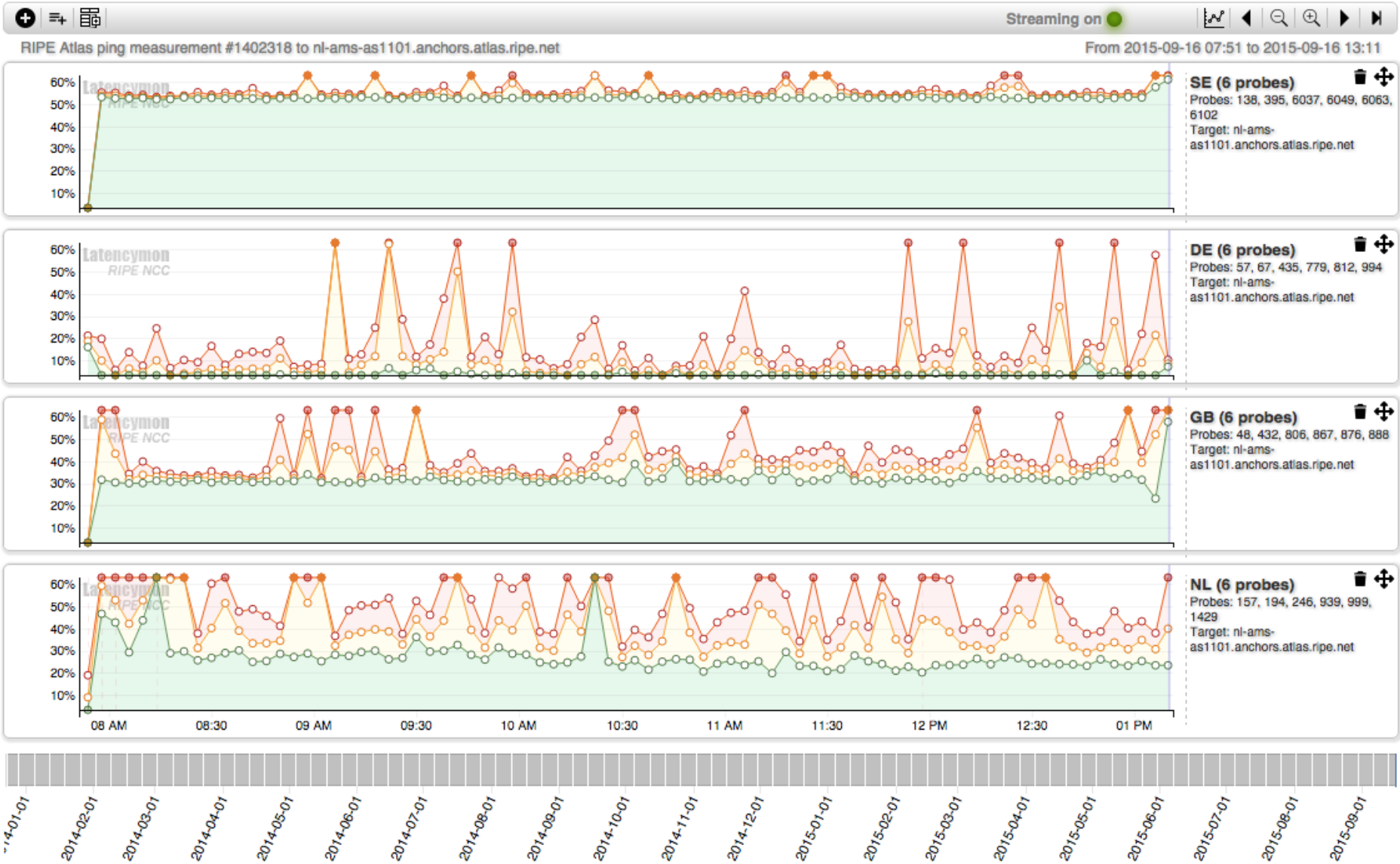
- List of probes, sortable by response time

DNS measurement to ns1.opteamax.de

General Information | Probes | Map | Download Results | Modification Log

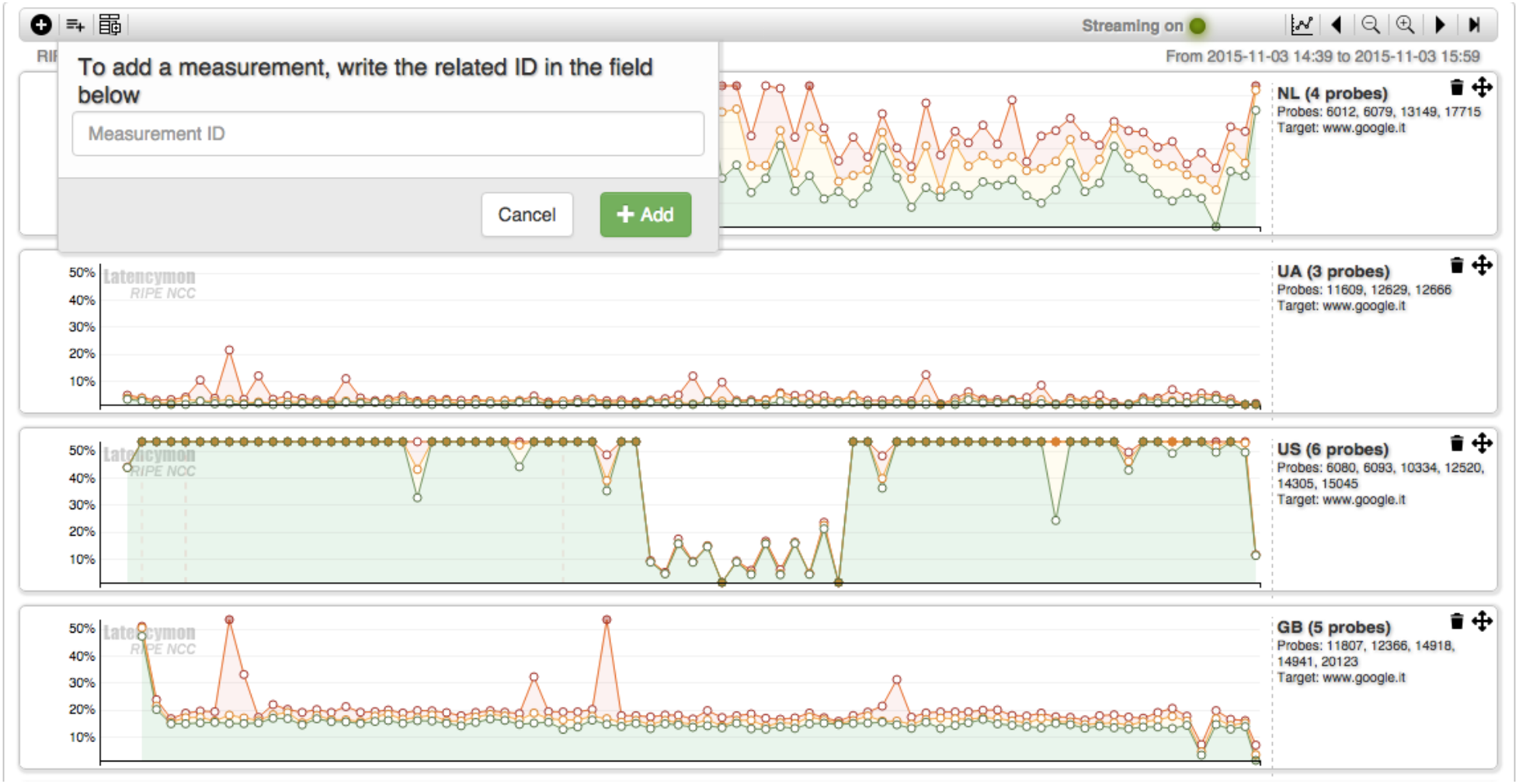
Probe	ASN (v4)	ASN (v6)		Time	Name	Response Time
17840	6327			2015-05-19 09:38	null	362.009
18035	43030			2015-05-19 09:50	null	347.39
18129	327805			2015-05-19 09:49	null	207.743
15844	32098			2015-05-19 09:48	null	184.237
17857	852			2015-05-19 09:37	null	177.694
19894	6327			2015-05-19 09:36	null	168.689
19204	21513			2015-05-19 09:50	null	141.199
15922	30036			2015-05-19 09:47	null	133.309

NEWEST DataViz tool: LatencyMON



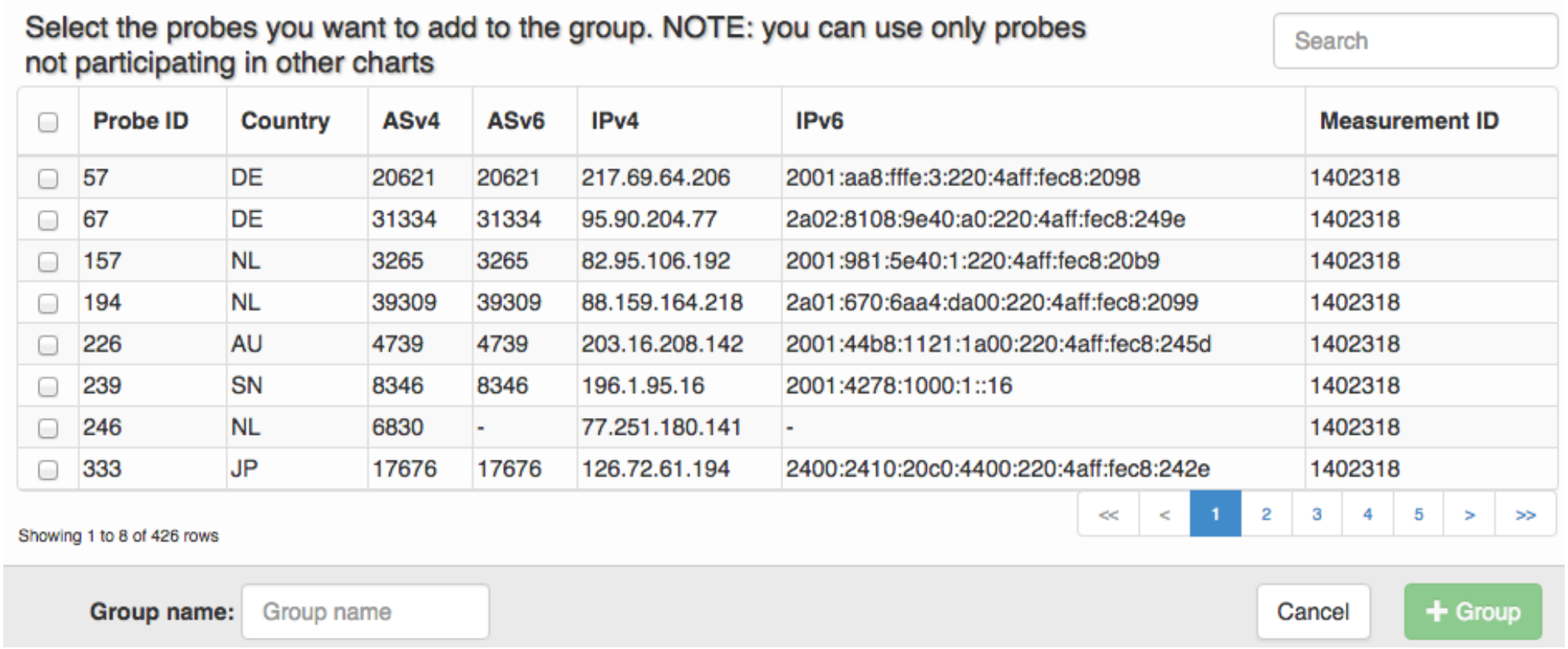
Adding multiple measurements

- If multiple targets are involved, the auto-grouping will be by target.



Adding a group of probes

- You can search for any probe attribute
- You can specify a group name



LatencyMon docs & source code

- https://atlas.ripe.net/docs/tools-latencymon/
- https://labs.ripe.net/Members/massimo_candela/new-ripe-atlas-tool-latencymon
- https://github.com/MaxCam/viz-atlas-latencymon