RIPE NCC
RIPE NETWORK COORDINATION CENTRE

RIPE Atlas Tutorial Advanced

Michela Galante

LACNIC 25 | 2 May 2016

Goals

- Learn how to use RIPE Atlas measurements for network monitoring and troubleshooting
- Get answers to your questions

Overview

- Real-time performance monitoring
- IXP Country Jedi
- Additional Topics
  - Command Line Interface Tools

Introduction to RIPE Atlas

Global Active Measurement Platform

- Goal is to view Internet reachability
- Probes hosted by volunteers
- Measurements performed towards root name servers
  - Visualised as Internet traffic maps
- Users can also run customised measurements
  - Ping, traceroute, DNS, TLS/SSL, NTP and (limited) HTTP
- Data publicly available

RIPE Atlas in Numbers

- Countries: 181
- Originating ASNs:
  - 3,333 (IPv4) = 6,33% coverage
  - 1,212 (IPv6) = 11,22% coverage
- 9,400 active probes
  - Only 230+ probes in LAC region
  - Only one in Cuba
- Active users: 10,000 in 2015

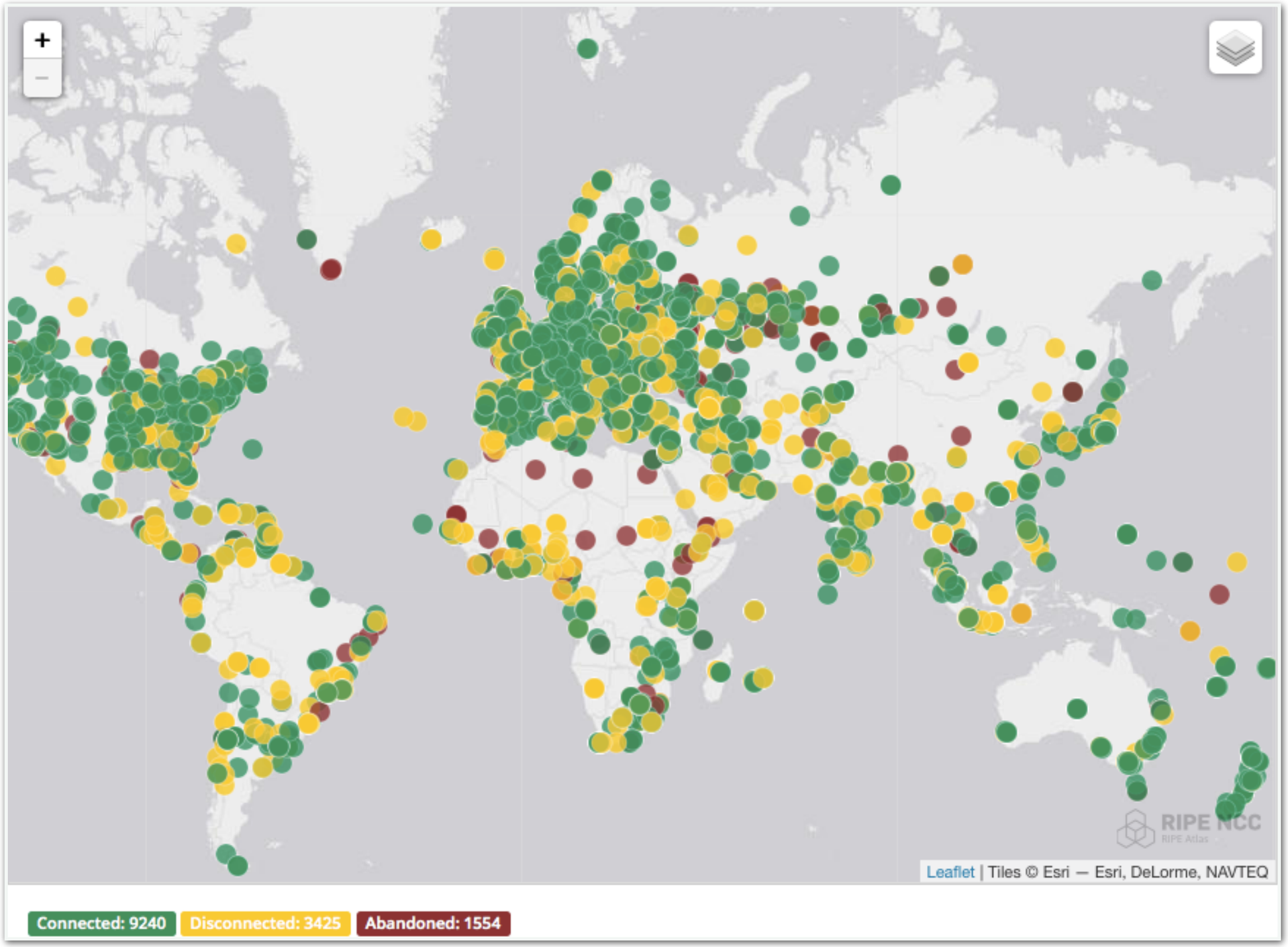
Country	Probes
Brazil	60
Argentina	35
Chile	27
Uruguay	17
Dominican Rep.	12
Mexico	9
Venezuela	8
Costa Rica	8
Peru	7
Ecuador	6

RIPE Atlas Coverage in LAC Region

Regional overview of probes density

- Courtesy of LACNIC: https://simon.lacnic.net/simon/atlas/

RIPE Atlas Global Coverage



RIPE Atlas Anchors

- 188 RIPE Atlas Anchors

- Three sponsored by LACNIC and hosted by NIC Chile, NIC Costa Rica and RIU (Argentina)
- One hosted by LACNIC (Uruguay)
- One hosted by NIC Brazil
- One hosted by Optical Technologies (Peru)

Monitoring

See Your Network From the Outside

- Integrate “status checks” with existing monitoring tools (such as Icinga)
- Developed by community: RIPE Atlas Monitor
- Uses real-time data streaming
  - Server monitoring
  - Detecting and visualising outages
  - Filtering and re-using measurement results

Steps for integration

1. Create a RIPE Atlas ping measurement
2. Go to “status checks” URL (RESTful API call)
   - https://atlas.ripe.net/api/v1/status-checks/2340408/?median_rtt_threshold=10
   - https://atlas.ripe.net/docs/status-checks/
3. Add your alerts in Nagios or Icinga
   - Make use of the built-in “check_http” plugin
   - https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/scripts_for_nagios_icinga_alerts

Versatile “RIPE Atlas Monitor”

- Pier Carlo Chiodi’s work (@pierky)
  - https://github.com/pierky/ripe-atlas-monitor
  - https://ripe-atlas-monitor.readthedocs.org/
- Additional use cases: traceroute analysis, hostname resolution, AS path detection, verifying TLS connections…
- Recently published on RIPE Labs
  - https://labs.ripe.net/Members/pier_carlo_chiodi/ripe-atlas-monitor

RIPE Atlas Streaming

- 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 WebSockets
- There are three types of data:
  - Measurement results
  - Probe connection status events
  - Measurement metadata

RIPE Atlas Streaming (cont'd)

- Visualising network outages
  - http://sg-pub.ripe.net/demo-area/atlas-stream/conn/
- Real-time server and performance monitoring
- Filtering and reusing measurement results by target, source, and more
- Documentation:
  - https://atlas.ripe.net/docs/result-streaming/

RIPE Atlas Result Streams

Stream type: result

When stream_type is set to "result", the client will receive measurement results.

```
<script src="https://atlas-stream.ripe.net/socket.io.js"></script>
<script>

    // Create a connection (it can be also http on port 80)
    var socket = io("https://atlas-stream.ripe.net:443", { path : "/stream/socket.io" });

    // Subscribe to results coming from all the probes involved in the measurement 1791207
    socket.emit("atlas_subscribe", { stream_type: "result", msm: 1791207 });

    // Declare a callback to be executed when a measurement result is received
    socket.on("atlas_result", function(result){
        console.log("I received ", result);
    });

</script>
```

Stream type: probestatus

When stream_type is set to "probestatus", the client will receive connection and disconnection events of probes.

```
<script src="https://atlas-stream.ripe.net/socket.io.js"></script>
<script>

    // Create a connection
    var socket = io("https://atlas-stream.ripe.net:443", { path : "/stream/socket.io" });

    // Subscribe to the connection events of the probe 22527
    socket.emit("atlas_subscribe", { stream_type: "probestatus", prb: 22527 });

    // Declare a callback to be executed when a probe connection event is received
    socket.on("atlas_probestatus", function(status){
        console.log("I received ", status);
    });

</script>
```

Exercise

Use Streaming API

Monitoring Server Reachability (1)

- Scenario: customers are complaining that it occasionally takes a long time to reach your service or server
- Action: ping your server from 500 probes
  - Decide what is acceptable latency threshold to apply
  - Notice and react when you start receiving samples
- Task: use the ping measurement ID 2340408
  - Choose your threshold (e.g. greater than 30ms)
  - Impose the threshold on “min” (the minimum result of the three ping attempts)

Steps

- Go to:
  - http://atlas.ripe.net/webinar/streaming01.html
- Open the development console
- Wait for results to arrive
- Save the HTML file locally and edit the code

Page Source

view-source:sg-pub.ripe.n

view-source:sg-pub.ripe.net/webinar/streaming01.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Streaming exercise 01</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>Current maximum RTT: <b><span id="output">nothing yet</span></b></div>
        <div>Open the source code to see how it works. Create your tool/visualisation with the
RIPE Atlas streaming!</div>
    </body>

    <script src="https://stat.ripe.net/widgets/lib/js/jquery/jquery-1.11.2.min.js"></script>


    <!-- The following file is needed for the streaming -->
    <script src="https://atlas-stream.ripe.net/socket.io.js"></script>
    <script>
        var $outputDiv = $("#output");

        // Create a connection
        var socket = io("https://atlas-stream.ripe.net", { path : "/stream/socket.io" });

        // Declare a callback to be executed when a measurement result is received
        socket.on("atlas_result", function(result){

            console.log("I received ", result); // Print the result in the console

            if (result.hasOwnProperty("max")) {
                $outputDiv.html(result["max"]); // Print the result in the html page
            }

        });

        // Subscribe to results coming from all the probes involved in the measurement 2340408
        socket.emit("atlas_subscribe", { stream_type: "result", msm: 2340408 });

    </script>
</html>
```

Example of Results

Elements Network Sources Timeline Profiles Resources Audits Console AngularJS

<top frame> Preserve log

Filter Regex All Errors Warnings Info Logs Debug Hide network messages

```
XHR finished loading: GET "http://atlas-stream.ripe.net/stream/socket.io/?EIO=2&transport=polling&t=1431095373684-0".
XHR finished loading: GET "http://atlas-stream.ripe.net/stream/socket.io/?EIO=2&transport=polling&t=1431095373739-1&sid=eB0kM7zfWFT2c-ScAAaH".
I received ▶ Object {af: 4, prb_id: 16669, result: Array[3], ttl: 42, avg: 326.841…}
I received ▶ Object {af: 4, prb_id: 16669, result: Array[3], ttl: 42, avg: 325.7933333333…}
I received ▶ Object {af: 4, prb_id: 16669, result: Array[3], ttl: 42, avg: 326.048…}
I received ▶ Object {af: 4, prb_id: 16669, result: Array[3], ttl: 42, avg: 327.3253333333…}
I received ▶ Object {af: 4, prb_id: 15965, result: Array[3], ttl: 45, avg: 47.6313333333…}
I received ▶ Object {af: 4, prb_id: 15965, result: Array[3], ttl: 45, avg: 47.6996666667…}
I received ▶ Object {af: 4, prb_id: 15965, result: Array[3], ttl: 45, avg: 47.4816666667…}
I received ▶ Object {af: 4, prb_id: 19566, result: Array[3], ttl: 40, avg: 47.054…}
I received ▶ Object {af: 4, prb_id: 19566, result: Array[3], ttl: 40, avg: 47.8626666667…}
I received ▶ Object {af: 4, prb_id: 19566, result: Array[3], ttl: 40, avg: 47.5946666667…}
I received ▶ Object {af: 4, prb_id: 19566, result: Array[3], ttl: 40, avg: 47.5003333333…}
I received ▶ Object {af: 4, prb_id: 18311, result: Array[3], ttl: 49, avg: 32.577…}
I received ▶ Object {af: 4, prb_id: 18311, result: Array[3], ttl: 49, avg: 34.0843333333…}
I received ▶ Object {af: 4, prb_id: 18311, result: Array[3], ttl: 49, avg: 32.7513333333…}
I received ▶ Object {af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 182.4463333333…}
I received ▶ Object {af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 193.9953333333…}
I received ▶ Object {af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 182.2913333333…}
I received ▶ Object {af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 191.6103333333…}
I received ▶ Object {af: 4, prb_id: 14918, result: Array[3], ttl: 49, avg: 34.817…}
I received ▶ Object {af: 4, prb_id: 14918, result: Array[3], ttl: 49, avg: 35.0093333333…}
I received ▶ Object {af: 4, prb_id: 14918, result: Array[3], ttl: 49, avg: 35.0843333333…}
I received ▶ Object {af: 4, prb_id: 20668, result: Array[3], ttl: 45, avg: 38.8846666667…}
I received ▶ Object {af: 4, prb_id: 20668, result: Array[3], ttl: 45, avg: 38.8626666667…}
I received ▶ Object {af: 4, prb_id: 20668, result: Array[3], ttl: 45, avg: 38.8806666667…}
I received ▶ Object {af: 4, prb_id: 6093, result: Array[3], ttl: 49, avg: 128.7273333333…}
I received ▶ Object {af: 4, prb_id: 6093, result: Array[3], ttl: 49, avg: 128.7373333333…}
I received ▶ Object {af: 4, prb_id: 6093, result: Array[3], ttl: 49, avg: 128.8883333333…}
```

Monitoring Server Reachability (2)

- Imagine the same situation, but you didn't schedule a measurement in advance
  - You don't have a measurement ID
- You want to get all the measurements reaching 193.0.10.197
- Now restrict the results to just include ping measurements

“IXP country Jedi”

Measuring Impact of IXPs on Keeping Traffic Local

Benefits (1)

- Operators
  - Routing and traffic optimisation
- IXP operators
  - Shows how IXPs help keep traffic local and regional
- IPv6 advocates
  - Comparing IPv4 and IPv6 paths

Benefits (2)

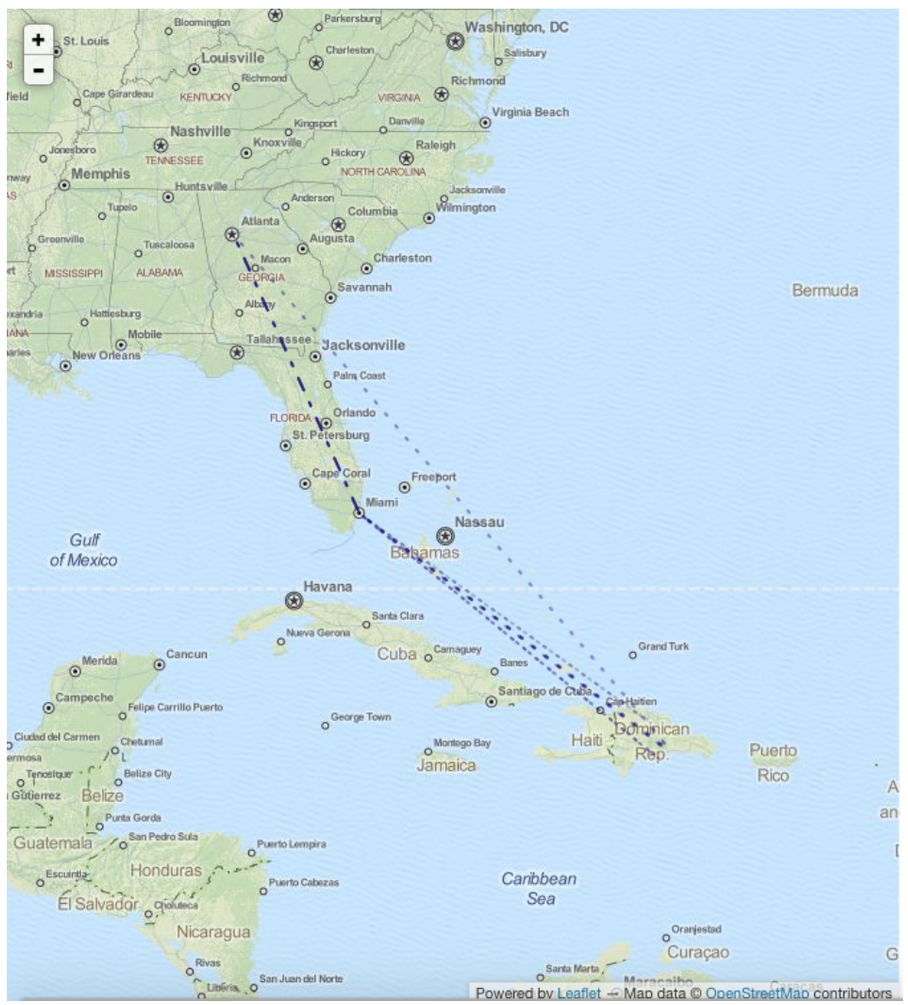
- Country level: regulators, politicians, cyber-security…
  - How much traffic stays within the country? Where do the paths go?
  - Comparing countries with each other
- RIPE Atlas community
  - More probes in more networks = higher quality of measurement data

Benefits (3)

- Geolocation data community
  - Use case for improving data quality
- 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
  - http://sg-pub.ripe.net/emile/ixp-country-jedi/CL+AR-2015-04/geopath/

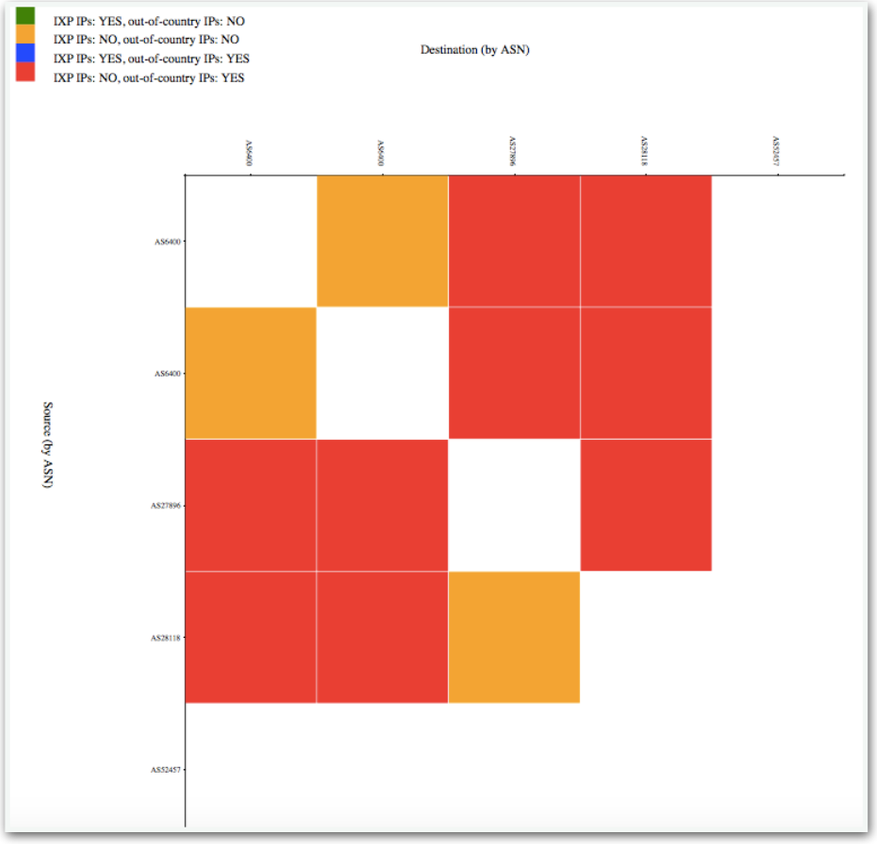
Do paths stay in country?

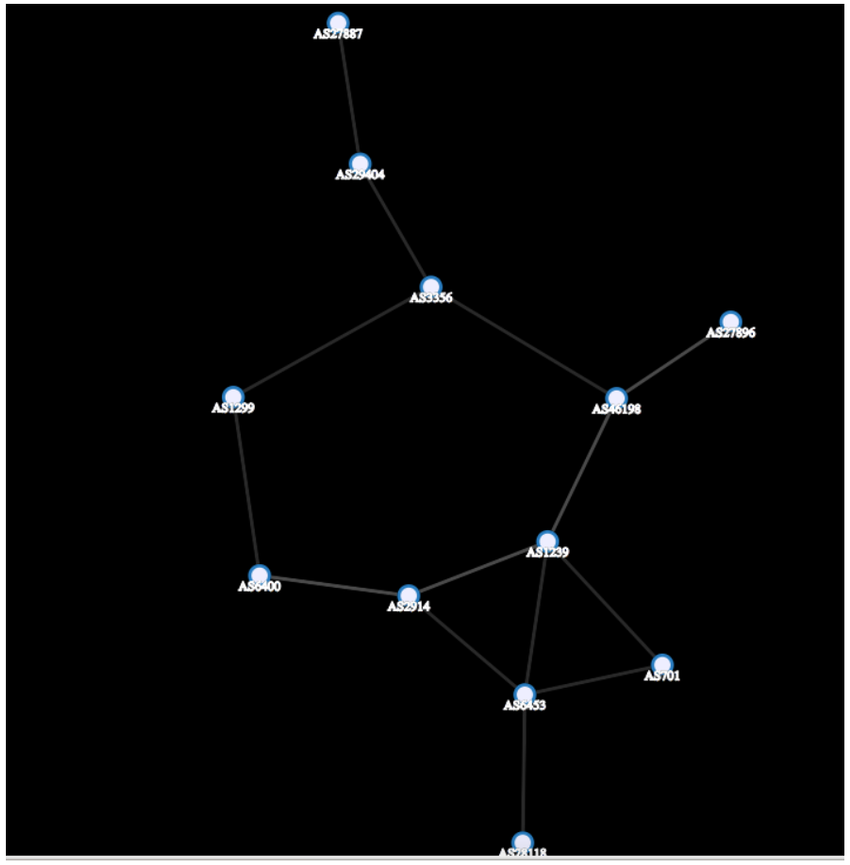
- Difference between IPv4 and IPv6 paths
  - http://sg-pub.ripe.net/emile/ixp-country-jedi/history/2016-04-01/DO/geopath



Do paths go via an IXP?

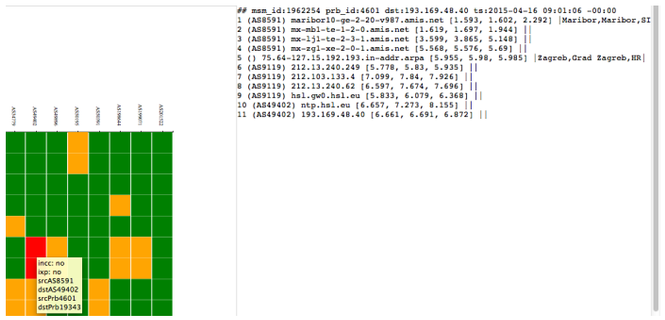
- http://sg-pub.ripe.net/emile/ixp-country-jedi/history/2016-04-01/DO/ixpcountry





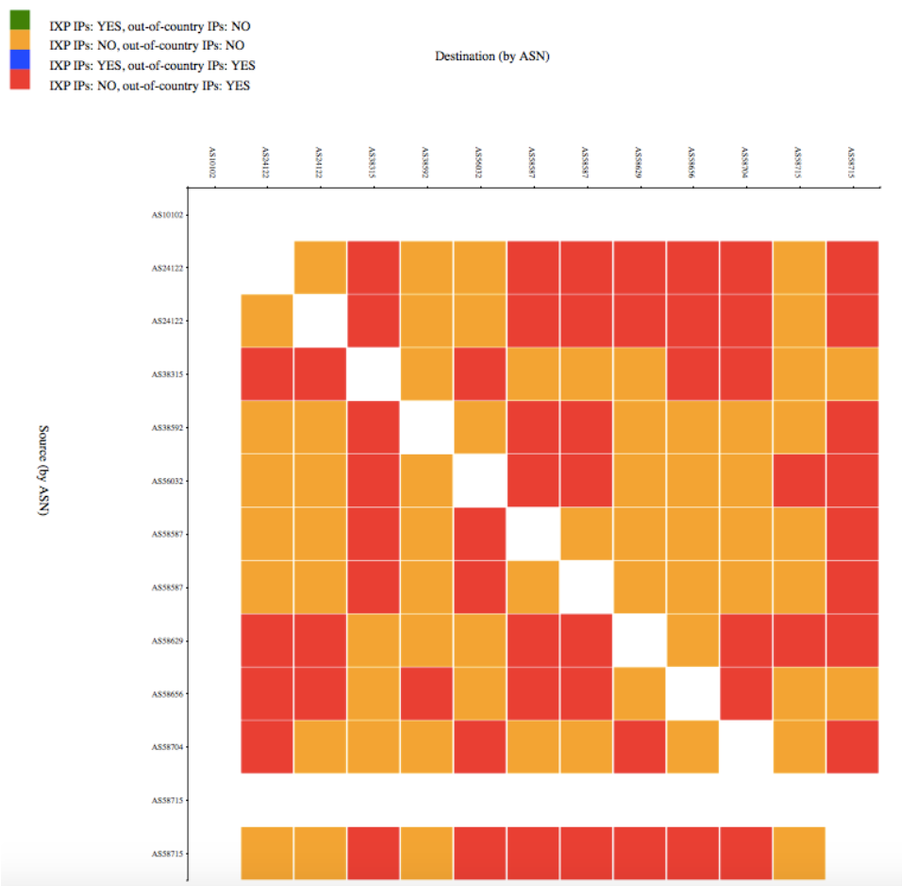
Interactive Diagnostic Tool

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

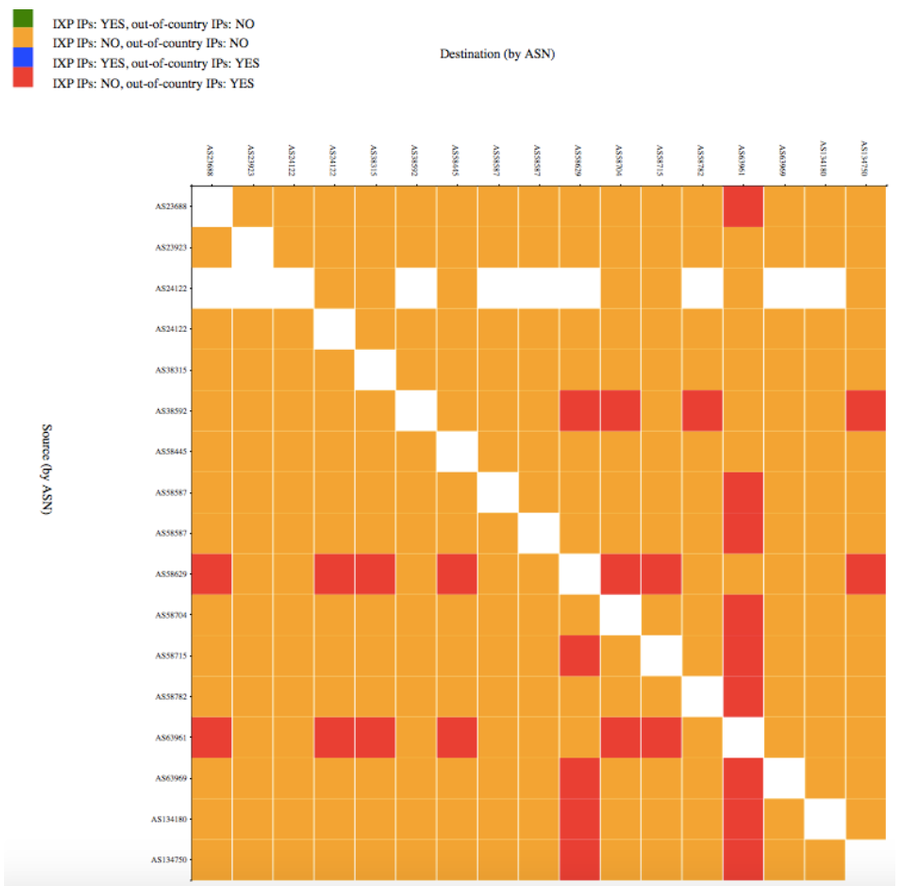


A Successful Use Case: Bangladesh

http://sg-pub.ripe.net/emile/ixp-country-jedi/history/2016-04-01/BD/ixpcountry/



Overview of Bangladesh in September 2015



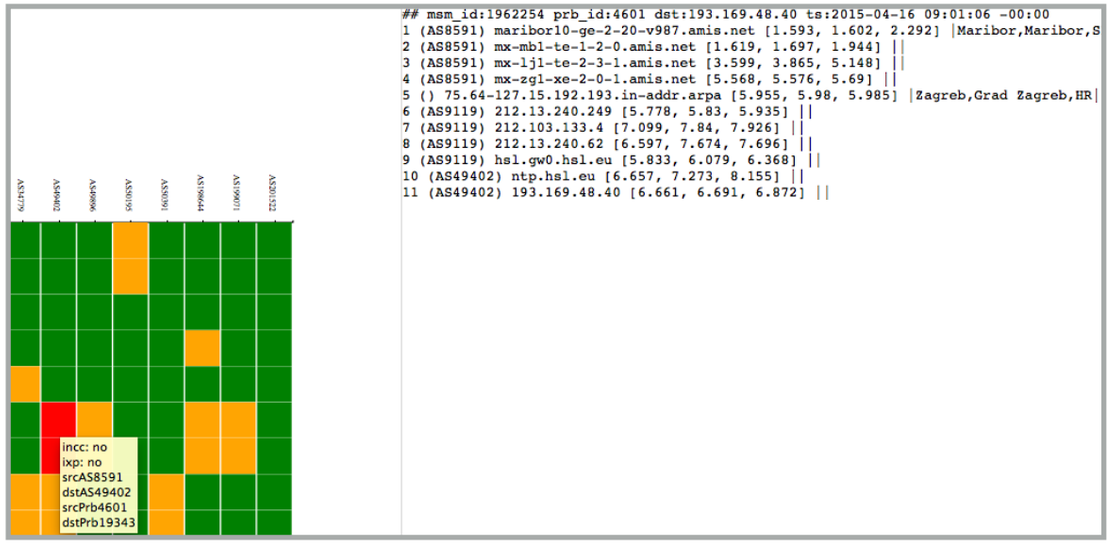
Overview of Bangladesh in April 2016

Method

- Traceroute measurements using RIPE Atlas
- 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 two probes per ASN and only "public" probes with good geolocation)
- Hops geolocated using OpenIPMap database

Homework

- Use this tool to find potentially suboptimal routing and optimise it
  - Find your ASN in the mesh (as source)
  - Find the other non-green ASN (as destination) in the matrix
  - Find the person from another ASN
  - Take them out for coffee :)



Help Us!

- To improve accuracy of this diagnostic tool
  - If your ASN is not on the graph, apply for a RIPE Atlas probe
  - Add more probes to your country to increase “resolution”
  - If you move, remember to update your probe’s geolocation
- Improve infrastructure geolocation: contribute data to OpenIPMap
  - https://marmot.ripe.net/openipmap/
  - https://github.com/RIPE-Atlas-Community/openipmap

Contribute to the Code!

- Re-use and rewrite the code: it is free and open-source software
  - https://github.com/RIPE-Atlas-Community
  - https://github.com/emileaben/ixp-country-jedi

Take Part in the RIPE Atlas Community

RIPE Atlas Community

- Individual volunteers host probes in their homes or offices
- Organisations host RIPE Atlas anchors
- Sponsors give financial support or host multiple probes in their own networks
- Read more in your language:
  - http://www.lacnic.net/web/lacnic/ripe-atlas

RIPE Atlas Community

- Ambassadors distribute probes at conferences, give presentations, etc.
- Network operators create measurements to monitor and troubleshoot
- Researchers and students write papers

RIPE Atlas Developers' Community

- Programmers contribute analysis code:
  - https://github.com/RIPE-Atlas-Community/
- IXP Country Jedi:
  - https://github.com/emileaben/ixp-country-jedi
- Measurement source code available:
  - https://labs.ripe.net/Members/philip_homburg/ripe-atlas-measurements-source-code

Hosting a Probe

- Create a RIPE NCC Access account
- Apply online and we will ship a probe to you:
  - https://atlas.ripe.net/apply
- Once you receive it, register your probe:
  - https://atlas.ripe.net/register
- Plug in your probe
- If you receive a probe at a conference or training course, just register it and plug it in!

More Hackathons!

- Join the hackathons in 2016
- Before each RIPE Meeting - save the dates!
  - Copenhagen: 21-22 May
  - Madrid: 22-23 October

A Gift for You!

- Get one million RIPE Atlas credits to run your own measurements:
  - Create a RIPE NCC access account: ripe.net/register
  - Redeem this voucher “LACNIC25ATLASADVANCED”: https://atlas.ripe.net/user/credits/#!redeem
- Enjoy using RIPE Atlas and let us know about your findings!

Contact Us

- Mailing list for users: ripe-atlas@ripe.net
- Articles and updates: https://labs.ripe.net/atlas
- Questions and bugs: atlas@ripe.net
- Twitter: @RIPE_Atlas and #RIPEAtlas
- GitHub: https://github.com/RIPE-Atlas-Community
- Roadmap: https://atlas.ripe.net/docs/roadmap/

Questions

atlas@ripe.net
@RIPE_Atlas

Additional Slides

Command-line (CLI) Toolset

RIPE Atlas CLI

- Network troubleshooting for command-line pros
- Familiar output (ping, dig, traceroute)
- Linux/OSX
  - http://ripe-atlas-tools.readthedocs.org/en/latest/installation.html#requirements-and-installation
- Windows (experimental)
  - https://github.com/chrisamin/ripe-atlas-tools-win32

Install RIPE Atlas Tools on Ubuntu

- https://ripe-atlas-tools.readthedocs.org/en/latest/
- $ sudo apt-get install python-dev libffi-dev libssl-dev
- $ sudo apt-get install python-virtualenv python-pip
- Setup vritualenv
- pip install ripe.atlas.tools

Install RIPE Atlas Tools on *nix

- Install virtualenv
  $ sudo easy_install pip
  $ sudo pip install virtualenv
- Create virtualenv for atlas-tools
  $ virtualenv venv-atlas
- Activate virtualenv (note the ‘.’)
  $ .venv-atlas/bin/activate
- Install atlas-tools
  $ pip install ripe.atlas.tools
- Add to PATH
  export PATH=$PATH:~/venv-atlas/bin

Install RIPE Atlas Tools on Windows

- github.com/chrisamin/ripe-atlas-tools-win32
- github.com/chrisamin/ripe-atlas-tools-win32/releases/download/v0.1.1/RipeAtlasToolsSetup.exe

RIPE Atlas CLI

- Open-source
  - Community contributions led by RIPE NCC
- Documentation:
  - https://ripe-atlas-tools.readthedocs.org/
- Source:
  - https://github.com/RIPE-NCC/ripe-atlas-tools/
- Contribute:
  - https://github.com/RIPE-NCC/ripe-atlas-tools/blob/master/CONTRIBUTING.rst

Configure RIPE Atlas CLI

- Reuse the API key of the previous exercise
  - Or create a new one at https://atlas.ripe.net/keys/
- Configure your CLI
  - ripe-atlas configure --set authorisation.create=MY_API_KEY

Fetch an Existing Measurement

- Fetch the ping measurement 2340408
  - ripe-atlas report 2340408

Search Probes

- Search all probes in AS3333
  - ripe-atlas probes --asn 3333
- Show specific fields
  - ripe-atlas probes --asn 3333 --field asn_v6 --field country --field is_public --field description --field status
- Search for probes in and around Istanbul
  - ripe-atlas probes --location "Istanbul, Turkey" --radius 15

Create a Measurement

- Create a ping measurement to wikipedia.org
  - One-off, default parameters
  - ripe-atlas measure ping --target wikipedia.org

```
Looking good!  Your measurement was created and details about it can be found here:

  https://atlas.ripe.net/measurements/3499718/


Connecting to stream...

48 bytes from probe #18433 94.112.176.45   to 91.198.174.192 (91.198.174.192): ttl=50 times:41.979,  41.492,  40.769,
48 bytes from probe #20111 37.151.230.180  to 91.198.174.192 (91.198.174.192): ttl=57 times:100.511, 100.136, 100.325,
48 bytes from probe #25003 176.193.48.211  to 91.198.174.192 (91.198.174.192): ttl=59 times:47.967,  47.476,  47.403,
48 bytes from probe #20313 5.199.160.9     to 91.198.174.192 (91.198.174.192): ttl=58 times:36.501,  36.245,  36.285,
48 bytes from probe #22573 89.176.43.44    to 91.198.174.192 (91.198.174.192): ttl=52 times:28.747,  27.712,  28.446,
48 bytes from probe #19413 89.71.47.56     to 91.198.174.192 (91.198.174.192): ttl=51 times:49.89,   49.779,  50.277,
48 bytes from probe #18635 78.52.132.137   to 91.198.174.192 (91.198.174.192): ttl=57 times:37.462,  38.095,  37.73,
48 bytes from probe #23223 62.65.126.46    to 91.198.174.192 (91.198.174.192): ttl=53 times:23.169,  23.412,  33.067,
48 bytes from probe #17511 87.81.148.2     to 91.198.174.192 (91.198.174.192): ttl=56 times:13.281,  12.885,  13.039,
48 bytes from probe #12584 46.175.22.202   to 91.198.174.192 (91.198.174.192): ttl=59 times:36.073,  35.788,  35.883,
```

Other Examples Using Ping

- Geo-specific from 20 probes from Canada:
  - ripe-atlas measure ping --target example.com --probes 20 --from-country ca
- 20 Canadian probes that 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

Exercise

Using RIPE Atlas CLI

Search Probes

- Use the traceroute command to test the reachability of wikipedia.org on TCP port 443 from 20 probes in France
- Render the results collected in the previous exercise in JSON format

Search Probes

- Use the traceroute command to test the reachability of wikipedia.org on TCP port 443 from 20 probes in France
  - ripe-atlas measure traceroute --protocol TCP --target wikipedia.org --port 443 --probes 20 --from-country fr
- Render the results collected in the previous exercise in JSON format
  - ripe-atlas report {MSM_ID} --renderer raw

More RIPE Atlas Features

Good to Know

- Six types of measurements: ping, traceroute, DNS, SSL/TLS, NTP and HTTP (to anchors)
- APIs to start measurements and get results
- Powerful and informative visualisations
- CLI tools
- Streaming data: real-time results
- Plus: “Time Travel”, LatencyMON, DomainMON
- Roadmap

Latest Results API

- https://atlas.ripe.net/docs/measurement-latest-api/
  - Widget monitoring value in real time (100 probes pinging websites worldwide)
  - Alert based on average measurements per hour
  - Big network event, e.g. Internet outage in a region
  - DNS domain monitoring; configurable measurements using ten RIPE Atlas anchors
- https://labs.ripe.net/Members/suzanne_taylor_muzzin/ripe-atlas-latest-results-api-and-parsing-library

Measurement Security

- Use API keys to:
  - Create measurements without logging in
  - Securely share your measurement data with others
- To create, manage and delete API keys:
  - https://atlas.ripe.net/keys/
  - https://atlas.ripe.net/docs/keys2/
- Examples:
  - https://atlas.ripe.net/docs/rest/

More Security

- Probes:
  - Hardware trust material (regular server address, keys)
  - No open ports; initiate connection; NAT is okay
  - Don’t listen to local traffic
  - No passive measurements
- Measurements triggered by “command servers”

More Security

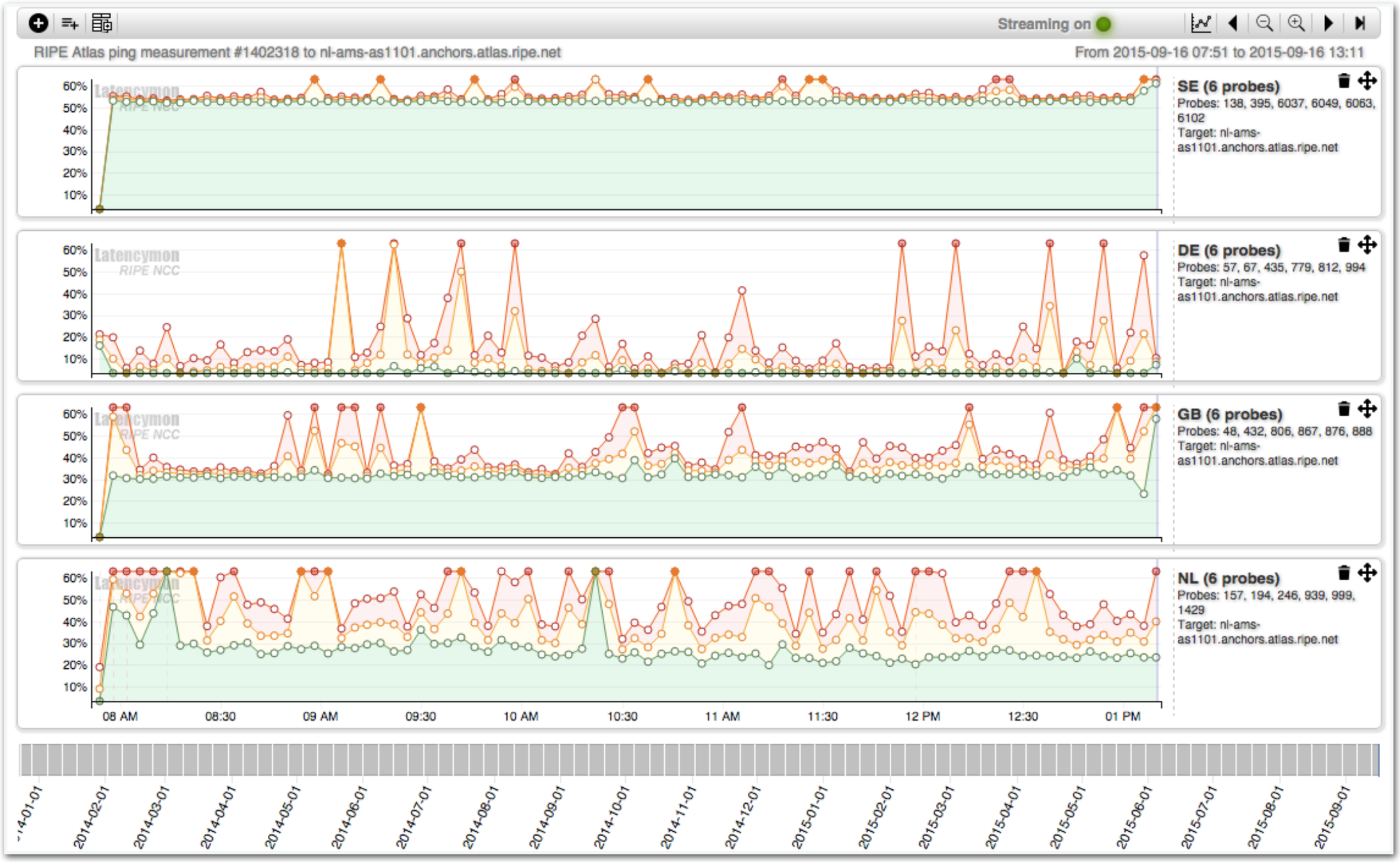
- Inverse SSH tunnels
- Source code published
- Reported vulnerabilities:
  - https://atlas.ripe.net/docs/security/

Crowdsourced Infrastructure Geolocation: OpenIPMap

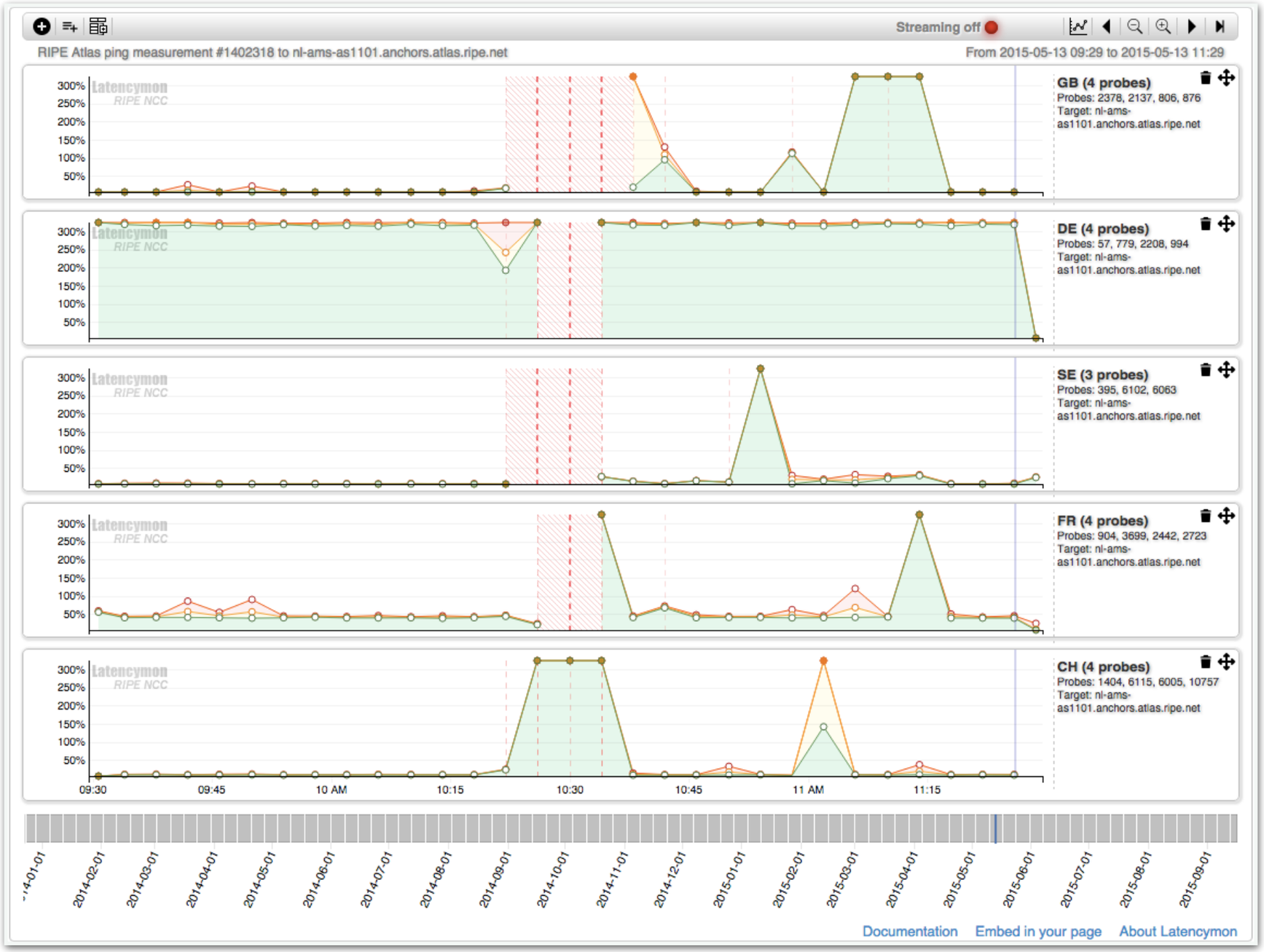
- Visualising traceroutes on the map is difficult!
  - Routers' geolocation data is often very inaccurate
  - RIPE Atlas performs many traceroutes through Internet core
- Community of operators contributes data to Open IP Map (think OpenStreetMap for IPs)
  - https://marmot.ripe.net/openipmap/
- You can modify, reuse and improve the code
  - https://github.com/RIPE-Atlas-Community/openipmap

LatencyMON

LatencyMON

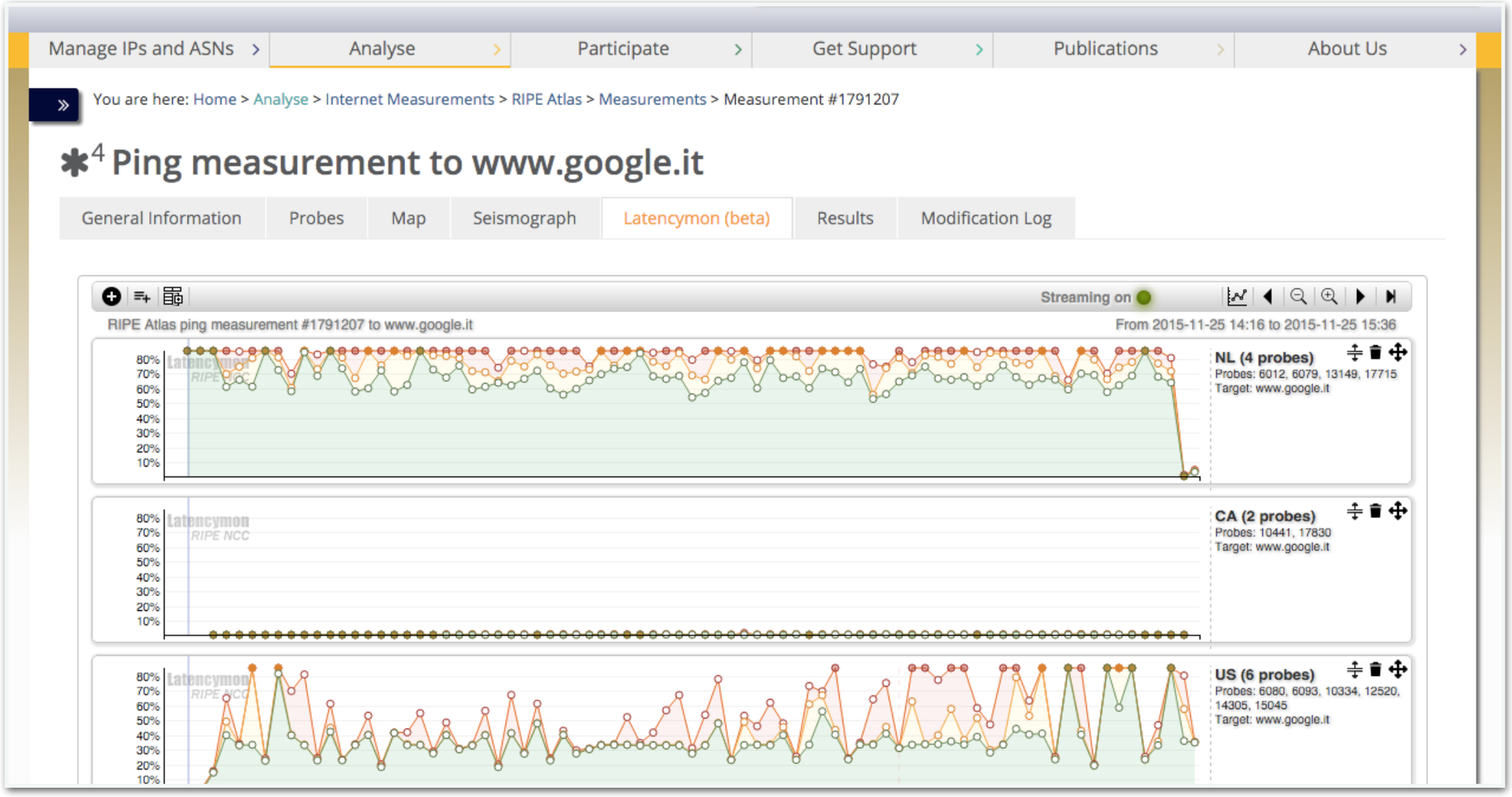

Streaming on
RIPE Atlas ping measurement #1402318 to nl-ams-as1101.anchors.atlas.ripe.net
From 2015-09-16 07:51 to 2015-09-16 13:11
SE (6 probes)
Probes: 138, 395, 6037, 6049, 6063, 6102
Target: nl-ams-as1101.anchors.atlas.ripe.net
DE (6 probes)
Probes: 57, 67, 435, 779, 812, 994
Target: nl-ams-as1101.anchors.atlas.ripe.net
GB (6 probes)
Probes: 48, 432, 806, 867, 876, 888
Target: nl-ams-as1101.anchors.atlas.ripe.net
NL (6 probes)
Probes: 157, 194, 246, 939, 999, 1429
Target: nl-ams-as1101.anchors.atlas.ripe.net
Latencymon
RIPE NCC
60%
50%
40%
30%
20%
10%
08 AM
08:30
09 AM
09:30
10 AM
10:30
11 AM
11:30
12 PM
12:30
01 PM
2014-01-01
2014-02-01
2014-03-01
2014-04-01
2014-05-01
2014-06-01
2014-07-01
2014-08-01
2014-09-01
2014-10-01
2014-11-01
2014-12-01
2015-01-01
2015-02-01
2015-03-01
2015-04-01
2015-05-01
2015-06-01
2015-07-01
2015-08-01
2015-09-01

LatencyMON Packet Loss



LatencyMON

- Demo:
  - https://atlas.ripe.net/measurements/1791207/



LatencyMON Goals

- Performance comparisons to reach a service or website from different countries or providers
- Measuring the spread of a network outage
- Measuring and comparing CDN or DNS resolution in multiple geographic areas

LatencyMON Goals (cont'd)

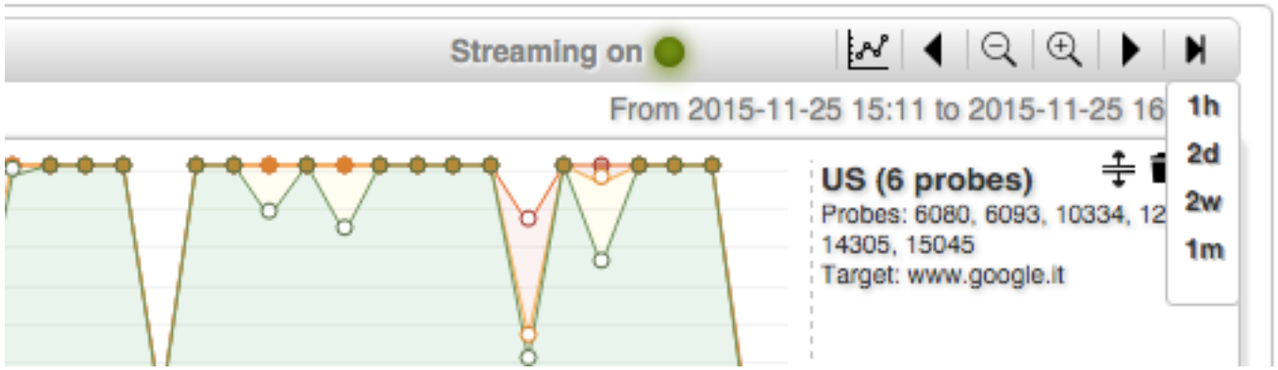
- Re-using measurements - even for measurement types other than ping - to get information about latencies
- Comparing multiple ISPs or hosting providers at the same time from vantage points with characteristics similar to those at the user end
- Creating views that are easily shareable and can be embedded in reports

Monitoring with LatencyMON

- Embed latencyMON

```
<script src="https://atlas.ripe.net/resource/latencymon/latencymon-widget-main.js"></script>
<div id="place-here"></div>
<script>
    initLatencymon(
        '#place-here',
        {}, // Tool options, see table below for more info
        { measurements:[1791207, 2083078] } // Query options, see table below for more info
        );
</script>
```

- Enable real-time streaming



- Documentation:
  - https://atlas.ripe.net/docs/tools-latencymon/#embed

Exercise

Check your Geographical Distribution

Task 1

- You want to improve the geographical distribution of your contents by determining where delay can be reduced
  - You created a measurement in the previous exercise and you have the measurement ID (otherwise use: 1791207)
  - Click on the LatencyMON tab of your measurement
  - Delete all the default-created charts

Task 2

- Create one group of two probes from DE (Germany)
- Create one group of two probes from NL (Netherlands)
- Create one group of two probes from US (United States)
- What is the country with the greatest latency in your case? Type it in the chat room!

Exercise

Compare HTTP Over IPv4 and IPv6

More Tasks

- You want to check performance over IPv4 and IPv6 (two measurements) as a real end user
  - You need an anchor for HTTP measurements! Use https://atlas.ripe.net/probes/6001/
  - Open the LatencyMON tab of the HTTP IPv6 (2841527) measurement
  - Add the HTTP IPv4 measurement ID in LatencyMON (2841526)
  - Create **two groups of eight probes** each: one per measurement
- Share the link of the view of the last two days

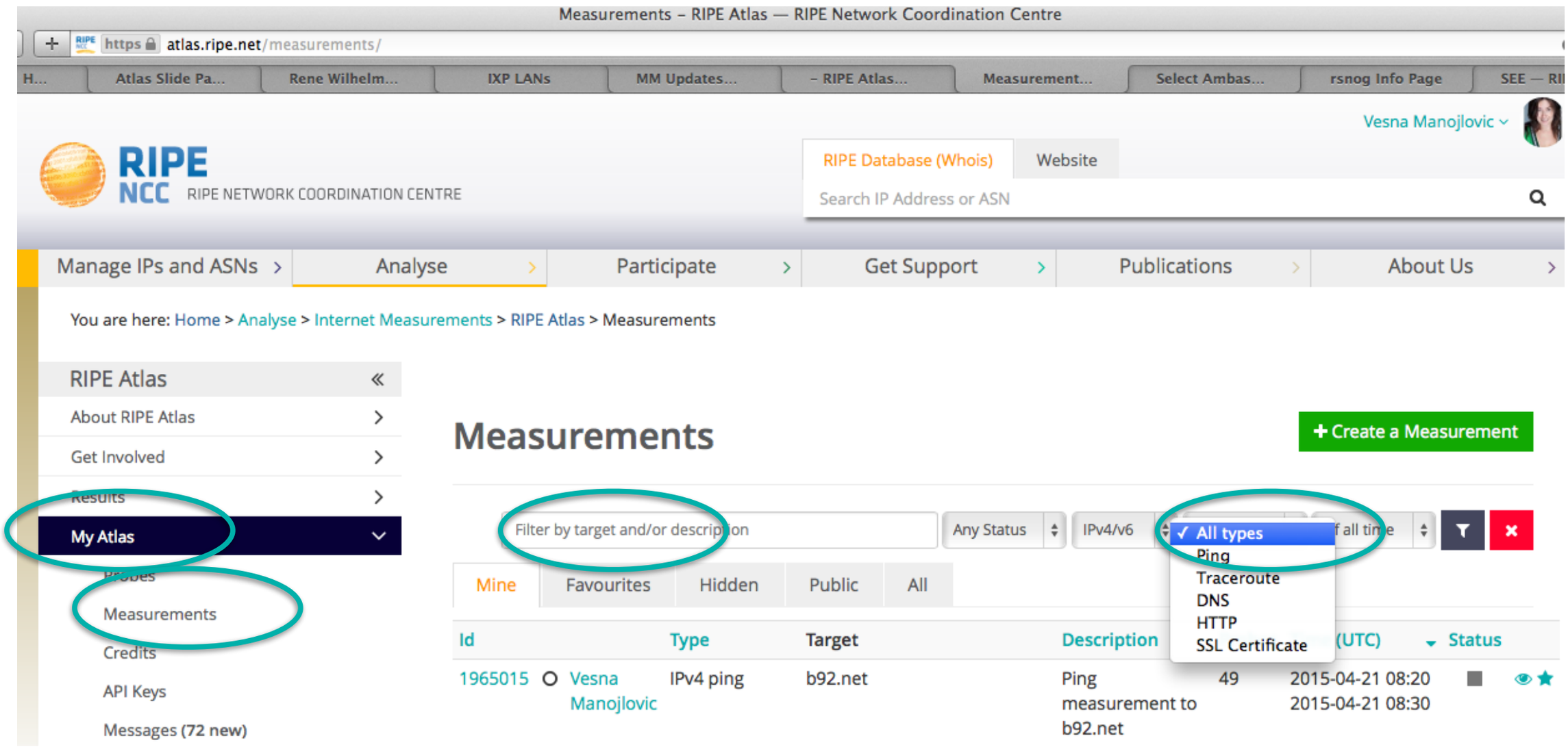
Finding Results of Public Measurements

Use Existing Measurements

- There are many measurements already running!
- Search for existing public measurements first
- Schedule your own measurement if you don't find what you're looking for

Looking Up Measurement Results

- Go to “My Atlas” > “Measurements”



Documentation

- Documentation for analysing measurement results:
  - https://atlas.ripe.net/docs/rest/
  - https://github.com/RIPE-NCC/ripe.atlas.sagan
- More tools:
  - https://github.com/RIPE-Atlas-Community
  - https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/README.md

Exercise

Analyse Measurement Results

Tasks

- Download results of a specific public measurement
- Read the text of the result, to understand structure

Task 1: Download Measurement Results

- Find the measurement
  - IPv6 ping to google.com
  - msm-ID: 1004005
- Click on measurement, then “Download”
  - Specify the time period (for example, yesterday)
- Results in JSON

Tips for Downloading Results

- Solution URL:
  - https://atlas.ripe.net/api/v1/measurement/1004005/result/?start=1435104000&stop=1435276799&format=json
- Save the measurement(s) locally:
  - `$ curl https://atlas.ripe.net/api/v1/measurement/1004005/result/?start=1435104000&stop=1435276799&format=json > measurement-test.json`

Task 2: Look at the Result


```
[{“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)

Reference (msm ID)

Packet loss: difference between sent & received!

Task 3: Analyse Results (optional)

- Find out how many times RTT was above 60ms
  - Use Python, JavaScript or something else
- For the JavaScript solution, you can use this as a starting point:
  - https://stat.ripe.net/widgets/demo/script_me.html

Task 4: Code Examples

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"]
if rtt >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>
```