Using RIPE Atlas for network measurement and monitoring

Sebastian Castro, NZRS
(Material kindly provided by RIPE NCC)
APRICOT, Auckland, February 2016
Goals

• Learn how to:
  Benefit from using RIPE Atlas measurements for network monitoring and troubleshooting
  Use API calls to create measurements
  Integrate RIPE Atlas with existing monitoring systems

• Opportunity for hands-on practice
Prerequisites

• I assume you have already used RIPE Atlas

• Do you have a RIPE NCC Access account?
  • If not - quickly create one: ripe.net/register

• Do you have credits to spend?
  • If not - tell me to transfer some credits
Warm-up question

• What is your background?
  network operator?
  software engineer / programmer?
  data scientist?
  sysadmin?
  other? please specify :-(
Overview

- Introduction to RIPE Atlas
- Creating measurements
- Integration with network monitoring systems
- Real-time performance monitoring
- Take part in the RIPE Atlas community
- News:
  - CLI Toolset
- Additional slides
Introduction to RIPE Atlas
Introduction

• RIPE Atlas is a global active measurements platform

• Goal: view Internet reachability

• Probes hosted by volunteers
  Small probes
  Anchors

• 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 and NTP
Numbers for January 2016

• 9,200+ probes connected
• 5,000+ active users in the last quarter
• 2,500+ results collected per second
• 35,000+ customised measurements weekly
Creating a Measurement
Benefits of your own measurements

• A customer reports a problem: they cannot reach one of your servers
  You can schedule pings or traceroutes from up to 500 RIPE Atlas probes from a particular region to check where the problem might be
• Measuring packet loss on a suspected “bad” link
• Testing anycast deployment
• Identify DNS cache poisoning/lies
Credits system

• Running your own measurements cost credits
  ping = 10 credits, traceroute = 20, etc.
• Why? Fairness and to avoid overload
• Daily spending limit & max measurements user can create
• Hosting a RIPE Atlas probe earns credits
• Earn extra credits by:
  Being a RIPE NCC member
  Hosting an anchor
  Sponsoring probes
Credits overview

Account Information
This is where you're able to view the history of your credit use. There are visualisations available, and you can also transfer credits to someone else.

My Atlas > Credits
Give credits to someone
Scheduling a measurement

• Log in to atlas.ripe.net
• “My Atlas” > “Measurements”
• Three methods:
  1. Quick & Easy
     • Type
     • Target
     • Done!
  2. Advanced GUI usage
  3. CLI scripting using API
2: Using GUI to schedule a measurement

- Mostly used for a periodic, long time measurement
  If just once, ASAP, choose “One-off”
- Choose type, target, frequency, # of probes, region...
  Interactive interface helps you at each step
- Each measurement will have unique ID
- “API Compatible Specification” is generated too
3: Using API to schedule a measurement

- Using command-line & scripting: Application Programming Interface (API)
  
  https://atlas.ripe.net/docs/measurement-creation-api/
  https://atlas.ripe.net/keys/

- You will need API keys
  To create measurements without logging in
  To securely share your measurement data
API documentation:

- [https://atlas.ripe.net/docs/measurement-creation-api/](https://atlas.ripe.net/docs/measurement-creation-api/)
- [https://atlas.ripe.net/doc/credits](https://atlas.ripe.net/doc/credits)
- [https://atlas.ripe.net/doc/udm](https://atlas.ripe.net/doc/udm)
- [https://atlas.ripe.net/keys/](https://atlas.ripe.net/keys/)
- [https://atlas.ripe.net/docs/keys2/](https://atlas.ripe.net/docs/keys2/)
Looking up measurements results

- Go to “My Atlas” > “Measurements”
Available visualisations: ping

- List of probes: sortable by RTT
- Map: colour-coded by RTT
- Seismograph: stacked multiple pings with packet loss
Exercise

Create a Measurement
Tasks

• Create a **ping** measurement:
  Involving ten probes
  To a target of your choice
  Source is your country
  Duration of two days

1. Warm-up: Create a measurement using the GUI
2. Create API Key
3. Schedule a measurement using the API
Sub-task 1: Use web interface

- Useful hint: once you generate a measurement, copy “API Compatible Specification” to text file
- Note Measurement-ID
Sub-task 2: Create API key

- Click on “Create an API Key”
- Choose type: “create a new user-defined measurement”
- “Object” is not applicable (N/A) for this type
- Give it a label
- Give it a duration of validity (leave empty for defaults)
- “Key” value to be passed on to the API call (next step)
Sub-task 3: Use API

- Schedule a measurement using API
  Use the “key” you just generated
  Hint: copy and past API call syntax from the measurement generated by the GUI

- Example:

  ```bash
  $ 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
  ```
Example with command line

```
air-becha:~ becha$ curl -H "Content-Type: application/json" -H "Accept: application/json" -X POST -d '{ "definitions": [ { "target": "ping.xs4all.nl", "description": "My First Measurement", "type": "ping", "af": 4 } ], "probes": [ { "requested": 10, "type": "country", "value": "RS" } ] }' https://atlas.ripe.net/api/v1/measurement/?key=7b4c3441-4504-4d83-9ed7-fbf1a007d060
{"measurements":[2421551]}air-becha:~ becha$
```
Integration with Network Monitoring Systems
Network monitoring

- Network operators use tools for monitoring network health (e.g. Nagios and Icinga)
- These tools can receive input from RIPE Atlas via the API
- Benefits:
  pings from 500 out of 8,000+ probes around the world
  See your network from the outside
  Plug into your existing practices
Steps for integration

1. Create a RIPE Atlas ping measurement
2. Go to “Status Checks” URL
3. Add your alerts in Nagios or Icinga
Creating status checks

- Status checks work via RIPE Atlas’ RESTful API
  [https://atlas.ripe.net/api/v1/status-checks/MEASUREMENT_ID/](https://atlas.ripe.net/api/v1/status-checks/MEASUREMENT_ID/)

- You define the alert parameters, for example:
  - Threshold for percentage of probes that successfully received a reply
  - How many of the most recent measurements to base it on
  - The maximum packet loss acceptable

- Documentation:
  [https://atlas.ripe.net/docs/status-checks/](https://atlas.ripe.net/docs/status-checks/)
Icinga examples

- Community of operators contributed configuration code!
  Making use of the built-in “check_http” plugin

- GitHub examples:

- Post on Icinga blog:
  https://www.icinga.org/2014/03/05/monitoring-ripe-atlas-status-with-icinga-2/
Exercise

Setting up “Status Checks”
• Set up and configure a “status check”
  For an existing ping measurement
  [link]

• Configure the status check in such a way that you will trigger an alert for this measurement

• Optional: set-up status check for your own ping measurement!
One possible solution: Set the median RTT to a lower level.

https://atlas.ripe.net/api/v1/status-checks/3369314/?median_rtt_threshold=10

Example of the alerts

```json
{"total_alerts":32,"global_alert":true,"probes":{
  "18433":{"all":null,null,null},
  "last":null,"last_packet_loss":100.0,"alert":true,"source":"Area: WW","alert_reasons":["loss"]},
  "15041":{"source":"Area: WW","last_packet_loss":0.0,"last":19.928,"alert":false},
  "18696":{"all":null,null,null},
  "last":null,"last_packet_loss":100.0,"alert":true,"source":"Area: WW","alert_reasons":["loss"]},
  "16265":{"source":"Area: WW","last_packet_loss":0.0,"last":22.72,"alert":false},
  "20236":{"all":null,null,null},
  "last":null,"last_packet_loss":100.0,"alert":true,"source":"Area: WW","alert_reasons":["loss"]},
  "12944":{"all":null,null,null},
  "last":null,"last_packet_loss":100.0,"alert":true,"source":"Area: WW","alert_reasons":["loss"]},
  _2195":{"all":null,null,null},
  "last":null,"last_packet_loss":100.0,"alert":true,"source":"Area: WW","alert_reasons":["loss"]},
```
Real-time performance monitoring
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 web sockets

- There are two types of data:
  Measurement results
  Probe connection status events
Real time streaming uses

- Visualising network outages
- Server and performance monitoring
- In March 2015: used by almost all hackathon teams:
  https://labs.ripe.net/Members/becha/ripe-atlas-hackathon-results
- Documentation
  https://atlas.ripe.net/docs/result-streaming/
  https://labs.ripe.net/Members/suzanne_taylor_muzzin/data-streaming-in-ripe-atlas
Probe (dis)connection events

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

Using streaming API
Monitoring server reachability

• 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 which threshold (e.g. greater than 30ms)
  Imposes the threshold on “min” (the minimum result of the three ping attempts)
1. Go to http://atlas.ripe.net/webinar/streaming01.html
2. Open the development console
3. Wait for results to arrive
4. Optional: Save the HTML file locally and edit the code to your liking
<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>&lt;span id="output"&gt;nothing yet&lt;/span&gt;</b></div>
    <div>Open the source code to see how it works. Create your tool/visualisation with the RIPE Atlas streaming!</div>
  </body>
</html>

<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>
Example of results

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=e80kM7zfF2c-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: 325.79333333333...}
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: 327.32533333333...}
I received ▶ Object { af: 4, prb_id: 15965, result: Array[3], ttl: 45, avg: 47.63133333333...}
I received ▶ Object { af: 4, prb_id: 15965, result: Array[3], ttl: 45, avg: 47.69966666666...}
I received ▶ Object { af: 4, prb_id: 15965, result: Array[3], ttl: 45, avg: 47.48166666666...}
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: 47.86266666666...}
I received ▶ Object { af: 4, prb_id: 19566, result: Array[3], ttl: 40, avg: 47.59466666666...}
I received ▶ Object { af: 4, prb_id: 19566, result: Array[3], ttl: 40, avg: 47.50033333333...}
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: 34.08433333333...}
I received ▶ Object { af: 4, prb_id: 18311, result: Array[3], ttl: 49, avg: 32.75133333333...}
I received ▶ Object { af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 182.44633333333...}
I received ▶ Object { af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 193.99533333333...}
I received ▶ Object { af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 182.29133333333...}
I received ▶ Object { af: 4, prb_id: 16010, result: Array[3], ttl: 46, avg: 191.61033333333...}
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: 35.00933333333...}
I received ▶ Object { af: 4, prb_id: 14918, result: Array[3], ttl: 49, avg: 35.00433333333...}
I received ▶ Object { af: 4, prb_id: 20668, result: Array[3], ttl: 45, avg: 38.88466666666...}
I received ▶ Object { af: 4, prb_id: 20668, result: Array[3], ttl: 45, avg: 38.80266666666...}
I received ▶ Object { af: 4, prb_id: 20668, result: Array[3], ttl: 45, avg: 38.88066666666...}
I received ▶ Object { af: 4, prb_id: 6093, result: Array[3], ttl: 49, avg: 128.72733333333...}
I received ▶ Object { af: 4, prb_id: 6093, result: Array[3], ttl: 49, avg: 128.73733333333...}
I received ▶ Object { af: 4, prb_id: 6093, result: Array[3], ttl: 49, avg: 128.88833333333...}
Task 2: View (dis)connect events

See in the console of your browser the connection and disconnection events of all the RIPE Atlas probes

Steps:

- Create your empty HTML page
- Connect to the streaming
- Subscribe to stream_type: “probestatus”
Solution

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

    socket.on("atlas_probestatus", function(status){
        console.log("I received ", status);
    });

    socket.emit("atlas_subscribe", { stream_type: "probestatus" });
</script>
```
The Newest Features

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
Command Line Interface (CLI) Toolset
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 about ongoing measurement
3. look at the results
4. collect results from the ongoing measurement (streaming)
Creating ping measurement

- Simplest: one-off, using default values: (50 probes to “target”)

ripe-atlas measure ping
--auth API_KEY
--target wikipedia.org
Other 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
  ```
• Report for the traceroute
• `ripe-atlas measure traceroute --probes 2 --target google.ca`
Searching for existing measurements

ripe-atlas measurements --af 6 --status ongoing --limit 15 --search google

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1004085</td>
<td>ping</td>
<td>google - v6</td>
<td>Ongoing</td>
</tr>
<tr>
<td>1004732</td>
<td>traceroute</td>
<td>google v6 traceroute</td>
<td>Ongoing</td>
</tr>
<tr>
<td>1007128</td>
<td>dns</td>
<td>Google.f1 AAAA reply</td>
<td>Ongoing</td>
</tr>
<tr>
<td>1012449</td>
<td>sslcert</td>
<td><a href="http://www.google.com">www.google.com</a></td>
<td>Ongoing</td>
</tr>
<tr>
<td>1024911</td>
<td>ping</td>
<td>IPV6 Google DNS</td>
<td>Ongoing</td>
</tr>
<tr>
<td>1404300</td>
<td>ping</td>
<td>IPV6 Ping to Google</td>
<td>Ongoing</td>
</tr>
<tr>
<td>1665737</td>
<td>ping</td>
<td>google.com - 2404:6800:4003:c00::71</td>
<td>Ongoing</td>
</tr>
<tr>
<td>1796260</td>
<td>ping</td>
<td>Ping measurement to <a href="http://www.google.com">www.google.com</a></td>
<td>Ongoing</td>
</tr>
<tr>
<td>1889086</td>
<td>traceroute</td>
<td>Traceroute measurement to ipv6.google.com</td>
<td>Ongoing</td>
</tr>
<tr>
<td>2062542</td>
<td>traceroute</td>
<td>Traceroute measurement to ipv6.google.com</td>
<td>Ongoing</td>
</tr>
<tr>
<td>2062543</td>
<td>ping</td>
<td>Ping measurement to ipv6.google.com</td>
<td>Ongoing</td>
</tr>
<tr>
<td>2143865</td>
<td>ping</td>
<td>Ping measurement to ipv6.google.com</td>
<td>Ongoing</td>
</tr>
<tr>
<td>2486602</td>
<td>traceroute</td>
<td>IPV6 Traceroute measurement to snapchat.com</td>
<td>Ongoing</td>
</tr>
<tr>
<td>2486820</td>
<td>ping</td>
<td>Google IPV6</td>
<td>Ongoing</td>
</tr>
</tbody>
</table>

Showing 14 of 14 total measurements
Searching for very specific probes

ripe-atlas probes --asn 3333 --field id --field asn_v6 --field country --field is_public --field description --field status

<table>
<thead>
<tr>
<th>ID</th>
<th>Asn_v6</th>
<th>Country</th>
<th>Public</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>SG office 1</td>
<td>Connected</td>
</tr>
<tr>
<td>14</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>VTY probe</td>
<td>Connected</td>
</tr>
<tr>
<td>15</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>SG office 2</td>
<td>Connected</td>
</tr>
<tr>
<td>111</td>
<td>3333</td>
<td>br</td>
<td>✗</td>
<td>NIC br</td>
<td>Connected</td>
</tr>
<tr>
<td>237</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>The Traveling Probe</td>
<td>Connected</td>
</tr>
<tr>
<td>1108</td>
<td>3333</td>
<td>us</td>
<td>✔</td>
<td>probe 1</td>
<td>Connected</td>
</tr>
<tr>
<td>2009</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>NCC Office 2009</td>
<td>Connected</td>
</tr>
<tr>
<td>3407</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>6001</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td>AA nl-ams-as3333</td>
<td>Connected</td>
</tr>
<tr>
<td>6012</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td>AA pre-production</td>
<td>Connected</td>
</tr>
<tr>
<td>6018</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>6019</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>RIPE NCC Anchor v2</td>
<td>Connected</td>
</tr>
<tr>
<td>6137</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>nl-ams-as3333-preprod</td>
<td>Connected</td>
</tr>
<tr>
<td>10004</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>RIPE NCC R&amp;D Office</td>
<td>Connected</td>
</tr>
<tr>
<td>10105</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>10888</td>
<td>3333</td>
<td>nl</td>
<td>✔</td>
<td>Ridderkerk - UPC 120/10Mbits</td>
<td>Connected</td>
</tr>
<tr>
<td>12989</td>
<td>nl</td>
<td>de</td>
<td>✔</td>
<td>TeraStream Test Lab</td>
<td>Connected</td>
</tr>
<tr>
<td>13343</td>
<td>3333</td>
<td>ch</td>
<td>✔</td>
<td>FSIT AG - CH-DIE061 - out of O</td>
<td>Connected</td>
</tr>
<tr>
<td>14004</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>14013</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>14020</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>14024</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>14026</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>14154</td>
<td>3333</td>
<td>us</td>
<td>✗</td>
<td></td>
<td>Connected</td>
</tr>
<tr>
<td>15027</td>
<td>3333</td>
<td>nl</td>
<td>✗</td>
<td>RETURNED TO RIPE FOR REPAIR</td>
<td>Connected</td>
</tr>
</tbody>
</table>

Showing 25 of 33 total probes
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
Take part in the RIPE Atlas community
RIPE Atlas community (part 1)

- Individual volunteers host **probes** in homes or offices
- Organisations host RIPE Atlas **anchors**
- **Sponsor** organisations give financial support or host multiple probes in their own networks
RIPE Atlas community (part 2)

- **Ambassadors** help distribute probes at conferences, give presentations, etc.
- **Developers** contribute free and open software
- **Network operators** create measurements to monitor and troubleshoot
- **Researchers and students** write papers
Contact us

- Users’ mailing list: [ripe-atlas@ripe.net](mailto:ripe-atlas@ripe.net)
- Articles and updates: [https://labs.ripe.net/atlas](https://labs.ripe.net/atlas)
- Questions and bugs: [atlas@ripe.net](mailto:atlas@ripe.net)
- Twitter: [@RIPE_Atlas](https://twitter.com/RIPE_Atlas) and [#RIPEAtlas](https://twitter.com/hashtag/RIPEAtlas)
https://atlas.ripe.net/docs/rest/
https://github.com/RIPE-NCC/ripe.atlas.sagan
https://atlas.ripe.net/docs/measurement-creation-api/
  • https://atlas.ripe.net/doc/credits
  • https://atlas.ripe.net/doc/udm
https://atlas.ripe.net/keys/
https://atlas.ripe.net/docs/keys2/
Additional slides
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
• Log in to atlas.ripe.net
  • Use your RIPE NCC Access account
  • Same account for LIR Portal, RIPE Atlas, RIPEstat, RIPE Labs...
  • Create an account if you don’t have one already
Looking up measurements results
Available visualisations: ping

- List of probes: sortable by RTT
- Map: colour-coded by RTT
- Seismograph: stacked multiple pings with packet loss
Seismograph tips

Vertical pattern: possible problem in your network

Out of each 3 pings max, min & average are calculated

Filtering & grouping by:
- country
- IPv4 ASN
- IPv6 ASN

Choose absolute or relative RTT values

Horizontal pattern: possible problem in that ASN

Auckland, Feb 2016

APRICOT: RIPE Atlas Tutorial

70
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.optteamax.de

<table>
<thead>
<tr>
<th>Probe</th>
<th>ASN (v4)</th>
<th>ASN (v6)</th>
<th>Time</th>
<th>Name</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>17840</td>
<td>6327</td>
<td></td>
<td>2015-05-19 09:38</td>
<td>null</td>
<td>362.009</td>
</tr>
<tr>
<td>18035</td>
<td>43030</td>
<td></td>
<td>2015-05-19 09:50</td>
<td>null</td>
<td>347.39</td>
</tr>
<tr>
<td>18129</td>
<td>327805</td>
<td></td>
<td>2015-05-19 09:49</td>
<td>null</td>
<td>207.743</td>
</tr>
<tr>
<td>15844</td>
<td>32098</td>
<td></td>
<td>2015-05-19 09:48</td>
<td>null</td>
<td>184.237</td>
</tr>
<tr>
<td>17857</td>
<td>852</td>
<td></td>
<td>2015-05-19 09:37</td>
<td>null</td>
<td>177.654</td>
</tr>
<tr>
<td>19894</td>
<td>6327</td>
<td></td>
<td>2015-05-19 09:36</td>
<td>null</td>
<td>168.689</td>
</tr>
<tr>
<td>19204</td>
<td>21513</td>
<td></td>
<td>2015-05-19 09:50</td>
<td>null</td>
<td>141.199</td>
</tr>
<tr>
<td>15922</td>
<td>30036</td>
<td></td>
<td>2015-05-19 09:47</td>
<td>null</td>
<td>133.309</td>
</tr>
</tbody>
</table>
Documentation

• Documentation for analysing measurements results:
  https://atlas.ripe.net/docs/rest/
  https://github.com/RIPE-NCC/ripe.atlas.sagan

• More tools:
  https://github.com/RIPE-Atlas-Community
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 ping, IPv6 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"},

Source: RIPE NCC
APRICOT: RIPE Atlas Tutorial
Auckland, Feb 2016
Task 3: Analyse results (optional)

- Find out how many times RTT was above 60ms
  Use Python or 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](https://stat.ripe.net/widgets/demo/script_me.html)
Task 4: Examples of code

Python:

Parse json and find total avg:

```python
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:

```javascript
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" });
```
RIPE Atlas anchors
RIPE Atlas anchors

• Well-known targets and powerful probes
  Regional baseline and “future history”

• Anchoring measurements
  Measurements between anchors
  200 probes targeting each anchor with measurements
  Each probe measures 4-5 anchors
  Vantage points for DNSMON service

• 130+ RIPE Atlas anchors
Locations of anchors

https://atlas.ripe.net/results/maps/network-coverage/#anchors
IXP country Jedi

Measuring Impact of IXPs on Keeping Traffic Local
Benefits (part 1)

- Operators
  Routing and traffic optimisation
- IXP operators
  Shows how IXPs help keep traffic local and regional
- IPv6 advocates
  Comparing IPv4 and IPv6 paths
- Country level: regulators, politicians, cybersecurity…

How much traffic stays within the country? Where do the paths go? Comparing countries with each other
Benefits (part 2)

• RIPE Atlas community
  More probes in more networks = higher quality of measurements data

• 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
Paths going via an IXP?

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

• 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 ASN and only “public” probes with “good” geolocation

• Hops geolocated using “OpenIPMap” database
Actions (part 1)

• Use this tool to find possible suboptimal routing and fix it
  Find your ASN in the mesh
  Find the person from another ASN
  Take them out for tea :)

• 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
Actions (part 2)

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

- Improve infrastructure geolocation: contribute data to OpenIPMap!
  https://marmot.ripe.net/openipmap/
  https://github.com/RIPE-Atlas-Community/openipmap
Want to host a probe?

Contact:  sebastian@nzrs.net.nz
www.nzrs.net.nz