Ark Topology Query System

Young Hyun
CAIDA

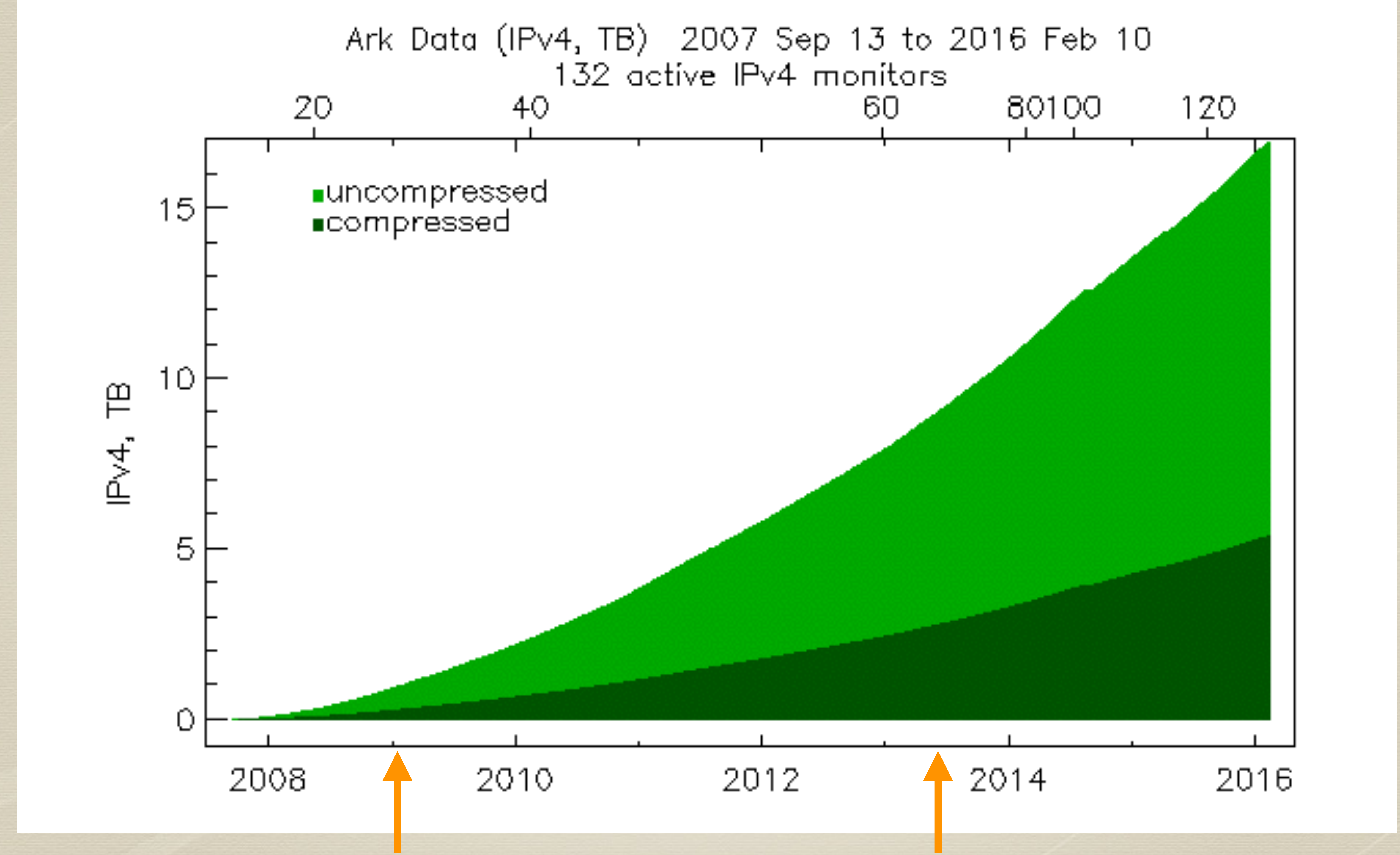
AIMS 2016 Workshop
Feb 11, 2016

Archipelago
Measurement Infrastructure

Data Stats

- 8+ years of Ark IPv4 topology data:
  - 41 billion traces, 18 TB (uncompressed warts files)
  - growing by 766 million traces, 316 GB per month
    - about **9 billion traces per year**
- 2 months of "prefix-probing" data:
  - probe every announced IPv4 BGP prefix (~609k) daily; independently from 37 monitors
  - growing by 700 million traces per month
    - **8.4 billion traces per year**

Data Stats



1 TB

doubled in last 2.6 years

Goals

- improve **data accessibility**
  - easier to find and retrieve data of interest
  - easier to process/analyze data

Goals

- target workflow:

  1. **find** traces with desired properties

  2. **analyze** traces

  3. **visualize** analyses/properties

Goals

- support full access from command line
  - execute all supported queries
  - researchers can write their own analysis/visualization scripts
- support simplified access with web interface
  - **widen audience** with pre-made queries, analyses, and visualizations
  - possible long tail of casual users

Design

- tradeoff: **efficiency vs. everything else** …
    - … flexibility/power/expressiveness/generality …
- guiding principles:
    - focus on **specific use cases**, not maximum generality
    - focus on **responsiveness** for *interactive* data exploration
        - at human time scales: ideally, tens of seconds or less per query

Design

- main focus:
  - querying of **topological** properties of traceroutes
    - (not performance; e.g., RTT that exceeds a threshold)
- query:
  - all traceroutes that pass through/reach a set of IP **addresses, prefixes, ASes,** or **countries**
    - any arbitrary prefix; not necessarily an announced BGP prefix
    - target AS = set of prefixes announced by an AS in BGP
    - target country = set of prefixes that geolocate to a country

Query Model

- terms:
  - target T = address / prefix / AS / country
  - target set $S = \{T_1, T_2, T_3, \ldots\}$
- examples:
  - T_1 = 1.1.1.1
  - T_2 = 192.168.0.0/16
  - T_3 = as3546
  - T_4 = .sy
  - S_1 = 1.1.1.1,192.168.0.0/16,as3546,.sy
  - .sy = 104.128.128.0/20,104.166.96.0/19,104.167.192.0/18,...

Query Model

- query: **addr** -d=*n* *S*
  - find all traces with at least one hop/destination address that matches **any** member of the target set *S*
  - *-d* option constrains matching addresses to be within *n* hops of start of trace (if *n* > 0) or end of trace (if *n* < 0)
  - example: addr -d 5 10.0.0.0/8,192.168.0.0/16
- query: **dest** -d=*n* *S*
  - similar to **addr** but only matches the destination address

Query Model

- query: **neigh** -d=*n* S_1 S_2 …
  - find all traces that have at least one matching hop/destination address for each target set S_i
  - *-d* option constrains matched addresses to be within *n* hops of each other
  - example: neigh -d 3 as3546 as701,as702
  - example: neigh .il .sy

Query Model

- other query options:
  - ***-t*** option constrains trace time range
  - ***-m*** option constrains trace source (monitor/vantage point)

Command-Line Interface

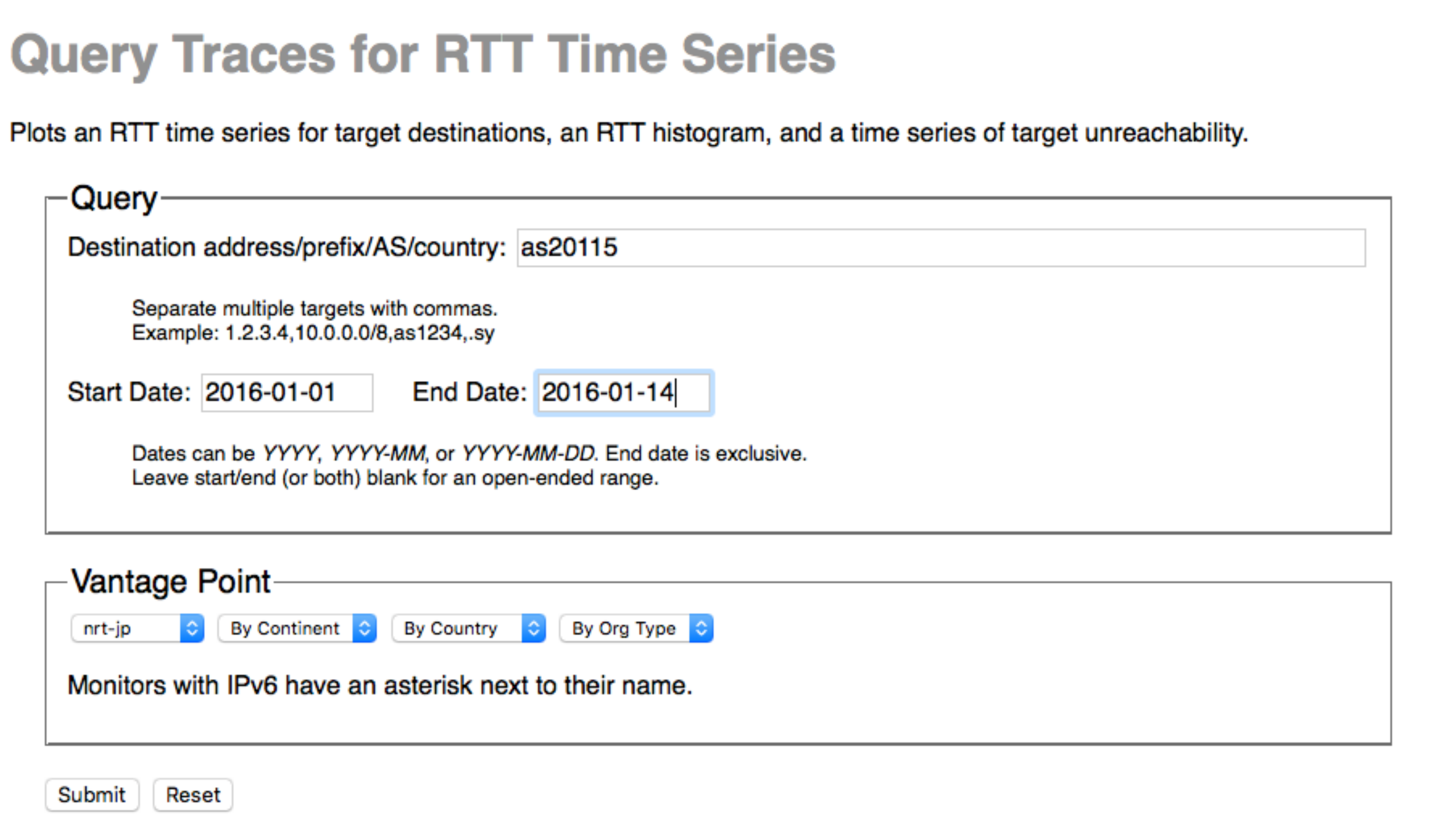
```
$ pypy ./toq dest -m san-us -q -D .sy
country sy => 87 prefixes: 104.128.128.0/20,104.166.96.0/19,104.167.192.0/18,...
dest   2007-09-13 02:08:40 UTC    2015-05-18 22:56:24 UTC  236833
```

236,833 matching traces in 725 million san-us traces collected 2007-2015

```
$ pypy ./toq dest -m san-us -l 1 -D .sy
2014-08-17 19:36:19 UTC (1408304179@0002) from 192.172.226.247
traceroute to 104.128.128.125
 1   192.172.226.252 0.480 ms
 2   192.12.207.65 0.588 ms
...
10   202.43.176.46 196.304 ms
11   103.10.198.33 201.496 ms
12   103.10.198.17 202.734 ms
```

```
$ pypy ./toq neigh -m san-us -q -D .il .sy
country il => 711 prefixes: 104.130.80.0/20,104.132.0.0/14,104.171.112.0/20,...
country sy => 87 prefixes: 104.128.128.0/20,104.166.96.0/19,104.167.192.0/18,...
neigh   2007-09-13 01:58:46 UTC    2015-05-18 22:08:14 UTC  2704012
```

Web Interface



Query Traces for RTT Time Series

Plots an RTT time series for target destinations, an RTT histogram, and a time series of target unreachability.

Query

Destination address/prefix/AS/country: as20115

Separate multiple targets with commas.
Example: 1.2.3.4,10.0.0.0/8,as1234,.sy

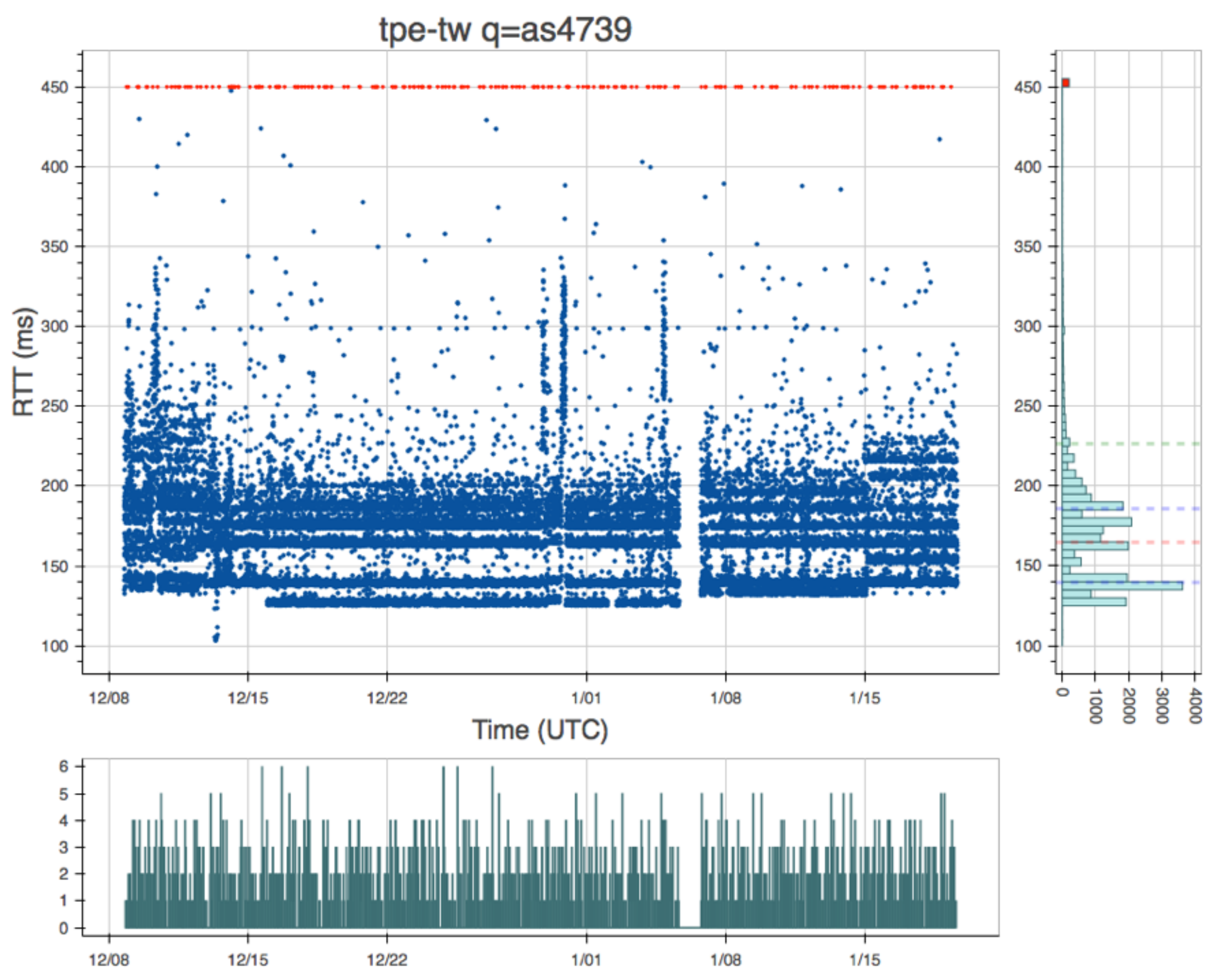
Start Date: 2016-01-01 End Date: 2016-01-14

Dates can be *YYYY*, *YYYY-MM*, or *YYYY-MM-DD*. End date is exclusive.
Leave start/end (or both) blank for an open-ended range.

Vantage Point

nrt-jp

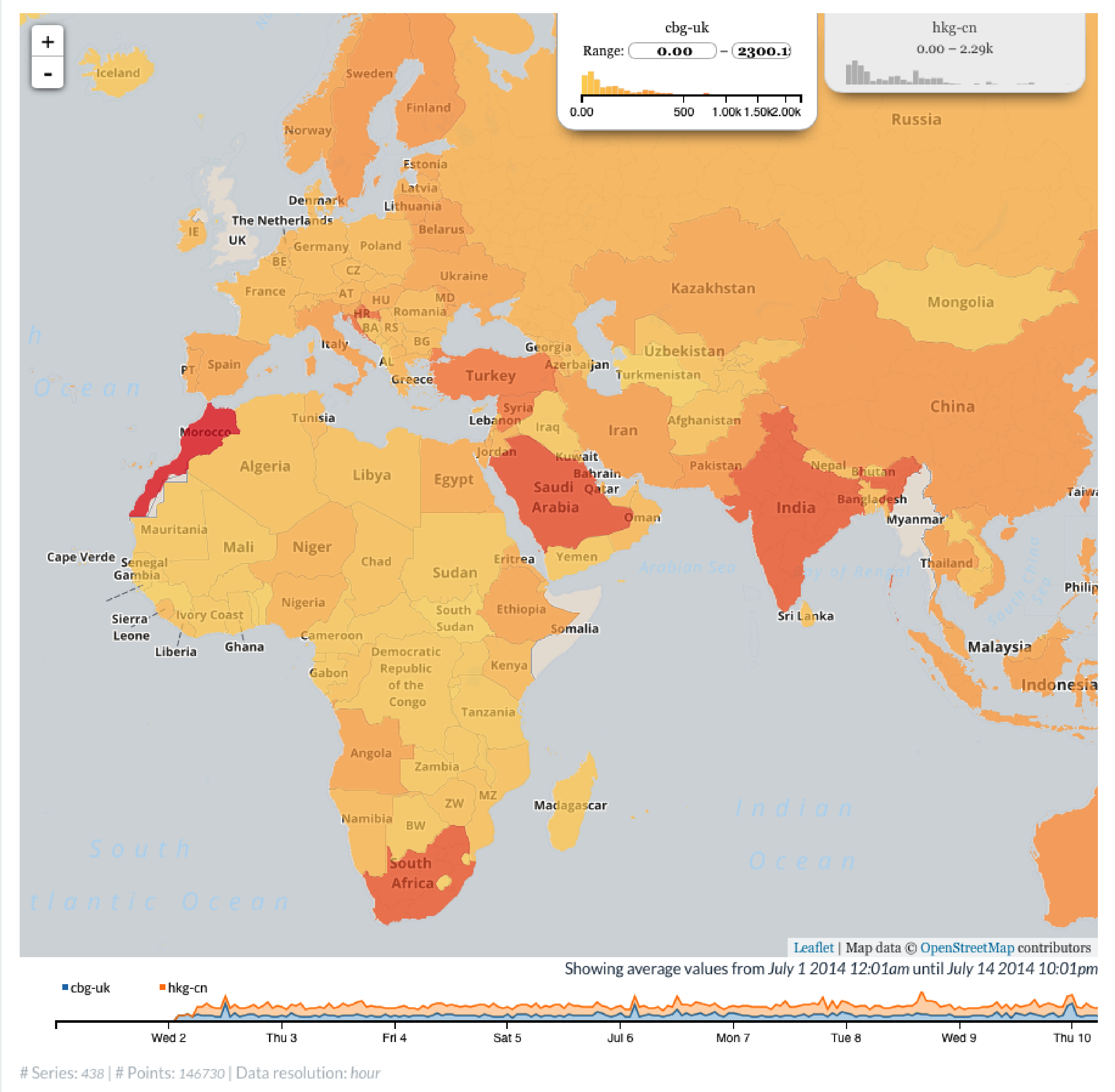
Monitors with IPv6 have an asterisk next to their name.


tpe-tw q=as4739
RTT (ms)
450
400
350
300
250
200
150
100
12/08
12/15
12/22
1/01
1/08
1/15
Time (UTC)
0
1000
2000
3000
4000
6
5
4
3
2
1
0

Future Work

- take advantage of multiple cores
  - some queries can take minutes with single core
- rewrite performance-critical code in C/C++
  - currently, several thousand line Python script
- implement commonly-desired analyses and viz

Future Work



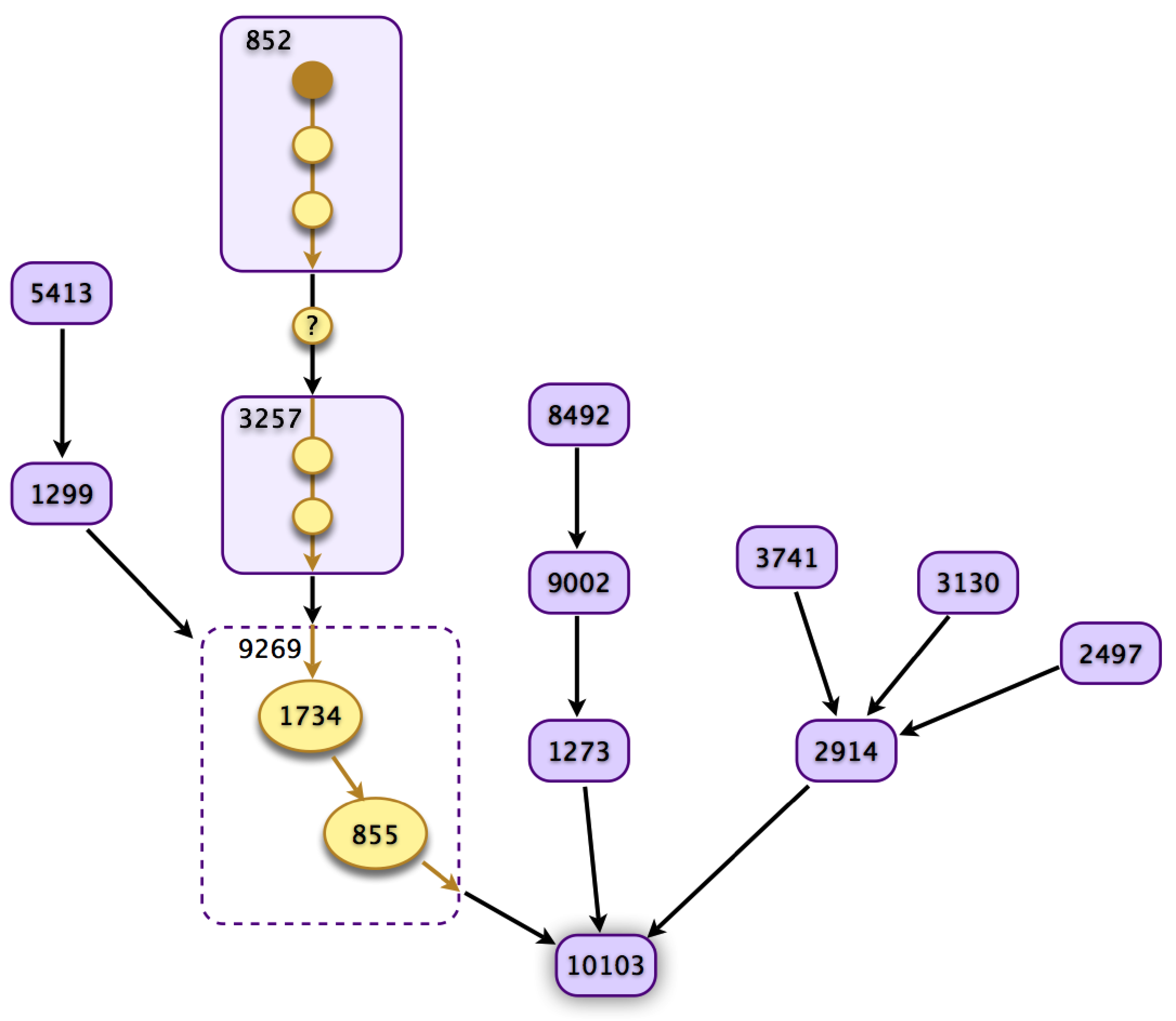
Leaflet | Map data © OpenStreetMap contributors

Showing average values from *July 1 2014 12:01am* until *July 14 2014 10:01pm*

Series: *438* | # Points: *146730* | Data resolution: *hour*

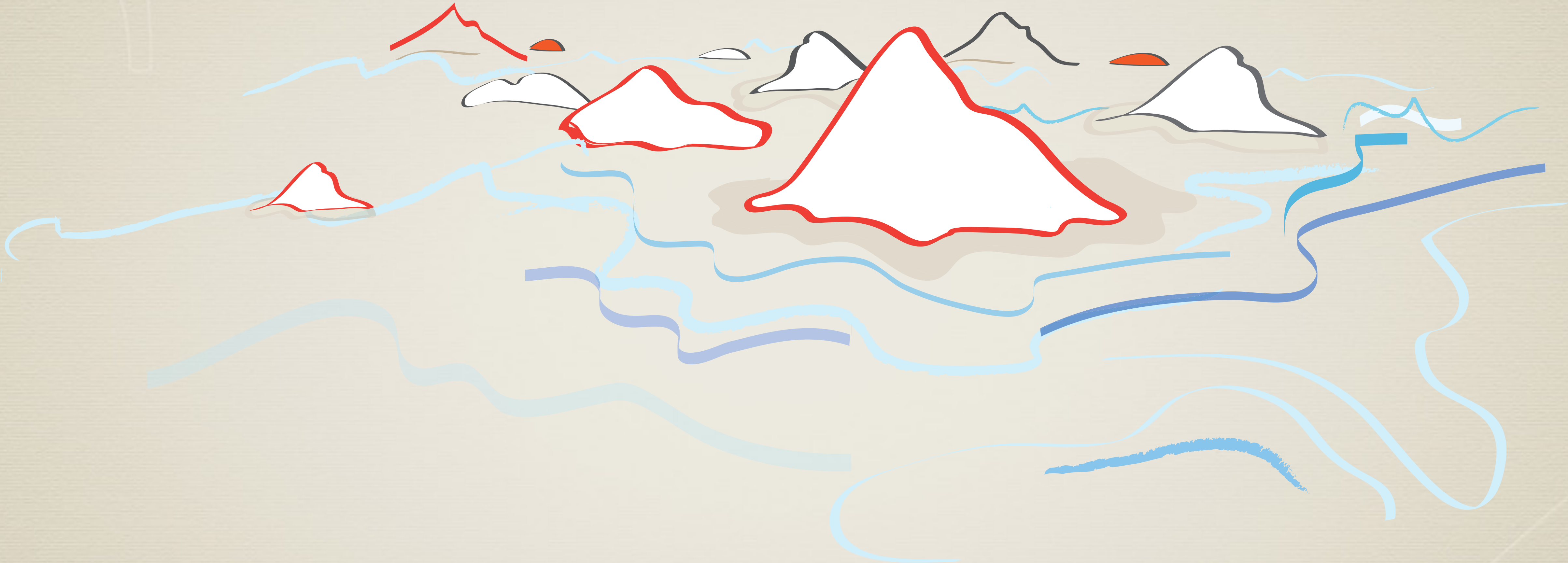
prototype view of traceroute RTTs in CAIDA's Charthouse

Future Work



prototype viz showing differences between a traceroute path and BGP AS paths

Thanks!



www.caida.org/projects/ark

For questions: ark-info@caida.org