IP Anycasting: Understanding routing impact of adding/removing instances

Project Summary:

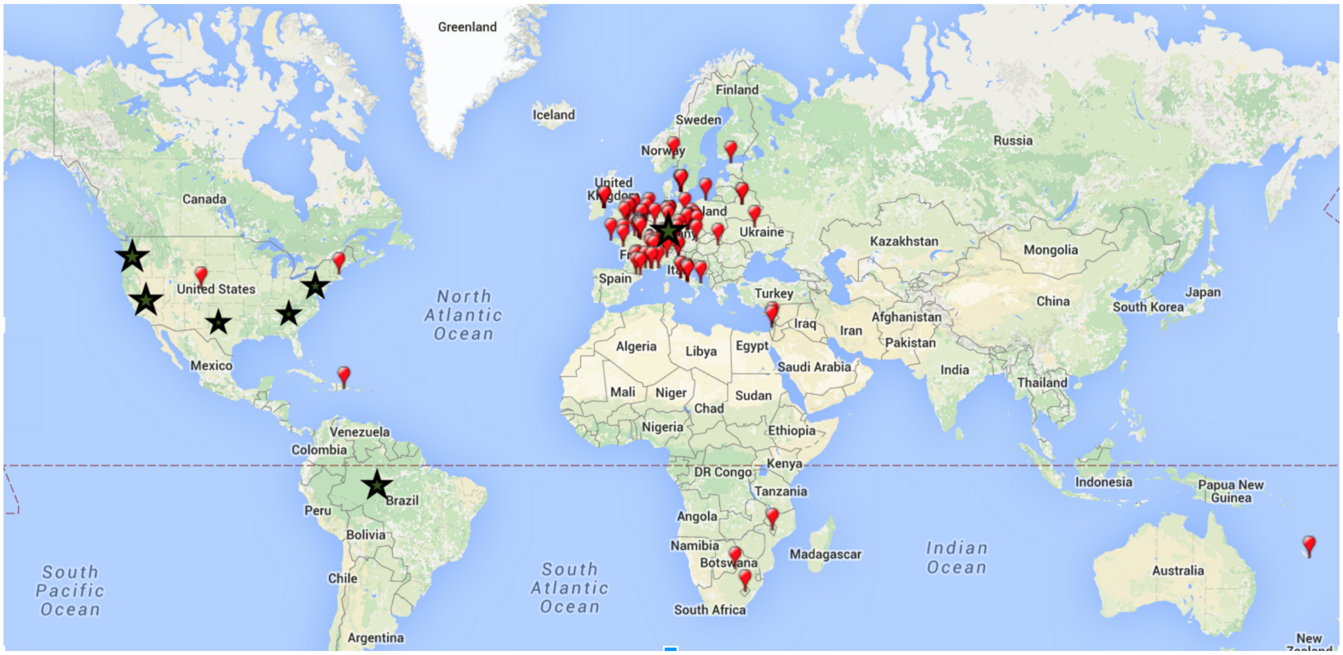
In this project, we focus on IP anycasting. With IP anycast, multiple machines can share the same IP address. When a request is sent to an anycasted IP address, routers will direct it to the machine on the network that is closest. Over the past year many CDNS implemented IP anycasting [1]. IP anycasting can hugely improve the speed, resiliency and attack mitigation. Companies that want to adopt any casting have the following:

- How should they decide how to configure their anycast?
- What will happen if a peer goes down?

Experiment Methodology:

Using the PEERING [2] platform, we simulate a service that hopes to benefit from anycasting. In short, we set up 7 muxes a.k.a peers for the chosen prefixes. We use RIPE atlas probes to repeatedly traceroute to the prefix from different geolocation. Then, we announce this prefix from the muxes. Wait for a while and then after sometimes we terminate the most popular muxe. We use both the traceroutes and BGP ripe collector(looking glasses) to monitor the changes.

Bellow is the google map of geolocation of the muxes(green stars) and Atlas ripe probes(red circles) where the traceroutes to muxes are originated from:



Any casted prefix: announce 184.164.241.0/24

Location of the muxes:

- Prepended AS, Node, Host, Type, IP
- 61574 ,amsterdam01, Amsterdam IX, IX, 100.126.4.3
- 61575, cornell01, Cornell University
- 61576, gatech01,Georgia Tech, University, 100.126.5.4

- 263842, isi01, ISI, University, 100.126.1.2
- 263844, seattle01, Seattle IX, IX, 100.126.0.4
- 263843, ufmg01, UFMG, University, 100.126.6.2
- 47065, phoenix01, Phoneix IX, IX, 100.126.3.2

Control Plane view:

Anycasting is fundamentally done on the control plane at the AS level per prefix. This means it's important to observe what happens during a routing change at the AS level. We made this easier by appending an unique AS number tied to each mux(Anycast node). This data is easily accessible from Looking glasses peered around the world. We chose to utilize RIPE's RRC(remote route collectors) as they are easiest to use during our limited timeframe but this controlplane information could easily come from bgpstream or other tools to allow time based views.

This was all done with the following steps:

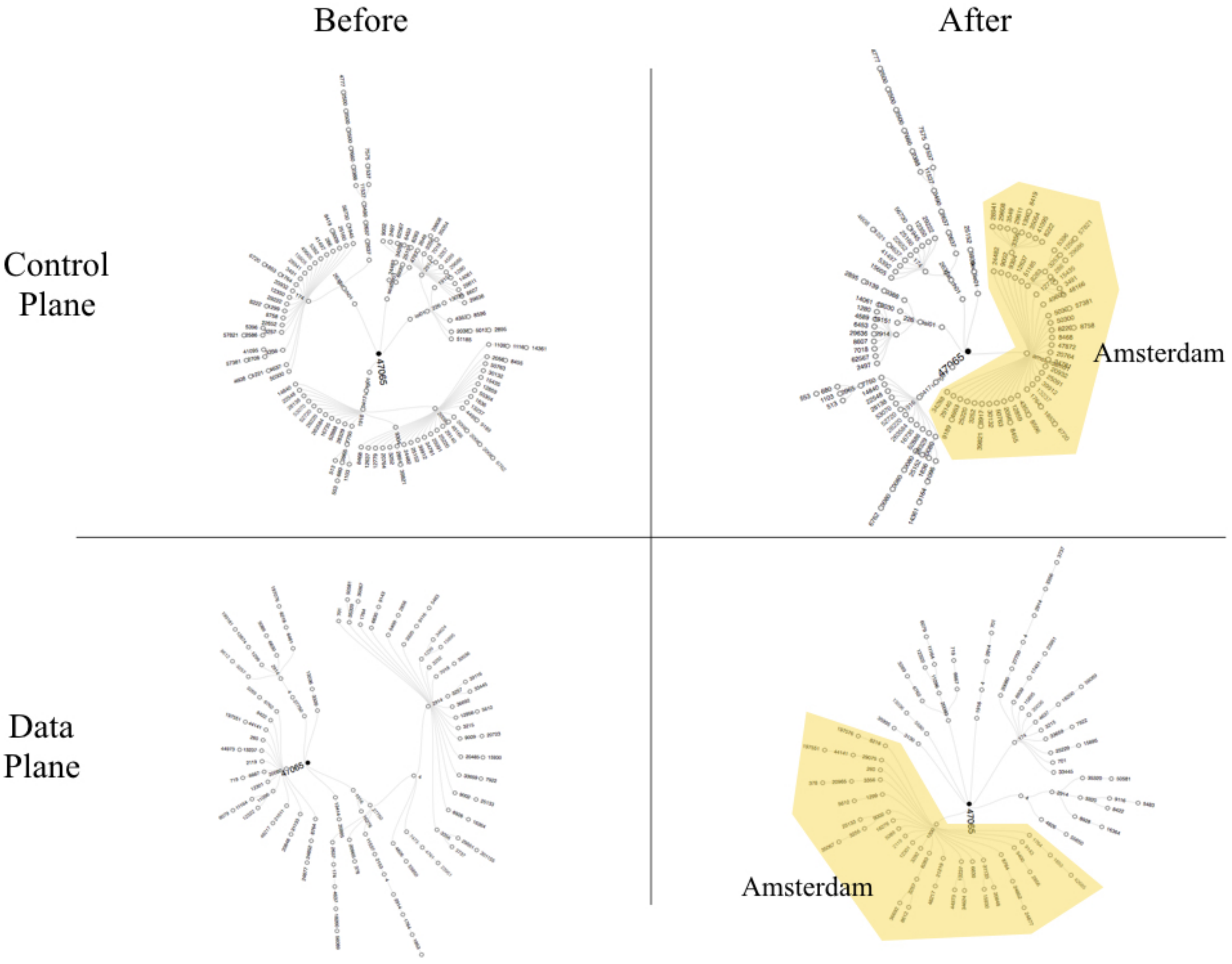
- Set up access to USC's PEERING
- Announce Routes to all mux(anycast nodes)
- Add prepending of unique AS per mux to tag muxes
- Retrieve and process Ripe's RRC(Remote Route collectors) in python
- Simple analysis of global convergence by looking at as-path using Ripe RRC using simple shell commands
    - Verified distribution of RRC peers to
- Decide most dramatic change would be withdrawal/annouce of amsterdam
- Take snapshot pre and post amsterdam announcements
- Process by reversing as path and removing extra AS47065 and translating mux tag AS to mux name.
    - We peer to PEERING testbed who then peers to internet peers putting as47065 on twice this would leave a useless level on the diagram
    - 25152 6939 47065 263844 47065 becomes 47065 seattle01 6939 25152
- Pass output to Visualization generation tools

Data Plane view:

Changes in the control plane usually have an effect on the data plane. In order to quantify these changes we perform periodic traceroute measurements, using RIPE Atlas. A direct comparison is not possible because traceroutes provide a more detailed view of the network than is possible when looking at the control plane. Therefore, to be able to compare the data plane with the control plane, we decrease the level of detail of the data plane measurement, from the IP-level to the AS-level. We do this by converting each hop in the traceroute on the dataplane to its corresponding AS number, and merging sequential AS numbers that are equal. Lastly, the output is converted to the format that is required for visualization using the D3-framework.

Results:

The figure below shows the AS-level mapping of vantage points (VPs) to a prefix announced by the PEERING system. The Before figures show the AS-level mapping without the anycast instance in Amsterdam. The After figures show the AS-level mapping after announcing the prefix from the Amsterdam instance of PEERING. We see a shift of routing that was before going to the USA to Amsterdam after the announcement. The Control Plane is created using data from RIPE RRC and the Data Plane with data from RIPE Atlas.



Data and Code:

All our codes and data are on github: http://github.com/darkfiberiru/anycast-1

Data clean up code:

- parse-prb-ids.py: parses JSON file and prints the IDs of all Atlas probes used in the traceroute measurement.
- peel-traceroute.py: parses JSON file and saves individual files per probe with their respective trace route measurements data.
- traceroute-as-hop.py: processes files created by peel-traceroute.py and generates a JSON for data visualization (data plane).
- traceroute-from-as.py: processes files with pre-processed RRC data and generates a JSON for data visualization (control plane).
- probe-extract-lat-long.py: extracting the lat,long,asn for all the machines involved in traceroutes from RIPE atlas,

Visualization code:

- plotaddrs_ip_lat_long.py : plotting the vantage points on google-map.
- D3: http://bl.ocks.org/mbostock/4063550

Team Members:

- Ricardo Schmidt, University of Twente

- Wouter de Vries, University of Twente
- Azzam Alsudais, University of Colorado at Boulder
- Roya Ensafi, Princeton University
- Nick Wolff, OARnet

[1] http://perso.telecom-paristech.fr/~drossi/index.php?n=Dataset.Anycast

[2] https://peering.usc.edu/about/

The doc version of the overview is athttps://docs.google.com/document/d/1xcjRaMVKEQeCmFxe_EgGDkJdMhZMYTg8Ln_FO2qkH0E/pub