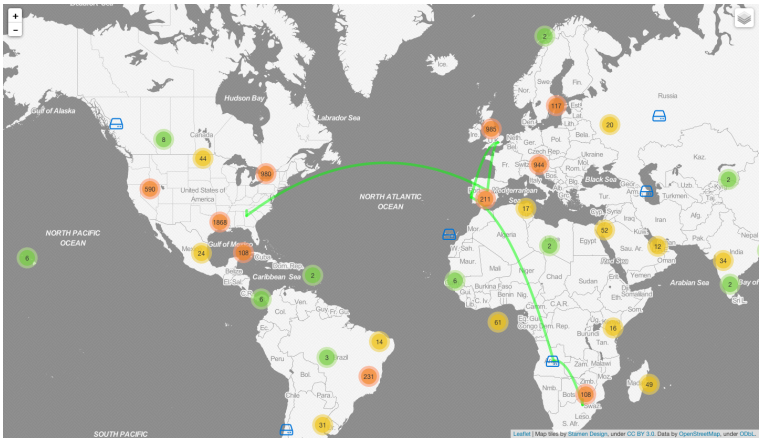
Neutrality Project

Network Graph Creation
Inference Algorithm
Inference Outcome Visualization

Eric Jollès and Pr. Katerina Argyraki

Network Architecture Lab, EPFL

Bachelor Semester 5



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

1 Introduction

The goal of this project was to read some traceroute measurements, create a graph of the network and apply an inference algorithm in order to guess if a link is neutral or not. All the scripts, files of this project are available on the GitHub repository.

2 Graph Creation (CreateGraphDb.py)

Repository file path "ReadData/".

2.1 How to use it

To use CreateGraphDb.py, execute the following command on a terminal:

```
$ python3 CreateGraphDb.py
```

There are in this script 3 important variables:

- the folder path of the measurements;
- the download speed threshold;
- the packet loss rate threshold.

Arguments can be passed to the script when it is executed in order to change these values. Table 1 gives more details about passing arguments in CreateGraphDb.py. Passing arguments works as follows:

```
$ python3 CreateGraphDb.py
-d measurements_data/json_neubot_raw
-s 50000
-p 0.0001
```

or

```
$ python3 CreateGraphDb.py
directory=measurements_data/json_neubot_raw
speed=50000
plr=0.0001
```

In both cases the default file is changed to measurements_data/json_neubot_raw the download speed threshold to 50’000 bits per second and the packet loss rate to 0.0001.

2.2 How it works

CreateGraphDb.py is a python script, which makes a network graph from traceroute measurements through 3 principal steps:

- Read the traceroute measurements and create a graph with the IPs and related information,

Parameter letter	Parameter name	Default value	Description
-d	directory=	Data/	Change the file location of the measurements.
-s	speed=	0	Change the lower-bound speed limit (in b/s) of the inference algorithm of the script.
-p	plr=	0.01	Change the upper-bound packet loss rate limit of the inference algorithm of the script.
-h	-	-	Display help.

Table 1: CreateGraphDb.py argument list

- Apply an inference algorithm to this graph;
- Write everything to the Database.

2.3 Measurements Reading

The measurements used in this project are contained in json files, which need to follow a specific pattern (See Table 2). You can see an example of a measurement on the Figure 1.

2.4 Inference Algorithm

When all graph information are collected, measurement anomalies, which can be linked to a non-neutrality, can be detected. For that a simple inference algorithm is used, which works as follows:

1. Computes the average loss packet and the average download speed rate of each link;
2. Compare each link with a threshold to see if it can be considered as neutral or not.

2.5 Database Creation

A database is used in this project in order to store the network graph and information linked. The script uses a MySQL database to store the graph

Json Keyword	Description
src_ip	Source IP of the measurement.
dst_ip	Destination IP of the measurement.
traceroute	Traceroute of the measurement. A router may not respond to traceroute requests in that case the traceroute must contain "***" in place of the missing routers. This router will be ignored. Thus the network graph can be disconnected.
timestamp_start	Beginning time of a measurement.
dt_s	Duration of the measurement (in ms).
download_speed	Download speed of the measurement in bits per second (bit/s).
transport_p	Transport protocol (TCP or UDP).
plr	Packet loss rate of the measurement.
packets_number	Number of packets sent during the measurement.

Table 2: Json Measurement Arguments

```
{
  "src_ip" : "31.162.4.123",
  "dst_ip" : "213.242.86.109",
  "traceroute" : [
          "213.242.86.106",
          "***",
          "213.242.86.107",
          "213.242.86.110"
    ],
  "timestamp_start": "2018-10-17T00:03:39Z",
  "dt_s" : "0.170701737734" ,
  "download_speed" : "80804.6331529",
  "transport_p":"TCP",
  "plr":"0.009689987087647137",
  "packets_number": "10000"
}
```


Figure 1: Json Measurement Template

created by the measurements. You can easily use your own database by changing the database field in the main part of the script.
The database is composed of 2 tables :

- The Edges table contains all edges of the graph and their related information (for more details see Table 3). Here is a sample of the Edges table in the database (Figure 2).

#	begin_ip	end_ip	neutral	paths	packet_loss_rate	speed	packet_number
122	62.115.137.167	213.248.68.63	1	1144	0	2981180	20390
123	195.2.2.154	195.2.24.246	0	754	0.0218579	47298.5	366
124	62.115.151.47	216.239.62.163	1	241	0.000370828	1030190	8090
125	68.86.95.238	162.151.13.130	1	69,624,1269	0.0000146396	4646910	68308
126	202.158.194.173	64.124.200.233	1	132,186,472,526,967,1007,1147,1...	0.000111868	3609540	107269

Figure 2: Sample of the Edges table

- The Vertices table contains all vertices of the graph and their location (latitude, longitude) (for more details see Table 4). Here is a sample of the Vertices table in the database (Figure 3);

#	ip	latitude	longitude
1	96.34.2.213	33.9534	-117.396
2	193.56.145.226	40.4172	-3.684
3	4.68.62.18	51.5085	-0.12574
4	193.251.128....	48.8582	2.3387

Figure 3: Sample of the Vertices table

MySQL keyword	Description
begin_ip	Origin IP of a link.
end_ip	Destination IP of link.
neutral	Result of the inference algorithm, 1 if the link is neutral 0 otherwise.
paths	List of all paths, which pass through this link (each path is represented by a unique id).
packet_loss_rate	Average packet loss rate of the link.
speed	Average download speed of the measurement in bits per second (b/s).
packet_number	Total number of packets, which pass through this link.

Table 3: Data stored in the Edges table of the database

MySQL keyword	Description
latitude	Latitude of a Vertex (router).
longitude	Longitude of a Vertex (router).

Table 4: Data stored in the Vertices table of the database

2.6 IP-Location Dataset

During the database creation each router (vertex) is linked to its location (latitude and longitude). For this a dataset, which translates IP to location, is needed. This dataset should be stored on the same directory as the script (not available in the GitHub repository, please download it locally). Any IP-location dataset can be used by changing the implementation of the function get_lat_lon_from_ip.
In this project 2 datasets are used:

- IP2Location : an accurate dataset with some limitation (only IP addresses between 0.0.0.0 an 100.0.0.0 are accessible) ;
- Geoip2 : dataset used for other IP addresses .

You can download these datasets on the following links:

- https://www.ip2location.com/developers/python ;
- https://dev.maxmind.com/geoip/geoip2/geolite2/ .

These datasets must be updated regularly since the network topology always changes (a router can appear, be removed, . . .).

2.7 Required Python Libraries

Several python libraries are used in this script. They must be installed before using the script. They can be installed from the terminal (using pip):

- pip install mysql-connector-python-rf ;
- pip install geoip2 .

For the IP2Location database the library must be downloaded following the link steps:

- https://www.ip2location.com/developers/python .

3 Graph Visualization (NeutGraph.html)

Repository file path: "GraphVisualization/main/html/".
Website to test it: "https://ejolles.github.io/GraphVisualization/".

3.1 Requirements

NeutGraph.html is a component that visualizes a graph with an inference outcome. CreateJS.py (section 4) will create 2 files, which can then be used by NeutGraph.html:

- edges.js : this file contains information about edges of the graph. This file must follow this format to be read by the Graph-Visualization application:

```
{
    "ipb":"72.52.92.198",
    "ipe":"184.105.65.245",
    "neut":"1",
    "paths": ["865", "1395"]
}
```

  where "ipb" is the source ip, "ipe" is the destination ip, "neut" is to 1 if the edge is neutral 0 otherwise and "paths" is a list containing all paths, which pass through the link.

- vertices.js : this file contains information about vertices of the graph. This file must follow this format to be read by the Graph-Visualization application:

```
{
    "ip":"66.110.32.102",
    "lat":"37.3861",
    "lon":"-122.084"
}
```

  where "ip" is the IP of the vertex, "lat" is the latitude of the vertex and "lon" is the longitude of the vertex.

3.2 Functions

Some key points about the tool are:

- To avoid displaying all vertices at the same time, they are clustered to show only useful data (Figure 4).

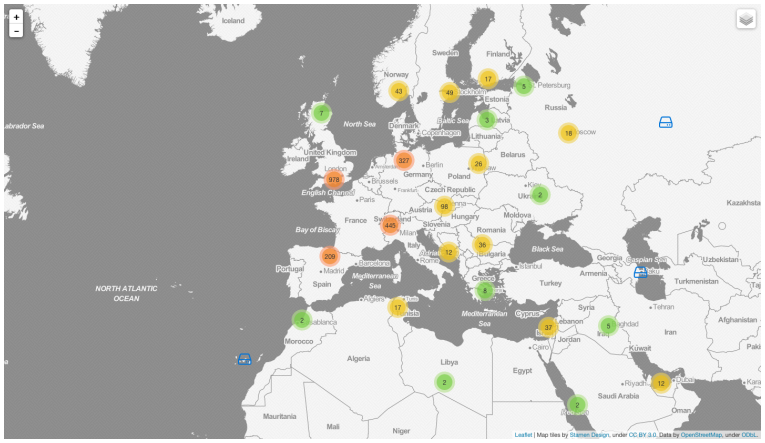


Figure 4: Cluster vertices

- A link which is suspected to be non-neutral is displayed in red, otherwise it is displayed in green (Figure 5).

Figure 5: Neutral links are displayed in green, non-neutral in red

- Clicking on a router shows all links related to it (Figure 6).

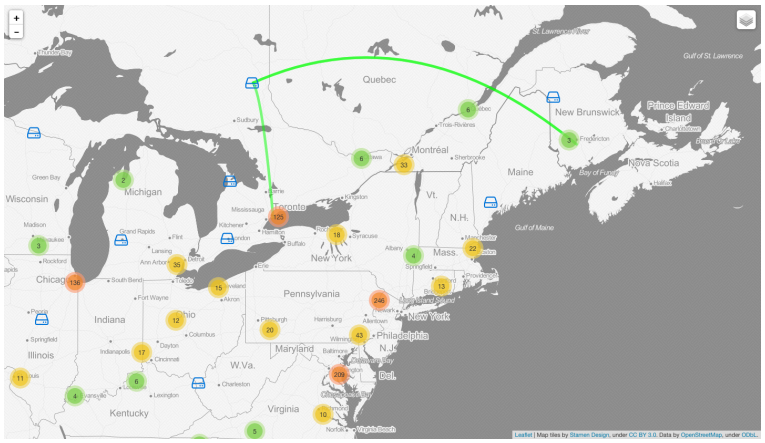


Figure 6: Clicking on a router shows links related to it

- Positioning the mouse on a vertex shows its IP (Figure 7).

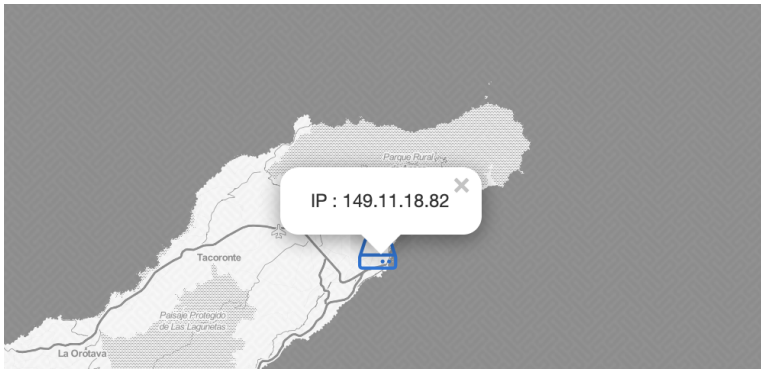


Figure 7: Positioning the mouse on a router shows its IP

- Once an edge is displayed, positioning the mouse on a link shows useful information (source IP, destination IP, neutral value) (Figure 8).

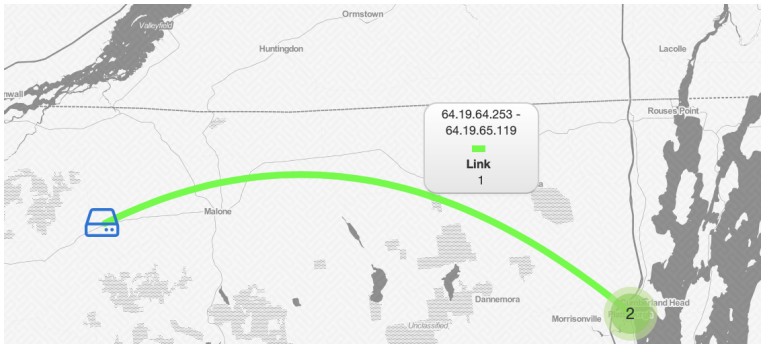


Figure 8: Positioning the mouse on a link shows useful information

- Clicking on an edge shows all paths passing through this edge (Figure 9).

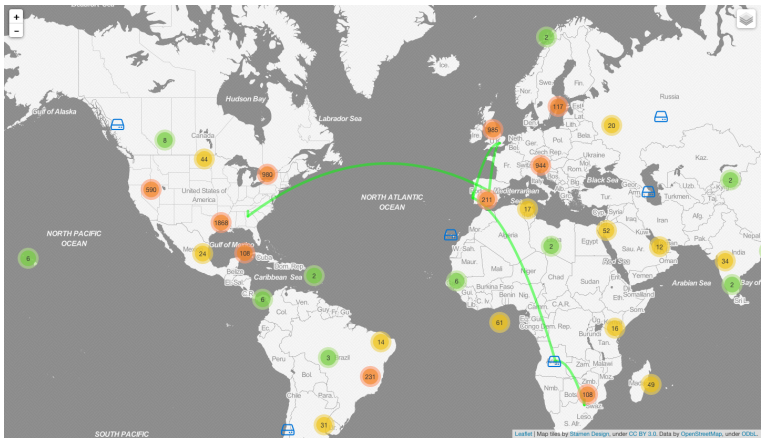


Figure 9: Clicking on an edge shows all paths related to it

- Using the right top menu allows to choose if routers must be displayed or not (Figure 10).

Figure 10: Display option of the vertices

- If there is router overlapping (same position), they are disposed in spiral and can be used in the same way as other routers (Figure 11).

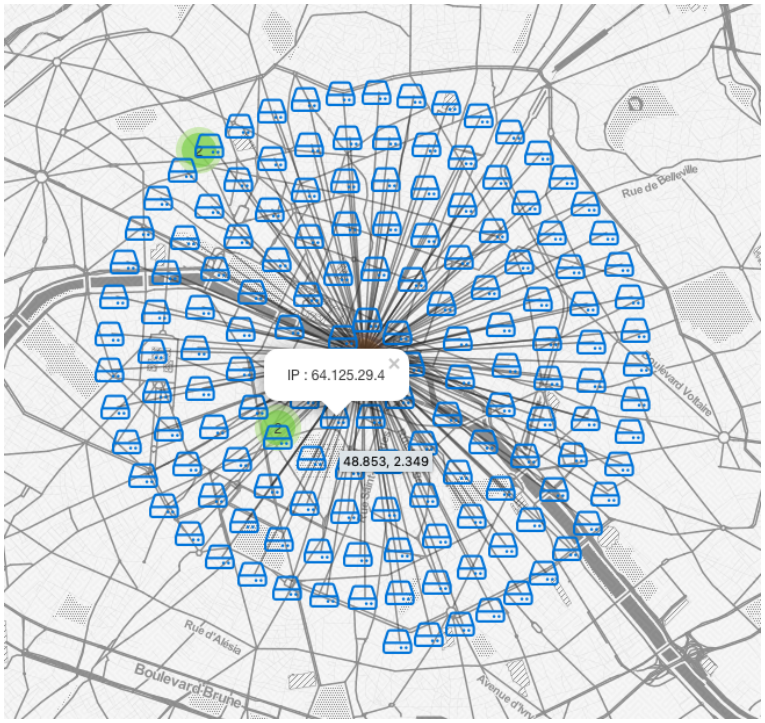


Figure 11: Displays routers in spiral when overlapping appears

4 Create JavaScript documents for Graph Visualization (CreateJS.py)

Repository file path: "GraphVisualization/main/data/".

CreateJS.py is an adapter, which reads data from the database and creates 2 JavaScript files with it (edges.js and vertices.js). These files will be used by the Graph-Visualization application to display the map (section 3).

To use it, the following command on a terminal must be executed:

```
$ python3 CreateJS.py
```

edges.js and vertices.js will be created in the same folder as the script.

5 Limitations

5.1 IP-Location Datasets

A major problem for this map visualization is to find the location of a specific IP. The location can be deduced by knowing the ISP, the router name, some hidden name etc. This is a work that is beyond the scope of this project and for this reason some existing datasets are used. But IP-location databases are not complete, and for some unknown IP they give default location (Chenney Reservoir, earth central point).

5.2 Inference algorithm

Saying that a link violates neutrality is a difficult task, and the inference rule used in this project is not very sophisticated. A lot of false positives and false negatives happen. Since there are no sample measurements of links that violate neutrality, determining a threshold for this algorithm is not an easy task.