Visualizaing the Internet Topology

Anshul Sangrame (CS21BTECH11004)
Gautam Singh (CS21BTECH11018)
Varun Gupta (CS21BTECH11060)

Contents

This document contains a report on the Python library `netvis` created by the authors, to visualize the topology of the internet. It covers some implementation details, challenges in implementation and conclusions from using this library.

1 Implementation

`netvis` is a `python` package, installable using `pip`. In the following subsections, we describe the installation and sailent features provided by this library.

The entire source code is hosted on GitHub. The verbose implementation details are provided as comments in the source code, which have been omitted here for brevity.

1.1 Installation

`netvis` can be installed by typing the following in a terminal window.

```
pip install -e "git+https://github.com/
   goats-9/cs3530-assignments.git#egg=
   netvis&subdirectory=netvis"
```

Alternatively, the library can be installed by cloning the aforementioned GitHub repository and issuing the following command in the `netvis` directory.

```
pip install -e .
```

It is *highly recommended* to install the package in a Python virtual environment.

1.2 Requirements

To help in the visualization of the topology of the Internet, `netvis` depends on the following Python packages.

1) `pandas`. To process the raw data obtained for visualization.
2) `openpyxl`. To handle raw data in Excel files.
3) `pyvis`. To visualize the network in the form of a graph.

Additionally, note that this library is suitable for Linux systems only, and also requires the executable `mtr` to be installed on the system.

1.3 Features

The sailent features of `netvis` are the following.

1) A `NetGraph` object to abstract the topology of the Internet into a directed graph, containing a list of nodes and edges.
2) Use of `mtr` (short for *my traceroute*) with appropriate arguments to obtain the entire sequence of hops from source to destination, along with AS numbers of ISPs (if available) in CSV form for easy data handling.
3) Provisions to save and load `NetGraph` objects to and from Excel files so that they can be used in other Python scripts.
4) Provision to create a union of two `NetGraph` objects so that the topology from multiple sources to multiple destinations is *integrated*.
5) Rendering of the graph depicting the topology in HTML, along with the use of CSS and JS to make the graph interactive as well as show the legend. The graph is present in `graph.html`,

but the integrated legend is displayed on rendering `index.html`.

2 Challenges

The authors wanted to automate the entire process of visualizing the internet topology. The following challenges were faced in the process.

1) Whenever `netvis` is imported, a dataset about 30 MB in size is downloaded and loaded into a `DataFrame` object. This may not be suitable for systems with constraints on resources like memory or internet speed (such as embedded systems).
   A more suitable solution could have been hosting the data on a webserver and making HTTP GET requests to a URL.
2) `pyvis` does not have a provision to provide a legend for easy use of the generated graph. The authors worked around this using JavaScript to create a legend.
3) The outputs of `mtr` did not include the source IP address. It was manually added by making use of the Python standard library module `socket`.

3 Observations

The following conclusions can be made based on the data collected.

1) From the raw data, sites hosted in countries east of India have lesser RTTs (less than 200 ms) compared to sites hosted in countries to the west of India (at least 200 ms).
2) The maximum RTT on average was experienced while pinging the site `hi.is`, which makes sense since the site was hosted in Iceland. On the other hand, the minimum RTT was experienced while pinging `www.isro.gov.in`, which is hosted in India, where all the source devices are.
3) a) Using the LAN service available in IITH (source IP `10.5.80.104`), the packets always travelled through the ISP at IITH, as expected.
   b) Using mobile data, the packets always went through Airtel, since both SIM cards were registered to Airtel.
   c) Using personal WiFi network (not in campus, source IP `192.168.1.38`), the packets went through CtrlS datacenters, which might mean that the local ISP uses these datacenters to host and provide Internet services to their customers.
4) By looking up the AS numbers, most of the sites chosen as destinations are hosted in the USA (even those with non-American domains).

4 Resources

1) The raw and processed data is present in the `data` directory.
2) The scripts used to collect and visualize the data are present in the `tests` directory.
3) The ASN dataset `data/asn.tsv` was taken and edited from `asntool.com`.
4) The hatch build system was used to package the code.