ver 2.0

traIXroute

NAME

traIXroute [1] — A tool that detects if and where a traceroute path crosses an Internet Exchange Point (IXP).

SYNOPSIS

traIXroute [-h] [-u] [-m] [-db] [-asn] [-dns] [-rule] [-o *file*] [probe *options*] [ripe *options*] [import *options*]

DESCRIPTION

traIXroute is a python 3 tool that infers IXP hops on traceroute paths by applying IXP detection rules using data provided by PeeringDB (PDB) [6], Packet Clearing House (PCH) [5] and CAIDA [7]. The user can specify either a single destination IP address or a list of addresses in an input file (*batch execution)* to probe. **traIXroute** can be configured to use in the background either the well-known traceroute tool [4] or scamper [2], a more sophisticated tool provided by CAIDA, which implements the Paris traceroute technique [3]. **traIXroute** first probes a path using traceroute or scamper. Next, it seeks for an IXP IP address in the path and applies the IXP detection rules in a sliding window fashion to deduce, if possible, the exact hop (i.e. before or after the IXP IP address) where the IXP has been crossed. **traIXroute** does not support yet IPv6.

Available **traIXroute** options:

-h

Prints a list of the command line options with a synopsis for each one.

-u

Downloads all the necessary files for the IXP detection rules (see section Datasets). The tool automatically enables this option in case the files are missing.

-m

Exports the database to the files *ixp_membership.txt* and *ixp_prefixes.txt.* For further information see the Datasets section.

-db

Enables printing information about the constructed database of IXPs based on the datasets' import.

-asn

Enables printing the ASN for each IP hop in the traIXroute output (see section Output).

-dns

Enables printing the FQDN of each IP hop in the traceroute path.

-rule

Enables printing the hit IXP detection rule(s) in the *IXP Hops* output (see section Output) in a traceroute path.

-o file

Specifies the name of the output file that contains all traIXroute's results. Otherwise, the default output file name (i.e. *output_Y_M_D-h_m_s.txt*) will be used.

probe *options*

The *probe* command with the available options (see below) are used to send probes to certain destination(s) to receive IP path(s). **traIXroute** can send probes either using the scamper [8] or the traceroute tool [9]. The available options for the *probe* command **in traIXroute** are:

-s [scamper options]

Sets scamper as the default probing mechanism for **traIXroute**. The tool is configured to use scamper with the **-trace** option only (see scamper's documentation). All the available **-trace** options can only be used in double quotes after ***-s*** traIXroute's argument (see *Examples*). For further information about the available options check scamper [8].

-t [traceroute options]

Sets traceroute as the default probing mechanism for **traIXroute**. All the traceroute options can only be used in double quotes after **-t** traIXroute's argument. For further information about the available options check traceroute [9].

-dest IP/FQDN

Specifies the destination IP address or the fully qualified domain name (FQDN) to probe.

-doc file

As an alternative way to send probes to multiple destinations, the user can import a .txt file with a list of IP addresses or FQDNs, following the format shown below. **traIXroute** will probe all the destinations from the imported file.

…
IP1/FQDN1
IP2/FQDN2
IP3/FQDN3
…

ripe *options*

The *ripe* command with the available options (see below) are used to create traceroute-based RIPE Atlas [12] measurements or fetch traceroute results directly from the RIPE Atlas database [12]. Check RIPE Atlas Cousteau documentation [13] for the available arguments for each case. The available options for the *ripe* command **in traIXroute** are:

-c [RIPE Atlas arguments]
Creates a new measurement in the RIPE Atlas platform with the desired options as specified in the arguments. The default RIPE Atlas argument values **in traIXroute** are:

{ 'target': 'www.inspire.edu.gr',
'type': 'traceroute',
'protocol': 'ICMP',
'resolve_on_probe': True,
'is_oneoff': True,
'af': 4,
'description': 'traIXroute (IPv4)'
}

{ 'type':'probes',
'Value':'23385',
'Requested':1
}

-r [RIPE Atlas arguments]
Fetches the traceroute results from the RIPE's database of a RIPE Atlas measurement to detect IXP crossing links. The results can be filtered by start, end time and probes.

import *options*
The *import* command with the available options (see below) are used to import json based files with traceroute paths. The available options for the *import* command **in traIXroute** are:

-json file
Imports a list of traceroute paths from a traIXroute format (json based) file to detect IXP crossing links (check the directory *Examples/* in traIXroute's repository for examples).

-ripejson file
Imports a list of traceroute paths from a RIPE Atlas json format file to detect IXP crossing links.

EXAMPLES
Some examples on how to use the tool are described below. Depending on the configuration of the operating system, "sudo" might not be required.

- traIXroute uses scamper with the option max ttl 12 towards the destination inspire.edu.gr:
  $ python3 traIXroute.py probe -dest inspire.edu.gr -s="-m 12"

- traIXroute uses traceroute towards the destination 8.8.8.8 printing the ASNs of each IP hop:
  $ python3 traIXroute.py -asn probe -t -dest 8.8.8.8

- traIXroute uses traceroute with ICMP ECHO probes towards destinations read from an input file *input.txt*.
  $ python3 traIXroute.py probe -doc input.txt -t="-I"

- traIXroute rebuilds the database with up-to-date data (if available), uses traceroute towards the destination 192.198.10.10, and writes the output to a file with name “myoutput.txt”.
  $ python3 traIXroute.py -u -m probe -dest 192.198.10.10 -o myoutput.txt -t

- traIXroute fetches traceroute data from a RIPE Atlas measurement with measurement ID “6889889” and probe ID “14846”.
  $ python3 traIXroute.py ripe -r '{"msm_id": 6889889, "probe_ids":[14846]}'

- traIXroute creates a new RIPE Atlas measurement with default arguments.
  python3 traIXroute.py ripe -c '{}' '{}'

- traIXroute creates a new RIPE Atlas measurement with traceroute destination “inspire.edu.gr”, probe ID “14846” and the rest arguments in their default values.
  python3 traIXroute.py ripe -c '{"target":"inspire.edu.gr"}' '{"value":14846}'

OUTPUT

A sample of **traIXroute**’s output is shown below running the following command:
$ python3 traIXroute.py -u -m -asn -rule -db -dns probe -dest inspire.edu.gr -s

→ Before traIXroute starts probing, updates all the datasets, and exports the database in two distinctive files:

Updating the database...
Started downloading PDB dataset.
Started downloading PCH dataset.
Started downloading RouteViews dataset.
Routeviews has been updated successfully.
PDB dataset has been updated successfully.
PCH dataset has been updated successfully.
Database has been updated successfully.

The files ixp_prefixes.txt and ixp_membership.txt have been created.

→ Afterwards, it initializes its database with the IXP detection rules and the up-to-date IXP Membership and IXP Prefix Data. **However, it is worth mentioning that when traIXroute runs for the first time in a local machine, the tool exports the database organized in json format files. This serves to quickly load the database for the next traIXroute probes:**

Imported 9 IXP Detection Rules from Rules.txt.
Loading from PCH, PDB, Routeviews and additional_info.txt.
Imported 16 Reserved Subnets.
Extracted 0 IXP IPs from additional_info.txt.

Extracted 0 IXP Subnets from additional_info.txt.
Extracted 16038 IXP IPs from PDB.
Extracted 3846 IXP IPs from PCH.
Extracted 456 IXP Subnets from PDB.
Extracted 358 IXP Subnets from PCH.
Extracted 15977 no dirty IXP IPs after merging PDB, PCH and additional_info.txt.
Extracted 1094 dirty IXP IPs after merging PDB, PCH and additional_info.txt.
Extracted 571 IXP Subnets after merging PDB, PCH and additional_info.txt.

→ Prints all the configuration parameters of the selected traceroute tool (i.e., scamper):

traIXroute using scamper with default options.

→ Prints the traceroute path enriched with AS and IXP information for each hop, resolving the domain name for every IP address.

traIXroute to inspire.edu.gr (139.91.152.72)
1) AS 192.168.1.1 (192.168.1.1) 0.482 ms*
2) AS1241 bbras-llu-her-01L500.forthnet.gr (213.16.246.[redacted]) 30.750 ms
3) AS1241 213.16.247.[redacted] (213.16.247.X) 30.683 ms
4) AS1241 te0-4-0-11.core-kln-13.forthnet.gr (194.219.199.[redacted]) 41.178 ms
5) AS1241 distr-kln-02Be2.forthnet.gr (213.16.247.[redacted]) 37.480 ms
6) AS1241 core-kln-12Be3.forthnet.gr (213.16.247.[redacted]) 40.440 ms
7) GR-IX->AS5408 grnet.gr-ix.gr (176.126.38.1) 39.864 ms
8) AS5408 forth-her-4.eier.access-link.grnet.gr (62.217.98.[redacted]) 45.995 ms
9) AS * (*) -*
10) AS * (*) -*
11) AS * (*) -*
12) AS * (*) -*
13) AS * (*) -*

→ Prints the IXP hop(s), if there exist, following the format:
IXP detection rule number --- Hop number) IP Address (AS number) - IXP Name - Hop number) IP Address (AS number).
For instance, in the output below, Rule 1 of the IXP detection rules infers an IXP crossing (GR-IX) between hops 6 and 7 (AS1241 and AS5408, respectively).

IXP Hops:
Rule: 1 --- 6) 213.16.247.[redacted] (AS1241) <--- GR-IX ---> 7) 176.126.38.1 (AS5408)

DATASETS

traIXroute utilizes three types of data to detect IXPs [1], which can be updated from the command line automatically:

1. IXP Membership Data
2. IXP Prefix Data
3. Routeviews Prefix to AS mappings

Particularly, the first two datasets are retrieved from the PDB and PCH to detect IXP IP addresses in an IP path. These are merged by matching IXP IP addresses, prefixes and names since they do not use consistent identifiers for IXPs. The third dataset is used to map IP addresses to ASes. Such mappings might be wrong [11], therefore **traIXroute** does not consider this evidence alone conclusive to infer IXP crossings in traceroute paths.

During the merging process, the tool finds and filters out inconsistencies between the two datasets:

1. Excludes IXP IP addresses that are assigned to multiple ASes.
2. Excludes IXP IP addresses without ASN information.
3. Excludes IXP IP addresses without IXP name identifiers.
4. Excludes IXP IP addresses and prefixes that belong to the Reserved IP Subnets [10].
5. Excludes IXP prefixes without IXP name identifiers.

However, some controversial entries remain in the database. Instead of deducting those entries, we keep and mark this information as **"dirty"**. Specifically, they are:

1. IXP Prefixes with multiple IXP name identifiers.
2. Sub-prefixes with different IXP name identifiers compared to their corresponding prefixes.
3. IXP IP addresses with multiple IXP name identifiers.

Since under the IXP detection process, **traIXroute** also utilizes the dirty part of the database, when an IP hop in the traceroute path hits a dirty IXP IP address or Prefix a **"?"** notation is used at the beginning of the row in the traIXroute output.

For convenience, when the first two datasets have been finalized and sanitized, all the IXP Membership and Prefix Data are exported into two files, *ixp_membership.txt* and *ixp_prefixes.txt*. Note that each entry in both files is determined as *no dirty* or *dirty* or *additional* (coming from the *additional_info.txt* file) using the notations **"!"**, **"?"**, **"+"** respectively at the beginning of each line. Note that for only dirty entries, in both files, there exists IXP information from PDB and PCH datasets.

ADDITIONAL FILES

traIXroute also uses the *additional_info.txt* file which contains IXP related data (prefixes and/or IP addresses) specified by the user. The file includes one IXP Membership or IXP Prefix data-entry per line. User can put data in the *additional_info.txt* file to either add fresh information in the database or overwrite existing entries in PDB and PCH data. If one entry in the file does not exist in the PDB and PCH datasets, then it is included in the database. Otherwise, it overwrites the existing entry. **However, in case of adding a sub-prefix, its corresponding prefix is removed from the database.**

IXP DETECTION RULES

To detect an IXP crossing link in the traceroute path, **traIXroute** applies various heuristic rules over the IP path in a sliding window fashion. The length of the window is set to two or three IP addresses depending on the rule. IXP detection rules can be appended in the *rules.txt* file (one rule per line). If detection rules have not been specified in the file, the tool cannot infer an IXP crossing. They are applied in the order they appear in the file. This means that the first rule in the file has the highest priority and so on.

IXP detection rules can be defined with a specified grammar, which allows to compose new rules. However, the tool is pre-configured with the rules proposed and evaluated in [1]. Each rule consists of the left (condition) and the right (assessment) part. The left part includes the conditions of the rule hops under which an IXP crossing is detected in a path, and the right assesses between which hops the IXP crossing takes place. The available keywords for defining IXP detection rules are the following:

- **IXP_IP :** denotes that the rule in the condition part requires an IXP IP address at the certain hop. This keyword is only used with **AS_M** and **!AS_M** notations, i.e., **IXP_IP and AS_M** to detect an IXP IP address in the trace based on the IXP Membership data while **IXP_IP and !AS_M** based on IXP Prefixes data respectively.
- **AS_M :** denotes that the rule in the condition part requires an AS to be IXP member. If the **AS_M** is used with the **IXP_IP** keyword (e.g., *IXP_IP and AS_M0*), then the AS should be member to the respective IXP at the certain hop. Otherwise, when the **AS_M** keyword is used as the only condition in a rule hop (i.e., without an **IXP_IP**), then the rule expects the AS, mapped from an IP based on Prefix-to-AS mappings, to be member to the candidate IXPs of the previous/next hops in the rule (e.g. rule example 7).
- **'!' :** is used as a 'not' operator to reverse the condition. This can be used only with the **AS_M** keyword. When the notation **!AS_M** is used with the **IXP_IP** (e.g., *IXP_IP and !AS_M*, as described earlier) denotes an IXP IP, detected based on IXP Prefixes data, while if it is used as the only condition in a certain rule hop, it determines an AS as a not IXP member.
- **a :** specifies that the IXP crossing link is located between the first two hops of the sliding window.
- **b :** specifies that the IXP crossing link is located between the last two hops of the sliding window. This can only be used with the rules of window size 3.
- **and :** can be used either in the condition or the assessment part of the rule. In the condition part, it is used to build a condition set for a certain hop (e.g., *IXP_IP and !AS_M*). If used in the assessment part, the rule must have window size 3 and it specifies that there exist IXP crossings between the first and the last two hops (e.g., *a and b*).
- **or :** can only be used in the assessment part of the rule to specify the IXP crossing link.
- **'-' :** separates the conditions of each hop in the condition part of the rule (e.g., *condition - condition - condition : assessment*).
- **'( )' :** determines a condition set (e.g., *condition - (IXP_IP and !AS_M) - condition : assessment*) in the condition part of the rule.
- **':' :** separates the condition from the assessment part of the rule (e.g., *rule_conditions: rule assessment*).
- To formulate rules that require more than one (different) IXPs and/or (different) AS members in the condition part, the user can concatenate to the keywords **IXP_IP** and **AS_M**, same or different numbers (i.e., **0, and/or 1, and/or 2**) that allows to differentiate between IXPs (e.g., *(IXP_IP0 and AS_M) - (IXP_IP1 - AS_M)*) and AS members (*AS_M0 - AS_M1 - (IXP_IP and !AS_M)*) (see rule examples 4 and 5).

RULE EXAMPLES

Examples of how to formulate IXP detection rules are proposed below. Some of the following rules are only used for demonstrating possible ways to syntax the rule keywords. **This means that their assessment does not necessarily infer a valid IXP crossing.**

1. **AS_M0 - (IXP_IP and AS_M1) - AS_M1: *a***

This rule examines a window of size three. (1) It requires one IXP IP in the middle hop and (2) AS based information for all the three hops, for the middle one based on IXP Membership data while for the border ASNs based on Prefix-to-AS mappings. (3) The ASN in the first hop should be different from the next two ASNs, since the first concatenated keyword number, *'0'*, differs from the one in the next two hops (i.e., *'1'*). (4) All the three ASes are required to be members of the candidate IXP in the middle hop. Its assessment is an IXP crossing in link *a*, i.e., in the first hop.

2. **AS_M0 - (IXP_IP and !AS_M) - !AS_M1: *a***

This rule examines a window of size three. (1) It requires one IXP IP in the middle hop based on IXP Prefixes data, (2) AS information only for the border IPs and (3) the two border ASNs must be different. (4) In contrast with the previous rule, only the first AS must be member of the candidate IXP of the middle hop. Its assessment is an IXP crossing in link *a*, i.e., in the first hop.

3. **AS_M0 - (IXP_IP and AS_M1) - AS_M2: *a or b***

This rule examines a window of size three. (1) It requires one IXP IP in the middle hop, (2) AS information with (3) different ASNs for all the three hops and (4) all the three ASes to be members of the candidate IXP of the middle hop. Its assessment is an IXP crossing in hops *a* or *b*.

4. **(IXP_IP0 and AS_M0) - (IXP_IP0 and AS_M1): *a***

This rule examines a window of size two. (1) It requires two different IXP IPs (2) from the same IXP subnet (3) with AS information based on IXP Membership data. (4) The rule specifies different ASNs between the two hops since there are different numbers (i.e., *0 and 1*) concatenated with the AS_M keywords. Its assessment is an IXP crossing in link *a*, i.e., in the first hop.

5. **(IXP_IP0 and AS_M) - (IXP_IP1 and AS_M): *a***

This rule examines a window of size two. (1) It requires two different IXP IPs for both IPs (2) from different IXPs (3) with AS information based on IXP Membership data. (4) The rule does not compare the ASNs of the two IPs, in contrast to the previous rule. This means that both ASes might be the same. Its assessment is an IXP crossing in link *a*, i.e., in the first hop.

6. **(IXP_IP0 and AS_M0) - (IXP_IP1 and AS_M1) - (IXP_IP2 and AS_M2): *a and b***

This rule examines a window of size three. (1) It requires IXP IPs for all the three hops (2) coming from three different IXPs (3) with AS information based on IXP Membership data, and (4) different ASNs for all the ASes. Its assessment is two consecutive IXP crossings in hops *a* and *b.*

7. **(IXP_IP0 and AS_M0) - AS_M - (IXP_IP1 and AS_M1): *a or b***

This rule examines a window of size three. (1) It requires two IXP IPs in the border IPs based on IXP Membership data (2) from different IXP subnets and (3) AS information for all the three hops. However (4) the ASN in the middle hop is not compared with the rest ASNs (i.e., it can be either the same or different; this would not affect the rule assessment). (5) Also, the AS in the middle hop has to be member in both IXPs. Its assessment is an IXP crossing in hops *a* or *b*.

MODULES

A detailed representation of the modules of **traIXroute** is shown in Figure 1.

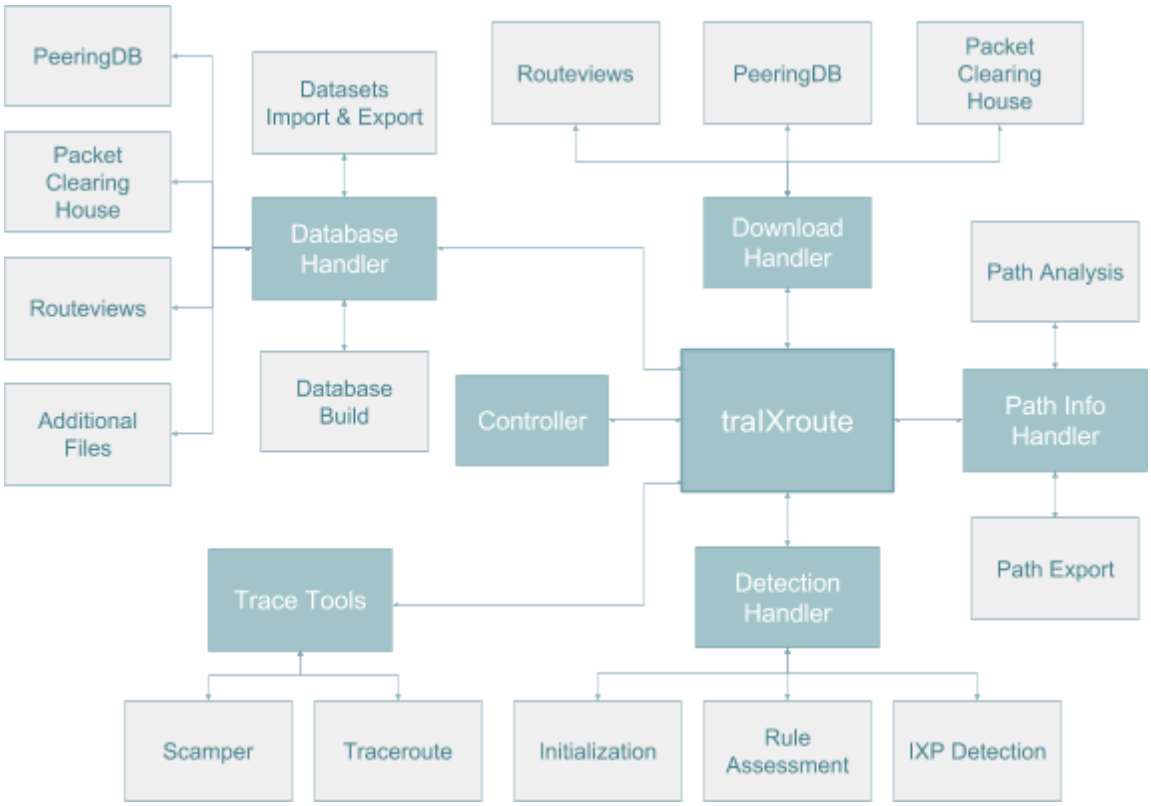

Figure 1: Overview of **traIXroute** modules.

A brief description of each module is given below:

traIXroute: This is the core module that orchestrates all the tool modules to detect and identify if and at which hops an IXP has been crossed.

Download Handler: It is responsible for downloading the PDB and PCH datasets and the Routeviews AS-to-Prefix mappings.

Database Handler: This module is responsible for managing the **traIXroute**'s database. It utilizes the downloaded datasets, imports the files with the user-defined data and after merging, it finally builds the database.

Controller: It takes over printing all the informational and error based messages throughout the IXP detection process. Also, it enables the command line parser to resolve the arguments set by the user to traIXroute instructions.

Trace Tools: It configures and runs the selected traceroute tool (i.e., scamper or traceroute) to send probes.

Path Info Handler: It analyses and enriches the IP path, received from a traceroute probe, with AS and IXP information before applying the detection rules in *Detection Handler* module.

Detection Handler: It loads the IXP detection rules, applies the rules on the enriched traceroute path, pinpoints the IXP crossing links and finally prints the detected IXP hops.

AUTHORS

traIXroute was written by Michalis Bamiedakis (***mbam [at] ics [dot] forth [dot] gr***), Dimitris Mavrommatis (***mavromat [at] ics [dot] forth [dot] gr***) and George Nomikos (***gnomikos [at] ics [dot] forth [dot] gr***) from the **IN**ternet **S**ecurity, **P**rivacy, and **I**ntelligence **RE**search (INSPIRE) Group [14] in the Institute of Computer Science of the Foundation for Research and Technology - Hellas (FORTH). The research was supervised by Prof. Xenofontas Dimitropoulos (fontas [at] ics [dot] forth [dot] gr).

ACKNOWLEDGEMENTS

The research that led to **traIXroute** was supported by the European Research Council (ERC) Grant 338402 - The NetVolution Project (www.netvolution.eu).

REFERENCES

[1] Nomikos, G. and Dimitropoulos, X., "traIXroute: Detecting IXPs in traceroute paths." In *Passive and Active Measurement Conference*, pp. 346-358. Springer International Publishing, 2016.

[2] Luckie, M., "Scamper: a scalable and extensible packet prober for active measurement of the internet." In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pp. 239-245. ACM, 2010.

[3] Augustin, B., Friedman, T. and Teixeira, R., "Multipath tracing with Paris traceroute." In *End-to-End Monitoring Techniques and Services, 2007. Workshop on*, pp. 1-8. IEEE, 2007.

[4] Traceroute. (https://en.wikipedia.org/wiki/Traceroute)

[5] Packet Clearing House - Internet Exchange Directory. (https://prefix.pch.net/applications/ixpdir/menu_download.php)

[6] PeeringDB. (http://www.peeringdb.com)

[7] Routeviews Prefix to AS mappings Dataset (pfx2as) for IPv4. (http://www.caida.org/data/routing/routeviews-prefix2as.xml)

[8] Scamper Documentation. https://www.caida.org/tools/measurement/scamper/man/scamper.1.pdf

[9] Traceroute Documentation. (http://linux.die.net/man/8/traceroute)

[10] Reserved IP addresses. (https://en.wikipedia.org/wiki/Reserved_IP_addresses)

[11] Mao, Z.M., Rexford, J., Wang, J. and Katz, R.H., 2003, "Towards an accurate AS-level traceroute tool." In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 365-378. ACM, 2003.

[12] RIPE Atlas. (https://labs.ripe.net/Members/george_nomikos/detecting-ixps-in-traceroute-paths-using-traixroute)

[13] RIPE Atlas Cousteau. (https://ripe-atlas-cousteau.readthedocs.io/en/latest/use.html)
[14] INSPIRE Group, TNL, ICS-FORTH. (http://www.inspire.edu.gr)