Project Description of Work

Context

Routing is one, if not the most important task undertaken by the Internet. If the goal is to deliver data to a specific destination, the decision process is complex and opaque. It involves multiple stakeholders, each with different objectives. Despite the absence of a central coordinator, the resulting paths are expected to be bug-free.

The goal of this project is to analyze whether Internet routing processes packets differently depending on the data they carry and the impact on the performance as seen by the endpoints.

Objectives

The objective of this project is twofold. It involves programming:

1. a traceroute-like tool and
2. a visualizer of the discovered topology.

Background

Traceroute Command. The *traceroute* command is a command-line tool for tracing Internet paths. It takes as an argument the IP address of a destination and creates multiple sequences of IP packets addressed to this destination. Once sent, traceroute will track these probe packets by tracing the path they take before reaching their destination.

Traceroute Implementation. The implementation of the traceroute command (tracert on Windows) varies from one operating system to another:

- On Unix, Linux, and MacOS, the exploration packets carry a UDP datagram. Upon receiving such packet, routers return an ICMP *Time Exceeded* message, while the final destination returns an ICMP *Port Unreachable* message.
- On Windows, the exploration packets carry an ICMP *Echo Request* message. Routers return an ICMP *Time Exceeded* message, while the final destination returns an ICMP *Echo Reply* message.

It is by analyzing the returned ICMP messages that traceroute discovers the IP addresses of the routers located along the path followed by the exploration packets. Once the IP addresses known, the traceroute command can also resolve their names by sending a DNS request.

Traceroute Output. Figure 1 shows the output resulting from the execution of the traceroute command with the IP address 1.1.1.11.


Traceroute output

Traceroute shows the IP addresses and the names (if known) of the nodes discovered. The traceroute command also measures the round-trip time between sending a probe and receiving the response from one of the nodes located along the path taken by the probe.

Traceroute Options. The *traceroute* command comes with several options that allow for modification of the probe packets header.

These options allow, for example, modification of:

- The TTL header field (options -m and -M),
- The Protocol header field (option -P to specify UDP, TCP, or ICMP),
- The Destination port number for exploration packets encapsulating UDP or TCP (option -p).
- …

Other options will modify the command output. For instance, option -n will prevent traceroute from resolving the IP addresses of the nodes

Output result. The *traceroute* command returns the list of nodes that responded to the probe packets. By default, the traceroute command sends a series of 3 packets to each node being explored, including the final destination. Between each series, the TTL of the probe packets is incremented by 1.

For each series, the following is returned:

- The IP address of the node that sent the responses,
- The name of this node (if requested and if DNS resolution is successful).

For each received response within a series, the round-trip delay (RTT) between sending the probe packet and its response is measured. If no answer is received from one node, the traceroute command will display an asterisk (*). This is the case for nodes probed at steps 4 and 7 in Figure 1.

Description of work

You are asked to design **1)** a standalone implementation of the traceroute command including its standard options **and 2)** a visualizer of the discovered paths.

1/ Input

Your program will take as input a list containing dozens or even hundreds of IP addresses (listed in a txt or csv file) and a set of options similar to those offered by the standard *traceroute* command. Your program will generate probe packets according to the options provided as arguments.

At each step of path exploration, your program will send several series of probe packets. Each series should contain three probe packets: the first encapsulating the UDP protocol, the second encapsulating the TCP protocol, and the third encapsulating the ICMP protocol.

The user of your program should be able to set the number of series for each exploration step by passing a specific option as an argument to your program.

Between each exploration step, you will increment the TTL of the probe packets. The initial and maximum TTL values must be adjustable through arguments passed to your program.

The port number value for the probe packets encapsulating a UDP datagram or a TCP segment should also be passed as an argument if needed.

The user of your program should also be able to specify the packet size or the waiting time between sending two consecutive exploration packets, whether belonging to the same series or not.

Without any options, your program will use the standard default values used by the traceroute command.

2/ Output

Your program should return:

1. A text file with the raw results for each IP address provided as input to your program.
2. A interactive user interface representing the source path tree including discovered nodes for each IP address provided as input, along with the links connecting these nodes. Your representation should include the information collected during the topology discovery (e.g., the RTT measurements and the loss rate). For clarity, the topology information may be represented as follow:
   - Nodes may be labeled with their IP address(es) and their name if known.
   - Links may be highlighted with a specific color depending on the protocol used in the probe packets. The length of links may vary depending on the average RTT measured and their thickness on the measured throughput.

References

1. B. Donnet and T. Friedman, "Internet topology discovery: a survey," in *IEEE Communications Surveys & Tutorials*, vol. 9, no. 4, pp. 56-69, Fourth Quarter 2007, doi: 10.1109/COMST.2007.4444750.

2. M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute probe method and forward IP path inference." in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement* (IMC '08). Association for Computing Machinery, New York, NY, USA, 311–324.
3. B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute." in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (IMC '06). Association for Computing Machinery, New York, NY, USA, 153–158.

Grading

- You will conduct this project in **teams of 3 or 4**.
- You are free to choose **the programming language of your choice**.
- Submission date: **Friday, May 09, 06:29:59 pm**.
- Submission link: Project.
- Submissions files:

    1. **One zip file** to submit on Brightspace containing:
        a. your **source code**,
        b. a **binary file** or a **makefile** to run the execution of your analyzer,
        c. a **readme** file describing the structure of your source code,
        d. a **howto** file explaining how to install and run your program.

    2. **A prerecorded video presentation** of 10 minutes, uploaded to Youtube : First upload the video to your YT channel and then Add the video to the playlist. Note: Set your Audience to "No, it's not made for kids". In your video, you will present:

        1. a brief overview of your project,
        2. a description of your design choices, implementations, and personal contributions,
        3. a demo of your topology explorer in action.