<TraceMyRoute-A-BehindTheScenes-Look>

General Overview

Everyday we send and receive email, we search for a news article or new recipe to try, we interact with advertisements and social media posts; just to name a few points of interaction with the Internet. Every time we click to search or send, the results seem instantaneous but in actuality that TikTok video or news headline or email has made quite the journey to get to you. So, what journey was that?

TraceMyRoute is an interactive search engine to follow your data's journey, from start to end. Every step along the way is called a "hop" and has the ability to send a "ping" back. This interactive traces those hops and pings, providing information about the location of the hop including the GeoLocation and IP Address.

Why? There are a couple of reasons this is a helpful tool. To the everyday user this is a fun, albeit nerdy, way to get a peak behind the curtain of how the Internet works. It would be simple to use for someone unfamiliar with the technical side of networking or computer jargon or easily adjusted for those who have a greater understanding around IP Addresses. This is also a helpful tool for networking/IT students and those new to network management to begin seeing and understanding traffic of packet sending on the Internet. Network design is built around understanding traffic patterns, where packets are going, how they are getting there, etc. This could be a building block to that understanding.

Additionally, Traceroute is a command built into almost every computer, used across operating systems. [1] Building this interactive is building on something that is built into operating systems making it accessible while adding a helpful visual representation of this tool. Research also shows that traceroutes are performed almost daily by network administrators. [1] There is potential for this to be a tool not only helpful on its own but also a tool to be built on for future needs.

Here are the building blocks to make this interactive work:

- The interactive would be built into a Flask web app for easy use and accessibility
- The interactive would be initiated by the user
- The interactive is built on some key methods including: Traceroute, Ping, IP2GeoTools, and Folium Map
- The default of the interactive would be starting the traceroute with the user's built in IP address
    - An alternative user interaction would be the user manually inputting the source IP address
- The route would be plotted and visually traced on a world map (via Folium HTML https://realpython.com/python-folium-web-maps-from-data/)
- GUI: Ideally the interactive would be a web app using Flask but the starting point would be building in jupyter notebook ensuring the search properties, Folium, etc works well.

[1] Pramatarov, Martin. "Traceroute command and its options". 2022. https://www.cloudns.net/blog/traceroute-command-tracert/

    - Then Folium would be used with Flask (https://python-visualization.github.io/folium/latest/getting_started.html)
    - An alternative to using Flask would be building this interactive around TkInter, making it accessible on a desktop.
- Possible Enhancements:
    - Each plot would visualize properties about the individual hop including geolocation, IP address, and time it took for the package to get from the last hop to the current hop.
    - The API would give the user the opportunity to input a manual source address rather than the default user IP address.
    - Plots would be distinguishable by characteristics (i.e. within country hops, cross-country hops, blocked countries, etc.).
    - The drawn line between hops could be enhanced to show latency, indicating slower travel and longer travel between hops.
    - Consider if a user would have need for a csv of the Ping Location data (like if the user were performing numerous pings and wanted to compare cross locations), and if so, providing a csv for the user.

Task Vignettes

Task 1: Begin Traceroute

The user is looking to follow the data stream to a predetermined destination (i.e. www.airbnb.com, https://www.sydneyoperahouse.com/, mail.google.com, etc.). The user is only interested in the route from their personal device and is not looking to follow a search from another source, therefore the source is the interactive's default. The user inputs a predetermined destination address, like a website.

- Source location:
    - Default: is set to user's location/IP address
    - Optional input: manually setting source IP address
- User configurable fields:
    - Source Location: default to current
    - Destination Address: user can input website or IP address
- User activity:
    - User starts application
    - User inputs destination address (and optionally, source address)
    - User hits Start Trace button, which begins the traceroute

Task 2: Traceroute Interactive Map

Once the user conducts a traceroute, the interactive will respond with a digital map displaying the traceroute as plotted points on each hop's geographical location with a line connecting each hop from source to destination. The user can interact with the map, zooming in and out to focus on specific areas of the route, clicking on specific hop points to know about that specific location, discovering the city, country, IP address, and time it took to arrive at that hop

from the last hop. The user will notice some coloring differences between plot points indicating characteristics of that specific hop (i.e. closed countries that don't release certain IP or location information). *See Diagram below for a visual representation of the interactive map.*

- Traceroute Plot Points:
    - Each hop along the traceroute is identified and listed, plotted based on geolocation and highlighting the IP address and time it took for data packet to arrive at each location.
    - Some locations, due to firewalls or other security elements, are blocked, these locations are distinguished from traditional pings.
- Output Display:
    - List of Ping Locations
    - Starting Location
    - Starting IP Address (if not given by user, converted by program)
    - Destination
    - Destination IP Address
    - Geo Destination
    - Total Pings
    - Total Unknown Locations
    - Total Time
    - Option to start a new traceroute via "Start Trace"
- User Activity:
    - User views interactive map
    - User views the overview as they review Traceroute Details, seeing source and destination locations and IP addresses, total pings, total unknown locations, and total time.
    - User can scroll through Ping Locations to view all ping information
    - User views the map, zooms in to view specific locations more clearly.
    - User clicks on a specific location to view that hops' geolocation, IP address and specific time to this location.
    - User determines if a new traceroute should be performed

Task 3: User begins Traceroute with a specific source address in mind

For certain network management reasons, a user may need to follow a trace beginning not at their personal source but from another source (i.e. a coworker's computer), so the user opens this interactive application to follow a data stream from a designated source to a predetermined destination. This task is similar to Task 1 except instead of a default starting location, the user manually inputs a starting IP address to initiate a traceroute. The user inputs the destination (either a website address or an IP Address) and clicks "Start Trace."

- Source location:
    - Default is overridden by user when the user manually enters the source IP address
- User configurable fields:
    - Source IP address

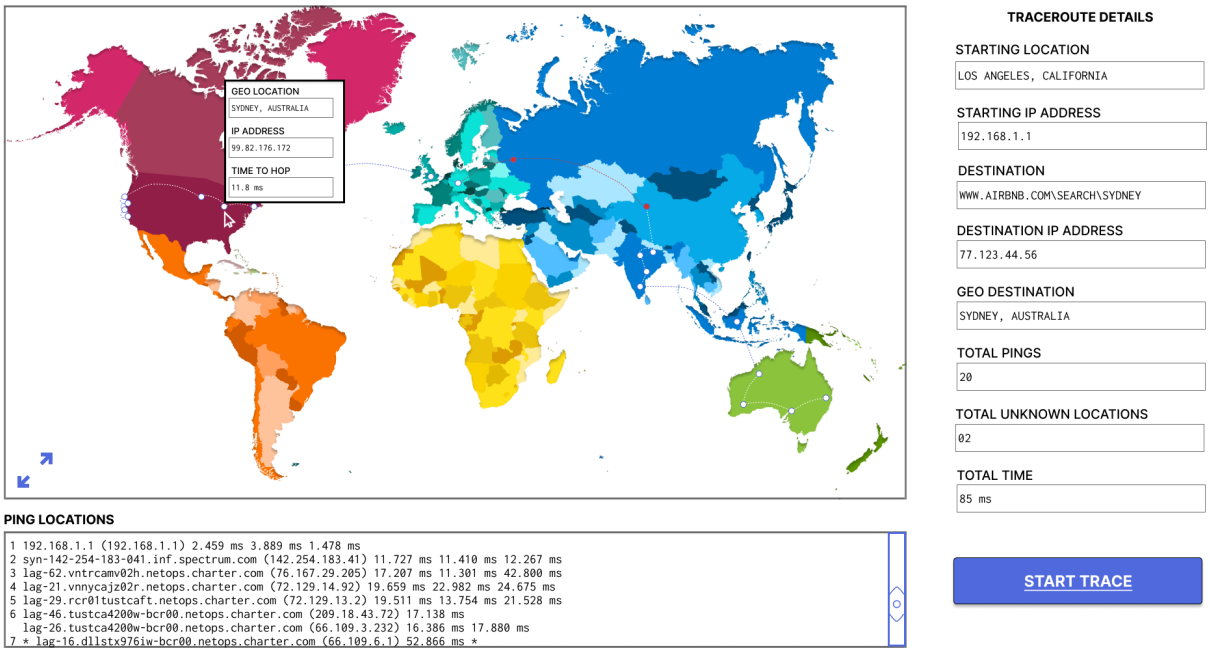
    - Destination Address
- User activity:
    - User starts application to begin a traceroute
    - User inputs source IP address
    - User inputs destination address
    - User hits “Start Trace” button, which begins the traceroute

The results/follow-up task for Task 3 would be the same as Task 2.

Diagram

Below is the sample diagram of what the user could see upon opening the application. The intent is to indicate which fields are required and the “Start Trace” button would change from gray to blue when a destination was input.

Below is a sample diagram of the results the user interface would provide following a user's traceroute. (Note: the coloring is optional and used to distinguish various points on the interactive.)



Technical Flow

From the user's perspective, upon launching the application, the only fields the user can manually input information into are the required fields: Destination (websites only) OR Destination IP Address, and the optional input of a Starting IP Address and Max Hops. To start the traceroute, the user will click the button "Start Trace."

Core functions:

- Def trace_input():
  - Create an initial user interface to receive user input
  - The user would have the option to input either a destination website OR IP address
    - One of these would be required to initiate "Start Trace"
  - Optional user inputs would be starting IP Address and setting the max hops
    - If not inputted by user, both will be set to the default of user's IP address and Max Hops being 50
- Def source_address():
  - If user does not manually input a source IP address the default IP address is used
    - Return an error string if user inputs faulty or incorrect input/address
  - The default IP address is determined based on user device

- Def dest_address():
    - Required input is either a destination OR a destination IP address
    - IF destination:
        - Determine destination meets website requirements
        - If false: return error string indicating incorrect destination
        - If true:
            - Import socket library to convert destination to an IP address using DNS functions
            - Set destination address
- Def trace_route()
    - Using source_address, dest_address and max hops set up traceroute function
    - To perform a traceroute, a ping has to be set up sending packets along the route from source to destination resulting in a list of hops along the route.
    - This would include a while loop that captures IP data, time data, and hop data using python libraries: socket, struct, time, scapy, and argparse
    - Returns a list, trace_list, of all pings from source to destination
        - List would include IP address, time to hop and packet arrivals
        - The list would also include destinations without replies; this is indicated by an *; this occurs for low priority data packets, firewalls, etc.
    - Error messages:
        - In a traceroute, there is a function built in maxing out on the number of hops a trace performs. This can be set by the user or set as a default. Options for this interactive would be to set a default, allow the user to set a default or set it to unlimited (unlimited could prompt issues as there could be some traceroutes that continue for a long time with no resolve causing issues). An error message would have to be set up to let the user know that max_hops was reached and therefore the traceroute timed out.
- Def hop_geolocation()
    - This class will either be independent or built with trace_route()
    - Loops through trace_list using ip2geotools python library to take the ping's ip address and determine the country, region, city, latitude and longitude.
    - Return geo_hoplist, populated with hop ip address and geolocation
    - These locations are then used to populate the folium map
- Def draw_map()
    - Using Folium (along with GeoJson and pandas) build an interactive map to then plot geographical locations of the traceroute hops
    - Draw points on map indicating each hop from source to destination, with a line attaching each hop
    - Include hop data at each point to allow for user to click on the point and discover IP address and geolocation
    - Optional Enhancements:
        - Designate open and close countries by color (red dot meaning a closed country)
        - Designate intercontinental hops by changing the line color

        - Indicate slower or faster latency speeds by a dashed line or smooth line connection

Program flow:

- Either a Flask app main page or TkInter window creates the GUI to get source location, destination address and render folium map for traceroute
- Data is parsed and delivered to the server
- If a user initiates a new trace, the folium map is cleared and re-drawn with the new traceroute

Self Assessment

General. Writing through the project specs, I am excited to see that each section is doable, but it is also important to keep each section distinct in order to build this correctly and smoothly. As I wrote through the project specs, I realized that there were a number of errors that could occur needing feedback responses given to the user. The spec was also a good reminder that it will be important to define each aspect clearly and hardcode only what's necessary to allow the program to be adaptive.

Further Research Required. There are also a couple of questions my initial research didn't provide clear answers to. For example, manually inputting a source address from a technical standpoint appears doable but might require some code finessing once I get into actually building it. Additionally, whenever working with the Internet and IP addresses you run into protocols and cybersecurity elements; I recognize this could impact some of the results even if the code is written correctly. It would be my goal to build this in a way that mitigates as many hurdles as possible.

Areas Lacking Confidence. I did put a Flask app as the primary mode with an optional TkInter window. A Flask application might be overall better for this design but I am much more confident using TkInter than Flask, which is why I hesitate to build the application around Flask. But I am also aware that I am building this on a Mac computer which in itself isn't bad, but I have historically had trouble viewing some of the TkInter features or having trouble with it causing Python not to respond. It will be good to build within a jupyter notebook first and then build out the GUI aspects.

There are a number of newer libraries that will make this code easier to build but that I haven't personally worked with it. It will be fun to explore these new capabilities but also a learning curve. These include ip2geolocation, GeoJson, and socket.

Biggest Problem. The biggest problem I could see happening is not being able to supply a visual for blocked hops. Traceroutes can often result in a hops that do not give feedback on location (i.e. firewalls, geopolitics, etc.); I will need to determine how to either a) set a default intermediary location or b) set up the possibility for "blanks" in the route. An initial thought for setting up default intermediaries could be determining how many "blanks" occur and divide the area between 2 "live" hops by the number of "blanks" and place filler plots in that space with an indication of unknown hops.

Final Assessment. I am excited to build this but I am also trying not to be overwhelmed. I know the building blocks of this are on the simpler side but it is putting them together, working seamlessly, and anticipating errors that sometimes overwhelm me.