Note: This is one of the variance of all of the main files that we as a team have developed. One could find different programs with similar functionality with some modifications in different team members' folders.

traceroute_at_location.py

```
from scapy.all import traceroute, IP, UDP, sr1
from random import randint
import os

'''
This python script:
Inputs:
Location

creates -> traceroute_at_location.txt and writes in it
raceroute_at_location.txt contains traceroute to the given ip address
'''


def traceroute_at_location(ipaddr, location, results):
    f = open(f"{directory}/traceroute_at_location.txt", "a")
    f.write(f"Traceroute to {ipaddr} at {location}:\n")
    traceroute_result, _ = traceroute(target=ipaddr, maxttl=100)
    for snd, rcv in traceroute_result:
        ipaddr = rcv.src
        f.write(f"{ipaddr}\n")
        results.add(f"{ipaddr}")
    f.write(f"{results}\n")
    f.write("----------------\n")
    f.close()
    return results

if __name__ == "__main__":
    # change ip address here - up to which you want to traceroute the packet in the network
    ipaddr = "8.8.8.8"
    location = input("Enter your Location: ")
    results = set()
    # change directory here - where you want to save the files
    directory = f"Location/1"
    if not os.path.exists(directory):
        os.makedirs(directory)
    traceroute_at_location_results = traceroute_at_location(ipaddr, location, results)
```

tracroute_prefix.py

```
from scapy.all import traceroute, IP, UDP, sr1
from random import randint
import os

'''
This python script:
creates -> trace_results.txt and active_ip_sets.txt and write in those files
trace_results.txt contains traceroute to the given ip address: from x.x.0.0 to x.x.255.255
active_ip_sets.txt contains the sets in each line where each set is a traceroute for the given ip address
'''

def trace_helper(ipaddr, directory, active_ip_sets):
    f = open(f"{directory}/trace_results.txt", "a")
    f.write(f"Trace Route: {ipaddr} \n")
    print(f"Trace Route: {ipaddr}")
    for ttl in range(1, 100):
        pkt = IP(dst=ipaddr, ttl=ttl) / UDP(dport=33434)
        reply = sr1(pkt, verbose=0, timeout=3)
        if reply is None:
            f.write(f"Timeout at {ttl} hops away for {ipaddr}\n")
            print(f"Timeout at {ttl} hops away for {ipaddr}")
            break
        elif reply.type == 3:
            f.write(f"Destiation unreachable at {ttl} hops away for {ipaddr}\n")
            print(f"Destiation unreachable at {ttl} hops away for {ipaddr}\n")
            break
        else:
            f.write(f"{ttl} hops away: {reply.src}\n")
            print(f"{ttl} hops away: {reply.src}")
            active_ip_sets.add(str(reply.src))
    f.write("-----------------------------------------------\n")
    print("-----------------------------------------------")
    f.close()
    return active_ip_sets

def trace(prefix, directory, active_ips_set, ts, te, fs, fe):
    f = open(f"{directory}/active_ip_sets.txt", "a")
    for i in range(ts, te):
```

```
        if i > 255:
            break
        for j in range(fs, fe):
            if j > 255:
                break
            ipaddr = f"{prefix}.{i}.{j}"
            trace_helper_result = trace_helper(ipaddr, directory, active_ips_set)
            f.write(f"routes: {trace_helper_result}\n")
            #print(f"routes: {trace_helper_result}\n")
    f.close()
    return trace_helper_result

if __name__ == "__main__":
    # change the values here - from where to where you want to run - 3rd octet and 4th octet
    ts = 251
    te = 256
    fs = 0
    fe = 256
    # change the prefix here - 1st octet and 2nd octet which you want to fix
    prefix = "138.238"
    active_ips_set = set()
    # change directory here - where you want to save the files
    directory = f"Data/{prefix}"
    if not os.path.exists(directory):
        os.makedirs(directory)
    trace_results = trace(prefix, directory, active_ips_set, ts, te, fs, fe)
```

all_active_ips.py

```
import networkx as nx
import matplotlib.pyplot as plt
from scapy.all import traceroute, IP, UDP, sr1
from random import randint
import os

'''
This Python script:
creates -> routes.txt by reading all.txt where all.txt is the file containing all the data.

'''

def active_ips_from_file(fname, outname):
```

```
f = open(f"{outname}", "a")

with open(fname, 'r') as file:

    lines = file.readlines()
    end = False
    route = []

    for line in lines:

        if "hops away" in line and ":" in line:

            ip = line.split(":")[1].strip()
            parts = ip.split(".")

            if parts[0] == "10":
                end = True
                print("Entered")
                route.append(ip)
                f.write(f"{ip}\n")

            if parts[0] == "138" and int(parts[1]) <= 238:
                end = True
                route.append(ip)
                f.write(f"{ip}\n")
                print("Entered")

        elif "STOP" in line:
            print(f"Stopped at line {line}")
            break

        elif end == True:
            f.write(f"Line End:\n")


        else:
            end = False
            continue

            #f.write(f"Route Started:\n")
            #route = []
f.close()
```

```
if __name__ == "__main__":


    active_ips_from_file("all.txt", "active_ips_from_file.txt")

    routes = []
    route = []

    with open("active_ips_from_file.txt", 'r') as file:

        lines = file.readlines()


        for line in lines:
            #print("Line: ", line)

            if "Line End:" in line:
                routes.append(route)
                #print("routes: ")
                #print(routes)
                route = []

            else:
                #print("Entered Append")
                route.append(line)


    print("routes", routes)

    f = open(f"routes.txt", "a")
    for i in routes:
        f.write(f"{i}\n")
    f.close()
```

topology.py

```
import networkx as nx
import matplotlib.pyplot as plt
import ast
import itertools
```

```
'''
This python Script:
reads routes.txt and creates a topology
'''

def get_colors(G):
    node_colors = {}
    color_iter = [(0/255.0, 0/255.0, 255/255.0), (255/255.0, 0/255.0, 0/255.0), (0/255.0, 255/255.0,
0/255.0), (255/255.0, 0/255.0, 255/255.0), (0/255.0, 255/255.0, 255/255.0), (127/255.0, 0/255.0,
255/255.0), (127/255.0, 0/255.0, 255/255.0), (255/255.0, 255/255.0, 0/255.0), (0/255.0,
102/255.0, 102/255.0)]
    l = len(color_iter)
    #print("Total colors = ",l)
    c = 0
    for node in G.nodes():
        node_colors[node] = color_iter[c]
        c += 1
        if c >= l:
            c = 0
    #print("node_colors")
    #print(node_colors)
    return node_colors

def topology(fname):
    G = nx.DiGraph()
    try:
        with open(fname, 'r') as file:
            lines = file.readlines()

            for line in lines:
                try:
                    result_list = ast.literal_eval(line)
                    result_list = [item.rstrip('\n') for item in result_list]

                    if len(result_list) == 0:
                        continue

                    if len(result_list) == 1:
                        G.add_node(result_list[0])

                    for i in range(len(result_list) - 1):
                        source = result_list[i]
                        dest = result_list[i + 1]
                        if source == dest:
```

```
                    continue
                G.add_edge(source, dest)
            except ValueError:
                print(f"Could not convert line {line} to list.")
    except FileNotFoundError:
        print("File not found.")

    return G

if __name__ == "__main__":
    try:
        G = topology("routes.txt")
        if G is None:
            print("Failed to create the graph.")
            exit(1)

        dict_colors = get_colors(G)

        edge_colors = list(dict_colors.values())

        plt.figure(figsize=(50, 50))
        pos = nx.spring_layout(G, seed=0, k=3, scale=5)

        nx.draw(G, pos, with_labels=False, node_size=1500, node_color=[(245/255.0, 250/255.0,
255/255.0)], edge_color=edge_colors, width=3, arrows=True, arrowsize=25)

        for node, (x, y) in pos.items():
            plt.text(x, y, node, fontsize=30, ha='center', va='center', color='black')
            #plt.text(x, y, node, fontsize=30, ha='center', va='center', color=font_colors.get(node,
'black'))

        plt.title("Network Topology from Traceroute Data")
        image_path = f"topology_final.png"
        plt.savefig(image_path, dpi=300)
        plt.close()

        with open(f"topology_final.md", "w") as md_file:
            print("Writing to the Markdown file...")
            md_file.write("# Building Network Topology\n")
            md_file.write(f"![Network Topology]({image_path})")
            print("Successfully written to topology_final.md")
    except Exception as e:
        print(f"An error occurred: {e}")
```