Our approach to construct the network topology consisted of developing two scripts - topology-tracing.py and topology-graphing.py, which together are capable of conducting traceroutes and graphing the network topology - and running the two scripts at various locations across campus.

topology-tracing.py

This Python script uses the scapy library to send ICMP packets to trace the route to a given host. It spawns multiple threads to perform traceroute to multiple IP addresses simultaneously, increasing efficiency.

The *trace_route(host, results)* function is responsible for executing the traceroute to a given host and updating the results with each hop's IP address. It uses the scapy library to send ICMP packets with varying Time-To-Live (TTL) values to trace the route.

The *trace_route_with_threads(ip_range, results)* function initiates traceroute threads for a range of IP addresses and manages the multithreading aspect.

The *generate_ips(start_ip, end_ip, skip=1)* generates a list of IP addresses within a given range, allowing for specifying a step (skip) between addresses.

The main part of the script initializes a results dictionary, defines the IP address range to trace (specified via command-line arguments), and invokes trace_route_with_threads to perform traceroutes. The results are then stored in a JSON file for further processing.

topology-graphing.py

This Python script is responsible for visualizing the network topology using the results obtained from the traceroute.

The *load_traceroute_results(filename)* function loads the traceroute results from a JSON file, returning them as a dictionary.

The *build_network_graph(results)* function constructs a network graph using the traceroute results. It creates nodes for each unique IP address and edges between consecutive hops, representing the network's connections.

The main part of the script loads the traceroute results, builds the network graph, and visualizes it using the networkx and matplotlib libraries.

In summary, topology-tracing.py traces routes to multiple IP addresses, records the results, and saves them to a JSON file, while topology-graphing.py reads the JSON file, builds a network graph, and visualizes the network topology. This provided us an efficient way to analyze and graph the campus network connectivity.

After developing the scripts, all that was left to do was to run the scripts at various buildings across campus, including but not limited to Lewis K Downing Building, Burr Gymnasium, Health Science Library and more.