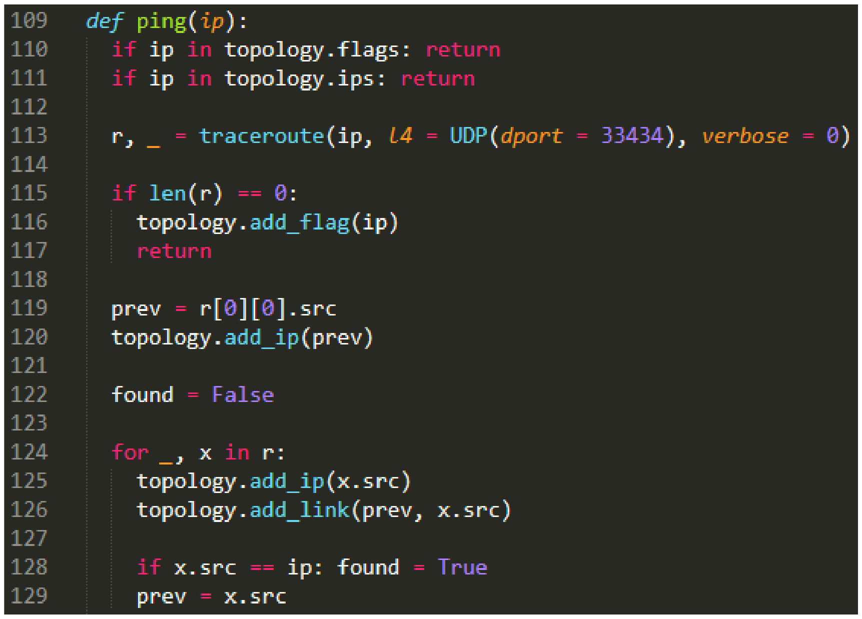
# Methodology

Howard Network Topology Project

## 1. IP Scanning & Route Tracing Approach

We used Scapy to do it. The core code snippet is as the following:



```python
def ping(ip):
  if ip in topology.flags: return
  if ip in topology.ips: return

  r, _ = traceroute(ip, l4 = UDP(dport = 33434), verbose = 0)

  if len(r) == 0:
    topology.add_flag(ip)
    return

  prev = r[0][0].src
  topology.add_ip(prev)

  found = False

  for _, x in r:
    topology.add_ip(x.src)
    topology.add_link(prev, x.src)

    if x.src == ip: found = True
    prev = x.src
```

It does not block for invalid IP as well.

## 2. Multi-Threading and Resuming Function for Effective Scan

We utilized multi-threading to ensure a large-scale scanning. Implementation of resuming from the interrupted point was essential. We added this function in our script.

## 3. Scanning Strategy

Even with multithreading, investigation of the entire range is surely a time-consuming process.

Therefore, we had to find an efficient scanning strategy.

Let's say the IP v4 address is w.x.y.z. For Howard, w=10 or w=138, x=238.

At each location of the campus, we first examined only IPs with z=1.

Since only x and y are variable, about 65536 IPs must be investigated.

It took one team member about 11 hours (MacBook) to finish this.

Next, we assumed that the IP ranges where z=1 was found are promising candidates and conducted a re-examination of them.
In other words, if 10.33.30.1 was discovered, the 10.33.30.2-10.33.30.254 area was reinvestigated.
In this way, one team member worked in one location for about 20 hours.

## 4. Visualization

Total 85,530 IPs and 78,870 links were discovered in our work. We didn't include IPs not in 10.x.y.z or 138.238.y.z range.
To visualize the topology efficiently and beautifully, we employed some approaches for extraction and pruning.
First of all, we calculated the distribution of node linkages.

| # of Linkage | # of Nodes |
|---|---|
| 0 | 7,552 |
| 1 | 77,557 |
| 2 | 388 |
| 3 | 8 |
| 4 | 1 |
| 5 | 1 |
| 7 | 1 |
| 9 | 1 |
| 21 | 2 |
| 22 | 2 |
| 23+ | 17 |

Distribution of node linkages

(It shows that 388 IPs are connected to two other IPs. This table is not the correct report, since It is only based on our discoveries.)

As see in the above table, 7,552 nodes are isolated, and 77,557(about 90.7%) nodes are just terminals which is connected to only one other node. If we include those nodes in our visualization, it could look poor. That's why we pruned those IPs.

We used NetworkX and Matplotlib for drawing graphs. We generated four visualizations.

(1) Whole Topology

The largest picture with all the nodes except for those with less than two connections.

(2) Public Topology

Only public IPs and their connections.

(3) Bridge Topology

Insight of connections from internal IPs to external IPs. This diagram only involves internal-public or public-internal connections.

(4) Gate Topology

Only IPs with more than three connections. We thought that they could be considered as gateways or local hubs.