CS249i: The Modern Internet **Winter 2023-2024**

Project 1: Routing and Internet Topology

Parts 1–2 Due: January 26 11:59pm PT **Part 3 Due:** February 2 11:59pm PT

Introduction

In this lab, you will BGP peer with an upstream Internet transit provider to receive the global Internet routing table. You'll then analyze the routing table to understand aspects of the Internet's topology.

We will provide you with a Linux virtual machine (VM), an ASN, an IP block to advertise, and privileges to BGP peer with an upstream transit provider. You'll need to configure a software router on your VM to BGP peer with the upstream router, accept their advertised routes, advertise your own IP range, and ultimately dump your router's routing table for analysis.

Internet Measurement Caveats

You'll find that questions about the Internet do not have a single exact or correct answer. Unlike most past courses you've taken, CS249i studies the real-world Internet, which is nuanced and constantly changing. Measurements taken at different times, from different locations, or using different pieces of software can have *dramatically* different results. Further, many results can be interpreted differently. You and your teammates may not agree. This lack of "ground truth" is what makes studying the Internet so interesting. Even among experts, there isn't a globally accepted list of Tier-1 ISPs; the ground truth is locked away behind legal contracts we will never see. Don't let this unnerve you. Instead focus on trying to understand and reason about what you have found to the best of your ability.

You'll also find that some of the answers to the questions we've asked can be found online. For example, there's a Wikipedia article that lists the globally accepted list of Tier-1 Networks. Based on what you can see publicly, you likely would not be able to come up with this list. For example, there's little publicly visible that clue you into the fact that Liberty Global is a Tier-1 ISP. Indeed, I don't think most people you ask would remember it exists. On the other hand, Hurricane Electric's peerings might hint that it is. At the end of the day, do not just copy an online article as your answer. Instead, look at the data you've collected, and come up with a compelling argument about why the answer you've come up with is correct. Critically, use the data you have collected to back up your argument.

Part 1: Peer onto the Internet

In the first part of the project, you'll configure a provided software router to BGP peer with an upstream transit provider and advertise your assigned IPv4 prefix.

Network Topology

As can be seen in Figure **??**, you'll be peering with a router in ASN 65400, which is an ISP dedicated to providing Internet transit to CS249i student groups. AS 65400 (CS249i) in turn receives Internet transit from an upstream provider (ESRG, AS 65105), which receives Internet transit from Stanford (AS 32). Stanford purchases transit from several Internet providers (e.g., Hurricane Electric) and also connects to other network consortiums. While AS 65400 and AS 65105 are special purpose, they're configured similarly to how a regional or local ISP is configured—except that these ASes use private ASNs rather than publicly routable ASNs. The IP address of the router you'll be peering with is `171.67.69.32`.

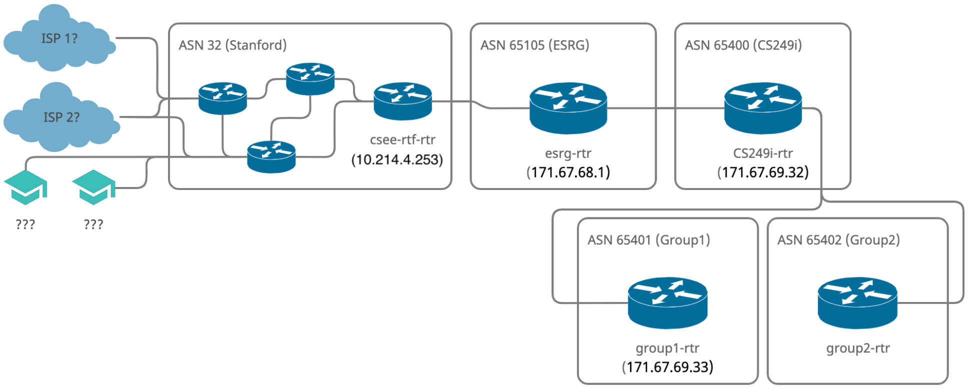

Figure 1: Network Topology

Group Specific Details

Your group should receive an email with several group-specific details:

- The public IP address of your group's router (you'll use this to connect to the upstream router)
- An assigned iBGP autonomous system number (ASN) to advertise
- An assigned /31 block of IP addresses that you will advertise on your ASN
- A private IP address for SSH'ing into your VM. (You won't be able to SSH in using your router's public IP address because it won't have public Internet access until you finish part one of the project!)

Make note of all of these facts; together, they will serve as your identity on the Internet.

Connecting to your VM

You will need to SSH through a "bastion" host to access your VM since it doesn't have a public IP address (yet). We'll provide you an exact SSH configuration to use in the email we send you with VM details. Please do not modify VM settings.

Initial Network Configuration

Your VM will be configured with two IP addresses: (1) a public IP address on the 171.67.69.0/24 subnet that you will use to route traffic, and (2) a private IP address in the 10.216.68.0/24 subnet that you will use only for SSH'ing into your VM, not for routing traffic. This configuration is similar to what you'd see in the real world. Routers have a "management" interface that's managed separately from the data plane on the router. That management interface will typically be a typical ethernet port that's connected to an isolated management network whereas the router ports will be high speed ports for transferring data at high rates.

You can view your IP configuration by running `ip a`. You can also view the routes that exist on your VM by running `ip route`. You'll notice that your VM only knows how to route traffic on two local subnets. Unlike other servers you interact with on a regular basis, there is no default route for `0.0.0.0/0`. Your routes to IP ranges beyond the upstream router will be populated later when you peer with the upstream router. You should see output that looks like:

```
cs249i-student@cs249i-group-XX:~$ ip route
10.216.68.0/24 dev ens3 proto kernel scope link src 10.216.68.YYY
171.67.69.0/24 dev ens9 proto kernel scope link src 171.67.69.ZZZ
```

Output from `ip route` contains the following details:

```
<A> dev <B> proto kernel scope link src <C>
A: routed IP prefix
B: interface to send traffic over
C: src IP address when sending traffic through this interface
```

Check your understanding: based on the output from `ip route`, what IP addresses does your machine know how to route traffic to? What IP addresses could your machine send traffic from?

Try running `ping 8.8.8.8`. As your machine is not currently has no routes to the Internet, you will receive a "Network Unreachable" result back. It's your goal over the course of this lab to fix that! Next, ping the CS249i router by running `ping 171.67.69.32`. That connection should work since the CS249i router is in the same L2 domain as your public IP address and does not need any L3 route.

Introduction to GoBGP

In order to peer with the 249i router and connect to the Internet, you'll be using a software routing package, GoBGP. Despite being a software-based router, GoBGP uses a configuration that's very similar to how you would configure a typical hardware router. In your home directory, you should see a folder named `gobgp` and a folder named `output`. Inside, you'll see four relevant files: `gobgpd`, `gobgp`, `config.toml`, and `run-gobgp.sh`.

`gobgpd` is the executable `gobgp` daemon. It is responsible for setting up and managing your BGP sessions to other routers based on the specifications in your config file. You will receive metadata about routes and the routers themselves through these BGP sessions.

`gobgp` is the gobgp command line interface. While `gobgpd` is running in the background, you can use `gobgp` to dynamically view BGP state and update the server's configuration in real time.

`config.toml` is the configuration file for the `gobgp` daemon; you will populate it with everything it needs to successfully peer with the upstream CS249i router. *This is the only configuration file you will need to edit for this lab.*

`run-gobgp.sh` is a bash script which will run the gobgp daemon and establish BGP connection as defined in `config.toml`, automatically shut it down after a few minutes.

Note: the `run-gobgp.sh` script runs `gobgp` interactively rather than as a daemon in order to make troubleshooting easier (logs will appear right on stderr). As such, if you want to run `gobgp` or troubleshoot while the daemon is running, you'll need to open a second SSH session to your VM or use a utility like `tmux` or `screen` that allows for multiple sessions within one terminal.

If you ever want to run the daemon for longer, you can up the length of the `sleep` command in this file, or run the daemon separately on your own. Just make sure you're careful about not overwriting your dumps (see Dumping Routes for more information) and that your VM doesn't run out of disk space from dumping out the routing table too many times.

For now, running gobgp won't do anything, as your config file is empty. Time to fix that!

Peering with the CS249i Router

In order to peer with the upstream router and connect to the Internet, your machine will need to:

- Identify your own AS
- Identify the neighbor router you want to peer with
- Advertise your IP prefix and AS to your neighbor

In this section, you will fill out your configuration file `config.toml` to accomplish those goals. Read about the TOML file format here. Note that if you ever want to comment out lines from a `.toml` file, you can do so by adding `#` at the front.

In BGP, you are identified via your ASN and your router's IP address, so other ASes know where and how to reach you. You can specify these in your config file as follows:

```
[global.config]
    as = <your-ASN>
    router-id = "<your-router-IP>"
```

Next, you will need to configure your connection with the 249i router.

```
[[neighbors]]
  [neighbors.config]
    neighbor-address = "<neighbor-IP>"
    peer-as = <neighbor-ASN>
```

And that's it! With just 4 lines, you're ready to peer with the CS249i router. Run the daemon and this time you should see it establish a successful connection with the router. When you establish this connection, the other router will advertise to you all of the routes that it knows about. However, based on this configuration, you won't be advertising anything to the other router.

In a separate terminal (or with the daemon in a screen), run `./gobgp neighbor` to see a summary of your connection with the 249i router.

Check your understanding: after running `./gobgp neighbor`, do you see the peer you'd expect? If you run it a few times, does the number of routes you've received fluctuate at all? If so, why?

Pushing Routes to the Linux Kernel

Currently, although you are successfully receiving routes from the upstream CS249i router, these routes only live in `gobgp`'s memory, in what's known as the Routing Information Base (RIB). None of the routes are pushed to the router's forwarding plane. In this case, the Linux kernel is the forwarding plane since we're using a software router. You can confirm that none of the routes have been pushed from GoBGP to the kernel by running `ip route`.

To deploy routes from the GoBGP RIB to your Linux kernel, we'll use Zebra, a complimentary piece of routing software that interfaces between GoBGP and Linux. To do so, add the following configuration to your `config.toml` file:

```
[zebra]
  [zebra.config]
    enabled = true
    version = 6
    url = "unix:/var/run/frr/zserv.api"
    redistribute-route-type-list = ["connect"]
```

It's a bit in the weeds for our purposes, but essentially this configuration tells the GoBGP daemon that it should use Zebra and where to connect to a Zebra process. Go ahead and restart the daemon once you've made the change.

Once that's done, try running `ip route | wc -l` to count the number of lines in your routing table. Where before it was 2, now it should be many, many more. This number should eventually stabilize around 800K–1M routes.

Once you have all of the routes, you should be able to verify that you have Internet connectivity by running `ping google.com`. Once that's successful, congratulations, you've officially peered onto the Internet! Unlike a normal computer, which simply sends all packets to the upstream router, your kernel knows about every publicly announced prefix on the public Internet and will send packets to the other router because it is the shortest path to those prefixes. You can send a packet to 8.8.8.8 because you have Google's advertised prefix that contains that IP address in your routing table.

Advertising a Prefix to the World

So far, you've peered with an upstream neighbor, but you haven't advertised any IP prefix to your upstream provider. You're able to see routes because you're running commands on the router itself, but if you had any customers, they still would be without Internet connectivity.

Last but not least, you'll need to advertise your AS's prefix:

```
./gobgp global rib add -a ipv4 <subnet-to-advertise>/31
```

This will let your neighbors know that you own the /31 you were assigned and that any traffic sent to those IPs should be routed to you (i.e., you will advertise the prefix to your neighbors).

Dumping Routes

Now that you are able to successfully fetch all the routes, you can begin to asking questions about them. That's hard to do with all the routes just living in memory, so first we'll want to dump them to a file. To do that, add the following to your configuration file:

```
[[mrt-dump]]
  [mrt-dump.config]
    dump-type = "table"
    file-name = "/home/cs249i-student/output/table.mrt"
    dump-interval = 120
```

This will generate a dump ("MRT") file inside the output folder every 2 minutes with the date and time that the dump was generated as the name. The MRT format is a binary format that is used to export routing protocol messages, state changes, and routing information base contents. If you're curious, more information is available in RFC 6396.

You can check if it's done by running:

```
wc -c table.mrt
```

Once the output number is nonzero, the table has been successfully dumped.

Note that gobgp will append to the dump file once every two minutes. Unfortunately, there's no way to configure GoBGP to overwrite instead. Thus you'll want to be careful that you shut off the daemon after the file has been written to once to avoid duplicating data or corrupting it. If you run with `run-gobgp.sh`*, this will be handled for you. The script also erases the old table in the output folder folder if there was one; if you want to preserve multiple tables, you should rename the old one to something else.*

MRTs are a specific binary file type designed for dumping BGP data. If you're curious, you can read about them, but the details of this filetype aren't important for our purposes. Rather than dig into the details of the file type, we'll instead convert it to JSON lines, which we can later analyze however we want (e.g., by writing a Python script).

In order to convert it to JSON lines, you can use the `mrt2json` utility. In your home folder, there should be a binary called `mrt2json`. To use it, you can use the command:

```
./mrt2json raw --input-file=/home/cs249i-student/output/table.mrt \
--output-file=/home/cs249i-student/output/table.json
```

This will convert the MRT file to JSON Lines where each line contains a JSON object that contains the record for the routed block. If you're curious, you can read the code for the utility here: `https://github.com/zmap/zannotate/blob/master/cmd/mrt2json/main.go`.

Reading the MRT JSON File

Let's take a look at an example line from `table.json`:

```
{
   "sub_type":"rib_ipv4_unicast",
   "sequence_number":0,
   "prefix":{
      "prefix":"103.127.54.0/24"
   },
   "entries":[
      {
         "peer_index":1,
         "orginated_time":1632281047,
         "path_identifier":0,
         "path_attributes":[
            {
               "type":1,
               "value":0
            },
            {
               "type":2,
               "as_paths":[
                  {
                     "segment_type":2,
                     "num":8,
                     "asns":[65400, 65105, 32, 46749, 46749, 6939, 137367, 17995]
                  }
               ]
            },
            {
               "type":3,
               "nexthop":"171.67.69.32"
            },
            {
               "type":4,
               "metric":0
            },
            {
               "type":8,
               "communities":[32, 454801053]
            }
         ],
         "type":""
      }
   ],
   "route_family":65537
}
```

This MRT entry is of type `rib_ipv4_unicast`, which is a data structure that can contain many Routing Information Base (RIB) entries. RIB entries, typically used by a router, tell you the paths (routes) that the router can take to get to the specific `prefix`. You can find the specifics of each path attribute field here, but the most important to understand are the ones listed type 2 and type 3. Type 2 attributes, or `AS_PATHS`, give you a path that starts from your ASN, `64512`, and tells you how to get to ASN `17995`, which is the ASN that advertised a route for the prefix `103.127.54.0/24`. Routers follow this path by using the IP address located in the `NEXT_HOP` attribute, which is typically an external router that can forward packets to its appropriate neighbor to reach your destination prefix.

Note that we have additionally given you a file called `asn_names.txt` in your `output/` folder which maps AS numbers to their corresponding names. You can use this file to help make the MRT you dumped more readable by translating AS numbers to human-readable names.

In the upcoming parts of the project, it will be up to you to answer questions about the internet at large using the real routes that you've downloaded. You have a lot of flexibility in how you go about answering them. Much like a real researcher, you'll get to figure it out as you go! You can copy the JSON files to your local machine if you want or just write some scripts in you group's VM (but beware that project VMs may not have a lot of disk space. If you're feeling stuck, don't hesitate to reach out to the teaching team!

Part 2: Stanford Internet

Before analyzing the Internet, let's analyze your connection (and Stanford's Internet connections). For each of the following, **you must explain how you came up with your answer**.

1. How many routed prefixes do you see advertised to you? (Hint: Your answer should be between 500K–1M). How many of those are for routes outside of Stanford?

2. What is Stanford's (public) ASN? What public IP ranges does Stanford publicly advertise on that ASN? What other iBGP ASNs are used by Stanford? How did you determine this?

3. Based on your routing table, how would you determine who Stanford BGP peers with? Who are these ASes? Of those, which are Stanford's upstream Internet transit providers? Who are Stanford's peers topologically? Does Stanford peer with any content service providers (e.g., Google, Netflix, or Amazon)? Provide evidence to support your answers.

4. What routers on campus does your traffic pass through before reaching an upstream provider when connecting to 8.8.8.8? Which router is used to peer with our upstream provider? (Hint: Use traceroute.)

5. What ASes is your traffic to google.com routed through? What about to 78.134.74.239? In English, explain how your traffic gets to Google – who are these organizations? What role do they play based on what you can find online? Explain how you found your answers.

Part 3: Internet Topology

BGP forwards the best route to a given prefix to its neighbors, which means that edge nodes receive routes to *all* reachable prefixes. In this section, we'll use these routes to better understand how the Internet functions as a whole.

Note: the run-time for some of these may be long (e.g., several hours) depending on your solution, make sure you give yourself enough time and **you must describe your methodology**.

1. Do you see all public Internet peerings in the routing table you've constructed? Why or why not?
2. Do you see all private Internet peerings in the routing table you've constructed? Why or why not?
3. How many ASes do you see worldwide? How many ASes originate routes? What are the top 10 ASes that originate the largest number of prefixes and the largest number of IP addresses? Are there any ASes that transit traffic but do not originate any of their own prefixes? Please provide both the names and AS numbers (if possible).
4. Which ISPs appear to transit traffic for the most IP addresses (by either originating the prefix or transiting traffic for the prefix)?
5. Who are the largest "last mile" ISPs (i.e., ISPs that advertise lots of IPs but don't transit traffic for other networks)? Which ones advertise the largest number of prefixes and largest number of IP addresses?
6. Based on your routing table, what are the most connected ASes in terms of largest number of publicly inferrable peerings? Will your answer be skewed towards any ISPs in particular? If so, why?
7. Based on your routing table, who are the "best" connected ASes? In other words, who reliably has the shortest paths to other ASes? Describe how you determined these ASes.
8. Do you see evidence of AS path prepending? How many routes use path pre-pending (repeated ASes)?