

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 project, you will BGP peer with an upstream Internet transit provider to receive the global Internet routing table, which you will use to analyze the Internet’s routing topology.

We will provide you with a Linux virtual machine (VM), autonomous system number (ASN), IPv4 block to advertise, and privileges to BGP peer with an upstream transit provider. You will 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

CS249i studies the real-world Internet, which is constantly changing and nuanced. Unlike most homework or projects, you will find that questions about the Internet do not have a singular, “correct” answer. Internet measurements taken at different times, from different locations, or with different pieces of software can have dramatically different results. Many results can be interpreted differently and course staff may or may not know the answer as to why something is as it is. This lack of “ground truth” is what makes studying the Internet so interesting. Even among experts, there is no globally accepted list of Tier-1 ISPs; the ground truth is locked away behind legal contracts we will never see. This project is an exercise on understanding and reasoning about empirical observations of Internet activity to the best of your ability.

Some of the answers to the questions we ask can be found online. For instance, Wikipedia lists the globally accepted list of [Tier-1 Networks](#). However, based on what you can publicly observe, you would likely be unable to generate the same list. For example, there is little public data that suggests that Liberty Global is a Tier-1 ISP. In fact, most people probably would not recognize Liberty Global. Yet, Hurricane Electric’s peerings might hint that it is indeed a Tier-1 ISP. Therefore, you cannot simply copy answers found online for this project. You must use observations from the data you’ve collected to compose a compelling argument that justifies each of your answers’ correctness.

It is *critical* that you put conscious thought into your answers, “gut check” whether they make sense, and explain how you came to the conclusions that you did. Treat the projects less like an *exercise* and more as *research* where you need to make a conclusion backed up with evidence that you have collected, and then explain to the reader how you came to that conclusion. Google the networks you find! Check other resources. Consider how you could ask the question multiple different ways of the data you have. If your gut says your answer doesn’t make sense, reach out and ask the teaching team on Ed.

Project Contents

Part 1: Peer onto the Internet	2
Part 2: Stanford Internet	7
Part 3: Internet Topology	8

Part 1: Peer onto the Internet

In the first part of the project, you will 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 1, you will 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 (AS 65105, ESRG), 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 are 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 will be peering with is 171.67.69.1.

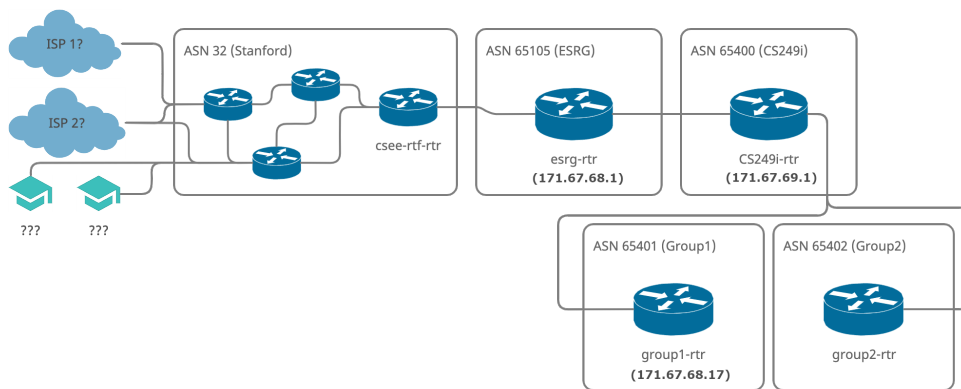


Figure 1: Network Topology

Group-Specific Details

Your group will receive an email with group-specific details that define your identity on the Internet:

- The public IP address of your group’s router (to connect to the upstream router)
- An assigned iBGP ASN to advertise
- An assigned /32 block of IP addresses that you will advertise on your ASN
- A private IP address for SSH’ing into your VM (*Note*: you will not be able to SSH using your router’s public IP address because it will not have public Internet access until you finish Part 1).

Connecting to your VM

You will need to SSH through a **bastion host** to access your VM since it does not have a public IP address (yet). We will provide you an exact SSH configuration to use in the email we send you with VM details. Do not modify the VM settings.

Initial Network Configuration

Your VM will be configured with two IP addresses: (1) a public IP address on the 171.67.69.0/25 subnet that you will use to route traffic, and (2) a private IP address in the 10.216.10.0/24 subnet that you will use only for SSH’ing into your VM, not for routing traffic. This configuration is similar to what you would see in the real world. Routers have a “management” interface that is managed separately from the data plane

on the router. That management interface will typically be a typical Ethernet port that is connected to an isolated management network whereas the router ports will be high speed ports for transferring data.

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 (since these do not require any L3 routing). 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
10.216.10.0/24 dev ens3 proto kernel scope link src 10.216.10.YYY
171.67.69.0/25 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 can your machine send traffic from?

Try running `ping 8.8.8.8`. As your machine currently has no routes to the Internet, you will receive a “Network Unreachable” result. The goal of Part 1 is for you to fix this!

Next, ping the CS249i router by running `ping 171.67.69.1`. This should work since the CS249i router is in the same L2 domain as your public IP address; therefore, you do not need an L3 route to reach the CS249i router.

Introduction to GoBGP

In order to peer with the CS249i router and connect to the Internet, you will use a software routing package called GoBGP. Despite being a software-based router, GoBGP uses a configuration that is very similar to that of a typical hardware router. In your VM home directory are two folders: `gobgp/` and `output/`. Inside `gobgp/` are four files: `gobgpd`, `gobgp`, `config.toml`, and `run-gobgp.sh`.

File Overview

`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 configuration file. You will receive metadata about routes and their corresponding routers 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 project.*

`run-gobgp.sh` is a bash script which will run the `gobgp` daemon and establish BGP connection as defined in `config.toml`, then automatically shut it down after 180 seconds.

Note: `run-gobgp.sh` runs `gobgp` interactively, rather than as a daemon, to make troubleshooting easier (i.e., logs are output directly to `stderr`). As such, if you want to run `gobgp` or troubleshoot while the

daemon is running, you will need to open a second SSH connection to your VM or use a utility like `tmux` or `screen` that allows for multiple shell sessions within one terminal.

To run the daemon for longer than 180 seconds, you can adjust the length of the `sleep` command in this file or run the daemon yourself—in which case, remember to manually provide the configuration.

Be 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` does nothing, as your config file is empty. Time to fix that!

Peering with the CS249i Router

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

- Establish the identity of your AS
- Identify the neighbor router you want to peer with
- Advertise your IP prefix and AS to your neighbor
- Receive routes to other networks from your neighbor

In this section, you will complete these steps by providing the appropriate configurations for your BGP session in `config.toml`.

Recall that in BGP, other ASes must know where and how to reach you. Therefore, your identity is composed of your ASN and your router's IP address. You can specify these in `config.toml` as follows:

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

Next, you will need to configure your connection with the CS249i router:

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

With just these seven lines, you are ready to peer with the CS249i router! Run the daemon, and you should see a successful connection established with the CS249i router. When you establish this connection, the peer router will advertise to you all of its known routes. However, based on the current configuration, you are not yet advertising anything to the peer router.

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

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

Pushing Routes to the Linux Kernel

Although you are now successfully receiving routes from the upstream CS249i router, these routes currently only exist in `gobgp`'s memory, in what is known as the Routing Information Base (RIB). None of the routes are pushed to the router's forwarding plane. Because we are using a software-based router for this project, the Linux kernel serves as the forwarding plane. 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, you will use Zebra, a complementary piece of routing software that interfaces between GoBGP and Linux. You can configure GoBGP to use Zebra by adding the following to `config.toml`:

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

At a high level, this configuration specifies that the GoBGP daemon should use Zebra and indicates where to connect the Zebra process. Go ahead and restart the daemon to reflect these changes.

Next, run `ip route | wc -l` to count the number of lines in your routing table as it populates with the upstream router's advertised routes. Your routing table initially had two entries, and after peering, it should now contain many, many more. The number of entries should eventually stabilize around 800K–1M routes. It may take a little bit of time for all of the routes to be installed.

Once you have all of the routes deployed in your kernel's routing table, you should be able to verify that you have Internet connectivity by running `ping 8.8.8.8`. If this is successful, congratulations: you have officially peered onto the Internet!

Unlike a normal computer, which simply sends all packets to the upstream router (through a single route for 0.0.0.0/0), 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 have peered with an upstream neighbor, and received other routes, but you have yet to advertise any IP prefix to your upstream provider. You can see routes because you are running commands on the router itself, but if you had any customers, they would still be disconnected from the Internet.

To fix this, you will need to advertise your AS's prefix:

```
./gobgp global rib add -a ipv4 <subnet-to-advertise>/32
```

This informs your neighbors that you own the /32 you were assigned and that any traffic sent to IPs within that prefix should be routed to you.

Dumping Routes

At this point, you have successfully fetched all of the routes to connect to the Internet by peering with the CS249i router and are advertising your own IP range onto the Internet. In Parts 2 and 3, you will explore and analyze Internet connectivity at Stanford and on the global Internet. To investigate the routes you receive from the upstream router, you will have to pull them from memory (in GoBGP) and dump them to a file for analysis. To do so, add the following to your GoBGP 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 directory every 2 minutes, with the date and time that the dump was generated as the filename. The MRT format is a binary format that is used to export routing protocol messages, state changes, and routing information base contents.

You can check if the dump has completed by running:

```
wc -c /home/cs249i-student/output/table.mrt
```

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

Note: gobgp will append to the dump file once every two minutes. Unfortunately, there is no way to configure GoBGP to overwrite the file instead. Thus, you must be careful that you shut off the daemon after the file has been written to once to avoid duplicating or corrupting data. This will be *handled automatically* if you run gobgp with `run-gobgp.sh`. This script also erases the old output table if it exists; if you want to preserve multiple tables, you should rename the previous tables.

MRT is a specific binary filetype designed for dumping BGP data. For analysis purposes, we will convert it to a more human-readable format, **JSON Lines**. This convenient format can later be analyzed however you want (e.g., by Python script).

To convert it to JSON Lines, you can use the `mrt2json` utility, found in your home directory as a binary file. Convert the MRT file to JSON Lines by running:

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

In the output file, `table.json`, each line contains a JSON object for the record of the routed block.

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,  
      "originated_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": "171.67.69.1"  
        },  
        {  
          "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 RIB entries. RIB entries, typically used by a router, tell you the paths (routes) that the router can take to reach a specific prefix. You can find the specifics of each path attribute field [here](#), but the most important to understand are the ones listed as Type 2 and Type 3.

The above example of a Type 2 attribute, or the `AS_PATHS` entry, gives you a path from your AS to the prefix `103.127.54.0/24`, beginning at our upstream CS249i router (AS 65400) and ending at AS 17995. This ASN belongs to the AS that advertised a route for the specified prefix. 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 the appropriate neighbor to reach your destination prefix.

In the remaining parts of the project, you will answer questions about the Internet using the real routes that you obtained through BGP peering. To explore the JSON dumps more closely, you can copy the files to your local machine using `scp` or write analysis scripts in your group's VM (but beware that project VMs may not have a lot of disk space).

Finally, within `output/` is another file, `asn_names.txt`. This file maps ASNs to their corresponding human-readable names and country codes. You can use this file to reason about the MRT dumps.

If your group feels stuck, please don't hesitate to reach out to the teaching staff. Good luck!

Part 2: Stanford Internet

In Part 2, you will analyze your VM and Stanford's Internet connections using your collected routes.

Solution Requirements. Each of your answers **must include a conceptual explanation** of your approach. Your explanations should ground your answers in empirical observations made from your routing table and the provided ASNs list—you should not need other tools, such as `whois`. *Do not* provide the code you wrote in lieu of an explanation.

- (4pt) To start, let's build some familiarity with your routing table data.
 - How many routed prefixes do you see advertised to you? (Hint: Your answer should be between 500K–1M).
 - How many of those routes are from organizations outside of Stanford?
 - How many of the prefixes advertised to you overlap with at least one other advertised prefix?
 - Why might overlapping prefixes be advertised to you?
- (5pt) Stanford has multiple ASes. In this question, you will explore what they are.
 - What ASes belong to Stanford? Draw a diagram showing their relationships: extend Figure 1 up to our ISPs. You do not have to include iBGP ASNs. (Hint: You should see between 5-10 ASes)
 - What is Stanford's main AS? How did you deduce that from your data?
 - What IP ranges does Stanford publicly advertise on the Internet from its main AS? Merge prefixes for readability.
 - For each of Stanford's other public ASes you identified in (a), what role might it serve for the university?
- (5pt) Now let's extend your investigation to Stanford's peers.
 - Based on your routing table, which ASNs does Stanford BGP peer with? (Hint: Consider all of Stanford's ASes. If you don't see AS2153 in your list of peers, something is probably wrong.)

- (b) Who are the organizations that own these ASes? Sort them into two groups: (i) Stanford's upstream transit providers, (ii) Stanford's topological (and likely settlement-free) peers.
 - (c) Explain what these organizations do and why establishing settlement-free peering with Stanford might be mutually beneficial.
 - (d) Choose an organization that you are surprised to *not* see among Stanford's peers. Why would you have expected this peering, and what does your routing table tell you about how Stanford traffic currently gets routed there instead?
4. (8pt) Finally, let's examine how your traffic actually flows through Stanford's network.
- (a) Use `traceroute` to examine what routers your traffic passes through when connecting to 8.8.8.8 (Google's DNS resolver). Use the output to answer and explain each of the following:
 - i. Which routers are on campus?
 - ii. Which router peers with our upstream provider?
 - iii. Who do the remaining routers between campus and 8.8.8.8 belong to?

Please include both your `traceroute` output and your interpretation of it. (Hint: You can adjust the `traceroute` options and flags to reduce the asterisks in your output.)
 - (b) Now run `traceroute` to see how your traffic to google.com is routed. Who are these organizations, and does your traffic take the same path to Google's infrastructure as it did for 8.8.8.8?
 - (c) Let's examine a path to a non-Google destination. Using `traceroute`, what ASes is your traffic routed through on its way to 78.134.74.239, and where geographically do you infer your traffic is passing through?
 - (d) Compare your `traceroute` outputs from (a)-(c) with your routing table. What paths would you have expected your traffic to take based on your routing table? Do they match your `traceroute` outputs?
 - (e) Your `traceroute` output may not have always matched your routing table data when you compared them in (d). Why might they differ?

Part 3: Internet Topology

BGP forwards the best route for a given prefix to each of its peers, which means that edge nodes receive routes to all reachable prefixes. In Part 3, you will use these routes to better understand how the Internet functions as a whole.

Note: The run-time for answering some of these may be long (e.g., several hours) depending on your solution, so make sure to give yourself enough time. As in Part 2, **you must explain your approach for each question.**

1. (1pt) To start, let's consider our vantage point. Do you see all BGP peerings in your routing table? Why or why not?
2. (5pt) Now build some intuition for participation on the public Internet.
 - (a) How many ASes do you see worldwide on the public Internet?
 - (b) Take a random sample of 20 ASes in your dataset. What type of organizations do they belong to, based on what can you find online?
 - (c) Are there any public ASes that transit traffic but do not originate any of their own prefixes? If so, how many are there, and which are the top 3 that show up the most in your data? Please provide ASNs, names, and a short description of what these organizations do.
3. (3pt) Now, you will measure one particular AS behavior: path prepending.

- (a) Do you see evidence of AS path prepending? What percent of the routes in your data use path prepending (repeated ASes)?
 - (b) Do any Stanford ASes use prepending? If you remove these from consideration, how does that affect your results from (a)?
 - (c) What are the 3 longest chains of prepending that you see in your data?
4. (5pt) Next, let's focus on some of the big players.
- (a) What are the top 10 ASes that originate the largest number of *prefixes*? Specify both ASN and name. (Hint: If you don't see Amazon in your list, something is probably wrong.)
 - (b) What are the top 10 ASes that originate the largest number of *IP addresses*? Specify both ASN and name. (Hint: If you don't see Comcast in your list, something is probably wrong.)
 - (c) For both of your lists from (a) and (b), what kinds of organizations do they contain (e.g., cloud providers, last-mile ISPs, mobile carriers, etc.)? Use this to explain your intuition for why these ASes are so large.
 - (d) Which 10 ISPs appear to transit traffic for the most IP addresses, by either originating the prefix or transiting traffic for the prefix? (Hint: You will need to filter your results to just the ISPs. HE should be in your list; DNIC should not be.)
 - (e) Why does your answer to (d) depend on your vantage point?
5. (5pt) Finally, let's look at well-connected ASes.
- (a) Based on your routing table, what are the 10 most connected ASes in terms of largest number of publicly inferrable peerings? (Hint: if you don't see Cogent in your list, something is probably wrong.)
 - (b) Is your answer to (a) skewed towards any ASes in particular? If so, why?
 - (c) Create a concrete definition for the "best" connected ASes: in other words, those that reliably have the shortest paths to other ASes. Why is your definition a good way to computationally represent this concept? (Hint: Draw out an example RIB to evaluate your definition.)
 - (d) Using your definition from (c), who are the "best" connected ASes? Provide both ASNs and names, and explain your process for validating your results.