Build your own Internet - Stage C

Computer Networks: Project 2

Current version: **12/12/2020**
Corrections:

1. **3.3.2 Example: community value 251:30**
2. **Looking Glass is available for all ASes- 2.4 Note modified.**
3. **Clarified IXP vs route server**
4. **3.4.3: updated stub AS router to say ZURI (not NEWY)**
5. **Further details on submission format for OSPF weights here and here. Details on fetching web content here.**

1 Introduction

In this assignment, you will build and operate a layer-3 network using traditional distributed routing protocols, how different networks managed by different organizations interconnect with each other, and how protocols, configuration, and policy combine in Internet routing.

More specifically, you will first learn how to set up a valid forwarding state within an autonomous system (AS) using OSPF, an intra-domain routing protocol (Stage A). Then, you will learn how to set up valid forwarding state between different ASes, so that an end-host in one AS (e.g., your laptop connected to the university wireless network) can communicate with an end-host in another AS (e.g., Google's server). To do that, you will need to use the only inter-domain routing protocol deployed today: BGP (Stage B). After that, you will implement different BGP policies to reflect business relationships or traffic engineering that exist in the real Internet (Stage C). You will configure both OSPF and BGP through the FRRouting Software Suite, which runs on several software routers assigned to you.

We first describe the setup you will have to use (Section 2). Then, we list the tasks you should perform (Section 3), submission and other general information (Section 4), and the collaboration and academic integrity policies. We are also providing a separate document giving a crash course on how to configure FRRouting routers.

1.1 Schedule

- ***You can use slip days and submit late for Stages C, but you cannot use slip days or submit late for Preliminary Stage, Stage A, or Stage B--any late submissions for the Preliminary Stage, Stage A, or Stage B will receive a 0.***
- The deadline for Stage C of this project is Dec. 14, at 11pm.
    - 2020/12/12 update: Since we just added some details on what to submit, we will not count assignments as late until Dec. 15 at 11pm, and you can use slip days or take late penalties after that time.

1.2 Collaboration policy

This is an *individual project*, but you can discuss at a conceptual level with other students or consult Internet material, as long as the final code and configuration you submit is completely yours and as long as you do not share code or configuration. Before starting the project, be sure to read the collaboration and academic integrity policy later in this document.

1.3 Instructor/TA information

As explained in the Preliminary Stage document, we will assign each student an autonomous system. Each student is assigned to a TA, as listed on Piazza. Please ask any general questions related to this assignment on Piazza, visible to all unless they reveal private details of your solution, and only contact your responsible TA for questions that will not be relevant to other students (e.g. you are unable to access your autonomous system).

2. General project setup

See the Preliminary Stage document for further details.

2.1 - Network Topology

2.1.1 Intra-AS topology (same as previous stages, with more details added)

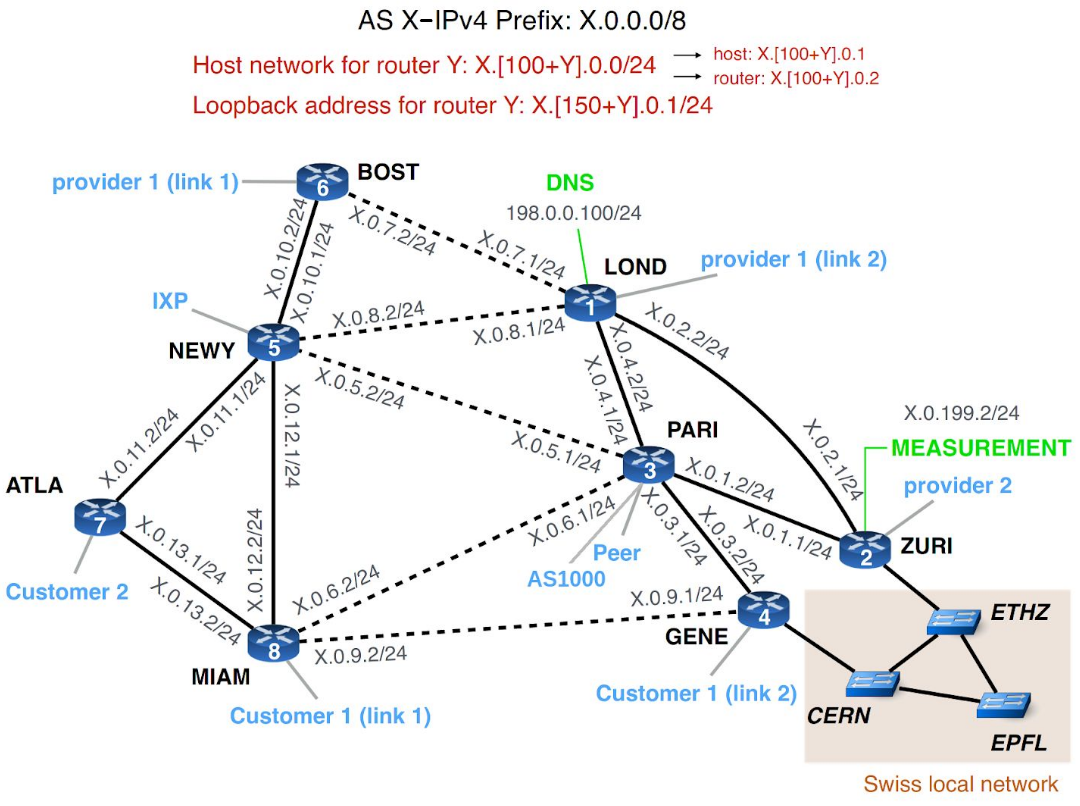

Figure 1: The network topology of your AS. Your AS is composed of 8 routers. A /8 prefix has been assigned to you. You can use it to configure your local networks. The subnets you must use for each of your local networks are indicated on each interface.

The intradomain topology of your network remains the same as described in the Preliminary Stage and Stage A. However, we have added one more detail to the figure:

- Every router has an external connection to one of your neighboring ASes. Some are connected to a provider, some to a customer and others to a peer. NEWY is connected to an Internet eXchange Point (IXP).
    - *PARI has a connection to AS1000 (which you configured in the Preliminary Stage) and to an additional peer, as shown in the updated Figure above.*
- The fact that some links are solid and some are dotted is **FINALLY** relevant.

2.1.2 Inter-AS Topology

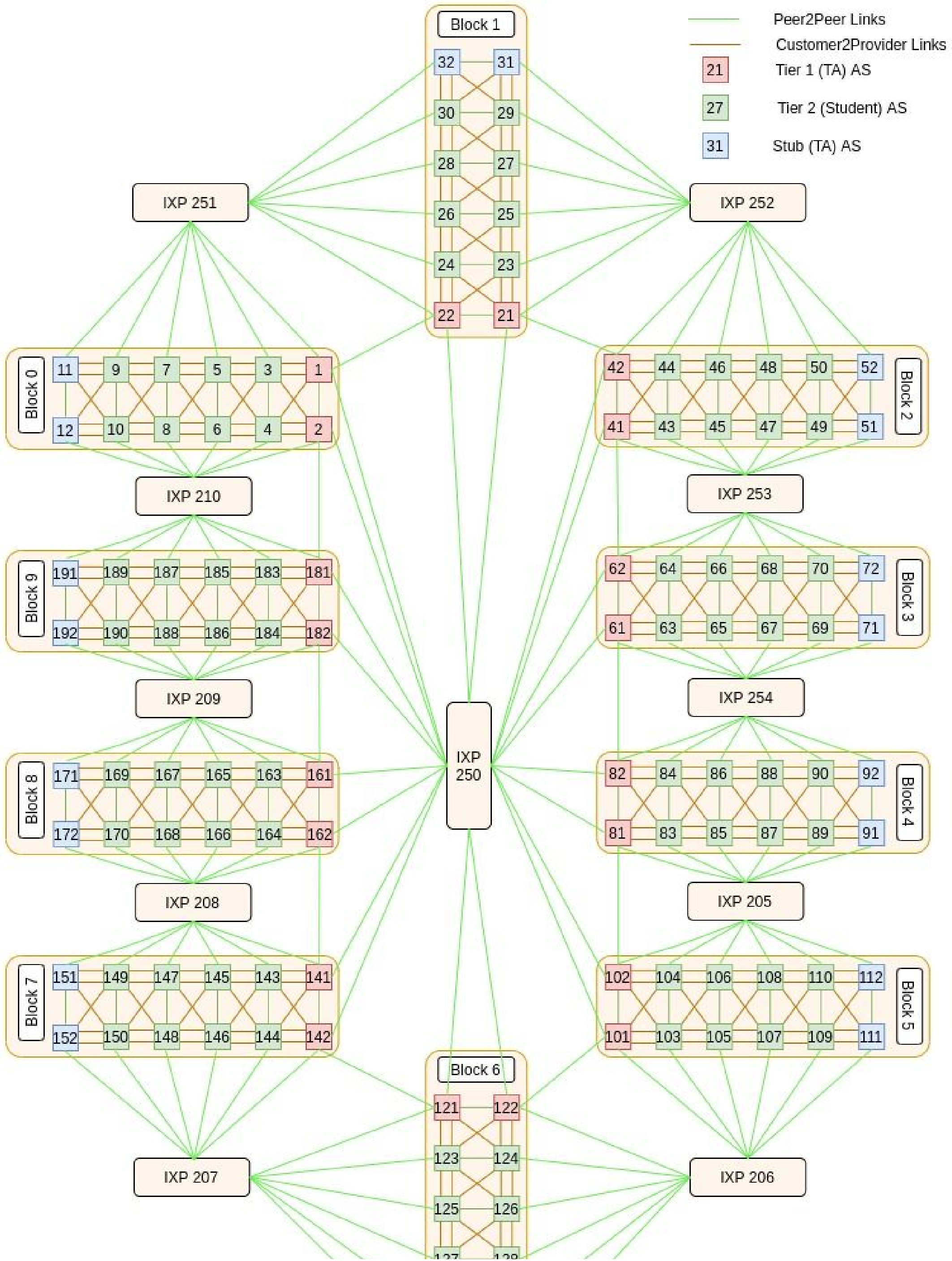

Figure 2: The interdomain topology.

- The class internet is divided into 10 blocks (numbered 0-9)
- There are eleven IXPs within our Internet. IXPs can make it easy to interconnect with many other ASes.
    - The IXP functions as a layer2 switch. At layer3 (IP), this just means that instead of a point-to-point link, it is a multi-point subnet that can have many routers with interfaces on it, not just two.
    - One advantage of using an IXP is that an AS can directly peer with another AS through the IXP, instead of reaching it via a provider that it has to pay.
        - The following example illustrates the benefit of being connected to an IXP: AS6 can send traffic to AS187 via the IXP210, instead of paying AS 4 to send the traffic via the path 4-2-181-183-105 if IXP210 is not used.
    - Another advantage is that only one physical connection with an IXP is needed to potentially interconnect with all the other IXP participants. An IXP operates a switch, and multiple ASes connect to the switch. An AS can connect to other ASes across the switch via pairwise (bilateral) BGP sessions, or by a (multilateral) BGP session to a Route Server, which redistributes routes among all participating ASes that connect to it. We will use Route Servers at our IXPs.
        - The router server lets you establish one BGP connection to the router server, and then you can send your announcement through it to all the other ASes that connect to it, and receive announcements from them. So the control plane goes your router -> (L2 IXP switch) -> route server -> (L2 IXP switch) -> other routers, but the data plane goes your router -> (L2 IXP switch) -> other router. In Stage C tasks, you will control which ASes receive your announcement from the route server, and from which ASes your router will accept routes from the route server.
    - One IXP is connected to all the Tier1 ASes, allowing them to be connected in a full-mesh fashion. The other IXPs are always interconnecting two blocks. This enables these ASes to peer between them (as long as they respect the BGP customer/provider policies), instead of using (and paying!) their providers.
- Every student AS is connected to five other ASes, and one IXP.
- The neighboring ASes in your block that are closer to the block's Tier 1 ASes are your providers.
    - Eg: For AS 123, ASes 121 and 122 are providers
    - Eg: For AS 108, ASes 105 and 106 are providers.
- The neighboring ASes in your block that are closer to the block's stub ASes are your customers.
    - Eg: For AS 123, ASes 125, 126 are customers.
    - Eg: For AS 108, ASes 109 and 110 are customers.
- The neighboring AS in your block lateral to your position is your peer.
    - Eg: AS 123 and AS 124 are peers.

2.2 The measurement container

- We have set up a measurement container which will enable you to launch traceroute from any Tier2 AS (and not necessarily only your own AS), towards any destination in the class's internet. This will help you to know the paths used towards or through your network. The entire class will share a single measurement container.
- The measurement container is connected to each AS via the interface **`measurement_X`** (where X is your AS number) of the router **`ZURI`**. This interface is preconfigured to use the IP address X.0.199.1/24 (see Fig. 1), where X is your AS number.
    - For example if you are AS 15, your interface **`measurement_15`** at ZURI is preconfigured to use 15.0.199.1/24.
- The X.0.199.1/24 subnet must be reachable from anywhere in your network. You must therefore add it in your OSPF configuration.
- To access the measurement container, use the following command:
  ```
  > ssh -p 2099 root@34.72.133.64
  ```
  The password is posted on Piazza.
- To launch a traceroute, you can use the script **`launch_traceroute.sh`**, which takes as arguments the AS number from which the traceroute starts and the destination IP address (possibly in another AS).
    - For instance,if you want to perform a traceroute from AS 11 to 42.107.0.1 (i.e., the host connected to ATLA in AS 42), just use the following command in the measurement container:

      ```
      > ./launch_traceroute.sh 11 42.107.0.1
      ```

  Note that the traceroute will start from the router ZURI of AS 11, since the measurement container is connected to that router.

2.3 The Connectivity Matrix

For the ease of debugging, we have set up a connectivity matrix that shows you whether an AS can ping another AS. You can access the connectivity matrix from your browser at the following link:

`http://34.72.133.64:2500/matrix.html`

The entry X,Y is the result of AS X pinging AS Y from the matrix interface connected to your PARI router (pre-configured) to the destination AS's ATLA host. The matrix updates ~every 15 minutes, reflecting reachability in the most recent snapshot. Before you set up eBGP sessions, everything will be red. As you set up the eBGP sessions with your neighbors, the matrix will turn green for some pairs of ASes. At the end of this assignment, we hope to see this matrix completely green!

2.4 Looking Glasses

In general, a Looking Glass is a tool that helps us peek into a router's routing information. We have set up a similar server which will enable all the students to look at the BGP routes being received by an AS (not necessarily yours).

To access the Looking Glass, use the following link template:
`http://34.72.133.64:2600/X_[Router]router.txt`

where X is the AS you want to peek into, and [Router] is the name of the router within the AS. For example, http://34.72.133.64:2600/8_BOSTrouter.txt gives you the output of 'show ip bgp' command run at that particular router.

The information on the web interface is updated every 5 minutes.
Note: The Tier1 and Stub ASes run a smaller topology, and thus have only two routers- ZURI and LOND.

3 Your Tasks

This project is composed of a very short preliminary Stage, then three main Stages (A-C), with Stage B being primarily completed during the class-wide "Internet Hackathon."

- Stage A involves setting up routing within your own network via OSPF and iBGP configuration, and must be finished before the Internet Hackathon.
- Stage B (Internet Hackathon) involves bringing up your eBGP sessions with your neighboring ASes and advertising your prefixes to your neighbors. We will provide details closer to the date of the Hackathon.
- Stage C involves implementing BGP policies according to the business relationships that you have with your neighbors (we will assign the relationships). We will provide details closer to the date of the Hackathon.

Possible plan of attack:

- Familiarize yourself with the previous section, and, using the instructions in the tutorial, access your AS and navigate to routers/hosts.
- For any stage, familiarize with the goals of the stage. Then, refer to the tutorial we provide to find the basics of the commands you will need to enact the goals. Our expectation is that you may need to experiment and try things out to figure out how to accomplish a task based on the guide.

3.1 Preliminary Stage: Your first BGP session (30 points)

Please look at the preliminary stage document for this section of the project.

3.2 Stage A: Configure IGP and iBGP (40 points)

Please look at the Stage A document for this section of the project.

3.3 Stage B: Establish BGP interconnectivity (50 points)

Please look at the Stage B document for this section of the project.

3.4 Stage C: Routing policy & content delivery (80 points)

3.4.1 Task - Implement intradomain routing policy (10 points)

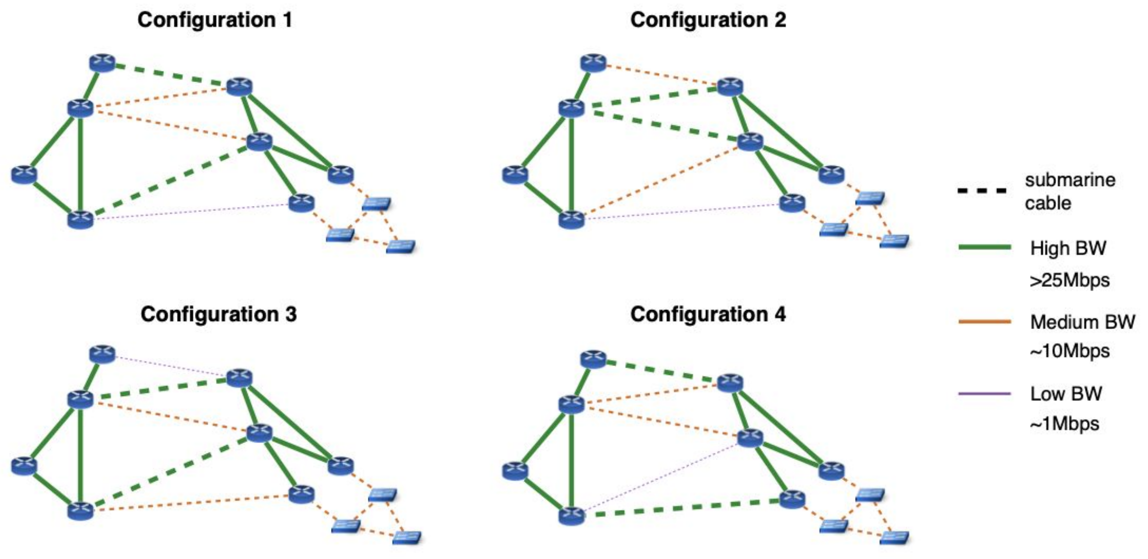

Fig 3: The bandwidth (BW) of your submarine cables follows one of the depicted configurations.

As a network operator, your goal is now to provide the best performance to your customers. In this task, your goal is to minimize latency and prevent traffic congestion.

Your top priority is to minimize latency, and to do so you must configure OSPF weights such that the traffic never traverses two submarine links (dashed links in Figure 2). For example, you do not want the traffic from BOST to MIAM to pass through Europe, but to stay on the same continent. Then, to minimize congestion, you must configure the OSPF weights such that submarine paths with higher bandwidth are preferred whenever it is possible.

1. For the submarine links, there are four different bandwidth configurations as shown in the figure. In a first step, identify the configuration your AS has by using iperf3.
2. Once you have identified it, you can assign the weights such that the high bandwidth links are preferred.
3. Assign weights such that traffic from MIAM to NEWY is load balanced evenly between the two paths MIAM-NEWY and MIAM-ATLA-NEWY
4. Assign weights such that the traffic from ZURI to LOND is load balanced evenly between two paths ZURI-LOND and ZURI-PARI-LOND.
5. As a final requirement, you need to balance the traffic between ATLA and ZURI across the two submarine links with high bandwidth.

In your report: List OSPF weights you used (***see updated format next)***. For goals (1)-(5) provide evidence (with explanation) that you achieved the goals. Include a traceroute from ATLA-host to the ZURI loopback interface. Comment on your traceroutes: do you see what you expect according to the weights you have configured, why?

In your report, you can list the OSPF weights in the format of your choice, for example by including the topology figure and adding annotations of the link weights. ***In addition***, please submit a file **`as`*<X>*`-links.txt`** ASCII file (where ***<X>*** is your AS number) with the file format: `ROUTER1 ROUTER2 weight\n` to mean that the cost from `ROUTER1` to `ROUTER2` is `weight`
Please note the following:

- The format is one link per line, space separated values on the line, and ending each line with a newline character (and not with the slash character and the n character).
- The links are ordered. For example, list `ATLA NEWY 100` for the link from `ATLA` to `NEWY`, and `NEWY ATLA 100` for the link from `NEWY` to `ATLA`.

3.4.2 Task - Advertise your prefix at the IXP (10 points)

By default, we have configured the IXPs' route servers to not relay your BGP advertisements to their other peers. To announce a prefix to another peer via a router server, you must specify it using a BGP community value. Router servers are configured to relay a BGP advertisement to a peer **`X`** if the advertisement has a community value equal to **`N:X`** with **`N`** the IXP number. For example, if you are AS7 and you want to advertise a prefix to AS30 via the IXP251, you must add the community value **`251:30`** in your BGP advertisements.

Use the community values to send BGP advertisements to the peers connected to you through an IXP. For now, send the advertisements to all ASes connected to the IXP that are *not* in your block.

To include in your report: Take a screenshot of the relevant parts of the out route-map you configured on the session from the router in NEWY to the IXP. In a few sentences explain what all the lines in the route-map mean and do. Then, show a Looking Glass entry of another AS which proves that your prefix has been advertised through the route server. Finally, use the measurement container to perform a traceroute from another AS (in another region) to your AS for a destination where the traffic should go through the IXP. Show the result in your report.

3.4.3 Task - Implement policy at IXP (10 points)

The AS topology (see figure) is divided into blocks. Configure your BGP policies such that you can leverage your connection with your IXP at NEWY. You **do want** to peer through this IXP with ASes that are located in another block. However, for business reasons, you **do not want to peer through this IXP with ASes that are located in your block**. To not peer through the IXP with ASes in your block, you must (i) not advertise any prefixes to them and (ii) deny any advertisements coming from them.

To help you check whether you properly configured (ii), we have configured the stub ASes to advertise their prefix to all the ASes connected to their IXP.

To include in your report: Again, take a screenshot of the relevant parts of the route-map at NEWY and explain what the different lines mean and do. Show that the advertisement from the stub AS in the same region and connected to the same IXP as you are denied by showing the result of a `show ip bgp` in your router NEWY. For clarity, you do not need to write the full output, just the part that is interesting (i.e., the part which could have the prefix of the stub AS). Then, include in the report the output of the Looking Glass for the router ZURI of the stub AS in the same region and connected to the same IXP as you. Finally, include in your report the output of the Looking Glass for a group in another region but connected to the IXP. When you include the output of a Looking Glass in your report, only keep the parts that prove the correctness of your configuration and omit the irrelevant ones.

3.4.4 Task - Implement BGP routing policy (20 points)

Configure your local-preference as well as the exportation rules to implement your customer/provider and peer/peer business relationships with neighboring ASes, according to the topology from the Stage A diagram. Configure AS1000 (from the Preliminary Stage) to be your customer. You should implement (1) no-valley routing and (2) prefer-customer routing. No-valley routing specifies that you should not advertise routes learned from one provider/peer to another provider/peer, and prefer-customer routing specifies that, if you can reach a destination through multiple AS-paths, you should always prefer the paths that go through your customers first, then peers, then providers. If you have paths through multiple neighbors of the same class, prefer-customer does not specify which to choose; the decision should depend on other BGP tiebreakers.

Hint. To implement no-valley routing, we advise you to use BGP communities to keep track of where the routes have been learned, and set up export rules based on BGP communities. You can use other BGP attributes as well if you like. However, it is a requirement that all best routes that would not cause a valley should be announced. Similarly, all the routes being announced shouldn't result in a valley. Also, your solution should not depend on the actual prefixes announced by your neighbors, and it should continue to work however your neighbors change what they are announcing.

To partially test that your BGP policies work correctly, you can use the Measurement container to launch traceroute from another AS, as mentioned in Section 2.2, or a Looking Glass. For example, you can launch a traceroute from one of your peers towards another peer, and verify that your AS does not provide transit service between these two peers. You should also test by checking what routes you are advertising to each neighbor (**show ip bgp neighbor <NEI-IP> advertise)**

In the report, describe (1) how you use BGP attributes to implement no-valley routing, (2) how you implement prefer-customer routing, (3) the actual router configuration you made, (4) evidence to support that you have implemented these two policies correctly, and (5) explanation about how to interpret the evidence.

3.4.5 Task - Implement inbound traffic engineering (10 points)

Part 1: the goal will be to configure your BGP policies in order to influence the inbound traffic destined to your own prefix. More precisely, your goal is to configure BGP policies such that the inbound traffic coming from a provider and destined to your own prefix uses the provider connected to ZURI in priority.
Special case: If you find that your ZURI provider does not work and your other provider does, and you are sure (and have triple checked) that your configuration is correct and the provider has a problem, please skip this step for now and, by Dec 10 11pm (4 days before the deadline) post a private note on Piazza addressed to Instructors and titled "Primary provider AS***X*** not working for AS***Y***" (substituting the AS numbers for X and Y). We will try to help you resolve the problem and will not penalize you if the problem is out of your control.

To include in your report: Explain in a few sentences the technique you used and discuss any potential drawbacks. Then, include the result of the Looking Glass for both of your providers. You can omit parts of the output that are irrelevant, and only show the part that shows that your configuration is correct (i.e., the part where your own prefix is shown).

Part 2: the goal will be to configure your BGP policies such that the inbound traffic coming from the provider with whom you have two external BGP sessions arrives in priority via your router BOST (instead of LOND).

To include in your report: Explain in a few sentences the technique you used and discuss any potential drawbacks. Then, include the result of the Looking Glass of your provider with whom you have two external BGP sessions which shows that your configuration is working as expected (for instance from the router MIAM). You can omit parts of the output that are irrelevant.

3.4.6 Task - Fetch content across Internet (20 points)

During the Preliminary Stage, AS1000 was announcing a single route (222.0.0.0/8) to you. We have reconfigured it to announce an additional route to you, which you should be receiving at your PARI router if you have configured it properly.

The route is a /32, meaning it is a single IP address. That IP address is hosting a web server. For this Task, you will fetch content from that web server, demonstrating that the internet (and in particular your network) is working end-to-end!

1. The web server is running on the IP address that is announced to you. Figure out its IP address by looking at the BGP announcements you are receiving.
2. It is running on port 4119.
3. You need to issue an HTTP GET request to that IP address and port for the content "`/stagec/stagec.txt`"
4. For full credit, you must fetch the file from your LOND host.
5. The file `stagec.txt` will end with a ten character code.
6. You will submit a file called `stagec_code.txt` in the following format:

`uni,YOUR-AS,code`

For example, if Ethan is AS20 and receives a code 'THISisCODE':

```
ebk2141,20,THISisCODE
```

Notes:

- The hosts do not have standard linux tools for fetching content, like `wget` or `curl`. Instead, we set up a download script that you can use via commands like '`root@NEWY_host:~# download <URL>`'. You can view the script in the host's `.bashrc` (for example, via '`cat ~/.bashrc`'). The script operates in an unusual and interesting manner. You may find it interesting to read up on it and work to understand how it works.
- Please note that the file you submit includes the code from the web content you fetch, but the rest of the file format is not identical, so copy the code into a new file with the appropriate format.
- Recall that AS1000 is your customer. For full credit, make sure you have properly configured your routing policy, and make sure that you don't set LOCAL-PREF to different values for different customers.
- If you cannot get it to work from LOND host, you may fetch from another host or router for partial credit. In descending order of credit:
    a. LOND host worth the most credit
    b. Any host other than LOND host or PARI host
    c. PARI host for the least credit
- You are welcome to fetch the file multiple times if you improve or change your configuration. The code changes (even if you fetch from the same place twice in a row),

so just submit the code corresponding to your best configuration, and we will use the code you submit to know which fetch you consider your best one.

- Make sure to submit the code from your “best” fetch.
- You may not share the code with anyone in the class. Doing so will be considered a violation of the academic integrity policy and will be subject to the policy described below.

4 Submission and other information

The project is worth 200 points (Preliminary Stage: 30 points, Stage A: 40 points, Stage B: 50 points, Stage C: 80 points). In reports that you submit, clearly label which question you are answering with your answer/screenshot by writing the Task number and name, e.g., “4.1.3 Task - Advertise prefix to AS1000.”

Remember to run ./save_configs.sh, copy the corresponding file generated to the machine you plan to submit from, and rename the folder before submitting to make sure you submit the version you intend.

You will submit it via Courseworks. The format of your submission is the same as your preliminary stage.
You must include the following files in a compressed file called **project2c_Lastname_Firstname_ASN.zip** (for example, *project2c_Katz-Bassett_Ethan_AS10.zip* if the professor were AS10 in the topology). The files are:

1. Written report, with filename **report.pdf**
2. The code you fetched from the webserver in a file stagec_code.txt with the format described above.
3. The `as<X>-links.txt` ASCII file in the format explained above.
4. The entire saved **configs[date][time]** directory on your AS under home directory renamed as **configs**.

The result of the above will be a list of text files named as **routers (PARI.txt, NEWY.txt, ATLA.txt etc) or Switches (ETHZ.txt, CERN.txt, EPFL.txt)**. Zip this folder along with your report.pdf. Your zipped submission file, for example, should have the following directory structure **after being unzipped** (the top level must contain the project2c* directory):

```
project2c_Venkatagiri_Shiv_AS7/
├──── report.pdf
├──── stagec_code.txt
├──── as<X>-links.txt
├──── configs
│    ├──── ATLA.txt
│    └──── PARI.txt
```

```
|   ├──.
|   ├──.
|   ├──.
```

5 Academic integrity: Zero tolerance on plagiarism

The rules for Columbia University, the CS Department, and the EE Department (via SEAS: 1 and 2) apply. It is your responsibility to carefully read these policies and ask the professor (via Piazza) if you have any questions about academic integrity. **Please ask the professor before submitting the assignment, with enough time to resolve the issue before the deadline**. A misunderstanding of university or class policies is not an excuse for violating a policy.

This class requires closely obeying the policy on academic integrity, and has zero tolerance on plagiarism for all assignments, including both projects/programming assignments and written assignments. By zero tolerance, we mean that the minimum punishment for plagiarism/cheating is a 0 for the assignment, and all cases will be referred to the Dean of Students.

This assignment must be completed individually. For programming assignments, in particular, you must write all the code you hand in yourself, except for code that we give you as part of the assignments. You are not allowed to look at anyone else's solution (including solutions on the Internet, if there are any), and you are not allowed to look at code from previous years or ask people who took the class in previous years for help. You may discuss the assignments with other students at the conceptual level, but you may not write pseudocode together, or look at or copy each other's code. Helping other students violate the policy (for example, letting them look at your code) is a violation, even if you completed the code yourself. Please do not publish your code or make it available to future students -- for example, please do not make your code visible on Github. Uploading course materials to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10).

You may look at documentation from the tools' websites. However, you may not use external libraries or any online code unless granted explicit permission by the professor or TA. For written (non-programming) answers, if you quote material from textbooks, journal articles, manuals, etc., you **must** include a citation that gives proper credit to the source to avoid suspicion of plagiarism. If you are unsure how to properly cite, you can use the web to find references on scientific citations, or ask fellow students and TAs on Piazza.

Deliberately disrupting shared project infrastructure or the ability of others to use shared infrastructure is a serious violation. If you are unsure how to use shared infrastructure, please ask classmates or instructors first.

For each programming assignment, we will use software to check for plagiarized code. **So, be really careful and do not read or copy code or text.**

Note: You must set permissions on any homework assignments so that they are readable only by you. You may get reprimanded for facilitating cheating if you do not follow this rule.