CS463 Systems and Networking Lab **Fall 2017**

Lab 1: Internet and ISP Networking

Instructor: Matthew Caesar

1 Introduction

In this lab you'll learn how to build your own Internet Service Provider (ISP) network. We will not be working with simulators or dummy tools - you will get the "real experience" working with the actual software that runs in carrier grade routers. You gain hands-on experience in how these routers work, the protocols that run between them, and how to implement configurations using configuration languages.

2 Background

When you connect to the Internet from your home or office, you typically purchase service from an ISP. An ISP runs a network that transits traffic between various locations. An ISP will often maintain "points of presence" (PoPs) in various locations - places where it provides service. At each of these locations it will run a (possibly leased) wire/connection to other PoPs. Each PoP consists of a set of routers.

In this lab you'll build a simple ISP network. Each point of presence will be represented by a single router. You will create long-haul links by configuring interfaces on each of the routers. You will then set up routing protocols. Routing protocols are distributed algorithms that run across routers and which help them figure out which way to route packets. ISPs typically run two protocols: an "EGP" (Exterior Gateway Protocol), and an IGP (Interior Gateway Protocol). As you can probably guess from the names, one runs inside (interior) the ISP's network, and the other (exterior) runs outside the ISP's network (to distribute routes to other ISPs).

To do this lab you will use the GNS3 (https://www.gns3.com/) network virtualization platform. GNS3 is a powerful tool because it lets you run real router images. You will be getting the "full experience" working with real operator software. Not the hardware yet - that will come in a later lab. Please be careful with GNS3 - it is advanced software and accidents can happen. Make sure you regularly save your work. You will be required to turn in the configuration files you create as part of this assignment.

3 Setting up GNS3

Under the hood, GNS3 is basically some software that glues together a bunch of VMs. Each VM runs the operating system for a router. GNS3 provides a visual environment for you to do various operations on those VMs, e.g., connect them together.

Follow the steps below:

(1) Download and install GNS3. You can install it on your laptop or lab machines. GNS3 runs under MacOS, Windows, and Linux.

(2) Play around with it and get a sense of how it works. Watch tutorials on youtube. Set up a small test network, connect some routers together. Log into router consoles and type configuration commands. Just get a sense of how things work.

(3) To save/load your configs, you can choose the Import/Export node configs option from the Tools menu from where you can save/load configs into a directory.

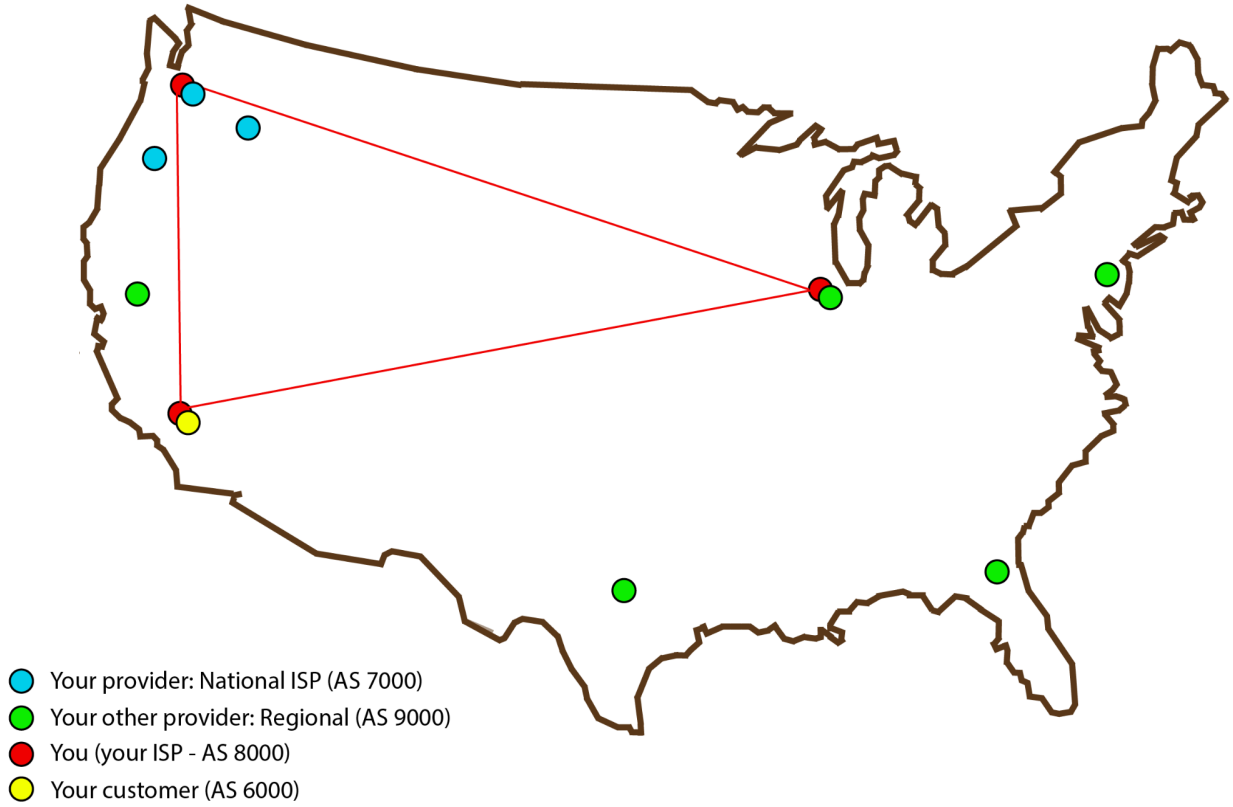

Figure 1: Network of ISP's

4.1 Design the topology

As a first step, we need to design the network topology. You're an ISP and (most) ISPs are businesses with an aim of making money. What kind of network will make you the most money? You have a fixed amount of seed capital, so you can't put PoPs everywhere. Should you put one in New York City or in Minneapolis? How big of routers should you put in each city, do you want to handle 10Gbps or 1Tbps or what? What capacity links should we establish between PoPs – how much traffic do you expect to go from Seattle to Miami? These are all planning problems that require business input. Real ISPs do market studies, run optimization algorithms, and make strategic guesses to figure these things out. In this lab I'll make things easier by giving you a good topology to build, but please go through the thought exercise of putting yourself in the shoes of a real network engineer.
Please do the following:

(1)Please take a few minutes to think about the issues below. Sit back and close your eyes. Pretend you really do want to start your own ISP. How would you design the topology? How specifically would you go about answering the questions above? Do a few google searches - can you uncover any information to help you take some first steps? Suppose you are meeting with your investors tomorrow and you need to tell them what moves you are making, you really have to tell them a plan. What topology would you make if you had to do it right now? Write a paragraph or so with your thoughts.

(2) To make things easier I'll go ahead and give you a good topology to build. Please look at the figure below.

You can see that each cloud represents an ISP. Each of the three ISPs are represented as clouds, and each has a border router through which we can communicate to other ISPs. AS-8000 is another ISP which is connected to three other ISPs through different border routers. These border routers can also communicate with each other to connect all the ISP.

(3) Answer the following questions (in your lab writeup)

(a) What do you think about the topology, overall? Do you have any ideas to make it better?

(b) What do you think about the resilience of this network? If a router fails, or a link fails, are your customer well-protected?

(c) There are no firewalls here. Only routers. Is that ok? Is that safe?

(d)What would you do if there was a huge increase in traffic between site1 and site2? What bad things might happen? What are some of your options to mitigate the situation?

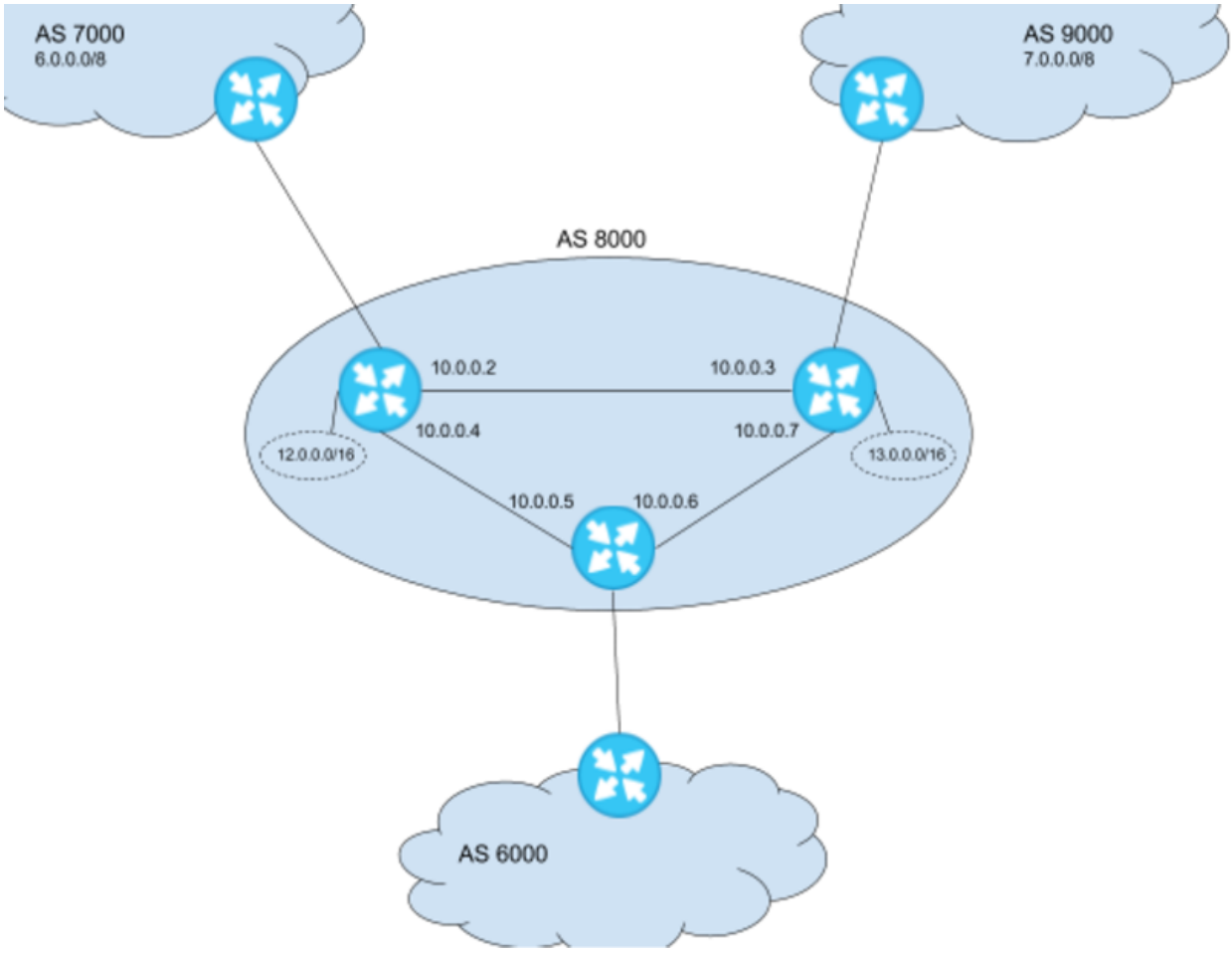

Figure 2: The topology

4.2 Lay physical backhaul

If this were a real, physical, ISP, we would now start building it. First you'd need to purchase your physical routers and ship them out to each location. You might purchase rack space in colocation facilities in each city and pay someone to mount them for you there, so you don't have to fly out to each place. Next, you'll need to get a shovel and start digging, so you can dig the 1000 mile trenches to lay fiber optics between each of your PoP cities - just kidding. If ISPs had to dig trenches and negotiate with every farm owner between every time they wanted to connect a pair of their PoPs they'd go out of business. What you would actually do to connect your PoPs is you'd purchase layer 2 or layer 1 connectivity from another business who has already done the hard work of laying fiber (e.g., AT&T). So you'd rent long-distance dedicated lines between each of your PoPs. Again for this step, I don't want you guys wasting large amounts of time on the phone and large amounts of money leasing backhaul, that doesn't really teach you anything. But I think it's good to go through the thought exercise a bit so let's do that.
Please do the following:
(1) Pretend you want to connect your PoP in Dallas, TX with your PoP in St Louis, MO. Go on google and figure out what your options are. Feel free to call sales departments at any businesses that can help. Write down in your lab report what you figure out. What business can you purchase from, what plans do they have, what info do they need from you to set it up? Write a paragraph or so on what you figure out.
(2) Now, let's actually set up our virtual topology. GNS3 makes things easier since we can just drag to create links. Please go ahead and create the routers and links as shown in the figure above.
(3) Before we start the routers, we need to tell them information about the topology (e.g. the IP addresses of other routers it will peer with) and policy (e.g. which IP prefixes it should originate and export). This is done by editing a "configuration file". The configuration file is written in a special programming language that is used for programming networks. These languages differ across vendors but there are many similarities, so if you learn one you should easily be able to pick up others. Since we are using Cisco routers, You'll be working with Cisco IOS (Cisco's operating system).
(4) Use the configuration file to give each router a hostname. Ie, to make your routers easy to identify (hey, what router am I logged into?) in log messages etc, give each a hostname. It is good to use a format that encodes useful information about the router such as its geographic region and type. It is also good to keep the name short so it is easy to read in logs and shows up concisely in graphical tools (e.g., see this approach). For example you could use the form [3 char city name]-[router type]-[router function]-[router number]. For example, if there are three access routers in Chicago, and the first one is a Cisco C3750X, you could name it chi-c3750x-access-0. If you were naming your routers how would you name them? Come up with a good naming scheme and name them appropriately.
(5) Save your configuration to disk. Input do wr when running in console then save all project by choose File¿¿save project as.... Make sure to wr to save the configurations, otherwise all the settings would be wiped up from the routers next time you work on the project.

4.3 Configure interfaces

Next, we have to tell the routers who their neighbors are. We established the physical connections, but now routers have to know who is the other guy (router) on the other side of the link, so they know who they're talking to. Otherwise Chicago doesn't know if it's forwarding packets to NYC or San Diego.

You'd think routers could just automatically figure this out, I mean it's 2017 for crying out loud. But forming the logical connections between routers is often a manual process, and one reason for that is security. If someone comes along and plugs a router into your router, you don't want to be just bringing up a peering relationship with it. By forcing the operator to manually set up connections between routers (or by having a script that they maintain do it), we arguably get some additional safety. At the network edge that's a bit less critical and things change more often, that's why it's a bit safer the unmanaged ethernet switch or DSL gateway you have at home just plugs-and-plays with

whatever you plug in.

All that said, there are protocols that automatically discover the topology. One is called LLDP (link-layer discovery protocol). Another is called CDP (Cisco Discovery Protocol - this is Cisco's proprietary version of LLDP). You'll hear recommendations not to run these protocols for security reasons. And there are dangers with these protocols. That said, not knowing what your network looks like seems like a pretty darn big security risk to me too. So I personally think it's good to run LLDP on your network internals and just be really careful to disable it on public- and host-facing interfaces. But people have different opinions on this (what's yours?)

Please do the following:
(1) Configure all interfaces. On each pair of physical routers, configure their interfaces. You need to assign ip addresses and subnets to all the interfaces before using them. In order for these routers to communicate both routers need to be connected through one of their interfaces which would have a specific ip address and a subnet mask.
(2) Verify you have this working by running show commands (eg "show interface") - inspect show command output and verify it is correct. Copy show command output into your lab report. Also, log into each router and verify you can ping its directly-connected neighbors.
(3) Configure LLDP on all interfaces. Again verify you have it set up correctly by checking that you can see the router on the other side of the link (you should be able to see the hostname of the router on the other side of the link). Again copy relevant show command output into your lab report. Hint: You can also check the default LLDP settings of a router image. You might need to modify the LLDP default settings according to your need.

```
!This example shows how to configure LLDP characteristics.
Switch# configure terminal
Switch(config)# lldp holdtime 120
Switch(config)# lldp reinit 2
Switch(config)# lldp timer 30
Switch(config)# end
```

To help you out, we give some sample configuration code below, which shows how to configure an interface:

```
!!
hostname zebrad
password zebra
!
interface eth0
ip address 192.168.31.0/24
!
interface eth1
ip address 192.168.35.0/24
!
interface lo
ip forwarding
!
log stdout
line vty
!
```

In this example, you can see the two interfaces f0/0 and f0/1 being configured. Comparing this configuration to figure 1, you can verify that f0/1 is connected to the rightmost router in AS 8000 and f0/0 is connected to the router below it.

Try this out - configure one of your interfaces. Then configure the interface on the other side of the link. As shown in the figure, each link has a small subnet assigned to it from which IP addresses are assigned (this is a common practice). Assign IP addresses to either side of the link from this shared subnet.

Then, let's test things out. Log into one of the routers and verify you can ping the interface of the adjacent router.

You can use a similar process to configure and test your other interfaces. Please test each interface as you bring it up, so there are no big surprises later.

4.4 Set up OSPF

As mentioned earlier, networks run distributed algorithms called "routing protocols" to discover reachability. These protocols enable routers to figure out where to route packets. If Seattle has a packet destined for New York, but they're not directly connected, how does it know where to send it? Does it send it through Denver? But what if Denver thinks that Seattle is the next hop to New York? Routing protocols sort all this out - they create a globally correct set of "routes" (paths that packets can follow), and store these routes as state (entries in a routing table that compactly describe the next hop to forward packets of various types).

However, things are a bit more tricky than that in practice. In the Internet, there are typically multiple routing protocols that run at the same time. See, here's the deal - within my own local ISP, I am in control of things. I know I'm not going to mess things up. So I want all my routers to just share information. But ISPs need to route to each other as well. So I need to share routes with my neighboring ISPs. But I don't trust them as much. Business issues are at play (ISPs sometimes try to trick others into transiting their traffic for them), as well as security issues (sometimes little ISPs run by inexperienced operators accidentally propagate bogus routes and mess up global routing tables). So to

deal with this ISPs maintain isolation by running a different routing protocol internally. To neighbors, ISPs run the Border Gateway Protocol (BGP) - this protocol distributes routes from the local ISP to other ISPs. It is run on peering sessions between my border routers and by neighboring ISP's border routers. But internally, ISPs run an IGP (interior gateway protocol). There are different IGPs - some ISPs prefer OSPF, others prefer one called IS-IS. There is also one called RIP that is not used as much anymore. ISPs write configurations to "redistribute" routes learned from IGP into BGP and vice versa. For example if one of my internal routers has a particular subnet 169.23.33.0/24, I may distribute that internally through my IGP, and then my border routers receive that route advertisement and redistribute to other ISPs using BGP. That way both my local routers and routers in other ISPs can route to my 169.23.33.0/24 prefix.

To simplify this step, we will give you configuration files for the routers. However, since the IP addresses and interfaces in your setup may differ, you will likely have to edit these files to take that into account.

```
! -*- ospf -*-
! OSPF Config file for router A (ospfA.conf)
hostname ospfd
password zebra
!
interface f0/1
interface f0/0
!
router ospf 1
Router-id 12.0.0.0
    network 10.0.0.0  0.0.0.1 area 0
    network 10.0.0.0 0.0.0.1 area 1
```

Explanation: First, we need to tell IOS which interfaces on the local router may be used by the routing protocol. This is done with the "interface" command above. Here, we tell the router that interface with IP address 10.0.0.1 should be called f0/1 and is made visible to routing protocols running at this router. OSPF will then automatically discover routers on the other side of the link. Next, the "router" command tells IOS to create a new OSPF router instance. OSPF creates logical groups of routers called "areas" to improve scalability. Routing updates are constrained within OSPF areas. Here, we only create a single network-wide area called "area 0" (the top-level area in OSPF is always numbered zero). We tell Quagga this area contains prefixes 10.0.0.1/31 and 10.0.0.3/31, by saying that it contains the network with super-prefix 10.0.0.0/7. Finally, we tell Quagga that routes to 11.0.0.0/8 can be reached at the local router within area 1 with the command "network 11.0.0.0/8 area 1".

After doing this, router A can discover router B's presence, forward data packets to hosts connected to B, and vice versa. Verify this by pinging eg: ping 10.0.0.2.

4.5 Set up iBGP

After completing the previous step, router A can discover router B's presence, forward data packets to hosts connected to B, and vice versa. In this step, we will also allow A and B to exchange externally-learned routes. In particular, router A and B will function as border routers connected to other ISP networks. Routes received from other networks need to be propagated internally, to ensure all internal routers know how to reach externally-learned prefixes. For

example, if router A learns that it can reach ”12.0.0.0/8” from router C, it needs to inform B of that fact, so B knows it can reach ”12.0.0.0/8” by routing through A.

Routes between ISPs are propagated using the Border Gateway Protocol (BGP). BGP is a “path-vector” protocol – it propagates advertisements containing a list of hops to a particular destination. BGP operates on prefixes – each advertisement contains a list of prefixes, and the AS-level path used to reach those prefixes. The AS-path is used to avoid routing loops and ”count-to-infinity” problems, by having each router check to see if its own AS number already appears in the path, and if so, dropping the advertisement. The AS-path may also be used in routing policies, e.g., ASes may add multiple copies of their AS number in routing advertisements to increase the path length on certain routes, thereby shifting traffic to alternate paths.

Here, we will configure a BGP session between routers A and B. When BGP is run internally within an AS, it is referred to as ”iBGP” (internal-BGP). Again, we will give you the configuration files to use:

Configuration file for router A:

```
! -*- bgp -*-
! Config file for router A (bgpdA.conf)
!
hostname bgpd
password zebra
!
router bgp 8000
neighbor 13.1.1.1 remote-as 8000
neighbor 13.1.1.1 update-source loopback 0
exit
```

Explanation: First, we need to tell IOS to create a new bgp router instance. This is done with the ”router bgp 8000” command. The ”8000” at the end tells the IOS bgp daemon the name of the local AS number, so the router can append its AS number to advertisements, etc. Next, we tell Quagga the IP address that identifies this BGP router with the ”bgp router-id” command. Finally, we tell IOS to create an iBGP session to router B, which has loopback address 10.0.0.2. Since B is within the same AS as A, we want to tell BGP to forward the traffic via OSPF, rather than a BGP route. We do this by specifying the command ”update-source lo0”, which tells the router to contact 10.0.0.2via the loopback address.

Similarly, here’s the configuration file for router B:

```
! -*- bgp -*-
! Config file for router B (bgpdB.conf)
!
hostname bgpd
password zebra
!
router bgp 8000
neighbor 12.1.1.1 remote-as 8000
neighbor 12.1.1.1 update-source loopback 0
exit
```

(a) Type *do wr* to save this change

(b) Now, verify they are correctly peering. Console supports to check the current state of the router and its connections. You can access by typing *"show ip bgp"*. The output should look similar to this on router A:

```
   Network          Next Hop            Metric LocPrf Weight Path
*  10.0.0.0/31    13.123.137.124         0      7018  15169   i
*> 10.0.0.0/31    13.123.1.236           0      7018  15169   i
```

and similar to this on router B:

```
   Network          Next Hop            Metric LocPrf Weight Path
*  10.0.0.0/31     12.123.9.241          0     4323  56203   i
*> 10.0.0.0/31     12.123.1.236          0     4323  56203   i
```

4.6 Configure eBGP

Now, extend the configuration to the topology shown in Figure 2. To do this, you'll need to start up 3 more instances of IOS, each running on a different machine, to create a network composed of 5 software routers total. Have routers in AS 8000 advertise a static route to 10.0.0.0/7 (the aggregation of 10.0.0.0/8 and 11.0.0.0/8). Print your configuration files as well as the terminal output of "show ip bgp", and attach them to your writeup.

Hint: configuring external BGP sessions is a little different from iBGP sessions. You may want to do something along these lines, which combines the iBGP file from above with commands to create the eBGP sessions:

```
! -*- bgp -*-
! Config file for router A (bgpdA.conf)
!
hostname bgpd
password zebra
!
router bgp 8000

 neighbor 13.1.1.1 remote-as 8000
 neighbor 13.1.1.1 update-source loopback 0
 neighbor 13.1.1.1 ebgp-multihop 2
 neighbor 13.1.1.1 next-hop-self

 neighbor 6.1.1.1 remote-as 7000
 Neighbor 6.1.1.1 update-source loopback 0
 neighbor 6.1.1.1 ebgp-multihop 2
 neighbor 6.1.1.1 next-hop-self
!
```

Explanation: The next-hop-self command ensures that router A's IP address is advertised as the next-hop used to reach routes it advertises. The ebgp-multihop command allows connections to peers that are not directly connected. While this isn't strictly necessary in this example, it may be used in cases where the border router's peer is multiple router-level hops away.

4.7 Policy configuration

Next, we will configure policies to limit route export. We will consider routers C and E to be *providers*, and router D to be a *customer* of the ISP containing A and B, as shown in

(a) If router C advertises a route, which routers should receive that route? What about if router E advertises a route? What about if router D advertises a route?

(b) Modify your configuration (hint: you may want to do this by tagging route advertisements with a community attribute indicating what kind of peer they are received from, and then also add an export filtering rule for providers that filters routes from customers. An example of this is shown below.) Print your configuration files as well as the terminal output of "show ip bgp", and attach them to your writeup.

You can tag route advertisements with community attributes as follows:

```
! Config file for router A
ip community-list 13 permit 8000:13
route-map IMPORT-CUST permit 10
match community 13
set community 8000:13
router bgp 8000
neighbor 10.0.0.2 remote-as 8000
neighbor 10.0.0.2 send-community
neighbor 10.0.0.2 route-map IMPORT-CUST out
```

Explanation: A "route-map" is a construct in router configuration languages used to express policies that should be applied to groups of routes. A route map consists of a name (*IMPORT-CUST*, a set of rules denoting which a match to this rule should be triggered (with the "match" command, here we leave out the match command to match all routes), and the commands to be applied to updates triggering a match (*set community 0:1000*, which adds the community attribute "0:1000" to the routing update). The "neighbor" command at the end applies the route-map to all updates received from *customer-neighbor-IP-address*. The "in" statement at the end of that line means the rule is applied to inbound updates, as opposed to updates this router advertises to customer-neighbor-IP-address.

You can prevent route advertisements with a given community attribute from being advertised to a peer (export filtering) by as follows:

```
! Config file for router A
route-map EXPORT-CUST permit 10
set community 8000:12
Router bgp 8000
Neighbor 10.0.0.2 remote-as 8000
Neighbor 10.0.0.2 send-community
Neighbor 10.0.0.2 route-map EXPORT-CUST out
```

Note: Understand these examples and apply them to your case.

5 What to report

Results and outputs: The instructions above mentioned some things you should include in your writeup - please do this.

Configuration files: In this lab you created some configuration files. Please save your configuration files for each router. For example: router_A.conf.

After completing this cycle, you have new knowledge and experience with building and managing networks. Please demonstrate your knowledge by thinking about, then writing answers to the following questions in your report. Grading will be done based on the clarity and completeness of your responses, as well as the simplicity and appropriateness of your solution.

- Suppose you're working for a large enterprise network. Your upstream provider is Sprint. You maintain a BGP peering with Sprint at two separate PoPs. You wake up on Friday and your users can't send traffic to the Internet. How would you debug the issue? Please give details.
- You fix the above issue but then late the following week you get hit with a major DoS attack. The attack targets your web infrastructure and is overloading your servers, but not any part of your network infrastructure. The traffic is composed of SYN floods sourced from IP address range 175.45.176.0/24. How do you protect your servers from this attack?
- You fix the above problem but a week later a new problem comes up - you have too much traffic going over one of your links. Describe in detail how you would get traffic to shift to the other PoP (both inbound and outbound).