Network lab – 3A Ensimag & M2 MOSIG
Inter-domain routing
Border Gateway Protocol (BGP)

Martin Heusse Baptiste Jonglez

December 15, 2022

1 Introduction: inter-domain routing

BGP (Border Gateway Protocol) is a routing procol that allows to exchange routing information between *Automonous Systems* (AS). An AS is an administrative entity that owns and manages a network or "domain", and as of January 2020 there are 67,000 such entities wordwide. For instance:

- *Internet Service Providers:* Orange, Verizon...
- *Hosting and content providers:* OVH, Amazon, Google, Netflix...
- *Research & Education Networks:* Renater, Geant, SURFnet, Janet...
- *Governments and big companies:* IBM, Ford...
- *Transit providers:* Cogent, Level3...
- *Smaller actors and non-profits:* Mozilla, Wikimedia, PCH.net, Gitoyen, Grenode...

More information can be found at `https://www.peeringdb.com/` and there are statistics at `https://bgp.potaroo.net/as2.0/bgp-active.html`.

In inter-domain routing, **border routers** handle communication with neighbouring Autonomous Systems, as shown in Figure 1, thanks to BGP.

The goal of BGP is to propagate routes across the network, just like intra-domain routing protocols such as OSPF, but policy considerations are primordial when dealing with inter-domain routing. For instance, on the network in Figure 1:

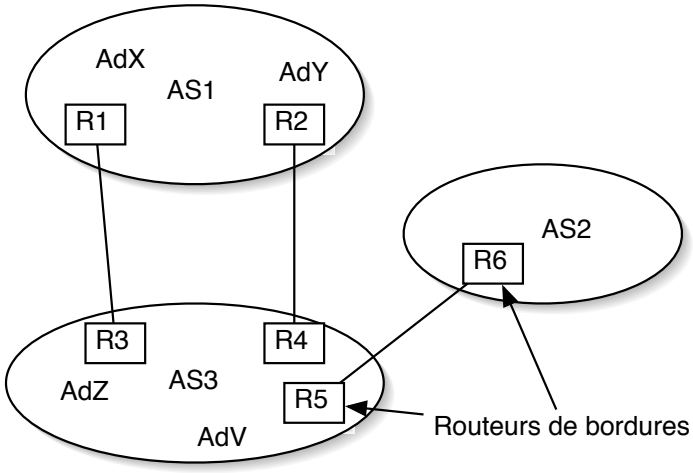

Figure 1: Inter-domain routing

- **AS3** can decide that all traffic destinated to network **AdV** must come through router **R4**.
- **AS3** can decide *not* to provide connectivity from network **AdX** to network **AdY**.
- **AS3** can decide that all traffic from **AS1** to **AS2** must go through router **R4**.
- **AS3** can decide that all traffic from **AS2** to **AS1** must go through router **R3**.

Important: A route towards an internal prefix will be propagated, only if

- it is published (by means of a `network` command);
- and it appears in the routing table of the ASBR that publishes it.

This last condition avoids that a router receive traffic for which there is no entry in it routing table. Also, an ASBR will propagate a route only if it knows how to reach the BGP `NEXT_HOP` for this route. (The subnet of the BGP next hop may not be local to the ASBR!!)

2 Network lab

2.1 Network setup

Your lab network has three Autonomous Systems: AS10, AS12, AS14. You are supposed to be a network engineer in the organization managing **AS12**. Your organization has a contractual relationship with a provider AS14 that gives you access to all other ASes in the world (that is, the Internet!). In contrast, AS10 is one of your clients, and it relies on you to be connected to the rest of the Internet. It is a "stub AS" in the sense that you are its unique provider AS.

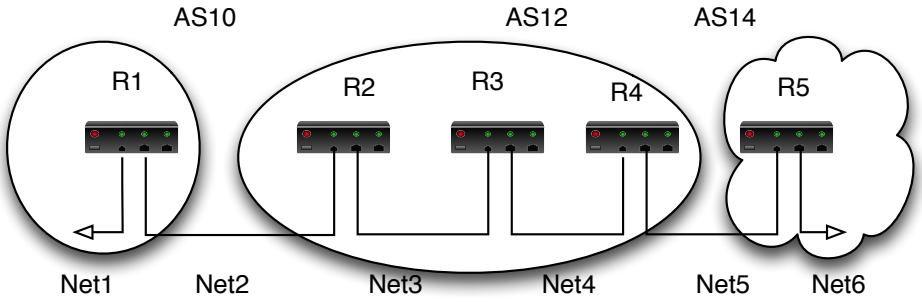

Figure 2: Network setup for this lab. AS numbers are indicative, make sure to choose unique AS numbers for your group.

2.2 Addressing plan

Choose unique addresses and AS numbers! This will allow you to interconnect with other groups at the end of the lab.

Write down an addressing plan with an IP range for each network and IP address for each interface. Configure the five routers.

2.3 Internal routing in AS12

You will use OSPF to interconnect the routers within your AS.

- Setup OSPF in all routers of AS12:
  - `router ospf`
  - `network` *network_address_of_interface* for each interface on which you want to run OSPF.

- Check that the routing tables of R2, R3 and R4 are correct.
- Check that OSPF is indeed disabled on external links that connect you with other ASes: you don't want internal information to leak outside of your network, eBGP will be used instead.

2.4 Inter-domain routing: eBGP

2.4.1 How to debug a BGP network

`show ip bgp`: display the BGP routing table
`show ip bgp neighbors`: display the state of sessions with BGP neighbours
`show ip route`: display the IP forwarding table

2.4.2 Exchange routes with another AS

You will now setup BGP between R1 and R2. The goal is that AS10 learns about the addresses used by AS12, and that AS12 learns about the addresses used by AS10.

- **Start a packet capture on network `net2`.**
- Start BGP on R1 and R2:
  `router bgp` *your_AS_number*
- Establish a BGP session between R1 and R2:
  `router BGP` then
  `neighbor` *neighbour_address* `remote-as` *neighbour_AS_number*
- Check that the BGP session is working and look for BGP packets in the capture.
- In AS10, "announce" network `net1` to the outside world:
  `router` then
  `network` *network_to_announce*

  *Note: the `network` command is slightly different here compared to OSPF or RIP. Here, it means that the network will be published to BGP neighbours **provided it is present in the routing table**. Indeed, it does not make sense to announce a route for a network we don't know how to reach: if other ASes send us traffic destinated to this network, we wouldn't be able to forward them to the destination.*
- **Consult and interpret the BGP routing table of R1 and R2.**
- Analyze the BGP packet sent by R1 to R2 when you announced `net1`. In this packet, find the route and its attributes.
- **In AS12, do R3 and R4 know about `net1`? Why?**

2.4.3 Redistributing external routes into AS12

It is possible to redistribute routes learned from BGP into OSPF. This way, all internal routers will know how to reach these addresses.

`router ospf` then:
`redistribute bgp` *your_AS_number* `subnets`

Check that R3 and R4 now know about `net1`.

Look at the routing table of R3 and R4 and explain which interesting information you find in it.

Important: on R2 and R4, under `router bgp`, add `no synchronization` so that redistribution can work as expected.

2.4.4 Announcing routes from AS12 to AS10

There is another way to announce routes to external ASes: redistribute all routes known from OSPF into BGP. However, it is **not recommended**: BGP cannot be sure that these routes really come from the local AS. They could have originated from another AS, but this information is lost if they have been redistributed into OSPF. Furthermore, this method means that any change to the internal routing state will be reflected to the outside world, which is unnecessary and generally frown upon to ensure stability.

In the following, you will simply announce the networks from AS12 to AS10 manually, exactly as before:

- Start a packet capture on net2.
- On R2, announce net3 and net4.
- Analyze the BGP packets exchanged between R1 and R2.

Check that R1 now knows about net3 and net4. Write down the BGP routing table of R1.

2.5 Intra-AS route exchange: iBGP

We would now like to distribute the routes known in AS12 to AS14. Instead of doing the same configuration as before on R4, there is an easier way: we can configure R2 to give all its routing information to R4, so that R4 can then distribute them to AS14.

This is done through a BGP session between R2 and R4 (*iBGP* for "internal BGP"):

- Start a BGP session between R2 and R4 (`neighbor`)
  Check that R4 has learnt the routes from R2 through this iBGP session.
- Start an eBGP session between R4 and R5.

Write down and analyze the BGP routing table and the forwarding table of R4 and R5. Why is `net1` missing in the routing table of R5?

This may seem strange, because `net1` appears in the routing table of R4. The key to understand this behavior is the "next hop" of this route, *i.e.* the neighbouring router through which traffic will be forwarded. In this case, the next-hop is R1's address on `net2`, but R4 does not know how to reach `net2`! Thus, R4 decides *not* to inform R5 about this route because it is not itself capable of forwarding traffic to this destination.

The solution is to tell R4 how it can reach `net2`. To do this, R2 can announce `net2` in OSPF, and R4 will eventually learn about it. But we don't want to talk OSPF with an external AS! This interface should be "passive" for OSPF, which means that no HELLO packet will be sent, and no incoming OSPF packet will be accepted.

Another solution would be to announce `net2` using BGP, either from R1 or from R2.

Note: the `NEXT_HOP` attribute, attached to a route, is only modified when the route crosses an AS boundary. When R2 passes a route to R4, it keeps the `NEXT_HOP` unchanged. The idea is that it is the job of the IGP (such as OSPF) to determine how to best reach this next_hop.

Try one of these solutions and check that the route to `net1` is now known from R5.
What is the path seen by R5 to reach `net1`?
Try to ping R1 from R4 and from R5. If the second one fails, can you see why?
Fix the problem if it happens.

2.6 Adding a default route in AS12

At this point, the interaction between BGP and OSPF in AS12 is far from satisfactory: all external routes that we learn from AS14 (that is, all routes of the Internet) are redistributed in OSPF, which is certainly not designed for so many routes!

One solution would to use BGP on R3, which, in this case, would completely replace the IGP. This would be the simplest solution in this network.

We will use another idea: in AS12, most routes go out through R4 and then to AS14. Only a few routes need to go through R2, corresponding to the IP addresses announced by AS10. Thus, all internals routers in AS12 could use a **default route** that goes through R4, and just know about the routes of AS10 (in addition to the routes that are internal to AS12, of course).

To add this default route on all routers, it is possible to announce it in OSPF from R4:

```
router ospf
default-information originate always
```

However, this route will be announced by R4 even if its connection to the outside world disappears.

A more satisfactory solution, using OSPF features, would be to use a **stub area**: the whole network would be in area 1, while R4's interface connected to `net5` would form a (very small) area 0. This way, a default route would be created by R4 and announced in area 1.

Why would this solution fail in your current network?

To make it work, you actually need to make area 1 a **Not So Stubby Area**, in which external LSA can propagate. On R4:
`area 1 nssa default-information-originate`. (You need to remove the `redistribute bgp` from R4 to stop advertising the external routes.)

Look at LSA exchanged in the NSSA area: they are now of Type 7, *i.e.* they are limited to the area they come from (regular external LSA have Type 5: they propagate everywhere **except** in stub areas).

2.7 Using *loopback* interfaces

For BGP communication *within an AS*, we usually use the *loopback* or *discard* interface (`lo`). The goal is to avoid depending on the state of internal links: if a link is broken, the IGP will find a new route between border routers, and the TCP connection associated to the BGP session can continue working.

Add an address to the *loopback* interface of your border routers in AS12.

Change the iBGP session to use these new addresses (don't forget to remove the old sessions)

You also need to specify the source address used for the TCP connection:
`neighbor neighbour_address update-source lo0`

Setup the topology shown in Figure 3 and give a scenario that shows why using the loopback interface is useful.

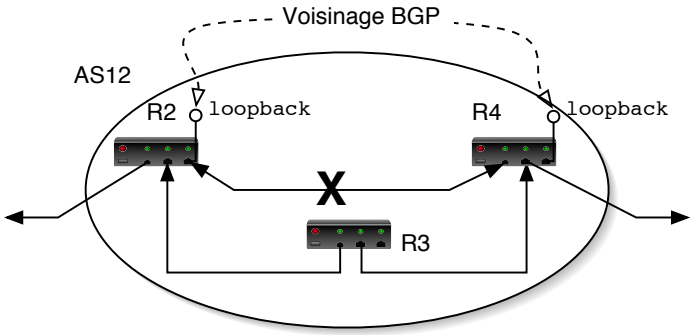

Figure 3: *loopback* interface usage

Why don't we use this method for BGP sessions between two different ASes? Look to the TTL field of the BGP packets for a possible explanation.

2.8 Filtering BGP routes

BGP is an example of **policy routing**, and filtering the routes that we send or receive is a way to implement such policies.

There are three filtering methods in BGP:

- By **route**: we can forbid propagation of routes for specific addresses
- By **path**: we can forbid propagation of all routes with a given AS Path
- By **community**: any router can "tag" routes with arbitrary numbers called "communities" before propagating them, and other routers can then filter routes based on these communities. They typically encode geographical information about where the route was learnt, or indicative policies ("Please don't propagate this route to AS X").

2.8.1 Filtering by route

This uses ACL (Access Control List) to decide which route should be sent to a neighbour (`out`) or accepted from this neighbour (`in`).

`neighbor` *neighbour_address* `distribute-list` *ACL_ID* `in/out`

The ACL are defined in this way:
`acces-list` *ACL_ID* `permit/deny` ip *dst_address reversed_mask src_address reversed_mask*

ACL_ID is the ACL identifier (use an identifier of at least 101, so that you use an extended access list, which can take any mask for any prefix), *net_address* is the network address to allow or deny, *reversed_mask* works like a netmask but it encodes *non*-significant bits.

In a BGP context the source and destination of the ACL take a different meaning:
`acces-list 101 deny ip 160.10.0.0 0.0.0.0 255.255.0.0 0.0.0.0`
This defines an ACL that denies 160.10.0.0/16 (and only a prefix of this exact length). You need to add a rule that allows all other networks:
`access-list 101 permit ip any any`: `any` is equivalent to 0.0.0.0 255.255.255.255

Configure your network so that AS12 does not propagate a route for `net6` to AS10. Check that the filtering works using ping and looking at the various routing tables. Then tell R1 to refuse to learn about `net4` from R2.

If your configuration is not taken into account immediately, you can use the command `clear ip bgp *`. You can see the content of access lists with `show access-list` *ACL_ID*.

2.8.2 Filtering on AS paths

`ip as-path access-list` *ID* `deny/permit` *regular_expression*

The regular expression is used to specify which AS or AS path should be matched. It can contain AS numbers and special characters:
`^` : start of path
`$`: end of path
`.` : any character
`*` : any number of times

You can use `show ip bgp regexp` *expression* to check the validity of your regular expressions.

Examples:
```
ip as-path access-list 101 deny ^200$
ip as-path access-list 101 permit .*
```

`^200$` matches routes coming directly from AS200
`.*` matches routes with any AS Path
`^200 300$` matches routes received from AS200 that received them from AS300 (originator of the routes)
`^200.*` matches any routes received from AS200

Delete previous route fiters: `no neighbor` *neighbour_address* `distribute-list`

Enable path filtering:
`neighbor` *neighbour_address* `filter-list` *ID* `in/out`

Use this method on R1 to refuse routes originated from AS12 but allow routes originated from AS14.

2.8.3 Filtering by community

An AS X can decide to share a given set of routes to AS Y, while specifying to **not** share them to AS Z. (The community appears in the update message)

2.9 Choice between multiple route

In case multiple routes are available for a given destination, many parameters and features have an effect on the choice of route to insert into the routing table. The choice may be local to a router, local to the AS of passed from AS to AS. Several attributes attached to the route updates allow the routers and ASes to make their decisions.

The precedence between the route attributes is the following:

1. Weight (a router may associate weights to its neighbor and prefer the higher values)
2. Local preference (LocalPref) (local to an AS)
3. AS path length (in number of ASes)
4. Origin: Internal protocol should be preferred to external
5. BGP metric
6. IGP metric to the BGP next hop

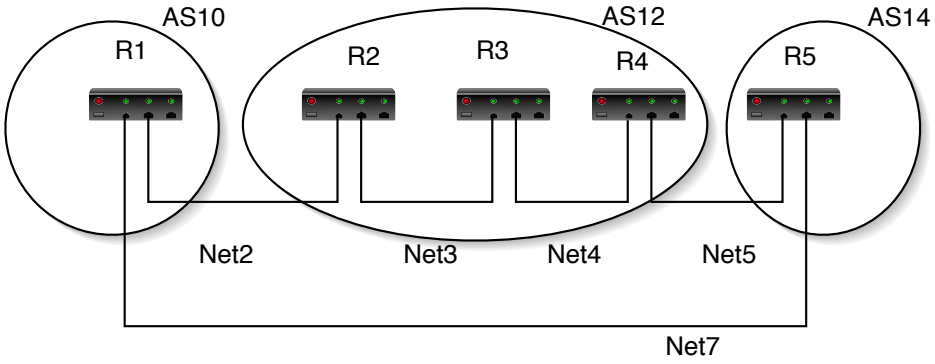

Figure 4: Adding net7 to the network

Remove the lists and filters from above. Interconnect R1 and R5 with net7 of network address 7.0.0.0/8. Advertise net7 with a `network` command in R1 **and** R5. (We want to have two concurrent paths to the same prefix.)
Observe the routing tables and BGP routes in R2 , R3 e t R4.
How to they each reach net7 ? Why ?

We are going to find ways to decide how the routers of AS12 reach net7.

But before we do this, switch off net2. What happens? How long does it take for R2 to reach net7 again?

1. Local preference
   This information is carried by the LOCAL_PREF attribute of the BGP messages, its code is 5. The default value is 100. The router advertising the highest LOCAL_PREF for a given destination will be privileged. This allows to choose which neighbor AS to use for a given destination.

   The command `bgp default local-preference` *value* may be used to set this attribute for all routes advertised by a given router. (Other commands allow to set the LOCAL_PREF on a per-route basis.)

Can you set this attribute so that R2, R3 and R4 all use R2 to send packets to net7.

Yo may want to use `clear bgp` to speed up the propagation of this information throughout AS12.

2. Information shared between AS
   An AS may have an influence on the choices made in another AS by changing the metric attribute in the updates. The METRIC attribute carries this information between neigh-

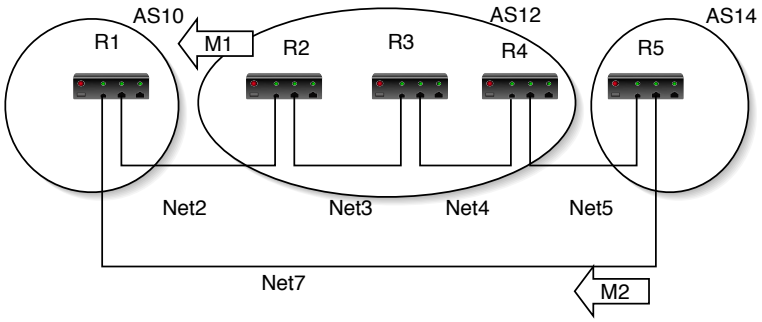

Figure 5: BGP metric

boring ASes. Is is called MED (Multi_Exit_Discriminator) in BGP version 4 (and INTER_AS dans BGP 3) (code 3). This attribute is not transitive: it is not carried from AS to AS, but only used between neighboring ASes. If a router gets several (otherwise equivalent in terms of weight, local pref, AS path length, origin) routes for a given destination with differing MED, it will use the one with the smaller metric. These commands change the metric for all routes advertised by a router:
`neighbor` *Ad_voisin* `route-map` *ma_route-map* `out`
`route-map` *ma_route-map* `permit 10`
`set metric` *M1*

You need to type :
`bgp always-compare-med` to make the routers compare MED received from different AS. (Normally compared only between routes received by the same neighbor AS.)

Advertise different metrics to R1 from R2 and R5 (larger from R2). Check that this information reaches R1 (regarding net3 for instance), and that it is the route with the lower MED that is used in the end (labeled best in show ip bgp).

BGP in real:

On the RIPE site (Réseaux IP Européens) (address-: `http://www.ripe.net/perl/whois`) you can find a data base (whois) which specifies which AS advertises which routes to which neighbors. You may want to query this database with the AS number AS2200 of RENATER.

We used AS numbers 10,12 et 14 in this lab, but the private AS numbers are in the range 64512 to 65534.