DE-CIX



# BGP for networks who peer

Wolfgang Tremmel, DE-CIX Academy, July 2026

Draft

# Contents

**1 Introduction** **7**
1.1 What is BGP? . . . 7
1.2 Elements of routing . . . 7
1.2.1 IP prefixes . . . 7
1.2.2 The Autonomous System . . . 9
1.2.3 Autonomous System Numbers . . . 10
1.2.4 The AS path . . . 11
1.3 Some theory about routing protocols . . . 12
1.3.1 Classification of routing protocols . . . 12
1.3.2 Static vs. dynamic routing . . . 12
1.3.3 Interior protocols vs. exterior protocols . . . 12
1.3.4 Link state vs. distance vector protocols . . . 13
1.4 How a router works . . . 13
1.5 Examples in this book . . . 14
1.6 BGP and IPv6 . . . 14
1.7 Recommendations . . . 14

**2 Setting up an Interior Gateway Protocol (IGP)** **15**
2.1 Motivation . . . 15
2.2 Configuring OSPF . . . 15
2.3 Configuring IS-IS . . . 16
2.4 Configuring OSPFv3 . . . 18

**3 iBGP - BGP within one Autonomous System** **21**
3.1 What is iBGP? . . . 21
3.2 Configuring iBGP . . . 21
3.2.1 About the Loopback interface . . . 22
3.2.2 Lifecycle of a BGP connection . . . 23
3.2.3 Fully Meshed vs. Route Reflector . . . 25
3.2.4 Keeping the configuration short . . . 27
3.2.5 Cisco example . . . 27
3.2.6 Mikrotik example . . . 28

**4 eBGP - Connecting to the outside world** **31**
4.1 What is eBGP? . . . 31
4.1.1 Differences between iBGP and eBGP . . . 31
4.1.2 Best practices . . . 32
4.1.3 Sources and destinations of prefixes . . . 32

Draft

4.1.4 Cisco example . . . 33
4.1.5 Mikrotik example . . . 34
4.1.6 FRRouting example . . . 34

**5 Becoming multi-homed 37**
5.1 What is being multi-homed (in terms of BGP)? . . . 37
5.2 What changes when you become multi-homed? . . . 37
5.3 Configuring multi-homing . . . 38
5.3.1 Network setup . . . 38
5.3.2 Receiving prefixes . . . 38
5.3.3 Sending prefixes . . . 38
5.4 Configuration examples . . . 38
5.4.1 Cisco example . . . 38
5.4.2 Mikrotik example . . . 39
5.4.3 FRRouting example . . . 40

**6 BGP best path selection 41**
6.1 Motivation . . . 41
6.2 Best path selection in contrast to routing . . . 41
6.3 Best path selection algorithm . . . 42
6.3.1 Before we start... . . . 42
6.3.2 Local Preference . . . 42
6.3.3 AS path length . . . 42
6.3.4 Origin type . . . 43
6.3.5 Multi Exit Discriminator (MED) . . . 43
6.3.6 eBGP vs. iBGP . . . 44
6.3.7 Network exit . . . 44
6.3.8 Age of prefix announcement . . . 44
6.3.9 Tie breakers: Router ID and neighbor IP . . . 45
6.4 Advanced topics . . . 45
6.4.1 Multipath . . . 45
6.4.2 Tweaking the algorithm . . . 45

**7 BGP Communities 47**
7.1 Introduction . . . 47
7.2 Original communities . . . 47
7.3 Extended communities . . . 48
7.4 Large communities . . . 48
7.5 Some well-known communities . . . 48
7.5.1 NO-EXPORT . . . 49
7.5.2 NO-ADVERTISE . . . 49
7.5.3 GRACEFUL-SHUTDOWN . . . 49
7.5.4 BLACKHOLE . . . 49
7.5.5 ANYCAST . . . 50
7.6 Examples . . . 50
7.6.1 Setting communities when receiving . . . 50
7.6.2 Setting communities when redistributing . . . 50
7.6.3 Removing communities . . . 51
7.6.4 Scrubbing communities incoming . . . 52
7.7 What to do with them? . . . 53

Draft

7.8 Informational communities . . . 53
7.9 Action communities . . . 54
7.10 Encoding . . . 54

**8 Traffic Engineering** **57**
8.1 Remarks about wording . . . 57
8.2 Tools for outgoing traffic - received prefixes . . . 57
8.2.1 BGP Best Path Selection . . . 57
8.2.2 Local Preference . . . 58
8.2.3 AS path length . . . 58
8.2.4 MED . . . 58
8.3 Tools for incoming traffic - announced prefixes . . . 60
8.3.1 AS path length . . . 60
8.3.2 Announcing *more specific* prefixes . . . 60

**9 BGP Security** **63**
9.1 Introduction . . . 63
9.2 Automation . . . 63
9.3 Simple measures . . . 64
9.3.1 Maximum prefixes . . . 64
9.4 Protecting your router . . . 64
9.5 Protecting your BGP sessions . . . 65
9.5.1 Motivation . . . 65
9.5.2 MD5 session password . . . 65
9.5.3 TCP Authentication Option . . . 65
9.5.4 TTL security . . . 65
9.6 BGP filtering . . . 66
9.7 Inbound: Prefix filtering . . . 66
9.7.1 Filtering against prefix sizes . . . 67
9.7.2 Filtering against RPKI-Invalid prefixes . . . 68
9.7.3 Filtering against non-routable prefixes . . . 70
9.7.4 More unwanted prefixes . . . 71
9.8 Route flap dampening . . . 71
9.8.1 Motivation and history . . . 71
9.8.2 How does it work? . . . 72
9.8.3 Current recommendation . . . 72
9.9 Inbound: Next-hop filtering . . . 73
9.10 Inbound: AS-Path based filtering . . . 74
9.10.1 Private AS numbers . . . 74
9.10.2 Special AS numbers . . . 74
9.10.3 Implementation example: Cisco . . . 75
9.10.4 Inbound from customers: AS filtering . . . 75
9.11 Inbound and outbound: BGP community handling . . . 75
9.12 Outbound: Sending prefixes . . . 76
9.12.1 Prevent propagation of incorrect routing information . . . 76
9.13 [RFC9234] on preventing accidential floods . . . 76
9.13.1 Roles . . . 76
9.13.2 *Only to customer* (OTC) Attribute . . . 77
9.14 Blackholing . . . 77
9.14.1 Theory . . . 78

Draft

9.14.2 Implementation . . . . . 78
9.14.3 Operation . . . . . 80

**10 BGP - Advanced Concepts** **83**
10.1 BGP Confederations . . . . . 83
10.1.1 Configuration . . . . . 84
10.1.2 The AS Path . . . . . 84
10.2 FlowSpec . . . . . 85
10.2.1 Motivation . . . . . 85
10.2.2 Toolbox . . . . . 85
10.3 BGP as routing-protocol in data centers . . . . . 86

**A Acknowledgements** **87**

**Glossary** **89**

**Acronyms** **93**

**B License** **99**

Draft

# Chapter 1

# Introduction

There are many BGP related trainings out there. Most of them opt for the “one training for all” approach. This is not the goal of this training. This BGP training is aimed at network engineers working for networks who are connected to one or more upstream providers and also participate in peering at an Internet Exchange (IXP).

Therefore, the focus will be on eBGP, traffic engineering, filtering, and security.

Draft

## 1.1 What is BGP?

Some quick facts about BGP:

- BGP is a routing protocol
- BGP stands for **B**order **G**ateway **P**rotocol
- BGP was first defined in 1989 in [RFC1105]
- Since then, BGP has been continuously extended and updated
- The predecessor of BGP was called EGP (Exterior Gateway Protocol, defined in 1982, [RFC827])
- BGP is standardized. Internet standards are called Requests for Comment (RFCs). There is a process for creating and updating them. The institution responsible for this is called the Internet Engineering Task Force (IETF). All RFCs are public and can be viewed at https://rfc-editor.org.
- BGP runs on top of TCP - which takes care of reliable transport of BGP messages.

## 1.2 Elements of routing

### 1.2.1 IP prefixes

One of the key elements in Internet routing is the *prefix*. A prefix is the network part of an IP address. Prefixes are expressed as follows: The prefix itself, a slash, and a prefix length.




**Figure 1.1:** An IP address and the corresponding IP prefix

See figure 1.1 for an example for IPv4. It is important to know, that the host part of an IP prefix contains only zeros.

IPv6 prefixes are similar; they are written as follows: 2001:db8:517::/48. Here also the host part is all-zero.

Draft

The number behind the '/' is called the prefix length. This is the number of bits in the network part. The smaller the number, the larger the prefix (larger because there is more space in the host part of the prefix).

If one prefix is contained in another prefix, you can say that the smaller prefix (the one with the larger number behind the '/') is *more specific*.

Examples:

- 198.51.100.0/26 is more specific than 198.51.100.0/24.
- 2001:db8:6695::/48 is more specific than 2001:db8::/32.

See table 1.1 for some examples of prefixes (and things which are not a prefix).

Table 1.1: Prefix or not?

| Item | Prefix? | Explanation |
|---|---|---|
| 192.168.1.0/24 | Yes | Network part and length |
| 192.168.1.12/24 | No | Host part not zero, this is an IP address and the corresponding netmask |
| 10.0.0.4/30 | Yes | Host part is zero (check!) |
| 80.81.193.0/21 | No | Host part is not zero (convert to binary and check!) |
| 2001:db8::/32 | Yes | IPv6 standard prefix |
| dead:beef:food::/48 | Yes | IPv6 is in hexadecimal. So letters a-f are allowed. |
| 2001:db8::4/126 | Yes | Host part is zero. |
| 2001:db8:6695::221:1/96 | No | IPv6 address and netmask. |

IPv4 prefixes in the Global Routing Table (announced between providers via BGP) do not come in all sizes. See table 1.2 for common network sizes. Usually you will not find IPv4 prefixes smaller then /24 in the Global Routing Table.

Table 1.2: Common IPv4 network sizes

| Netmask | Subnet mask | Addresses | Typical use |
|---|---|---|---|
| /8 | 255.0.0.0 | $2^{24}$ = 16777216 | Largest block allocated to Internet Service Providers (ISPs) or assigned to end users |
| /9 … /15 | | | |
| /16 | 255.255.0.0 | 65536 | Large ISP |
| /17 … /20 | | | |
| /21 | 255.255.248.0 | 2048 | DE-CIX Frankfurt Peering LAN |
| /22 | 255.255.252.0 | 1024 | Current allocation of IPv4 space to new ISPs in RIPE region |
| /23 | 255.255.254.0 | 512 | |
| /24 | 255.255.255.0 | 256 | Smallest network routable between providers |
| /25 | 255.255.255.128 | 128 | |
| /26 | 255.255.255.192 | 64 | |
| /27 | 255.255.255.224 | 32 | |
| /28 | 255.255.255.240 | 16 | |
| /29 | 255.255.255.248 | 8 | Smallest multi-host network (6 hosts) |
| /30 | 255.255.255.252 | 4 | Often used on point-to-point links (two usable host addresses) |
| /31 | 255.255.255.254 | 2 | Point-to-point, not possible on all routers |
| /32 | 255.255.255.255 | 1 | Single host route. Used for loopback interfaces or Blackholing. |

Draft

In IPv6 the smallest routable prefix in the Global Routing Table is /48. Unlike in IPv4, when talking about the size of an IPv6 network it is not important how many hosts an IPv6 network can contain (the standard IPv6 LAN is a /64 and it can contain

$$2^{64} = 18446744073709551616$$

hosts). In IPv6, the question is rather how many /64 sub-networks a network can contain. See table 1.3 for common IPv6 network sizes.

### 1.2.2 The Autonomous System

Another element which needs to be introduced before we can begin with BGP is the *Autonomous System*.

#### Motivation

The Internet is and always has been a "Network of Networks". So a big network made up from a lot of small, *independent* networks. Instead of "independent" you could also say these part-networks are "*autonomous*", so we call them "*Autonomous Systems*". And to identify themselves against other part-networks of the Internet, each of them gets an identifier, a so-called *Autonomous System Number*.

[1] $2^{24}$ = 18446744073709551616

**What is an Autonomous System?**

In [RFC1930] an Autonomous System (AS) is defined as follows:

> "The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol (IGP) and common metrics to determine how to route packets within the AS, and using an inter-AS routing protocol to determine how to route packets to other ASes."

This definition is now replaced by the following more general wording (defined in [RFC1930]):

> "An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy."

So the focus is on prefixes and how they are routed:

- *...connected...*: An Autonomous System is continuous. All entities within are connected with each other. [1]
- *...group of one or more IP prefixes...*: This is about IP prefixes, not about devices. Routers are not even mentioned. All prefixes (or prefix, it can range from one to many) are grouped together under an AS and identified by an AS number.
- *...run by one or more network operators...*: An AS does not have to be run by only one operator if all other conditions are matched.
- *...SINGLE and CLEARLY DEFINED routing policy...*: This is the most important part. Internally and externally all prefixes belonging to the same AS are routed the same way. That does not mean you cannot adjust the announcement of single prefixes.

Draft

### 1.2.3 Autonomous System Numbers

An AS is uniquely identified by an *Autonomous System Number* (ASN). ASNs used to be 16-bit numbers (defined in [RFC1930]) but some years ago the Internet Assigned Numbers Authority (IANA) was running out of numbers to distribute, so the number space of ASNs

[1] There are exceptions. AS112 for example is independently operated at multiple locations for DNS reverse resolving of private IP space.

Table 1.3: Common IPv6 networks sizes

| Netmask | Addresses | Used for | Why? |
|---|---|---|---|
| /127 | 2 | Point-to-point | |
| /126 | 4 | Point-to-point | |
| ... | | | |
| /64 | $2^{64}$ see[1] | LAN | Can address as many hosts as you need |
| ... | | | |
| /56 | $2^{56}$ | Assigned to residential users | Contains 256 /64 subnets |
| ... | | | |
| /48 | $2^{48}$ | Assigned to business users | Contains 65536 /64 subnets |
| ... | | | |
| /32 | $2^{32}$ | Current minimum allocation size of IPv6 space to ISPs | |

has been extended to 32-bit. For this, BGP had to be extended as well; this was done in [RFC6793].

Today, AS numbers are 32-bit. You can no longer request a 16-bit number unless you have a very good reason.

#### How to get an AS number

ASNs are administrated and handed out like IP addresses. The IANA assigns blocks of ASNs to Regional Internet Registries; they assign them to Local Internet Registries (= Internet Service Providers), and they hand them out to end users. To get an AS number you can either:

- Become a customer (Local Internet Registry) of your local Regional Internet Registry (RIR), or
- ask an existing Local Internet Registry to get an ASN for you.

The procedures for getting an AS number are different for each region, so please check online, depending on where you are:

**Europe, Russia or the Middle East:** `https://www.ripe.net/publications/docs/ripe-679`

**North Amercia:** `https://www.arin.net/resources/request.html#asn`

**South America:** `http://www.lacnic.net/1016/2/lacnic/ip-request`

**Asia:** `https://www.apnic.net/get-ip/get-ip-addresses-asn/asn-requests/`

**Africa:** `https://www.afrinic.net/library/policies/1829-afrinic-consolidated-policy-manual#s7_0`

Draft

In general, you must justify your need for an AS number (for example, you want to peer or have multiple upstream providers).

#### Looking up AS numbers

There are a number of ways you can look up AS numbers.

**Whois** is available on most systems with a command line. Simply type in *whois AS*nnnn and check the output.

**PeeringDB** is an online database of networks who peer. Go to `https://peeringdb.com` and type in the AS number in the search field.

**RADB** mirrors all the databases of the RIRs. Just type in the AS number in the search field.

**Regional Internet Registries** websites can be used also to search for Autonomous System Numbers (ASNs).

### 1.2.4 The AS path

The AS path is built when announcing prefixes via BGP. Each AS adds its number to the front of the path. The AS can also be added multiple times to make an artificially longer AS path. See picture 1.2 for how an AS path is built by announcing and re-announcing an IP prefix.

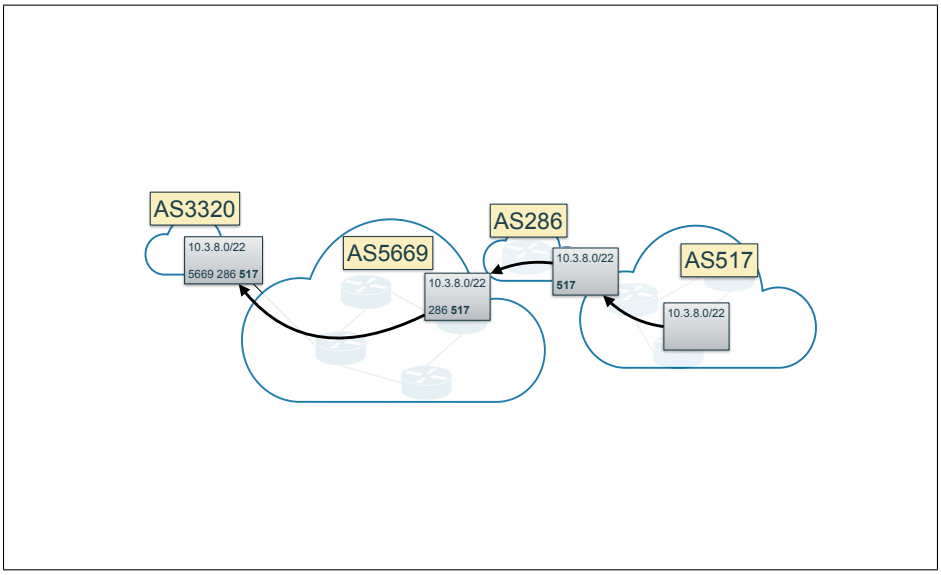



**Figure 1.2:** How an AS path is built

## 1.3 Some theory about routing protocols

### 1.3.1 Classification of routing protocols

Draft

There are several ways to classify routing protocols:

- Static vs. dynamic
- Interior protocols vs. exterior protocols
- Link state vs. distance vector

### 1.3.2 Static vs. dynamic routing

In static routing, you configure all the routing decisions into the routers statically. So if anything in the network changes, routers have no way to react and change their routing decisions.

In dynamic routing, routers "talk" to each other using a routing protocol about (for example) the network state and/or reachability of IP addresses. So in case of a network change, routers can adapt to the new situation.

### 1.3.3 Interior protocols vs. exterior protocols

An Interior Gateway Protocol (IGP) is a protocol which routers under one administrative domain use to communicate with each other about network state and reachability. Examples of IGPs are: Open Shortest Path First, IS-IS, Enhanced Interior Gateway Routing Protocol (EIGRP), Routing Information Protocol (RIP).

Exterior routing protocols are protocols which different administrative entities use to exchange information. Nowadays, BGP4 is the only exterior protocol remaining.

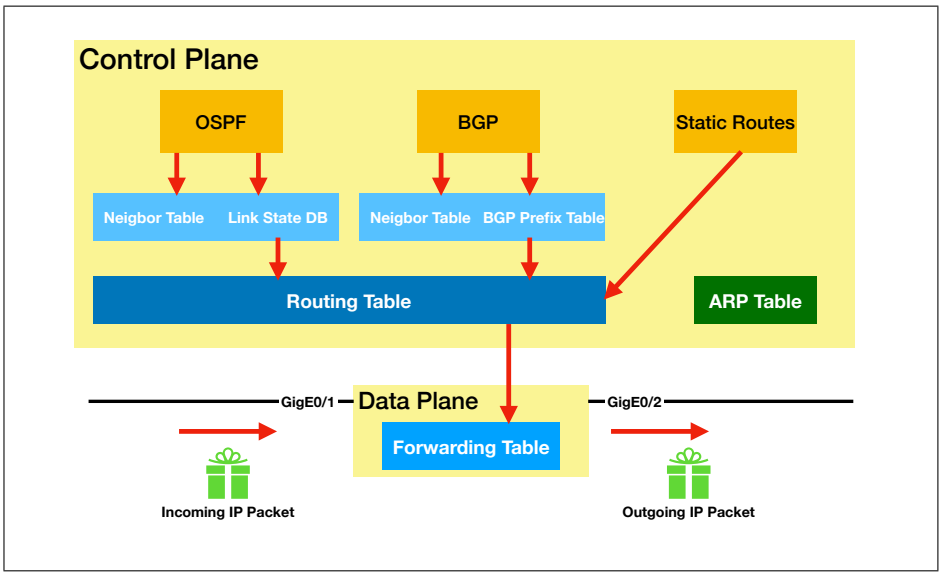



**Figure 1.3:** A simplified diagram of a router

### 1.3.4 Link state vs. distance vector protocols

In a link state protocol, each router has a map of the whole network and calculates the best path to a destination. Examples for link state protocols are OSPF and IS-IS.

In a distance vector protocol, the calculation of the best path is done by routers exchanging the routing tables with each other and choosing the path with the least number of hops (or in the instance of BGP: number of Autonomous Systems) to cross.

Examples of distance vector protocols are RIP and Border Gateway Protocol (BGP).

Draft

## 1.4 How a router works

But how do these different protocols act together in a router? Please see figure 1.3 - it shows the various tables in a router and how they interact.

**Forwarding Table** is often realized in hardware (ASICs). It has to be very fast and is consulted for each and every packet a router forwards. How it works and how it is implemented is different for each router.

**Routing Table** exists for each protocol the router is routing (so most of the time, one table for IPv4 and another table for IPv6). Usually for each prefix the router knows about, it contains one next-hop IP address and the interface via which this IP can be reached.

**Static Routes** are simply installed in the routing table.

**Link State Database** is used by OSPF and contains a complete "picture" of the network including all nodes and links in between the nodes. OSPF calculates the best path to a destination and installs it as a route (including a next-hop address) in the routing table.

**BGP Prefix Table** contains all received prefixes from all BGP neighbors. After the BGP process has done its *best prefix selection*, **one** of these routes is installed in the routing

table.

## 1.5 Examples in this book

Some sections contain configuration examples for a number of routers. As the document is (and perhaps always will be) in draft state these examples are incomplete. All examples have been tested for syntactical correctness, but are limited to routers available (real or in simulation) to the author. Pull requests with examples for additional routers are welcome.

## 1.6 BGP and IPv6

Usually BGP training programs and manuals have their own chapter about IPv6. But as it is now 2026 and IPv6 should be quite common, all information about how to implement IPv6 routing is directly integrated. You should not setup BGP for IPv4 first and add IPv6 capabilities later - best practice is to integrate IPv6 from the beginning.

BGP is much older then IPv6. The first incarnation of BGP was described in [RFC1105] in 1989 (building on experience with its predecessor protocol Exterior Gateway Protocol (EGP)) - IPv6 was specified in [RFC1883] in 1995.

BGP4 (the still-current version) and predecessors were built for distributing IPv4 prefixes only. But unlike IP itself and other routing protocols like Open Shortest Path First, BGP4 was designed with extensibility. So it was not necessary to introduce a new protocol; BGP4 was simply extended.

draft

And because nobody wanted to do this over and over again, the extension to BGP4 was not just to accommodate IPv6, but for multiple network protocols. This was published first in [RFC2283] (but the most current version of the extensions are in [RFC4760]). The extension was backward compatible, so routers which had them could communicate with routers which did not have them.

## 1.7 Recommendations

I rarely give recommendations. This book and the corresponding training seminar should give you the knowledge to make your own choices. However, sometimes perhaps an obvious choice migth not be the best one. In this case you might find my recommendations useful, but wether you follow them or not, is up to you.

Recommendations are clearly marked like this:



**Recommendation:** Read, understand, and make your own choice.

# Chapter 2

# Setting up an IGP

## 2.1 Motivation

BGP never runs alone. To distribute the IP addresses of interfaces within an AS, an Interior Gateway Protocol (IGP) is needed.

There are several options which IGP to use, but normally you will not be free to choose. Either someone has already made the decision, or you are restricted on what your router vendor supports.

Draft

If you really have several options, keep in mind the following points when making your decision:

**Openness:** The IGP must be supported not only by the routers you are using currently, but also by the routers you are purchasing over the next few years.

**IPv6 awareness:** Even if you are currently not using IPv6, you will some time in the future. Your IGP should either support IPv6 or you should be able to run a 2nd IGP in parallel for IPv6.

## 2.2 Configuring OSPF

Open Shortest Path First runs on top of IP. So you need to configure IP addresses for your interfaces, and then define the use of OSPF on these IPs. It really helps if you have all your interface IP addresses out of the same network block (and use this block for nothing else).

Also, OSPF has the concept of *areas*: You have to define at least one area, called either backbone or area 0 (zero).

Example (Cisco):

```
interface GigabitEthernet1/0
 ip address 192.168.2.2 255.255.255.252
!
router ospf 64500
 network 192.168.2.0 0.0.0.255 area 0
```

This would switch on OSPF on all interfaces with an IP address within 192.168.2.0/24.

Same example for Mikrotik routers:

```
/ip address
add address=192.168.2.2/30 interface=ether1 network=192.168.2.0

/routing ospf network
add area=backbone network=192.168.2.0/24
```

Same example for FRRouting software router:

```
interface wlan0.101
  ip address 192.168.2.2/30
!
router ospf
  network 192.168.2.2/30 area 0
```

On Juniper you enable OSPF on the interfaces:

```
  interfaces {
      em0 {
          unit 0 {
              family inet {
                  address 192.168.2.2/30;
              }
          }
      }
  }
  protocols {
    ospf {
      area 0.0.0.0 {
        interface em0.0;
      }
    }
  }
```

Draft

Keep in mind that OSPFv2 (the “standard” OSPF version) supports IPv4 only. This is completely fine; for IPv6 you can use OSPFv3 to keep things separate. Or you can run IS-IS which supports IPv4 and IPv6.

## 2.3 Configuring IS-IS

IS-IS does not run on top of IP but runs directly on layer 2 (Ethernet or other). So you do not have to define the network it runs on, but the interfaces you want it enabled on.

*Important*: The protocols you enable it for must be the same on both sides of a connection. So if you enable IS-IS for IPv4 and IPv6 on one interface, you must do so as well on the connected interface.

Example for IS-IS on Cisco:

```
interface GigabitEthernet0/0
  ip address 192.168.2.2 255.255.255.252
```

```
  ipv6 address autoconfig
  ip router isis
  ipv6 router isis
!
router isis
  net 49.0000.0000.0000.0001.00
```

The number behind the “net” statement is the router-id and must be unique within your network.

Mikrotik does not support IS-IS.

Example for IS-IS on FRRouting:

```
router isis myname
  net 49.0000.0000.0000.0001.00
!
interface eth1
  ip address 192.168.2.2/30
  ipv6 address autoconfig
  ip router isis myname
  ipv6 router isis myname
```

Example for IS-IS on Juniper is a little bit more complex. For the loopback interface *loo* we configure a static IPv6 address, for the ethernet interface we use auto-configuration. Also, the router-id goes into the loopback interface and *family iso* has to be enabled on all interfaces where IS-IS is to be used.

Draft

```
  interfaces {
      em0 {
          unit 0 {
              family inet {
                  address 192.168.2.2/30;
              }
              family iso;
              family inet6;
          }
      }
      lo0 {
          unit 0 {
              family inet {
                  address 192.168.1.3/32;
              }
              family inet6 {
                address 2001:db8:500::1:3/128;
              }
              family iso {
                  address 49.0000.0000.0000.0003.00;
              }
          }
      }
  }
  protocols {
```

```
    isis {
        interface em0.0;
        interface lo0.0;
    }
```

## 2.4 Configuring OSPFv3

To integrate IPv6 a new version of OSPF was necessary - this came into being as OSPFv3. Except for virtual links, OSPFv3 uses IPv6 link-local addressing. It is configured on the interface, not on a network like OSPFv2.

The only remains of 32bits are area and router ids - they are still 32bit like IPv4 addresses.

Example for OSPFv3 on Cisco IOS:

```
interface Loopback0
  ipv6 address 2001:DB8:500::1:3/128
  ipv6 ospf 64500 area 0
!
interface GigabitEthernet0/0
  ipv6 address autoconfig
  ipv6 ospf 64500 area 0
```

In this example, *64500* is the process id of the OSPFv3 process - you can choose any number. It is not even necessary to define the process itself - simply enable OSPv3 on all interfaces where you want to use it. Whether you want to use auto-configured IPv6 addresses on your link interfaces or static IPv6 addresses is up to you.

Example for Mikrotik:

```
/ipv6 address
  add address=2001:DB8:500::1:1/128 interface=loopback0
  add interface=ether0 address=2001:DB8:496::1:2/126

/routing ospf-v3 interface
  add area=backbone interface=loopback0
  add area=backbone interface=ether0
```

Example for FRRouting:

```
router ospf6
  router-id 192.168.1.1
  interface dummy0 area 0.0.0.0
  interface eth1 area 0.0.0.0
  interface eth2 area 0.0.0.0
```

You have to configure an (IPv4-like) router-id manually - the best approach is to use the IPv4 address of your Loopback interface. Also the area notation is written like an IPv4 address.

Example for Juniper:

```
  protocols {
    ospf3 {
        area 0.0.0.0 {
```

```
            interface lo0.0;
            interface em0.0;
        }
    }
}
```

Draft

Draft

# Chapter 3

# iBGP - BGP within one Autonomous System

## 3.1 What is iBGP?

iBGP is not a separate protocol. It is simply the “flavor” of BGP which is spoken between routers which are in the same Autonomous System (AS).

All BGP routers use Transmission Control Protocol (TCP) to communicate with each other.

Draft

Prefixes are distributed in iBGP according to the following rules:

- Each router only distributes the best route for a prefix
- Prefixes received via iBGP (from the same AS) are *not* distributed
- Prefixes received via eBGP (from other ASes) are distributed.

Since prefixes received from routers via iBGP are not redistributed (there are exceptions, see 3.2.3), iBGP has to be *fully meshed*, meaning that each router needs to have an iBGP session to all other routers.

By default, the next-hop address of a prefix received via eBGP is redistributed *unchanged*. If you do not want to redistribute the IP addresses (IPv4 and IPv6) of external interfaces within your AS, you need to set an option on your iBGP sessions to change this next-hop address to the address of your iBGP session. Usually this option is named *next-hop-self*.

BGP4 is multi-protocol capable. That means that you can use IPv4 to both distribute IPv4 and IPv6 prefixes. However, this is not really advisable, as the next-hop address (which is also distributed) has to be set manually then. Best practice is to use IPv4 transport to distribute IPv4 prefixes and IPv6 transport to distribute IPv6 prefixes.

## 3.2 Configuring iBGP

Like stated above, iBGP uses TCP to communicate. So to set up an iBGP session between two routers, you need to define:

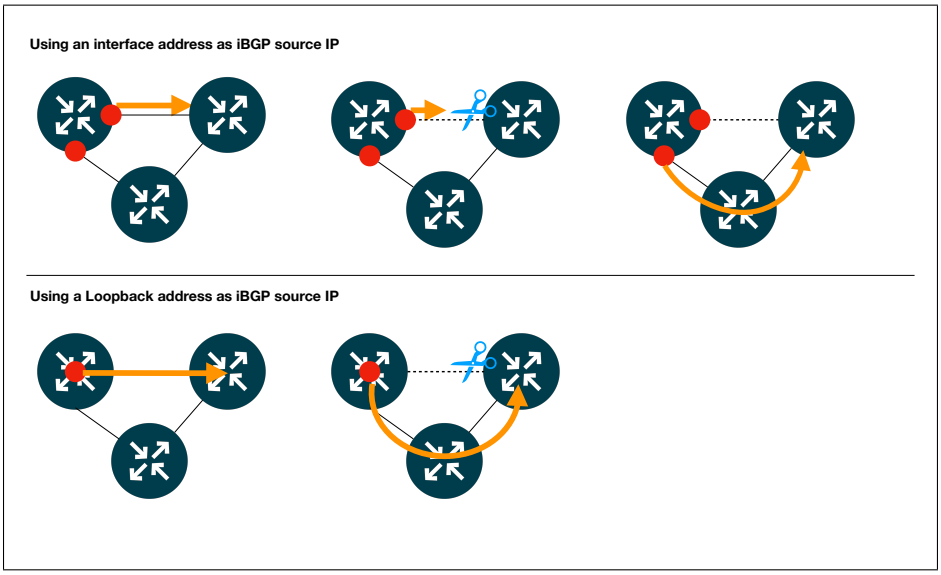



**Figure 3.1:** iBGP connection with and without Loopback interface

- IP address to connect to
- local IP address to be used as source (optional, but recommended)

Draft

### 3.2.1 About the Loopback interface

If you do not tell BGP which local IP address to use, routers use as source IP the IP address of that interface on which packets leave the router. This might not be advisable, because if that interface goes down all iBGP sessions using the IP address of this interface get disrupted, even if there is another path to the remote router.

Therefore, best practice is to define a *Loopback interface*, an artificial internal interface which is not connected to anything and always stays up. As the Loopback interface is not connected to anything, the address of the Loopback interface should be a /32 in IPv4 and a /128 in IPv6.

This is illustrated in figure 3.1 - if an interface IP is used, the TCP connection gets disrupted in the instance of a circuit failure and has to be re-established (of course the iBGP connection also has to be re-established, including the exchange of all BGP prefix information). If a Loopback interface is used, the TCP connection is simply re-routed and the iBGP connection will stay up.

Also, for IPv6 you can easily use link-local addresses on the interconnect and just have the Loopback statically configured and use it for iBGP. If you have the luxury to choose, you might have “similar looking” IPv4 and IPv6 addresses for the Loopback interface. But beware of giving your IPv6 addresses any “meaning” - just select an IPv6 prefix and assign addresses out of it. Otherwise, you will run into problems later on, as giving meaning to addresses seldom scales.

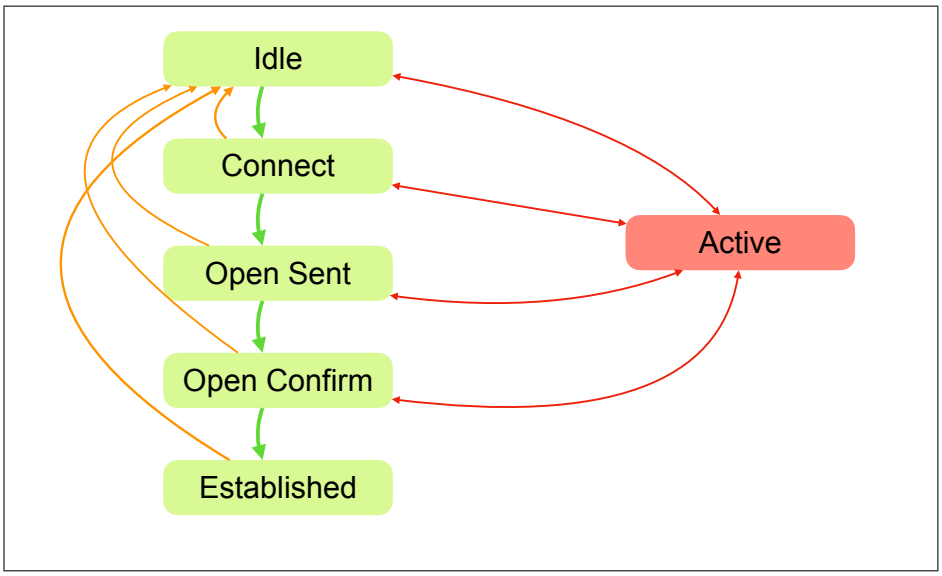



**Figure 3.2:** BGP finite state machine

### 3.2.2 Lifecycle of a BGP connection

If you configure BGP on a router, if the session comes up, everything is fine. But sometimes the session cannot be established, and if you check the BGP session table you might see a "status" of the session. Some of these states that a session can be in are self-explanatory, but some require more explanation. You will find below a list of session states you might see. These session states are defined in [RFC4271] as a finite state machine. All states described are *per peer*.

Draft

**Idle** is the initial state of any BGP connection. In this state, no resources are allocated to the session and incoming connection attempts are refused. Depending on events, the state is either changed to *active* or *connect*.

**Connect** means that BGP is waiting for a TCP connection to a neighbor to be completed. If this succeeds, the state of the session is changed to *OpenSent*, if not, it's either back to *idle* or to *active*.

**Active** - you will see this state a lot. Be aware that it means that a BGP session is *not* established. BGP in this state waits for an incoming TCP connection. If the retry timer expires, the state changes to *connect* to re-try connecting.

**OpenSent** - BGP has established a TCP connection, sent an *open* message to the peer and waits for an incoming *open* message. Once received and checked ok, the state is changed to *OpenConfirm*. In case of a timeout or an error, the state falls back to either *idle* or *active*.

**OpenConfirm** - here BGP waits for a *keepalive* or *notification* message from the peer. We are nearly there! If one of these messages are received, the state is changed to *established*. If something goes wrong, again we fall back to either *idle* or *active*.

**Established** is the state that you want your BGP session to be in. Prefixes can be exchanged and everything is fine. From here you can only go back to *idle*.

See also figure 3.2 for the states and how they may change.

#### Notifications on shutdown

When a BGP session is shut down, the party initiating the cease of the BGP session is sending out a notification. [RFC4486] lists a number of pre-defined reasons for shutting down a session.

**Maximum number of prefixes reached** - this means that the other side is sending more prefixes then the receiver allows (see 9.3.1). Sending of this notification is mandatory.

**Administrative shutdown** - the session has been shut down using a configuration command.

**Peer de-configured** - the BGP session has been de-configured.

**Administrative reset** - the session has been reset using a command. It will be re-established again.

**Connection rejected** - if the BGP session is disallowed *after* a TCP session has already been established this notification should be sent.

**Other configuration change** - if a session is reset for any other reason as the ones stated above.

**Connection collision resolution** - this should be sent if a collision occurs while establishing a session.

**Out of resources** - this may be sent if a router runs out of resources (like memory) and tears down the session.

Draft

The newer [RFC8203] enhances the *Administrative Reset* and *Administrative Shutdown* reasons with the possibility for the shutting down party to add a free-form text message.

It is up to the receiver of all these notifications to do something with them (or not) - for example the DE-CIX route server displays them in the looking glass - see `https://lg.de-cix.net`.

#### Timers

To keep BGP sessions alive (well, more to check *if* they are still alive) BGP speakers send each other *keepalive* messages. The following timers exist:

**Keepalive Timer:** This timer takes care of sending *keepalive* messages to the BGP neighbor. Everytime it expires a keepalive message is sent and the timer is reset (it is also reset every time a normal update message is sent). The start value of the Keepalive Timer may be configurable, it is usually one third of the Hold Timer.

**Hold Timer:** This timer determines how long a BGP session is considered being open without a *keepalive* or update message received. The start value of the timer is negotiated during session setup - it is usually three times of the initial value of the *Keepalive Timer*. It can be configured on a per-neighbor basis and in session initiation the lowest value configured of the two peers wins. A usual default value is 180 seconds. It is not a good idea to set this timer too aggressively short - this would increase the CPU load of the BGP process.

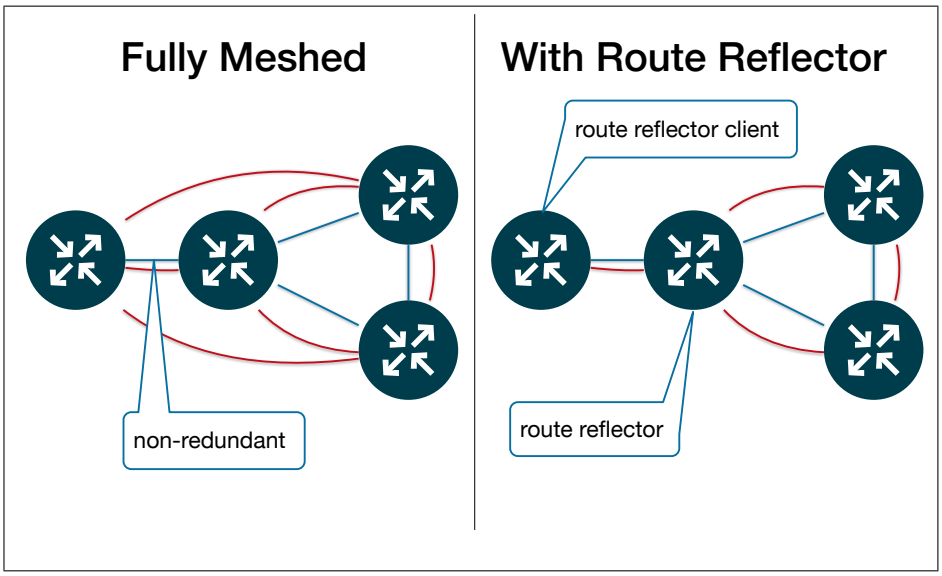



**Figure 3.3:** Application of a Route Reflector

If the hold timer expires, the BGP session is considered being down and all prefixes received from this session are removed from the BGP table and the session state is set to *idle*.

Given the usual default values it may take up to three minutes to detect if a BGP neighbor is down. During that time, if the interface the session is established on is still up, prefixes received via this session are still valid and so traffic may be dropped.

**Recommendation:** Do not modify the BGP timers unless you are absolutely sure you know what you are doing. If you want to take down BGP sessions fast if a neighbor goes down, use Bidirectional Forwarding Detection (BFD)

### 3.2.3 Fully Meshed vs. Route Reflector

As an iBGP speaker by default does not forward prefixes received via iBGP, your iBGP nodes need to be *fully meshed*. This means each BGP speaking router needs an iBGP connection to any other BGP speaking router (with $n$ routers, each router therefore has $(n-1)$ iBGP sessions; in your network you have $n*(n-1)/2$ iBGP sessions in total).

Now imagine one of your edge routers is connected in a non-redundant way to the rest of your network (see figure 3.3). There is no point setting up multiple iBGP sessions - if the connection to this edge router fails, *all* of them will go down. This is a good application for using a *Route Reflector*. A Route Reflector sends out all prefixes (even those received via iBGP) to its Route Reflector Clients. These clients usually have only one BGP connection - to their server.

Using route reflection, it is also possible to build iBGP networks using only Route Reflectors and clients. These architectures are beyond the scope of this document.

Configuration has only to be done on the reflector side; usually it's just one statement.

**Cisco:**
```
neighbor 172.16.1.2 route-reflector-client
neighbor 172.16.1.2 next-hop-self all
```

"next-hop-self all" sets the next hop of announced prefixes to the IP of the reflector for all prefixes announced to the client. With this set, you might not even have to run an IGP on your client.

**Mikrotik:**
```
/routing bgp peer
add name=R2 nexthop-choice=force-self \
  remote-address=172.16.1.2 \
  remote-as=64500 route-reflect=yes
```

The "force-self" in nexthop-choice sets the interface IP of the reflector as nexthop when announcing prefixes to the client.

**Juniper:** To configure a route reflector you simply have to give it a *cluster* id, best is to do that in a group:

```
protocols {
    bgp {
        local-as 64500;
        group internal-rr {
            type internal;
            family inet {
                unicast;
            }
            export rr-next-hop;
            cluster 172.16.1.1;
            peer-as 64500;
            neighbor 172.16.1.2;
        }
    }
}
```

Draft

Setting the next hop is done with a policy-statement:

```
policy-options {
  policy-statement rr-next-hop {
      term 1 {
          from protocol bgp;
          then {
              next-hop self;
          }
      }
  }
}
```

In this case, using a Loopback interface for the session and for next-hop is not advisable. The client is not connected redundantly, so if the connection goes down a Loopback interface does not help. Also, when using the interface IP for the iBGP session *and* a forced next-hop for announced prefixes, you do not even have to run an IGP to the client.

### 3.2.4 Keeping the configuration short

In general, it is advisable to keep the iBGP configuration as short as possible. Depending on the router you are using, there is the possibility to group BGP commands. On Cisco this is called a “peer group”.

To keep IPv4 and IPv6 configuration separate, use one peer group for each - also most routers do not allow mixing of configuration here.

### 3.2.5 Cisco example

Configuration example for Loopback interface and iBGP for Cisco IOS:

```
interface Loopback0
  ip address 172.16.1.1 255.255.255.255
  ipv6 address 2001:db8:500::1:1/128
!
router bgp 64500
  neighbor internal peer-group
  neighbor internal remote-as 64500
  neighbor internal update-source Loopback0
  !
  neighbor internal6 peer-group
  neighbor internal6 remote-as 64500
  neighbor internal6 update-source Loopback0
  !
  neighbor 172.16.1.2 peer-group internal
  neighbor 172.16.1.3 peer-group internal
  neighbor 172.16.1.4 peer-group internal
  !
  neighbor 2001:db8:500::1:2 peer-group internal6
  neighbor 2001:db8:500::1:3 peer-group internal6
  neighbor 2001:db8:500::1:4 peer-group internal6
  !
  address-family ipv4
    neighbor internal next-hop-self
  !
  address-family ipv6
    neighbor internal6 next-hop-self
  ...
```

This is the minimum configuration. The local AS number is specified in the router command and the remote AS in the peer group.

The “next-hop-self” command ensures that prefixes received from other ASes get this router’s IP set as next-hop address (so external IPs do not have to be redistributed using our IGP). Note that this is below the “address-family” statement.

To set up iBGP sessions to each remote router, you need only configure its IP address and that it belongs to the peer group “internal”.

To distribute the IP address of the Loopback interface, we extend the configuration of our IGP as shown in 2.2:

```
router ospf 64500
  network 192.168.2.0 0.0.0.255 area 0
  redistribute connected subnets
```

So again: We set up iBGP sessions between the Loopback interfaces of all routers.

For IPv6, if you use IS-IS, simply enable it on the Loopback interface:

```
interface Loopback0
  ip router isis
  ipv6 router isis
```

Similarly for OSPFv3; here you also have to add the number of the OSPFv3 process and an area:

```
interface Loopback0
  ipv6 ospf 64500 area 0
```

### 3.2.6 Mikrotik example

Mikrotik does not directly support Loopback interfaces, but you can use an empty bridge interface instead:

```
/interface bridge
add name=loopback0
/ip address
add address=172.16.1.1/32 interface=loopback0 network=172.16.1.1
/ipv6 address
add address=2001:DB8:500::1:1/128 interface=loopback0
```

Draft

As an IGP, Mikrotik only supports OSPFV2 and OSPFv3:

```
/routing ospf-v3 interface
add area=backbone interface=loopback0
add area=backbone interface=ether1
add area=backbone interface=ether2
...
```

Also, Mikrotik does not support peer groups; you have to configure everything in the peer entry:

```
/routing bgp instance
set default as=64500 router-id=172.16.1.1

/routing bgp peer
add name=R2 nexthop-choice=force-self remote-address=172.16.1.2 remote-as=\
  64500 update-source=loopback0
add name=R3 nexthop-choice=force-self remote-address=172.16.1.3 remote-as=\
  64500 update-source=loopback0
add name=R2-6 nexthop-choice=force-self remote-address=2001:db8:500::1:2 \
  remote-as=64500 address-families=ipv6
add name=R3-6 nexthop-choice=force-self remote-address=2001:db8:500::1:3 \
  remote-as=64500 address-families=ipv6
```

We restrict the IPv6 iBGP sessions to IPv6, using the “address-families” statement. Otherwise the IPv6 sessions would want to announce IPv4 prefixes as well, and we decided to keep this separate. On the IPv4 sessions, the address-families config is not required, as IPv4 is distributed by default.

Also keep in mind that on both sides of an iBGP (or any BGP) session, the same address families must be configured - otherwise the session does not come up.

Draft

Draft

# Chapter 4

# eBGP - Connecting to the outside world

## 4.1 What is eBGP?

We call a BGP session an eBGP (external BGP) session if it connects to a different Autonomous System.

Draft

### 4.1.1 Differences between iBGP and eBGP

There are a couple of differences between the two flavors of BGP.

One difference is that, while in iBGP it does not matter how many "hops" (in terms of the IP protocol) a neighbor is away, in eBGP the neighbor usually is directly connected. You can change this behavior if you configure the neighbor as being a "multi-hop" neighbor (`neighbor x.x.x.x ebgp-multihop`).

> How does it work that eBGP packets only travel to the next adjacent node? The answer lies with the IP *TTL (Time to Live)* counter: The sender sets the counter to one, so packets get discarded if the eBGP neighbor is more then one hop away. See chapter 9 to learn about a more secure approach to this.

Also the next-hop of IP addresses received via eBGP is set to the address the sender of the prefixes (your eBGP neighbor) specified. If you forward this prefix then via iBGP, this next hop is either unchanged or set to the IP of your router (`neighbor x.x.x.x next-hop-self`).

The router receiving a prefix via eBGP also checks if the next-hop IP address is accessible via the interface the prefix is received. If it is not, the eBGP announcement is discarded.

Your router also checks the AS-Path of all received prefixes. If your own AS number is in that path, the prefix is also discarded (this can be switched off by a config command which is *not recommended*).

Other measures to ignore unwanted prefixes etc. have to be configured manually using filter lists or route-maps. More about this you can read in chapter 9.

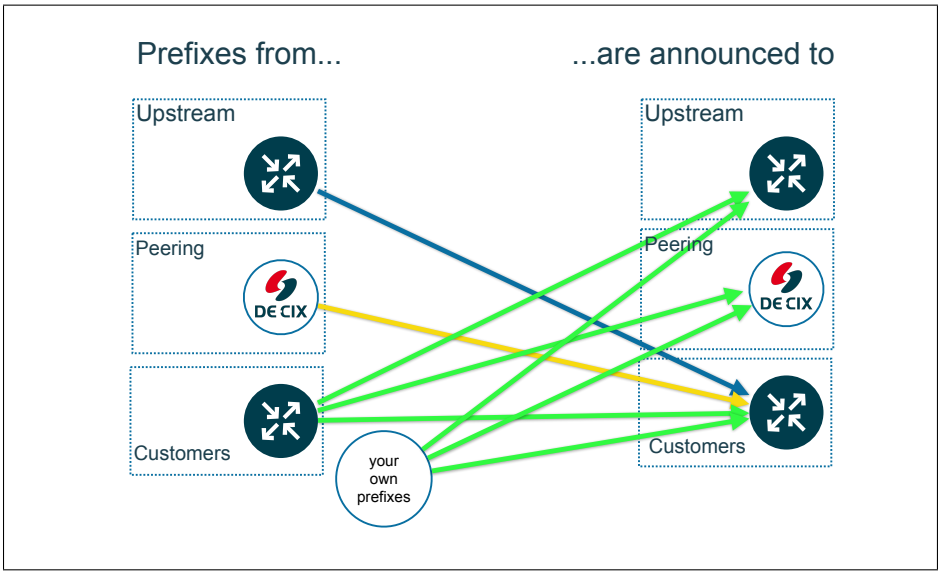



**Figure 4.1:** eBGP Roles

### 4.1.2 Best practices

It is best practice to not let prefixes in from outside without some filtering. Most of this filtering will be covered in chapter 9 when we talk about BGP security. At this time we will define empty filters for sending and receiving prefixes (named "-out" for sending "-in" for receiving). We will extend these filters later.

Draft

If your router supports it, it's again best practice to define a peer-group and have all common configuration in the peer-group (you need two peer-groups again, one for IPv4 and one for IPv6).

It is strongly recommended that you setup a filter to *not announce any* prefixes initially when setting up a completely new eBGP peer group. We will do so by configuring our "-out" route-map to deny everything.

Depending on your routers BGP implementation, configuration commands become active the moment you hit the "enter" key. So if you configure an eBGP neighbor *without* any filtering and the neighbor already has configure its side, you start flooding your complete BGP table immediately. [RFC8212] addresses this problem by requesting that eBGP sessions without any *configured* filtering get an automatic *deny-all* filter by default. Some vendors have already implemented this, some made it optional.

Some routers allow received information (before filter processing) to be stored in a separate table (at the cost of increased memory usage). To debug your filters, this is very useful, so I suggest you turn it on (see examples below).

### 4.1.3 Sources and destinations of prefixes

If you look at possible BGP neighbors, you come up with some categories you receive prefixes from and announce prefixes to. [RFC9234] defines to following BGP neighbor roles and relationships:

**Provider, Customer:** A provider announces all known prefixes to a customer, a customer announces only its own and its customers prefixes to a provider.

**Peer, Peer:** They announce their own and their customers prefixes to each other.

**Route Server, Route Server Client:** Similar to peers, a route server announces all prefixes to a route server client, which announces their own and their customers prefixes to a route server.

See picture 4.1 for a graphical expression of roles and relationships. These relationships must be expressed in filter configurations. You can do that manually (better: automated) with allow-lists or you can implement sophisticated mechanisms. Choose what ever fits best to your network size. Chapter 9 and chapter 7 will give you some ideas.

### 4.1.4 Cisco example

In this simple example we *deny* any outgoing prefixes.

```
route-map upstream-out deny 10
!
route-map upstream-in permit 10
!
route-map upstream6-out deny 10
!
route-map upstream6-in permit 10
!
router bgp 64500
  neigbor upstream peer-group
  neigbor upstream6 peer-group
  neighbor 192.168.3.2 remote-as 65550
  neighbor 192.168.3.2 peer-group upstream
  neighbor 2001:db8:300::2 remote-as 65550
  neighbor 2001:db8:300::2 peer-group upstream6
  !
  address-family ipv4
    neigbor upstream route-map upstream-in in
    neigbor upstream route-map upstream-out out
    neigbor upstream soft-reconfiguration inbound
    neigbor upstream6 route-map upstream6-in in
    neigbor upstream6 route-map upstream6-out out
    neigbor upstream6 soft-reconfiguration inbound
    neighbor 192.168.3.2 activate
    no neighbor 2001:db8:300::2 activate
  !
  address-family ipv6
    neigbor upstream route-map upstream-in in
    neigbor upstream route-map upstream-out out
    neigbor upstream soft-reconfiguration inbound
    neigbor upstream6 route-map upstream6-in in
    neigbor upstream6 route-map upstream6-out out
    neigbor upstream6 soft-reconfiguration inbound
    no neighbor 192.168.3.2 activate
```

Draft

```
    neighbor 2001:db8:300::2 activate
```

The *soft-reconfiguration inbound* ensures that you can check what your router receives before it is processed by the incoming route-map.

Take note of how the address-family parts and entry act together. “address-family” relates to the *announced* prefixes; the IP address of the neighbor defines the protocol of the BGP session. With the “activate” clauses we switch off IPv4 prefix announcement on the IPv6 sessions and switch on IPv6 announcement. On the IPv4 sessions, we do it the other way around. What increases the confusion is that some of these statements are considered to be default at Cisco and therefore are omitted in the config.

Do we need different route-maps for IPv4 and IPv6? That’s dependent on their complexity. *If* you want to do any IP related filtering (ip address lists, prefix lists) having two (or four) route-maps is recommended.

### 4.1.5 Mikrotik example

```
/routing bgp instance
set default as=64500 router-id=192.168.1.1

/routing bgp peer
add name=upstream-AS65550 in-filter=upstream-in \
  out-filter=upstream-out \
  remote-address=192.168.3.2 remote-as=65550
add name=upstream6-AS65550 in-filter=upstream6-in \
    out-filter=upstream6-out \
    remote-address=2001:db8:300::2 remote-as=65550 \
    address-families=ipv6

/routing filter
add chain=upstream-in action=accept
add chain=upstream-out action=discard
add chain=upstream6-in action=accept
add chain=upstream6-out action=discard
```

Mikrotik does not support peer-groups or similar, so everything needs to be configured on every peer. Here the “address-families” parameters determines what kind of prefixes are announced and accepted. Again, the default “IPv4 only” can be omitted - making the configuration rather more then less confusing.

We also do not announce any prefixes here (by using the *action=discard* statement in our filters).

### 4.1.6 FRRouting example

In FRRouting we can put the *activate* clauses into the peer group, also we enable [RFC8212] automatic filtering:

```
route-map upstream-out deny 10
!
route-map upstream-in permit 10
!
```

```
route-map upstream6-out deny 10
!
route-map upstream6-in permit 10
!
router bgp 64500
  bgp ebgp-requires-policy
  neigbor upstream peer-group
  neigbor upstream6 peer-group
  neighbor 192.168.3.2 remote-as 65550
  neighbor 192.168.3.2 peer-group upstream
  neighbor 2001:db8:300::2 remote-as 65550
  neighbor 2001:db8:300::2 peer-group upstream6
  !
  address-family ipv4
    neigbor upstream route-map upstream-in in
    neigbor upstream route-map upstream-out out
    neigbor upstream soft-reconfiguration inbound
    neighbor upstream activate
    neigbor upstream6 route-map upstream6-in in
    neigbor upstream6 route-map upstream6-out out
    neigbor upstream6 soft-reconfiguration inbound
    no neighbor upstream6 activate
  exit-address-family
  !
  address-family ipv6
    neigbor upstream route-map upstream-in in
    neigbor upstream route-map upstream-out out
    neigbor upstream soft-reconfiguration inbound
    neigbor upstream6 route-map upstream6-in in
    neigbor upstream6 route-map upstream6-out out
    neigbor upstream6 soft-reconfiguration inbound
    no neighbor upstream activate
    neighbor upstream6 activate
  exit-address-family
```

Draft

Draft

# Chapter 5

# Becoming multi-homed

## 5.1 What is being multi-homed (in terms of BGP)?

In the last chapter, we set up eBGP to one “upstream” provider. But in practice, this is not what we want - we would be dependent on only one connection to the outside world and would be offline if it failed (which is exactly the same as running no BGP at all).

So we need to connect to (at least) one more upstream. And to make our network even more resilient and to drive down cost, we will also connect to an Internet Exchange and set up peering sessions there.

Draft

This increases our network (and BGP) complexity somewhat, but if we plan (and configure) carefully, then this complexity is easily manageable. The purpose of this chapter is to offer you best practices, so that you do not run into an over-complex network set-up later on.

## 5.2 What changes when you become multi-homed?

When you have only one connection to the outside, your world is simple:

- You either have a default route to your upstream and send everything which is not inside your own network to there, or
- you get a full routing table from your upstream, but still have only one connection to the outside to send traffic to.

With two (or more) connections to the outside, it gets more interesting:

- You receive the same prefix(es) via BGP from provider A and provider B; which one do you prefer?
- What criteria does BGP use per default to decide which prefix announcement is “better”?
- Can you influence or override this decision? (yes, you can)
- When to influence best prefix selection (this is what this is called) and when to leave it to the defaults? (this is lifelong learning).

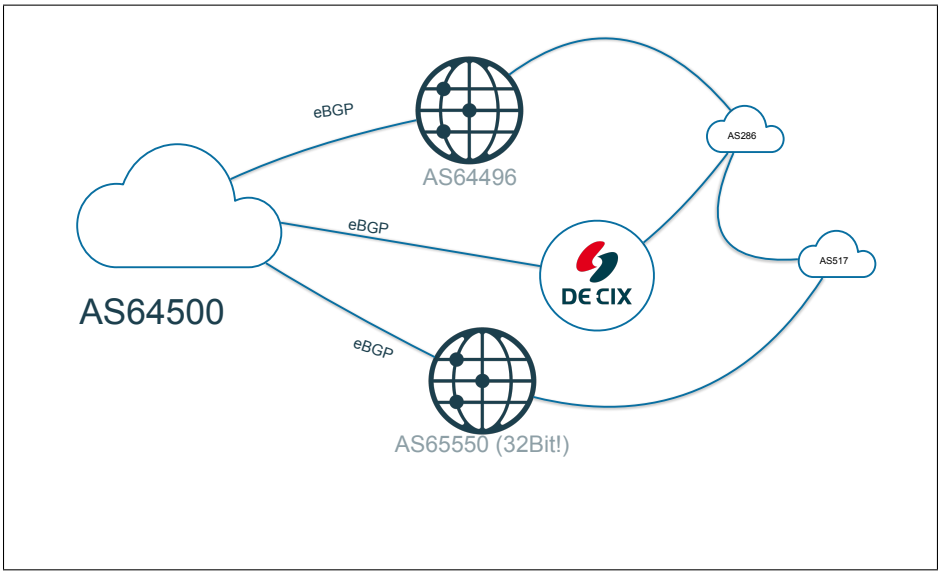



**Figure 5.1:** Example setup with two upstream providers and peering

## 5.3 Configuring multi-homing

### 5.3.1 Network setup

See figure 5.1 for our example network. Your AS64500 is connected two upstreams AS64496 and AS65550. Behind them we have a couple of ASes for distributing prefixes - their AS numbers are not really important. One of them also peers with you.

### 5.3.2 Receiving prefixes

For our somewhat simplified view of the Internet, we receive a handful of prefixes originated by AS517 over various paths.

### 5.3.3 Sending prefixes

You should always only send your own prefixes and those of your customers. Beware - if you do not configure any filtering, your router sends out *all best prefixes* it knows about. So if you connect to two upstream providers, you announce your full routing table to both of them unless you implement some filtering.

## 5.4 Configuration examples

These examples are purposely IPv4, merely to make them easier to read. Adding IPv6 is done according to the same principles as shown in chapter 4.

### 5.4.1 Cisco example

```
router bgp 64500
  neighbor upstream peer-group
```

```
neighbor upstream route-map upstream-in in
neighbor upstream route-map upstream-out out
neighbor upstream route-map send-community both
neighbor upstream route-map soft-reconfiguration inbound
!
neighbor 10.200.2.1 remote-as 64496
neighbor 10.200.2.1 peer-group upstream
!
neighbor 10.230.2.1 remote-as 65550
neighbor 10.230.2.1 peer-group upstream
!
neighbor peering peer-group
neighbor peering route-map peering-in in
neighbor peering route-map peering-out out
neighbor peering route-map send-community both
neighbor peering route-map soft-reconfiguration inbound
!
neighbor 80.81.193.66 remote-as 286
neighbor 80.81.193.66 peer-group peering
```

You see that we reference four route-maps here. For the moment we will just keep them empty; later we will add statements to them:

```
route-map upstream-in permit 100
route-map upstream-out deny 100
route-map peering-in permit 100
route-map peering-out deny 100
```

Draft

The incoming route-maps permit everything, the outgoing route-maps deny everything.

### 5.4.2 Mikrotik example

Mikrotik does not support peer groups, every peer entry needs every parameter.

```
/routing bgp instance
set default as=64500 out-filter=bgp-out router-id=\
  192.168.1.1

add in-filter=upstream-in name=AS64496 out-filter=upstream-out \
    remote-address=10.200.2.1 remote-as=64496
add in-filter=upstream-in name=AS65550 out-filter=upstream-out \
    remote-address=10.230.2.1 remote-as=65550

add in-filter=peering-in name=AS286 out-filter=peering-out \
        remote-address=80.81.193.66 remote-as=286
```

We also have defined filter lists here; for the moment, we will keep them empty:

```
/routing filter
add chain=upstream-in action=accept
add chain=upstream-out action=reject
add chain=peering-in action=accept
add chain=peering-out action=reject
```

### 5.4.3 FRRouting example

(the same as for Cisco IOS)

Draft

# Chapter 6

# BGP best path selection

## 6.1 Motivation

In BGP, a router receives prefix announcements via eBGP. If you are multi-homed or peer, you will receive announcements for one and the same prefix from multiple sources. Out of these multiple announcements, a router has to select *one* announcement as *best*. This best prefix announcement will then be used for routing and also propagated further (for the sake of simplicity, in this chapter we will not look at BGP multi-path where more then one announcement is selected).

Draft

This decision on which prefix announcement is *best* has to be based on the following criteria:

- Only one single path for each prefix is needed (and wanted)
- Decision must be based on attributes (of the BGP announcement)
- Decision must be deterministic (with the same parameters the decision is always the same)

In this chapter, we will uncover one by one the attributes best prefix selection is based on and explain the best path selection algorithm.

## 6.2 Best path selection in contrast to routing

In best path selection we compare prefix announcements of *the same prefix*, while in routing we select a route for a given destination IP address.

10.3.8.0/22 and 10.3.8.0/24 are not the same prefix. They have a different net-mask and 10.3.8.0/24 is *more specific* (smaller). So no matter what the BGP attributes are, *more specific* always wins (against less specific).

## 6.3 Best path selection algorithm

### 6.3.1 Before we start...

Before we start selecting the best path, it must be checked if the BGP announcement is valid. Only paths where the *next hop* is reachable by the router which does the best path selection are considered. If the next hop is not reachable the announcement is discarded.

Also discarded are prefixes where the next hop is not reachable through the same interface as the BGP session is on, as well as announcements, which have the same AS as the router itself has, in the AS path.

The following criteria are processed one by one. Announcements are only compared if the prefixes are identical. If no decision is made, the algorithm continues. If a decision is made, the selected prefix is considered "best" and processed accordingly (installed in the routing table, forwarded by iBGP and/or eBGP).

The algorithm shown is like one implemented by most router vendors. Some implementations vary and either skip criteria or use additional criteria. Also, some routers allow the algorithm to be "tweaked". So, for more details please consult your router vendors documentation.

### 6.3.2 Local Preference

| | |
|---|---|
| Value: | 32-bit integer (0...4294967295) |
| Better: | higher wins |
| Usually set: | at network edge by router receiving prefixes |

Draft

Local Preference is the first evaluated attribute in best path selection and therefore can override all the rest. Usually, it is set at the network edge by the receiving router according to your routing policy: Customer prefixes get a very high value (are preferred) while prefixes received from upstream providers get a low value. The policy of how to set Local Preference can be made as simple or as complex as you want.

It is recommended that you do not set anything to the default Local Preference (so if you see the default value of Local Preference in the BGP table, you know that it has not been explicitly set). Never standadized, but the usual default value for Local Preference is 100 (this is also recommended by [RFC4277]).

Example: If the default Local Preference is 100, you might use 10 for upstream, 1000 for peering, and 10000 for customers.

### 6.3.3 AS path length

| | |
|---|---|
| Value: | an ordered list of AS numbers |
| Better: | shorter wins |
| Usually set: | automatically when re-announcing prefixes via eBGP |
| Mandatory attribute | |

The AS path is built when announcing prefixes via eBGP: The sender's AS number is added to the front of the AS path. So the path grows each time a prefix is forwarded. The length of the AS path (number of AS numbers in the path) is now used as an attribute; a shorter path is considered to be "better".

The AS path length does not give any information about geographical distance, length of fibers, or "quality" of the path (in terms of congestion or similar). It only shows the number of Autonomous Systems traversed by the BGP announcement. The intention of the AS path and how it is built was loop prevention (routers do not accept a prefix announcement if they see their own AS number in the AS path).

AS path length comes into effect when Local Preference is equal, usually when having two upstream providers or receiving the same prefix from multiple peers.

### 6.3.4 Origin type

| | |
|---|---|
| Value: | IGP, EGP, incomplete |
| Better: | IGP over EGP over incomplete |
| Usualy set: | automatically when injecting prefixes into BGP by originating AS |
| Mandatory attribute | |

This is a "historical" attribute with little to no practical value today. Its purpose was to indicate from which source a prefix was put into BGP:

**IGP** - the prefix was generated statically with an BGP *network* statement

**EGP** - the prefix was received via Exterior Gateway Protocol (EGP), which is no longer used

**incomplete** - the prefix was redistributed from another routing protocol (including a static route) into BGP.

Draft

Be aware that the value of this attribute can be overwritten by any BGP speaking router. At RIPE91 James Bensley presented that some transit providers change Origin Type on purpose to attract more traffic[Ben25].

Therefore if you are a stub-AS (having no other ASes as customers) you might consider rewriting Origin Type on all received prefixes to *IGP*.

### 6.3.5 Multi Exit Discriminator (MED)

| | |
|---|---|
| Value: | 32-bit integer (0...4294967295) |
| Better: | lower wins |
| Usually set: | by prefix announcing router, can be overwritten by receiving router |
| Optional attribute | |

This attribute was intended for where two networks have more then one connection to signal for the BGP prefix sending (and traffic receiving) network where it prefers incoming traffic for its prefixes. As Multi Exit Discriminator (MED) can be different for each prefix, you can tell your neighbor where you prefer traffic for which prefix.

On the BGP prefix receiving side (and traffic sending side), the value for this attribute is only considered for best prefix selection between announcements from the same neighbor AS.

If you do not want this and want to keep full control over your outgoing traffic, you can override the received MED with a value you select, but I recommend that, if you do this, you might talk to someone at your neighboring AS first (in peering, it's called peering *partner* for a reason).

In most implementations of BGP, there is a configuration parameter to change the behavior of the best path selection algorithm to "always compare" MED (even between two different

AS neighbors). Using this is *not recommended* unless you know exactly what you are doing. If you turn on "always compare", you *must* override all received MED values. See chapter 8 for a full explanation.

MED is an optional attribute, so it can be missing. Normally, a router treats a missing MED as "best" - but this behavior can be changed by a configuration command to treat a missing MED as "worst".

Outgoing, if you do not intend to use MED with a customer or upstream provider, it's best to simply set all MEDs to zero. This avoids being downrated if the other side has either *always-compare-med* or *missing-as-worst* configured.

### 6.3.6 eBGP vs. iBGP

| | |
|---|---|
| Value: | eBGP or iBGP |
| Better: | eBGP wins |
| Usually set: | by receiving router |

Not really an attribute but the source where the prefix was received from. The intention is to get rid of your traffic as quickly as you can, so if a prefix is received from an external and an internal BGP speaker, prefer the external one.

### 6.3.7 Network exit

| | |
|---|---|
| Value: | distance to exit (in terms of your IGP) |
| Better: | nearest wins |
| Usualy set: | by IGP |

Draft

Again not really an attribute, but rather a parameter the router calculates. If it receives two (or more) announcements of the same prefix via iBGP, it selects the one pointing to the nearest network exit. For this, the metric of the IGP is used.

### 6.3.8 Age of prefix announcement

| | |
|---|---|
| Value: | age of prefix announcements |
| Better: | older wins |
| Usually set: | by router |

This is one of the most tricky parameters for best prefix selection. It is only evaluated if:

- All rules before did not select a best prefix announcement
- Announcements compared are *both external* (received via eBGP)

Usually, this rule is triggered when you receive a prefix from either two different peers or from two different upstream providers. One of them is "best". If this best prefix now disappears, another one becomes "best". If now the original best re-appears, the current one stays best as it is "older" than the now newly learned one.

This rule usually only has a significant impact if two or more connections to different upstreams or exchange points are connected to the same router. Otherwise, if they are connected to two different routers, the prefix announcements are not external on both (one learns it from the other via iBGP) and so the rule does not apply.

### 6.3.9 Tie breakers: Router ID and neighbor IP

| | |
|---|---|
| Value: | IP address |
| Better: | lower wins |
| Usually set: | sending router |

Like stated above, the path selection algorithm *must* select one and only one "best" announcement for each prefix. If the previous attribute comparisons still turn up a tie between two announcements, this last rule (or rather two last rules) make their decision based on the IP addresses of the BGP neighbor. First, the router-id is evaluated and if it is still a tie, the IP address of the BGP neighbor.

These two rules are rarely applied - it is much more likely that the decision is made at an earlier stage.

## 6.4 Advanced topics

### 6.4.1 Multipath

In a *multipath* capable environment, BGP installs not only the best path but multiple paths to the same destination in the IP routing table for load sharing. Enabling BGP multi-path does not affect best path selection - one path is still the best one and advertised to eBGP and iBGP neighbors.

Also, multiple paths are only installed if certain attributes match with the best path: Weight (Cisco), Local Preference, Origin, AS-path length, MED, Neighbor AS must match for eBGP multi-path.

Draft

In some environments (like running BGP in a data center) it might be advisable to do a more relaxed BGP multipath consideration, like accepting two prefix announcements AS paths to the same prefix if *not* everything is equal but simply the AS paths has the same length. In some BGP implementations you can configure this with a statement like in FRRouting `bgp bestpath as-path multipath-relax`.

### 6.4.2 Tweaking the algorithm

If you need this document to understand BGP and best path selection, do not do it.

Draft

# Chapter 7

# BGP Communities

## 7.1 Introduction

In the last chapter, we learned about a number of BGP attributes. This chapter is now about a single attribute called “BGP community”. Why a whole chapter about one attribute?

BGP communities are very important if you build your network for multiple peerings and multiple upstreams. They help you to keep your BGP infrastructure scalable and assist you in implementing your routing policy.

draft

BGP communities were not a part of BGP from the beginning. They were introduced in 1996 in [RFC1997]. This is a great example of how BGP got extended when needed: BGP was designed with future enhancement in mind. Two BGP speaking routers can still “talk” to each other, even if both do not have the same feature-set. When setting up a BGP session, both announce what features they support and if an optional feature marked as “optional” is not supported by both sides, a session still comes up.

So what are BGP communities? They are like a sticker on your suitcase. You can add information to a BGP prefix announcement with them, but they only have the meaning you define. In themselves, they are nothing more than a number stuck on your announcement. Some BGP communities have a pre-defined meaning; we call them “well-known communities”. Their meanings are defined in RFCs.

## 7.2 Original communities

What we now call “original” BGP communities were introduced in 1996 in [RFC1997]. An original BGP community attribute is just a 32-bit number, attached as an optional attribute to a BGP prefix announcement. You can attach as many communities as you want (within reason). The community attribute was defined as optional (does not have to exist) and transitive (is kept when re-announcing the prefix).

Some number ranges were defined as reserved: 0x00000000-0x0000ffff, 0xffff0000-0xffffffff. In these reserved ranges, a number of “well-known” communities were defined (see 7.5).

Although “just a 32-bit number” it was also defined that the first half should encode an AS number (Autonomous System Numbers were only 16bit in these days). To make reading

easier, the notation of communities in routers and documentation was later defined as two 16-bit numbers separated by a colon (like 6695:1200).

The purpose of communities was (and still is) to add a property to prefixes - the "translation" between the numerical value and its meaning should be up to the party defining the community.

## 7.3 Extended communities

With the arrival of 32bit AS-Numbers, it was no longer possible to encode the community-defining AS number in the first half. So in 2006 *extended communities* were defined in [RFC4360].

Unfortunately, the authors of the RFC did not only want to solve the 32bit problem but added also a number of other features:

- A 16-bit type field allowing different community types.
- A "transitive" bit as part of this field to define if a type was transitive (should be forwarded between ASes) or not.
- An IANA-bit, noting if a type was IANA-assigned or experimental.

The initial [RFC4360] still deals with only 16-bit AS numbers, only in [RFC5668] it is finally defined to have a 32-bit AS number as "global administrator" in an extended community. As 16 bits are already used for the type field, that leaves only 16 bit as an argument or value field.

Draft

Extended communities were never really liked by the operators - too confusing and over-engineered. So another type of community was needed.

## 7.4 Large communities

"Large communities" fix the shortcomings of extended communities. Defined in [RFC8092], they go back to the simplistic approach of the original communities: Three simple 32-bit values. The first 32-bit number is defined as the "Global Administrator" - an AS number (32bit!) giving meaning to the two remaining numbers. These two can be seen as simply two values or as one function number plus one numerical parameter.

The notation is three numbers separated by colons (":"). As large communities are kind of new (defined in 2017), you need quite recent software versions for your router. Implementation status is being tracked at `http://largebgpcommunities.net/implementations/`, there you also can see the minimum sofware you need for most implementations.

Mind that *Juniper* routers notate large communities like "large:global administrator:value1:value2" - so you have to prefix them with "large:" otherwise the router reports a syntax error.

Figure 7.1 shows all three community types.

## 7.5 Some well-known communities

So-called well-known communities are defined in RFCs and must be processed by all BGP speaking entities. The following four well-known communities are quite useful.

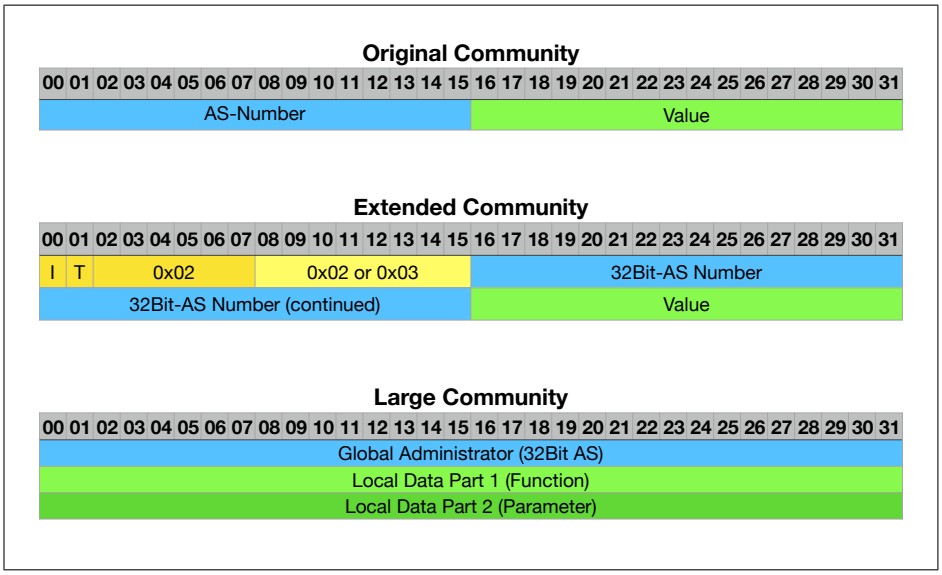



**Figure 7.1:** Original, Extended, and Large BGP Communities

### 7.5.1 NO-EXPORT

BGP prefixes tagged with *NO-EXPORT* must not be advertised via eBGP to other Autonomous Systems. Defined in [RFC1997], this well-known community is useful for:

- Announcing *more specific* prefixes to your eBGP neighbors and making sure they are not announced further. To do this, set NO-EXPORT on your outgoing route-map (or similar).
- Keeping your own specific routes inside your own AS. For this, set NO-EXPORT when injecting your prefixes into BGP and leave it off the network blocks which should be announced externally.

Draft

### 7.5.2 NO-ADVERTISE

This well-known community is even stricter then NO-EXPORT. *NO-ADVERTISE* forbids a router from announcing a prefix to any BGP neighbor; this also includes iBGP.

### 7.5.3 GRACEFUL-SHUTDOWN

Defined in [RFC8326], the receiver of prefixes tagged with this community sets *Local Preference* of these prefixes to zero, so they become the least preferred prefixes in the BGP table. This allows the announcing AS to slowly drain traffic towards these prefixes (via the announcing AS). Once the traffic drops below a certain threshold, the BGP session can be shut down without disrupting any traffic.

### 7.5.4 BLACKHOLE

Defined 2016 in [RFC7999], this well-known community asks the receiving operator to blackhole or block all traffic destined to the prefix with this community attached.

For this to work properly, the receiver should accept longer than usual prefixes (up to /32 in IPv4 and up to /128 in IPv6). Also, the receiver should not propagate these prefixes further, so either the sender also attaches a NO-EXPORT or the receiver should.

The purpose, of course, is to fight DOS or DDOS attacks.

### 7.5.5 ANYCAST

Not yet an RFC, this community is for tagging *anycast* routes. There is no default processing of the community - it is up to an operator to define what to do with so tagged routes.

## 7.6 Examples

### 7.6.1 Setting communities when receiving

Set a community when receiving prefixes from the outside. The community should be added to the existing communities.

#### Cisco

```
route-map customer-in permit 200
  set community 64500:47000 additive
```

The “additive” keyword ensures that the community is added to the list of already existing communities.

Draft

#### FRRouting

Same syntax as Cisco.

#### Mikrotik

Instead of route-maps, Mikrotik has filters. Please check the documentation for details.

```
/routing filter
add chain=customer-in append-bgp-communities=64500:47000
```

Using the “append-bgp-communities” keyword appends the community to the list of already attached communities. To clear the community list first, use “set-bgp-communities”.

### 7.6.2 Setting communities when redistributing

In this case, we do not receive the prefixes via eBGP, but we distribute our own network blocks.

#### Cisco

```
router bgp 64500
  redistribute static route-map static-to-bgp
!
route-map static-to-bgp permit 100
  set community 64500:41000
```

Note that the “additive” keyword is missing here; usually, this would remove all existing communities. But as we inject new prefixes into BGP, there are no communities set yet.

#### FRRouting

Same syntax as Cisco.

#### Mikrotik

Communities can be added directly to static routes, which are then redistributed:

```
/routing  bgp instance
set redistribute-static=yes

/ip route
add dst-address=198.51.100.0/24 gateway=loopback0 bgp-communities=64500:41000
```

### 7.6.3 Removing communities

Removing communities can be done either by removing all communities, specific ones, or ones matching certain criteria. All three cases will be shown.

#### Cisco

Draft

Delete one community (or more than one, explicitly listed):

```
ip community-list standard delete-list permit 64500:10000
!
route-map customer-in permit 100
  set comm-list delete-list delete
```

You can add more communities to *delete-list* but with a standard list in Cisco, you have to explicitly list all you want to have deleted.

With an expanded list, you can delete communities based on a pattern:

```
ip community-list expanded delete-pattern permit 64500:1[0-9][0-9][0-9][0-9]
!
route-map customer-in permit 100
  set comm-list delete-pattern delete
```

Here we delete all communities with AS64500 in the first part and have 10000-19999 in the 2nd part. The community number is treated as a string and a regular expression is used for matching. For details on how regular expressions are built, please check Cisco documentation.

If you want to delete all communities incoming, you can use the same method:

```
ip community-list expanded delete-all permit .*
!
route-map customer-in permit 100
  set comm-list delete-all delete
```

#### FRRouting

Deleting *all* communities in FRRouting is easy:

```
route-map customer-in permit 100
  set community none
```

#### Mikrotik

Mikrotik does not have regular expression matching for communities, nor does it have a delete community command.

### 7.6.4 Scrubbing communities incoming

In this example, we want to do a more sophisticated check and change.

- Customers are allowed to send communities like 64500:4xxxx
- Everything else starting with 64500: has to be removed
- If they do not send any community starting with 64500:4xxxx, per default 64500:47000 is set

#### Cisco

```
ip community-list expanded delete-incoming permit 64500:[0-35-9][0-9]*
!
ip community-list expanded command-community permit 64500:4[0-9][0-9][0-9][0-9]
!
route-map customer-in permit 100
  continue
  set comm-list delete-incoming delete
!
route-map customer-in permit 200
  match community command-community
!
route-map customer-in permit 300
  set community 64500:47000 additive
```

Draft

Explanation:

- In entry 100, we simply remove what we do not want. The “continue” statement makes sure that processing continues even if we have a successful match.
- In entry 200, we only check if a command-community has been set by the customer. Because there is no “continue” statement, the route-map terminates at success. If no command-community was set, the next entry is processed.
- Entry 300 has no match-statement so it always succeeds. A community of 64500:47000 is set and the route-map terminates.

#### FRRouting

Same as Cisco IOS.

**Mikrotik**

You cannot do this on Mikrotik. If you allow your customers to set certain communities, you must filter-check for exactly these and either directly apply some action or let them through and remove anything else. Example:

```
/routing filter
add action=accept bgp-communities=64500:41000 chain=customer-in \
  set-bgp-communities=64500:41000
```

This matches if community 64500:41000 is set, removes all other communities, and sets 64500:41000 exclusively. Also, processing of the filter chain is terminated. This gives you some basic possibilities, but more complex filters are not possible.

## 7.7 What to do with them?

Now that we know what communities are and how to add them to prefixes (and remove them) the question remains: What to do with them?

Before they were introduced, rules for filtering and rules for implementing your routing policy had to be configured on every device. Communities now give you the possibility that you can tag prefixes with a specific community once you receive them, and act later depending on that community.

If we go back to the "sticker on a suitcase" comparison, we can distinguish between two types of communities:

Draft

**Informational Communities:** They add information to a prefix, like where it was received or whom it was received from.

**Action Communities:** They tell one of your routers later what should happen to that prefix, like announce it to your upstreams or announce it only to customers.

In the next sections, a few ideas on how to implement and encode informational and action communities will be given. Although all examples are with original (two times 16bit) communities, they all work with extended and large communities as well. If your router already supports large communities it is recommended to use them - skip the extended communities.

## 7.8 Informational communities

Using communities you can attach additional information to a prefix, like:

- Where it was received (geographically)
  - Continent
  - Country (United Nations M49 code is great for that: `https://unstats.un.org/unsd/methodology/m49/`)
  - City
  - Data center
- On which of your routers it was received

- From whom it was received (like from upstream, customer, or peering)
- In the instance that it is one of your own blocks or prefixes:
  - Which LIR the allocation is from
  - Whether it is an “internal only” route
  - Or if it is a PI prefix allocated to a customer

You can encode all kinds of information. To make the most out of it, publish your communities to your customers so they can make their own BGP routing decisions based on them.

## 7.9 Action communities

With so-called “action communities” you can encode commands into your prefixes your routers should act upon. Examples:

- Announce this prefix to customers (or to peers, or to upstream)
- Announce this prefix in Europe (or North America, or Asia, or ...)
- Announce this prefix with NO-EXPORT set
- Announce this prefix with a longer AS path (repeat your own AS number several times when announcing)
- Change the Local Preference value of this prefix to a higher / lower value (example of a community you can allow your customers to use)

Draft

You can use these yourself at all points where you receive prefixes. This includes eBGP from upstream, peers, or customers but also when you inject your own network blocks into BGP.

Also, you can allow your BGP transit customers to use communities - be careful when you do this to only allow this to customers but block it on upstream or peering connections.

If you start this - first look at what others have done and what communities they offer their customers. Plan carefully. Write down and document what you want to offer. Then think about implementation. As not all routers yet offer large communities, you should simply use what is offered.

## 7.10 Encoding

If you are stuck with original communities, you do not have much space to encode the information you need. The general rule is, for every community, you want others (outside of your own Autonomous System) to either know about or want them to send you actions, you should put your own AS number into the first part. If your own AS is 32bits long, you either have to use extended or large communities.

For internal-only communities, you can use original communities with a private AS number (64512 - 65534) in the first half.

For encoding information, you can, of course, start with 1 in the second part and work up until 65535, but that would not be very effective. The recommendation here is to see that 2nd half like a string of characters and not like a number (caution - this is only valid if your

router can use regular expressions to match communities. Mikrotik for example cannot do that).

Example:

- Use the first digit to encode the type of community. Let's say "1" means this community encodes where you have received the prefix geographically
- In the 2nd digit you can encode the continent, like 1=Europe, 2=Asia, 3=North America, 4=South America...and so on
- You have 3 digits left. You can use the UN M49 code to encode countries using 3-digit numbers: `https://unstats.un.org/unsd/methodology/m49/`

A few examples:

| | |
|---|---|
| Prefix received in Germany | 11276 |
| ...in London | 11826 |
| ...in New York | 13840 |

To match these, many router vendors offer regular expressions. So to match for prefixes received in Europe, in Cisco it would look like:

```
ip community-list expanded geo-eu-any permit 64500:11[0-9][0-9][0-9]
```

To encode actions, if your router offers you regular expression matching, it is also easy. Let's assume your action communities start with "4" in the first digit, and the 2nd digit encodes bitwise on where to logically announce your prefixes, such as:

| | |
|---|---|
| 001 = 1 | announce to customers |
| 010 = 2 | announce to peers |
| 100 = 4 | announce to upstream |
| 110 = 6 | announce to peers and upstream |
| 111 = 7 | announce to all |

Draft

With this, you have three more digits left for your own ideas (for example you can encode specific upstream or Internet Exchanges in them).

Corresponding regular expression matching lists look like:

```
ip community-list expanded to-customers permit 64500:4[1357].*
ip community-list expanded to-peers     permit 64500:4[2367].*
ip community-list expanded to-upstream  permit 64500:4[4567].*
```

And apply them on your outgoing route-map:

```
route-map upstream-out permit 100
  match community to-upstream
```

If your router does not support regular expression like Mikrotik, you still can use this method; only your filters will get a bit longer:

```
/routing filter
add action=accept bgp-communities=64500:41000 chain=to-customers
add action=accept bgp-communities=64500:43000 chain=to-customers
add action=accept bgp-communities=64500:45000 chain=to-customers
add action=accept bgp-communities=64500:47000 chain=to-customers
add action=discard chain=to-customers
```

...you get the idea.

But some things will not work. If you want to use the community digit by digit, you do need regular expression parsing.

Draft

# Chapter 8

# Traffic Engineering

About half an hour after deploying BGP on your router and setting up eBGP sessions to your upstreams (I assume you have more than one) and peers (peering makes sense, remember?) you will be unhappy with the default decisions BGP makes on where to send your traffic. Half an hour is an optimistic estimation.

So you want to influence BGP. Everybody does. This chapter should give you a toolset to do this. It is up to you to decide which of these tools are useful for you. All networks are different. All use cases are different. Sometimes you need a sledgehammer; sometimes you need tweezers.

Draft

As a lot of the things explained here are a matter of personal opinion, the tone of this chapter might be a little "lighter" than the rest of this book. Nevertheless it will stick to the facts and give you a toolset you can use.

## 8.1 Remarks about wording

"*Peer*" in this chapter will be used as a synonym for a peering partner, an upstream provider, or a BGP customer; so, someone you exchange prefixes with via eBGP.

## 8.2 Tools for outgoing traffic - received prefixes

### 8.2.1 BGP Best Path Selection

In chapter 6, the BGP *Best Path Selection* algorithm was explained. Several vendors allow the best path selection to be "tweaked". Sometimes you might need that to solve a specific routing problem; however, always *document* if you are using these features in your internal documentation or otherwise your colleagues might get confused if BGP does not behave like it should.

The following methods may not be complete - check your router's documentation for more.

#### Ignore the AS path length

All router vendors I checked allow the AS path length comparison step to be skipped. Configuration command for this on Cisco and FRRouting is `bgp bestpath as-path ignore` (be aware that this command is not available in all Cisco software versions).

On Mikrotik the command is `add ignore-as-path-len=yes` inside the BGP routing instance.

Usually this does not make a lot of sense - the AS path length is one of the most useful criteria to determine the best path.

#### Do not prefer the older path

Default behavior is that, for two otherwise identical paths, the older (more stable) path is preferred. To switch that off and jump directly to the comparison of the router ID as a last-resort criterion you can use on Cisco and FRRouting `bgp bestpath compare-routerid`.

This can actually make sense if you want to have traffic fall back once an eBGP peer reappears after an outage.

### 8.2.2 Local Preference

This is the very first criterion in Best Path Selection and should be used with care. How and where you use it depends on your routing policy:

Draft

- if you have BGP customers, you *should* set a high local preference value to all prefixes received from them - you do not want to send traffic to your customers over any other path then the one they pay you for. At the same time, you must implement some filtering to prevent your customers from sending you malicious prefixes which do not belong to them.
- in case one of your upstream providers is much more expensive and should only be used as a very last resort, set a very low preference value received from it.
- otherwise, you can use it to prefer peering over upstream or to prefer certain AS paths via specific peers (using rules to change local preference dependent on matching AS numbers in the path).

Local Preference is the strongest criterion available - use it carefully.

### 8.2.3 AS path length

Incoming you cannot change much here - although there are commands to manipulate the as-path incoming (by prepending AS numbers). For all possible use cases of this, usually it's better to manipulate local preference.

Especially you should *never* prepend your AS multiple times if you are single homed! And if you prepend - doing it more than three times will not accomplish anything more.

### 8.2.4 MED

MED was intended to be used so you can signal to your peer with whom you have multiple eBGP sessions at multiple locations as to where you prefer traffic.

So for example if you peer with someone in New York and Frankfurt and announce two prefixes 192.0.2.0/24 and 203.0.113.0/24 on both eBGP sessions, you can adjust MEDs like this:

| Prefix | Location | MED |
|---|---|---|
| 192.0.2.0/24 | Frankfurt | 0 |
| 192.0.2.0/24 | New York | 1000 |
| 203.0.113.0/24 | Frankfurt | 1000 |
| 203.0.113.0/24 | New York | 0 |

If your peer honors your MED (not everybody does) they now send you traffic for 203.0.113.0/24 via your New York peering and traffic for 192.0.2.0/24 via your Frankfurt peering.

Especially nice is that routers allow some internal metric to be announced as MED via eBGP. Cisco example:

```
router bgp 64500
  redistribute ospf 64500 route-map redistribute-filter
!
```

In this case, routes from OSPF (filtered through route map *redistribute-filter*) are announced via BGP with their OSPF metric used as MED.

#### How to treat a missing MED

If no MED is sent (it is an optional attribute), default behavior is to treat a missing med as zero - the best MED value.

If you do not want this, on Cisco you can use the `bgp bestpath med missing-as-worst` command, which changes the default behavior so that a missing MED is treated as the worst possible MED value of 4294967294.

Draft

#### Always compare

One of the criticisms of BGP is that Local Preference is often seen as “too early” in the Best Path Selection Algorithm. Operators would like to have something similar *after* the AS Path Length comparison.

MED *is* evaluated at the right location, but standard behavior is to only compare MEDs if the neighbor AS is the same. To overcome this, router vendors created a configuration option like Ciscos `bgp always-compare-med`. If this is set, MEDs are also compared between different next-hop ASes.

**Important:** *If you turn this feature on, you **must** adjust the MED for **all** received prefixes.*

This would give the MED a complete new meaning - instead of using it to signal a metric to your neighbors you would use it as a kind of 2nd class local preference. This is completely valid *if you do it right*. Meaning:

- Turn it on at *all* of your routers, even the ones only speaking iBGP.
- Set a new MED incoming on *all* eBGP sessions.
- Do *not* do this if you are a large ISP and have peers with geographically diverse multiple connections. In this case use MED like it was defined.

The new adjusted best path selection then looks like:

1. Local Preference - highest wins
2. AS path length - shortest wins
3. self-set MED - lowest wins
4. (the rest of best path selection stays as it was)

## 8.3 Tools for incoming traffic - announced prefixes

### 8.3.1 AS path length

The length of the AS path is step two after Local Preference - a shorter path wins.

Unfortunately, we cannot shorten the path we send out to our peers - we can only make it longer. To conform with Internet standards, you have to insert your own AS number, but you can insert it multiple times (best practice is not to overdo this - inserting an AS more than three times is discouraged).

When now somewhere "further out" (in terms of Internet), an AS receives both prefix announcements *and* has the same Local Preference set for these, it will prefer the shorter AS path. See figure 8.1 for an example.

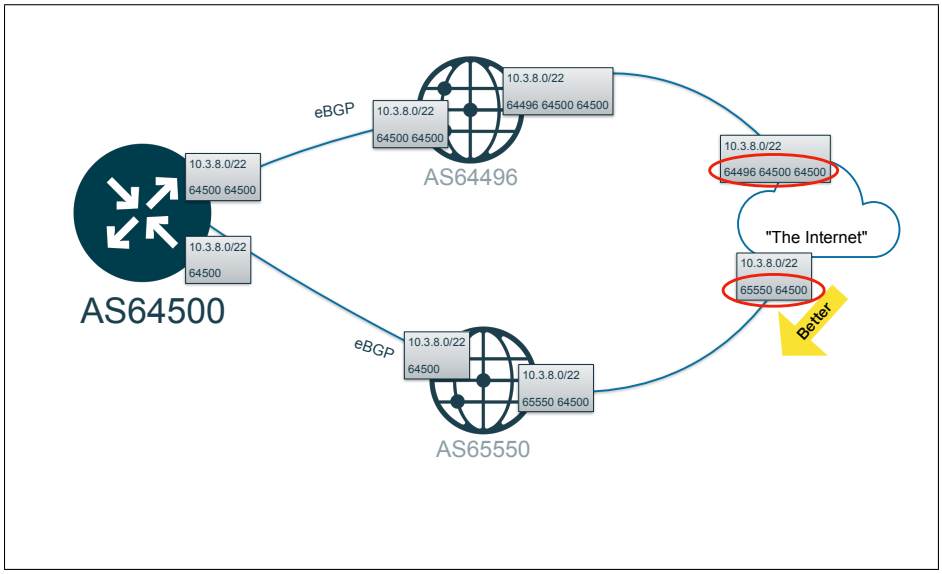



**Figure 8.1:** Effect of creating a longer AS Path

In practice, however, as providers do set a higher local preference to their customers, this has limited use. Local Preference beats AS path length every time.

### 8.3.2 Announcing *more specific* prefixes

Even higher than the BGP Best Path Selection in the hierarchy of routing is the general routing rule that a *more specific* route wins against a less specific one. Example:

- 198.51.100.0/24 contains IPv4 addresses for hosts from 198.51.100.1 to 198.51.100.254

- 198.51.100.0/25 is smaller or *more specific*; it contains only IPv4 addresses for hosts from 198.51.100.1 to 198.51.100.127

So if a router wants to look up a route to host 198.51.100.1, the more specific route (in this case the /25) wins (before any BGP best prefix selection).

Special care must be taken if you are using this method. Keep in mind the following:

- Routes for IPv4 networks smaller then /24 and IPv6 networks smaller then /48 are usually discarded by most providers.
- Peers will not be happy if you announce lots of de-aggregated prefixes to them. It inflates the global routing table unnecessarily.

How to overcome these restrictions? Some ideas:

- Add a *NO-EXPORT* community to your more-specific prefixes. With that community attached, your peer still receives them but does not propagate them further. Also *let your peers know that* what you are doing helps.
- If you really want to announce small prefixes (smaller then /24 on IPv4 and /48 on IPv6) *talk* to your peers. They might understand and adjust their filters. But also set *NO-EXPORT* to prevent further propagation.

Draft

Draft

# Chapter 9

# BGP Security

## 9.1 Introduction

BGP itself does not have much security mechanisms built in. You can secure your sessions with a MD5 hashed password, but that's about all.

That does not mean that your network's lifeline has to be insecure. There are methods you can use to protect yourself and also to protect the Internet from harm. This chapter shows these methods, and it is recommended that you at least implement some of them.

Draft

The *robustness principle* as formulated by Jon Postel does not really apply to BGP announcements:

> "Be conservative in what you send, be liberal in what you accept"

(It does apply to the BGP protocol itself, though).

You must check and filter what others send you in terms of prefixes, and you must also be strict in what prefixes you send to others. Especially for the latter, it is often helpful to over-provision filters (like filter using communities and also filter out IPv4 networks smaller then /24).

[RFC7454] is the reference document for BGP routing security. This chapter will heavily quote from it. Please read the original for further reference.

## 9.2 Automation

A lot of the measures explained here work on rules and data that might change over time. So it is a good idea to build some automation:

- Automate updating rules on all of your routers at once
- Automate checking rules and then update your implementation of these rules.

What you use for automation is up to you - it depends on your IT and network management environment.

## 9.3 Simple measures

### 9.3.1 Maximum prefixes

This parameter is configured for each eBGP session and is the simplest and easiest security measure you can use. Unfortunately, many stop here. Please do not.

Maximum prefix defines a limit for the number of prefixes you accept from an eBGP peer. If the peer sends more, the eBGP session is shut down. Usually, routers keep the session down for some time, then it is automatically re-enabled. If the peer still sends more prefixes than allowed, it is shut down again.

For selecting this limit, the following rules of thumb can be used:

- For sessions to *peers*, the limit should be less than the total number of prefixes in the Internet. Set it at least to ten times the normal number of prefixes your peer announces. This protects you against your peer announcing the full routing table to you, but still allows normal growth. Check and adjust from time to time (or even better: Automate this).
- For sessions to your *upstream* provider, you must, of course, set the limit higher than the total number of prefixes in the Internet. It must be high enough to accommodate normal growth, so either set it *very* high or check and adjust it regularly. Otherwise, there can be surprising session shutdowns. This protects you against gross misconfigurations at your upstream provider (like sending you a lot of de-aggregated prefixes).

Draft

#### Maximum prefixes for announcements

Some BGP implementations (notably *FRRouting*) have implemented a maximum prefix parameter for announcements: This protects your peers from you accidentally flooding. Simply set it to the number of prefixes you normally announce, add some leeway for growth, and use automation to keep track of the value you set.

In case you make a mistake with your filtering this guarantees to not send more than the configured number of prefixes. However, *which* prefixes you send you cannot influence, this limits the usefulness of this feature.

Example for FRRouting:

```
router bgp 64500
...
neighbor 10.96.1.1 maximum-prefix-out 85
```

This allows you to announce up to 85 prefixes to your neighbor.

## 9.4 Protecting your router

A complete discussion about how to protect a router is outside the scope of this document; here we will focus on BGP and how to protect yourself.

BGP itself has some protection mechanisms - BGP packets from IP addresses not configured are discarded. However, in some routers, this happens in the control-plane and consumes CPU cycles. A possible attack vector could exploit this by simply trying to overload your

CPU. A countermeasure can be to use an explicit filter to disallow everything to port 179 from sources which could never be a BGP peer.

## 9.5 Protecting your BGP sessions

### 9.5.1 Motivation

The idea of these measures is to protect your TCP-based BGP sessions against attacks. Keep in mind, these TCP sessions are long-living (speaking of weeks and months), so an attacker can take its time to try to destroy a BGP session by sending crafted packets.

### 9.5.2 MD5 session password

The easiest countermeasure against TCP based attacks on BGP sessions is to use an MD5 protection as described in [RFC2385]. When implementing this, keep in mind to also implement some key (password) handling procedures (just imagine your router has to be replaced and you have to re-create all eBGP configurations).

Example for setting an MD5 password on Cisco:

```
router bgp 64500
...
neighbor 10.96.1.1 password mysecretpassword
```

Example for Mikrotik:

```
add name=AS64496 remote-as=64496 \
  remote-address=10.96.1.1 tcp-md5-key=mysecretpassword
```

Draft

### 9.5.3 TCP Authentication Option

MD5 which is widely used is considered to be insecure and deprecated. To replace it [RFC5925] defines a mechanism called *TCP Authentication Option*, please read the RFC for details. In short, it uses stronger codes to protect your session.

### 9.5.4 TTL security

Instead, relying on the Time To Live () value of incoming TCP packets is easier to handle and to implement. [RFC5082] describes how setting the TTL value of packets when sending to 255, and checking that value when receiving, makes it an impossible-to-spoof security measure. As the TTL is decreased by every hop, when you receive a packet with TTL 255, it *must* have been sent by a directly adjacent node.

This feature must be set on both ends to work - if you set it on one end only, one side sends IP packets with a TTL of 1, and the other with a TTL of 255, and a session cannot be established.

On Cisco, you configure TTL security such as

```
router bgp 64500
...
neighbor 10.96.1.1 ttl-security hops 1
```

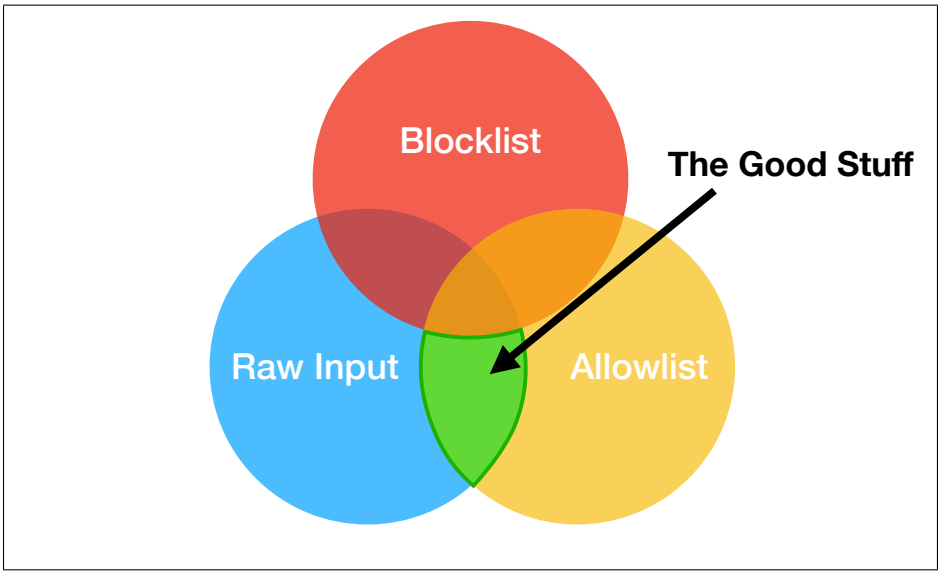



**Figure 9.1:** BGP prefix lists as intersecting sets according to [Sni17]

On Mikrotik, you do not configure how many maximum hops a peer can be away, but the TTL value, which is 255 for directly adjacent peers (this is also the default value):

```
add in-filter=upstream-in name=AS64496 out-filter=upstream-out \
  remote-address=10.96.1.1 remote-as=64496 ttl=255
```

Draft

## 9.6 BGP filtering

A "raw" BGP full feed (the so-called "global routing table") contains a lot of junk you do not want in your routers. In the best case, it contains prefixes to non-routable networks; in the worst case, it can break your internal routing.

There are a number of measures to filter a full BGP feed which will be explained in the following sections.

In general, we have three sources of information to fill your BGP table:

- the raw input you receive from your peers
- one or more *block lists* where you define what you not want from that specific peer or from all peers
- an *allow list* or whitelist, where you define what you allow from that peer

Job Snijders defined that in [Sni17] as intersecting sets, see Figure 9.1.

## 9.7 Inbound: Prefix filtering

Prefix filters work on received prefixes only. Some of them are easy to implement, while some require more effort. Sometimes the shortest solution to implement is not the best, it often is better to have more lines of config to increase readability.

When implementing, it's helpful to write down your rules in pseudo-code to find out the best order of statements.

### 9.7.1 Filtering against prefix sizes

Normally (there are exceptions), prefixes are announced in certain minimum and maximum sizes in the global Internet routing table. Currently, they are:

**IPv4, minimum size** is /24. No smaller networks should be announced. Possible exceptions: Blackholing, or an announcement in combination with a NO-EXPORT community set from customers.

**IPv6, minimum size** is /48. Same exceptions as within IPv4.

**IPv4, maximum size** is a /8. Larger networks are not announced. Depending on your set-up, you might want to accept the Default Route 0.0.0.0/0 from one of your upstreams.

**IPv6, maximum size** is currently a /16 (please take this with a grain of salt, this number might already have changed). The remark about accepting a Default Route ::0 is the same.

#### Implementation example: Cisco

As often, there is more than one way to implement this:

```
ip prefix-list ipv4-small-networks permit 0.0.0.0/0 ge 25 le 32
ip prefix-list ipv4-large-networks permit 0.0.0.0/0 ge 1 le 7
!
route-map upstream-in deny 50
  match ip address prefix-list ipv4-small-networks
!
route-map upstream-in deny 55
  match ip address prefix-list ipv4-large-networks
```

Draft

Explanation:

- We first define two prefix-lists, matching all prefixes ("0.0.0.0/0" here means all prefixes, not the default route) with a length greater or equal to 25 "ge 25" and less or equal to 32 "le 32" (the "le 32" is not really necessary and may be removed by the router's command parser).
- We do the same for too large networks (from length 1 to 7).
- We then insert a deny-rule into our upstream-in route-map with the prefix-lists we defined as match-part. Deny-rule means that when a prefix matches all match-statements, the whole route-map terminates and the prefix is *not* let through.

Alternative (shorter, more elegant) implementation with just one prefix-list:

```
ip prefix-list ipv4-unwanted permit 0.0.0.0/0 ge 25 le 32
ip prefix-list ipv4-unwanted permit 0.0.0.0/0 ge 1 le 7
!
route-map upstream-in deny 50
  match ip address prefix-list ipv4-unwanted
```

This has the additional advantage that you can add even more prefixes to it that you not want (see below). This shortens your configuration but might also decrease readability.

And for IPv6:

```
ipv6 prefix-list ipv6-unwanted permit ::/0 ge 49 le 128
ipv6 prefix-list ipv6-unwanted permit ::/0 ge 0 le 16
!
route-map upstream-in deny 45
  match ipv6 address prefix-list ipv6-unwanted
```

**Implementation example: Mikrotik**

Mikrotik works with filter-lists instead of route maps, but for easier readability you can use sub-filters:

```
/routing filter
add action=jump chain=upstream-in jump-target=ipv4-size
...
add action=reject chain=ipv4-size prefix-length=0-7
add action=reject chain=ipv4-size prefix-length=25-32
```

### 9.7.2 Filtering against RPKI-Invalid prefixes

Draft

RPKI allows the holder of a resource (an IPv4/IPv6 prefix) to cryptographically prove that it is really the holder and allows via defining ROAs how that prefix can be announced via BGP.

RPKI is defined in [RFC6480], ROAs are defined in [RFC6482]. For more information about RPKI see `https://rpki.readthedocs.io/`.

A ROA is a triple containing the following values:

- The prefix itself (network plus prefix length)
- An AS which is allowed to originated that prefix
- A maximum prefix length for which BGP announcements are allowed, this can be the same as the prefix length of the network (in this case the announcement of more specifics for this prefix is not allowed).

To use RPKI and ROAs you need a host running a RPKI validator. This validator fetches resource certificates and Route Origin Authorizations (ROAs) from RIRs, checks their signatures and is then available being contacted by routers using RPKI-RTR protocol (defined in [RFC8210]). You can (and should) have more than one validator.

Routers simply receive a list of validated prefixes, their allowed originating AS number and a range of allowed networks masks. This can be used to check prefixes received via eBGP. Result of this check is one of three possible values:

**Valid:** A ROA for this prefix exists, the originating AS matches and the announced prefix length is covered.

**Invalid:** A ROA for this prefix exists, but either it is for a different originating AS number or the prefix length is *not* covered (too specific).

**Unknown:** For this prefix a ROA does not exist. This result is also returned if no validator is reachable by the router.

Recommendation for your filtering rules:

- *Accept* prefixes with *valid* or *unknown* result.
- *Deny* prefixes with result *invalid*.

#### Implementation example: FRRouting

You have to use a route-map to filter out *RPKI invalid* prefixes (only relevant config statements are shown):

```
rpki
  rpki cache a.b.c.d 3323 preference 1
  exit
!
router bgp 64500
 neighbor upstream peer-group
 neighbor upstream-v6 peer-group
 address-family ipv4 unicast
  neighbor upstream route-map upstream-in in
 exit-address-family
 address-family ipv6 unicast
  neighbor upstream-v6 route-map upstream-v6-in in
 exit-address-family
!
route-map upstream-in deny 50
  match rpki invalid
!
route-map upstream-v6-in deny 50
  match rpki invalid
```

Draft

#### Cisco

When RPKI is active and a connection to a validator is established, Cisco filters out *invalids* by default. To prevent this, you have to either disable it completely or allow invalid announcements to become “best” explicitly (see commented out commands below):

```
router bgp 64500
  bgp rpki server tcp a.b.c.d port 3323 refresh 300
  address-family IPv4
   ! bgp bestpath prefix-validate allow-invalid
   ! bgp bestpath prefix-validate disable
  exit-address-family
  address-family IPv6
   ! bgp bgp bestpath prefix-validate allow-invalid
   ! bgp bestpath prefix-validate disable
  exit-address-family
```

Route-map statements are the same as for FRRouting - but you only need them if you have “*bgp bestpath prefix-validate allow-invalid*” configured.

### 9.7.3 Filtering against non-routable prefixes

When IPv4 was created, the inventors reserved certain part of the address space for specific purposes. These were the times of class-A,B,C networks (if anybody still mentions them - the concept was abolished in 1993 in some RFCs starting with [RFC1517]).

The following IPv4 space is still considered to be not routable and should never be announced via BGP:

**Private IPv4 space** as defined in [RFC1918]. Networks *10.0.0.0/8*, *172.16.0.0/12* and *192.168.0.0/16* are reserved for private use and should never be announced.

**IPv4 networks reserved for documentation purposes** defined in [RFC5737]. These three networks are reserved and should not be routed (but you might see them in this document as example networks).

**Reserved for multicast:** The address block *224.0.0.0/4* was reserved for multicast and cannot be used for anything else. Do not accept announcements out of it via BGP.

**So-called "Class-E":** The network block *240.0.0.0/4* was always reserved "for future use" which never came. Today this range is considered to be not usable and therefore should not be accepted via BGP.

**More can be found** at this IANA website: `https://www.iana.org/assignments/iana-ipv4-special-regis` `iana-ipv4-special-registry.xhtml`. Everything with "Globally Reachable False" should be filtered out.

Draft

In IPv6, there is a similar list at IANA `http://www.iana.org/assignments/ipv6-address-space`. However, for IPv6 it is easier to positive-filter for *2000::/3*, as this is the only block where unicast address assignments were made from. Currently. You might check frequently if other blocks have been added. It is strongly recommended that you automate this task.

#### Implementation example: Cisco

For IPv4, you can simply add all unwanted prefixes to the list we defined in the previous section:

```
ip prefix-list ipv4-unwanted permit 192.168.0.0/16 le 32
ip prefix-list ipv4-unwanted permit 172.16.0.0/12 le 32
ip prefix-list ipv4-unwanted permit 10.0.0.0/8 le 32
...
```

#### Implementation example: Mikrotik

You can add this to your existing filter or you can create a sub-filter for better readability:

```
/routing filter
add action=reject chain=ipv4-unwanted prefix=192.168.0.0/16 prefix-length=16-32
add action=reject chain=ipv4-unwanted prefix=172.16.0.0/12 prefix-length=12-32
add action=reject chain=ipv4-unwanted prefix=10.0.0.0/8 prefix-length=8-32
...
```

### 9.7.4 More unwanted prefixes

In the last section, we covered non-routable prefixes. But these are not the only ones you want to block.

#### IXP LAN Prefixes

When you are connected to an Internet Exchange Point, you have an interface configured with an IP address and netmask of that IXP. If, now, someone else announces the same network (or worse: a more specific sub-network) via BGP and you accept this announcement, your router might prefer this announcement over the one of its own interface (especially if the announcement *is* more specific).

So it is strongly recommended that you block BGP announcements of all IXP LANs you are connected to.

For DE-CIX Frankfurt, a filter for Cisco would look like:

```
ip prefix-list ipv4-unwanted permit 80.81.192.0/21 le 32
ipv6 prefix-list ipv6-unwanted permit 2001:7f8::/64 le 128
```

For Mikrotik:

```
/routing filter
add action=reject chain=ipv4-unwanted prefix=80.81.192.0/21 prefix-length=21-32
add action=reject chain=ipv6-unwanted prefix=2001:7f8::/64 prefix-length=64-128
```

Draft

#### Your own prefixes

You also should protect yourself against hijacking of your own prefixes and against accepting announcement of your customers' prefixes.

Just imagine you accept an announcement of a more-specific subnet of the network you are using for your office…or for your routers!

Commands to protect against this are the same - simply add your prefixes to the *unwanted* lists you have already defined.

#### Your customers' prefixes

If your customers are single-homed to you, you should use the same measures as with your own prefixes.

In case your customers are multi-homed however, you should accept their prefixes, but perhaps not more-specifics of their prefixes.

## 9.8 Route flap dampening

### 9.8.1 Motivation and history

BGP is a protocol using incremental updates. Routes are announced and withdrawn if they are no longer valid. If this announce - withdraw happens too fast for prefixes we speak about *flapping* routes. This consumes CPU cycles on all BGP speaking routers as each time a prefix flaps the BGP table (and also the routing table) needs to be updated.

So in 1998 [RFC2439] was published to introduce *route flap dampening* - which means that routes which flap too often are suppressed and only re-used once they become stable again. The original values for this dampening have been proven too aggressive, so in 2006 in [RIPE378] it was recommended to disable dampening completely.

### 9.8.2 How does it work?

Dampening works by increasing a penalty value each time a route flaps, which is then decreased over time. Once a configurable threshold has been reached, the route is suppressed. If over time the penalty is lower then another (lower) threshold, the route is no longer suppressed and re-used.

### 9.8.3 Current recommendation

More recent studies show that by adjusting the dampening parameters to be less aggressive route flap dampening can be made useful again. The documents [RFC7196] and [RIPE580] give recommendations:

- The "penalty" when starting suppression should be between 6000 (more aggressive) and 12000 (less aggressive). Default on most routers is 2000 (way too aggressive).
- To avoid "surprises" for operators, the default should not be changed.

Draft

#### Configuration examples

On Cisco you turn on flap dampening for each address family separately and you use a route-map to adjust parameters:

```
router bgp 64501
  address-family ipv4
    bgp dampening route-map set-dampening-parameters
  exit-address-family
  address-family ipv6
    bgp dampening route-map set-dampening-parameters
  exit-address-family
!
route-map set-dampening-parameters permit 100
  set dampening 15 750 6000 60
!
```

Parameters set in the route-map are explained below.

FRRouting is similar, except you do not need a route-map to set the parameters (this has the slight disadvantage that you cannot tune dampening individually through match statements in the route-map). Also in the current release of FRRouting BGP dampening is only working for IPv4 unicast and multicast:

```
router bgp 64501
  address-family ipv4 unicast
    bgp dampening 15 750 6000 60
  exit-address-family
```

**Parameters**

Both Cisco and FRRouting allow you to set the following parameters:

```
... dampening <half-life> <reuse-threshold> <suppress-threshold> <max-suppress>
```

To understand them, you need to know that a *penalty* is calculated for each route. The penalty is increased every time a route flaps.

These are the configurable parameters for the Cisco and FRRouting implementation:

**<half-life>** is the time value in minutes in which the penalty is reduced by half.

**<reuse-threshold>** if the penalty gets lower than this value, the route becomes valid (unsuppressed) again.

**<suppress-threshold>** if the penalty is higher than this, the route is started being suppressed.

**<max-suppress>** is the maximum time (in minutes) a stable (non-flapping) route stays suppressed.

The following parameters are calculated:

**<max-suppress-penalty>** calculated value: $< reuse\text{–}limit > * 2^{<max-suppress>/<half-life>}$

When setting the parameters you have to keep in mind:

- *<max-suppress-penalty>* must be larger than *<suppress-threshold>*, otherwise a route never gets suppressed.
- the shorter you choose *<half-life>* the faster a route gets unsuppressed.
- Cisco has a *<maximum-allowed-penalty>* of 20000, your *<max-suppress-penalty>* must be kept lower than this.
- choose *<supress-threshold>* large enough that a small number of flaps are still allowed, otherwise you risk that too many parts of the Internet become unreachable for you.
- [RFC7196] gives recommendations how to set these values.
- Cisco simply turns off BGP dampening if you enter invalid values in the route-map (it gives a warning in the logging).

This paper explicitly gives no recommendation on how to set these values. Please read the RFCs and make your own decision depending on your operational needs.

## 9.9 Inbound: Next-hop filtering

When peering on an IXP LAN, your BGP peers can send you any IP address of this LAN as a next hop (not only their own). This is ok when peering with a route server (who does that as part of its functionality), but a “standard” peer should only send its own IP address as next hop (there might be an exception when a special address is used for signaling Blackholing).

So you can set the following in your route-map for direct peers (in this example you peer with AS64496 on 80.81.192.22 and with AS64497 on 80.81.192.43)

```
ip as-path access-list 1 permit ^64496_
ip as-path access-list 1 permit ^64497_
!
access-list 1 permit 80.81.192.22
access-list 2 permit 80.81.192.43
!
route-map direct-peer-in permit 10
  match ip as-path 1
  match ip next-hop 1
  continue 500
!
route-map direct-peer-in permit 20
  match ip as-path 2
  match ip next-hop 2
  continue 500
!
route-map direct-peer-in deny 100
!
route-map direct-peer-in permit 500
! rest of processing starts here
```

AS-Path and next-hop IP both have to match; if they do, the route-map jumps to entry 500 for further processing. If none of the route-map entries 10 or 20 matches, entry 100 stops processing with a "deny" result. In this way you can have one route-map for all direct peers. Of course, you can also use a separate route-map for each peer.

Draft

## 9.10 Inbound: AS-Path based filtering

Even if a prefix is completely legit, it is still advisable to also check the AS path.

### 9.10.1 Private AS numbers

Like prefixes, there are AS numbers reserved which should never be seen in the global routing table.

So-called private ASes are like private IP addresses; they may be used within a provider's network, but should never be seen in the global routing table. They are defined in [RFC6996]:

- 16-Bit ASes: 64512 - 65534
- 32-Bit ASes: 4200000000 - 4294967294

### 9.10.2 Special AS numbers

Also, some AS numbers are set aside for documentation purposes. [RFC5398] lists them:

- 16-Bit ASes: 64496 - 64511
- 32-Bit ASes: 65536 - 65551

At IANA, you can check which other AS numbers are reserved; they also should not be in an AS path: `https://www.iana.org/assignments/as-numbers/as-numbers.xhtml`

### 9.10.3 Implementation example: Cisco

Cisco supports regular expressions for parsing AS paths; in this case (filtering against unwanted ASes in a path) this is not really helpful. You need to build a regular expression list so that all unwanted ASes are matched:

```
! match 64496 - 131071
! match 64496 - 64499
ip as-path access-list 100 permit _6449[6-9]_
! match 64500 - 64999
ip as-path access-list 100 permit _64[5-9][0-9][0-9]_
! match 65000 - 69999
ip as-path access-list 100 permit _6[5-9][0-9][0-9][0-9]_
...
route-map upstream-in deny 40
  match as-path 100
```

Numerical ranges would be more helpful here, but we can only use what router vendors implement.

When building regular expressions for filtering, keep in mind that your co-workers also need to understand them. Most of the time, it's better to add more lines to your filter list and keep your regular expressions simple.

### 9.10.4 Inbound from customers: AS filtering

Draft

ISPs should only accept prefixes from customers where every AS in the path either belongs directly to that customer or to a sub-customer. To scale this, use automation. This protects you and the whole Internet community from your customers' hijacking prefixes which do not belong to them (by faking an AS path).

## 9.11 Inbound and outbound: BGP community handling

We covered BGP communities in chapter 7. A lot of Autonomous Systems use them. And attach them to prefixes. Which means that the prefixes you receive via eBGP might have a lot of (mostly) useless communities attached to them. However, even if they are useless to you, your transit customers might need or want them.

So it is recommended that you leave any BGP communities untouched, *except* if they have your AS number in the high order part (this applies to all types of BGP communities: original, extended, and large).

You should only allow BGP Communities with your AS number in them in via eBGP:

- If you allow customers (or peers) to use them to send you commands and
- on BGP connections to customers (or peers)

All other BGP communities with your AS in them should be removed inbound.

Outbound, you should only send out what you have documented so your customers or peers (or anybody else receiving your prefixes) can make use of it. y You should especially remove any communities you have set which have a private AS number in the higher part.

## 9.12 Outbound: Sending prefixes

The best way of not polluting the global routing table with bad prefixes would be if every provider behaved according to some code of conduct.

This is the goal of the MANRS initiative - MANRS stands for Mutually Agreed Norms for Routing Security. This is a set of rules an ISP (or IXP) can sign and so express its intent to keep the routing table (and the Internet) clean.

Details can be found at `manrs.org`, but basically to be compliant you need to agree to:

- Prevent propagation of incorrect routing information.
- Prevent traffic with spoofed source IP addresses (outside of the scope of this paper).
- Facilitate global operational communication and coordination between network operators (= "talk to each other and listen when someone talks to you").
- Facilitate validation of routing information on a global scale (we covered this in 9.7).

### 9.12.1 Prevent propagation of incorrect routing information

The first step in not propagating incorrect prefixes via BGP is not to accept them. What is not in your prefix table cannot be propagated. So the measures described in 9.7 should be applied.

Draft

Also, you can apply the same filter rules you already configured for incoming prefixes also in outgoing direction. Usually this would not be needed, but just in case a filter is removed, disabled, or misconfigured on the incoming side, the outgoing filter would still prevent bad prefixes from being distributed.

Particularly, you need to:

- Not originate any prefixes that are not yours (or your customers).
- Properly aggregate all prefixes you announce.
- Do not announce prefixes with private or reserved AS numbers in the path.
- Make sure the AS path of prefixes you re-announce is clean. This should also include all prefixes and announcements of customers - make sure they originate only their own prefixes.

## 9.13 [RFC9234] on preventing accidential floods

### 9.13.1 Roles

This RFC defines eBGP - *roles* which can be applied to each peering session. Note that the own (local) role is configured only (which might be different on each session, like a the own AS can be a *Customer* to one peer and a *Provider* to another peer). The following roles are defined, the descriptions reflect that the local role is set:

- *Provider*: My AS is a transit provider for the remote AS, I may announce any prefix to it.

- *Customer*: My AS is a transit customer of the remote AS, I only announce prefixes learned from my own *Customer*s or my own prefixes.
- *Peer*: My AS and the remote AS are peers, I announce only prefixes learned from my own *Customer*s or my own prefixes.
- *Route Server*: My AS is a route server, I may announce any prefix to a remote *Route Server Client*
- *Route Server Client*: My AS is a route server client, the remote AS is a *Route Server*. I only announces prefixes learned from my own *Customer*s or my own prefixes.

These roles are applied to eBGP sessions. Rhe eBGP session will not be established unless the pairing of roles is valid. The following pairs of roles are considered to be valid:

- Provider ⟷ Customer
- Route Server ⟷ Route Server Client
- Peer ⟷ Peer

If you enable *strict* checking, the session will also not come up if the remote side has set no role, *loose* check still prevents invalid pairs, but the session will come up if the remote side has not configured any role.

### 9.13.2 *Only to customer* (OTC) Attribute

Draft

This is an optional (it does not have to be there) and transitive (it is forwarded via eBGP to other ASes) BGP Path Attribute of a BGP prefix. Purpose is to enforce a “common sense” BGP announcement policy: Announce BGP prefixes received from peers, transit or route servers only to customers.

The OTC attribute is set according to the following rules:

- if a route is received *without* OTC from a peer, transit provider or route server, OTC is added with the AS number of that peer or transit provider or route server.
- if a route is advertised and OTC is not present, add OTC with your local AS number.

Once the OTC attribute is set, it must remain unchanged.

For all routes with OTC present, the checking occurs using these rules. Routes should be dropped (considered being route leaks), if

- received from a customer or route server client. Rationale: OTC means ‘only to customers’ and in this case you are a transit provider (or a route server).
- received from a peer and the AS number set in OTC is different from the peers AS number (that means that some other AS than your peer has set OTC).

Also, if OTC is set, you must not advertise the route to any transit provider, peers or route servers.

## 9.14 Blackholing

Blackholing means that traffic to specific targets within the network operators infrastructure is blocked outside the network operators network.

This chapter should give you an idea how you can implement a blackholing triggering infrastructure. First we talk about a mechanism which allows you to send blackhole requests to your upstream providers and peers, second we show how you can offer a blackhole service to your BGP customers.

### 9.14.1 Theory

If DOS or DDOS packets get dropped as early as possible, the target system is no longer reachable but there is no collateral damage or at least collateral damage is kept to a minimum. Goal is to drop packets outside the attacked network, or if this is not possible at the earliest possible stage within the attacked network.

### 9.14.2 Implementation

#### Design Principles

Being under attack is stressful. So once you have determined the target of any attack and know which IP address(es) should be blackholed, only a single action should be necessary to start (and stop) the blackholing. Also some "success monitoring" should be possible.

#### Option 1: Use one of your existing routers for signaling

You need to have something in place already to import prefixes into BGP. The idea is to use the same mechanism to import prefixes to be blackholed into BGP. In general there are two ways importing prefixes into BGP in Cisco IOS:

- using a network statement
- using "redistribute" with a route-map for filtering

In case you are using network statements, you must also use a route-map to set the necessary parameters like the BLACKHOLE community and you need a corresponding route in your routers routing table.

Configuration example for Cisco IOS:

```
ip route 192.0.2.1 255.255.255.255 Null0
!
route-map set-blackholing permit 1000
  set community 65535:666 additive
!
router bgp 64500
  network 192.0.2.1 mask 255.255.255.255 route-map set-blackholing
```

The route-map can be pre-configured any time, but the network statement in BGP and the static ip route must both be entered to activate blackholing and both removed to disable it (actually it would be enough to remove one of the statements for deactivation, but after some time your router config would look very messy).

If you redistribute static routes into BGP using some sort of filtering you can easily extend this to also accommodate blackholing. Again you need a static route in your routing table, here we also add tag statement to it:

```
ip route 192.0.2.1 255.255.255.255 Null0 tag 666
```

```
!
route-map static-to-bgp permit 1000
  match tag 666
  set community 65535:666 additive
!
! other rules for redistribution here
!
route-map static-to-bgp deny 65000
!
router bgp 64500
 address-family ipv4 unicast
  redistribute static route-map static-to-bgp
```

Note that the route-map static-to-bgp is not complete, of course you also need statements to add regular (not to be blackholed) routes to BGP. Using a "tag" statement is very elegant as you only have to add one line of configuration to start blackholing.

Without "tag" you can also achieve the same using an access-list or prefix-list, but then again you have to add/remove two lines of config, so this is possible but not really recommended.

For IPv6 it works very similar:

```
ipv6 route 2001:DB8:0:1::1/128 Null0 tag 666
!
route-map static-to-bgp permit 1000
  match tag 666
  set community 65535:666 additive
!
route-map static-to-bgp deny 65000
!
router bgp 64500
 address-family ipv6 unicast
  redistribute static route-map static-to-bgp
!
```

Draft

All this requires that your router is still reachable and responsive during the attack. It is strongly recommended that you use some out-of-band connection to configure your router.

#### Option 2: Use a BGP Injector

You can also use a separate router with an iBGP session to inject blackholing prefixes. Or some software (like ExaBGP) on a server. On your router you would configure an iBGP session to a server with ExaBGP, on your server an example config file for ExaBGP looks like this:

```
  neighbor 192.168.2.13 {
    router-id 192.168.2.14;
    local-address 192.168.2.14;
    local-as 64500;
    peer-as 64500;

    static {
```

```
        route 10.1.1.1/32 {
            community [ 65535:666 ];
            next-hop 192.168.66.66;
        }
    }
}
```

**Option 3: Use a separate BGP speaker with your upstreams**

To make this work you need a physically separate connection to your upstream providers and to your IXPs. Use either a BGP injector (like in option 2) or a router like in option 1 and either connect it on a separate physical circuit to your upstream provider or use *eBGP multihop*, making sure the eBGP session will not be affected from any attack.

### 9.14.3 Operation

**Be prepared!**

At least once every half year you should schedule an emergency exercise. How you do this depends on your organizational and network structure, but it should involve everyone in your operational departments who also would be involved in case of a real attack. You can have this scheduled and announced to your teams well beforehand (recommended for the first few exercises) or as a "surprise" (not recommended if you do not have a well trained team).

draft

All documentation on how to start blackholing and how to monitor it should be kept up to date (a good idea is to check after each emergency exercise if the documentation still matches reality) and should be easily accessible for your team. This might include keeping a printed version or a PDF document on your teams phones. Keep in mind attacks do also happen during the night and on weekends when your staff might not be in the office.

Also you need to make sure that your management network (the network you use to configure your routers) is separate from your production network and is shielded from attacks. You have to be able to initiate the blackholing after the attack has started.

**When under attack**

Use your prepared plan to initiate blackholing. An example plan might read like this:

1. Find out what the target is. Sounds easy, but if multiple attacks happen at the same time, this might be challenging.
2. Initiate blackholing of the targets IP address(es). This will:
    - sink the attack traffic within your network as early as possible
    - signal your upstream provider(s) and peers to blackhole at their side
3. Notify your customer! Let your customer know that he is under attack and that you have taken steps.
4. Check if the blackholing is effective. Not all of your upstreams or peers might honor the blackholing request. Talk to your upstreams and peers who do not. If the attack is extremely severe and still hurting your network be prepared to shut down connections to parties which do still send you attack traffic. This should be only seen as a

last measure. Talk to your peers. Everybody has experienced attacks and they might be able to help.

5. Monitor the attack. If it subsides, stop blackholing (but be prepared to re-initiate it if the attack increases again).

Draft

Draft

# Chapter 10

# BGP - Advanced Concepts

## 10.1 BGP Confederations

Scaling BGP for large networks (large in 1000s of BGP speaking routers) is hard. Although eBGP runs all of the Internet on millions of devices, an Autonomous System, as we know, has to:

- Be continuous. All routing elements interconnect with each other.
- Run either:
  - iBGP between its BGP speaking routers fully meshed - with 1000s of routers there are a *lot* of iBGP sessions to be configured and monitored.
  - one or more Route Reflectors(see 3.2.3) to distribute prefixes, this also needs careful design so you do not loose redundancy.
- Have a common management. Although not a formal requirement for an AS, you need to manage your routers somehow and if multiple groups of operators manage a common AS, chaos is ensured.

Draft

So when the first global ISPs emerged, a solution was sought to address these issues. From personal experience, I experienced the following network organization:

- Independent country organizations, running their in-country network.
- An international backbone, run by a central organization (by me).

Some country networks were members of their local Internet Exchanges, necessary both from a technical and also from a marketing point of view. Also, most country organizations had customers with an Autonomous System of their own. To get better pricing, it was decided that upstream capacity was purchased centrally. So from a technical point of view, before BGP confederations were introduced, the network looked like:

- Each country was running their own AS, some connecting to local IXPs, some serving customers with AS.
- The backbone was also member of some IXPs, and connected to multiple upstream providers.

- No customers were connected to the backbone, customers were only connected to the in-country networks.

So the AS path customers received were unnecessary long - two ASes were added instead of one (the AS number of the country network and the AS number of the backbone).

To improve the situation it was decided to migrate the network to a BGP Confederations setup.

How to describe a BGP Confederation? Best picture is kind of small envelopes inside a big envelope - to the outside only the big envelope is visible.

Some terms regarding BGP Confederations:

**Member Autonomous System:** This is an Autonomous System that is contained inside a Confederation - like a small envelope inside a bigger one. It is identified by an AS number (called "Member AS-Number"), but this AS number is only visible within the Confederation, so most of the time private AS numbers are used here.

**Confederation Identifier:** This is the AS number visible to the outside (kind of the number on the big envelope). As it is visible in the global routing table, a real public AS number has to be used.

### 10.1.1 Configuration

A typical BGP configuration of a Confederation Member looks like this (FRRouting example):

```
router bgp 65501
 bgp confederation identifier 5669
 bgp confederation peers 65502 65503 65504 65505
!
```

Draft

Note that all *Member Autonomous Systems* are listed here except the own one.

Sessions to neighbors are configured like all BGP sessions: With a "remote AS" number and an IP address. The routing process knows from the Confederation configuration at the top which ASes are Confederation Members and which are outside of the Confederation.

### 10.1.2 The AS Path

The usual handling of the AS path is, when a prefix is announced via eBGP to another AS, the announcing AS is inserting its own AS number at least once at the front of the AS path.

As between Confederation Members eBGP is spoken, the same is true within a Confederation: The announcing ASes number is added to the AS path (at the front).

When announcing a prefix to another AS *outside* of the Confederation the handling is different:

- All Confederation Member AS numbers are removed from the AS path
- The *Confederation Identifier* (= AS number visible to the outside) is added to the front of the AS path.

With this AS path handling, the "inside" of a Confederation becomes invisible to outside ASes.

## 10.2 FlowSpec

Usually BGP announces prefixes to neighbors. But the protocol itself is completely open to also announce other things. In case of *FlowSpec* not prefixes, but rules how to handle traffic are announced. And of course this is possible for both IPv4 and IPv6. FLowspec is defined in a number of RFCs:

- [RFC8955] - Dissemination of Flow Specification Rules
- [RFC8956] - Dissemination of Flow Specification Rules for IPv6
- [RFC9117] - Revised Validation Procedure for BGP Flow Specifications

### 10.2.1 Motivation

To stop certain types of DOS or DDOS attacks a network operator might require filtering at a remote network (or on multiple locations within your own network). To signal, what should be filtered, often BGP Communities are used. This has the disadvantage, that all possible filter types need to be pre-configured and only filters for which published BGP Communities exist might be applied.

BGP *Flow Specification* or *FlowSpec* in short solves this problem by announcing complex filter definitions and actions via BGP. This is realized as a Subsequent Address Family Identifier. In a router configuration, FlowSpec usually is configured in a separate *Address Family* context.

Draft

### 10.2.2 Toolbox

A *Flow Specification* consists of two parts: Rule(s) and Action. If fully implemented, a packet matching all the rules (if there are more then one rule in a flowspec they are *and*ed together) is handled according to the defined actions. If you want to *or* rules, you have to define more than one flowspec.

#### Rules

The following components are available to build rules:

- Destination Prefix
- Source Prefix
- IP Protocol
- Port (matches source- or destination-port)
- Destination Port
- Source Port
- ICMP Type (matches only ICMP packets)
- ICMP Code
- TCP Flags
- Packet length
- Diffserv code point

- Fragment (First, not first, last, don't fragment)

Numerical values like port numbers and length can be compared using an operator with a value. The operator can be equal, greater, less, not equal, greater equal or less equal. For flag values a bitmask operator can be used.

#### Actions

The actions are defined in BGP Extended Communities (see section 7.3). Multiple actions can be set. The following actions currently exist:

- Set traffic rate in bytes/s or in packets/s: If you set this to zero (or a negative number), *all* traffic matching the rule is discarded.
- Do "traffic action" - a bitmask of 48 bits of which are only two defined:
  - Bit 46 - Sample bit: Enable traffic sampling and logging.
  - Bit 47 - Terminal bit: If set, more rules following this one will be evaluated. If zero, evaluating of rules stops here.
- Redirect: This action redirects matching packets to a specified VRF routing instance.
- Mark: This instructs the router to modify the DSCP bits in the IP header to a specific value.

#### Ordering

Draft

More than one rule can match a flow, so there must be an ordering procedure which does not depend on the order of arrival of the rules. So ordering is done by comparing the components of a rule:

- Components (rules) are compared by numerical value, lower takes precedence.
- If same, IP prefixes are ordered "more specific first"
- If same or does not apply, component data is compared using *memcmp()*

[RFC8955] and [RFC8956] both even give an example in *Python* how to implement ordering.

#### Validation

Flow Specification can be used to send filters into a neighbor Autonomous System. For example a customer can use FlowSpec to announce filters to a transit provider. To make sure your customer cannot send you random filter rules (like block all traffic to port 179), a validation procedure is defined. The original validation procedure has been updated, the current one is defined in [RFC9117].

## 10.3 BGP as routing-protocol in data centers

Disclaimer: The author has never deployed BGP in a pure datacenter scenario, neither operated any production datacenter network. Therefor it is recommended you seek out documentation by authors who have. Please see [RFC7938] and [Dut17].

# Appendix A

# Acknowledgements

A document like this does not exist in a vacuum. I would like to thank everybody who helped me with suggestions and advice, especially

- The folks at DENOG, who suggested I should not have a separate IPv6 chapter but integrate IPv6 into every chapter.
- The participants of the very first incarnation of this training for their feedback.
- *Johannes Moos* for lots of fruitful discussions about BGP.
- *Bernd Spiess* for suggestions and corrections.
- *Job Snijders* for his presentation at RIPE77 and the idea with the three sets.
- *Eilin Geraghty* for her patience with my spelling.
- *Julian Fölsch* for introducing Latexmk and other improvements.

Draft

Draft

# Glossary

**Address Family** Type of network protocol which is being announced by BGP. The address family is speficied by an Address Family Identifier (AFI). BGP does allow not only address families IPv4 and IPv6 but more than 40 others. The list of Address Families can be found at IANA . 87, 89

**Address Family Identifier** Numerical value to identify an Address Family. IPv4 has the identifier 1, IPv6 is 2, the whole list can be found at this IANA page: `https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml`. 87

**AS Confederation** is according to [RFC5065] a collection of autonomous systems represented and advertised as a single AS number to BGP speakers that are not members of the local BGP confederation.. 87

**AS Confederation Identifier** is according to [RFC5065] an externally visible autonomous system number that identifies a BGP confederation as a whole.. 87

**Autonomous System** is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy. 9, 87, 89, 91

**Autonomous System Number** is a 32-bit number uniquely identifying an Autonomous System. 87

**BGP Path Attribute** BGP update messages for prefixes contain not only the AS-Path but also other attributes. These fall into the following four categories:

**well-known mandatory:** Needs to be understood by all BGP implementations and must be included in all prefix updates.

**well-known discretionary:** Needs to be understood by all BGP implementations but do not have to be there.

**optional transitive:** Do not have to be there, do not have to be understood by all implementations, but *stay* on the prefix and are forwarded to other BGP speakers (even if not understood)

**optional non-transitive:** Similar, but is not passed along to other BGP speakers

. 77, 87, 91

**Bidirectional Forwarding Detection** Bidirectional Forwarding Detection (BFD) is a protocol to check if a configured neighbor is alive. For this packets are sent quite rapidly

draft

between two systems (rapidly means in the 100ms time range), if no packets are received from the neighbor for a given time, the neighbor is considered to be no longer reachable which is then signaled to other protocols like BGP. BFD is defined in [RFC5880], its application on IPv4 and IPv6 is defined in [RFC5881]. [RFC5882] is about the general application of BFD and [RFC5883] describes BFD on multihop paths.. 87

**Blackholing** is a method to discard unwanted or malicious traffic. Instead of forwarding unwanted packets to their destination, they are discarded as early as possible. 9, 67, 73, 87

**Border Gateway Protocol** is a distance vector routing protocol used to exchange routing information between providers. 87

**Broadcast** A network communcation model where one sender sends packets to all possible receivers. Broadcast can be seen as a special case of Multicast.. 87, 90

**DDOS** Distributed Denial of Service attack (DDOS) is an attack against a system via the Internet. The attacker uses multiple (sometimes millions of) network sources to send more traffic towards the attacked system than it can handle. Collateral damage is quite often the network infrastructure to which the attacked system is connected to.. 87

**Default Free Zone** Part of the Internet where no default route is needed for routing but all routers know all prefixes. 87, 90

**Default Route** is a route which covers every destination for which there is no specific route in the routing table. The destination of the default-route is often called the default destination or the gateway of last resort. 67, 87

Draft

**Enhanced Interior Gateway Routing Protocol** is a IGP defined by Cisco in the 1980s to distribute routing information within a network. It was later openly specified in [RFC7868]. 87

**Exterior Gateway Protocol** was a predecessor to BGP. First defined 1982 in [RFC827] it became obsolete once BGP was widely used (around 1994). 87

**Global Routing Table** is the table in a router which contains all prefixes currently being routed in the Default Free Zone of the Internet. 9, 87

**ICMP** Internet Control Message Protocol - this protocol is used to signal errors when forwarding packets. 87, 92

**Interior Gateway Protocol** is a protocol running inside an Autonomous System to distribute the IP addresses of router interfaces. 87

**Internet Assigned Numbers Authority** is an entity responsible for all number resources in the Internet. This includes addresses, protocol identifiers, and more. 87

**Internet Protocol** is a protocol responsible for end-to-end communication on the Internet. There are currently two versions in use, named Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). 87

**IS-IS** (Intermediate System to Intermediate System) is an IGP running directly on top of layer 2. It is used to distribute interface addresses within a network. 12, 16, 87

**Local Internet Registry** is an organzation/company which receives IP address resources or Autonomous System Numbers as an allocation from a Regional Internet Registry and assigns these resources to end users. 11, 87

**Local Preference** is the first evaluated BGP Path Attribute in best path selection. It is an integer value, where a higher value is "better". It is redistributed via iBGP inside an Autonomous System. 4, 5, 42, 43, 54, 58, 60, 87

**MD5** is a hash algorithm, used to generate a checksum on given data. 63, 65, 87

**Multi Exit Discriminator** is a metric in BGP which is used to your neighbor where you prefer traffic for a prefix. 87

**Multicast** A network communication model where one sender sends packets which are addressed to multiple receivers. 87, 90, 91

**Open Shortest Path First** is a link state routing protocol. It is used as an IGP. 12, 14, 15, 87

**Regional Internet Registry** is an entity responsible for allocating IP addresses and AS numbers to Internet Providers. 11, 87

**RIPE** short for *Réseaux IP Européens*, is the community of network operators in the European, Russian, and Middle Eastern region. See also RIPE NCC. 9, 87

**RIPE NCC** is the Regional Internet Registry (RIR) for the European, Russian, and Middle Eastern region. 87, 91

Draft

**ROA** Route Origin Authorization - a cryptographically signed record which defines how a prefix can be announced, it defines the originating Autonomous System and the maximum prefix length. 68, 87, 91

**Round Trip Time** is the time measured in seconds or milliseconds it takes from sending out a packet until receiving a reply.. 87

**Route Reflector** is an iBGP speaker which sends *all* prefixes it receives out to its Route Reflector Clients. 25, 83, 87, 91

**Route Reflector Client** is an iBGP speaking node with usually only one iBGP connection to a Route Reflector. 25, 87, 91

**Routing Information Protocol** is an old and quite obsolete protocol which was used to distribute routing information. RIP is no longer in use. 87

**RPKI** Resource Public Key Infrastructure is a framework of certificates and ROAs which enables resource holders to cryptographically prove that a resource is theirs and to define how it can be announced via BGP. 68, 87, 91

**RPKI validator** is a piece of software which fetches RPKI certificates and ROAs from RIRs, checks the signatures of the certificates and ROAs and communicates with routers providing a list of certified prefixes and their allowed originating AS numbers.. 68, 87

**Subsequent Address Family Identifier** This value, together with AFI tells BGP exactly what routing information is announced. A SAFI value of 1 for example stands for glsunicast while 2 stands for Multicast. . 85, 87

**TCP** is part of the TCP/IP protocol stack. It is a connection oriented protocol taking care that everything which is sent is also received. 87

**Time To Live** is a counter in the Internet Protocol (IP) header which is decreased every time a packet is forwarded by a router. If this counter hits zero, the packet is discarded and an ICMP Time Exceeded message is sent back to the originator of the packet. 65, 87, 94

**Unicast** A network communication model where one sender sends packets to one receiver.. 87

Draft

# Acronyms

**AFI** Address Family Identifier. 87, 89, 91

**AS** Autonomous System. 10, 15, 21, 68, 87

**ASN** Autonomous System Number. 11, 87

**BFD** Bidirectional Forwarding Detection. 25, 87, 89

**BGP** Border Gateway Protocol. 13, 15, 87

**DDOS** Distributed Denial of Service attack. 87, 90

**EGP** Exterior Gateway Protocol. 14, 43, 87

**EIGRP** Enhanced Interior Gateway Routing Protocol. 12, 87

**IANA** Internet Assigned Numbers Authority. 10, 11, 87, 89

**IETF** Internet Engineering Task Force. 7, 87

**IGP** Interior Gateway Protocol. 3, 12, 15–20, 26, 28, 44, 87, 90, 91

**IP** Internet Protocol. 87, 92

**IPv4** Internet Protocol Version 4. 87, 90

**IPv6** Internet Protocol Version 6. 87, 90

**LIR** Local Internet Registry. 87

**MED** Multi Exit Discriminator. 4, 43, 87

**RFC** Request for Comments. 7, 47, 87

**RIP** Routing Information Protocol. 12, 13, 87

**RIR** Regional Internet Registry. 11, 68, 87, 91

**ROA** Route Origin Authorization. 68, 87, 91

**RTT** Round Trip Time. 87

Draft

**SAFI** Subsequent Address Family Identifier. 87

**TCP** Transmission Control Protocol. 21, 87

**TTL** Time To Live. 65, 87

Draft

# Bibliography

[RFC827] *Exterior Gateway Protocol (EGP)*. Tech. rep. 827. Oct. 1982. 46 pp. DOI: 10.17487/RFC827. URL: https://www.rfc-editor.org/info/rfc827.

[RFC1105] *Border Gateway Protocol (BGP)*. Tech. rep. 1105. June 1989. 17 pp. DOI: 10.17487/RFC1105. URL: https://www.rfc-editor.org/info/rfc1105.

[RFC1517] Bob Hinden. *Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)*. Tech. rep. 1517. Sept. 1993. 4 pp. DOI: 10.17487/RFC1517. URL: https://www.rfc-editor.org/info/rfc1517.

[RFC1883] Dr. Steve E. Deering and Bob Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. Tech. rep. 1883. Dec. 1995. 37 pp. DOI: 10.17487/RFC1883. URL: https://www.rfc-editor.org/info/rfc1883.

[RFC1930] John A. Hawkinson and Tony J. Bates. *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. Tech. rep. 1930. Mar. 1996. 10 pp. DOI: 10.17487/RFC1930. URL: https://www.rfc-editor.org/info/rfc1930.

[RFC1997] Tony Li, Ravi Chandra, and Paul S. Traina. *BGP Communities Attribute*. Tech. rep. 1997. Aug. 1996. 5 pp. DOI: 10.17487/RFC1997. URL: https://www.rfc-editor.org/info/rfc1997.

[RFC1918] Robert Moskowitz et al. *Address Allocation for Private Internets*. Tech. rep. 1918. Feb. 1996. 9 pp. DOI: 10.17487/RFC1918. URL: https://www.rfc-editor.org/info/rfc1918.

[RFC2385] Andy Heffernan. *Protection of BGP Sessions via the TCP MD5 Signature Option*. Tech. rep. 2385. Aug. 1998. 6 pp. DOI: 10.17487/RFC2385. URL: https://www.rfc-editor.org/info/rfc2385.

[RFC2283] Yakov Rekhter et al. *Multiprotocol Extensions for BGP-4*. Tech. rep. 2283. Feb. 1998. 9 pp. DOI: 10.17487/RFC2283. URL: https://www.rfc-editor.org/info/rfc2283.

[RFC2439] Curtis Villamizar, Ravi Chandra, and Dr. Ramesh Govindan. *BGP Route Flap Damping*. Tech. rep. 2439. Nov. 1998. 37 pp. DOI: 10.17487/RFC2439. URL: https://www.rfc-editor.org/info/rfc2439.

[RFC4486] Enke Chen and Vincent Gillet. *Subcodes for BGP Cease Notification Message*. Tech. rep. 4486. Apr. 2006. 6 pp. DOI: 10.17487/RFC4486. URL: https://www.rfc-editor.org/info/rfc4486.

[RFC4277] Danny R. McPherson and Keyur Patel. *Experience with the BGP-4 Protocol*. Tech. rep. 4277. Jan. 2006. 19 pp. DOI: 10.17487/RFC4277. URL: https://www.rfc-editor.org/info/rfc4277.

[RFC4271] Yakov Rekhter, Susan Hares, and Tony Li. *A Border Gateway Protocol 4 (BGP-4)*. Tech. rep. 4271. Jan. 2006. 104 pp. DOI: 10.17487/RFC4271. URL: https://www.rfc-editor.org/info/rfc4271.

Draft

[RIPE378] Philip Smith and Christian Panigl. *RIPE Routing Working Group Recommendations On Route-flap Damping*. RFC 378. RIPE, May 2006.

[RFC4360] Dan Tappan, Srihari R. Sangli, and Yakov Rekhter. *BGP Extended Communities Attribute*. Tech. rep. 4360. Feb. 2006. 12 pp. DOI: 10.17487/RFC4360. URL: https://www.rfc-editor.org/info/rfc4360.

[RFC4760] Ravi Chandra et al. *Multiprotocol Extensions for BGP-4*. Tech. rep. 4760. Jan. 2007. 12 pp. DOI: 10.17487/RFC4760. URL: https://www.rfc-editor.org/info/rfc4760.

[RFC5082] Carlos Pignataro et al. *The Generalized TTL Security Mechanism (GTSM)*. Tech. rep. 5082. Oct. 2007. 16 pp. DOI: 10.17487/RFC5082. URL: https://www.rfc-editor.org/info/rfc5082.

[RFC5065] Paul S. Traina, John Scudder, and Danny R. McPherson. *Autonomous System Confederations for BGP*. Tech. rep. 5065. Aug. 2007. 14 pp. DOI: 10.17487/RFC5065. URL: https://www.rfc-editor.org/info/rfc5065.

[RFC5398] Geoff Huston. *Autonomous System (AS) Number Reservation for Documentation Use*. Tech. rep. 5398. Dec. 2008. 4 pp. DOI: 10.17487/RFC5398. URL: https://www.rfc-editor.org/info/rfc5398.

[RFC5668] Dan Tappan, Yakov Rekhter, and Srihari R. Sangli. *4-Octet AS Specific BGP Extended Community*. Tech. rep. 5668. Oct. 2009. 5 pp. DOI: 10.17487/RFC5668. URL: https://www.rfc-editor.org/info/rfc5668.

[RFC5737] Jari Arkko, Michelle Cotton, and Leo Vegoda. *IPv4 Address Blocks Reserved for Documentation*. Tech. rep. 5737. Jan. 2010. 4 pp. DOI: 10.17487/RFC5737. URL: https://www.rfc-editor.org/info/rfc5737.

[RFC5880] Dave Katz and David Ward. *Bidirectional Forwarding Detection (BFD)*. Tech. rep. 5880. June 2010. 49 pp. DOI: 10.17487/RFC5880. URL: https://www.rfc-editor.org/info/rfc5880.

[RFC5881] Dave Katz and David Ward. *Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)*. Tech. rep. 5881. June 2010. 7 pp. DOI: 10.17487/RFC5881. URL: https://www.rfc-editor.org/info/rfc5881.

[RFC5883] Dave Katz and David Ward. *Bidirectional Forwarding Detection (BFD) for Multihop Paths*. Tech. rep. 5883. June 2010. 6 pp. DOI: 10.17487/RFC5883. URL: https://www.rfc-editor.org/info/rfc5883.

[RFC5882] Dave Katz and David Ward. *Generic Application of Bidirectional Forwarding Detection (BFD)*. Tech. rep. 5882. June 2010. 17 pp. DOI: 10.17487/RFC5882. URL: https://www.rfc-editor.org/info/rfc5882.

[RFC5925] Dr. Joseph D. Touch, Ron Bonica, and Allison J. Mankin. *The TCP Authentication Option*. Tech. rep. 5925. June 2010. 48 pp. DOI: 10.17487/RFC5925. URL: https://www.rfc-editor.org/info/rfc5925.

[RFC6480] Matt Lepinski and Stephen Kent. *An Infrastructure to Support Secure Internet Routing*. Tech. rep. 6480. Feb. 2012. 24 pp. DOI: 10.17487/RFC6480. URL: https://www.rfc-editor.org/info/rfc6480.

[RFC6482] Matt Lepinski, Derrick Kong, and Stephen Kent. *A Profile for Route Origin Authorizations (ROAs)*. Tech. rep. 6482. Feb. 2012. 9 pp. DOI: 10.17487/RFC6482. URL: https://www.rfc-editor.org/info/rfc6482.

[RFC6793] Quaizar Vohra and Enke Chen. *BGP Support for Four-Octet Autonomous System (AS) Number Space*. Tech. rep. 6793. Dec. 2012. 12 pp. DOI: 10.17487/RFC6793. URL: https://www.rfc-editor.org/info/rfc6793.

[RIPE580] R. Bush et al. *RIPE Routing Working Group Recommendation for Route Flap Damping*. RFC 580. RIPE, Jan. 2013.

Draft

[RFC6996] Jon Mitchell. *Autonomous System (AS) Reservation for Private Use*. Tech. rep. 6996. July 2013. 4 pp. DOI: 10.17487/RFC6996. URL: https://www.rfc-editor.org/info/rfc6996.

[RFC7196] Cristel Pelsser et al. *Making Route Flap Damping Usable*. Tech. rep. 7196. May 2014. 8 pp. DOI: 10.17487/RFC7196. URL: https://www.rfc-editor.org/info/rfc7196.

[RFC7454] Jerome Durand, Ivan Pepelnjak, and Gert Döring. *BGP Operations and Security*. Tech. rep. 7454. Feb. 2015. 26 pp. DOI: 10.17487/RFC7454. URL: https://www.rfc-editor.org/info/rfc7454.

[RFC7999] Thomas King et al. *BLACKHOLE Community*. Tech. rep. 7999. Oct. 2016. 9 pp. DOI: 10.17487/RFC7999. URL: https://www.rfc-editor.org/info/rfc7999.

[RFC7938] Petr Lapukhov, Ariff Premji, and Jon Mitchell. *Use of BGP for Routing in Large-Scale Data Centers*. Tech. rep. 7938. Aug. 2016. 35 pp. DOI: 10.17487/RFC7938. URL: https://www.rfc-editor.org/info/rfc7938.

[RFC7868] Donnie Savage et al. *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. Tech. rep. 7868. May 2016. 80 pp. DOI: 10.17487/RFC7868. URL: https://www.rfc-editor.org/info/rfc7868.

[RFC8210] Randy Bush and Rob Austein. *The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1*. Tech. rep. 8210. Sept. 2017. 35 pp. DOI: 10.17487/RFC8210. URL: https://www.rfc-editor.org/info/rfc8210.

[Dut17] Dinesh G. Dutt. *BGP in the Data Center*. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc, 2017. URL: https://resource.nvidia.com/en-us-bgp-datacenter.

[RFC8092] Jakob Heitz et al. *BGP Large Communities Attribute*. Tech. rep. 8092. Feb. 2017. 8 pp. DOI: 10.17487/RFC8092. URL: https://www.rfc-editor.org/info/rfc8092.

[RFC8212] Jared Mauch, Job Snijders, and Greg Hankins. *Default External BGP (EBGP) Route Propagation Behavior without Policies*. Tech. rep. 8212. July 2017. 7 pp. DOI: 10.17487/RFC8212. URL: https://www.rfc-editor.org/info/rfc8212.

[Sni17] Job Snijders. "Robust Routing Policy Architecture". In: *RIPE77*. Oct. 2017. URL: https://ripe77.ripe.net/presentations/59-RIPE77_Snijders_Routing_Policy_Architecture.pdf.

[RFC8203] Job Snijders, Jakob Heitz, and John Scudder. *BGP Administrative Shutdown Communication*. Tech. rep. 8203. July 2017. 6 pp. DOI: 10.17487/RFC8203. URL: https://www.rfc-editor.org/info/rfc8203.

[RFC8326] Pierre Francois et al. *Graceful BGP Session Shutdown*. Tech. rep. 8326. Mar. 2018. 12 pp. DOI: 10.17487/RFC8326. URL: https://www.rfc-editor.org/info/rfc8326.

[RFC8956] Christoph Loibl, Robert Raszuk, and Susan Hares. *Dissemination of Flow Specification Rules for IPv6*. Tech. rep. 8956. Dec. 2020. 19 pp. DOI: 10.17487/RFC8956. URL: https://www.rfc-editor.org/info/rfc8956.

[RFC8955] Christoph Loibl et al. *Dissemination of Flow Specification Rules*. Tech. rep. 8955. Dec. 2020. 36 pp. DOI: 10.17487/RFC8955. URL: https://www.rfc-editor.org/info/rfc8955.

[RFC9117] Jim Uttaro et al. *Revised Validation Procedure for BGP Flow Specifications*. Tech. rep. 9117. Aug. 2021. 12 pp. DOI: 10.17487/RFC9117. URL: https://www.rfc-editor.org/info/rfc9117.

Draft

[RFC9234] Alexander Azimov et al. *Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages*. Tech. rep. 9234. May 2022. 12 pp. DOI: 10.17487/RFC9234. URL: https://www.rfc-editor.org/info/rfc9234.

[Ben25] James Bensley. "Rewriting BGP Origin". In: *RIPE91*. Oct. 2025. URL: https://ripe91.ripe.net/programme/meeting-plan/sessions/30/GJ8PFJ/.

Draft

# Appendix B

# License

Draft

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work

or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions.

Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate

```
    comment syntax for the file format. We also recommend that a
    file or class name and description of purpose be included on the
    same "printed page" as the copyright notice for easier
    identification within third-party archives.

Copyright 2025 DE-CIX-Group / Team Academy / BGP

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

Draft