UCLouvain

École polytechnique de Louvain

Making BGP aware of the data plane

Author: **Emilie Deprez**
Supervisor: **Olivier Bonaventure**
Readers: **Cristel Pelser, Thomas Wirtgen**
Academic year 2023–2024
Master [120] in Computer Science

Abstract

The Border Gateway Protocol (BGP) is an inter-domain routing protocol that makes decisions based on information from the control plane. These decisions are used in the data plane to forward traffic to their destination. However, the control plane does not contain information about the current state of the data plane. Our objective is to explore the possibility and implications of adding knowledge from the data plane in BGP.

This master thesis is divided into two main steps. First, we explore the possibility of modifying the behavior of BGP if we know information about the data plane. We used a planned administrative shutdown as a study case. We settled a sub-objective, which is to reduce the number of BGP withdrawals sent between Autonomous Systems (ASes) when a route planned to be shut down is removed. We modified the goBGP implementation. Our experiment demonstrates that despite some limitations, such as dropping traffic or delaying route advertisement, we can modify the BGP behavior.

In the second step, we perform active measurements in the data plane and use the results in the BGP decision process. We created the Route Quality Measurement (RQM) system, a prototype that measures and quantifies the quality of a route in the data plane according to a metric. We added this metric into goBGP. Our experiments show that with our solution, routes with better quality are chosen as the best route instead of routes that would have been chosen for policy reasons. However, our prototype shows limitations in terms of capacity and precision when faced with a high load.

Acknowledgements

I would like to express my sincere gratitude to everyone who contributed from near and far and supported me in the realization of this master thesis.

First of all, I would like to thank my supervisor, Professor Olivier Bonaventure, for his sound advice, his expertise, and his trust decisive for the success of this work.

I would like to thank all members of its team who were present at meetings. Their presence and constructive discussions were a great help. I would particularly like to thank Thomas Wirtgen for his availability and his supervision.

I would like to thank my family, and more particularly my parents who believed in me, for their support and encouragement throughout this master thesis.

I would like to thank my friends for their support and encouragement, and more particularly, Guillaume Steveny and Paul-Edouard Bertrand Van Ouytsel.

Finally, I would like to thank Cristel Pelser and Thomas Wirtgen for agreeing to be part of the reviewers of this master thesis.

Use of AI tools

In this master thesis, we used Grammarly [1] an AI to help detect grammar mistakes in this manuscript (unfortunately it is not perfect). We also used DeepL Translator [2] to help translate some sentences of this manuscript into English. Finally, we have used chatGPT [3] to help the understanding of the Go programming language, a programming language that I did not know about before using it in this master thesis. It also provided the inspiration for some of the sentences in this manuscript, which have been modified to meet our needs.

[1] `https://app.grammarly.com`
[2] `https://www.deepl.com/en/translator`
[3] `https://openai.com/chatgpt/`

Contents

Chapter 1

Introduction

The Internet evolves continuously. The Border Gateway Protocol (BGP), defined for the first time in June 1989 in the RFC 1105 [1], is now in its fourth version [2]. Many extensions, for example, to secure BGP [3], have been added since the definition of this last version, and many more will be proposed in the future. The number of BGP messages exchanged per hour has increased exponentially these last years [4]. Currently, BGP takes routing decisions based on information from the control plane and does not take into account information from the data plane. Information about the current state of the data plane, such as the reachability, the delay, or the congestion are ignored. Only a few research have been conducted to take it into account. That is why we will ask ourselves in this master thesis whether it is possible to make the BGP "data plane aware" and what are the consequences of adding data plane information to it.

This master thesis is divided into two main parts. For the first part, we asked ourselves if it is possible to modify the behavior of BGP when it learns an element about the data plane. In the second one, we perform measurements directly in the data plane to use the results in the BGP decision process.

In the first part, we will look at the case of a planned administrative shutdown. We will try to influence the BGP behavior if it knows in advance that the data plane will no longer be reachable via this route, but an alternative route to reach the destination exists. We settled a sub-objective that is reducing the number of BGP withdraw messages sent between ASes if we know in advance that a shutdown will occur. Looking at the related work, we can see that solutions have been proposed to reduce the number of BGP withdrawals in case of failure [5, 6].

In the second part, we will add the results of quality measurements directly performed in the data plane in the BGP decision process. Currently, decisions are based on information present in the control plane. This information can be

manipulated by ASes operators for traffic engineering purposes [7, 8]. They can modify every attribute of the route, leading to selecting the best route based on manipulated information. Moreover, the route can be unreachable, have a delay abnormally high, be congested, ... This information is not taken into account by BGP for the decision. In the literature, we can find a proposition that verifies the reachability in the data plane of a route before accepting it in the Routing Information Base (RIB) of a router [9]. In this master thesis, we will measure in the data plane a metric that will allow us to quantify the quality of a BGP route. We will modify the BGP process to take into account this metric coming from the data plane.

The code developed for this work is available on GitHub [1] so that it can be used in further work.

This master thesis is structured as follows:

- Chapter 2 presents the background needed to understand this work. We cover the concepts of Border Gateway Protocol (BGP), Resource Public Key Infrastructure (RPKI), Virtual Routing and Forwarding (VRF), and statistical hypothesis testing.
- Chapter 3 and 4 present the study cases. Both chapters follow the same structure. After setting the context, we make a brief related work overview. Next, we explain the design of our solution, before exploring the experiments and their results. We finish with a brief discussion on this subject.
- Chapter 5 provides leads for pursuing this study based on the results and limitations that we found.

[1] `https://github.com/emiliedeprez/Making-BGP-data-plane-aware`

Chapter 2

Background

In this chapter, we present four concepts used in this master thesis. First, we describe BGP, the Border Gateway Protocol. Then, we take a look at what the Resource Public Key Infrastructure (RPKI) is and how it is used in BGP. Next, we explain what VRF (Virtual Routing and Forwarding) is. Finally, we explicate the concept of statistical hypothesis testing.

2.1 The Border Gateway Protocol

The Border Gateway Protocol (BGP) was defined for the first time in RFC 1105 [1]. Currently, we are at the fourth version defined by RFC 4271 [2]. This protocol, which runs above TCP on port 179, is a routing protocol between Autonomous Systems (ASes). Inside an AS, the same routing policies are used. It is a network or a group of network operators that has one or multiple prefixes [10]. A unique number called Autonomous System Number, ASN, is assigned to each AS.

A router running BGP is called a BGP speaker. It establishes one or multiple BGP sessions with other BGP speakers called peers. These peers may come from the same AS or different ASes. An internal peer is a peer from the same AS, while an external peer is from a different AS. We use the terms "internal BGP" (iBGP) and "external BGP" (eBGP) to describe BGP sessions between, respectively, internal and external peers.

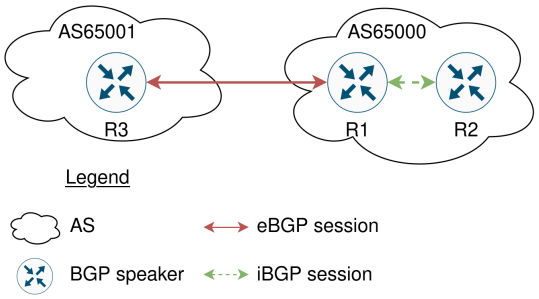


Figure 2.1: A topology example

Figure 2.1 shows an example of topology. Let us take the point of view of the BGP speaker R1, represented by a router icon. R1 is part of AS65000. It has two peers. The first one, R2 is an internal peer. The iBGP sessions are represented by green dotted arrows. The second one, R3 is an external peer lying inside AS65001. The eBGP sessions are represented by solid red arrows. We use this notation throughout this master thesis.

2.1.1 Messages

RFC 4271 [2] defines four types of messages:

1. OPEN message: establish a BGP session between two BGP speakers.
2. UPDATE message: advertise available routes and withdraw unavailable routes. A route is an IP prefix associated with the attributes of a path to this prefix.
3. KEEPALIVE message: notify the peer that the connection is still alive. It is used to determine if a peer is down or not. It uses a hold timer, which expires if no UPDATE, KEEPALIVE, and/or NOTIFICATION message arrives during the hold time interval. When the hold timer expires, the peer is considered down. The hold time is negotiated in the OPEN message.
4. NOTIFICATION message: advertise an error and close the BGP session afterward.

RFC 2918 [11] proposes a new type of message, ROUTE_REFRESH, which asks a peer to readvertise some prefixes. In the following part, we will focus on two messages: the OPEN message and the UPDATE message. The Appendix A.1 gives a summary of the current existing messages, their types, and the RFC where they are defined.

The OPEN message

The OPEN message defines multiple parameters used during the BGP session. In RFC 4271 [2], fives are mandatory. By taking the point of view of the message

sender, the message contains:

- the version (The current value is 4);
- its ASN;
- the hold time wanted;
- its BGP identifier, an IPv4 address assigned to the BGP speaker and the same for every BGP peer;
- the length of optional parameters. If the value is not zero, it contains the field optional parameters, which is a list of the 3-tuples <parameter type, parameter length, and parameter value>.

Appendix A.3 provides the currently available optional parameters. The one that interests us is the Capabilities Optional Parameter defined by the RFC 5492 [12]. This optional parameter informs that the BGP speaker can support one or multiple parameters. For example, to use the ROUTE_REFRESH message, the BGP peer must have advertised the Route Refresh Capability in the OPEN message. The list of current Capabilities is summarized in Appendix A.4

The UPDATE message

An NLRI, Network Layer Reachability Information, is an IP prefix associated with the length of the prefix. RFC 4271 [2] specifies the possible presence of 5 fields.

- Withdrawn routes length: a mandatory field that represents the length of the withdrawn routes field. If the value is 0, no route is withdrawn, and the field "withdrawn routes" is not present.
- Withdrawn routes: a list of IPv4 NLRI to withdraw.
- Total path attribute length: a mandatory field that represents the length of the path attributes field. If the value is 0, no route is advertised, and the field "path attributes" and "network layer reachability information" are not present.
- Path attributes: the list of attributes of the advertised routes.
- Network Layer Reachability Information (NLRI): a list of IPv4 NLRI that share the path attributes.

Attributes are categorized into four types [2]: well-known mandatory, well-known discretionary, optional transitive, and optional non-transitive. Well-known attributes mean that every BGP implementation must support these attributes, while optional attributes may or may not be supported. A mandatory attribute means

that it must exist in BGP messages, unlike discretionary attributes. A transitive attribute must be passed to other BGP peers, even if it is unrecognized. That is not the case for non-transitive attributes that must not be transmitted to other BGP peers. The following attribute list summarizes the attributes used in this master thesis and is defined in RFC 4271 [2] and in RFC 1997 [13] for communities. This list is not exhaustive. More attributes are present in the RFC 4271 [2] and multiple other RFCs [14, 15, 16, 17, 18, 19] define new optional attributes to add functionalities to BGP such as route-reflection [14] or to allows traffic engineering [15]. Appendix A.2 lists all attributes.

- ORIGIN: A well-known mandatory attribute that defines the origin of the path. It can be IGP (interior gateway protocol), EGP (exterior gateway protocol), or incomplete.
- AS_PATH: A well-known mandatory attribute that contains an ordered or unordered set of ASes that the prefixes specified in the NRLI went through.
- NEXT_HOP: A well-known mandatory attribute that contains the IPv4 address of the router that should be used as next hop to reach the routes in the NLRI.
- MULTI_EXIT_DISC (MED): An optional non-transitive attribute used to favor incoming neighbor traffic on one of the multiple links to a neighboring AS.
- LOCAL_PREF: A well-known attribute used to calculate the degree of preference for the outgoing traffic for each external route of an AS. It shall be included in all BGP messages sent to an internal peer and never included to an external peer.
- COMMUNITIES [13]: An optional transitive attribute that consists of a list of communities. A community is composed of two octets for an AS number and two octets for the meaning defined by the AS. It is written with the format `<AS_number>:<community>`.

We have to mention two other attributes MP_REACH_NLRI and MP_UNREACH_NLRI from the RFC 4760 "Multiprotocol Extensions for BGP-4" [19]. We previously mentioned that NLRI and the next hop are IPv4 prefixes and addresses. These two attributes allow using IPv6 or other Network Layer protocols in UPDATE messages. The MP_REACH_NLRI makes possible the advertising of IPv6 prefixes or the use of IPv6 addresses as next hop. The MP_UNREACH_NLRI attribute enables to withdraw IPv6 prefixes. Instead of adding the prefixes in the NLRI fields, these attributes keep them. The use of these attributes requires the advertisement of the Multiprotocol Extensions capability in the OPEN message. To send a message with these attributes to a BGP peer, this one should have sent the capability.

2.1.2 The ADD-PATH capability

In this master thesis, we are going to discuss the ADD-PATH capability. This capability, defined by the RFC 7911 [20], allows multiple paths to be advertised in BGP.

In RFC 4271 [2], one prefix is associated with one path. When a route with the same prefix is readvertised, the new one replaces the old one. The RFC 7911 [20] allows advertising multiple paths for a prefix. It extends the NLRI (composed of the prefix length and IP) by adding a Path Identifier. The address prefix combined with the path identifier distinguishes a particular path for a prefix. When a route with the same prefix is advertised, the new one does not necessarily replace the old one. If the combination <prefix, path identifier> does not exist, it adds the path. If the combination <prefix, path identifier> exists, the new route replaces the old one with the same path identifier.

2.1.3 BGP workflow

The BGP workflow is explained based on the RFC4271 [2]. A BGP session starts with an OPEN message. It stops when the BGP speaker is administratively shut down, detects an error in BGP messages, receives a NOTIFICATION message, or if the hold timer expires.

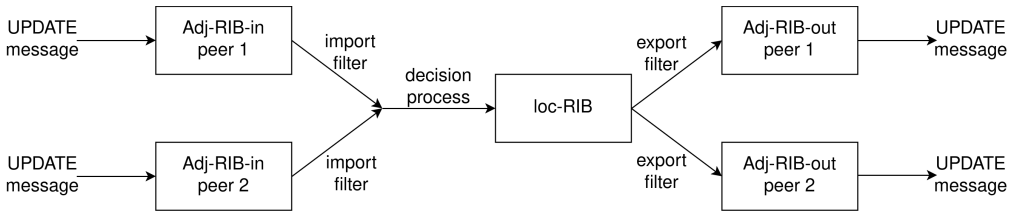


Figure 2.2: The workflow of UPDATE messages for 2 BGP peers

Now let us take a closer look at UPDATE messages. Learned routes are stored in Routing Information Bases, RIBs. RFC 4271 [2] defines three types of RIBs: Adj-RIB-in, loc-RIB, and Adj-RIB-out, represented in Figure 2.2. There is one Adj-RIB-in and one Adj-RIB-out per peer. An Adj-RIB-in contains the routes learned from one peer. These routes are available for the decision process after being processed by an import filter. It applies policies configured by the BGP speaker. More explanations are provided in the policies subsection. The decision process aims to select only one route, the best towards a prefix. It is detailed in the decision process subsection. The best route is placed in the loc-RIB. The loc-RIB stores every best route learned over all sessions with the peers. Then, the best

route is exported to the Adj-RIB-out for each peer, after being processed by an export filter. As the import filter, it applies policies. The Adj-RIB-out contains routes selected to be advertised to the corresponding peer. These three types of RIBs are theoretical. Implementations do not need to keep the three.

Policies

Policies are routing policies that are configured by BGP speakers operators. We have seen that there are import policies and export policies. These policies match some routes and apply actions to them. The matching is done on the prefix, the peering neighbor, a particular AS-PATH, or the presence/absence of one or more communities or one or more ASes in the AS-PATH. It is also possible to apply action directly to all routes without making a match. The first action is to reject or accept the route. Usually, the default parameter is to accept the route. So, it is possible to reject routes. For rejected routes, these are not taken into account in the decision process and are not present in the loc-RIB if it is at the import stage. They are not present in the Adj-RIB-out and are not advertised if it is at the export stage. If the route is accepted, multiple actions on attributes can be done. At the import stage, it can add or remove communities, set the next hop, set the local preferences,... At the export stage, it can add or remove communities, set the next hop, set the MED, or prepend the AS-PATH more than once, ... Prepending its ASN means adding multiple time its ASN in the AS-PATH to enlarge it.

The BGP decision process

The BGP decision process chooses the best route toward a prefix. It compares all routes with each other. It is composed of tie-break rules. Each rule uses a feature to compare and remove routes. It stops when there is only one route left, the best route among all others. The following enumeration presents the different tie-break rules described in RFC 4271 [2]. Some RFCs, such as the RFC 4456 [14], can update this decision process by adding steps to it.

1. Remove all routes route with an unreachable BGP next hop.
2. Keep routes with the highest local-pref attribute. Remove all others.
3. Keep routes with the shortest AS-PATH. Remove all others.
4. Keep routes with the lowest origin number (IGP over EGP over incomplete)
5. Keep routes with the smallest MED. Remove all others. MED can only be compared between routes learned from the same neighboring AS.
6. If at least one route was learned via eBGP session, remove all routes learned via iBGP session else, keep all routes.
7. Keep routes with the closest next hop. Remove all others.

8. Keep routes learned via BGP speakers with the lowest BGP Identifier Value. Remove all others.
9. Keep the route learned from the lowest peer address.

2.1.4 ASes peering relation

ASes are connected between each other. Handling traffic has a cost. Then, there are relations between ASes to define which peers paid and which routes are announced. There are two types of relationships [21]. The first one is customer-provider, and the second one is peer-to-peer. A customer-provider relationship is such that one AS, the customer, pays another AS, the provider. A peer-to-peer relation is such that each peer AS handles its own cost. According to the type of relations, a router will advertise a route or not.

- A provider advertises all routes that he knows to its customers.
- A customer advertises only the routes learned by its own customers and its own prefixes.
- Over a peer-to-peer relation, each peer advertises only the routes learned by its own customers and its own prefixes.

The choice of the best route considers the peering relationship. A route learned from a customer is preferred over a route learned from a peer-to-peer relation or a provider. And a route learned from a peer-to-peer relation is preferred to a route learned from a provider.

These relations are enforced using policies. The Appendix B provides an example of how policies can be used to apply relationships. Misconfigurations or no route filtering leads to advertising prefixes that do not respect the peering relationship. This is called route leaks. RFC 9234 [22] aims to prevent this problem. It adds a capability in the OPEN message to establish an agreement of the peering relationships. This capability defines the role of the local BGP speaker and the remote BGP speaker. Among these roles, we find our three previously mentioned roles: provider, customer, and peer. These roles ensure the correctness of the relation. RFC 9234 [22] also proposes a new path attribute called "Only to Customer". This attribute is optional and transitive. It allows to prevent creation of leaks and detects them if present in an AS-PATH.

2.1.5 Topologies

Over an eBGP session, a router announces its best route. Over an iBGP session, a router announces its best route if it was not learned by iBGP. It is not a problem

if the topology of an AS is in BGP full mesh. Every BGP speaker is aware of all the best routes. Figure 2.3 is an example of BGP full mesh.

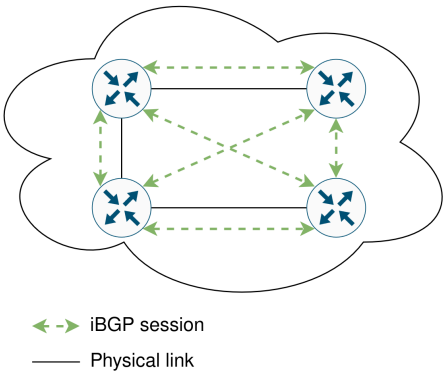


Figure 2.3: A simple BGP full mesh topology

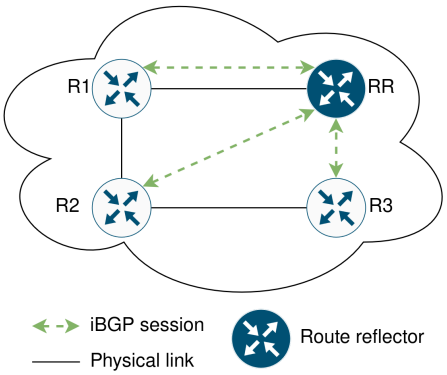


Figure 2.4: A simple route-reflector topology

A BGP full mesh topology inside an AS is unmanageable from an operational point of view. To create a BGP full mesh, one needs to configure $\frac{n*(n-1)}{2}$ BGP sessions where n is the number of routers inside the AS. Then, for 15 routers, it requires 105 peering sessions. Knowing that the interfaces on each side need to be configured, there are 210 interfaces to configure. It can lead to configuration errors. It does not scale with larger ASes [14]. There is an alternative to BGP full mesh topologies: Route Reflection [14]. Route reflection is when a router announces a route learned over an iBGP session. These routers are called route reflectors. They are different from simple routers. An iBGP peer can be a client or a non-client of the route reflector. The announcement of the best path of the route reflector depends on the type of peer it learned from. If the best path is a route learned from a non-client iBGP peer, the route reflector announces this best path to only its client. If the best path is learned from a client iBGP peer, it reflects the route to all iBGP peers. It avoids creating a BGP full mesh because route reflectors can, in some cases, announce their best routes to their iBGP peers.

Figure 2.4 provides an example of a topology that uses route-reflectors and is not in BGP full mesh. We can quickly see that instead of having six iBGP sessions for the BGP full mesh topology 2.3, we have only three iBGP sessions. R1, R2, and R3 are clients of RR. If we assume that R1 and R2 have received a prefix p from an eBGP session, they will send this prefix to the route reflector RR. It will then choose its best path among the two routes. Let us assume the best path for the RR is through R1. It will announce this path to R2 and R3. It is important to note that the route reflector can hide routes. R1 and R3 do not know the route via

R2. Maybe the best route for R3 is the route learned by R2.

2.2 The Resource Public Key Infrastructure

The Resource Public Key Infrastructure (RPKI) is a security framework to secure Internet routing [23]. It authenticates the ownership of IP address blocks and Autonomous System (AS) with certificates. The RPKI architecture is composed of three main elements: RPKI, digitally signed routing objects, and a distributed repository system to hold the objects.

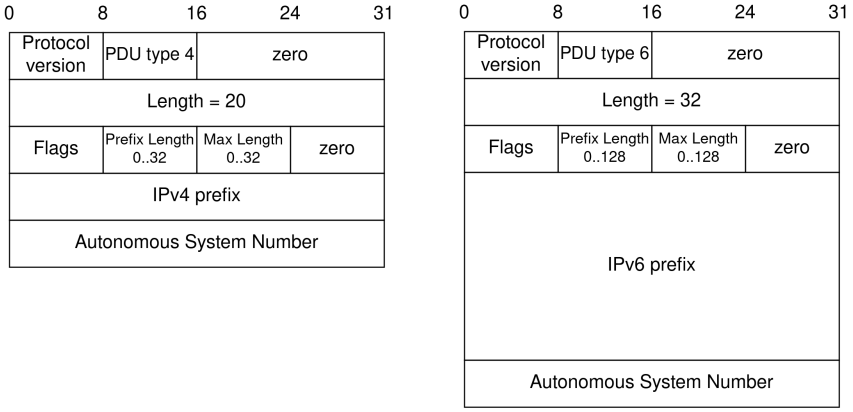
The distribution of IP address blocks and Autonomous System Numbers follows a hierarchical structure. The IANA, Internet Assigned Numbers Authority, [24] is an organization that is responsible in particular for IP addressing and ASN allocation. It allocates blocks of IP addresses and ASNs to Regional Internet Registries (RIRs). There are five RIRs: AFRINIC, APNIC, ARIN, LACNIC, and RIPE NCC. These RIRs distribute resources allocated to them to Local Internet Registries (LIRs), National Internet Registries (NIRs), and subscribers such as Internet Service Providers (ISPs). The IANA issues certificates to the RIRs to prove that they can use these IP prefixes and ASNs. In their turn, the RIRs issue certificates to LIRs, NIRs, and subscribers [23].

In this master thesis, we focus on one particular RPKI digitally signed object [25], the Route Origin Authorization or ROA [26]. It binds an AS with one or multiple IP prefixes and is signed by a certificate. It allows the owner of an IP address block to specify which ASes are authorized to announce this prefix. An ROA contains the following information: the version, the AS number, and a set of ROA IP addresses that are the prefixes authorized to advertise. A ROA IP address is a structure that contains the prefix and an optional max length. This max length authorizes announcing more specific prefixes with this maximum prefix length. For example, if the prefix is 203.0.113.0/24 with a max length of 30, the AS is authorized to announce the prefix 203.0.113.0/30 but not 203.0.113.0/31.

The deployment structure of RPKI is the following. We have RPKI distributed repositories that contain certificates, signed objects, and Certificate Revocation Lists [27]. Then, we have local caches which contain a copy of all RPKI data from this repository. It checks RPKI certificates and signed objects. Finally, we have routers. They communicate with the local cache that transmits validated signed object [28, 29]. Routers must have a trusted relationship with the local cache.

The Resource Public Key Infrastructure (RPKI) to Router Protocol [29], or RTR, is a protocol that exchanges data between a trusted local cache and a router.

This allows the router to receive cryptographically validated RPKI. It exchanges Protocol Data Units (PDUs). The ones that interest us are the IPv4 and IPv6 prefixes. These PDUs announce or withdraw the information of which IP addresses can be advertised by which AS. The local cache must only advertise validated RPKI, i.e., the certificate is verified and validated. Figure 2.5 shows the structure of IPv4 and IPv6 Prefix PDUs. These PDUs transport only one IP prefix and one ASN. If several ASes can announce the same prefix, multiple PDUs are sent, one per AS authorized. In other words, these PDUs announce or withdraw validated ROAs to the router.



(a) PDU IPv4 Prefix (b) PDU IPv6 Prefix

Figure 2.5: Structure of the IPv4 and IPv6 Prefix PDU

In BGP, to validate a route, it first loads the validated object from the cache into a local storage with the RTR protocol [29, 30, 31]. There are three validation states: *not found / unknown*, *valid*, and *invalid*. A route is *unknown* or *not found* if no validated ROA matches this IP address. A route is *valid* if there is one or multiple ROAs that cover the route prefix, i.e., the route prefix is among the ROA prefix and the maximum length, and one of the ROAs has the ASN equal to the origin AS of the route. A route is *invalid* if at least one ROA covers the route prefix and no ROA that covers the route prefix has the ASN equal to the origin AS of the route. Routing policies can be used to reject/accept the route or modify attributes according to its validity.

2.3 Virtual routing and forwarding

A VRF, Virtual Routing and Forwarding, isolates the traffic at the network layer [32, 33]. It allows the creation of multiple routing tables in addition to the default

one on the same router. A network device is associated with a VRF and uses its routing table. We use Figure 2.6 to present how it works. The middle router has two VRFs. In total, it has three routing tables: the default one and one for each VRF. The interfaces eth1 and eth3 are associated with the red VRF. The interfaces eth2 and eth4 are associated with the blue VRF. Finally, the interfaces eth5 and eth6 are associated with no VRFs. In this schema, R1 can only communicate with R3, R2 with R4, and R5 with R6. The traffic is isolated between the VRFs and the default routing table. For example, R3 can not send traffic to R2 because the red VRF, where the traffic from R3 comes from, does not know the prefix 192.168.2.0/24.

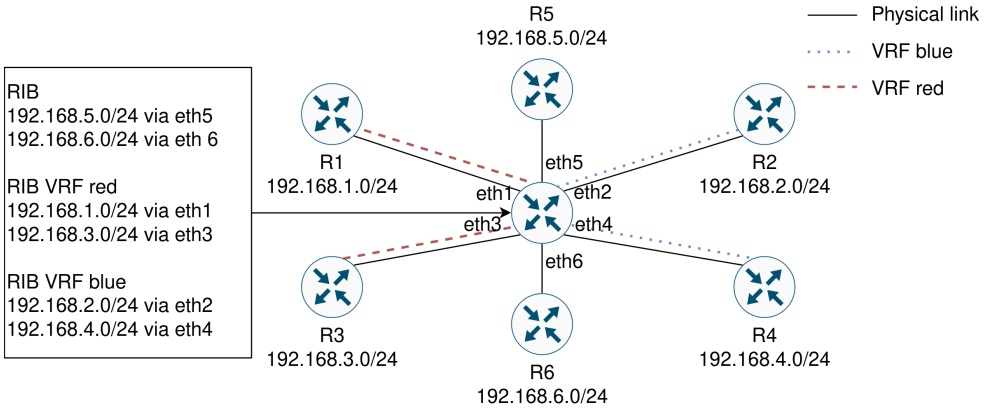


Figure 2.6: Example of VRFs

2.4 Statistical hypothesis testing

In this section, we introduce the concept of statistical tests and more precisely the two-sample t-test. It is used to compare the means of two samples [34]. It assumes that the samples are normally distributed, they have equal standard deviations, and they are independent.

Based on the e-Handbook of Statistical Methods [34] and the lesson "Statistics and data sciences" given at UCLouvain [35], we can define several variables and formulas. Let x_1 and x_2 represent the two samples where n_1 and n_2 are their respective size.

We have

$$\bar{X}_i = \sum_{j=0}^{n_i} \frac{x_{ij}}{n_i} \tag{2.1}$$

which is the empirical mean of sample x_i. x_{ij} is the j-th measure of sample x_i. We also have

$$s_i^2 = \sum_{j=0}^{n_i} \frac{(x_{ij} - \bar{X}_i)^2}{n_i - 1} \tag{2.2}$$

which is the empirical standard deviation of sample x_i. The pooled variance is calculated with

$$S_{pool}^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \tag{2.3}$$

The test statistic where we assume equal variances is

$$T(x_1, x_2) = \frac{(\bar{X}_1 - \bar{X}_2)}{S_{pool}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{n_1+n_2-2}$$

where $t_{n_1+n_2-2}$ is a t-Student distribution with a degree of freedom equal to $n_1 + n_2 - 2$.

Now we can formulate our hypothesis:

$$\begin{array}{ll} H_0 : \mu_1 = \mu_2 & \\ \text{We reject } H_0 \text{ at the level } \alpha \text{ if} & \\ H_1 : \mu_1 > \mu_2 & T(x_1, x_2) > t_{n_1+n_2-2,\, 1-\alpha} \\ H_1 : \mu_1 < \mu_2 & T(x_1, x_2) < t_{n_1+n_2-2,\, \alpha} \end{array}$$

α is the significance level, often set to 0.05. $t_{n_1+n_2-2,\, \alpha}$ represents the critical value used to reject H_0. H_1s are hypotheses that, if verified, reject the H_0 hypothesis.

We can also use the p-value to reject the hypothesis H_0. The p-value is "the probability of obtaining, by chance alone, results at least as extreme as those obtained" [36], assuming the null hypothesis is true. We reject the null hypothesis at level α if p-value $< \alpha$.

$$\text{p-value} = 2 * P(|T(x_1, x_2)| < t_{n_1+n_2-2,\, \alpha} | H_0 \text{ is true}) \tag{2.4}$$

Chapter 3

Advertising BGP of a planned administrative shutdown

The objective of this chapter is to show that it is possible to improve BGP to be aware of its environment by considering information about the data plane. We are going to study the case of a planned administrative shutdown of a link. These shutdowns are used for maintenance, for example. In such cases, many routes are withdrawn. We will analyze these withdrawals and how to reduce or even avoid them in these cases.

In the following, we give more detail about the context. Next, we do an overview of the literature. Then, we analyze BGP withdrawals in the real world. We designed a solution and an experiment to validate it. We conclude with a brief discussion.

3.1 Context

To shut down a link between two BGP peers, we stop the session by, for example, modifying the BGP configurations to disable the peer or deactivate the physical interface. As the peer becomes unreachable, all routes learned from it are withdrawn. If no alternative routes are found, these withdraw messages are propagated to all peers.

Withdrawing a route might have unwanted consequences. Let us take the figure 3.1 to illustrate our point. We have an AS, AS65000, that advertises the prefix 203.0.113.0/24. At the convergence, AS65002 knows the direct route to this prefix and a route through AS65001. AS65003 learned only the route via AS65001. $R1$ knows the route via AS65002 and via R2. As the AS-PATH via R2 is shorter, it prefers this route. R2 knows only the route directly via AS65000. R1 has not

advertised the alternative route via AS65002 because it is not its favorite route. Now, if the link between AS65000 and R2 is shut down, R2 has to remove the route that it knows and sends a withdraw message for the prefix to AS65003 and R1. R1 removes the route, and now, as it knows the alternative route, it prefers the only route through AS65002. It sends a withdraw for the route via AS65000 to AS65002 and sends an update to R2 with the route via AS65002. R2, then, sends this route to AS65003. Due to the hidden route, we have sent a withdraw message to the external AS65003 and a little later an update. This triggers multiple convergences in AS65003, and in turn, it can also send withdraws for prefixes that will receive a new route later.

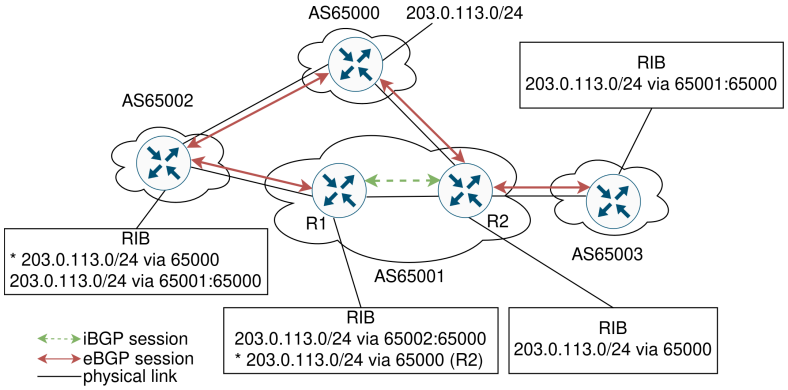


Figure 3.1: A topology example with hidden routes

Another important source of hidden alternative routes is route reflectors. These are widely used since BGP full meshes present significant operational management issues [14]. For more details about route advertisement, consider reading section 2.1.5 from our Background. As route reflectors can announce only their best path to some iBGP peers, alternative paths are hidden to some BGP speakers from the AS. Topology 3.2 shows an example of hidden routes with a route reflector. We assume that every iBGP peer that has a session with the route reflector is a client. Like this, the route reflector announces its best route to all its iBGP peers. In this topology, AS65000 announces the prefix 203.0.113.0/24. The route reflector learns this prefix via R1 and R2. We assume that R1 has a lower BGP Identifier Value than R2. That is used as the tie-break rule between the route from R1 and R2. Then, the route reflector announces its best route to R2 but does not advertise the route via R2 to R1. Therefore, in the case of R1 receiving a withdrawal for this route, it will send a withdrawal message to AS65003, like the previous example.

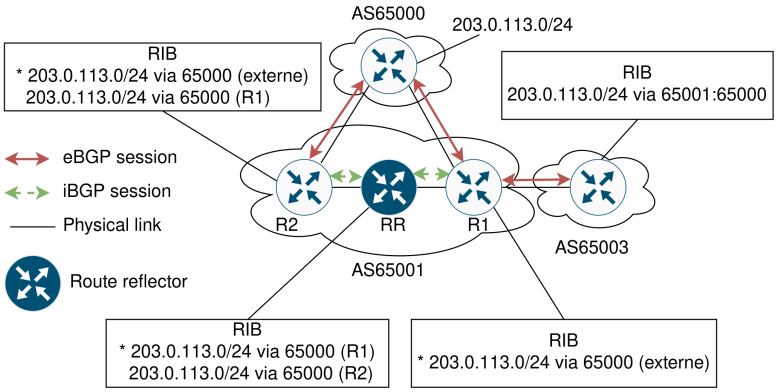


Figure 3.2: A topology example with hidden routes and route reflector

We propose a solution to avoid announcing a prefix withdrawal followed by an update for the same prefix in case of a planned administrative shutdown.

3.2 Related work

In 2009, Virgine Van den Schrieck et al. [5] studied the case of route failure, the impact on BGP, and a solution to counter this problem. This solution relies on using communities to advertise the presence of alternative routes inside an AS. Inside an AS, if a router knows an alternate path, it sets the community PATH_DIVERSITY on its best route before advertising it to its iBGP peers. When a BGP speaker receives a withdraw message for a route with the community PATH_DIVERSITY and does not have an alternate path, it sets the local preference to 0 and re-advertises the route to iBGP peers. It does not advertise eBGP peers. It also sets a timer that, when it expires and no alternate path has been received, the router sends a withdraw message on its eBGP sessions to avoid blocking BGP convergence in case the alternative route no longer exists. They also use two communities EPC_DIVERSITY and NON_EPC_DIVERSITY to know if the alternative route is advertised to at least the same eBGP sessions as the best route.

ADD-PATH, defined in the RFC 7911 [20], is a solution to avoid hidden alternative route. By advertising multiple paths for the same prefix, a router knows the alternatives to its best route. However, even if ADD-PATH has advantages, it also has drawbacks. Virgine Van den Schrieck et al. [6] have investigated the tradeoff. ADD-PATH is fast in case of failure recovery as it can use a backup path. It reduces the propagation burst of withdraw and update messages to eBGP sessions. However, the number of messages exchanged to advertise the additional paths and the memory used is higher. The overhead depends on the AS size, the number of

peering ASes, and the mode used. They depicted multiple modes that can be used, their advantages, and their drawbacks.

Alberto García-Martínez and Marcelo Bagnulo [37] have described the evolution of the route propagation delay between 2012 and 2018. They found that the time of convergence for IPv4 and IPv6 prefixes announcement has not changed between 2012 and 2018. The time of convergence for IPv6 prefixes withdrawal has also not changed. However, the time to withdraw an IPv4 prefix has increased.

3.3 Measuring of BGP withdrawals in the real world

We have collected data from two platforms that collect BGP data: RouteViews [38] and RIPE NCC RIS [39].

Our goal is to measure the number of BGP messages exchanged due to a prefix withdrawal that is still reachable by an alternative route. RIPE RIS [39] has provided a service of this kind for the prefix 84.205.72.0/24. This service has stopped on 2 January 2024, but we have data before this date. The prefix 84.205.72.0/24 was simulating an anycast failover with two RIPE beacons: RRC03 at Amsterdam (Netherlands, Europe) and RRC14 at Palo Alto (California, United-State, America). The prefix was permanently announced by RRC14. At RRC03, it was announced at 00:00, 04:00, 08:00, 12:00, 16:00, and 20:00 UTC and was withdrawn at 02:00, 06:00, 10:00, 14:00, 18:00 and 22:00 UTC.

We have retrieved from 57 collectors all updates and ribs for 14 days between September 1 and 14 2023. Appendix C lists the collectors used and information about them. We have focused our analysis on when RRC03 withdraws the prefix. We define a sample as the set of BGP messages sent to a collector from one peer during the 10 minutes after RRC03 has withdrawn the prefix. In total, we have 51413 samples for these 14 days.

Let us analyze the number of BGP updates and withdrawals that are sent per sample. Figure 3.3 shows the percentage of cumulated number of samples according to the number of BGP messages. We can observe, for withdrawals, that for 84% of samples, no withdrawals are sent. 14% of samples contain one BGP withdraw, and the remaining 2% contains between two and four BGP withdraws. For updates, 5% does not send a BGP update. We can deduce that only BGP withdrawals are sent for these 5% samples, and among the 16% of samples that contain BGP withdrawals, 11% contains at least an update. We can see, for updates, that the curve grows rapidly and then stabilizes. Most of 94% contain between 0 and 10

BGP updates. 99.5% of samples contain at most 20 BGP updates. For 0.5%, there can be up to 41 BGP updates. After 10 minutes, 12% of the samples contain a BGP withdrawal at the end and no longer advertise the prefixes. 4% of the samples contain at least one BGP withdrawal followed by an update. This last statistic is what we want to avoid with our solution.

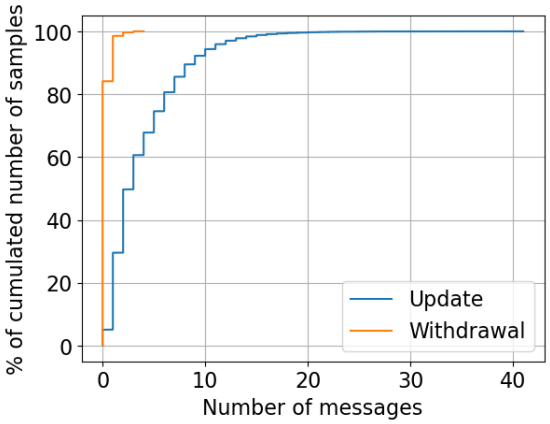


Figure 3.3: Percentage of cumulated number of samples according to the number of BGP messages

3.4 Design of the solution

Our solution is based on a community to advertise the shutdown of a route. This community allows BGP to learn that the data plane will be unavailable, but the prefix can be reached by another route. We define a new well-known community 0xFFFFFF05 (65535:65285). This well-known community is unasigned at the IANA at the time writing this master thesis [1]. We will call this community PLANNED_SHUTDOWN. When there is a planified shutdown for a link, the administrator has to advertise the impacted routes with this community for all prefix that will stay available via another route. This must be done before withdrawing the routes. A few moment later, at least the time of route propagation [37] the routes are withdrawn. This community does not imply the presence of an alternative route inside the AS. When a BGP speaker receives an update message from an external peer with this community, two choices are available to it. First, it can use policies, for example, to set the route to a lower preference. This would allow the

[1]The Internet Assigned Numbers Authority, `https://www.iana.org/assignments/bgp-well-known-communities/bgp-well-known-communities.xhtml`

alternative route to be pulled out before the route is withdrawn. The example at 3.4 shows policies that are BGP inter-AS relationship compliant as defined by Lixin Gao and Jennifer Rexford [21]. In this example, we set a local preference of 150 for customers, 100 for peers, and 50 for providers if they do not have the community PLANNED_SHUTDOWN. If they have the community, the local preferences are 149, 99, and 49 respectively. If the route having the community PLANNED_SHUTDOWN comes from a customer and no alternatives are coming from a customer, even if there is an alternative from a peer or a provider, it will prefer the route having the community PLANNED_SHUTDOWN. The same reasoning can be applied if the route comes from a peer or a provider. This respects the relationship between ASes as described in the Background section 2.1.4. The second approach one can take when receiving an update with the PLANNED_SHUTDOWN community is to ignore it. The operator may want to use the best route according to its policies until the withdrawal of it. The community does not define when the shutdown is planned. It can be in one minute or one day. The reactions depend on the choices of the ASes administrators and policies.

```
match from a customer and not have community 65535:5 :
  set local-pref 150
match from a customer and have community 65535:5 :
  set local-pref 149
match from a peer and not have community 65535:5 :
  set local-pref 100
match from a peer and have community 65535:5 :
  set local-pref 99
match from a provider and not have community 65535:5 :
  set local-pref 50
match from a provider and have community 65535:5 :
  set local-pref 49
```

Figure 3.4: Pseudocode representing policies

When a BGP speaker that implements our solution receives a withdraw message from an external peer for a route that previously was announced with the PLANNED_SHUTDOWN community, two cases can occur. The first is that the BGP speaker knows an alternative route. In this case, it removes the route and prefers the new one. In the second case, the BGP speaker does not know an alternative route. We have based on a mechanism used by Virgine Van den Schrieck et al. [5]: set the route with a local preference of 0 and advertise it only to iBGP peers. So,

in our solution, the BGP speaker does not remove the route. It keeps the previous route and sets its local preference to 0. It advertises this new local preference to all its iBGP peers. A local preference of 0 means that the route is no longer available. If a BGP speaker implementing our solution receives a route with a local preference of 0, it must not advertise this route to external peers, even if it is the best one. When the speaker sets the local preference to 0, it sets a timer. When the timer expires, the BGP speaker verifies if the route still exists in the RIB and the local preference is 0. In this case, it removes it. It sends withdraw messages to iBGP peers. If there is no alternative path or the alternative path is rejected by policies, it sends a withdraw message to them.

Implementation

We have implemented our solution with goBGP from OSRG [40]. It is an implementation of BGP in the Go programming language that supports route-reflector as defined in RFC 4456 [14].

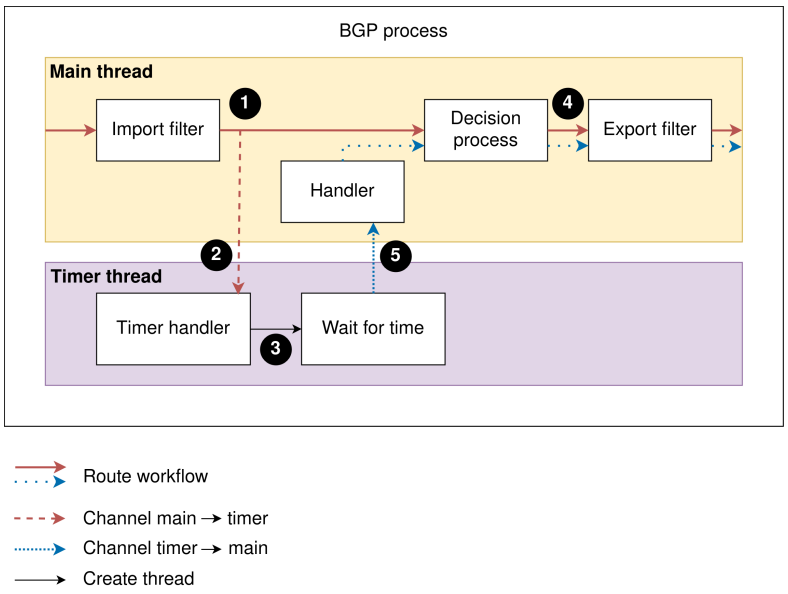


Figure 3.5: The workflow of BGP routes in goBGP

In goBGP, there is a main thread that handles BGP messages and route modifications. Figure 3.5 presents the architecture of BGP with our solution. When a route is advertised, it passes through the import filter, and then directly to the decision process. After the decision process ❹, it checks if the best route does not have a local preference of 0. If the local preference is 0, it does not export

the route for external peers. When a route is withdrawn, it passes through the import filter. Then, before being used by the decision process ❶, it checks if the route is withdrawn by an external peer and it was previously announced with the PLANNED_SHUTDOWN community. In this case, it keeps the previous route and sets its local preference to 0. It also sets the timer. To do this, we created a parallel thread that manages the timer. It communicates with messages through channels. This thread listens to arriving messages on a channel. The main thread posts a message ❷ that indicates the time to wait, a reference to the route, and the peer that has sent this route. This timer thread launches a thread ❸ that will wait for the defined time. Meanwhile, in the main thread, the old route with the local preference set to 0 is used in the decision process. When the timer is triggered, the timer thread communicates to the main thread ❺ the peer and the route that it has to check. It communicates through a channel with the main thread. The main thread listens to multiple channels for events. We added this channel to listen. When it receives the message from the timer thread, the main thread handles this message by verifying if the route exists and if the local preference is 0. If this is the case, it marks the route as withdrawn and rerun the decision process. It will remove the route from all RIB and send withdraw to peers who need it.

Limitations

This solution has some limitations and assumptions.

The local preference 0. No policy should assign a local preference to 0. The 0-value is reserved for our solution and has the meaning "the route is withdrawn". Using a local preference 0 outside our solution might have unwanted consequences. For example, the AS sets a local preference of 0 for all routes coming from providers. An AS must send all prefixes that it knows to its customers. However, if a BGP speaker attached to a customer knows only routes with the local preference 0 for a prefix, it will not advertise this prefix due to the rule that we set "no route with a local preference sets to 0 is sent to external peer". Not advertising these routes to customers results in breaking the AS-relationships defined by Lixin Gao and Jennifer Rexford [21].

The importance of the timer. The time for the timer to verify a route when we set its local preference to 0 has to be chosen carefully. A timer that expire too quickly can result in removing the route and sending a withdraw message while an alternative route exists but arrives later. Then we have the case of withdrawing a prefix and then reannounce it. However, a timer that expire very late can also be problematic. If no alternative route exists, it will delay the withdrawal for at least the time of the timer.

Multiple alternative hidden routes and policies. In cases of multiple alternative hidden routes, we can withdraw and then update a prefix to an eBGP peer. Let us illustrate the problem with the Figure 3.6. AS65001 is attached to two customers, AS65002 and AS65004, and two providers AS65003 and AS65005. AS65002, AS65003, and AS65004 announces the prefix 203.0.113.0/24. At convergence, R1 knows only one route through AS65002. RR1 knows two routes, the one via AS65002, the best, and the one via its provider AS65003. RR2 knows one route, the one via AS65002. And finally, R2 knows two routes, the one via AS65002, the best one due to the AS-PATH length, and its external via AS65004. We assume that AS65002 has previously announced the community PLANNED_SHUTDOWN. If AS65002 withdraws the prefix, it does not know an alternative path. It will send the route that it knows with a local preference of 0. RR1 knows an alternative route. It will then prefer it because the local preference is higher. It will send this best route to R1 and RR2. R1 learns this route and prefers it because the local preference is higher. However, as this route is learned by a provider, it can not send it to its provider AS65005. Hence, it sends a withdraw to AS65005 for the prefix. RR2 will learn that now RR1 prefers the route via its provider and the route via AS65002 does not exist anymore. RR2 will prefer this new route and announce it to R2. R2 learn this route via the provider. As the local preference is lower than its route via AS65004, it prefers and announces it to RR2. At its turn RR2 will prefer this route, then RR1, and finally R1. As this route is learned by a customer, it can send this route to AS65005. So, R1 announces an update for this prefix to AS65005. We can conclude that if there is multiple alternatives routes some of which can be sent to the eBGP peer and some not, the order of arriving alternative routes can produce a withdraw and then update.

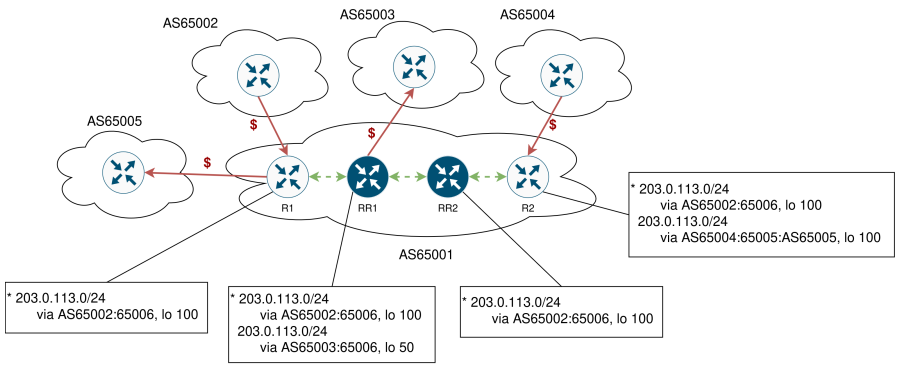


Figure 3.6: Example of topology showing the limitations of the solution

The impact on the data plane. When a BGP speaker receives a withdrawal for a prefix that has the community PLANNED_SHUTDOWN instead of removing it

from the RIB if there are no alternative routes currently known by the router, the route stays in the routing table. When the timer expires, either there is already an alternative route that has replaced the old one in the routing table, or there is no alternative route. In the first case, there is always a route in the routing table, even if the alternative route can not be sent to the eBGP peer and is withdrawn. No traffic is dropped. Indeed, between the time that the eBGP peer finds an alternative route or removes at its turn the route from the routing table, it still uses this peer to forward traffic to the removed prefix, and its peers have a route in its routing table, then it can forward packets for this prefix. In the second case, the router removes the route from the routing table and then sends a withdrawal to its eBGP peer. Between the received withdrawal and finding the alternative route or removing the route, the eBGP peer keeps forwarding traffic for the prefix removed to the router. This last will drop the packets because it no longer knows a route for this prefix.

3.5 Experiment

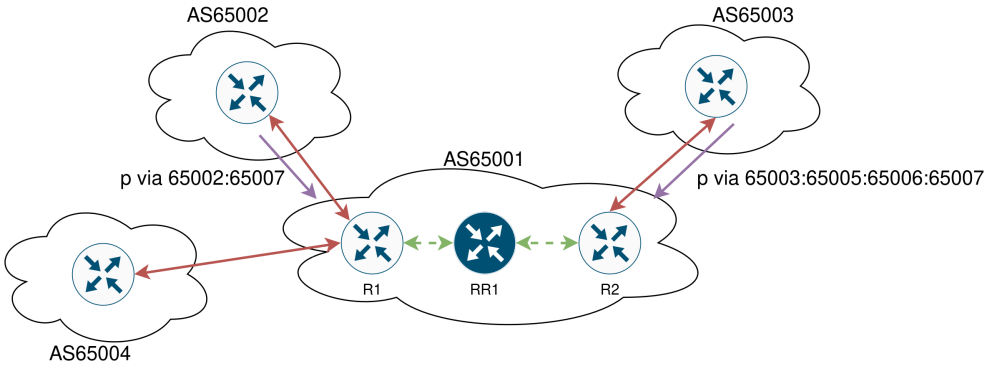


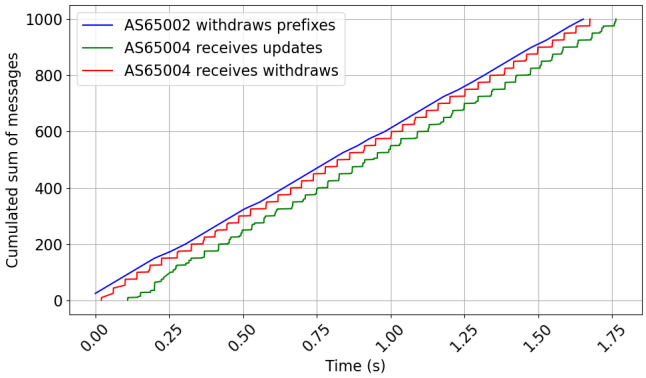
Figure 3.7: Topology for experiment

We have performed an experiment to compare the BGP messages received by an external peer when the associated peer receives a withdraw message for the only route that it knows. The scenario illustrated by the Figure 3.7 is the following. AS65001 implements our solution on each router. AS65001 is composed of two BGP speakers, R1 and R2, and one route-reflector, RR1. R1 and R2 are clients of RR1. AS65002 and 65003 are providers of AS65001. AS65004 is a customer of AS65001. AS65002 announces 1000 prefixes with the AS-PATH 65002:65007 and the community PLANNED_SHUTDOWN (65535:65285). AS65003 announces the same prefixes with the AS-PATH 65003:65005:65006:65007 and no community. We wait for convergence. At convergence, R1 does not know the path via AS65003 because

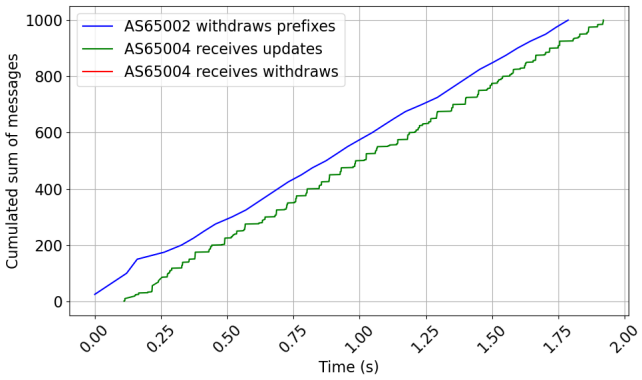
RR1 and R2 prefer the path via AS65002 due to the shortest AS-PATH. In goBGP, R2 announces its external best path to the route-reflector. It implements the "Best external" feature, which allows a BGP speaker to advertise its best external path, i.e., a path learned by an eBGP peer, in addition to its best path to an iBGP peer [41]. That is why RR1 knows the alternative path. Then, AS65002 withdraws all prefixes. On BGP speakers implementing our solution, we set up a timer of 1000 ms. It ensures that if no alternative route is found, the route is removed. We monitored messages received by AS65004 when AS65002 withdrew its prefixes and until convergence. We have set a delay of 20 ms on each link with the tc (traffic control) [2] command. To monitor BGP update and withdraw messages, we have used tcpdump [3] to capture all TCP packets on the port 179 used by BGP. We repeated the experiment five times. We also have performed this experiment with AS65001 BGP speakers that did not implement the solution. The experiment took place on a server running on Debian with 20 cores and 128GB RAM. We used containerlab [4] to create our topology virtually.

Let us look at a timeline view of BGP announcements and withdrawals when AS65002 withdraws all prefixes. Figure 3.8 compares the numbers of BGP messages sent and received according to the time and the AS. We can observe that without our solution, 1000 withdraws are sent, one per prefix, and then 1000 updates are sent. While, with our solution, only 1000 updates are sent with the alternative route. No withdraw messages are received by AS65004. For each iteration of the experiment, the timeline is quite similar. 1000 withdraws are sent without our solution before the 1000 updates with the alternatives route. Appendix D summarizes all the timelines for each experiment.

[2] `https://man7.org/linux/man-pages/man8/tc.8.html`
[3] `https://www.tcpdump.org/manpages/tcpdump.1.html`
[4] `https://containerlab.dev/`



(a) Without the solution



(b) With the solution

Figure 3.8: Timeline representing the cumulated sum of messages over the time

Another interesting point is the delay between AS65002 withdrawing a prefix and AS65004 receiving the update for this prefix with the alternative path. Figure 3.9 compares the cumulated number of prefixes according to the delay of the five iterations with and without our solution. At first sight, we can observe that with our solution, the alternative routes arrive a bit slower. With our solution, for 78% of prefixes, the alternative routes arrive within a delay under 125 ms, while without our solution, around 93% of routes arrive within a delay under 125 ms. The maximum delay without our solution is around 180 ms. With our solution, 96% of the routes have arrived within this delay. The four remaining percent arrive with a delay between 180 and 270 ms. Our solution avoids sending withdraw messages. However, it slightly slows down the propagation of the alternative route.

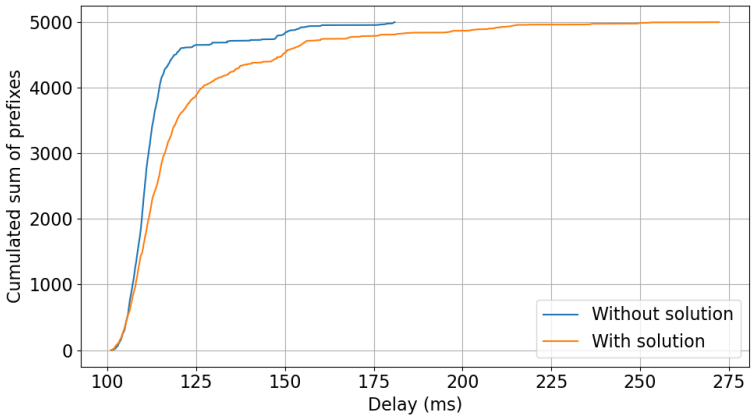


Figure 3.9: Cumulated number of prefixes according to the delay

Table 3.1 and 3.2 show the mean, median, and standard deviation in ms per iteration. We can observe that, without our solution, as expected from the previous graph, the mean and the median are lower than with our solution. We can also see that with our solution, the standard deviation is higher than without. Indeed, without the solution, the median is around 110 ms, while with our solution, it is between 112 and 116 ms, which is higher and varies more. For the fourth iteration with our solution, we can observe a higher standard deviation and a higher mean than the other iterations. This outlier might explain the 4% of routes having a delay between 180 and 270 ms in the previous graph 3.9. We can conclude that with our solution, even if it avoids sending withdraw messages, it increases for a few milliseconds the delays of the arrival of the alternative routes, and this delay varies more.

Iteration	Mean (ms)	Standard deviation (ms)	Median (ms)
1	112.844	10.125	110.643
2	113.151	10.566	110.948
3	112.794	10.633	110.242
4	111.657	8.251	110.114
5	115.787	16.21	110.935

Table 3.1: Mean, standard deviation, and median per iteration without our solution

Iteration	Mean (ms)	Standard deviation (ms)	Median (ms)
1	121.559	18.296	115.88
2	115.737	11.567	112.511
3	115.505	13.286	112.233
4	135.281	39.21	116.506
5	118.124	13.532	114.05

Table 3.2: Mean, standard deviation, and median per iteration with our solution

3.6 Discussion

In this chapter, we have seen that it is possible to avoid sending BGP withdrawal messages to eBGP peers in the context of a planned shutdown under some conditions. Our experiment has shown that with this solution, when a route receives a withdraw message for a prefix for which it only knows one route, and there is one alternative hidden route in an AS that is not rejected by export policies, no withdrawals are sent to eBGP peers of this router. However, we have noticed that the propagation time of the alternative routes is higher than without the solution. In the case of multiple alternative routes with different policies, we have seen in the limitations that this solution might not work and have the same behavior as without. However, the goal of this chapter was to show that we can influence the BGP process by informing it that the data plane will be shut down soon, but the prefix remains available by other routes on the Internet. In further work, we will discuss how we can benefit more from this information to avoid dropping packets in the data plane.

Chapter 4

BGP Route Quality Measurement

Nowadays, the decisions taken by BGP are based on the control plane. However, ASes operators can manipulate data exchanged in it for traffic engineering purposes [7, 8]. They can modify every attribute of a path. It leads to selecting the best route among routes where the attributes are manipulated without taking the data plane into account. This chapter aims to add knowledge of the data plane to the BGP decision process. We want to take into account the quality of a route, measured in the data plane, into the decision process.

The following section introduces the motivation to make BGP data-plane aware. Then, we present a brief review of the literature. Next, we present the design of our solution. We explain the mechanisms, our choices, and our assumptions to modify BGP to make it aware of the data plane. Following that, we conducted some experiments to prove the correctness and performance of this solution.

4.1 Context

Currently, the decision process of BGP is based on control-plane information. One of the first tie-break rules is the AS-PATH length of a route [2]. However, for traffic engineering purposes, this AS-PATH can be prepended [7, 8]. Marcos, Prehn, et al. [42] showed, in 2020, that 25% of the route advertised for IPv4 prefixes was prepended by the origin AS. The AS-PATH length of a route does not represent the true length between the AS that originated the prefix and the BGP speaker receiving the route. No rule from the decision process takes into account the data-plane information. Therefore, we propose to add a measurement made in the data plane to quantify the quality of a route and use it in the decision process. Quality is a subjective metric. It can be the delay [43], the number of lost packets [44], the TCP throughput [45], or many more measures. In this master thesis, we focus on the delay.

Let us take a small example with the topology in Figure 4.1 to illustrate a practical case where measuring delay could be useful. AS65000 wants to be connected to the Internet. AS65001 and AS65002 are providers of AS65000 and are connected to the rest of the Internet. AS65000 does not need to advertise prefixes. It only needs to have the routes with the lowest delay. Let us assume that AS65001 announces the prefix 10.0.0.0/24 with the AS-PATH 65001:65003, and the mean delay to reach the prefix is 60ms. AS65002 advertises the same prefix but with the AS-PATH 65002:65004:65003, and the mean delay to reach the prefix is 15 ms. Due to the decision process, the route through AS65001 is selected and said "the best route". However, for AS65000, the "best route" is the one with the lowest delay, so the route through AS65002. In this chapter, we propose a solution to avoid choosing a route with poor quality, in this scenario case, the delay.

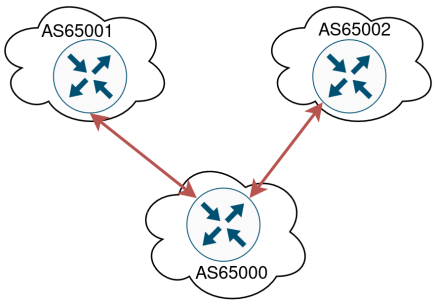


Figure 4.1: A motivating topology

4.2 Related work

In 2021, Maria Apostolaki et al. proposed RouteScout [46]. It is a system that monitors the performances of routes in the data plane and reroutes traffic through routes with an equivalent policy. This system is compatible with BGP. It replaces arbitrary tie-break rules, such as selecting the route with the lowest BGP Identifier Value, with information about the performances of the data plane while ensuring that policies are respected. This system requires no change in BGP.

Thomas Wirtgen and Olivier Bonaventure [9] have introduced in 2022 the idea of adding knowledge of the data plane directly in BGP. They propose to check if a route is reachable in the data plane before adding it to the routing table. To verify a route, they opted for a decentralized solution. They have created a new type of service, the validation server. Inside each AS, a validation server is available, and each router can contact it to check the route reachability. To verify a route, they use one IP address from the route prefix. They establish a TLS session with this

IP address. If the establishment was a success, the route is validated and added to the routing table. However, if it has failed, the route is invalid and not added to the routing table.

Ryo Nakamura et al. [47] have measured BGP Egress Peer Engineering, BGP-EPE [48]. In a nutshell, BGP-EPE uses segment routing. A BGP-EPE router represents eBGP peers as segments with a unique ID. Packets for a specific eBGP peer are encapsulated with information about the corresponding segment ID. When a border router receives an encapsulated packet, it decapsulates it and sends it to the specified eBGP peer. Instead of using the path attributes to select the best peer to forward, it uses information carried in the packet. The route used may be different compared with the route that would have been preferred by BGP without EPE. In their study, Ryo Nakamura et al. [47] have shown that 77% of prefixes have a better latency with BGP-EPE. They have also compared the round-trip time (RTT) between the best route and the minimum RTT between all alternative routes. They have demonstrated that the alternative path has a better latency than the best path.

4.3 Design of a data-plane aware BGP

4.3.1 Global design

To begin, we will give a brief overview of the architecture of our solution. Figure 4.2 shows a global view of it.

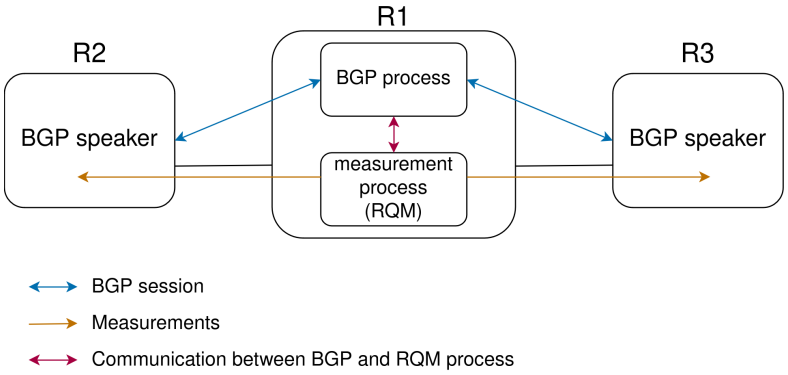


Figure 4.2: Global design of the solution

We have a router R1 that implements our solution. It runs two processes: the BGP process and the measurement process. The BGP process has BGP sessions with other BGP speakers, R2 and R3 in this figure. The measurement process

performs measurements in the data plane passing through these peers. These two processes communicate together. The BGP process asks for the quality of a route, and the measurement process makes the measurement and provides it. We called our measurement process the Route Quality Measurement or RQM. Every BGP speaker that wants to take quality into account is equipped with a system for taking measurements. Ideally, this system should be installed on the same router to get measurements as if the BGP process had done it.

Making a measurement is similar to the problem of validating a route with RPKI. We want to use a piece of information about the route to select the best one, but the BGP router does not initially have it. With the RPKI, we want to know if the route is valid, unknown, or invalid, and with the measurement, we want to learn the quality of the route. To validate a route, a BGP speaker loads the set of valid ROAs from a local cache [31] and uses them to know if the route is valid, unknown, or invalid. Due to the similarity of the need, i.e., accessing information that a BGP router does not initially know, we decided to create a system outside the BGP process that takes the measurements and communicates the result to it. The measurement system gives the quality as the local cache provides the validity of a route. Moreover, having an external measurement system provides adaptability and the ability to choose the metric to represent the quality, such as the delay [43], the number of lost packets [44], or the TCP throughput [45].

In a network, prefixes do not have the same importance. Numerous prefixes handle only a slight proportion of the traffic, while few prefixes handle a large amount of the traffic. In 2022, in France, 54% of the traffic to the customers of the main Internet Service Providers (ISPs) came from five companies: Netflix, Google, Akamai, Meta, and Amazon. The remaining 46% traffic was generated by plentiful actors who interconnect with the ISPs [49]. Operators usually know which prefixes are the most important. For this reason, we decided to allow operators to choose which routes they want and which routes they do not want to measure. By measuring only the small proportion of routes that have the most important prefixes, a good proportion of traffic can use the route with the best quality, and it reduces the number of measures by making only the most important one. In this way, the cost of the measurement in proportion to the traffic used for this prefix is worthwhile. This also allows scaling up, as it is not efficient to measure all the prefixes received, even those that do not generate traffic.

The quality can change over time. For example, in case of congestion, the delay can increase. Consequently, we decided to measure the quality periodically. We designed a subscribe-unsubscribe mechanism between the BGP process and the RQM process. The BGP process must subscribe the route to the RQM to receive

its quality. The RQM periodically measures this route and notifies the quality to the BGP process. If the latter wants to stop receiving the quality for a route because, for example, it becomes unavailable, it unsubscribes it. The RQM process stops the periodic measurement and the notifications for this route. Two routes can have different intervals. The duration is requested at the subscription. We recommend at least 10 min or 1 hour between two measurements for the same prefix according to the possibility of quality variations. The period chosen by the operator must be selected to not overload the data plane by making too many measures.

The quality given is not a value from the measurement but a ranking place between all routes with the same prefix. We have chosen this design for two main reasons. The first is due to the large number of metrics that can be used to quantify the quality. Each metric has a different meaning and must, therefore, be interpreted differently. For example, if we measure the delay, we want to have the lowest delay, so the lowest number. However, if we measure the throughput, we want to have the highest throughput, so the highest number. A ranking allows having the same meaning regardless of the metric. The route with a ranking place of 1 has the best quality, while the route with the highest ranking number has the lowest quality. The second reason for this design is to take into account the variability of measurement results. These variations can create oscillations that could cause route flaps. Let us illustrate the problem with an example. We have two routes with nearly the same delay. After a first measurement, the mean (or median) delay of the first route is 1 ms higher than the delay of the second route. Later, at a second measurement, the mean (or median) delay of the first route is now 2 ms lower than the second route. The second route that has a better quality in the first measurement becomes gets a lower quality. In the following measurements, it can become again better. By making a ranking, we can take into account these little variations and consider these two routes as equal in terms of quality. We need more than one measure to be representative while comparing two routes. With multiple measures for each route, we can use a two-sample t-test, for example. It allows knowing if the two routes can be considered equal or not or if one route is better than the other.

As we make a ranking between routes from the same prefix, we decided to measure all routes from the same prefix before making the ranking. The measurement of a prefix consists of performing the quality measurement for each route of this prefix and then making the ranking. All the new qualities are sent to the BGP process. We decided to do this because if the ranking changes, all the new positions must be notified. If we measure route by route and send the new qualities each time, it increases the number of quality changes per route, and the best route might change

more often. Furthermore, comparing the results of a measurement made in the past with a measurement made now is not representative. The measurement made in the past is potentially no longer valid.

4.3.2 Protocol

This part aims to show how the Border Gateway Protocol is adapted to support and take into account the qualities, how the BGP and the RQM process communicate, and how the RQM must behave.

In the BGP protocol, we add a new optional non-transitive attribute of type 41 for the quality. The type 41 is unassigned at the IANA [1]. This attribute contains a value that represents the quality.

We modified the BGP decision process to take into account this attribute. Here is the new tie-break rule and its order:

1. Remove all routes route with an unreachable BGP next hop.
2. Keep routes with the highest local-pref attribute. Remove all others.
3. **Keep routes with the highest quality (lower number) and the ones with no quality. Remove all others.**
4. Keep routes with the shortest AS-PATH. Remove all others.
5. Keep routes with the lowest origin number (IGP over EGP over incomplete)
6. Keep routes with the smallest MED. Remove all others. MED can only be compared between routes learned from the same neighboring AS.
7. If at least one route was learned via eBGP session, remove all routes learned via iBGP session else, keep all routes.
8. Keep routes with the closest next hop. Remove all others.
9. Keep routes learned via BGP speakers with the lowest BGP Identifier Value. Remove all others.
10. Keep the route learned from the lowest peer address.

We have chosen not to define a default quality. It is an attribute that represents a ranking between routes. If the route does not have the quality attribute, we can not determine if it has a better quality. We put this rule after the local-pref tie-break rule because the local-pref attribute represents the degree of preference of a route in an AS and must remain the most important attribute. We put the quality before the AS-PATH length tie-break rule because, as presented in the introduction, the AS-PATH length is not representative of the real route length. The quality can give a better estimation of the path length. For example, the higher the delay is, probably

[1]The Internet Assigned Numbers Authority, `https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-parameters-2`

higher the route length is. Moreover, the shortest path can have worse quality than the longest path due to several reasons, such as congestion, high latency, ...

Two pieces of information are needed to uniquely identify a route. The first is the prefix, and the second is the peer that has announced the route, which we also call neighbor. We assume that each neighbor is identifiable by a unique ID, its IP address, for example. To communicate with the RQM, we decided to only use this information to distinguish each route.

Subscribing and unsubscribing a route require some conditions. Only an advertised route can be subscribed. A new subscription with different parameters for a route that BGP has already subscribed to replaces the old parameters with the new ones. Unsubscription must be done if a route with a subscription is withdrawn. An unsubscribe message for a route can only be sent if the route has been previously subscribed. We let the possibility to subscribe for a route for a limited time. When the subscription expires, the BGP process does not need to unsubscribe. However, the RQM process has to stop measuring and notifying this route. It also has to send a notification to advertise that the subscription of the route has expired. A quality message contains the quality metrics for every subscribed neighbor for the prefix and does not contain the quality metrics of neighbors who are unsubscribed or whose subscription has expired.

We designed a protocol to communicate between the BGP and the RQM process. It uses TCP as the transport protocol. We have chosen TCP to be able, later, to use TLS to secure the connection. We did not use TLS in this master thesis because this solution is a prototype to prove the possibility of adding information from the data plane in BGP. Security risks and improvements are discussed as further work. For the port, we choose the number 4, an unasigned port at the IANA [2]. The JSON format is used to structure the packets content. This format, rather than a binary format, makes it easier to adapt the content of the package in further work. A modification only requires the addition or removal of fields instead of having to modify the binary parser and rethink the packet shape. If no fields are removed in the new version, it allows compatibility with older versions by ignoring added fields. That would be more difficult and depend on the old implementation with binary packets.

Each packet contains a field "type" to represent the type of information transmitted inside. There are four types: subscribe, unsubscribe, quality, and expired_timer. The terms IP prefix and IP address represent an IPv4 or IPv6 prefix and an IPv4 or

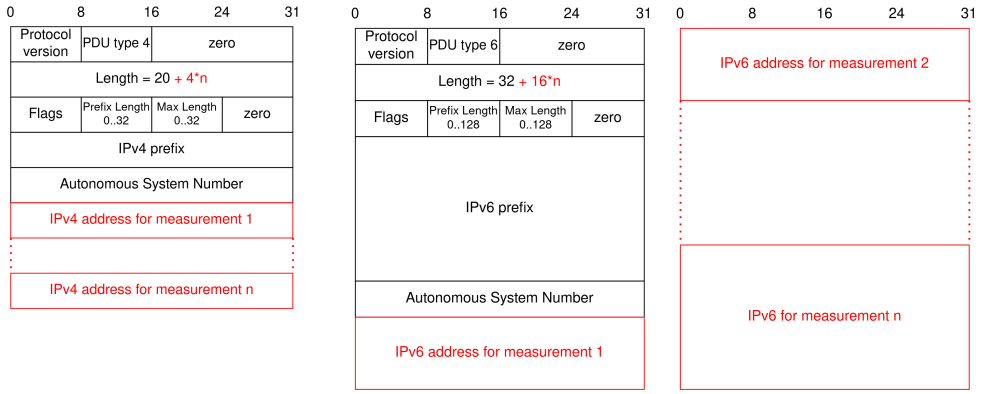
[2]The Internet Assigned Numbers Authority, `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt`

IPv6 address. This protocol supports IPv4 and IPv6. Each neighbor is identifiable by a unique ID, its IP address, for example.

- **subscribe**: allows BGP to ask for quality measurements for a specific neighbor to a specific prefix
  - prefix: A string that represents the IP prefix to measure.
  - neighbor: A string that represents the neighbor to measure.
  - measurementIPs: A list of strings that represents the list of IP addresses that can be used to measure.
  - timeLimit: The timestamp in seconds at which this subscription is no longer valid. 0 to be always valid.
  - samplingInterval: The time in milliseconds between two measurements.
- **unsubscribe**: allows BGP to remove quality measurement for a specific neighbor to a specific prefix.
  - prefix: A string that represents the prefix to measure.
  - neighbor: A string that represents the neighbor to measure.
- **quality**: allows the RQM process to communicate the qualities
  - prefix: A string that represents the prefix measured.
  - neighbors: A list of neighbors and their ranking. The list is composed of tuples `(<neighbor> string, <rank> int)`
- **expired_timer**: message sent by the RQM to advertise that the subscription of a neighbor for a prefix has reached the validity end.
  - prefix: A string that represents the measured prefix.
  - neighbor: A string that represents the measured neighbor.

In the subscribe message, the BGP process has to provide a list of IP addresses that are available to make the measurement. We decided to use RPKI to bring this information. For the prototype, we modified the ROAs by adding a field "measurementIP" that contains a list of IP addresses from the prefix available to make the measurement. We decided to use the RPKI and modify the ROAs for legitimacy and security purposes. As the ROAs are cryptographically signed object that allows an IP address space holder to authorize an AS to advertise a prefix from this address space [3]. It is best placed to authorize IP addresses that can be used for measurement purposes. As we have modified the ROAs, we have to modify the

RTR protocol to support this new field and communicate the available IP addresses for a prefix to routers. For that, we had to modify two types of PDU: The IPv4 prefix PDU and the IPv6 prefix. The modifications are quite similar between these two PDUs as their role are the same, only the type of address changes. We added at the end of the packet, the list of IP addresses available for the measurement. Figure 4.3 shows the modifications in red. As an IPv4 Prefix PDU always advertises IPv4 prefixes, the IP addresses for the measurement, which are among this prefix, will always be IPv4 addresses. Then, we can assume that every address is 4 bytes long. The same reasoning and assumption can be made for IPv6 PDU with 16 bytes prefixes and addresses. As we have added information in the PDUs, we have to update the length file from the header. We have to add the size of the address (4 for IPv4 or 16 for IPv6) times the number of addresses added.



(a) PDU IPv4 Prefix (b) PDU IPv6 Prefix

Figure 4.3: Structure of the IPv4 and IPv6 Prefix PDU

The interval between two measurements for a prefix is computed among the list of neighbors and takes the lowest sampling interval time. This interval is recomputed at each subscribe and unsubscribe for the prefix.

4.3.3 Implementation

BGP

We decided to reuse the goBGP implementation from OSRG [40] presented in the previous chapter to add the quality.

We created the quality attribute with an uint32 to represent the quality. As this attribute is local to the router, it is automatically removed before sending it to a peer.

We added in the configuration the possibility to add RQM configurations. It consists of the address and port that an RQM process is listening to and a set of rules used to know which routes need a measurement and which parameters have to be used. Pseudocode 4.4 shows an example of RQM configurations in the TOML format.

```
[[rqm-server]]
  [rqm-server.config]
  address = "127.0.0.1"
    port = 4
  [[rqm-server.rule]]
    time-limit = 0
    community = "65000:100"
    sampling-interval = 3600000
  [[rqm-server.rule]]
    time-limit = 0
    community = "65000:200"
    sampling-interval = 86400000
  [[rqm-server.rule]]
    time-limit = 1728518400
    community = "65000:300"
    sampling-interval = 864000000
```

Figure 4.4: Example of RQM configurations

To verify if a route is to measure, we decided to use communities. At the import, communities that say if a route needs to be measured or not are added. As the measurement choice is specific to a BGP speaker, communities for measurement are chosen by the operator and should be removed before advertising the route to a peer. We have to provide two pieces of information while subscribing: the time limit and the sampling interval. Consequently, we decided to use multiple communities. Each community is associated with a time limit and a sampling interval. To configure these communities, the operator has to configure a set of rules. We uses pseudocode 4.4 to shows an example of rules. We assume that only one community of this set is present in the list of communities of a route. If there are multiple communities from this set, the first matched community is the one used for the parameters. Let us explain the rules in the pseudocode configuration 4.4. In this set of rules, if a route has the community 65000:100, the time limit is 0 (no time limit), and the sampling interval is one hour. If the route has the

community 65000:200, the time limit is 0 (no time limit), and the sampling interval is one day. If the route has the community 65000:300, the time limit is set to 10 September 2024 at 00:00:00 UTC, an arbitrary date in the future when this manuscript is written, and the sampling interval is ten days. If a route does not have one of these three communities, no subscription will be made for this route.

In goBGP, we have a main thread that handles BGP messages. Figure 4.5 presents the architecture in goBGP. We have created a thread, the RQM thread that handles communications with the RQM process. This thread is inspired by the RPKI threat already present that handles communication with the local cache and from our previous solution. The RQM thread establishes the connection with the RQM process. It listens and handles messages that come from the RQM ❹ and sends the subscribe and unsubscribe messages to it ❸. In Go, threads can communicate through channels that are structure robust to concurrency. We use channels in one way. One thread puts data on the channel and the other listens continuously for messages. Our RQM thread monitors one channel to listen for subscriptions and unsubscriptions that come from the main thread ❷. The main thread monitors multiple channels for arriving messages and handles one message at a time. We add one more channel through the RQM that can send the quality information to the main thread ❺. We choose to modify the RIB only in the main thread to avoid memory concurrency.

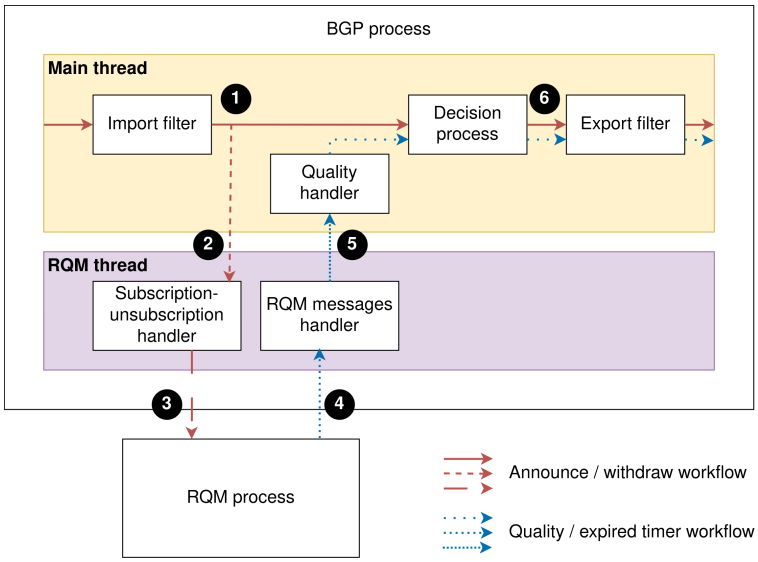


Figure 4.5: The BGP workflow

We keep the subscriptions in a hashmap with a string as key that concatenates the prefix and the neighbor address and as value the community, sampling interval,

time limit, and a pointer to the route.

Let us present the workflow when a route is advertised or withdrawn with the figure 4.5. First, routes pass the import filter. Then, at ❶, it checks if the route is an announcement or a withdrawal. If it is an announcement, it checks if there exists an RQM service, one or multiple IP addresses for the measurement available, and at least one community. If this is the case, it sends a message to the RQM thread through the channel with the prefix, the neighbor IP, and a pointer to the route ❷. In the main thread, it adds the quality to the maximum uint32 and passes the route to the decision process. We will explain later why we set the quality to the maximum. In the RQM thread, it checks if the route does not already have a subscription in the hashmap subscription and has a community that indicates the request of measurement. If this is the case, the RQM thread sends a subscription message to the RQM process ❸ and saves the subscription in the hashmap. We go back to ❶ and see the workflow of a withdrawn route. For every withdrawn route, it sends a message with the prefix and the neighbor IP address to the RQM thread through ❷. Noting more is done in the main thread and the route passes to the decision process to be removed. In the RQM thread, it checks that no subscription has been made for this route. If there exists a subscription, an unsubscribe message is sent to the RQM process ❸. Now, let us see the workflow after the decision process ❻. If the best route has the quality attribute and the value is the maximum uint32, we set it to 1. Then, it passes the export filter.

The choice to set a quality of maximum value and then set it to one for the best route is to avoid converging multiple times before receiving the real qualities from the RQM. Let us take an example represented in figure 4.6 to illustrate the problem. We receive three routes for the same prefix. The route arrives nearly at the same time. The first route that arrives has an AS-PATH of length 3, the second one of 4, and the third one of 2. The quality is 2 for the first arriving route, 1 for the second, and 3 for the last. But, because the qualities are not available before the measurement and the attribute quality is not set, we prefer in a first time the route of length 3, and then when the route of length 2 arrives, it is preferred. When qualities are available, the route with the length of 4 is preferred. By setting the quality to the maximum and the best route to 1 before the qualities arrive, we can avoid that. When the first arrives, its quality value is set to maximum. As it is the only route, it will be the preferred one. Then, the quality is set to 1. When the second route arrives, the quality value is set to maximum. It still prefers the first route because the quality is better. And the same when the third route arrives. Only when the qualities are notified by the RQM the route with the best quality is selected. In this scenario, we assume that the local preferences are equal.

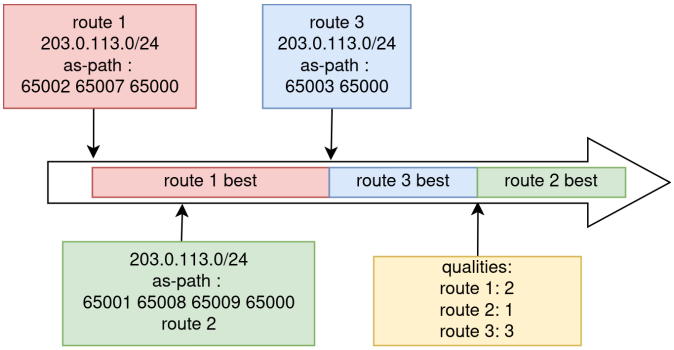


Figure 4.6: Timeline of multiple update messages

Now, let us look at the workflow when the RQM process sends a message. Figure 4.5 is again used to show the workflow. When qualities or timer expirations are notified by the RQM process, the RQM thread receives them ❹. If the message is a quality message, the RQM thread checks whether each route still exists. Then, it sends the resulting qualities to the main thread through the dedicated channel ❺. The main thread updates the qualities for each route, and if they have changed, the decision process is rerun. If the message is an expired timer, the RQM thread removes the subscription from the hashmap. We decided to remove the quality attribute for this route. For that, we advertise a quality of 0 for this route to the main thread through the channel ❺. We used the value 0 because the ranking place from 1 to maximum integer can be used. Hence, we decided that a quality of 0 means that the attribute has to be removed. In the main thread, if the quality is 0, the attribute is removed, and the decision process is rerun.

RPKI

We modified the binary parsers from the goRTR [50] and goBGP [40] implementation to add the list of measurement IPs at the end of IPv4 and IPv6 PDUs. We also added the field "measurement IPs" in the ROAs.

RQM

As we worked with the Go language for the BGP implementation and for RTR, we have decided to use Go for the RQM process. However, we could have chosen another language, such as Rust or C.

Figure 4.7 shows the global architecture of the RQM process.

The RQM server listens for connections on an address and a port defined in parameters. Once established, we launch a thread that listens for messages. All

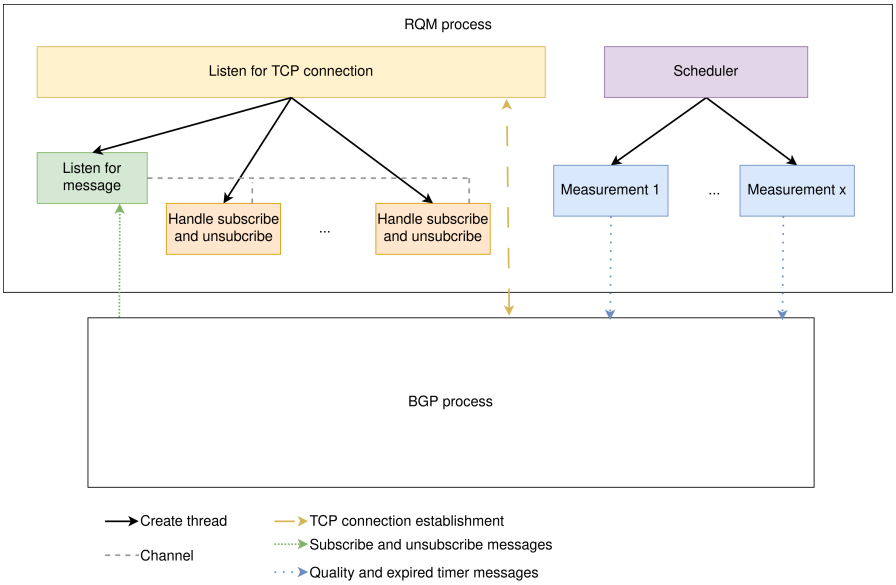


Figure 4.7: The RQM architecture

these messages are sent through a channel. One or multiple threads are listening to this channel and handle subscribe and unsubscribe messages and ignore all other messages. The number of threads is defined in the configurations.

We stock all information in a hashmap. The keys are the prefixes to measure, and the values are a structure that contains information for the measurement: the interval, the neighbor with their information that consists in the time limit, the sampling interval, the list of the results of the measurements done and the list of IP addresses available to make the measurement. The structure that contains this information (not the complete hashmap) is locked for subscription, unsubscription, and measurements. That is the reason why we recommend using multiple threads to handle subscriptions and unsubscriptions. When the structure for a prefix is locked for a measurement, it can take time to test each neighbor.

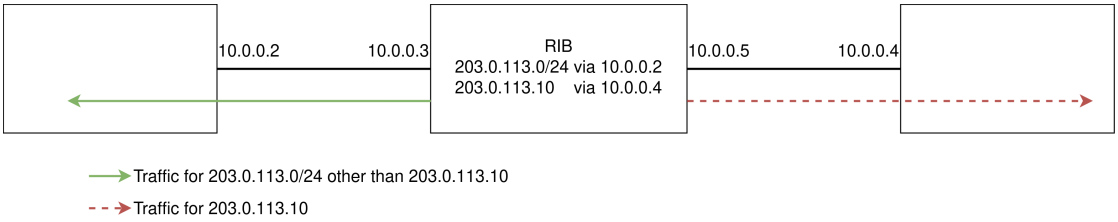
We designed a scheduler mechanism to launch the measurements. In the first instance, we have opted to use one thread per prefix. These threads waited for the period and then performed the measurement in an infinite loop. Due to the large number of threads that implied, we designed a principle of scheduler. This scheduler is launched in one thread. It checks every 100 ms if a prefix needs to be measured. Each prefix measurement is a task that is put in a priority queue. This priority queue is used to retrieve the next measurement to be performed and is ordered according to the next runtime. If the prefix is to be measured, the scheduler launches a thread that makes the measurement and, at the end, reschedule the

measurement.

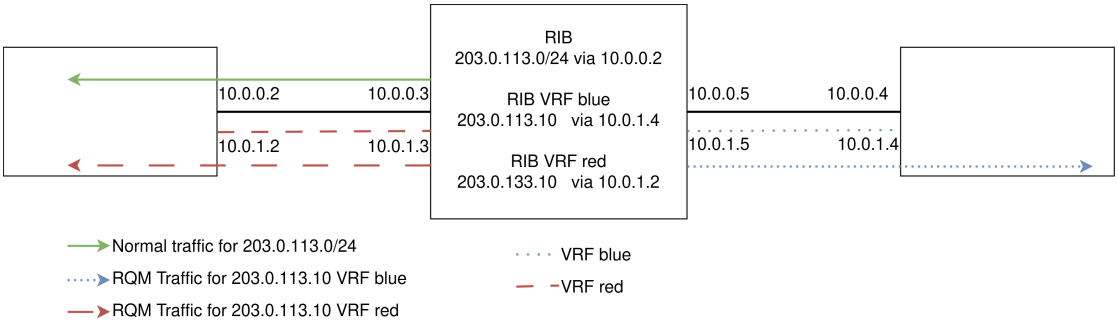
In this master thesis, we decided to use the delay as the quality metric. It is an important metric that can be measured by many means, and in particular, the ping command [3]. The RFC 2681 [43] defines the metric for the round-trip delay, and ping corresponds to the definition. It sends ECHO REQUEST packet and waits for ECHO REPLY to this packet. It measures the round-trip delay by measuring the time taken between the request and reply. We execute 3 pings, and we put the result in the structure with the measurement information. We remove the x older measure to have at most the last y measures. The y is defined in configurations.

To measure a specific neighbor, we have to put in the RIB of the router a new temporary route which indicates that the measurement traffic destined to this IP address must be routed to this peer. However, this causes problems. We do not want all packets with this IP address to pass through this neighbor. The normal traffic must be routed to the peer selected by BGP which has the best path. Only the measurement traffic destined to the IP address must be routed by the peer. To achieve this, we use VRFs. Each neighbor is associated with a VRF. All information about VRFs and interfaces to use for each neighbor are given in the parameters of the RQM process. Figure 4.8 shows an example without and with VRFs. Without VRFs, all the traffic for the IP address 203.0.113.10 passes through the right neighbor whereas normally all traffic for the prefix 203.0.113.0/24 must be directed to the left neighbor. By making measurements, we do not want to disturb the traffic by diverting it to another route than the current best route given by BGP. With VRFs, the normal traffic is directed to the right neighbor, the current best route given by BGP. For measurement, we use VRF. To measure the right neighbor, we use a blue VRF where we put the IP address. Normal traffic does not use this VRF, however, the RQM must use this VRF to make the measurement. To ensure that the measurement goes to the right neighbor, each has its own VRF. To measure the left neighbor, we use the red VRF.

[3] `https://linux.die.net/man/8/ping`



(a) Example of traffic without VRFs



(b) Example of traffic with VRFs

Figure 4.8: Example of traffic with and without VRFs

To make the ranking, we use the two-sample t-test described in the background 2.4. As a reminder, the null hypothesis is that the two means are equal. To reject the null hypothesis, we verify if the p-value is lower than the alpha and if this is the case, we check if the hypothesis "one sample is greater than the other" is valid. The rejecting hypotheses allow the ranking of two samples. If the null hypothesis is not rejected, the two samples have the same rank. To rank our neighbors, we first compare each of them two by two and perform the t-test. Then, we sort them by creating a comparator that uses the result of the t-test. We choose the two-sampling t-test for two reasons. First, it allows taking into account little variances. Second, we ran t-tests with real-life values, which showed us that it was worth using these.

Two-sample t-test on real data. Maxime Piraux et al. [51] have compared the latency of IPv4 and IPv6 request completion and put in three categories, IPv4 is better, IPv6 is better and none are strongly better. For that, they have used a t-test on data from RIPE Atlas [52]. RIPE Atlas [52] provides public measurements and the results are available via Google Big Query [53]. Our goal is to see if the latency between two routes in the same BGP speaker can be used to discriminate them. For that, we have created clusters of five kilometers. Inside these clusters, we compare two by two the latency of each probe to open a TCP connection, sending

an HTTP/1.1 GET request and receiving a 4KB HTTP response toward the same IP address. The hypothesis of the t-test consists of the mean of the two samples being the same. It is rejected if the p-value is lower than 0.05. We retrieved all HTTP completion times on June 3, 2024. Appendix E gives more details of the SQL query used in Big Query. We have grouped the data in sets of two hours, and we performed the t-test on each two hours group. Figure 4.9 shows the percentage of the hypothesis rejection among all clusters per two hours. We can observe that the percentage varies between 59.96% and 63.69%. We can say that more than half of the routes can be distinguished by the latency. Small variations over time comes from the fact that not all measurements are repeated every two hours. The number of clusters and t-tests performed is not the same, but relatively close. Appendix E gives more details on the number of clusters and t-test measurements available for each two hours.

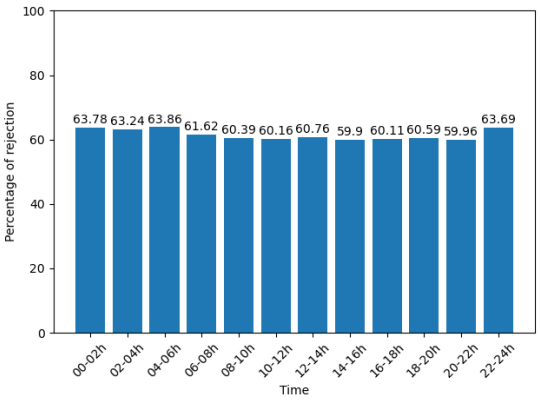


Figure 4.9: Percentage of t-test rejection grouped by two hours

4.4 Experiments

This part aims to show two experiments. The first experiment shows the functioning of BGP with the route quality measurement and points out the differences with the original BGP. The second experiment shows the time taken by different parts of the RQM implementation. These two experiments took place on a server on Debian with 20 cores and 128Go RAM. We used containerlab [4] to create our topology virtually.

[4] `https://containerlab.dev/`

4.4.1 First experiment

Our first experiment aims to show the global functioning with the modified BGP, RPKI, and our created RQM. Figure 4.10 shows the topology used. The AS of interest is AS65001. It implements our solution. AS65002 and 65003 are provider ASes. They run above exaBGP [5]. They will announce routes for our scenario. AS65002 always announces prefixes with the best path: 65002:65005, while AS65003 announces the same prefixes but with a longer AS-PATH: 65003:65007:65006:65005. We manage to have different path qualities. For 35% of prefixes, AS65002 has a better path quality, AS65003 is better for 35% of prefixes, and the quality is the same among the 30% remaining prefixes. To simulate the different delays, we use three routers that are only there for pinging. Two of them are directly connected to the router running BGP and RQM. They simulate the delay for routes. We created two VRFs, a blue and a red. The blue is associated with the routes coming from AS65002 and is connected to host 1. The red is associated with the routes coming from AS65003 and is connected with host 3. We set a delay of 10 ms with a jitter of 1 ms with the traffic control command, tc [6], between each host and the hosts and the router. To control the delay for each prefix, we assign a range of IP addresses to each host. Host 1 answers for ping for the IPs inside prefix 110.0.0.0/16, host 2 for 120.0.0.0/16, and host 3 for 130.0.0.0/16. Exceptionally, we do not use an IP address from the prefix, but it is for experimental purposes. 35 % of the routes have a measurement IP address in the range 110.0.0.0/16 which means that AS65002 has the best delay. Indeed, when the RQM pings for this address to measure the route from AS65002 passing through the blue VRF, it will have a 20 ms delay because it reaches host1 directly. For the route from AS65003 passing through the red VRF, it will have a 60 ms delay because it has to pass by host 3, then host 2, and finally host 1, and head back in the opposite direction. 35% of the routes have a measurement IP address in the range 130.0.0.0/16, which means that AS65003 has the best delay, and finally, 30% of the routes have a measurement IP address in the range 120.0.0.0/16, which means that the two routes have the same delay and then the AS-PATH has to be used to distinguish these routes. We use AS65004 as a customer of AS65001. It allows us to monitor every update that AS65001 sends and then which best route AS65001 has selected. We use the MRT, Multi-Threaded Routing Toolkit [54], format to record the BGP routing information. We also have used MRT in AS65001 to monitor when it receives messages from the providers. AS65002 starts advertising prefixes every 0.1 second twenty seconds after its configuration. AS65003 starts eighty seconds after its configuration. For the RQM configuration, we used an alpha of 0.005 for the t-tests after multiple experiments. This alpha allows the classification of most of

[5] `https://github.com/Exa-Networks/exabgp`

[6] `https://linux.die.net/man/8/tc`

the routes correctly. We use ten threads to manage subscriptions.

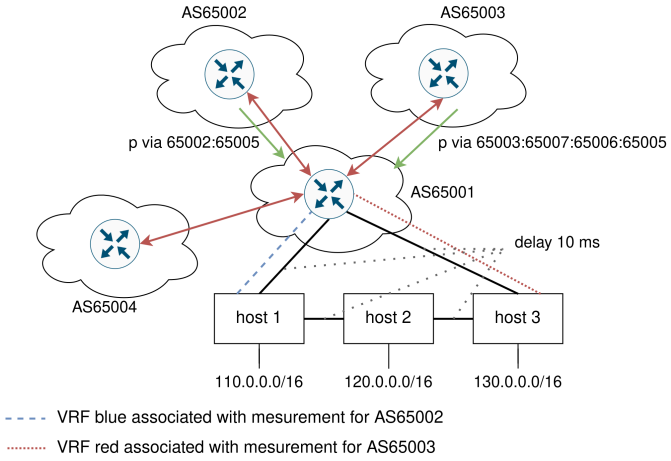
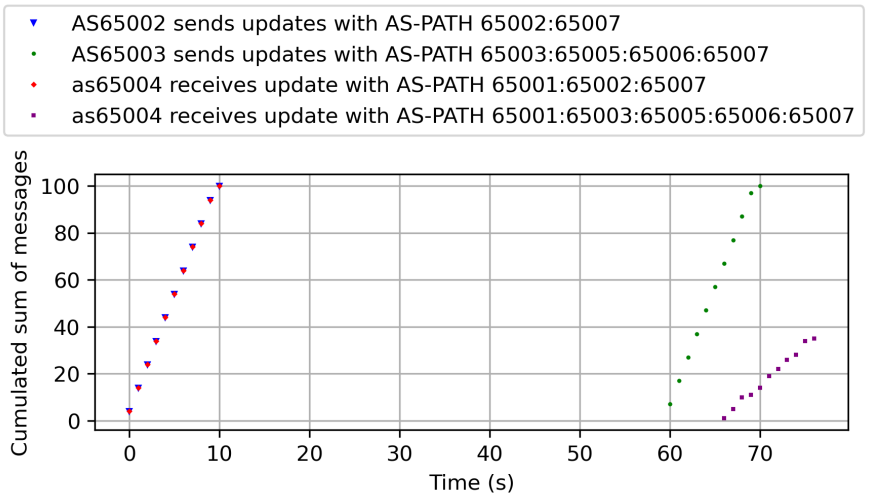
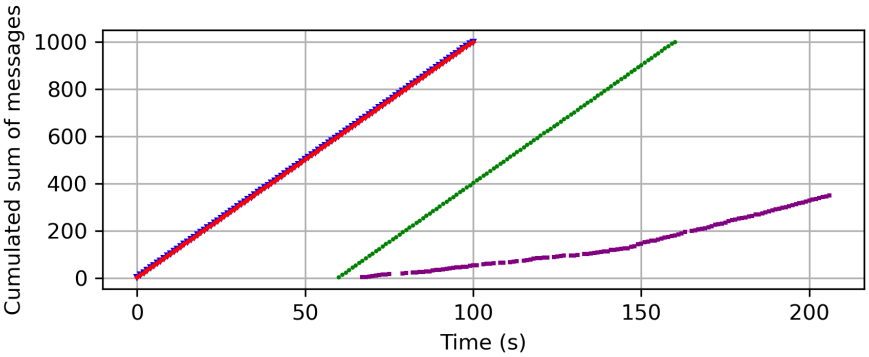


Figure 4.10: Topology for experiment 1

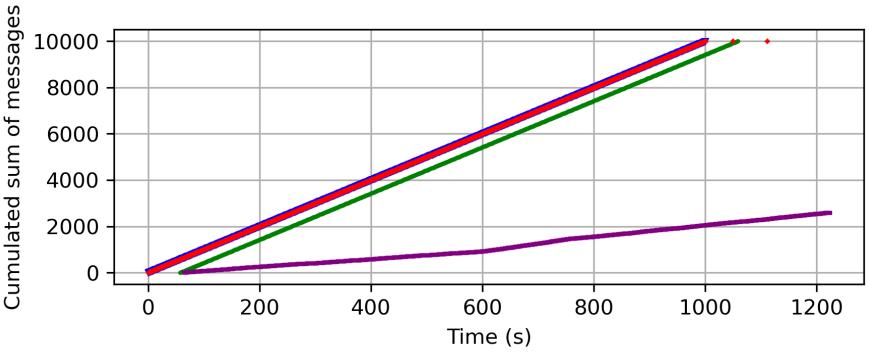
In the first instance, let us look at the timeline of BGP messages exchanged for 100, 1000, and 10000 prefixes. We have chosen a delay of 5 minutes between two measurements for the RQM. Figure 4.11 shows these timelines. We can observe that the different graphs have nearly the same appearance but with varying scales. When AS65002 sends the prefixes, very shortly after, AS65004 receives the updates associated with it. It results in the lines on the graph being nearly indistinguishable from each other. When AS65003 sends the prefixes, we can observe that AS65004 receives updates with the longer path. 35% of updates are sent with this path for 100 and 1000 prefixes. This corresponds to the 35% of routes having a better quality with a longer AS-PATH. However, only 26% of the routes with the longer path are sent to AS65004 with 10000 prefixes. We can make the hypothesis that the different hosts for pinging or the RQM might be overloaded. To verify this hypothesis, in further work, we can measure how many messages the RQM can process at a time, how many measurements it can make at the same time, and how many ping messages a host can hold. In this scenario, every five minutes, the RQM has to perform more and more pings until reaching 60000 pings per five minutes (two neighbors, three pings per measurement, 10000 prefixes). It results in 200 pings per second that the hosts and the RQM must handle at the end.



(a) 100 prefixes



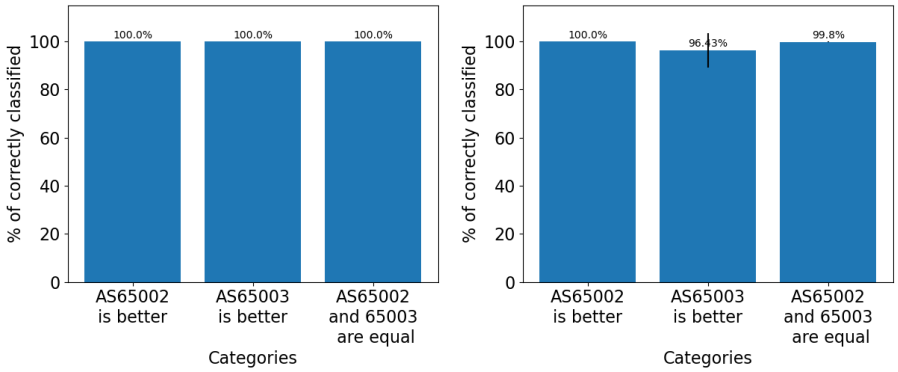
(b) 1000 prefixes



(c) 10000 prefixes

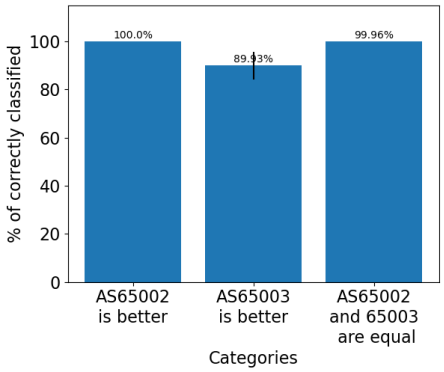
Figure 4.11: Timeline representing the cumulated sum of messages over the time

Now, let us look at the mistakes made by the RQM, i.e., when the quality of the prefix is not correct. We have defined three ranges of IP addresses used to ping, and each of them represented if AS65002 has a better quality, AS65003 has a better quality, or both have the same quality. Then, we can categorize the IP prefixes into three categories according to the IP address used for measurement: "AS65002 is better" with 35% of prefixes, "AS65003 is better" with 35% of prefixes, and "AS65002 and AS65003 are equal" with 30% of prefixes. For categories "AS65002 is better" and "AS65002 and AS65003 are equal", the prefix from one of these categories is well classified when AS65004 receives for prefixes only one update with the AS-PATH 65002:65007. A prefix from the category "AS65003 is better" is well-classified when AS65004 receives the route 65002:65007 and after the route with the longest AS-PATH via AS65003. Now, we are looking at the percentage of prefixes correctly classified. For that, we repeated five times the previous experiment and let it run for about 20 minutes. Figure 4.12 shows the mean percentage of correctly classified prefixes for 100, 1000, and 10000 prefixes. Appendix F summarizes for each iteration the number of correctly and badly classified prefixes. We can observe that with 100 prefixes, all of them are well-classified. In all cases, the prefixes in the category "AS65002 is better" are always well-classified. For prefixes from the category "AS65002 and AS65003 are equal", more than 99% are correctly classified for experiments with 1000 and 10000 prefixes, and the standard deviation is very low and practically invisible on the graph. However, for prefixes from the category "AS65003 is better" only at least in mean 89.93% of them are well-classified. We might explain this difference with the unbalanced load among the hosts used to ping. As we gradually add the prefixes, we progressively subscribe to the RQM, at the beginning for one peer and at the end for the two peers. AS65002 announces its prefixes before AS65003. When routes from AS65003 are progressively subscribed to the RQM, this last already reruns some measurement for prefixes that have only one peer to measure. So, when a new route is added for AS65003, the quality may be affected by the number of pings made that are not compared at this moment.



(a) 100 prefixes

(b) 1000 prefixes

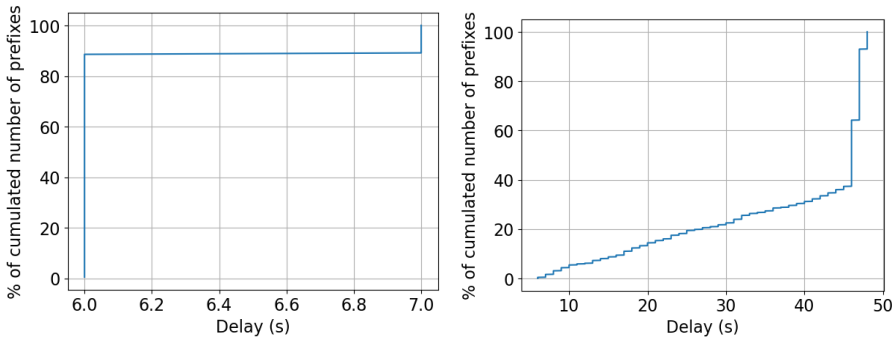


(c) 10000 prefixes

Figure 4.12: Pecenteage of correctly classified prefixes per categories with 100, 1000, and 10000 prefixes

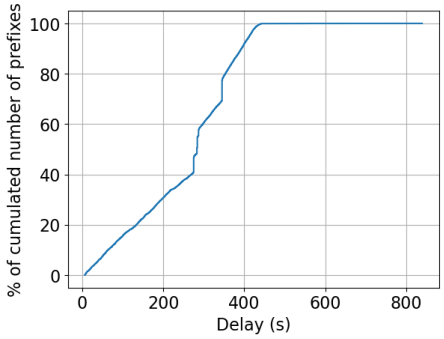
In a third time, let us see the delay between the announcement of the route where AS65003 is the best and when AS65004 receives the route. We take into account only prefixes that are correctly classified. Figure 4.13 shows the percentage of the cumulated number of prefixes according to the delay. For the 100 prefixes experiment, over the five repetitions, a total of 175 prefixes are taken into account. For the 1000 prefixes experiment, a total of 1483 prefixes are taken into account. And for the 10000 prefixes experiment, a total of 8691 prefixes are taken into account. For the 100 prefixes experiment, the delay is around 6-7 seconds. For the 1000 prefixes experiment, the curve linearly increases with a slight slope between 2 s to 45 s. We can see that 38% of prefixes have the best route that is announced with a delay under 45 s. Then the curve peaks sharply after 45 s to reach 48 s where 100% of prefixes have been advertised within this delay. So, the 62% remaining prefixes have this best route announced to AS65004 with a delay between 45 s and 48 s. For the 10000 prefixes experiment, the curve increases linearly until 275 s

(4min35), where its slope changes. Around 40% of the prefixes are announced to AS65004 with a delay under 4min35. This delay is under the time of the interval that the RQM uses to make the measurement. Then, the curve increases more sharply to reach a plateau of around 440 s. The delay for advertising the best route to AS65004 for 99.8% of the prefixes is under 440 s (7min20), so after one iteration of all measurements by the RQM. For the remaining 0.2% of prefixes, the best route by AS65003 is announced with a delay between 440 s (7min20) and 840 s (14 min). After a few iterations, the RQM has found that it is the best route. It seems that the RQM service is overloaded by the requests and can not handle all of them at the same time. The more routes are present to measure, the slower the response will be. In further work, we can send subscription messages at different rates and see when the RQM starts to slow down to process all these messages.



(a) 100 prefixes (175 prefixes taken into account)

(b) 1000 prefixes (1438 prefixes taken into account)



(c) 10000 prefixes (8691 prefixes taken into account)

Figure 4.13: Percentage of cumulated number of prefixes according the delay for 100, 1000, and 10000 prefixes

4.4.2 Second experiment

Our second experiment aims to show the time taken by the different parts of the RQM service. We focus on the time to perform the t-tests and the ranking.

First, we measured the time it took to perform a t-test. We have measured the time to make 500 t-tests for sample sizes from 1 to 50. We generated samples with a normal distribution and the same variance for each t-test. We have made this experiment with samples that have the same mean, a mean of 20, and with samples that have different means, one of 20 and one of 40. We repeated these experiments 10 times. Figure 4.14 shows the mean time taken for the 500 t-tests and its standard deviation according to the sample size. First, let us analyze the two curves together. We can observe that there is a rapid increase between a sample size of 1 and 2. This can be explained by the fact that a t-test requires at least two elements in the samples. The t-test function checks that if it is possible to do the t-test or not. After that, we see a slight increase to reach 56 ms or 66 ms for 500 t-tests for 50 elements in samples. We can expect this curve. In the t-test, we have to compute the mean and the standard deviation of each sample. This computation has a linear complexity. The remaining part of the t-test has a constant complexity. We can see that there is still a difference of 10 ms at the end between the curves with the same sample mean and with the mean of the various samples. When the mean is different, it seems that the time taken is lower.

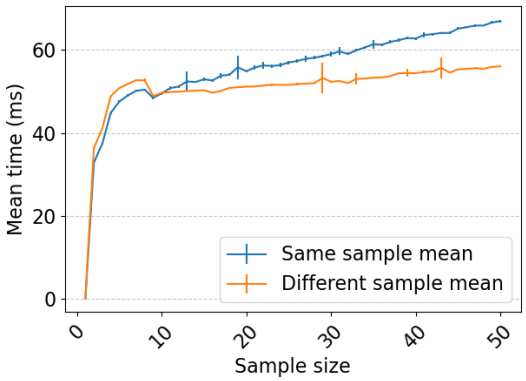


Figure 4.14: Mean time take to perform 500 t-tests according the sample size

In a second time, we measured the time to perform the ranking. We have measured the time to make 1000 ranking for 2, 4, 8, 16, 32, 64 and 128 neighbors. The sample for each neighbor has a size of 5, and the sample mean was 20 or 40 randomly selected. We perform ten times the measurements. Figure 4.15 shows the mean time taken and its standard deviation according to the number of neighbors. Firstly,

we can see that the standard deviation is very small and completely invisible on the diagram. Next, we can observe that the time increases exponentially with the number of neighbors to rank. For 128 neighbors, it takes around 22 s so, 22 ms per ranking. This corresponds to the fact that the ranking function has an exponential complexity. For the ranking, it makes two by two the t-test and then sorts all neighbors using the results of the t-test. Making the t-test two by two has an exponential complexity. In the further work section 5.2, we discuss a way of avoiding this behavior.

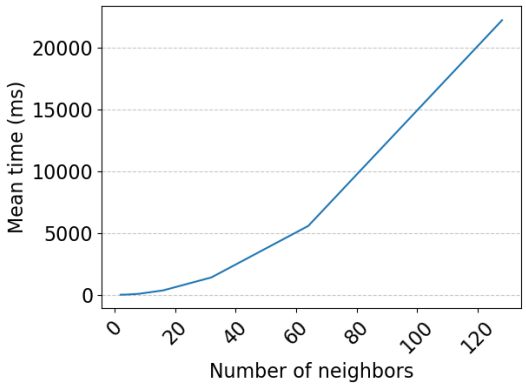


Figure 4.15: Mean time taken to perform 1000 ranking according the number of neighbors

4.5 Discussion

In this chapter, we have built a measurement system to monitor the quality of BGP routes. This quality is used in the BGP decision process as a tie-break rule to select the best route. We have seen that for a few routes, all qualities are correctly detected, and the delay to receive them is around 6-7 seconds. However, with more routes to measure, the quality is not correctly detected for some routes. In our experiment scenario, it was the quality of the route with the longer AS-PATH that was not well quantified and underestimated. This resulted in selecting the route with the shortest AS-PATH. In other scenarios, it could be the opposite. The route with the shortest AS-PATH and theoretically a better quality, if underestimated, is not selected as the best route. A route with a longer AS-PATH and theoretically a lower quality is preferred. We also have seen that the more routes we subscribed to the RQM, the longer it takes to receive quality. It might be due to two reasons. First, we made a burst of announcements with all routes to subscribe to the RQM. Second, our virtual environment, and more specifically for measurements, might

not handle the load. We have noticed that the RQM does not scale well if many neighbors subscribe to the same prefix. This is due to an exponential complexity in the ranking that could be solved in further work.

The RQM that we have built has currently some limitations, such as the capacity or the precision. However, our goal was to create a prototype that proves that it is possible to take into account the monitoring of the data plane in the BGP decision process. The results show that most routes with a longer AS-PATH but better quality are selected as the best route.

Chapter 5

Further work

In this chapter, we will show what future work could be done based on our solutions and experiments.

5.1 Improving the BGP shutdown solution

Currently, the BGP shutdown suffers from some limitations. The limitation part in Section 3.4 explain them.

Handling multiple alternative hidden routes with different export policies. In the case of multiple alternative routes with different export policies, it can result in sending withdraws to eBGP peers even if an alternative route that can be sent to them is present. To improve that, we can add a verification condition before sending a withdrawal to an eBGP peer. If the currently announced route to this eBGP peer has a local preference of 0, and the export policies reject the new best route, instead of sending a withdrawal to the peer, it waits for the expiration of the timer. In a such manner, only when the timer expires, a withdrawal message is sent to an eBGP peer if no alternative routes can be announced to it.

Countering the drop of traffic in the data plane. In case of a planned shutdown, we can imagine letting the link up after withdrawing the routes and then, ten minutes later, really shut down the link. We assume that all routers implement the solution that sets the local preference to 0 and wait for an alternative route until the timer expires. If a withdrawal is sent to the eBGP peer, this peer still uses the route until it has found an alternative route or has removed, in its turn, the route for this prefix. If there is an alternative route, packets for the prefix are always forwarded to it. However, if there is no route, packets are dropped. We can imagine, in this last case, keeping the route for the prefix in the routing table for

ten minutes after sending the withdrawal. Meanwhile, if the peer has traffic for the prefix, it will not be dropped. If every router implements the solution, normally, no traffic for this prefix is dropped during the ten minutes that the link is up. In further work, we could implement this with attributes to warn a peer if the route is still available for some time in the routing table.

5.2 Improving the Route Quality Measurement

Testing different metrics. In this master thesis, we focused on the delay. Many metrics, such as the number of lost packets [44] or the TCP throughput [45], need to be explored. Combining metrics can also be a research opportunity.

Reducing the algorithmic complexity of the ranking. Currently, we perform t-tests between each neighbor and then order them according to pairwise result. This has an exponential complexity. We can explore whether if we can assume the transitivity. For three samples A, B, and C, if the statistical test says A $<$ B and B $<$ C, then can we say A $<$ C ? If we can do that we do not need to make two-by-two comparisons, but only at the sorting stage. We can also use the memoization to keep the results of the t-test already done so we do not have to redo them.

Searching for different statistical hypothesis testing. With the two-sample t-test, we assumed that the samples are normally distributed, they have equal standard deviations, and they are independent [34, 35]. This might not be the case. We can make another statistical test to verify if we can assume equal standard deviations. Many other statistical tests are available, with different assumptions and hypotheses. In future work, we may search for the best applicable test for a certain type of measurement.

Securing the measurement. Currently, the measurements are made with ping. It is not a secure mean to measure the quality. An adversary may drop these packets, delay them, or spoof the answer to manipulate the decision. Sharon Goldberg et al. [55] proposes a way to monitor the quality in the presence of these adversaries. They have designed multiple path-quality monitoring (PQM) protocols. One of these protocols adapted to our situation is the Asymmetric Secure Sampling, where a single router (server), in our case, the router behind the IP address available to measure, manages multiple routers (clients), the different RQM services in our solution. This secure sampling provides an accurate round-trip delay measurement using cryptographic and statistical techniques.

5.3 Measurements

Measurements are made with Containerlab, which relies on Docker containers. Each BGP process does not have its own core and shares the twenty with the other processes. This might influence measurements. More measurements can be done with other tools such as Dμne [1], which is Distributed Micro Network Emulation where processes are pinned to CPU cores. Other measurements can be practiced by peering one or multiple routers with our solutions with an experimental network such as DN42 [2] or PEERING Testbed [3].

[1] `https://github.com/nrybowski/DUNE`
[2] `https://dn42.eu/Home`
[3] `https://peering.ee.columbia.edu/`

Chapter 6

Conclusion

The goal of this master thesis was to make the Border Gateway Protocol data plane aware. To do that, we proceeded in two steps. The first one was to see if we can modify the behavior of BGP when it learns an element about the data plane. The second one was to perform measurements directly in the data plane and use the results in the BGP decision process.

For the first step, we used the scenario case of a planned administrative shutdown. Our sub-goal was to reduce the number of BGP withdraw between ASes when a route that was previously advertised that a shutdown will occurs and an alternative route exists. A BGP route that will no longer be available due to a planned shutdown is advertised with a special community a little while before it is withdrawn. When a router receives a BGP withdraw for a route previously advertised with the special community, if it was the only route that it knew, instead of sending a BGP withdraw to all its eBGP peers, it waits for some time to see if the AS that it is in knows an alternative route. We have added our solution to goBGP. We have seen that the current solution suffers from some limitations, such as the drop of packets in the data plane, in case of multiple alternative routes with different export policies, the solution may not work, or causes a higher delay to advertise the alternative route to eBGP peer. Nonetheless, we succeeded in changing the behavior of BGP. In the case where a BGP speaker receives a BGP withdraw from a route that it does know an alternative one, but at least one is in the AS and the first announced alternative can be advertised to an eBGP peer, no withdraw messages are sent to this peer. This proved to us that it is possible to modify the BGP behavior.

Then, we moved on to the second step, making active measurements in the data plane and using the results in BGP. We have created the RQM, Route Quality Measurement, to perform measures in the data plane and communicate them to BGP. We have defined a new attribute, the quality, which represents a ranking

place between peers that have announced the same prefix. We have modified the BGP decision process to take that quality into account. We have created a protocol between the RQM and BGP to communicate. We have imagined the RQM and the protocol for communicating with BGP in a way that can be easily adapted in future work. We modified the goBGP implementation to add our solution. We also have created the RQM dispositif in Go. Through our experiments, we saw that, with our solution, routes with a better quality in terms of delay were chosen as the best route over routes with a shorter AS-PATH. Our prototype showed signs of limitations and overload when we increased the number of measurements to perform. More experiments to find the reasons could be conducted in further work, as well as experiments with other metrics than the delay.

We can conclude by the fact that we succeded in taking into account information from the data plane in the control plane of BGP. We have obtained promising results that show an interest in pursuing research to add data plane knowledge to BGP and extend this principle to other routing protocols.

Appendix

A Border Gateway Protocol Parameters

This appendix lists the different BGP messages, BGP path attributes, BGP OPEN Optional Parameter Types and Capabilities registered at the IANA on the date the 5th of August 2024. The IANA [24], Internet Assigned Numbers Authority, coordinates internet resouces. It ensures unique ID used in some protocols like BGP.

A.1 BGP Message Types

The following list takes all referenced BGP message types at the IANA [1] on the 5th of August 2024.

	Name	Reference
0	Reserved	
1	OPEN	[RFC4271]
2	UPDATE	[RFC4271]
3	NOTIFICATION	[RFC4271]
4	KEEPALIVE	[RFC4271]
5	ROUTE-REFRESH	[RFC2918]
6-255	Unassigned	

Table A.1: List of BGP message types on the 5th of August 2024

[1] https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-parameters-1

A.2 BGP Path Attributes

The following list takes all referenced BGP Path Attributes at the IANA [2] on the 5th of August 2024.

	Code	Reference
0	Reserved	
1	ORIGIN	[RFC4271]
2	AS_PATH	[RFC4271]
3	NEXT_HOP	[RFC4271]
4	MULTI_EXIT_DISC	[RFC4271]
5	LOCAL_PREF	[RFC4271]
6	ATOMIC_AGGREGATE	[RFC4271]
7	AGGREGATOR	[RFC4271]
8	COMMUNITIES	[RFC1997]
9	ORIGINATOR_ID	[RFC4456]
10	CLUSTER_LIST	[RFC4456]
11	DPA (deprecated)	[Chen, E., Bates, T., "Destination Preference Attribute for BGP", Work in progress, March 1996.][RFC6938]
12	ADVERTISER (historic) (deprecated)	[RFC1863][RFC4223][RFC6938]
13	RCID_PATH / CLUSTER_ID (Historic) (deprecated)	[RFC1863][RFC4223][RFC6938]
14	MP_REACH_NLRI	[RFC4760]
15	MP_UNREACH_NLRI	[RFC4760]
16	EXTENDED COMMUNITIES	[Eric_Rosen][draft-ramachandra-bgp-ext-communities-00][RFC4360]
17	AS4_PATH	[RFC6793]
18	AS4_AGGREGATOR	[RFC6793]
19	SAFI Specific Attribute (SSA) (deprecated)	[Gargi_Nalawade][draft-kapoor-nalawade-idr-bgp-ssa-00][draft-nalawade-idr-mdt-safi-00][draft-wijnands-mt-discovery-00]
20	Connector Attribute (deprecated)	[RFC6037]
21	AS_PATHLIMIT (deprecated)	[draft-ietf-idr-as-pathlimit]

[2] `https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-parameters-2`

22	PMSI_TUNNEL	[RFC6514]
23	Tunnel Encapsulation	[RFC9012]
24	Traffic Engineering	[RFC5543]
25	IPv6 Address Specific Extended Community	[RFC5701]
26	AIGP	[RFC7311]
27	PE Distinguisher Labels	[RFC6514]
28	BGP Entropy Label Capability Attribute (deprecated)	[RFC6790][RFC7447]
29	BGP-LS Attribute	[RFC9552]
30	Deprecated	[RFC8093]
31	Deprecated	[RFC8093]
32	LARGE_COMMUNITY	[RFC8092]
33	BGPsec_Path	[RFC8205]
34	BGP Community Container Attribute (TEMPORARY - registered 2017-07-28, extension registered 2023-07-13, expires 2024-07-28)	[draft-ietf-idr-wide-bgp-communities-11]
35	Only to Customer (OTC)	[RFC9234]
36	BGP Domain Path (D-PATH) (TEMPORARY - registered 2019-07-08, extension registered 2024-05-30, expires 2025-07-08)	[draft-ietf-bess-evpn-ipvpn-interworking-10]
37	SFP attribute	[RFC9015]
38	BFD Discriminator	[RFC9026]
39	BGP Next Hop Dependent Capabilities (NHC) (TEMPORARY - registered 2022-12-20, extension registered 2023-11-22, expires 2024-12-20)	[draft-ietf-idr-entropy-label-13]
40	BGP Prefix-SID	[RFC8669]
41-127	Unassigned	
128	ATTR_SET	[RFC6368]
129	Deprecated	[RFC8093]
130-240	Unassigned	
241	Deprecated	[RFC8093]
242	Deprecated	[RFC8093]
243	Deprecated	[RFC8093]
244-254	Unassigned	
255	Reserved for development	[RFC2042]

Table A.2: List of BGP Path attributes on the 5th of August 2024

A.3 BGP OPEN Optional Parameter Types

The following list takes all referenced Optional parameters in a BGP OPEN message at the IANA [3] on the 5th of August 2024.

	Name	Reference
0	Reserved	[RFC5492]
1	Authentication (deprecated)	[RFC4271][RFC5492]
2	Capabilities	[RFC5492]
3-254	Unassigned	
255	Extended Length	[RFC9072]

Table A.3: List of BGP OPEN Optional Parameter on the 5th of August 2024

A.4 Capability Codes

The following list takes all referenced capabilities at the IANA [4] on the 5th of August 2024.

	Description	Reference
0	Reserved	[RFC5492]
1	Multiprotocol Extensions for BGP-4	[RFC2858]
2	Route Refresh Capability for BGP-4	[RFC2918]
3	Outbound Route Filtering Capability	[RFC5291]
4	Multiple routes to a destination capability (deprecated)	[RFC8277]
5	Extended Next Hop Encoding	[RFC8950]
6	BGP Extended Message	[RFC8654]
7	BGPsec Capability	[RFC8205]
8	Multiple Labels Capability	[RFC8277]
9	BGP Role	[RFC9234]
10-63	Unassigned	
64	Graceful Restart Capability	[RFC4724]
65	Support for 4-octet AS number capability	[RFC6793]
66	Deprecated (2003-03-06)	

[3] https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-parameters-11

[4] https://www.iana.org/assignments/capability-codes/capability-codes.xhtml

67	Support for Dynamic Capability (capability specific)	[draft-ietf-idr-dynamic-cap]
68	Multisession BGP Capability	[draft-ietf-idr-bgp-multisession]
69	ADD-PATH Capability	[RFC7911]
70	Enhanced Route Refresh Capability	[RFC7313]
71	Long-Lived Graceful Restart (LLGR) Capability	[RFC-ietf-idr-long-lived-gr-06]
72	Routing Policy Distribution	[draft-ietf-idr-rpd-04]
73	FQDN Capability	[draft-walton-bgp-hostname-capability]
74	BFD Strict-Mode Capability	[draft-ietf-idr-bgp-bfd-strict-mode-13]
75	Software Version Capability	[draft-abraitis-bgp-version-capability-11]
76	PATHS-LIMIT Capability	[draft-abraitis-idr-addpath-paths-limit-00]
77-127	Unassigned	
128	Prestandard Route Refresh (deprecated)	[RFC8810]
129	Prestandard Outbound Route Filtering (deprecated), prestandard Routing Policy Distribution (deprecated)	[RFC8810]
130	Prestandard Outbound Route Filtering (deprecated)	[RFC8810]
131	Prestandard Multisession (deprecated)	[RFC8810]
132-183	Unassigned	
184	Prestandard FQDN (deprecated)	[RFC8810]
185	Prestandard OPERATIONAL message (deprecated)	[RFC8810]
186-238	Unassigned	
239-254	Reserved for Experimental Use	[RFC8810]
255	Reserved	[RFC8810]

Table A.4: List of BGP capability codes on the 5th of August 2024

B An example of ASes peering relation with policies

This appendix aims to provide an example of how policies can be used to enforce relationships. We set different policies according to the peering relation. Figure B.1 shows an example of peering relationships with import and export policies. To know if a route is learned from a customer, a provider, or a peer, we can use local-preference and communities. We apply them via import policies. For example, if a route is learned from a customer, we set the community 65000:10 and a local-preference of 150 for the route. If it is learned from a provider, we set the community 65000:20 and a local-preference of 50. And if it is learned from a peer-to-peer relation, we set the community 65000:30 and a local-preference of 100. Communities allow us to know if we can advertise a route to a provider, a customer, or a peer. The export policies check if we can advertise the route by checking the presence of communities. If the route is sent to a provider or a peer, it checks if the route does not have the community 65000:20 or 65000:30. If the route contains one of these communities, it does not advertise the route. If the route is sent to a customer, as we are a provider, we do not need to check the communities. These communities have a signification only inside the AS, so we have to remove them with the export policies before advertising. Local-preferences allow the BGP decision process to prefer a route learned by a customer, if any, by a peer, and if any, by a provider. As we set a higher local-preference for customer routes compared to the local-preference peer and provider routes, customer routes are always selected. The same reasoning holds between the peers and the providers.

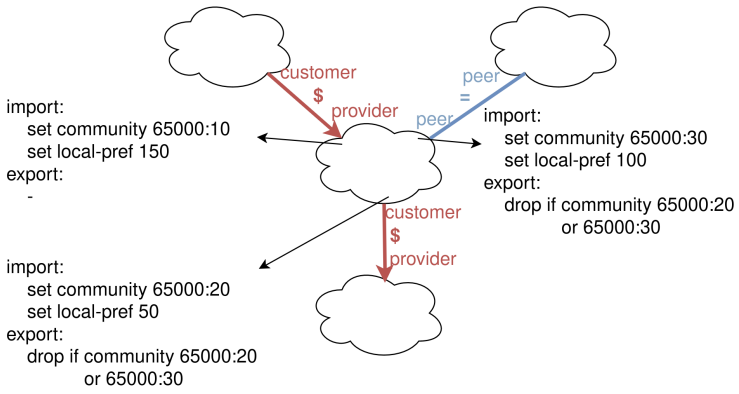


Figure B.1: An example of peering relationships

C RouteViews and RIPE RIS datasets

This appendix provides the list of collectors coming from RouteViews [38] and RIPE RIS [39] used in Section 3.3 and the number of samples used from each collector. We have defined a sample as the set of BGP messages sent to a collector from one peer during the 10 minutes after RRC03 has withdrawn the prefix.

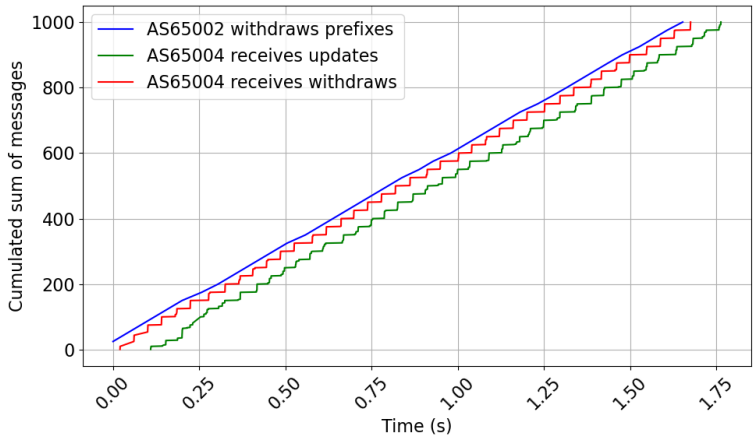
Collector	Source	Location	Number of samples used
RRC00	RIPE RIS	Amsterdam, Netherlands	3728
RRC01	RIPE RIS	London, United Kingdom	3138
RRC03	RIPE RIS	Amsterdam, Netherlands	3978
RRC04	RIPE RIS	Geneva, Switzerland	672
RRC05	RIPE RIS	Vienna, Austria	924
RRC06	RIPE RIS	Otemachi, Japan	228
RRC07	RIPE RIS	Stockholm, Sweden	839
RRC10	RIPE RIS	Milan, Italy	994
RRC11	RIPE RIS	New York, NY, US	264
RRC12	RIPE RIS	Frankfurt, Germany	3700
RRC13	RIPE RIS	Moscow, Russia	504
RRC14	RIPE RIS	Palo Alto, California, US	84
RRC15	RIPE RIS	Sao Paulo, Brazil	1666
RRC16	RIPE RIS	Miami, Florida, US	2
RRC18	RIPE RIS	Barcelona, Spain	84
RRC19	RIPE RIS	Johannesburg, South Africa	434
RRC20	RIPE RIS	Zurich, Switzerland	1987
RRC21	RIPE RIS	Paris, France	2049
RRC22	RIPE RIS	Bucharest, Romania	202
RRC23	RIPE RIS	Singapore, Singapore	884
RRC24	RIPE RIS	Montevideo, Uruguay	672
RRC25	RIPE RIS	Amsterdam, Netherlands	3586
RRC26	RIPE RIS	Dubai, United Arab Emirates	328
route-views.amsix	RouteViews	Amsterdam, Netherlands	2304
route-views.bdix	RouteViews	Dhaka, Bangledesh	0
route-views.bknix	RouteViews	Bangkok, Thailand	0
route-views.chicago	RouteViews	Chicago, Illinois, US	486
route-views.chile	RouteViews	Santiago, Chile	0
route-views.eqix	RouteViews	Ashburn, Virgina, US	740
route-views.flix	RouteViews	Miami, Florida, US	16

route-views.fortaleza	RouteViews	Fortaleza, Brazil	430
route-views.gixa	RouteViews	Ghana, Africa	0
route-views.gorex	RouteViews	Guam, US Territories	0
route-views.isc	RouteViews	Palo Alto, California, US	84
route-views.kixp	RouteViews	Nairobi, Kenya	201
route-views.linx	RouteViews	London, United Kingdom	3087
route-views.mwix	RouteViews	Indianapolis, Indiana, US	0
route-views.napafrica	RouteViews	Johannesburg, South Africa	759
route-views.nwax	RouteViews	Portland, Oregon, US	112
route-views.ny	RouteViews	New York, NY, US	84
route-views.perth	RouteViews	Perth, Australia	1163
route-views.peru	RouteViews	Lima, Peru	0
route-views.phoix	RouteViews	Diliman, Quezon City, Philippines	24
route-views.rio	RouteViews	Rio de Janeiro, Brazil	473
route-views.sfmix	RouteViews	San Francisco, California	65
route-views.sg	RouteViews	Singapore, Singapore	1594
route-views.soxrs	RouteViews	Belgrade, Serbia	252
route-views.sydney	RouteViews	Sydney, Australia	1110
route-views.telxatl	RouteViews	Atlanta, Georgia	0
route-views.uaeix	RouteViews	Dubai, United Arab Emirates	326
route-views.wide	RouteViews	Tokyo, Japan	84
route-views2	RouteViews	Eugene Oregon, US	1604
route-views3	RouteViews	Eugene Oregon, US	1428
route-views4	RouteViews	Eugene Oregon, US	1243
route-views5	RouteViews	Eugene Oregon, US	1225
route-views6	RouteViews	Eugene Oregon, US	0
route-views2.saopaulo	RouteViews	Sao Paulo, Brazil	1572

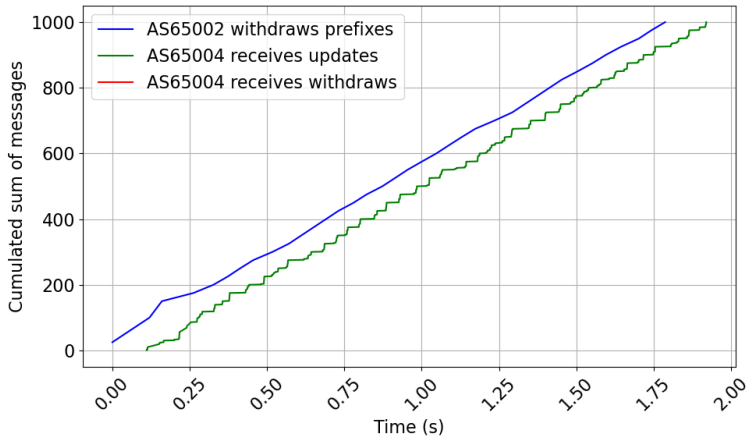
Table C.1: List of collector used for the analysis of BGP withdraws in the real world

D All results of the BGP planned shutdown experiment

In this appendix, we provides all timeline for each iteration of the experiment.

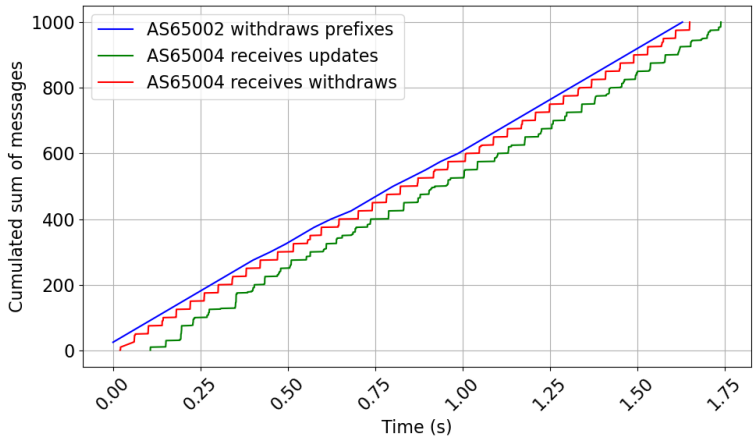


(a) Without our solution

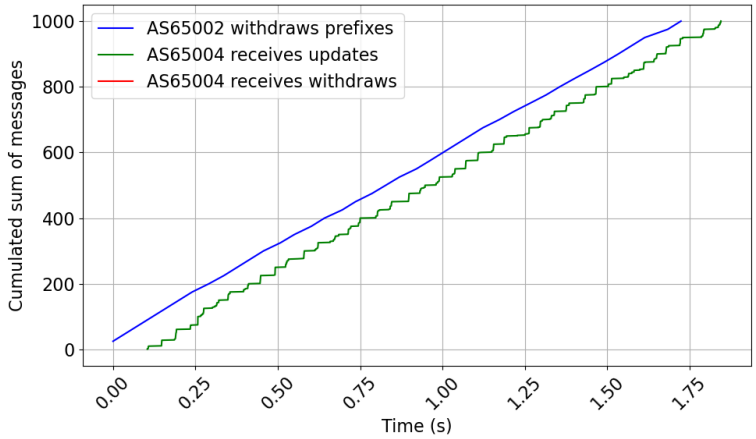


(b) With our solution

Figure D.1: Timeline representing the cumulated sum of messages over the time for iteration 1

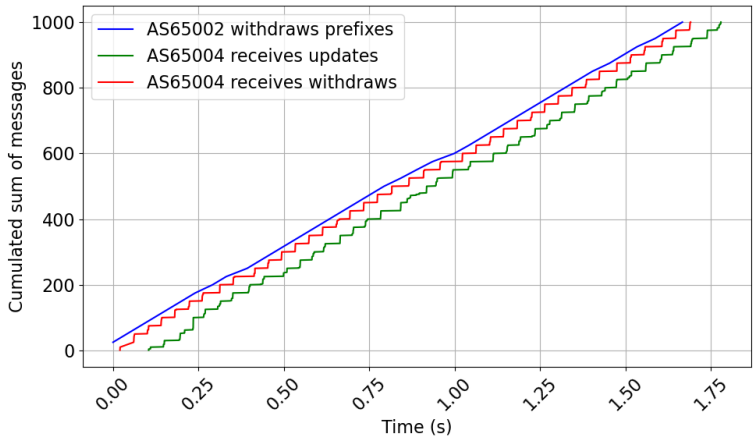


(a) Without our solution

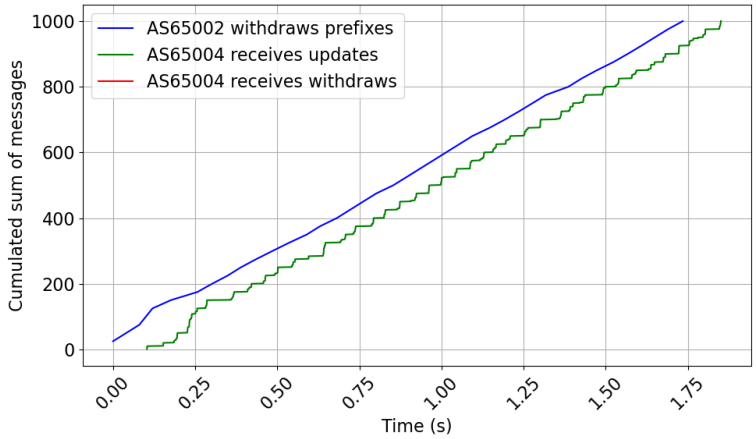


(b) With our solution

Figure D.2: Timeline representing the cumulated sum of messages over the time for iteration 2

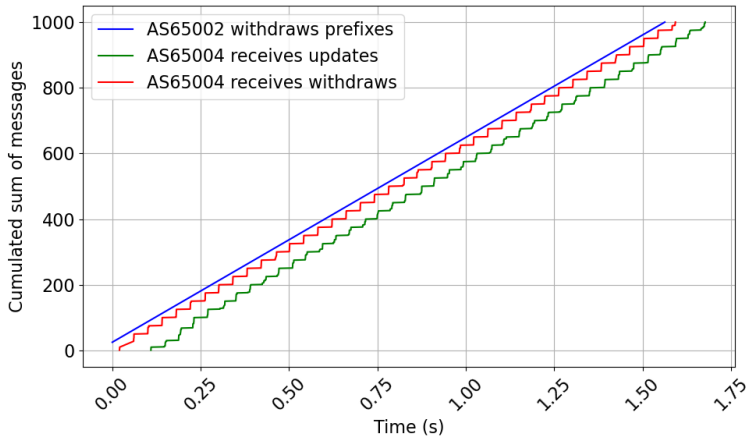


(a) Without our solution

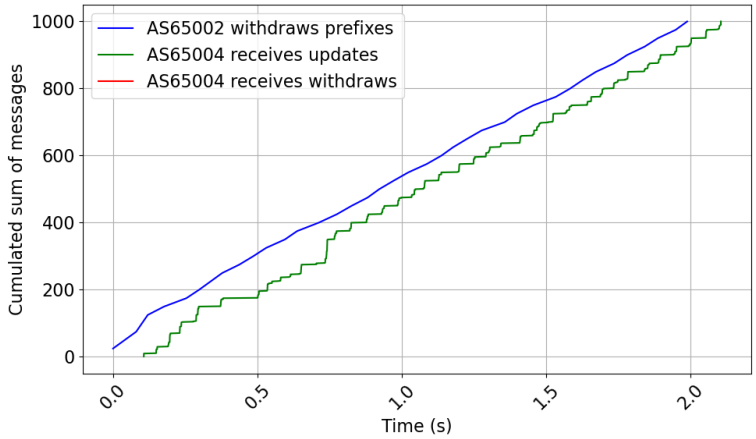


(b) With our solution

Figure D.3: Timeline representing the cumulated sum of messages over the time for iteration 3

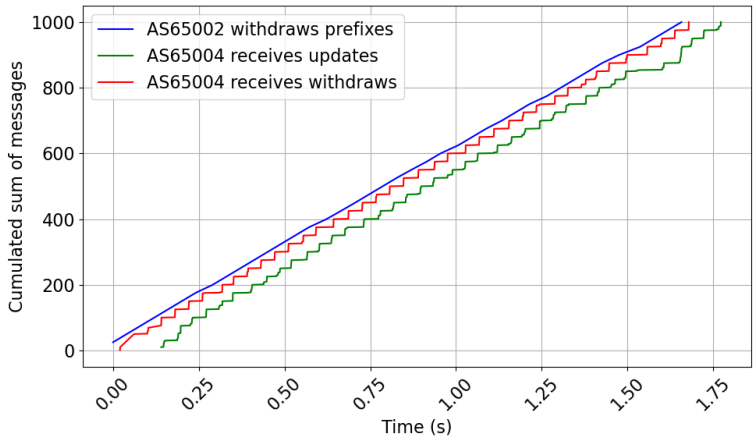


(a) Without our solution

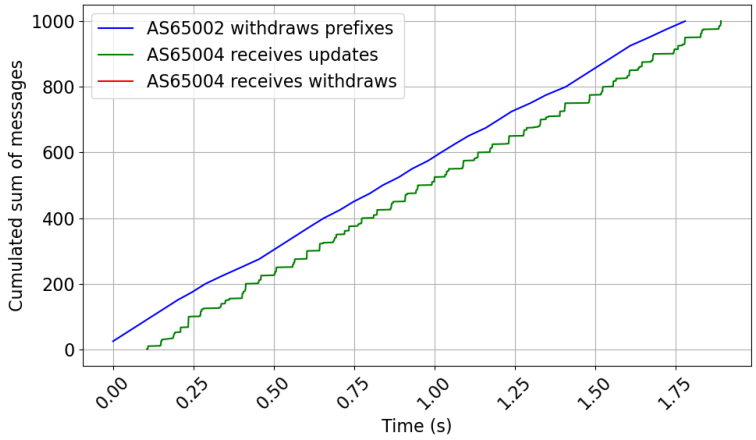


(b) With our solution

Figure D.4: Timeline representing the cumulated sum of messages over the time for iteration 4



(a) Without our solution



(b) With our solution

Figure D.5: Timeline representing the cumulated sum of messages over the time for iteration 5

E Additional information about the t-tests in the real world

This appendix shows the SQL query E.1 used in Google Big Query to retrieve data used, and more detail on the number of clusters and t-test measurements available for each two hours. We provide information about the cluster in Table E.1 with the mean, standard deviation, and median rejection percentage per cluster. In Table E.2, we provide information about the number of t-tests made per range of two hours.

```
WITH Probes as (
SELECT prb_id, asn_v4, asn_v6, ST_CLUSTERDBSCAN(geography, 5e3, 2)
    OVER () AS cluster_num
FROM `ripencc-atlas.probes.probes`
WHERE geography IS NOT NULL
AND status = 'Connected'
)
SELECT HTTP.prb_id, dst_addr, http_duration, start_time, asn_v4,
    asn_v6, cluster_num
FROM `ripencc-atlas.measurements.http` AS HTTP
JOIN Probes AS P ON HTTP.prb_id = P.prb_id
WHERE
    start_time BETWEEN '2024-06-03 00:00:00' AND '2024-06-04 00:00:00'
    AND cluster_num IS NOT NULL
    AND http_status = 200
```

Figure E.1: SQL query

This query retrieves HTTP durations of the successful HTTP measurements made on the 3th of June 2024, and are grouped by clusters of 5km.

Hour range	Nbr cluster	Mean rejection percentage per cluster	Std rejection percentage per cluster	Median rejection percentage per cluster
00-02	1118	0.592	0.347	0.667
02-04	1117	0.596	0.349	0.667
04-06	1119	0.585	0.351	0.667
06-08	1117	0.581	0.349	0.66
08-10	1105	0.582	0.348	0.643
10-12	1102	0.585	0.344	0.646
12-14	1100	0.559	0.35	0.616
14-16	1103	0.548	0.35	0.6
16-18	1101	0.559	0.354	0.622
18-20	1102	0.557	0.35	0.625
20-22	1104	0.553	0.351	0.611
22-24	1106	0.572	0.353	0.638

Table E.1: Mean, standard deviation and median of the percentage rejection per cluster

Hour range	Nbr t-test	Mean t-test per cluster	Std t-test per cluster	Median t-test per cluster
00-02	2492414	2229.351	17444.881	6
02-04	2464111	2206.008	17257.972	6
04-06	2464292	2202.227	17244.515	6
06-08	2465288	2207.062	17271.197	6
08-10	2511432	2272.789	17817.153	6
10-12	2495584	2264.595	17636.401	6
12-14	2485335	2259.395	17668.912	6
14-16	2461846	2231.955	17709.036	6
16-18	2493021	2264.324	17816.799	6
18-20	2500246	2268.826	17832.496	6
20-22	2495527	2260.441	17803.676	6
22-24	2478925	2241.343	17783.483	6

Table E.2: Mean, standard deviation and median of t-test made per two hours

F Detail results of the classification of qualities experiment

This appendix shows the number of badly classified route for each iteration of the experiment of the RQM with 100, 1000, and 10000 prefixes.

Iteration	"AS65002 is better" /35	"AS65003 is better" / 35	"AS65002 and AS65003 are equal" /30
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0

Table F.1: Number of badly classified qualities per categories for each iteration with 100 prefixes

Iteration	"AS65002 is better" /350	"AS65003 is better" / 350	"AS65002 and AS65003 are equal" /300
1	0	0	5
2	0	312	1
3	0	0	4
4	0	0	2
5	0	0	3

Table F.2: Number of badly classified qualities per categories for each iteration with 1000 prefixes

Iteration	"AS65002 is better" /3500	"AS65003 is better" / 3500	"AS65002 and AS65003 are equal" /3000
1	0	919	5
2	0	2339	9
3	0	1046	7
4	0	3500	0
5	0	1005	8

Table F.3: Number of badly classified qualities per categories for each iteration with 10000 prefixes

Bibliography

[1] K. Lougheed and J. Rekhter, "Border Gateway Protocol (BGP)," RFC 1105, Jun. 1989. [Online]. Available: https://www.rfc-editor.org/info/rfc1105

[2] Y. Rekhter, S. Hares, and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4271

[3] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol," RFC 6810, Jan. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6810

[4] T. Alfroy, T. Holterbach, T. Krenc, K. Claffy, and C. Pelsser, "Internet science moonshot: Expanding bgp data horizons," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, ser. HotNets '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 102–108. [Online]. Available: https://doi.org/10.1145/3626111.3628202

[5] V. Van den Schrieck, P. Francois, C. Pelsser, and O. Bonaventure, "Preventing the unnecessary propagation of bgp withdraws," in *NETWORKING 2009: 8th International IFIP-TC 6 Networking Conference, Aachen, Germany, May 11-15, 2009. Proceedings 8*. Springer, 2009, pp. 495–508.

[6] V. Van den Schrieck, P. Francois, and O. Bonaventure, "Bgp add-paths: the scaling/performance tradeoffs," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1299–1307, 2010.

[7] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig, "Interdomain traffic engineering with bgp," *IEEE Communications magazine*, vol. 41, no. 5, pp. 122–128, 2003.

[8] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 5, pp. 19–30, 2003.

[9] T. Wirtgen and O. Bonaventure, "A first step towards checking bgp routes in the dataplane," in *Proceedings of the ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, 2022, pp. 50–57.

[10] J. A. Hawkinson and T. J. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)," RFC 1930, Mar. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1930

[11] E. Chen, "Route Refresh Capability for BGP-4," RFC 2918, Sep. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2918

[12] R. Chandra and J. Scudder, "Capabilities Advertisement with BGP-4," RFC 5492, Feb. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5492

[13] T. Li, R. Chandra, and P. S. Traina, "BGP Communities Attribute," RFC 1997, Aug. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1997

[14] E. Chen, T. J. Bates, and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)," RFC 4456, Apr. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4456

[15] Y. Rekhter, H. H. Ould-Brahim, and D. Fedyk, "BGP Traffic Engineering Attribute," RFC 5543, May 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5543

[16] D. Tappan, S. R. Sangli, and Y. Rekhter, "BGP Extended Communities Attribute," RFC 4360, Feb. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4360

[17] Y. Rekhter, "IPv6 Address Specific BGP Extended Community Attribute," RFC 5701, Nov. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5701

[18] K. Kumaki, K. Patel, R. Raszuk, P. Marques, and T. Yamagata, "Internal BGP as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)," RFC 6368, Sep. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6368

[19] R. Chandra, T. J. Bates, Y. Rekhter, and D. Katz, "Multiprotocol Extensions for BGP-4," RFC 4760, Jan. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4760

[20] D. Walton, A. Retana, E. Chen, and J. Scudder, "Advertisement of Multiple Paths in BGP," RFC 7911, Jul. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7911

[21] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Transactions on networking*, vol. 9, no. 6, pp. 681–692, 2001.

[22] A. Azimov, E. Bogomazov, R. Bush, K. Patel, and K. Sriram, "Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages," RFC 9234, May 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9234

[23] S. Weiler and D. Blacka, "Clarifications and Implementation Notes for DNS Security (DNSSEC)," RFC 6840, Feb. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6840

[24] Internet Assigned Numbers Authority. (2024) Internet assigned numbers authority. Last access: 2024-08-4. [Online]. Available: https://www.iana.org/

[25] M. Lepinski, A. Chi, and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)," RFC 6488, Feb. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6488

[26] J. Snijders, B. Maddison, M. Lepinski, D. Kong, and S. Kent, "A Profile for Route Origin Authorizations (ROAs)," RFC 9582, May 2024. [Online]. Available: https://www.rfc-editor.org/info/rfc9582

[27] G. Huston, R. Loomans, and G. G. Michaelson, "A Profile for Resource Certificate Repository Structure," RFC 6481, Feb. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6481

[28] R. Bush, "Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)," RFC 7115, Jan. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7115

[29] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1," RFC 8210, Sep. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8210

[30] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein, "BGP Prefix Origin Validation," RFC 6811, Jan. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6811

[31] G. Huston and G. G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)," RFC 6483, Feb. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6483

[32] Michael Kerrisk (man-pages maintainer), "ip-vrf(8) - linux manual page," 2024, accessed: 2024-08-16. [Online]. Available: https://man7.org/linux/man-pages/man8/ip-vrf.8.html

[33] Linux Kernel Documentation, "Virtual routing and forwarding (vrf)," 2024, accessed: 2024-08-16. [Online]. Available: https://docs.kernel.org/networking/vrf.html

[34] NIST/SEMATECH. e-handbook of statistical methods.

[35] J. L. Hainaut Donatien, "Statistiques et science des données," 2024-08-16. [Online]. Available: https://sites.uclouvain.be/archives-portail/cdc2021/en-cours-2021-lepl1109

[36] O. Reference, *Oxford Reference Online*. Oxford University Press, 2011, last access: 2024-08-16. [Online]. Available: https://www.oxfordreference.com/display/10.1093/oi/authority.20110803100355922

[37] A. Garcia-Martinez and M. Bagnulo, "Measuring bgp route propagation times," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2432–2436, 2019.

[38] (2024) University of oregon route views project. Last access: 2024-05-10. [Online]. Available: https://www.routeviews.org/

[39] RIPE NCC. (2024) RIPE RIS. Last access: 2024-08-12. [Online]. Available: https://www.ripe.net/

[40] Osrg. (2024) gobgp. [Online]. Available: https://github.com/osrg/gobgp

[41] P. Marques, R. Fernando, E. Chen, P. Mohapatra, and H. Gredler, "Advertisement of the best external route in BGP," Internet Engineering Task Force, Internet-Draft draft-ietf-idr-best-external-05, Jan. 2012, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-idr-best-external/05/

[42] P. Marcos, L. Prehn, L. Leal, A. Dainotti, A. Feldmann, and M. Barcellos, "As-path prepending: there is no rose without a thorn," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 506–520.

[43] S. Kalidindi, M. J. Zekauskas, and D. G. T. Almes, "A Round-trip Delay Metric for IPPM," RFC 2681, Sep. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2681

[44] G. Almes, S. Kalidindi, M. J. Zekauskas, and A. Morton, "A One-Way Loss Metric for IP Performance Metrics (IPPM)," RFC 7680, Jan. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7680

[45] R. Schrage, G. Forget, R. Geib, and B. Constantine, "Framework for TCP Throughput Testing," RFC 6349, Aug. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6349

[46] M. Apostolaki, A. Singla, and L. Vanbever, "Performance-driven internet path selection," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, 2021, pp. 41–53.

[47] R. Nakamura, K. Shimizu, T. Kamata, and C. Pelsser, "A first measurement with bgp egress peer engineering," in *International Conference on Passive and Active Network Measurement*. Springer, 2022, pp. 199–215.

[48] C. Filsfils, S. Previdi, G. Dawra, E. Aries, and D. Afanasiev, "Segment Routing Centralized BGP Egress Peer Engineering," RFC 9087, Aug. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9087

[49] Autorité de Régulation des Communications Électroniques et des Postes. (2023) Le numérique au 4 juillet 2023. Last access: 2024-08-6. [Online]. Available: https://www.arcep.fr/actualites/actualites-et-communiques/detail/n/numerique-040723.html

[50] Cloudflare. (2024) gortr. [Online]. Available: https://github.com/cloudflare/gortr

[51] M. Piraux and O. Bonaventure, "Adaptive address family selection for latency-sensitive applications on dual-stack hosts," *arXiv preprint arXiv:2309.05369*, 2023.

[52] RIPE NCC. (2024) RIPE Atlas. Last access: 2024-08-12. [Online]. Available: https://atlas.ripe.net/

[53] Google. (2024) Google bigquery. Last access: 2024-08-12. [Online]. Available: https://cloud.google.com/bigquery

[54] C. Petrie and T. King, "Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format with BGP Additional Path Extensions," RFC 8050, May 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8050

[55] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-quality monitoring in the presence of adversaries," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 1, pp. 193–204, 2008.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**